Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Microsoft SQL Server data source

Grafana ships with built-in support for Microsoft SQL Server (MS SQL). You can query and visualize data from any Microsoft SQL Server 2005 or newer, including Microsoft Azure SQL Database.

This topic explains configuration specific to the Microsoft SQL Server data source.

For instructions on how to add a data source to Grafana, refer to the administration documentation. Only users with the organization administrator role can add data sources. Administrators can also configure the data source via YAML with Grafana's provisioning system.

Once you've added the Microsoft SQL Server data source, you can configure it so that your Grafana instance's users can create queries in its query editor when they build dashboards and use Explore.

Configure the data source

To configure basic settings for the data source, complete the following steps:

1. Click **Connections** in the left-side menu.
2. Under Your connections, click **Data sources**.
3. Enter `Microsoft SQL Server` in the search bar.
4. Select **Microsoft SQL Server**.

   The **Settings** tab of the data source is displayed.
5. Set the data source's basic configuration options:

Name	Description
Name	Sets the name you use to refer to the data source in panels and queries.

Name	Description
Default	Sets the data source that's pre-selected for new panels.
Host	Sets the IP address/hostname and optional port of your MS SQL instance. Default port is 0, the driver default. You can specify multiple connection properties, such as `ApplicationIntent`, by separating each property with a semicolon (`;`).
Database	Sets the name of your MS SQL database.
Authentication	Sets the authentication mode, either using SQL Server Authentication or Windows Authentication (single sign-on for Windows users).
User	Defines the database user's username.
Password	Defines the database user's password.
Encrypt	Determines whether or to which extent a secure SSL TCP/IP connection will be negotiated with the server. Options include: `disable` - data sent between client and server is not encrypted; `false` - data sent between client and server is not encrypted beyond the login packet; `true` - data sent between client and server is encrypted. Default is `false`.
Max open	Sets the maximum number of open connections to the database. Default is `100`.
Max idle	Sets the maximum number of connections in the idle connection pool. Default is `100`.
Auto (max idle)	If set will set the maximum number of idle connections to the number of maximum open connections (Grafana v9.5.1+). Default is `true`.
Max lifetime	Sets the maximum number of seconds that the data source can reuse a connection. Default is `14400` (4 hours).

You can also configure settings specific to the Microsoft SQL Server data source. These options are described in the sections below.

Min time interval

The **Min time interval** setting defines a lower limit for the `$__interval` and [`$__interval_ms`][add-template-variables-interval_ms] variables.

This value *must* be formatted as a number followed by a valid time identifier:

Identifier	Description
`y`	year
`M`	month
`w`	week
`d`	day
`h`	hour
`m`	minute

Identifier	Description
`s`	second
`ms`	millisecond

We recommend setting this value to match your Microsoft SQL Server write frequency. For example, use `1m` if Microsoft SQL Server writes data every minute.

You can also override this setting in a dashboard panel under its data source options.

Connection timeout

The **Connection timeout** setting defines the maximum number of seconds to wait for a connection to the database before timing out. Default is 0 for no timeout.

Database user permissions

Grafana doesn't validate that a query is safe, and could include any SQL statement. For example, Microsoft SQL Server would execute destructive queries like `DELETE FROM user;` and `DROP TABLE user;` if the querying user has permission to do so.

To protect against this, we strongly recommend that you create a specific MS SQL user with restricted permissions.

Grant only `SELECT` permissions on the specified database and tables that you want to query to the database user you specified when you added the data source:

sql Copy

```sql
CREATE USER grafanareader WITH PASSWORD 'password'
GRANT SELECT ON dbo.YourTable3 TO grafanareader
```

Also, ensure that the user doesn't have any unwanted privileges from the public role.

Diagnose connection issues

If you use older versions of Microsoft SQL Server, such as 2008 and 2008R2, you might need to disable encryption before you can connect the data source.

We recommend that you use the latest available service pack for optimal compatibility.

Provision the data source

You can define and configure the data source in YAML files as part of Grafana's provisioning system. For more information about provisioning, and for available configuration options, refer to Provisioning Grafana.

Provisioning example

yaml Copy

```yaml
apiVersion: 1

datasources:
  - name: MSSQL
    type: mssql
    url: localhost:1433
    user: grafana
    jsonData:
      database: grafana
      maxOpenConns: 100 # Grafana v5.4+
      maxIdleConns: 100 # Grafana v5.4+
      maxIdleConnsAuto: true # Grafana v9.5.1+
      connMaxLifetime: 14400 # Grafana v5.4+
      connectionTimeout: 0 # Grafana v9.3+
      encrypt: 'false'
    secureJsonData:
      password: 'Password!'
```

Query the data source

You can create queries with the Microsoft SQL Server data source's query editor when editing a panel that uses a MS SQL data source.

For details, refer to the query editor documentation.

Use template variables

Instead of hard-coding details such as server, application, and sensor names in metric queries, you can use variables. Grafana lists these variables in dropdown select boxes at the top of the dashboard to help you change the data displayed in your dashboard. Grafana refers to such variables as template variables.

For details, see the template variables documentation.

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Microsoft SQL Server query editor

You can create queries with the Microsoft SQL Server data source's query editor when editing a panel that uses a MS SQL data source.

This topic explains querying specific to the MS SQL data source. For general documentation on querying data sources in Grafana, see Query and transform data.

Choose a query editing mode

You can switch the query editor between two modes:

- Code mode, which provides a feature-rich editor for writing queries
- Builder mode, which provides a visual query designer

To switch between the editor modes, select the corresponding **Builder** and **Code** tabs above the editor.

To run a query, select **Run query** located at the top right corner of the editor.

The query editor also provides:

- Macros
- Annotations
- Stored procedures

Configure common options

You can configure a MS SQL-specific response format in the query editor regardless of its mode.

Choose a response format

Grafana can format the response from MS SQL as either a table or as a time series.

To choose a response format, select either the **Table** or **Time series** formats from the **Format** dropdown.

To use the time series format, you must name one of the MS SQL columns `time`. You can use time series queries, but not table queries, in alerting conditions.

For details about using these formats, refer to Use table queries and Use time series queries.

Code mode

In **Code mode**, you can write complex queries using a text editor with autocompletion features and syntax highlighting.

For more information about Transact-SQL (T-SQL), the query language used by Microsoft SQL Server, refer to the Transact-SQL tutorial.

Use toolbar features

Code mode has several features in a toolbar located in the editor's lower-right corner.

To reformat the query, click the brackets button (`{}`).

To expand the code editor, click the chevron button pointing downward.

To run the query, click the **Run query** button or use the keyboard shortcut Ctrl/Cmd + Enter/Return.

Use autocompletion

Code mode's autocompletion feature works automatically while typing. To manually trigger autocompletion, use the keyboard shortcut Ctrl/Cmd + Space.

Code mode supports autocompletion of tables, columns, SQL keywords, standard SQL functions, Grafana template variables, and Grafana macros.

Note: You can't autocomplete columns until you've specified a table.

Builder mode

In **Builder mode**, you can build queries using a visual interface.

Dataset and table selection

In the **Dataset** dropdown, select the MSSQL database to query. Grafana populates the dropdown with all databases that the user can access. Once you select a database, Grafana populates the dropdown with all available tables.

Note: If a default database has been configured through the Data Source Configuration page (or through a provisioning configuration file), the user will only be able to use that single preconfigured database for querying.

We don't include `tempdb`, `model`, `msdb`, `master` databases in the query editor dropdown.

Select columns and aggregation functions (SELECT)

Select a column from the **Column** dropdown to include it in the data. You can select an optional aggregation function for the column in the **Aggregation** dropdown.

To add more value columns, click the plus (`+`) button to the right of the column's row.

Filter data (WHERE)

To add a filter, toggle the **Filter** switch at the top of the editor. This reveals a **Filter by column value** section with two dropdown selectors.

Use the first dropdown to choose whether all of the filters need to match (`AND`), or if only one of the filters needs to match (`OR`). Use the second dropdown to choose a filter.

To filter on more columns, click the plus (`+`) button to the right of the condition dropdown.

To remove a filter, click the `x` button next to that filter's dropdown.

After selecting a date type column, you can choose Macros from the operators list and select timeFilter which will add the $__timeFilter macro to the query with the selected date column.

Group results

To group results by column, toggle the **Group** switch at the top of the editor. This reveals a **Group by column** dropdown where you can select which column to group the results by.

To remove the group-by clause, click the `x` button.

Preview the query

To preview the SQL query generated by Builder mode, toggle the **Preview** switch at the top of the editor. This reveals a preview pane containing the query, and an copy icon at the top right that copies the query to your clipboard.

Use macros

To simplify syntax and to allow for dynamic components, such as date range filters, you can add macros to your query.

Macro example	Replaced by
`$__time(dateColumn)`	An expression to rename the column to *time*. For example, *dateColumn as time*
`$__timeEpoch(dateColumn)`	An expression to convert a DATETIME column type to Unix timestamp and rename it to *time*. For example, *DATEDIFF(second, '1970-01-01', dateColumn) AS time*
`$__timeFilter(dateColumn)`	A time range filter using the specified column name. For example, *dateColumn BETWEEN '2017-04-21T05:01:17Z' AND '2017-04-21T05:06:17Z'*
`$__timeFrom()`	The start of the currently active time selection. For example, *'2017-04-21T05:01:17Z'*
`$__timeTo()`	The end of the currently active time selection. For example, *'2017-04-21T05:06:17Z'*
`$__timeGroup(dateColumn,'5m'[, fillvalue])`	An expression usable in GROUP BY clause. Providing a *fillValue* of *NULL* or *floating value* will automatically fill empty series in timerange with that value. For example, *CAST(ROUND(DATEDIFF(second, '1970-01-01', time_column)/300.0, 0) as bigint)*300*.
`$__timeGroup(dateColumn,'5m', 0)`	Same as above but with a fill parameter so missing points in that series will be added by grafana and 0 will be used as value.
`$__timeGroup(dateColumn,'5m', NULL)`	Same as above but NULL will be used as value for missing points.

Macro example	Replaced by
`$__timeGroup(dateColumn,'5m', previous)`	Same as above but the previous value in that series will be used as fill value if no value has been seen yet NULL will be used (only available in Grafana 5.3+).
`$__timeGroupAlias(dateColumn,'5m')`	Same as `$__timeGroup` but with an added column alias (only available in Grafana 5.3+).
`$__unixEpochFilter(dateColumn)`	A time range filter using the specified column name with times represented as Unix timestamp. For example, *dateColumn > 1494410783 AND dateColumn < 1494497183*
`$__unixEpochFrom()`	The start of the currently active time selection as Unix timestamp. For example, *1494410783*
`$__unixEpochTo()`	The end of the currently active time selection as Unix timestamp. For example, *1494497183*
`$__unixEpochNanoFilter(dateColumn)`	A time range filter using the specified column name with times represented as nanosecond timestamp. For example, *dateColumn > 1494410783152415214 AND dateColumn < 1494497183142514872*
`$__unixEpochNanoFrom()`	The start of the currently active time selection as nanosecond timestamp. For example, *1494410783152415214*
`$__unixEpochNanoTo()`	The end of the currently active time selection as nanosecond timestamp. For example, *1494497183142514872*
`$__unixEpochGroup(dateColumn,'5m', [fillmode])`	Same as `$__timeGroup` but for times stored as Unix timestamp (only available in Grafana 5.3+).
`$__unixEpochGroupAlias(dateColumn,'5m', [fillmode])`	Same as above but also adds a column alias (only available in Grafana 5.3+).

To suggest more macros, please open an issue in our GitHub repo.

View the interpolated query

The query editor also includes a link named **Generated SQL** that appears after running a query while in panel edit mode. To display the raw interpolated SQL string that the data source executed, click on this link.

Use table queries

If the **Format** query option is set to **Table** for a Table panel, you can enter any type of SQL query. The Table panel then displays the query results with whatever columns and rows are returned.

Example database table:

sql Copy

```sql
CREATE TABLE [event] (
  time_sec bigint,
```

```
  description nvarchar(100),
  tags nvarchar(100),
)
```

sql Copy

```sql
CREATE TABLE [mssql_types] (
  c_bit bit, c_tinyint tinyint, c_smallint smallint, c_int int, c_bigint bigint, c_money money, c_
  c_real real, c_decimal decimal(10,2), c_float float,
  c_char char(10), c_varchar varchar(10), c_text text,
  c_nchar nchar(12), c_nvarchar nvarchar(12), c_ntext ntext,
  c_datetime datetime,  c_datetime2 datetime2, c_smalldatetime smalldatetime, c_date date, c_time
)

INSERT INTO [mssql_types]
SELECT
  1, 5, 20020, 980300, 1420070400, '$20000.15', '£2.15', 12345.12,
  1.11, 2.22, 3.33,
  'char10', 'varchar10', 'text',
  N'☺nchar12☺', N'☺nvarchar12☺', N'☺text☺',
  GETDATE(), CAST(GETDATE() AS DATETIME2), CAST(GETDATE() AS SMALLDATETIME), CAST(GETDATE() AS DAT
```

Query editor with example query:

The query:

sql Copy

```sql
SELECT * FROM [mssql_types]
```

To control the name of the Table panel columns, use the standard `AS` SQL column selection syntax.

For example:

```sql
SELECT
  c_bit as [column1], c_tinyint as [column2]
FROM
  [mssql_types]
```

The resulting table panel:

Use time series queries

If you set the **Format** setting in the query editor to **Time series**, then the query must have a column named `time` that returns either a SQL datetime or any numeric datatype representing Unix epoch in seconds. Result sets of time series queries must also be sorted by time for panels to properly visualize the result.

A time series query result is returned in a wide data frame format. Any column except time or of type string transforms into value fields in the data frame query result. Any string column transforms into field labels in the data frame query result.

Create a metric query

For backward compatibility, there's an exception to the above rule for queries that return three columns and include a string column named `metric`. Instead of transforming the `metric` column into field labels, it becomes the field name, and then the series name is formatted as the value of the `metric` column. See the example with the `metric` column below.

To optionally customize the default series name formatting, refer to Standard options definitions.

Example with `metric` column:

```sql
SELECT
  $__timeGroupAlias(time_date_time, '5m'),
  min("value_double"),
  'min' as metric
FROM test_data
WHERE $__timeFilter(time_date_time)
GROUP BY time
ORDER BY 1
```

Data frame result:

text Copy

```
+---------------------+-----------------+
| Name: time          | Name: min       |
| Labels:             | Labels:         |
| Type: []time.Time   | Type: []float64 |
+---------------------+-----------------+
| 2020-01-02 03:05:00 | 3               |
| 2020-01-02 03:10:00 | 6               |
+---------------------+-----------------+
```

Time series query examples

Using the fill parameter in the $__timeGroupAlias macro to convert null values to be zero instead:

sql Copy

```sql
SELECT
  $__timeGroupAlias(createdAt, '5m', 0),
  sum(value) as value,
  hostname
FROM test_data
WHERE
  $__timeFilter(createdAt)
GROUP BY
  time,
  hostname
ORDER BY 1
```

Given the data frame result in the following example and using the graph panel, you will get two series named *value 10.0.1.1* and *value 10.0.1.2*. To render the series with a name of *10.0.1.1* and *10.0.1.2* , use a Standard options definitions display name value of `${__field.labels.hostname}`.

Data frame result:

text Copy

```
+---------------------+--------------------------+--------------------------+
| Name: time          | Name: value              | Name: value              |
| Labels:             | Labels: hostname=10.0.1.1 | Labels: hostname=10.0.1.2 |
| Type: []time.Time   | Type: []float64          | Type: []float64          |
+---------------------+--------------------------+--------------------------+
| 2020-01-02 03:05:00 | 3                        | 4                        |
```

```
| 2020-01-02 03:10:00 | 6                          | 7                          |
+---------------------+----------------------------+----------------------------+
```

Using multiple columns:

sql Copy

```sql
SELECT
  $__timeGroupAlias(time_date_time, '5m'),
  min(value_double) as min_value,
  max(value_double) as max_value
FROM test_data
WHERE $__timeFilter(time_date_time)
GROUP BY time
ORDER BY 1
```

Data frame result:

text Copy

```
+---------------------+-----------------+-----------------+
| Name: time          | Name: min_value | Name: max_value |
| Labels:             | Labels:         | Labels:         |
| Type: []time.Time   | Type: []float64 | Type: []float64 |
+---------------------+-----------------+-----------------+
| 2020-01-02 03:04:00 | 3               | 4               |
| 2020-01-02 03:05:00 | 6               | 7               |
+---------------------+-----------------+-----------------+
```

Apply annotations

Annotations overlay rich event information on top of graphs. You can add annotation queries in the Dashboard menu's Annotations view.

Columns:

Name	Description
`time`	The name of the date/time field. Could be a column with a native SQL date/time data type or epoch value.
`timeend`	Optional name of the end date/time field. Could be a column with a native SQL date/time data type or epoch value. (Grafana v6.6+)

Name	Description
`text`	Event description field.
`tags`	Optional field name to use for event tags as a comma separated string.

Example database tables:

```sql
CREATE TABLE [events] (
  time_sec bigint,
  description nvarchar(100),
  tags nvarchar(100),
)
```

We also use the database table defined in Time series queries.

Example query using time column with epoch values:

```sql
SELECT
  time_sec as time,
  description as [text],
  tags
FROM
  [events]
WHERE
  $__unixEpochFilter(time_sec)
ORDER BY 1
```

Example region query using time and timeend columns with epoch values:

Only available in Grafana v6.6+.

```sql
SELECT
  time_sec as time,
  time_end_sec as timeend,
  description as [text],
  tags
FROM
```

```
  [events]
WHERE
  $__unixEpochFilter(time_sec)
ORDER BY 1
```

Example query using time column of native SQL date/time data type:

sql Copy

```sql
SELECT
  time,
  measurement as text,
  convert(varchar, valueOne) + ',' + convert(varchar, valueTwo) as tags
FROM
  metric_values
WHERE
  $__timeFilter(time_column)
ORDER BY 1
```

Use stored procedures

Stored procedures have been verified to work. However, please note that we haven't done anything special to support this, so there might be edge cases where it won't work as you would expect. Stored procedures should be supported in table, time series and annotation queries as long as you use the same naming of columns and return data in the same format as describe above under respective section.

Please note that any macro function will not work inside a stored procedure.

Examples

For the following examples, the database table is defined in Time series queries. Let's say that we want to visualize four series in a graph panel, such as all combinations of columns `valueOne`, `valueTwo` and `measurement`. Graph panel to the right visualizes what we want to achieve. To solve this, we need to use two queries:

First query:

```sql
SELECT
  $__timeGroup(time, '5m') as time,
  measurement + ' - value one' as metric,
  avg(valueOne) as valueOne
FROM
  metric_values
WHERE
  $__timeFilter(time)
GROUP BY
  $__timeGroup(time, '5m'),
  measurement
ORDER BY 1
```

Second query:

```sql
SELECT
  $__timeGroup(time, '5m') as time,
  measurement + ' - value two' as metric,
  avg(valueTwo) as valueTwo
```

```
FROM
  metric_values
GROUP BY
  $__timeGroup(time, '5m'),
  measurement
ORDER BY 1
```

Stored procedure using time in epoch format

We can define a stored procedure that will return all data we need to render 4 series in a graph panel like above. In this case the stored procedure accepts two parameters `@from` and `@to` of `int` data types which should be a timerange (from-to) in epoch format which will be used to filter the data to return from the stored procedure.

We're mimicking the `$__timeGroup(time, '5m')` in the select and group by expressions, and that's why there are a lot of lengthy expressions needed - these could be extracted to MS SQL functions, if wanted.

sql Copy

```
CREATE PROCEDURE sp_test_epoch(
  @from int,
  @to   int
)   AS
BEGIN
  SELECT
    cast(cast(DATEDIFF(second, {d '1970-01-01'}, DATEADD(second, DATEDIFF(second,GETDATE(),GETUTCD
    measurement + ' - value one' as metric,
    avg(valueOne) as value
  FROM
    metric_values
  WHERE
    time >= DATEADD(s, @from, '1970-01-01') AND time <= DATEADD(s, @to, '1970-01-01')
  GROUP BY
    cast(cast(DATEDIFF(second, {d '1970-01-01'}, DATEADD(second, DATEDIFF(second,GETDATE(),GETUTCD
    measurement
  UNION ALL
  SELECT
    cast(cast(DATEDIFF(second, {d '1970-01-01'}, DATEADD(second, DATEDIFF(second,GETDATE(),GETUTCD
```

Expand code

Then we can use the following query for our graph panel.

sql Copy

```
DECLARE
  @from int = $__unixEpochFrom(),
  @to int = $__unixEpochTo()

EXEC dbo.sp_test_epoch @from, @to
```

Stored procedure using time in datetime format

We can define a stored procedure that will return all data we need to render 4 series in a graph panel like above. In this case the stored procedure accepts two parameters `@from` and `@to` of `datetime` data types which should be a timerange (from-to) which will be used to filter the data to return from the stored procedure.

We're mimicking the `$__timeGroup(time, '5m')` in the select and group by expressions and that's why there's a lot of lengthy expressions needed - these could be extracted to MS SQL functions, if wanted.

sql Copy

```
CREATE PROCEDURE sp_test_datetime(
  @from datetime,
  @to   datetime
)   AS
BEGIN
  SELECT
    cast(cast(DATEDIFF(second, {d '1970-01-01'}, time)/600 as int)*600 as int) as time,
    measurement + ' - value one' as metric,
    avg(valueOne) as value
  FROM
    metric_values
  WHERE
    time >= @from AND time <= @to
  GROUP BY
    cast(cast(DATEDIFF(second, {d '1970-01-01'}, time)/600 as int)*600 as int),
    measurement
  UNION ALL
  SELECT
    cast(cast(DATEDIFF(second, {d '1970-01-01'}, time)/600 as int)*600 as int) as time,
```

Expand code

Then we can use the following query for our graph panel.

sql Copy

```
DECLARE
  @from datetime = $__timeFrom(),
```

```
  @to datetime = $__timeTo()

EXEC dbo.sp_test_datetime @from, @to
```

Microsoft SQL Server template variables

Instead of hard-coding details such as server, application, and sensor names in metric queries, you can use variables. Grafana lists these variables in dropdown select boxes at the top of the dashboard to help you change the data displayed in your dashboard. Grafana refers to such variables as template variables.

For an introduction to templating and template variables, refer to the Templating and Add and manage variables documentation.

Query variable

If you add a template variable of the type `Query`, you can write a MS SQL query that can return things like measurement names, key names or key values that are shown as a dropdown select box.

For example, you can have a variable that contains all values for the `hostname` column in a table if you specify a query like this in the templating variable **Query** setting.

```sql
SELECT hostname FROM host
```

A query can return multiple columns and Grafana will automatically create a list from them. For example, the query below will return a list with values from `hostname` and `hostname2`.

```sql
SELECT [host].[hostname], [other_host].[hostname2] FROM host JOIN other_host ON [host].[city] = [c
```

Another option is a query that can create a key/value variable. The query should return two columns that are named `__text` and `__value`. The `__text` column value should be unique (if it is not unique then the first value is used). The options in the dropdown will have a text and value that

allow you to have a friendly name as text and an id as the value. An example query with `hostname` as the text and `id` as the value:

```sql
SELECT hostname __text, id __value FROM host
```

You can also create nested variables. For example, if you had another variable named `region`. Then you could have the hosts variable only show hosts from the current selected region with a query like this (if `region` is a multi-value variable, then use the `IN` comparison operator rather than `=` to match against multiple values):

```sql
SELECT hostname FROM host WHERE region IN ($region)
```

Using variables in queries

> From Grafana 4.3.0 to 4.6.0, template variables are always quoted automatically so if it is a string value do not wrap them in quotes in where clauses.
>
> From Grafana 5.0.0, template variable values are only quoted when the template variable is a `multi-value`.

If the variable is a multi-value variable then use the `IN` comparison operator rather than `=` to match against multiple values.

There are two syntaxes:

`$<varname>` Example with a template variable named `hostname`:

```sql
SELECT
  atimestamp time,
  aint value
FROM table
WHERE $__timeFilter(atimestamp) and hostname in($hostname)
ORDER BY atimestamp
```

`[[varname]]` Example with a template variable named `hostname`:

```sql
```

```
SELECT
  atimestamp as time,
  aint as value
FROM table
WHERE $__timeFilter(atimestamp) and hostname in([[hostname]])
ORDER BY atimestamp
```

Disabling Quoting for Multi-value Variables

Grafana automatically creates a quoted, comma-separated string for multi-value variables. For example: if `server01` and `server02` are selected then it will be formatted as: `'server01', 'server02'`. To disable quoting, use the csv formatting option for variables:

`${servers:csv}`

Read more about variable formatting options in the Variables documentation.

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

MySQL data source

Starting from Grafana v5.1 you can name the time column *time* in addition to earlier supported *time_sec*. Usage of *time_sec* will eventually be deprecated.

Grafana ships with a built-in MySQL data source plugin that allows you to query and visualize data from a MySQL compatible database like MariaDB or Percona Server.

For instructions on how to add a data source to Grafana, refer to the administration documentation. Only users with the organization administrator role can add data sources. Administrators can also configure the data source via YAML with Grafana's provisioning system.

Give it a try using Grafana Play

With Grafana Play, you can explore and see how it works, learning from practical examples to accelerate your development. This feature can be seen on MySQL: Cities of the World Sample Data Set.

Configure the data source

To access the data source configuration page:

1. Click **Connections** in the left-side menu.
2. Under Your connections, click **Data sources**.
3. Enter `MySQL` in the search bar.
4. Select **MySQL**.

   The **Settings** tab of the data source is displayed.

5 Set the data source's basic configuration options.

Name	Description
Name	The data source name. This is how you refer to the data source in panels and queries.
Default	Default data source means that it will be pre-selected for new panels.
Host	The IP address/hostname and optional port of your MySQL instance.
Database	Name of your MySQL database.
User	Database user's login/username
Password	Database user's password
Session Timezone	Specifies the timezone used in the database session, such as `Europe/Berlin` or `+02:00`. Required if the timezone of the database (or the host of the database) is set to something other than UTC. Set this to `+00:00` so Grafana can handle times properly. Set the value used in the session with `SET time_zone='...'`. If you leave this field empty, the timezone will not be updated. For more information, refer to MySQL Server Time Zone Support.
Max open	The maximum number of open connections to the database, default `100` (Grafana v5.4+).
Max idle	The maximum number of connections in the idle connection pool, default `100` (Grafana v5.4+).
Auto (max idle)	Toggle to set the maximum number of idle connections to the number of maximum open connections (available in Grafana v9.5.1+). Default is `true`.
Allow cleartext passwords	Allows the use of the cleartext client side plugin as required by a specific type of account, such as one defined with the PAM authentication plugin. **Sending passwords in clear text may be a security problem in some configurations.** To avoid password issues, it is recommended that clients connect to a MySQL server using a method that protects the password. Possibilities include TLS / SSL, IPsec, or a private network. Default is `false`.
Max lifetime	The maximum amount of time in seconds a connection may be reused. This should always be lower than configured wait_timeout in MySQL (Grafana v5.4+). The default is `14400` or 4 hours.

Min time interval

The **Min time interval** setting defines a lower limit for the `$__interval` and `$__interval_ms` variables.

This value *must* be formatted as a number followed by a valid time identifier:

Identifier	Description
`y`	year

Identifier	Description
`M`	month
`w`	week
`d`	day
`h`	hour
`m`	minute
`s`	second
`ms`	millisecond

We recommend setting this value to match your MySQL write frequency. For example, use `1m` if MySQL writes data every minute.

You can also override this setting in a dashboard panel under its data source options.

Database User Permissions (Important!)

The database user you specify when you add the data source should only be granted SELECT permissions on the specified database and tables you want to query. Grafana does not validate that the query is safe. The query could include any SQL statement. For example, statements like `USE otherdb;` and `DROP TABLE user;` would be executed. To protect against this we **Highly** recommend you create a specific mysql user with restricted permissions.

Example:

sql Copy

```sql
CREATE USER 'grafanaReader' IDENTIFIED BY 'password';
GRANT SELECT ON mydatabase.mytable TO 'grafanaReader';
```

You can use wildcards (`*`) in place of database or table if you want to grant access to more databases and tables.

Provision the data source

You can define and configure the data source in YAML files as part of Grafana's provisioning system. For more information about provisioning, and for available configuration options, refer to Provisioning Grafana.

Provisioning examples

Basic Provisioning

yaml Copy

```yaml
apiVersion: 1

datasources:
  - name: MySQL
    type: mysql
    url: localhost:3306
    user: grafana
    jsonData:
      database: grafana
      maxOpenConns: 100 # Grafana v5.4+
      maxIdleConns: 100 # Grafana v5.4+
      maxIdleConnsAuto: true # Grafana v9.5.1+
      connMaxLifetime: 14400 # Grafana v5.4+
    secureJsonData:
      password: ${GRAFANA_MYSQL_PASSWORD}
```

Using TLS verification

yaml Copy

```yaml
apiVersion: 1

datasources:
  - name: MySQL
    type: mysql
    url: localhost:3306
    user: grafana
    jsonData:
      tlsAuth: true
      database: grafana
      maxOpenConns: 100 # Grafana v5.4+
      maxIdleConns: 100 # Grafana v5.4+
      maxIdleConnsAuto: true # Grafana v9.5.1+
      connMaxLifetime: 14400 # Grafana v5.4+
    secureJsonData:
      password: ${GRAFANA_MYSQL_PASSWORD}
      tlsClientCert: ${GRAFANA_TLS_CLIENT_CERT}
      tlsCACert: ${GRAFANA_TLS_CA_CERT}
```

Use TLS and Skip Certificate Verification

yaml Copy

```yaml
apiVersion: 1

datasources:
  - name: MySQL
    type: mysql
    url: localhost:3306
    user: grafana
    jsonData:
      tlsAuth: true
      tlsSkipVerify: true
      database: grafana
      maxOpenConns: 100 # Grafana v5.4+
      maxIdleConns: 100 # Grafana v5.4+
      maxIdleConnsAuto: true # Grafana v9.5.1+
      connMaxLifetime: 14400 # Grafana v5.4+
    secureJsonData:
      password: ${GRAFANA_MYSQL_PASSWORD}
      tlsClientCert: ${GRAFANA_TLS_CLIENT_CERT}
      tlsCACert: ${GRAFANA_TLS_CA_CERT}
```

Expand code

Query builder

The MySQL query builder is available when editing a panel using a MySQL data source.

This topic explains querying specific to the MySQL data source. For general documentation on querying data sources in Grafana, see Query and transform data.

You can run the built query by pressing the `Run query` button in the top right corner of the editor.

Format

The response from MySQL can be formatted as either a table or as a time series. To use the time series format one of the columns must be named `time`.

Dataset and Table selection

> **NOTE**
>
> If your table or database name contains a reserved word or a prohibited character the editor will put quotes around the name. For example, the name `table-name` will be quoted with backticks - `` `table-name` ``.

In the dataset dropdown, choose the MySQL database to query. The dropdown is be populated with the databases that the user has access to. When the dataset is selected, the table dropdown is populated with the tables that are available.

Note: If a default database has been configured through the Data Source Configuration page (or through a provisioning configuration file), the user will only be able to use that single preconfigured database for querying.

Columns and Aggregation functions (SELECT)

Using the dropdown, select a column to include in the data. You can also specify an optional aggregation function.

Add further value columns by clicking the plus button and another column dropdown appears.

Filter data (WHERE)

To add a filter, toggle the **Filter** switch at the top of the editor. This reveals a **Filter by column value** section with two dropdown selectors.

Use the first dropdown to choose whether all of the filters need to match (`AND`), or if only one of the filters needs to match (`OR`). Use the second dropdown to choose a filter.

To filter on more columns, click the plus (`+`) button to the right of the condition dropdown.

To remove a filter, click the `x` button next to that filter's dropdown.

After selecting a date type column, you can choose Macros from the operators list and select timeFilter which will add the $__timeFilter macro to the query with the selected date column.

Group By

To group the results by column, flip the group switch at the top of the editor. You can then choose which column to group the results by. The group by clause can be removed by pressing the X

button.

Preview

By flipping the preview switch at the top of the editor, you can get a preview of the SQL query generated by the query builder.

Code editor

To make advanced queries, switch to the code editor by clicking `code` in the top right corner of the editor. The code editor support autocompletion of tables, columns, SQL keywords, standard sql functions, Grafana template variables and Grafana macros. Columns cannot be completed before a table has been specified.

You can expand the code editor by pressing the `chevron` pointing downwards in the lower right corner of the code editor.

`CTRL/CMD + Return` works as a keyboard shortcut to run the query.

Macros

To simplify syntax and to allow for dynamic parts, like date range filters, the query can contain macros.

Macro example	Description
`$__time(dateColumn)`	Will be replaced by an expression to convert to a UNIX timestamp and rename the column to `time_sec`. For example, *UNIX_TIMESTAMP(dateColumn) as time_sec*
`$__timeEpoch(dateColumn)`	Will be replaced by an expression to convert to a UNIX timestamp and rename the column to `time_sec`. For example, *UNIX_TIMESTAMP(dateColumn) as time_sec*
`$__timeFilter(dateColumn)`	Will be replaced by a time range filter using the specified column name. For example, *dateColumn BETWEEN*

Macro example	Description
	FROM_UNIXTIME(1494410783) AND FROM_UNIXTIME(1494410983)
`$__timeFrom()`	Will be replaced by the start of the currently active time selection. For example, *FROM_UNIXTIME(1494410783)*
`$__timeTo()`	Will be replaced by the end of the currently active time selection. For example, *FROM_UNIXTIME(1494410983)*
`$__timeGroup(dateColumn,'5m')`	Will be replaced by an expression usable in GROUP BY clause. For example, *cast(cast(UNIX_TIMESTAMP(dateColumn)/(300) as signed)*300 as signed),*
`$__timeGroup(dateColumn,'5m', 0)`	Same as above but with a fill parameter so missing points in that series will be added by grafana and 0 will be used as value (only works with time series queries).
`$__timeGroup(dateColumn,'5m', NULL)`	Same as above but NULL will be used as value for missing points (only works with time series queries).
`$__timeGroup(dateColumn,'5m', previous)`	Same as above but the previous value in that series will be used as fill value if no value has been seen yet NULL will be used (only works with time series queries).
`$__timeGroupAlias(dateColumn,'5m')`	Will be replaced identical to $__timeGroup but with an added column alias (only available in Grafana 5.3+).
`$__unixEpochFilter(dateColumn)`	Will be replaced by a time range filter using the specified column name with times represented as Unix timestamp. For example, *dateColumn > 1494410783 AND dateColumn < 1494497183*
`$__unixEpochFrom()`	Will be replaced by the start of the currently active time selection as Unix timestamp. For example, *1494410783*
`$__unixEpochTo()`	Will be replaced by the end of the currently active time selection as Unix timestamp. For example, *1494497183*
`$__unixEpochNanoFilter(dateColumn)`	Will be replaced by a time range filter using the specified column name with times represented as nanosecond timestamp. For example, *dateColumn > 1494410783152415214 AND dateColumn < 1494497183142514872*
`$__unixEpochNanoFrom()`	Will be replaced by the start of the currently active time selection as nanosecond timestamp. For example, *1494410783152415214*
`$__unixEpochNanoTo()`	Will be replaced by the end of the currently active time selection as nanosecond timestamp. For example, *1494497183142514872*
`$__unixEpochGroup(dateColumn,'5m', [fillmode])`	Same as $__timeGroup but for times stored as Unix timestamp (`fillMode` only works with time series queries).
`$__unixEpochGroupAlias(dateColumn,'5m', [fillmode])`	Same as above but also adds a column alias (`fillMode` only works with time series queries).

We plan to add many more macros. If you have suggestions for what macros you would like to see, please open an issue in our GitHub repo.

The query editor has a link named `Generated SQL` that shows up after a query has been executed, while in panel edit mode. Click on it and it will expand and show the raw interpolated SQL string that was executed.

Table queries

If the `Format as` query option is set to `Table` then you can basically do any type of SQL query. The table panel will automatically show the results of whatever columns and rows your query returns.

Query editor with example query:

The query:

```sql
SELECT
  title as 'Title',
  user.login as 'Created By' ,
  dashboard.created as 'Created On'
 FROM dashboard
INNER JOIN user on user.id = dashboard.created_by
WHERE $__timeFilter(dashboard.created)
```

You can control the name of the Table panel columns by using regular `as` SQL column selection syntax.

The resulting table panel:

Time series queries

The examples in this section query the following table:

text

Copy

```
+---------------------+--------------+---------------------+----------+
| time_date_time      | value_double | CreatedAt           | hostname |
+---------------------+--------------+---------------------+----------+
| 2020-01-02 03:05:00 | 3.0          | 2020-01-02 03:05:00 | 10.0.1.1 |
| 2020-01-02 03:06:00 | 4.0          | 2020-01-02 03:06:00 | 10.0.1.2 |
| 2020-01-02 03:10:00 | 6.0          | 2020-01-02 03:10:00 | 10.0.1.1 |
| 2020-01-02 03:11:00 | 7.0          | 2020-01-02 03:11:00 | 10.0.1.2 |
| 2020-01-02 03:20:00 | 5.0          | 2020-01-02 03:20:00 | 10.0.1.2 |
+---------------------+--------------+---------------------+----------+
```

If the `Format as` query option is set to `Time Series` then the query must have a column named time that returns either a SQL datetime or any numeric datatype representing Unix epoch in seconds. In addition, result sets of time series queries must be sorted by time for panels to properly visualize the result.

A time series query result is returned in a wide data frame format. Any column except time or of type string transforms into value fields in the data frame query result. Any string column transforms into field labels in the data frame query result.

> For backward compatibility, there's an exception to the above rule for queries that return three columns including a string column named metric. Instead of transforming the metric column into field labels, it becomes the field name, and then the series name is formatted as the value of the metric column. See the example with the metric column below.

To optionally customize the default series name formatting, refer to Standard options definitions.

Example with `metric` column:

sql

Copy

```sql
SELECT
  $__timeGroupAlias(time_date_time,'5m'),
  min(value_double),
  'min' as metric
FROM test_data
WHERE $__timeFilter(time_date_time)
```

```
GROUP BY time
ORDER BY time
```

Data frame result:

text Copy

```
+--------------------+-----------------+
| Name: time         | Name: min       |
| Labels:            | Labels:         |
| Type: []time.Time  | Type: []float64 |
+--------------------+-----------------+
| 2020-01-02 03:05:00 | 3              |
| 2020-01-02 03:10:00 | 6              |
| 2020-01-02 03:20:00 | 5              |
+--------------------+-----------------+
```

Example using the fill parameter in the $__timeGroupAlias macro to convert null values to be zero instead:

sql Copy

```
SELECT
  $__timeGroupAlias(createdAt,'5m',0),
  sum(value_double) as value,
  hostname
FROM test_data
WHERE
  $__timeFilter(createdAt)
GROUP BY time, hostname
ORDER BY time
```

Given the data frame result in the following example and using the graph panel, you will get two series named *value 10.0.1.1* and *value 10.0.1.2*. To render the series with a name of *10.0.1.1* and *10.0.1.2* , use a Standard options definitions display value of `${__field.labels.hostname}`.

Data frame result:

text Copy

```
+--------------------+--------------------------+--------------------------+
| Name: time         | Name: value              | Name: value              |
| Labels:            | Labels: hostname=10.0.1.1 | Labels: hostname=10.0.1.2 |
| Type: []time.Time  | Type: []float64          | Type: []float64          |
+--------------------+--------------------------+--------------------------+
```

```
| 2020-01-02 03:05:00 | 3                    | 4                    |
| 2020-01-02 03:10:00 | 6                    | 7                    |
| 2020-01-02 03:15:00 | 0                    | 0                    |
| 2020-01-02 03:20:00 | 0                    | 5                    |
+---------------------+----------------------+----------------------+
```

Example with multiple columns:

sql Copy

```sql
SELECT
  $__timeGroupAlias(time_date_time,'5m'),
  min(value_double) as min_value,
  max(value_double) as max_value
FROM test_data
WHERE $__timeFilter(time_date_time)
GROUP BY time
ORDER BY time
```

Data frame result:

text Copy

```
+---------------------+-----------------+-----------------+
| Name: time          | Name: min_value | Name: max_value |
| Labels:             | Labels:         | Labels:         |
| Type: []time.Time   | Type: []float64 | Type: []float64 |
+---------------------+-----------------+-----------------+
| 2020-01-02 03:05:00 | 3               | 4               |
| 2020-01-02 03:10:00 | 6               | 7               |
| 2020-01-02 03:20:00 | 5               | 5               |
+---------------------+-----------------+-----------------+
```

Currently, there is no support for a dynamic group by time based on time range and panel width. This is something we plan to add.

Templating

This feature is currently available in the nightly builds and will be included in the 5.0.0 release.

Instead of hard-coding things like server, application and sensor name in your metric queries you can use variables in their place. Variables are shown as dropdown select boxes at the top of the dashboard. These dropdowns make it easy to change the data being displayed in your dashboard.

Check out the Templating documentation for an introduction to the templating feature and the different types of template variables.

Query Variable

If you add a template variable of the type `Query`, you can write a MySQL query that can return things like measurement names, key names or key values that are shown as a dropdown select box.

For example, you can have a variable that contains all values for the `hostname` column in a table if you specify a query like this in the templating variable *Query* setting.

```sql
SELECT hostname FROM my_host
```

A query can return multiple columns and Grafana will automatically create a list from them. For example, the query below will return a list with values from `hostname` and `hostname2`.

```sql
SELECT my_host.hostname, my_other_host.hostname2 FROM my_host JOIN my_other_host ON my_host.city =
```

To use time range dependent macros like `$__timeFilter(column)` in your query the refresh mode of the template variable needs to be set to *On Time Range Change*.

```sql
SELECT event_name FROM event_log WHERE $__timeFilter(time_column)
```

Another option is a query that can create a key/value variable. The query should return two columns that are named `__text` and `__value`. The `__text` column value should be unique (if it is not unique then the first value is used). The options in the dropdown will have a text and value that allows you to have a friendly name as text and an id as the value. An example query with `hostname` as the text and `id` as the value:

```sql
SELECT hostname AS __text, id AS __value FROM my_host
```

You can also create nested variables. For example if you had another variable named `region`. Then you could have the hosts variable only show hosts from the current selected region with a query like this (if `region` is a multi-value variable then use the `IN` comparison operator rather than `=` to match against multiple values):

```sql
SELECT hostname FROM my_host  WHERE region IN($region)
```

Using `__searchFilter` to filter results in Query Variable

> Available from Grafana 6.5 and above

Using `__searchFilter` in the query field will filter the query result based on what the user types in the dropdown select box. When nothing has been entered by the user the default value for `__searchFilter` is `%`.

> Important that you surround the `__searchFilter` expression with quotes as Grafana does not do this for you.

The example below shows how to use `__searchFilter` as part of the query field to enable searching for `hostname` while the user types in the dropdown select box.

Query

```sql
SELECT hostname FROM my_host  WHERE hostname LIKE '$__searchFilter'
```

Using Variables in Queries

From Grafana 4.3.0 to 4.6.0, template variables are always quoted automatically so if it is a string value do not wrap them in quotes in where clauses.

From Grafana 4.7.0, template variable values are only quoted when the template variable is a `multi-value`.

If the variable is a multi-value variable then use the `IN` comparison operator rather than `=` to match against multiple values.

There are two syntaxes:

`$<varname>` Example with a template variable named `hostname`:

```sql
SELECT
  UNIX_TIMESTAMP(atimestamp) as time,
  aint as value,
  avarchar as metric
FROM my_table
```

```
WHERE $__timeFilter(atimestamp) and hostname in($hostname)
ORDER BY atimestamp ASC
```

`[[varname]]` Example with a template variable named `hostname`:

sql Copy

```sql
SELECT
  UNIX_TIMESTAMP(atimestamp) as time,
  aint as value,
  avarchar as metric
FROM my_table
WHERE $__timeFilter(atimestamp) and hostname in([[hostname]])
ORDER BY atimestamp ASC
```

Disabling Quoting for Multi-value Variables

Grafana automatically creates a quoted, comma-separated string for multi-value variables. For example: if `server01` and `server02` are selected then it will be formatted as: `'server01', 'server02'`. To disable quoting, use the csv formatting option for variables:

`${servers:csv}`

Read more about variable formatting options in the Variables documentation.

Annotations

Annotations allow you to overlay rich event information on top of graphs. You add annotation queries via the Dashboard menu / Annotations view.

Example query using time column with epoch values:

sql Copy

```sql
SELECT
  epoch_time as time,
  metric1 as text,
  CONCAT(tag1, ',', tag2) as tags
FROM
  public.test_data
WHERE
  $__unixEpochFilter(epoch_time)
```

Example region query using time and timeend columns with epoch values:

Only available in Grafana v6.6+.

sql Copy

```sql
SELECT
  epoch_time as time,
  epoch_timeend as timeend,
  metric1 as text,
  CONCAT(tag1, ',', tag2) as tags
FROM
  public.test_data
WHERE
  $__unixEpochFilter(epoch_time)
```

Example query using time column of native SQL date/time data type:

sql Copy

```sql
SELECT
  native_date_time as time,
  metric1 as text,
  CONCAT(tag1, ',', tag2) as tags
FROM
  public.test_data
WHERE
  $__timeFilter(native_date_time)
```

Name	Description
`time`	The name of the date/time field. Could be a column with a native SQL date/time data type or epoch value.
`timeend`	Optional name of the end date/time field. Could be a column with a native SQL date/time data type or epoch value. (Grafana v6.6+)
`text`	Event description field.
`tags`	Optional field name to use for event tags as a comma separated string.

Alerting

Time series queries should work in alerting conditions. Table formatted queries are not yet supported in alert rule conditions.

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

OpenTSDB data source

Grafana ships with advanced support for OpenTSDB. This topic explains configuration, variables, querying, and other features specific to the OpenTSDB data source.

For instructions on how to add a data source to Grafana, refer to the administration documentation. Only users with the organization administrator role can add data sources. Administrators can also configure the data source via YAML with Grafana's provisioning system.

OpenTSDB settings

To configure basic settings for the data source, complete the following steps:

1. Click **Connections** in the left-side menu.
2. Under Your connections, click **Data sources**.
3. Enter `OpenTSDB` in the search bar.
4. Select **OpenTSDB**.

   The **Settings** tab of the data source is displayed.
5. Set the data source's basic configuration options:

Name	Description
Name	The data source name. This is how you refer to the data source in panels and queries.
Default	Default data source that will be be pre-selected for new panels.
URL	The HTTP protocol, IP, and port of your OpenTSDB server (default port is usually 4242).

Name	Description
Allowed cookies	Listing of cookies to forward to the data source.
Version	The OpenTSDB version.
Resolution	Metrics from OpenTSDB may have data points with either second or millisecond resolution.
Lookup limit	Default is 1000.

Provision the data source

You can define and configure the data source in YAML files as part of Grafana's provisioning system. For more information about provisioning, and for available configuration options, refer to Provisioning Grafana.

Provisioning example

yaml Copy

```yaml
apiVersion: 1

datasources:
  - name: OpenTSDB
    type: opentsdb
    access: proxy
    url: http://localhost:4242
    jsonData:
      tsdbResolution: 1
      tsdbVersion: 1
```

Query editor

Open a graph in edit mode by click the title. Query editor will differ if the data source has version <=2.1 or = 2.2. In the former version, only tags can be used to query OpenTSDB. But in the latter version, filters as well as tags can be used to query OpenTSDB. Fill Policy is also introduced in OpenTSDB 2.2.

> **NOTE**
>
> While using OpenTSDB 2.2 data source, make sure you use either Filters or Tags as they are mutually exclusive. If used together, might give you weird results.

Auto complete suggestions

As soon as you start typing metric names, tag names and tag values , you should see highlighted auto complete suggestions for them. The autocomplete only works if the OpenTSDB suggest API is enabled.

Templating queries

Instead of hard-coding things like server, application and sensor name in your metric queries you can use variables in their place. Variables are shown as dropdown select boxes at the top of the dashboard. These dropdowns make it easy to change the data being displayed in your dashboard.

Check out the Templating documentation for an introduction to the templating feature and the different types of template variables.

Query variable

Grafana's OpenTSDB data source supports template variable queries. This means you can create template variables that fetch the values from OpenTSDB. For example, metric names, tag names, or tag values.

When using OpenTSDB with a template variable of `query` type you can use following syntax for lookup.

Query	Description
`metrics(prefix)`	Returns metric names with specific prefix (can be empty)
`tag_names(cpu)`	Returns tag names (i.e. keys) for a specific cpu metric
`tag_values(cpu, hostname)`	Returns tag values for metric cpu and tag key hostname
`suggest_tagk(prefix)`	Returns tag names (i.e. keys) for all metrics with specific prefix (can be empty)
`suggest_tagv(prefix)`	Returns tag values for all metrics with specific prefix (can be empty)

If you do not see template variables being populated in `Preview of values` section, you need to enable `tsd.core.meta.enable_realtime_ts` in the OpenTSDB server settings. Also, to populate metadata of the existing time series data in OpenTSDB, you need to run `tsdb uid metasync` on the OpenTSDB server.

Nested templating

One template variable can be used to filter tag values for another template variable. First parameter is the metric name, second parameter is the tag key for which you need to find tag values, and after that all other dependent template variables. Some examples are mentioned below to make nested template queries work successfully.

Query	Description
`tag_values(cpu, hostname, env=$env)`	Return tag values for cpu metric, selected env tag value and tag key hostname
`tag_values(cpu, hostname, env=$env, region=$region)`	Return tag values for cpu metric, selected env tag value, selected region tag value and tag key hostname

For details on OpenTSDB metric queries, check out the official OpenTSDB documentation

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Grafana Cloud | Enterprise | Open source

Parca data source

Grafana ships with built-in support for Parca, a continuous profiling OSS database for analysis of CPU and memory usage, down to the line number and throughout time. Add it as a data source, and you are ready to query your profiles in Explore.

Configure the Parca data source

To configure basic settings for the data source, complete the following steps:

1. Click **Connections** in the left-side menu.
2. Under Your connections, click **Data sources**.
3. Enter `Parca` in the search bar.
4. Click **Parca**.

   The **Settings** tab of the data source is displayed.
5. Set the data source's basic configuration options:

Name	Description
`Name`	A name to specify the data source in panels, queries, and Explore.
`Default`	The default data source will be pre-selected for new panels.
`URL`	The URL of the Parca instance, e.g., `http://localhost:4100`
`Basic Auth`	Enable basic authentication to the Parca data source.
`User`	User name for basic authentication.

Name	Description
`Password`	Password for basic authentication.

Querying

Query Editor

Query editor gives you access to a profile type selector, a label selector, and collapsible options.

Select a profile type from the drop-down menu. While the label selector can be left empty to query all profiles without filtering by labels, the profile type must be selected for the query to be valid. Grafana does not show any data if the profile type isn't selected when a query is run.

Use the labels selector input to filter by labels. Parca uses similar syntax to Prometheus to filter labels. Refer to Parca documentation for available operators and syntax.

Select a query type to return the profile data which can be shown in the Flame Graph, metric data visualized in a graph, or both. You can only select both options in a dashboard, because panels allow only one visualization.

Profiles query results

Profiles can be visualized in a flame graph. See the Flame Graph documentation to learn about the visualization and its features.

Parca returns profiles aggregated over a selected time range, and the absolute values in the flame graph grow as the time range gets bigger while keeping the relative values meaningful. You can zoom in on the time range to get a higher granularity profile up to the point of a single Parca scrape interval.

Metrics query results

Metrics results represent the aggregated value, over time, of the selected profile type. Parca returns ungrouped data with a series for each label combination.

This allows you to quickly see any spikes in the value of the scraped profiles and zoom in to a particular time range.

Provision the Parca data source

You can modify the Grafana configuration files to provision the Parca data source. To learn more, and to view the available provisioning settings, see provisioning documentation.

Here is an example config:

yaml Copy

```yaml
apiVersion: 1

datasources:
  - name: Parca
    type: parca
    url: http://localhost:3100
```

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Grafana Pyroscope data source

Grafana Pyroscope is a horizontally scalable, highly available, multi-tenant, OSS, continuous profiling aggregation system. Add it as a data source, and you are ready to query your profiles in Explore.

Refer to Introduction to Pyroscope to understand profiling and Pyroscope.

To use profiling data, you should:

- Configure your application to send profiles
- Configure the Grafana Pyroscope data source.
- View and query profiling data in Explore

Integrate profiles into dashboards

Using the Pyroscope data source, you can integrate profiles into your dashboards. In this case, the screenshot shows memory profiles alongside panels for logs and metrics to be able to debug out of memory (OOM) errors alongside the associated logs and metrics.

Visualize traces and profiles data using Traces to profiles

You can link profile and tracing data using your Pyroscope data source with the Tempo data source. To learn more about how profiles and tracing can work together, refer to Profiling and tracing synergies.

Combined traces and profiles let you see granular line-level detail when available for a trace span. This allows you pinpoint the exact function that's causing a bottleneck in your application as well as a specific request.

For more information, refer to the Traces to profile section and Link tracing and profiling with span profiles.

Provision the Grafana Pyroscope data source

You can modify the Grafana configuration files to provision the Grafana Pyroscope data source. To learn more, and to view the available provisioning settings, refer to provisioning documentation.

Here is an example configuration:

yaml Copy

```yaml
apiVersion: 1

datasources:
  - name: Grafana Pyroscope
    type: grafana-pyroscope-datasource
    url: http://localhost:4040
    jsonData:
      minStep: '15s'
```

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Configure the Grafana Pyroscope data source

To configure basic settings for the data source, complete the following steps:

1. Click **Connections** in the left-side menu.
2. Under Your connections, click **Data sources**.
3. Enter `Grafana Pyroscope` in the search bar.
4. Click **Grafana Pyroscope** to display the **Settings** tab of the data source.
5. Set the data source's basic configuration options:

Name	Description
`Name`	A name to specify the data source in panels, queries, and Explore.
`Default`	The default data source will be pre-selected for new panels.
`URL`	The URL of the Grafana Pyroscope instance, for example, `http://localhost:4100`.
`Basic Auth`	Enable basic authentication to the data source.
`User`	User name for basic authentication.
`Password`	Password for basic authentication.
`Minimal step`	Used for queries returning timeseries data. The Pyroscope backend, similar to Prometheus, scrapes profiles at certain intervals. To prevent querying at smaller interval, use Minimal step same or higher than your Pyroscope scrape interval. This prevents returning too many data points to the frontend.

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

How profiling and tracing work together

Profiles, continuous profiling, and distributed traces are all tools that can be used to improve the performance and reliability of applications. However, each tool has its own strengths and weaknesses, and it is important to choose the right tool for the job as well as understand when to use both.

Profiling

Profiling offers a deep-dive into an application's performance at the code level, highlighting resource usage and performance hotspots.

Usage	During development, major releases, or upon noticing performance quirks.
Benefits	• Business: Boosts user experience through enhanced application performance. • Technical: Gives clear insights into code performance and areas of refinement.
Example	A developer uses profiling upon noting slow app performance, identifies a CPU-heavy function, and optimizes it.

Continuous profiling

Continuous profiling provides ongoing performance insights, capturing long-term trends and intermittent issues.

Usage	Mainly in production, especially for high-priority applications.

Benefits	• Business: Preemptively addresses inefficiencies, potentially saving costs. • Technical: Highlights performance trends and issues like potential memory leaks over time.
Example	A month-long data from continuous profiling suggests increasing memory consumption, hinting at a memory leak.

Distributed tracing

Traces requests as they cross multiple services, revealing interactions and service dependencies.

Usage	Essential for systems like microservices where requests touch multiple services.
Benefits	• Business: Faster issue resolution, reduced downtimes, and strengthened customer trust. • Technical: A broad view of the system's structure, revealing bottlenecks and inter-service dependencies.
Example	In e-commerce, a user's checkout request might involve various services. Tracing depicts this route, pinpointing where time is most spent.

Combined power of tracing and profiling

When used together, tracing and profiling provide a powerful tool for understanding system and application performance.

Usage	For comprehensive system-to-code insights, especially when diagnosing complex issues spread across services and codebases.

Benefits	• Business: Reduces downtime, optimizes user experience, and safeguards revenues.
	• Technical:
	  • Holistic view: Tracing pinpoints bottle-necked services, while profiling delves into the responsible code segments.
	  • End-to-end insight: Visualizes a request's full journey and the performance of individual code parts.
	  • Efficient diagnosis: Tracing identifies service latency; profiling zeroes in on its cause, be it database queries, API calls, or specific code inefficiencies.

Example	Tracing reveals latency in a payment service. Combined with profiling, it's found that a particular function, making third-party validation calls, is the culprit. This insight guides optimization, refining system efficiency.

Query profile data

The Pyroscope data source query editor gives you access to a profile type selector, a label selector, and collapsible options.

To access the query editor:

1. Sign into Grafana or Grafana Cloud.
2. Select your Pyroscope data source.
3. From the menu, choose **Explore**.
4. Select a profile type from the drop-down menu.

Profile selector

5 Use the labels selector input to filter by labels. Pyroscope uses similar syntax to Prometheus to filter labels. Refer to Pyroscope documentation for available operators and syntax.

   While the label selector can be left empty to query all profiles without filtering by labels, the profile type or app must be selected for the query to be valid.

   Grafana doesn't show any data if the profile type or app isn't selected when a query runs.

6 Expand the **Options** section to view **Query Type** and **Group by**.

7 Select a query type to return the profile data. Data is shown in the Flame Graph, metric data visualized in a graph, or both. You can only select both options in Explore. The panels used on dashboards allow only one visualization.

Using **Group by**, you can group metric data by a specified label. Without any **Group by** label, metric data aggregates over all the labels into single time series. You can use multiple labels to group by. Group by only effects the metric data and doesn't change the profile data results.

Profiles query results

Profiles can be visualized in a flame graph. Refer to the Flame Graph documentation to learn about the visualization and its features.

Pyroscope returns profiles aggregated over a selected time range. The absolute values in the flame graph grow as the time range gets bigger while keeping the relative values meaningful. You can zoom in on the time range to get a higher granularity profile up to the point of a single scrape interval.

Metrics query results

Metrics results represent the aggregated sum value over time of the selected profile type.

This allows you to quickly see any spikes in the value of the scraped profiles and zoom in to a particular time range.

PostgreSQL data source

Grafana ships with a built-in PostgreSQL data source plugin that allows you to query and visualize data from a PostgreSQL compatible database.

For instructions on how to add a data source to Grafana, refer to the administration documentation. Only users with the organization administrator role can add data sources. Administrators can also configure the data source via YAML with Grafana's provisioning system.

PostgreSQL settings

To configure basic settings for the data source, complete the following steps:

1. Click **Connections** in the left-side menu.
2. Under Your connections, click **Data sources**.
3. Enter `PostgreSQL` in the search bar.
4. Select **PostgreSQL**.

   The **Settings** tab of the data source is displayed.
5. Set the data source's basic configuration options:

Name	Description
Name	The data source name. This is how you refer to the data source in panels and queries.
Default	Default data source means that it will be pre-selected for new panels.
Host	The IP address/hostname and optional port of your PostgreSQL instance. *Do not* include the database name. The connection string for connecting to Postgres will not be correct and it may cause errors.

Name	Description
Database	Name of your PostgreSQL database.
User	Database user's login/username
Password	Database user's password
SSL Mode	Determines whether or with what priority a secure SSL TCP/IP connection will be negotiated with the server. When SSL Mode is disabled, SSL Method and Auth Details would not be visible.
SSL Auth Details Method	Determines whether the SSL Auth details will be configured as a file path or file content. Grafana v7.5+
SSL Auth Details Value	File path or file content of SSL root certificate, client certificate and client key
Max open	The maximum number of open connections to the database, default `100` (Grafana v5.4+).
Max idle	The maximum number of connections in the idle connection pool, default `100` (Grafana v5.4+).
Auto (max idle)	If set will set the maximum number of idle connections to the number of maximum open connections (Grafana v9.5.1+). Default is `true`.
Max lifetime	The maximum amount of time in seconds a connection may be reused, default `14400`/4 hours (Grafana v5.4+).
Version	Determines which functions are available in the query builder (only available in Grafana 5.3+).
TimescaleDB	A time-series database built as a PostgreSQL extension. When enabled, Grafana uses `time_bucket` in the `$__timeGroup` macro to display TimescaleDB specific aggregate functions in the query builder (only available in Grafana 5.3+). For more information, see TimescaleDB documentation.

Min time interval

A lower limit for the `$__interval` and `$__interval_ms` variables. Recommended to be set to write frequency, for example `1m` if your data is written every minute. This option can also be overridden/configured in a dashboard panel under data source options. It's important to note that this value **needs** to be formatted as a number followed by a valid time identifier, e.g. `1m` (1 minute) or `30s` (30 seconds). The following time identifiers are supported:

Identifier	Description
`y`	year
`M`	month
`w`	week
`d`	day

Identifier	Description
`h`	hour
`m`	minute
`s`	second
`ms`	millisecond

Database user permissions (Important!)

The database user you specify when you add the data source should only be granted SELECT permissions on the specified database and tables you want to query. Grafana does not validate that the query is safe. The query could include any SQL statement. For example, statements like `DELETE FROM user;` and `DROP TABLE user;` would be executed. To protect against this we **highly** recommend you create a specific PostgreSQL user with restricted permissions.

Example:

```sql
CREATE USER grafanareader WITH PASSWORD 'password';
GRANT USAGE ON SCHEMA schema TO grafanareader;
GRANT SELECT ON schema.table TO grafanareader;
```

Make sure the user does not get any unwanted privileges from the public role.

Query builder

PostgreSQL query builder

The PostgreSQL query builder is available when editing a panel using a PostgreSQL data source. The built query can be run by pressing the `Run query` button in the top right corner of the editor.

Format

The response from PostgreSQL can be formatted as either a table or as a time series. To use the time series format one of the columns must be named `time`.

Dataset and table selection

The dataset dropdown will be populated with the configured database to which the user has access. The table dropdown is populated with the tables that are available within that database.

Columns and Aggregation functions (SELECT)

Using the dropdown, select a column to include in the data. You can also specify an optional aggregation function.

Add further value columns by clicking the plus button and another column dropdown appears.

Filter data (WHERE)

To add a filter, toggle the **Filter** switch at the top of the editor. This reveals a **Filter by column value** section with two dropdown selectors.

Use the first dropdown to choose whether all of the filters need to match (`AND`), or if only one of the filters needs to match (`OR`). Use the second dropdown to choose a filter.

To filter on more columns, click the plus (`+`) button to the right of the condition dropdown.

To remove a filter, click the `x` button next to that filter's dropdown.

After selecting a date type column, you can choose Macros from the operators list and select timeFilter which will add the $__timeFilter macro to the query with the selected date column.

Group By

To group the results by column, flip the group switch at the top of the editor. You can then choose which column to group the results by. The group by clause can be removed by pressing the X button.

Preview

By flipping the preview switch at the top of the editor, you can get a preview of the SQL query generated by the query builder.

Provision the data source

It's now possible to configure data sources using config files with Grafana's provisioning system. You can read more about how it works and all the settings you can set for data sources on the

provisioning docs page.

Provisioning example

yaml Copy

```yaml
apiVersion: 1

datasources:
  - name: Postgres
    type: postgres
    url: localhost:5432
    user: grafana
    secureJsonData:
      password: 'Password!'
    jsonData:
      database: grafana
      sslmode: 'disable' # disable/require/verify-ca/verify-full
      maxOpenConns: 100 # Grafana v5.4+
      maxIdleConns: 100 # Grafana v5.4+
      maxIdleConnsAuto: true # Grafana v9.5.1+
      connMaxLifetime: 14400 # Grafana v5.4+
      postgresVersion: 903 # 903=9.3, 904=9.4, 905=9.5, 906=9.6, 1000=10
      timescaledb: false
```

> **NOTE**
>
> In the above code, the `postgresVersion` value of `10` refers to version PostgreSQL 10 and above.

Troubleshoot provisioning

If you encounter metric request errors or other issues:

- Make sure your data source YAML file parameters exactly match the example. This includes parameter names and use of quotation marks.
- Make sure the `database` name is not included in the `url`.

Code editor

To make advanced queries, switch to the code editor by clicking `code` in the top right corner of the editor. The code editor support autocompletion of tables, columns, SQL keywords, standard sql functions, Grafana template variables and Grafana macros. Columns cannot be completed before a table has been specified.

You can expand the code editor by pressing the `chevron` pointing downwards in the lower right corner of the code editor.

`CTRL/CMD + Return` works as a keyboard shortcut to run the query.

Macros

Macros can be used within a query to simplify syntax and allow for dynamic parts.

Macro example	Description
`$__time(dateColumn)`	Will be replaced by an expression to convert to a UNIX timestamp and rename the column to `time_sec`. For example, *UNIX_TIMESTAMP(dateColumn) as time_sec*
`$__timeEpoch(dateColumn)`	Will be replaced by an expression to convert to a UNIX timestamp and rename the column to `time_sec`. For example, *UNIX_TIMESTAMP(dateColumn) as time_sec*
`$__timeFilter(dateColumn)`	Will be replaced by a time range filter using the specified column name. For example, *dateColumn BETWEEN FROM_UNIXTIME(1494410783) AND FROM_UNIXTIME(1494410983)*
`$__timeFrom()`	Will be replaced by the start of the currently active time selection. For example, *FROM_UNIXTIME(1494410783)*
`$__timeTo()`	Will be replaced by the end of the currently active time selection. For example, *FROM_UNIXTIME(1494410983)*
`$__timeGroup(dateColumn,'5m')`	Will be replaced by an expression usable in GROUP BY clause. For example, *cast(cast(UNIX_TIMESTAMP(dateColumn)/(300) as signed)*300 as signed),*
`$__timeGroup(dateColumn,'5m', 0)`	Same as above but with a fill parameter so missing points in that series will be added by grafana and 0 will be used as

Macro example	Description
	value (only works with time series queries).
`$__timeGroup(dateColumn,'5m', NULL)`	Same as above but NULL will be used as value for missing points (only works with time series queries).
`$__timeGroup(dateColumn,'5m', previous)`	Same as above but the previous value in that series will be used as fill value if no value has been seen yet NULL will be used (only works with time series queries).
`$__timeGroupAlias(dateColumn,'5m')`	Will be replaced identical to $__timeGroup but with an added column alias.
`$__unixEpochFilter(dateColumn)`	Will be replaced by a time range filter using the specified column name with times represented as Unix timestamp. For example, *dateColumn > 1494410783 AND dateColumn < 1494497183*
`$__unixEpochFrom()`	Will be replaced by the start of the currently active time selection as Unix timestamp. For example, *1494410783*
`$__unixEpochTo()`	Will be replaced by the end of the currently active time selection as Unix timestamp. For example, *1494497183*
`$__unixEpochNanoFilter(dateColumn)`	Will be replaced by a time range filter using the specified column name with times represented as nanosecond timestamp. For example, *dateColumn > 1494410783152415214 AND dateColumn < 1494497183142514872*
`$__unixEpochNanoFrom()`	Will be replaced by the start of the currently active time selection as nanosecond timestamp. For example, *1494410783152415214*
`$__unixEpochNanoTo()`	Will be replaced by the end of the currently active time selection as nanosecond timestamp. For example, *1494497183142514872*
`$__unixEpochGroup(dateColumn,'5m', [fillmode])`	Same as $__timeGroup but for times stored as Unix timestamp (`fillMode` only works with time series queries).
`$__unixEpochGroupAlias(dateColumn,'5m', [fillmode])`	Same as above but also adds a column alias (`fillMode` only works with time series queries).

We plan to add many more macros. If you have suggestions for what macros you would like to see, please open an issue in our GitHub repo.

Table queries

If the `Format as` query option is set to `Table` then you can basically do any type of SQL query. The table panel will automatically show the results of whatever columns and rows your query returns.

Query editor with example query:

The query:

```sql
SELECT
  title as "Title",
  "user".login as "Created By",
  dashboard.created as "Created On"
FROM dashboard
INNER JOIN "user" on "user".id = dashboard.created_by
WHERE $__timeFilter(dashboard.created)
```

Copy

You can control the name of the Table panel columns by using regular `as` SQL column selection syntax.

The resulting table panel:

Time series queries

If you set Format as to *Time series*, then the query must have a column named time that returns either a SQL datetime or any numeric datatype representing Unix epoch in seconds. In addition, result sets of time series queries must be sorted by time for panels to properly visualize the result.

A time series query result is returned in a wide data frame format. Any column except time or of type string transforms into value fields in the data frame query result. Any string column transforms into field labels in the data frame query result.

For backward compatibility, there's an exception to the above rule for queries that return three columns including a string column named metric. Instead of transforming the metric column into field labels, it becomes the field name, and then the series name is formatted as the value of the metric column. See the example with the metric column below.

To optionally customize the default series name formatting, refer to Standard options definitions.

Example with `metric` column:

sql Copy

```sql
SELECT
  $__timeGroupAlias("time_date_time",'5m'),
  min("value_double"),
  'min' as metric
FROM test_data
WHERE $__timeFilter("time_date_time")
GROUP BY time
ORDER BY time
```

Data frame result:

text Copy

```
+---------------------+-----------------+
| Name: time          | Name: min       |
| Labels:             | Labels:         |
| Type: []time.Time   | Type: []float64 |
+---------------------+-----------------+
| 2020-01-02 03:05:00 | 3               |
| 2020-01-02 03:10:00 | 6               |
+---------------------+-----------------+
```

Example using the fill parameter in the $__timeGroupAlias macro to convert null values to be zero instead:

sql Copy

```sql
SELECT
  $__timeGroupAlias("createdAt",'5m',0),
  sum(value) as value,
  hostname
FROM test_data
```

```
WHERE
  $__timeFilter("createdAt")
GROUP BY time, hostname
ORDER BY time
```

Given the data frame result in the following example and using the graph panel, you will get two series named *value 10.0.1.1* and *value 10.0.1.2*. To render the series with a name of *10.0.1.1* and *10.0.1.2* , use a Standard options definitions display value of `${__field.labels.hostname}`.

Data frame result:

text Copy

```
+---------------------+---------------------------+---------------------------+
| Name: time          | Name: value               | Name: value               |
| Labels:             | Labels: hostname=10.0.1.1 | Labels: hostname=10.0.1.2 |
| Type: []time.Time   | Type: []float64           | Type: []float64           |
+---------------------+---------------------------+---------------------------+
| 2020-01-02 03:05:00 | 3                         | 4                         |
| 2020-01-02 03:10:00 | 6                         | 7                         |
+---------------------+---------------------------+---------------------------+
```

Example with multiple columns:

sql Copy

```
SELECT
  $__timeGroupAlias("time_date_time",'5m'),
  min("value_double") as "min_value",
  max("value_double") as "max_value"
FROM test_data
WHERE $__timeFilter("time_date_time")
GROUP BY time
ORDER BY time
```

Data frame result:

text Copy

```
+---------------------+----------------+----------------+
| Name: time          | Name: min_value | Name: max_value |
| Labels:             | Labels:         | Labels:         |
| Type: []time.Time   | Type: []float64 | Type: []float64 |
+---------------------+----------------+----------------+
| 2020-01-02 03:04:00 | 3               | 4               |
```

```
| 2020-01-02 03:05:00 | 6                | 7                |
+---------------------+------------------+------------------+
```

Templating

Instead of hard-coding things like server, application and sensor name in your metric queries you can use variables in their place. Variables are shown as dropdown select boxes at the top of the dashboard. These dropdowns make it easy to change the data being displayed in your dashboard.

Refer to Templates and variables for an introduction to the templating feature and the different types of template variables.

Query variable

If you add a template variable of the type `Query`, you can write a PostgreSQL query that can return things like measurement names, key names or key values that are shown as a dropdown select box.

For example, you can have a variable that contains all values for the `hostname` column in a table if you specify a query like this in the templating variable *Query* setting.

```sql
SELECT hostname FROM host
```

A query can return multiple columns and Grafana will automatically create a list from them. For example, the query below will return a list with values from `hostname` and `hostname2`.

```sql
SELECT host.hostname, other_host.hostname2 FROM host JOIN other_host ON host.city = other_host.cit
```

To use time range dependent macros like `$__timeFilter(column)` in your query the refresh mode of the template variable needs to be set to *On Time Range Change*.

```sql
SELECT event_name FROM event_log WHERE $__timeFilter(time_column)
```

Another option is a query that can create a key/value variable. The query should return two columns that are named `__text` and `__value`. The `__text` column value should be unique (if it is not unique then the first value is used). The options in the dropdown will have a text and value that allows you to have a friendly name as text and an id as the value. An example query with `hostname` as the text and `id` as the value:

```sql
```

```
SELECT hostname AS __text, id AS __value FROM host
```

You can also create nested variables. Using a variable named `region`, you could have the hosts variable only show hosts from the current selected region with a query like this (if `region` is a multi-value variable then use the `IN` comparison operator rather than `=` to match against multiple values):

sql Copy

```
SELECT hostname FROM host  WHERE region IN($region)
```

Using `__searchFilter` to filter results in Query Variable

Available from Grafana 6.5 and above

Using `__searchFilter` in the query field will filter the query result based on what the user types in the dropdown select box. When nothing has been entered by the user the default value for `__searchFilter` is `%`.

Important that you surround the `__searchFilter` expression with quotes as Grafana does not do this for you.

The example below shows how to use `__searchFilter` as part of the query field to enable searching for `hostname` while the user types in the dropdown select box.

Query

sql Copy

```
SELECT hostname FROM my_host  WHERE hostname LIKE '$__searchFilter'
```

Using Variables in Queries

From Grafana 4.3.0 to 4.6.0, template variables are always quoted automatically. If your template variables are strings, do not wrap them in quotes in where clauses.

From Grafana 4.7.0, template variable values are only quoted when the template variable is a `multi-value`.

If the variable is a multi-value variable then use the `IN` comparison operator rather than `=` to match against multiple values.

There are two syntaxes:

`$<varname>` Example with a template variable named `hostname`:

```sql
SELECT
  atimestamp as time,
  aint as value
FROM table
WHERE $__timeFilter(atimestamp) and hostname in($hostname)
ORDER BY atimestamp ASC
```

`[[varname]]` Example with a template variable named `hostname`:

```sql
SELECT
  atimestamp as time,
  aint as value
FROM table
WHERE $__timeFilter(atimestamp) and hostname in([[hostname]])
ORDER BY atimestamp ASC
```

Disabling quoting for multi-value variables

Grafana automatically creates a quoted, comma-separated string for multi-value variables. For example: if `server01` and `server02` are selected then it will be formatted as: `'server01', 'server02'`. To disable quoting, use the csv formatting option for variables:

`${servers:csv}`

Read more about variable formatting options in the Variables documentation.

Annotations

Annotations allow you to overlay rich event information on top of graphs. You add annotation queries via the Dashboard menu / Annotations view.

Example query using time column with epoch values:

```sql
SELECT
  epoch_time as time,
  metric1 as text,
```

```
  concat_ws(', ', metric1::text, metric2::text) as tags
FROM
  public.test_data
WHERE
  $__unixEpochFilter(epoch_time)
```

Example region query using time and timeend columns with epoch values:

Only available in Grafana v6.6+.

sql Copy

```sql
SELECT
  epoch_time as time,
  epoch_time_end as timeend,
  metric1 as text,
  concat_ws(', ', metric1::text, metric2::text) as tags
FROM
  public.test_data
WHERE
  $__unixEpochFilter(epoch_time)
```

Example query using time column of native SQL date/time data type:

sql Copy

```sql
SELECT
  native_date_time as time,
  metric1 as text,
  concat_ws(', ', metric1::text, metric2::text) as tags
FROM
  public.test_data
WHERE
  $__timeFilter(native_date_time)
```

Name	Description
`time`	The name of the date/time field. Could be a column with a native SQL date/time data type or epoch value.

Name	Description
`timeend`	Optional name of the end date/time field. Could be a column with a native SQL date/time data type or epoch value. (Grafana v6.6+)
`text`	Event description field.
`tags`	Optional field name to use for event tags as a comma separated string.

Alerting

Time series queries should work in alerting conditions. Table formatted queries are not yet supported in alert rule conditions.

Prometheus data source

Prometheus is an open-source database that uses a telemetry collector agent to scrape and store metrics used for monitoring and alerting. If you are just getting started with Prometheus, see What is Prometheus?.

Grafana provides native support for Prometheus. For instructions on downloading Prometheus see Get started with Grafana and Prometheus.

For instructions on how to add a data source to Grafana, refer to the administration documentation. Only users with the organization `administrator` role can add data sources and edit existing data sources. Administrators can also configure the data source via YAML with Grafana's provisioning system.

Once you've added the Prometheus data source, you can configure it so that your Grafana instance's users can create queries in its query editor when they build dashboards, use Explore, and [annotate visualizations][annotate visualizations].

The following guides will help you get started with the Prometheus data source:

- Configure the Prometheus data source
- Prometheus query editor
- Template variables

Prometheus API

The Prometheus data source also works with other projects that implement the Prometheus querying API.

For more information on how to query other Prometheus-compatible projects from Grafana, refer to the specific project's documentation:

- Grafana Mimir

- Thanos

Provision the data source

You can define and configure the data source in YAML files as part of Grafana's provisioning system. For more information about provisioning, and for available configuration options, refer to Provisioning Grafana.

> **NOTE**
>
> Once you have provisioned a data source you cannot edit it.

Provisioning example

yaml Copy

```yaml
apiVersion: 1

datasources:
  - name: Prometheus
    type: prometheus
    access: proxy
    # Access mode - proxy (server in the UI) or direct (browser in the UI).
    url: http://localhost:9090
    jsonData:
      httpMethod: POST
      manageAlerts: true
      prometheusType: Prometheus
      prometheusVersion: 2.44.0
      cacheLevel: 'High'
      disableRecordingRules: false
      incrementalQueryOverlapWindow: 10m
      exemplarTraceIdDestinations:
        # Field with internal link pointing to data source in Grafana.
        # datasourceUid value can be anything, but it should be unique across all defined data so
```

Expand code

View Grafana metrics with Prometheus

Grafana exposes metrics for Prometheus on the `/metrics` endpoint. We also bundle a dashboard within Grafana so you can start viewing your metrics faster.

To import the bundled dashboard:

1. Navigate to the data source's configuration page.
2. Select the **Dashboards** tab.

This displays dashboards for Grafana and Prometheus.

1. Select **Import** for the dashboard to import.

For details about these metrics, refer to Internal Grafana metrics.

Amazon Managed Service for Prometheus

The Prometheus data source works with Amazon Managed Service for Prometheus.

If you use an AWS Identity and Access Management (IAM) policy to control access to your Amazon Elasticsearch Service domain, you must use AWS Signature Version 4 (AWS SigV4) to sign all requests to that domain.

For details on AWS SigV4, refer to the AWS documentation.

AWS Signature Version 4 authentication

> **NOTE**
>
> Available in Grafana v7.3.5 and higher.

To connect the Prometheus data source to Amazon Managed Service for Prometheus using SigV4 authentication, refer to the AWS guide to Set up Grafana open source or Grafana Enterprise for use with AMP.

If you run Grafana in an Amazon EKS cluster, follow the AWS guide to Query using Grafana running in an Amazon EKS cluster.

Azure authentication settings

The Prometheus data source works with Azure authentication. To configure Azure authentication see Configure Azure Active Directory (AD) authentication.

In Grafana Enterprise, update the .ini configuration file: Configure Grafana. Depending on your setup, the .ini file is located here. Add the following setting in the **[auth]** section :

bash Copy

```
[auth]
azure_auth_enabled = true
```

> **NOTE**

If you are using Azure authentication settings do not enable `Forward OAuth identity`. Both use the same HTTP authorization headers. Azure settings will get overwritten by the Oauth token.

Exemplars

Exemplars associate higher-cardinality metadata from a specific event with traditional time series data. See Introduction to exemplars in Prometheus documentation for detailed information on how they work.

> **NOTE**
>
> Available in Prometheus v2.26 and higher with Grafana v7.4 and higher.

Grafana 7.4 and higher can show exemplars data alongside a metric both in Explore and in Dashboards.

Screenshot showing the detail window of an Exemplar

See the Exemplars section in Configure Prometheus data source.

Exemplars

Incremental dashboard queries (beta)

As of Grafana 10, the Prometheus data source can be configured to query live dashboards incrementally, instead of re-querying the entire duration on each dashboard refresh.

This can be toggled on or off in the data source configuration or provisioning file (under `incrementalQuerying` in jsonData). Additionally, the amount of overlap between incremental queries can be configured using the `incrementalQueryOverlapWindow` jsonData field, the default value is `10m` (10 minutes).

Increasing the duration of the `incrementalQueryOverlapWindow` will increase the size of every incremental query, but might be helpful for instances that have inconsistent results for recent data.

Recording Rules (beta)

The Prometheus data source can be configured to disable recording rules under the data source configuration or provisioning file (under `disableRecordingRules` in jsonData).

Configure Prometheus

Grafana ships with built-in support for Prometheus. If you are new to Prometheus the following documentation will help you get started working with Prometheus and Grafana:

- What is Prometheus?
- Prometheus data model
- Getting started

Configure the data source

To add the Prometheus data source, complete the following steps:

1. Click **Connections** in the left-side menu.
2. Under **Connections**, click **Add new connection**.
3. Enter `Prometheus` in the search bar.
4. Select **Prometheus data source**.
5. Click **Create a Prometheus data source** in the upper right.

You will be taken to the **Settings** tab where you will set up your Prometheus configuration.

Configuration options

The following is a list of configuration options for Prometheus.

The first option to configure is the name of your connection:

- **Name** - The data source name. This is how you refer to the data source in panels and queries. Examples: prometheus-1, prom-metrics.

- **Default** - Toggle to select as the default name in dashboard panels. When you go to a dashboard panel this will be the default selected data source.

HTTP section

- **URL** - The URL of your Prometheus server. If your Prometheus server is local, use `<http://localhost:9090>`. If it is on a server within a network, this is the URL with port where you are running Prometheus. Example: `<http://prometheus.example.orgname:9090>`.
- **Allowed cookies** - Specify cookies by name that should be forwarded to the data source. The Grafana proxy deletes all forwarded cookies by default.
- **Timeout** - The HTTP request timeout. This must be in seconds. There is no default, so this setting is up to you.

Auth section

There are several authentication methods you can choose in the Authentication section.

> **NOTE**
>
> Use TLS (Transport Layer Security) for an additional layer of security when working with Prometheus. For information on setting up TLS encryption with Prometheus see Securing Prometheus API and UI Endpoints Using TLS Encryption. You must add TLS settings to your Prometheus configuration file **prior** to setting these options in Grafana.

- **Basic authentication** - The most common authentication method. Use your `data source` user name and `data source` password to connect.
- **With credentials** - Toggle on to enable credentials such as cookies or auth headers to be sent with cross-site requests.
- **TLS client authentication** - Toggle on to use client authentication. When enabled, add the `Server name`, `Client cert` and `Client key`. The client provides a certificate that is validated by the server to establish the client's trusted identity. The client key encrypts the data between client and server.
- **With CA cert** - Authenticate with a CA certificate. Follow the instructions of the CA (Certificate Authority) to download the certificate file.
- **Skip TLS verify** - Toggle on to bypass TLS certificate validation.
- **Forward OAuth identity** - Forward the OAuth access token (and also the OIDC ID token if available) of the user querying the data source.

Custom HTTP headers

- **Header** - Add a custom header. This allows custom headers to be passed based on the needs of your Prometheus instance.
- **Value** - The value of the header.

Additional settings

Following are additional configuration options.

Alerting

- **Manage alerts via Alerting UI** - Toggle to enable `Alertmanager` integration for this data source.

Interval behavior

- **Scrape interval** - Set this to the typical scrape and evaluation interval configured in Prometheus. The default is `15s`.
- **Query timeout** - The default is `60s`.

Query editor

- **Default editor** - Sets a default editor. Options are `Builder` or `Code`. For information on query editor types see Prometheus query editor.
- **Disable metrics lookup** - Toggle on to disable the metrics chooser and metric/label support in the query field's autocomplete. This helps if you have performance issues with large Prometheus instances.

Performance

- **Prometheus type** - The type of your Prometheus server. There are four options: `Prometheus`, `Cortex`, `Thanos`, `Mimir`.
- **Version** Select the version you are using. Once the Prometheus type has been selected, a list of versions auto-populates using the Prometheus buildinfo API. The `Cortex` Prometheus type does not support this API so you will need to manually add the version.
- **Incremental querying (beta)** - Changes the default behavior of relative queries to always request fresh data from the Prometheus instance. Enable this option to decrease database and network load.

Other

- **Custom query parameters** - Add custom parameters to the Prometheus query URL. For example `timeout`, `partial_response`, `dedup`, or `max_source_resolution`. Multiple parameters should be concatenated together with an '&'.
- **HTTP method** - Use either `POST` or `GET` HTTP method to query your data source. `POST` is the recommended and pre-selected method as it allows bigger queries. Change to `GET` if you have a Prometheus version older than 2.1 or if `POST` requests are restricted in your network.

Exemplars

Support for exemplars is available only for the Prometheus data source. If this is your first time working with exemplars see Introduction to exemplars. An exemplar is a specific trace representative of measurement taken in a given time interval.

- **Internal link** - Toggle on to enable an internal link. When enabled, reveals the data source selector. Select the backend tracing data store for your exemplar data.
- **URL** - *(Visible if you **disable** `Internal link`)* Defines the external link's URL trace backend. You can interpolate the value from the field by using the `${__value.raw}` macro.
- **Data source** - *(Visible if you **enable** `Internal link`)* The data source the exemplar will navigate to.
- **URL label** - Adds a custom display label to override the value of the `Label name` field.
- **Label name** - The name of the field in the `labels` object used to obtain the traceID property.
- **Remove exemplar link** - Click to remove existing links.

Prometheus query editor

Grafana provides a query editor for the Prometheus data source to create queries in PromQL. For more information about PromQL, see Querying Prometheus.

For general documentation on querying data sources in Grafana, see Query and transform data.

For options and functions common to all query editors, see Query editors.

Choose a query editing mode

The Prometheus query editor has two modes:

- Builder mode
- Code mode

Each mode is explained in greater detail below.

Query editor mode

Both modes are synchronized, so you can switch between them. However, if there is an issue with the query while switching modes, a warning message will appear.

Toolbar elements

The query editor toolbar contains the following elements:

- **Kick start your query** - Click to see a list of operation patterns that help you quickly get started adding multiple operations to your query. These include:
  - Rate query starters
  - Histogram query starters
  - Binary query starters

Click the arrow next to each to see available options to add to your query.

- **Explain** - Toggle to display a step-by-step explanation of all query components and operations.

Explain results

- **Builder/Code** - Click the corresponding **Builder** or **Code** tab on the toolbar to select a editor mode.

Configure common options

You can configure Prometheus-specific options in the query editor by setting several options regardless of mode.

Options

Legend

The **Legend** setting defines the time series's name. You can use a predefined or custom format.

- **Auto** - Displays unique labels. Also displays all overlapping labels if a series has multiple labels.
- **Verbose** - Displays all label names.
- **Custom** - Uses templating to select which labels will be included. For example, `{{hostname}}` is replaced by the label value for the label `hostname`. Clear the input and click outside of it to select another mode.

Min step

The **Min step** setting defines the lower bounds on the interval between data points. For example, set this to `1h` to hint that measurements are taken hourly. This setting supports the `$__interval` and `$__rate_interval` macros. Be aware that the query range dates are aligned to the step and this can change the start and end of the range.

Format

Switch between the following format options:

- **Time series** - The default time series format. See Time series kind formats for information on time series data frames and how time and value fields are structured.

- **Table** - This works only in a Table panel.
- **Heatmap** - Displays metrics of the Histogram type on a Heatmap panel by converting cumulative histograms to regular ones and sorting the series by the bucket bound.

Type

The **Type** setting sets the query type. These include:

- **Both** - The default option. Returns results for both a **Range** query and an **Instant** query.
- **Range** - Returns a range vector consisting of a set of time series data containing a range of data points over time for each time series. You can choose lines, bars, points, stacked lines or stacked bars
- **Instant** - Returns one data point per query and only the most recent point in the time range provided. The results can be shown in table format or as raw data. To depict instant query results in the time series panel, first add a field override, next add a property to the override named `Transform`, and finally select `Constant` from the **Transform** dropdown.

For more information, refer to the Time Series Transform option documentation.

> **NOTE**
>
> Grafana modifies the request dates for queries to align them with the dynamically calculated step. This ensures a consistent display of metrics data and Prometheus requires this for caching results. But, aligning the range with the step can result in a small gap of data at the right edge of a graph or change the start date of the range. For example, a 15s step aligns the range to Unix time divisible by 15s and a 1w minstep aligns the range to the start of the week on a Thursday.

Exemplars

Toggle **Exemplars** to run a query that includes exemplars in the graph. Exemplars are unique to Prometheus. For more information see Introduction to exemplars.

> **NOTE**
>
> There is no option to add exemplars with an **Instant** query type.

Inspector

Click **Inspector** to get detailed statistics regarding your query. Inspector functions as a kind of debugging tool that “inspects” your query. It provides query statistics under **Stats**, request response time under **Query**, data frame details under **{} JSON**, and the shape of your data under **Data**.

Inspector

Builder mode

Builder mode helps you build queries using a visual interface. This option is best for users who have limited or no previous experience working with Prometheus and PromQL.

This video demonstrates how to use the visual Prometheus query builder available in Grafana v9.0:

Metrics

Metric and label filters

When you are ready to create a query, you can choose the specific metric name from the dropdown list under **Metric**. The data source requests the list of available metrics from the Prometheus server based on the selected time rage. You can also enter text into the selector when the dropdown is open to search and filter the list.

Metrics explorer

Metrics explorer

If you would like to explore your metrics in the query builder further, you can open the **Metrics Explorer** by clicking the first option in the metric select component of the query builder.

The metrics explorer is different than the metrics browser. The metrics explorer is only found in the query builder section. The metrics browser is only found in the code editor. The metrics explorer does not have the ability to browse labels yet, but the metrics browser can display all labels on a metric name.

The metrics explorer displays all metrics in a paginated table list. The list shows the total number of metrics, as well as the name, type and description for each metric. You can enter text into the search input to filter results. You can also filter by type.

There are also additional settings for the following items:

- Include description in search. Search by name **and** description
- Include results with no metadata. Many Prometheus metrics have no metadata. This allows users to include metrics with undefined type and description.
- Disable text wrap.
- Enable regex search. This uses the Prometheus API to enable regex search for the metric name.

Label filters

Select desired labels and their values from the dropdown list. When a metric is selected, the data source requests available labels and their values from the server. Use the `+` button to add a label, and the `x` button to remove a label.

Operations

Operations

Select the **+ Operations** button to add operations to your query.

The query editor groups operations into the following sections:

- Aggregations - for additional information see Aggregation operators.
- Range functions - for additional information see Functions.
- Functions - for additional information see Functions.
- Binary operations - for additional information see Binary operators.
- Trigonometric - for additional information see Trigonometric functions.
- Time functions - for additional information see Functions.

All operations have function parameters under the operation header. Click the `operator` to see a full list of supported functions. Some operations allow you to apply specific labels to functions.

Functions and labels

Some operations make sense only when used in a specific order. If adding an operation would result in nonsensical query, the query editor adds the operation to the correct place.

Hints

Hint

The query editor can detect which operations are most appropriate for some selected metrics. If it does, it displays a hint next to the **+ Operations** button.

To add the operation to your query, click the **Hint**.

Once you are satisfied with your query, click **Run query**.

Code mode

Code mode is for the experienced Prometheus user with prior expertise in PromQL, Prometheus' query language. The Code mode editor allows you to create queries just as you would in Prometheus. For more information about PromQL see Querying Prometheus.

Code mode

The user interface (UI) also lets you select metrics, labels, filters and operations.

You can write complex queries using the text editor with autocompletion features and syntax highlighting. It also contains a Metrics browser to further help you write queries.

Use autocomplete

Code mode's autocomplete feature works automatically while typing. The query editor can autocomplete static functions, aggregations, keywords, and also dynamic items like metrics and labels. The autocompletion dropdown includes documentation for the suggested items where available.

Metrics browser

The metrics browser locates metrics and selects relevant labels to help you build basic queries. When you click **Metrics browser** in `Code` mode, it displays all available metrics and labels. If supported by your Prometheus instance, each metric also displays its `HELP` and `TYPE` as a tooltip.

Metrics browser

When you select a metric under Step 1, the browser narrows down the available labels to show only the ones applicable to the metric. You can then select one or more labels shown in Step 2. Select one or more values in Step 3 for each label to tighten your query scope. In Step 4, you can select **Use query** to run the query, **Use as rate query** to add the rate operation to your query (`$__rate_interval`), **Validate selector** to verify the selector is valid and show the number of series found, or **Clear** to clear your selections and start over.

> **NOTE**

> If you do not remember a metric name, you can also select a few labels to narrow down the list, then find relevant label values.

All lists in the metrics browser have a search field above them to quickly filter for metrics or labels that match a certain string. The values section has only one search field, and its filtering applies to all labels to help you find values across labels once selected.

For example, among your labels `app`, `job`, `job_name` only one might have the value you are looking for.

Once you are satisfied with your query, click **Run query**.

Prometheus template variables

Instead of hard-coding details such as server, application, and sensor names in metric queries, you can use variables. Grafana refers to such variables as **template** variables. Grafana lists these variables in dropdown select boxes at the top of the dashboard to help you change the data displayed in your dashboard.

For an introduction to templating and template variables, see Templating and Add and manage variables.

Use query variables

You have the option to use several different variable types, but variables of the type `Query` will query Prometheus for a list of metrics, labels, label values, a query result or a series.

Select a Prometheus data source query type and enter the required inputs:

Query Type	Input(* required)	Description	Used API endpoints
`Label names`	`metric`	Returns a list of all label names matching the specified `metric` regex.	/api/v1/labels
`Label values`	`label` *, `metric`	Returns a list of label values for the `label` in all metrics or the optional metric.	/api/v1/label/ `label` /values or /api/v1/series
`Metrics`	`metric`	Returns a list of metrics matching the specified `metric` regex.	/api/v1/label/__name__/values
`Query result`	`query`	Returns a list of Prometheus query result for the `query`.	/api/v1/query

Query Type	Input(* required)	Description	Used API endpoints
`Series query`	`metric`, `label` or both	Returns a list of time series associated with the entered data.	/api/v1/series
`Classic query`	classic query string	Deprecated, classic version of variable query editor. Enter a string with the query type using a syntax like the following: `label_values(<metric>, <label>)`	all

For details on *metric names*, *label names*, and *label values*, refer to the Prometheus documentation.

Query options

Under the query variable type, you can set the following query options:

Option	Description
Data source	Select your data source from the dropdown list.
Select query type	Options are `default`, `value` and `metric name`. Each query type hits a different Prometheus endpoint.
Regex	Optional, if you want to extract part of a series name or metric node segment.
Sort	Default is `disabled`. Options include `alphabetical`, `numerical` and `alphabetical case-sensitive`.
Refresh	When to update the values for the variable. Options are `On dashboard load` and `On time range change`.

Selection options

The following selection options are available:

- **Multi-value** - Check this option to enable multiple values to be selected at the same time.
- **Include All option** - Check this option to include all variables.

Use interval and range variables

You can use some global built-in variables in query variables, for example, `$__interval`, `$__interval_ms`, `$__range`, `$__range_s` and `$__range_ms`. For details, see Global built-in variables. The `label_values` function doesn't support queries, so you can use these variables in conjunction with the `query_result` function to filter variable queries.

Make sure to set the variable's `refresh` trigger to be `On Time Range Change` to get the correct instances when changing the time range on the dashboard.

Example:

Populate a variable with the busiest 5 request instances based on average QPS over the time range shown in the dashboard:

```
Query: query_result(topk(5, sum(rate(http_requests_total[$__range])) by (instance)))
Regex: /"([^"]+)"/
```

Copy

Populate a variable with the instances having a certain state over the time range shown in the dashboard, using `$__range_s`:

```
Query: query_result(max_over_time(<metric>[${__range_s}s]) != <state>)
Regex:
```

Copy

Use `$__rate_interval`

> **NOTE**
>
> Available in Grafana v7.2 and higher.

We recommend using `$__rate_interval` in the `rate` and `increase` functions instead of `$__interval` or a fixed interval value. Because `$__rate_interval` is always at least four times the value of the Scrape interval, it avoid problems specific to Prometheus.

For example, instead of using:

```
rate(http_requests_total[5m])
```

Copy

or:

```
rate(http_requests_total[$__interval])
```

Copy

We recommend that you use:

```
rate(http_requests_total[$__rate_interval])
```

Copy

The value of `$__rate_interval` is defined as *max(`$__interval` + Scrape interval, 4 * Scrape interval)*, where *Scrape interval* is the "Min step" setting (also known as `query*interval`, a setting per PromQL query) if any is set. Otherwise, Grafana uses the Prometheus data source's "Scrape interval" setting.

The “Min interval” setting in the panel is modified by the resolution setting, and therefore doesn’t have any effect on *Scrape interval*.

For details, refer to the Grafana blog.

Choose a variable syntax

The Prometheus data source supports two variable syntaxes for use in the **Query** field:

- `$<varname>`, for example `rate(http_requests_total{job=~"$job"}[$_rate_interval])`, which is easier to read and write but does not allow you to use a variable in the middle of a word.
- `[[varname]]`, for example `rate(http_requests_total{job=~"[[job]]"}[$_rate_interval])`

If you’ve enabled the *Multi-value* or *Include all value* options, Grafana converts the labels from plain text to a regex-compatible string, which requires you to use `=~` instead of `=`.

Use the ad hoc filters variable type

Prometheus supports the special ad hoc filters variable type, which you can use to specify any number of label/value filters on the fly. These filters are automatically applied to all your Prometheus queries.

Tempo data source

Grafana ships with built-in support for Tempo, a high-volume, minimal-dependency trace storage, open source tracing solution from Grafana Labs. This topic explains configuration and queries specific to the Tempo data source.

For instructions on how to add a data source to Grafana, refer to the administration documentation. Only users with the organization administrator role can add data sources. Administrators can also configure the data source via YAML with Grafana's provisioning system.

This video explains how to add data sources, including Loki, Tempo, and Mimir, to Grafana and Grafana Cloud. Tempo data source set up starts at 4:58 in the video.

Once you've added the data source, you can configure it so that your Grafana instance's users can create queries in its query editor when they build dashboards and use Explore.

- Configure the Tempo data source
  Guide for configuring Tempo in Grafana
- Best practices for tracing
  Use best practices to plan how you implement tracing.
- Query tracing data
  Guide for using the Tempo data source's query editor
- Upload a JSON trace file
  Upload a JSON trace file to the Tempo data source
- Service Graph and Service Graph view
  Use the Service Graph and Service Graph view
- Span filters
  Use span filters to filter spans in the timeline viewer
- Link to a trace ID
  Link to trace IDs from logs and metrics

Configure the Tempo data source

To configure basic settings for the Tempo data source, complete the following steps:

1. Click **Connections** in the left-side menu.
2. Under Your connections, click **Data sources**.
3. Enter `Tempo` in the search bar.
4. Select **Tempo**.
5. On the **Settings** tab, set the data source's basic configuration options:

Name	Description
Name	Sets the name you use to refer to the data source in panels and queries.
Default	Sets the data source that's pre-selected for new panels.
URL	Sets the URL of the Tempo instance, such as `http://tempo`.
Basic Auth	Enables basic authentication to the Tempo data source.
User	Sets the user name for basic authentication.
Password	Sets the password for basic authentication.

You can also configure settings specific to the Tempo data source.

This video explains how to add data sources, including Loki, Tempo, and Mimir, to Grafana and Grafana Cloud. Tempo data source set up starts at 4:58 in the video.

Trace to logs

The **Trace to logs** setting configures the trace to logs feature that is available when you integrate Grafana with Tempo.

There are two ways to configure the trace to logs feature:

- Use a simplified configuration with default query, or
- Configure a custom query where you can use a template language to interpolate variables from the trace or span.

Use a simple configuration

1. Select the target data source from the drop-down list.

   You can also click **Open advanced data source picker** to see more options, including adding a data source.

2. Set start and end time shift. As the logs timestamps may not exactly match the timestamps of the spans in trace it may be necessary to search in larger or shifted time range to find the desired logs.

3. Select which tags to use in the logs query. The tags you configure must be present in the span's attributes or resources for a trace to logs span link to appear. You can optionally configure a new name for the tag. This is useful, for example, if the tag has dots in the name and the target data source does not allow using dots in labels. In that case, you can for example remap `http.status` (the span attribute) to `http_status` (the data source field). "Data source" in this context can refer to Loki, or another log data source.

4. Optionally switch on the **Filter by trace ID** and/or **Filter by span ID** setting to further filter the logs if your logs consistently contain trace or span IDs.

Configure a custom query

1. Select the target data source from the drop-down list.

   You can also click **Open advanced data source picker** to see more options, including adding a data source.

2. Set start and end time shift. As the logs timestamps may not exactly match the timestamps of the spans in the trace it may be necessary to widen or shift the time range to find the desired logs.

3. Optional: Select tags to map. These tags can be used in the custom query with `${__tags}` variable. This variable interpolates the mapped tags as list in an appropriate syntax for the data source. Only the tags that were present in the span are included; tags that aren't present are omitted You can also configure a new name for the tag. This is useful in cases where the tag has dots in the name and the target data source doesn't allow dots in labels. For example, you can remap `http.status` to `http_status`. If you don't map any tags here, you can still use any tag in the query, for example, `method="${__span.tags.method}"`. You can learn more about custom query variables here.

4. Skip **Filter by trace ID** and **Filter by span ID** settings as these cannot be used with a custom query.
5. Switch on **Use custom query**.
6. Specify a custom query to be used to query the logs. You can use various variables to make that query relevant for current span. The link will only be shown only if all the variables are interpolated with non-empty values to prevent creating an invalid query.

Configure trace to logs

The following table describes the ways in which you can configure your trace to logs settings:

Setting name	Description
Data source	Defines the target data source. You can select Loki or any compatible log store.
Span start time shift	Shifts the start time for the logs query, based on the span's start time. You can use time units, such as `5s`, `1m`, `3h`. To extend the time to the past, use a negative value. Default: `0`.
Span end time shift	Shifts the end time for the logs query, based on the span's end time. You can use time units. Default: `0`.
Tags	Defines the tags to use in the logs query. Default: `cluster`, `hostname`, `namespace`, `pod`, `service.name`, `service.namespace`. You can change the tag name for example to remove dots from the name if they are not allowed in the target data source. For example, map `http.status` to `http_status`.
Filter by trace ID	Toggles whether to append the trace ID to the logs query.
Filter by span ID	Toggles whether to append the span ID to the logs query.
Use custom query	Toggles use of custom query with interpolation.
Query	Input to write custom query. Use variable interpolation to customize it with variables from span.

Trace to metrics

> **NOTE**
>
> This feature is behind the `traceToMetrics` feature toggle. If you use Grafana Cloud, open a support ticket in the Cloud Portal to access this feature.

The **Trace to metrics** setting configures the trace to metrics feature available when integrating Grafana with Tempo.

There are two ways to configure the trace to metrics feature:

- Use a basic configuration with a default query, or
- Configure one or more custom queries where you can use a template language to interpolate variables from the trace or span.

Simple config

To use a simple configuration, follow these steps:

1. Select a metrics data source from the **Data source** drop-down.
2. Optional: Choose any tags to use in the query. If left blank, the default values of `cluster`, `hostname`, `namespace`, `pod`, `service.name` and `service.namespace` are used.

   The tags you configure must be present in the spans attributes or resources for a trace to metrics span link to appear. You can optionally configure a new name for the tag. This is useful for example if the tag has dots in the name and the target data source doesn't allow using dots in labels. In that case you can for example remap `service.name` to `service_name`.
3. Do not select **Add query**.
4. Select **Save and Test**.

Custom queries

To use custom queriess with the configuration, follow these steps:

1. Select a metrics data source from the **Data source** drop-down.
2. Optional: Choose any tags to use in the query. If left blank, the default values of `cluster`, `hostname`, `namespace`, `pod`, `service.name` and `service.namespace` are used.

   These tags can be used in the custom query with `${__tags}` variable. This variable interpolates the mapped tags as list in an appropriate syntax for the data source and will only include the tags that were present in the span omitting those that weren't present. You can optionally configure a new name for the tag. This is useful in cases where the tag has dots in the name and the target data source doesn't allow using dots in labels. For example, you can remap `service.name` to `service_name` in such a case. If you don't map any tags here, you can still use any tag in the query like this `method="${__span.tags.method}"`. You can learn more about custom query variables here.
3. Click **Add query** to add a custom query.
4. Specify a custom query to be used to query metrics data.

   Each linked query consists of:

   - **Link Label:** *(Optional)* Descriptive label for the linked query.
   - **Query:** The query ran when navigating from a trace to the metrics data source. Interpolate tags using the `$__tags` keyword. For example, when you configure the query `requests_total{$__tags}` with the tags `k8s.pod=pod` and `cluster`, the result looks like `requests_total{pod="nginx-554b9", cluster="us-east-1"}`.
5. Select **Save and Test**.

Configure trace to metrics

Setting name	Description
Data source	Defines the target data source.
Span start time shift	Shifts the start time for the metrics query, based on the span's start time. You can use time units, such as `5s`, `1m`, `3h`. To extend the time to the past, use a negative value. Default: `0`.
Span end time shift	Shifts the end time for the metrics query, based on the span's end time. You can use time units. Default: `0`.
Tags	Defines the tags used in linked queries. The key sets the span attribute name, and the optional value sets the corresponding metric label name. For example, you can map `k8s.pod` to `pod`. To interpolate these tags into queries, use the `$__tags` keyword.
Link Label	*(Optional)* Descriptive label for the linked query.

Setting name	Description
Query	Input to write a custom query. Use variable interpolation to customize it with variables from span.

Trace to profiles

Using Trace to profiles, you can use Grafana's ability to correlate different signals by adding the functionality to link between traces and profiles.

Trace to profiles lets you link your Grafana Pyroscope data source to tracing data. When configured, this connection lets you run queries from a trace span into the profile data.

There are two ways to configure the trace to profiles feature:

- Use a basic configuration with default query, or
- Configure a custom query where you can use a template language to interpolate variables from the trace or span.

> **NOTE**
>
> Traces to profile requires a Tempo data source with Traces to profiles configured and a Pyroscope data source.
>
> As with traces, your application needs to be instrumented to emit profiling data. For more information, refer to Linking tracing and profiling with span profiles.

To use trace to profiles, navigate to **Explore** and query a trace. Each span links to your queries. Clicking a link runs the query in a split panel. If tags are configured, Grafana dynamically inserts the span attribute values into the query. The query runs over the time range of the (span start time - 60) to (span end time + 60 seconds).

To use trace to profiles, you must have a configured Grafana Pyroscope data source. For more information, refer to the Grafana Pyroscope data source documentation.

Embedded flame graphs are also inserted into each span details section that has a linked profile. This lets you see resource consumption in a flame graph visualization for each span without having to navigate away from the current view. Hover over a particular block in the flame graph to see more details about the consumed resources.

Configuration options

The following table describes options for configuring your Trace to profiles settings:

Setting name	Description
Data source	Defines the target data source. You can select a Pyroscope [profiling] data source.
Tags	Defines the tags to use in the profile query. Default: `cluster`, `hostname`, `namespace`, `pod`, `service.name`, `service.namespace`. You can change the tag name for example to remove dots from the name if they're not allowed in the target data source. For example, map `http.status` to `http_status`.
Profile type	Defines the profile type that used in the query.

Setting name	Description
Use custom query	Toggles use of custom query with interpolation.
Query	Input to write custom query. Use variable interpolation to customize it with variables from span.

Use a basic configuration

To use a basic configuration, follow these steps:

1. Select a Pyroscope data source from the **Data source** drop-down.
2. Optional: Choose any tags to use in the query. If left blank, the default values of `service.name` and `service.namespace` are used.

   The tags you configure must be present in the spans attributes or resources for a trace-to-profiles span link to appear.

   You can optionally configure a new name for the tag. This is useful if the tag has dots in the name and the target data source doesn't allow dots in labels. In that case, you can remap `service.name` to `service_name`.
3. Select one or more profile types to use in the query. Select the drop-down and choose options from the menu.

   The profile type or app must be selected for the query to be valid. Grafana doesn't show any data if the profile type or app isn't selected when a query runs.
4. Do not select **Use custom query**.
5. Select **Save and Test**.

If you have configured a Pyroscope data source and no profile data is available or the **Profiles for this span** button and the embedded flame graph is not visible, verify that the `pyroscope.profile.id` key-value pair exists in your span tags.

Configure a custom query

To use a custom query with the configuration, follow these steps:

1. Select a Pyroscope data source from the **Data source** drop-down.
2. Optional: Choose any tags to use in the query. If left blank, the default values of `service.name` and `service.namespace` are used.

   These tags can be used in the custom query with `${__tags}` variable. This variable interpolates the mapped tags as list in an appropriate syntax for the data source. Only the tags that were present in the span are included; tags that aren't present are omitted. You can also configure a new name for the tag. This is useful in cases where the tag has dots in the name and the target data source doesn't allow using dots in labels. For example, you can remap `service.name` to `service_name`. If you don't map any tags here, you can still use any tag in the query, for example: `method="${__span.tags.method}"`. You can learn more about custom query variables here.
3. Select one or more profile types to use in the query. Select the drop-down and choose options from the menu.
4. Switch on **Use custom query** to enter a custom query.
5. Specify a custom query to be used to query profile data. You can use various variables to make that query relevant for current span. The link is shown only if all the variables are interpolated with non-empty values to prevent creating an invalid query. You can interpolate the configured tags using the `$__tags` keyword.
6. Select **Save and Test**.

Configure trace to profiles

The following table describes options for configuring your trace to profiles settings:

Setting name	Description
Data source	Defines the target data source. You can currently select a Pyroscope [profiling] data source.
Tags	Defines the tags to use in the profile query. Default: `cluster`, `hostname`, `namespace`, `pod`, `service.name`, `service.namespace`. You can change the tag name for example to remove dots from the name if they are not allowed in the target data source. For example, map `http.status` to `http_status`.
Profile type	Defines the profile type that will be used in the query.
Use custom query	Toggles use of custom query with interpolation.
Query	Input to write custom query. Use variable interpolation to customize it with variables from span.

Custom query variables

To use a variable in your trace to logs, metrics or profiles you need to wrap it in `${}`. For example, `${__span.name}`.

Variable name	Description
__tags	This variable uses the tag mapping from the UI to create a label matcher string in the specific data source syntax. The variable only uses tags that are present in the span. The link is still created even if only one of those tags is present in the span. You can use this if all tags are not required for the query to be useful.
__span.spanId	The ID of the span.
__span.traceId	The ID of the trace.
__span.duration	The duration of the span.
__span.name	Name of the span.
__span.tags	Namespace for the tags in the span. To access a specific tag named `version`, you would use `${__span.tags.version}`. In case the tag contains dot, you have to access it as `${__span.tags["http.status"]}`.
__trace.traceId	The ID of the trace.
__trace.duration	The duration of the trace.
__trace.name	The name of the trace.

Service Graph

The **Service Graph** setting configures the Service Graph feature.

Configure the **Data source** setting to define in which Prometheus instance the Service Graph data is stored.

To use the Service Graph, refer to the Service Graph documentation.

Node Graph

The **Node Graph** setting enables the node graph visualization, which is disabled by default.

Once enabled, Grafana displays the node graph above the trace view.

Tempo search

The **Search** setting configures Tempo search.

You can configure the **Hide search** setting to hide the search query option in **Explore** if search is not configured in the Tempo instance.

Loki search

The **Loki search** setting configures the Loki search query type.

Configure the **Data source** setting to define which Loki instance you want to use to search traces. You must configure derived fields in the Loki instance.

TraceID query

The **TraceID query** setting modifies how TraceID queries are run. The time range can be used when there are performance issues or timeouts since it will narrow down the search to the defined range. This setting is disabled by default.

You can configure this setting as follows:

Name	Description
Enable time range	Use a time range in the TraceID query. Default: `disabled`.
Time shift start	Time shift for start of search. Default: `30m`.
Time shift end	Time shift for end of search. Default: `30m`.

Span bar

The **Span bar** setting helps you display additional information in the span bar row.

You can choose one of three options:

Name	Description
None	Adds nothing to the span bar row.
Duration	*(Default)* Displays the span duration on the span bar row.
Tag	Displays the span tag on the span bar row. You must also specify which tag key to use to get the tag value, such as `component`.

Provision the data source

You can define and configure the Tempo data source in YAML files as part of Grafana's provisioning system. For more information about provisioning and available configuration options, refer to

Provisioning Grafana.

Example provision YAML file:

yaml

Copy

```yaml
apiVersion: 1

datasources:
  - name: Tempo
    type: tempo
    uid: EbPG8fYoz
    url: http://localhost:3200
    access: proxy
    basicAuth: false
    jsonData:
      tracesToLogsV2:
        # Field with an internal link pointing to a logs data source in Grafana.
        # datasourceUid value must match the uid value of the logs data source.
        datasourceUid: 'loki'
        spanStartTimeShift: '-1h'
        spanEndTimeShift: '1h'
        tags: ['job', 'instance', 'pod', 'namespace']
        filterByTraceID: false
        filterBySpanID: false
```

Expand code

Best practices for tracing

This page provides some general best practices for tracing.

Span and resource attributes

Traces are built from spans, which denote units of work such as a call to, or from, an upstream service. Spans are constructed primarily of span and resource attributes. Spans also have a hierarchy, where parent spans can have children or siblings.

In the screenshot below, the left side of the screen (1) shows the list of results for the query. The right side (2) lists each span that makes up the selected trace.

A **span attribute** is a key/value pair that provides context for its span. For example, if the span deals with calling another service via HTTP, an attribute could include the HTTP URL (maybe as

the span attribute key `http.url`) and the HTTP status code returned (as the span attribute `http.status_code`). Span attributes can consist of varying, non-null types.

Unlike a span attribute, a **resource attribute** is a key/value pair that describes the context of how the span was collected. Generally, these attributes describe the process that created the span. This could be a set of resource attributes concerning a Kubernetes cluster, in which case you may see resource attributes, for example: `k8s.namespace`, `k8s.container_name`, and `k8s.cluster`. These can also include information on the libraries that were used to instrument the spans for a trace, or any other infrastructure information.

For more information, read the Attribute and Resource sections in the OpenTelemetry specification.

Naming conventions for span and resource attributes

The OpenTelemetry project defines a number of semantic conventions for attributes, which can help you to determine which attributes are most important to include in your spans. These conventions provide a common vocabulary for describing different types of entities, which can help to ensure that your data is consistent and meaningful.

When naming attributes, use consistent, nested namespaces to ensures that attribute keys are obvious to anyone observing the spans of a trace and that common attributes can be shared by spans. Using our example from above, the `http` prefix of the attribute is the namespace, and `url` and `status_code` are keys within that namespace. Attributes can also be nested, for example `http.url.protocol` might be `HTTP` or `HTTPS`, whereas `http.url.path` might be `/api/v1/query`.

For more details around semantic naming conventions, refer to the Recommendations for OpenTelemetry Authors and OpenTelemetry Semantic Conventions documentation.

Some third-party libraries provide auto-instrumentation that generate span and span attributes when included in a source base.

For more information about instrumenting your app for tracing, refer to the Instrument for distributed tracing documentation.

How many attributes should spans have?

The decision of how many attributes to include in your spans is up to you. There is no hard and fast rule. Keep the number of attributes to a minimum, as each attribute adds overhead to the tracing system. In Grafana Cloud, this results in higher tracing costs.

Only include attributes that are relevant to the operation that the span represents. For example, if you are tracing an HTTP request, you might include attributes such as the request method, the URL, and the response status code.

If you are unsure whether or not to include an attribute, it is always better to err on the side of caution and leave it out. You can always add additional attributes later if you need them.

In general, consider the following guidelines:

- Don’t include metrics or logs as attributes in your spans.
- Don’t use redundant attributes.

- When determining which attributes to add, consider an application's service flow, and execution in the context of only the current span.

The OpenTelemetry project doesn't specify a maximum number of attributes that a span can have. However, the default limits for the number of attributes per span is 128 entries, so you will have to adjust that. There are also default limits on attribute value and name character lengths.

Determining where to add spans

When instrumenting, determine the smallest piece of work that you need to observe in a trace to be of value to ensure that you don't over (or under) instrument.

Creating a new span for any work that has a relatively significant duration allows the observation of a trace to immediately show where significant amounts of time are spent during the processing of a request into your application or system.

For example, adding a span for a call to another services (either instrumented or not) may take an unknown amount of time to complete, and therefore being able to separate this work shows when services are taking longer than expected.

Adding a span for a piece of work that might call many other functions in a loop is a good signal of how long that loop is taking (you might add a span attribute that counts how many time the loop runs to determine if the duration is acceptable). However, adding a span for each method or function call in that loop might not, as it might produce hundreds or thousands of worthless spans.

Span length

While there are some (high) default limits to the length that a span (and by definition, the traces they belong to) can be, these can be adjusted by these configurations. Traces that include a large number of spans and/or long-running spans can have an impact on the time taken to query them once stored. Cloud Traces users should contact Grafana Support to modify overrides.

For long-running spans and traces, the best way to see this impact on requests is to send a few test cases and see what the performance looks like and to evaluate the trace size.

From there, you can modify the configuration for Tempo or determine ways to re-architect how the trace is produced.

You can consider breaking up the spans in several ways:

- Decompose the query
  - For example, if a complex SQL query involves multiple operations (for example, uses joins, subqueries, or unions), consider creating separate spans for each significant operation.
- Improve granulation of long-running spans
  - For long-running operations, you could create a new span for every predetermined interval of execution time.

> **NOTE**

This requires time-based tracking in your application's code and is more complex to implement.

- Use span linking
  - Should data flow hit bottlenecks where further operations on that data might be batched at a later time, the use of span links can help keep traces constrained to an acceptable time range, while sharing context with other traces that work on the same data. This can also improve the readability of traces.

Query tracing data

The Tempo data source's query editor helps you query and display traces from Tempo in Explore.

This topic explains configuration and queries specific to the Tempo data source. For general documentation on querying data sources in Grafana, see Query and transform data.

To add TraceQL panels to your dashboard, refer to the Traces panel documentation.

To learn more about Grafana dashboards, refer to the Use dashboards documentation.

Write TraceQL queries in Grafana

You can compose TraceQL queries in Grafana and Grafana Cloud using **Explore** and a Tempo data source. You can use either the **Query type** > **Search** (the TraceQL query builder) or the **TraceQL** tab (the TraceQL query editor). Both of these methods let you build queries and drill-down into result sets.

To learn more about how to query by TraceQL, refer to the TraceQL documentation.

TraceQL query builder

The TraceQL query builder, located on the **Explore** > **Query type** > **Search** in Grafana, provides drop-downs and text fields to help you write a query.

Refer to the Search using the TraceQL query builder documentation to learn more about creating queries using convenient drop-down menus.

TraceQL query editor

The TraceQL query editor, located on the **Explore** > **TraceQL** tab in Grafana, lets you search by trace ID and write TraceQL queries using autocomplete.

Refer to the TraceQL query editor documentation to learn more about constructing queries using a code-editor-like experience.

Query by search (deprecated)

CAUTION

Starting with Grafana v10.2, this query type has been deprecated. It will be removed in Grafana v10.3.

Use this to search for traces by service name, span name, duration range, or process-level attributes that are included in your application's instrumentation, such as HTTP status code and customer ID.

To configure Tempo and the Tempo data source for search, refer to Configure the data source.

To search for traces:

1. Select **Search** from the **Query** type selector.
2. Fill out the search form:

Name	Description
Service Name	Returns a list of services.
Span Name	Returns a list of span names.
Tags	Sets tags with values in the logfmt format, such as `error=true db.statement="select * from User"`.
Min Duration	Filters all traces with a duration higher than the set value. Possible values are `1.2s`, `100ms`, `500us`.
Max Duration	Filters all traces with a duration lower than the set value. Possible values are `1.2s`, `100ms`, `500us`.
Limit	Limits the number of traces returned.

Screenshot of the Tempo search feature with a trace rendered in the right panel

Search recent traces

You can search recent traces held in Tempo's ingesters. By default, ingesters store the last 15 minutes of tracing data.

To configure your Tempo data source to use this feature, refer to the Tempo documentation.

Search the backend datastore

Tempo includes the ability to search the entire backend datastore.

To configure your Tempo data source to use this feature, refer to the Tempo documentation.

Query by TraceID

To query a particular trace:

1. Select the **TraceQL** query type.
2. Enter the trace's ID into the query field.

Screenshot of the Tempo TraceID query type

Query Loki for traces

> **CAUTION**
>
> Starting with Grafana v11.0, the Loki query tab will no longer be available.

To find traces to visualize, you can use the Loki query editor. For results, you must configure derived fields in the Loki data source that point to this data source.

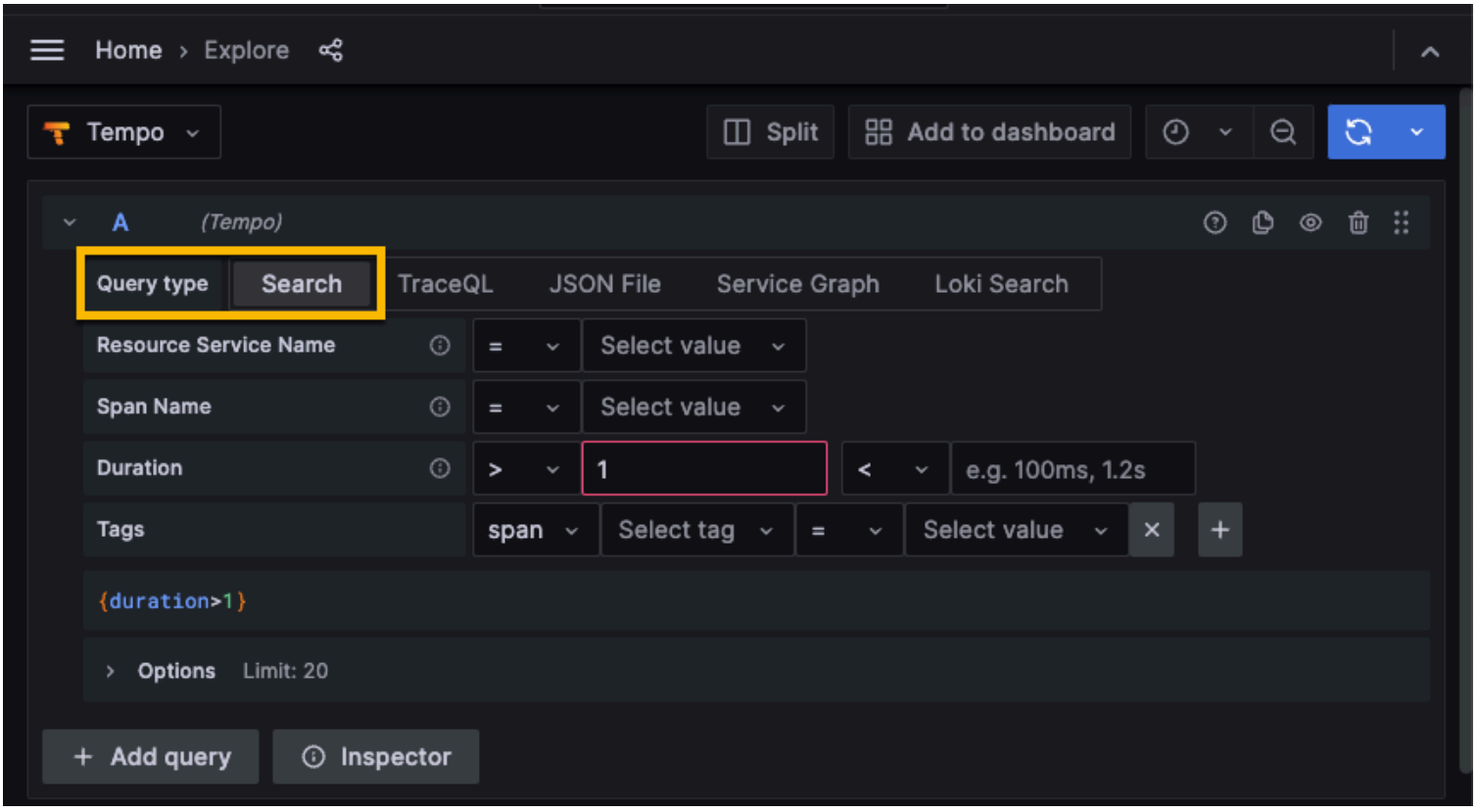
Screenshot of the Tempo query editor showing the Loki Search tab

Search traces using TraceQL query builder

Inspired by PromQL and LogQL, TraceQL is a query language designed for selecting traces. TraceQL provides a method for formulating precise queries so you can zoom in to the data you need. Query results are returned faster because the queries limit what is searched.

To learn more about how to query by TraceQL, refer to the TraceQL documentation.

The TraceQL query builder, located on the **Explore** > **Query type** > **Search** in Grafana, provides drop-downs and text fields to help you write a query.



Enable Search with the query builder

This feature is automatically available in Grafana 10 (and newer) and Grafana Cloud.

To enable the TraceQL query builder in self-hosted Grafana through version 10.1, enable the `traceqlSearch` feature toggle.

Write TraceQL queries using Search

The TraceQL query builder, located on the **Explore** > **Query type** > **Search** in Grafana, provides drop-downs and text fields to help you write a query.

The builder lets you run the most common queries in as few clicks as possible. You don't need to know the underlying query language or database architecture to use it.

The builder supports a subset of TraceQL capabilities. For example, if you wish to use structural operators (`>>`, `>`, `~`), you need to use the query editor on the **TraceQL** tab.

You can use the query builder's drop-downs to compose TraceQL queries. The selections you make automatically generate a TraceQL query.

To access **Search**, select your Tempo data source, and then choose **Explore** and select **Query type** > **Search**. You can use the query builder to search trace data by resource service name, span name, duration, and one or more tags. The examples on this page use the default filters.

In addition, you can add query builder blocks, view the query history, and use the **Inspector** to see details.

Screenshot of the Tempo Search query type

Perform a search

To perform a search, you need to select filters and/or tags and then run the query. The results appear underneath the query builder. The screenshot identifies the areas used to perform a search.

Parts of Tempo Search query type

Number	Name	Action	Comment
1	Data source	Use the data source drop-down to select a Tempo data source.	Each data source has its own version of search. This

Number	Name	Action	Comment
			Search is specific to the Tempo data source.
2	Query type tab	Select Search.	
3	Choose filter	Choose whether to filter using **Resource Service Name**, **Span Name**, and/or **Duration**.	Optional. You can execute an empty query in the Search tab. In TraceQL, `{}` is a valid query and is the default query until you choose options.
4	Filters conditions	Select options for one or more filters. For example, you can define a filter where **Resource Service Name** (`resource.service.name`) equals (`=`) `cloud-logs-archiver` .	Optional. At least one tag or filter must be defined.
5	Tags	Add tags for span, resource, or unscoped and define their conditions.	Optional.
6	TraceQL query	Displays the TraceQL query constructed by your selections.	This TraceQL query is executed when you select **Run query**.

Every query searches the data for the selected time frame. By default, queries run against data from the last hour. Select Time range to the left of Run query to choose the time range for the data your query runs against. Read the dashboard time range documentation to learn more.

To access Search, use the following steps:

1. Sign into Grafana.
2. Select your Tempo data source.
3. From the menu, choose **Explore** and select **Query type > Search**.

Define filters

Using filters, you refine the data returned from the query by selecting **Resource Service Name**, **Span Name**, or **Duration**. The **Duration** represents span time, calculated by subtracting the end time from the start time of the span.

Grafana administrators can change the default filters using the Tempo data source configuration. Filters can be limited by the operators. The available operators are determined by the field type. For example, **Span Name** and **Resource Service Name** are string fields so the comparison operators are equals (`=`), not equal (`!=`), or regular expressions (`=~`). **Duration** is a duration field type and uses range selections (`>` , `>=` , `<` , `<=`).

When you select multiple values for the same filter, Grafana automatically changes the operator to the regex operator `=~` and concatenates the values with a `|` . This capability only applies to fields with drop-down value selection.

For example, if you choose **Span Name** `= get` and then **Span Name** `= log_results_cache,` operator drop-down changes from `=` to `=~` and both `get` and `log_results_cache` are listed in the **Span Name** field. The resulting query is updated with this:

```
{duration>5ms && duration<10ms && name=~"get|log_results_cache"}
```

To define filters, follow these steps:

1. Choose one of the filters.
2. Select a comparison operator from the drop-down.
3. **Resource Service Name** and **Span Name** only: Select one or more values from the drop-down.
4. **Duration** only: Enter values and units for the range and choose comparison operators for the drop-downs. Units can be nanoseconds (`ns`), milliseconds (`ms`), seconds (`s`), minutes (`m`), and hours (`h`).

You can either select **Run query** to execute the query or define tags and then run the query.

Define tags

You can add any tags to your query to further filter the results. Tags can be selected by scoped (span or resource) or unscoped. If you select unscoped, then all tags are searched for matches.

To add a tag, follow these steps:

1. Select span, resource, or unscoped.
2. Select a tag from the **Select tag** drop-down.
3. Select a comparison operator.
4. Select a value from the **Select value** drop-down. This field is populated based upon the tag.
5. Optional: Select **+** to add an additional tag.

Optional: Use Aggregate by

> **WARNING**
>
> **Aggregate by** is an experimental feature. Engineering and on-call support is not available. Documentation is either limited or not provided outside of code comments. No SLA is provided. Enable the `metricsSummary` feature toggle in Grafana to use this feature. Your Grafana Tempo data source must also point to a Tempo database with the Metrics Summary API enabled. Contact Grafana Support to enable this feature in Grafana Cloud.

Using **Aggregate by**, you can calculate RED metrics (total span count, percent erroring spans, and latency information) for spans of `kind=server` that match your filter criteria, grouped by one or more attributes. This capability is based on the metrics summary API. Metrics summary only calculates summaries based on spans received within the last hour. For additional information, refer to Traces to metrics: Ad-hoc RED metrics in Grafana Tempo with `Aggregate by`.

When you use **Aggregate by**, the selections you make determine how the information is reported in the Table. Every combination that matches selections in your data is listed in the table. Each aggregate value, for example `intrinsic`:`name`, has a corresponding column in the results table.

For example, **names** matching `GET /:endpoint` with a **span.http.user_agent** of `k6/0.46` appeared in 31,466 spans. Instead of being listed by traces and associated spans, the query results are grouped by the selections in **Aggregate by**.

The RED metrics are calculated for every name and user agent combination found in your data.

The screenshot shows all of the successful HTTP `status_code` API calls against the `mystical-server` service. The results are shown in the same order used in **Aggregate by**. For example, **Aggregate by** lists `intrinsic.name` followed by `span.http.user_agent`. The first column in the results Table shows **name** and then **span.http.user_agent**.

To use this capability:

1. In the **Aggregate by** row, select a scope from the first drop-down box. For example, `span`.
2. Select an attribute from the second drop-down.
3. Optional: Select **+** to add an **Aggregate by** row.
4. Optional: Select a **Time range** to expand or narrow the data set for an hour's range.
5. Select **Run query**.

Optional: Add queries

Using **Add query**, you can have successive queries that run in sequential order. For example, query A runs and then query B. You can reorder the queries by dragging and dropping them above or below other queries. Select **+ Add query** to add another query block.

Run queries and view results

Select **Run query** to run the TraceQL query (1 in the screenshot).

Queries can take a little while to return results. The results appear in a table underneath the query builder. Selecting a Trace ID (2 in the screenshot) displays more detailed information (3 in the screenshot).

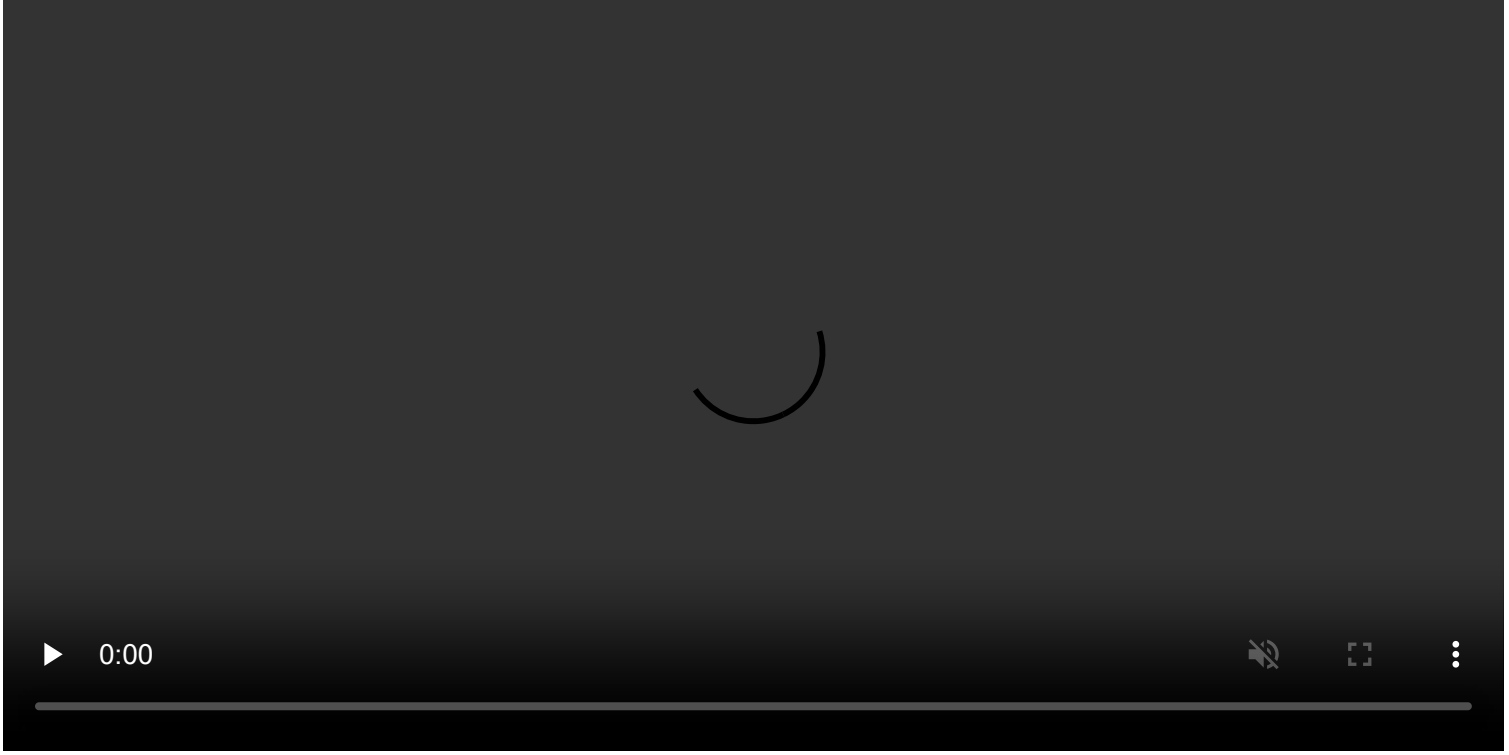
Tempo Search query type results

Streaming results

The Tempo data source supports streaming responses to TraceQL queries so you can see partial query results as they come in without waiting for the whole query to finish.

> **NOTE**
>
> To use this experimental feature, enable the `traceQLStreaming` feature toggle. If you're using Grafana Cloud and would like to enable this feature, please contact customer support.

Streaming is available for both the **Search** and **TraceQL** query types, and you'll get immediate visibility of incoming traces on the results table.


0:00

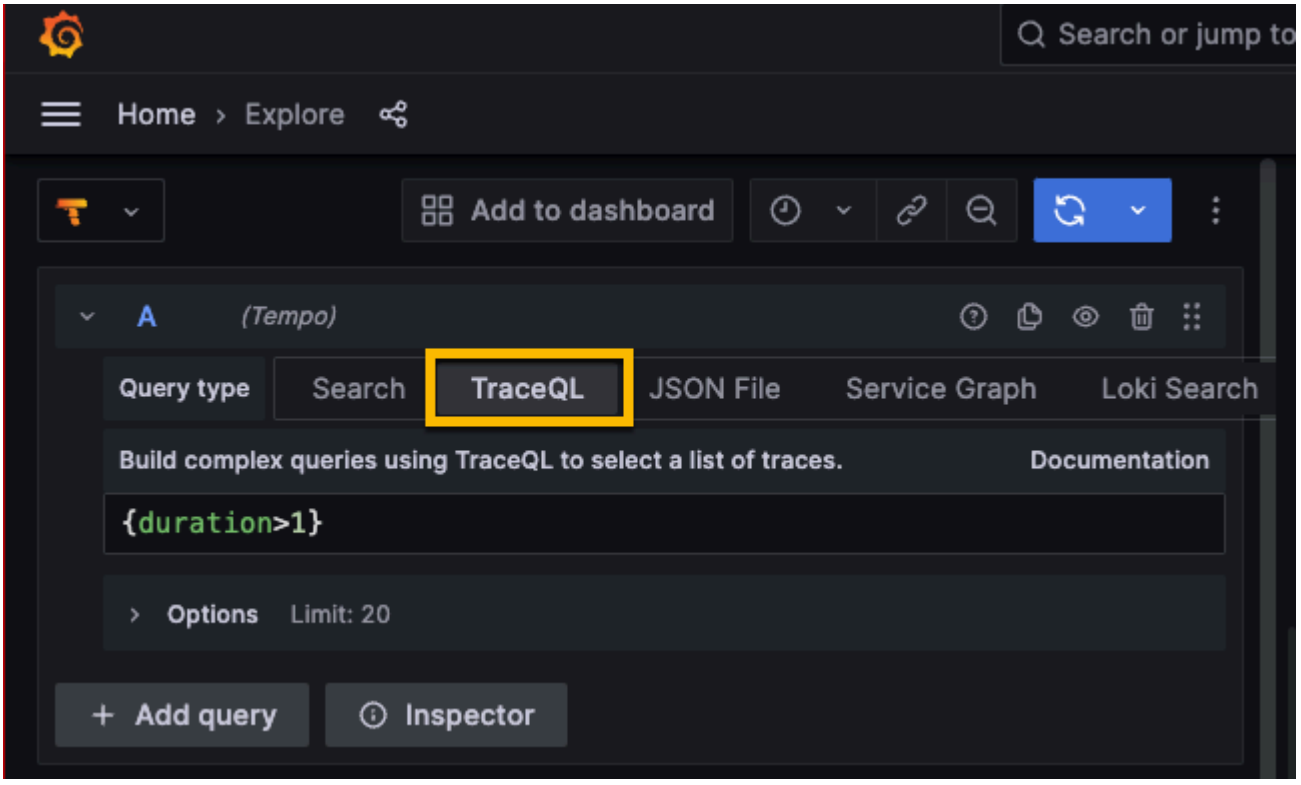
Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Write TraceQL queries with the editor

Inspired by PromQL and LogQL, TraceQL is a query language designed for selecting traces. TraceQL provides a method for formulating precise queries so you can zoom in to the data you need. Query results are returned faster because the queries limit what is searched.

To learn more about how to query by TraceQL, refer to the TraceQL documentation.

The TraceQL query editor, located on the **Explore** > **TraceQL** tab in Grafana, lets you search by trace ID and write TraceQL queries using autocomplete.



Enable the query editor

This feature is automatically available in Grafana 10 (and newer) and Grafana Cloud.

To use the TraceQL query editor in self-hosted Grafana 9.3.2 and older, you need to enable the `traceqlEditor` feature toggle.

Write TraceQL queries using the query editor

The Tempo data source's TraceQL query editor helps you query and display traces from Tempo in **Explore**.

To access the query editor, follow these steps:

1. Sign into Grafana or Grafana Cloud.
2. Select your Tempo data source.
3. From the menu, choose **Explore** and select the **TraceQL** tab.
4. Start your query on the text line by entering `{`. For help with TraceQL syntax, refer to the Construct a TraceQL query documentation.
5. Optional: Use the Time picker drop-down to change the time and range for the query (refer to the documentation for instructions).
6. Once you have finished your query, select **Run query**.

This video provides and example of creating a TraceQL query using the custom tag grouping.

Query by TraceID

To query a particular trace by its trace ID:

1. From the menu, choose **Explore**, select the desired Tempo data source, and navigate to the **TraceQL** tab.
2. Enter the trace ID into the query field. For example: `41928b92edf1cdbe0ba6594baee5ae9`
3. Click **Run query** or use the keyboard shortcut Shift + Enter.

Use autocomplete to write queries

You can use the query editor's autocomplete suggestions to write queries. The editor detects span sets to provide relevant autocomplete options. It uses regular expressions (regex) to detect where it's inside a spanset and provide attribute names, scopes, intrinsic names, logic operators, or attribute values from Tempo's API, depending on what's expected for the current situation.

To create a query using autocomplete, follow these steps:

1. From the menu, choose **Explore**, select the desired Tempo data source, and navigate to the **TraceQL** tab.
2. Enter your query. As you type your query, autocomplete suggestions appear as a drop-down. Each letter you enter refines the autocomplete options to match.
3. Use your mouse or arrow keys to select any option you wish. When the desired option is highlighted, press Tab on your keyboard to add the selection to your query.
4. Once your query is complete, select **Run query**.

View query results

Query results for both the editor and the builder are returned in a table. Selecting the Trace ID or Span ID provides more detailed information.

Selecting the trace ID from the returned results opens a trace diagram. Selecting a span from the returned results opens a trace diagram and reveals the relevant span in the trace diagram (the highlighted blue line).

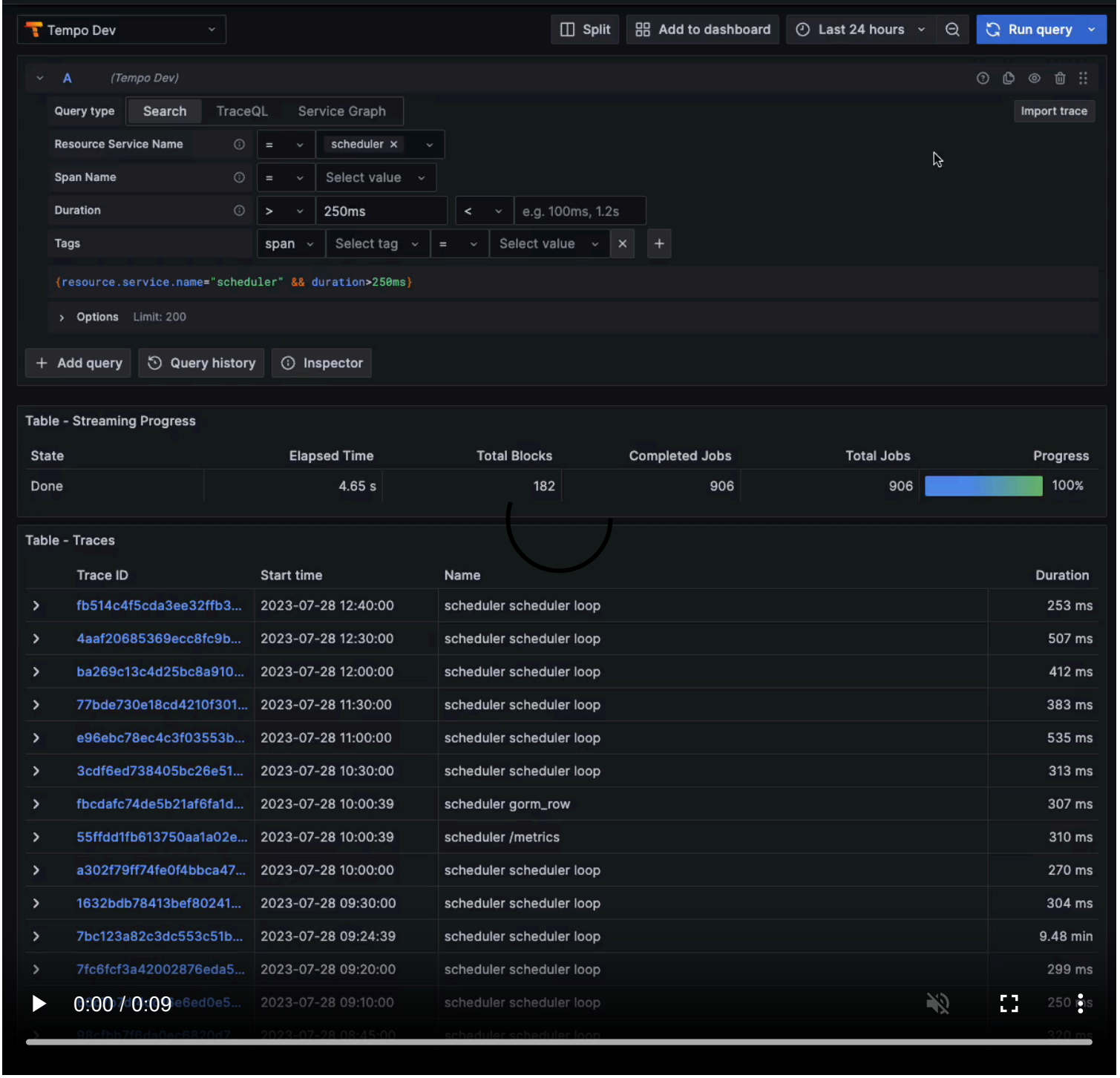
In the trace diagram, the bold text on the left side of each span indicates the service name, for example `mythical-requester: requester`, and it is hidden when subsequent spans have the same service name (nested spans). Each service has a color assigned to it, which is visible to the left of the name and timeline in the graph. Spans with the same color belong to the same service. The grey text to the right of the service name indicates the span name.

Streaming results

The Tempo data source supports streaming responses to TraceQL queries so you can see partial query results as they come in without waiting for the whole query to finish.

> **NOTE**
>
> To use this feature in Grafana OSS v10.1 and later, enable the `traceQLStreaming` feature toggle. This capability is enabled by default in Grafana Cloud.

Streaming is available for both the **Search** and **TraceQL** query types, and you'll get immediate visibility of incoming traces on the results table.


Tempo Dev
Split
Add to dashboard
Last 24 hours
Run query
A
(Tempo Dev)
Query type
Search
TraceQL
Service Graph
Import trace
Resource Service Name
=
scheduler
Span Name
=
Select value
Duration
>
250ms
<
e.g. 100ms, 1.2s
Tags
span
Select tag
=
Select value
{resource.service.name="scheduler" && duration>250ms}
Options
Limit: 200
Add query
Query history
Inspector
Table - Streaming Progress
State
Elapsed Time
Total Blocks
Completed Jobs
Total Jobs
Progress
Done
4.65 s
182
906
906
100%
Table - Traces
Trace ID
Start time
Name
Duration
fb514c4f5cda3ee32ffb3...
2023-07-28 12:40:00
scheduler scheduler loop
253 ms
4aaf20685369ecc8fc9b...
2023-07-28 12:30:00
scheduler scheduler loop
507 ms
ba269c13c4d25bc8a910...
2023-07-28 12:00:00
scheduler scheduler loop
412 ms
77bde730e18cd4210f301...
2023-07-28 11:30:00
scheduler scheduler loop
383 ms
e96ebc78ec4c3f03553b...
2023-07-28 11:00:00
scheduler scheduler loop
535 ms
3cdf6ed738405bc26e51...
2023-07-28 10:30:00
scheduler scheduler loop
313 ms
fbcdafc74de5b21af6fa1d...
2023-07-28 10:00:39
scheduler gorm_row
307 ms
55ffdd1fb613750aa1a02e...
2023-07-28 10:00:39
scheduler /metrics
310 ms
a302f79ff74fe0f4bbca47...
2023-07-28 10:00:00
scheduler scheduler loop
270 ms
1632bdb78413bef80241...
2023-07-28 09:30:00
scheduler scheduler loop
304 ms
7bc123a82c3dc553c51b...
2023-07-28 09:24:39
scheduler scheduler loop
9.48 min
7fc6fcf3a42002876eda5...
2023-07-28 09:20:00
scheduler scheduler loop
299 ms
2023-07-28 09:10:00
scheduler scheduler loop
0:00 / 0:09

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Upload a JSON trace file

You can upload a JSON file that contains a single trace and visualize it. If the file has multiple traces, Grafana visualizes the first trace.

To download a trace or Service Graph through the inspector:

1. Open the inspector.
2. Navigate to the **Data** tab.
3. Click **Download traces** or **Download Service Graph**.

Trace JSON example

json Copy

```
{
  "batches": [
    {
      "resource": {
        "attributes": [
          { "key": "service.name", "value": { "stringValue": "db" } },
          { "key": "job", "value": { "stringValue": "tns/db" } },
          { "key": "opencensus.exporterversion", "value": { "stringValue": "Jaeger-Go-2.22.1" } },
          { "key": "host.name", "value": { "stringValue": "63d16772b4a2" } },
          { "key": "ip", "value": { "stringValue": "0.0.0.0" } },
          { "key": "client-uuid", "value": { "stringValue": "39fb01637a579639" } }
        ]
      },
      "instrumentationLibrarySpans": [
```

```
{
  "instrumentationLibrary": {},
  "spans": [
    {
```


Expand code

Service Graph and Service Graph view

The Service Graph is a visual representation of the relationships between services. Each node on the graph represents a service such as an API or database.

You use the Service Graph to detect performance issues; track increases in error, fault, or throttle rates in services; and investigate root causes by viewing corresponding traces.

Screenshot of a Node Graph

Display the Service Graph

1. Configure Grafana Alloy or Tempo or GET to generate Service Graph data.
2. Link a Prometheus data source in the Tempo data source's Service Graph settings.
3. Navigate to Explore.
4. Select the Tempo data source.

5. Select the **Service Graph** query type.
6. Run the query.
7. *(Optional)* Filter by service name.

For details, refer to Node Graph panel.

Each circle in the graph represents a service. To open a context menu with additional links for quick navigation to other relevant information, click a service.

Numbers inside the circles indicate the average time per request and requests per second.

Each circle's color represents the percentage of requests in each state:

Color	State
Green	Success
Red	Fault
Yellow	Errors
Purple	Throttled responses

Open the Service Graph view

Service graph view displays a table of request rate, error rate, and duration metrics (RED) calculated from your incoming spans. It also includes a node graph view built from your spans.

Screenshot of the Service Graph view

For details, refer to the Service Graph view documentation.

To open the Service Graph view:

1. Link a Prometheus data source in the Tempo data source settings.
2. Navigate to Explore.
3. Select the Tempo data source.
4. Select the **Service Graph** query type.
5. Run the query.
6. *(Optional)* Filter your results.

NOTE

Grafana uses the `traces_spanmetrics_calls_total` metric to display the name, rate, and error rate columns, and `traces_spanmetrics_latency_bucket` to display the duration column. These metrics must exist in your Prometheus data source.

To open a query in Prometheus with the span name of that row automatically set in the query, click a row in the **rate**, **error rate**, or **duration** columns.

To open a query in Tempo with the span name of that row automatically set in the query, click a row in the **links** column.

Span Filters

Using span filters, you can filter your spans in the trace timeline viewer. The more filters you add, the more specific are the filtered spans.

You can add one or more of the following filters:

- Service name
- Span name
- Duration
- Tags (which include tags, process tags, and log fields)

To only show the spans you have matched, select the `Show matches only` toggle.

Link to a trace ID

You can link to Tempo traces from logs or metrics.

Link to a trace ID from logs

You can link to Tempo traces from logs in Loki, Elasticsearch, Splunk, and other logs data sources by configuring an internal link.

To configure this feature, see the Derived fields section of the Loki data source docs or the Data links section of the Elasticsearch or Splunk data source docs.

Link to a trace ID from metrics

You can link to Tempo traces from metrics in Prometheus data sources by configuring an exemplar.

To configure this feature, see the introduction to exemplars documentation.

TestData data source

Grafana ships with a TestData data source, which creates simulated time series data for any panel. You can use it to build your own fake and random time series data and render it in any panel, which helps you verify dashboard functionality since you can safely and easily share the data.

For instructions on how to add a data source to Grafana, refer to the administration documentation. Only users with the organization administrator role can add data sources.

Configure the data source

To configure basic settings for the data source, complete the following steps:

1. Click **Connections** in the left-side menu.
2. Under Your connections, click **Data sources**.
3. Enter `TestData` in the search bar.
4. Select **TestData**.

   The **Settings** tab of the data source is displayed. The data source doesn't provide any settings beyond the most basic options common to all data sources:

Name	Description
Name	Sets the name you use to refer to the data source in panels and queries.
Default	Defines whether this data source is pre-selected for new panels.

Create mock data

Adding test data

Once you've added the TestData data source, your Grafana instance's users can use it as a data source in any metric panel.

Choose a scenario

Instead of providing a query editor, the TestData data source helps you select a **Scenario** that generates simulated data for panels.

You can assign an **Alias** to each scenario, and many have their own options that appear when selected.

Using CSV Metric Values

Available scenarios:

- **Annotations**
- **Conditional Error**
- **CSV Content**
- **CSV File**

- **CSV Metric Values**
- **Datapoints Outside Range**
- **Exponential heatmap bucket data**
- **Flame Graph**
- **Grafana API**
- **Grafana Live**
- **Linear heatmap bucket data**
- **Load Apache Arrow Data**
- **Logs**
- **No Data Points**
- **Node Graph**
- **Predictable CSV Wave**
- **Predictable Pulse**
- **Random Walk**
- **Random Walk (with error)**
- **Random Walk Table**
- **Raw Frames**
- **Simulation**
- **Slow Query**
- **Streaming Client**
- **Table Static**
- **Trace**
- **USA generated data**

Import a pre-configured dashboard

TestData also provides an example dashboard.

To import the example dashboard:

1. Navigate to the data source's configuration page.
2. Select the **Dashboards** tab.
3. Select **Import** for the **Simple Streaming Example** dashboard.

To customize an imported dashboard:

To customize the imported dashboard, we recommend that you save it under a different name. If you don't, upgrading Grafana can overwrite the customized dashboard with the new version.

Use test data to report issues

If you report an issue on GitHub involving the use or rendering of time series data, we strongly recommend that you use this data source to replicate the issue. That makes it much easier for the developers to replicate and solve your issue.

Use a custom version of TestData

> **NOTE**
>
> This feature is experimental and requires Grafana version 10.3.0 or later.

If you want to use a version of TestData different from the one shipped with Grafana, follow these steps:

1. Enable the feature toggle `externalCorePlugins`.
2. Set the configuration field `as_external` for the plugin to `true`. An example configuration would be:

   ini — Copy

   ```
   [feature_toggles]
   externalCorePlugins = true

   [plugin.grafana-testdata-datasource]
   as_external = true
   ```

3. Restart Grafana.

These settings, if enabled, allow you to to install TestData as an external plugin and manage its lifecycle independently of Grafana.

With the feature toggle disabled (default) TestData can still be installed as an external plugin, but it has no effect as the bundled, Core version of TestData is already installed and takes precedence.

Zipkin data source

Grafana ships with built-in support for Zipkin, an open source, distributed tracing system. This topic explains configuration and queries specific to the Zipkin data source.

For instructions on how to add a data source to Grafana, refer to the administration documentation. Only users with the organization administrator role can add data sources. Administrators can also configure the data source via YAML with Grafana's provisioning system.

Once you've added the Zipkin data source, you can configure it so that your Grafana instance's users can create queries in its query editor when they build dashboards and use Explore.

You can also upload a JSON trace file, link to a trace ID from logs, and link to a trace ID from metrics.

Configure the data source

To configure basic settings for the data source, complete the following steps:

1. Click **Connections** in the left-side menu.
2. Under Your connections, click **Data sources**.
3. Enter `Zipkin` in the search bar.
4. Select **Zipkin**.

   The **Settings** tab of the data source is displayed.
5. Set the data source's basic configuration options:

Name	Description
Name	Sets the name you use to refer to the data source in panels and queries.

Name	Description
Default	Defines whether this data source is pre-selected for new panels.
URL	Sets the URL of the Zipkin instance, such as `http://localhost:9411`.
Basic Auth	Enables basic authentication for the Zipkin data source.
User	Defines the user name for basic authentication.
Password	Defines the password for basic authentication.

Trace to logs

> **NOTE**
>
> Available in Grafana v7.4 and higher. If you use Grafana Cloud, open a support ticket in the Cloud Portal to access this feature.

The **Trace to logs** setting configures the trace to logs feature that is available when you integrate Grafana with Zipkin.

There are two ways to configure the trace to logs feature:

- Use a simplified configuration with default query, or
- Configure a custom query where you can use a template language to interpolate variables from the trace or span.

Use a simple configuration

1. Select the target data source from the drop-down list.

   You can also click **Open advanced data source picker** to see more options, including adding a data source.

2. Set start and end time shift. As the logs timestamps may not exactly match the timestamps of the spans in trace it may be necessary to search in larger or shifted time range to find the desired logs.

3. Select which tags to use in the logs query. The tags you configure must be present in the spans attributes or resources for a trace to logs span link to appear. You can optionally configure a new name for the tag. This is useful if the tag has dots in the name and the target data source does not allow using dots in labels. In that case, you can for example remap `http.status` to `http_status`.

4. Optionally, switch on the **Filter by trace ID** and/or **Filter by span ID** setting to further filter the logs if your logs consistently contain trace or span IDs.

Configure a custom query

1. Select the target data source from the drop-down list.

   You can also click **Open advanced data source picker** to see more options, including adding a data source.

2. Set start and end time shift. Since the logs timestamps may not exactly match the timestamps of the spans in the trace, you may need to widen or shift the time range to find the desired logs.

3. Optionally, select tags to map. These tags can be used in the custom query with `${__tags}` variable. This variable will interpolate the mapped tags as list in an appropriate syntax for the data source and will only include the tags that were present in the span omitting those that weren't present. You can optionally configure a new name for the tag. This is useful when the tag has dots in the name and the target data source does not allow using dots in labels. For example, you can remap `http.status` to `http_status`. If you don't map any tags here, you can still use any tag in the query like this `method="${__span.tags.method}"`.

4. Skip **Filter by trace ID** and **Filter by span ID** settings as these cannot be used with a custom query.

5. Switch on **Use custom query**.

6 Specify a custom query to be used to query the logs. You can use various variables to make that query relevant for current span. The link will only be shown only if all the variables are interpolated with non-empty values to prevent creating an invalid query.

Variables that can be used in a custom query

To use a variable you need to wrap it in `${}`. For example `${__span.name}`.

Variable name	Description
__tags	This variable uses the tag mapping from the UI to create a label matcher string in the specific data source syntax. The variable only uses tags that are present in the span. The link is still created even if only one of those tags is present in the span. You can use this if all tags are not required for the query to be useful.
__span.spanId	The ID of the span.
__span.traceId	The ID of the trace.
__span.duration	The duration of the span.
__span.name	Name of the span.
__span.tags	Namespace for the tags in the span. To access a specific tag named `version`, you would use `${__span.tags.version}`. In case the tag contains dot, you have to access it as `${__span.tags["http.status"]}`.
__trace.traceId	The ID of the trace.
__trace.duration	The duration of the trace.
__trace.name	The name of the trace.

The following table describes the ways in which you can configure your trace to logs settings:

Setting name	Description
Data source	Defines the target data source. You can select only Loki or Splunk [logs][] data sources.
Span start time shift	Shifts the start time for the logs query, based on the span's start time. You can use time units, such as `5s`, `1m`, `3h`. To extend the time to the past, use a negative value. Default: `0`.
Span end time shift	Shifts the end time for the logs query, based on the span's end time. You can use time units. Default: `0`.
Tags	Defines the tags to use in the logs query. Default: `cluster`, `hostname`, `namespace`, `pod`, `service.name`, `service.namespace`. You can change the tag name for example to remove dots from the name if they are not allowed in the target data source. For example, map `http.status` to `http_status`.

Setting name	Description
Filter by trace ID	Toggles whether to append the trace ID to the logs query.
Filter by span ID	Toggles whether to append the span ID to the logs query.
Use custom query	Toggles use of custom query with interpolation.
Query	Input to write custom query. Use variable interpolation to customize it with variables from span.

Trace to metrics

> **NOTE**
>
> This feature is behind the `traceToMetrics` feature toggle. If you use Grafana Cloud, open a support ticket in the Cloud Portal to access this feature.

The **Trace to metrics** setting configures the trace to metrics feature available when integrating Grafana with Zipkin.

To configure trace to metrics:

1. Select the target data source from the drop-down list.

   You can also click **Open advanced data source picker** to see more options, including adding a data source.

2. Create any desired linked queries.

Setting name	Description
Data source	Defines the target data source.
Tags	Defines the tags used in linked queries. The key sets the span attribute name, and the optional value sets the corresponding metric label name. For example, you can map `k8s.pod` to `pod`. To interpolate these tags into queries, use the `$__tags` keyword.

Each linked query consists of:

- **Link Label:** *(Optional)* Descriptive label for the linked query.

- **Query:** The query ran when navigating from a trace to the metrics data source. Interpolate tags using the `$__tags` keyword. For example, when you configure the query `requests_total{$__tags}` with the tags `k8s.pod=pod` and `cluster`, the result looks like `requests_total{pod="nginx-554b9", cluster="us-east-1"}`.

Node Graph

The **Node Graph** setting enables the Node Graph visualization, which is disabled by default.

Once enabled, Grafana displays the Node Graph after loading the trace view.

Span bar

The **Span bar** setting helps you display additional information in the span bar row.

You can choose one of three options:

Name	Description
None	Adds nothing to the span bar row.
Duration	*(Default)* Displays the span duration on the span bar row.
Tag	Displays the span tag on the span bar row. You must also specify which tag key to use to get the tag value, such as `component`.

Provision the data source

You can define and configure the data source in YAML files as part of Grafana's provisioning system. For more information about provisioning and available configuration options, refer to Provisioning Grafana.

Provisioning example

yaml Copy

```yaml
apiVersion: 1

datasources:
  - name: Zipkin
    type: zipkin
    uid: EbPG8fYoz
    url: http://localhost:16686
    access: proxy
    basicAuth: true
```

```
    basicAuthUser: my_user
    readOnly: true
    isDefault: false
    jsonData:
      tracesToLogsV2:
        # Field with an internal link pointing to a logs data source in Grafana.
        # datasourceUid value must match the uid value of the logs data source.
        datasourceUid: 'loki'
        spanStartTimeShift: '1h'
```

Expand code

Query the data source

You can query and display traces from Zipkin via Explore

This topic explains configuration and queries specific to the Zipkin data source. For general documentation on querying data sources in Grafana, see Query and transform data.

Query by trace ID

To query a particular trace:

1. Select the **TraceID** query type.
2. Enter the trace's ID into the **Trace ID** field.

Screenshot of the Zipkin query editor

Query by trace selector

To select a particular trace from all traces logged in the time range you have selected in Explore, you can also query by trace selector. The trace selector has three levels of nesting:

- The service you're interested in.
- Particular operation, part of the selected service
- Specific trace in which the selected operation occurred, represented by the root operation name and trace duration

Screenshot of the Zipkin query editor with trace selector expanded

View data mapping in the trace UI

You can view Zipkin annotations in the trace view as logs with annotation value displayed under the annotation key.

Upload a JSON trace file

You can upload a JSON file that contains a single trace and visualize it. If the file has multiple traces, Grafana visualizes its first trace.

Screenshot of the Zipkin data source in explore with upload selected

Trace JSON example

json Copy


```
[
  {
    "traceId": "efe9cb8857f68c8f",
    "parentId": "efe9cb8857f68c8f",
    "id": "8608dc6ce5cafe8e",
    "kind": "SERVER",
    "name": "get /api",
```

```
    "timestamp": 1627975249601797,
    "duration": 23457,
    "localEndpoint": { "serviceName": "backend", "ipv4": "127.0.0.1", "port": 9000 },
    "tags": {
      "http.method": "GET",
      "http.path": "/api",
      "jaxrs.resource.class": "Resource",
      "jaxrs.resource.method": "printDate"
    },
    "shared": true
  }
]
```


Expand code

Span Filters

Using span filters, you can filter your spans in the trace timeline viewer. The more filters you add, the more specific are the filtered spans.

You can add one or more of the following filters:

- Service name
- Span name
- Duration
- Tags (which include tags, process tags, and log fields)

To only show the spans you have matched, you can press the `Show matches only` toggle.

Link to a trace ID from logs

You can link to Zipkin traces from logs in Loki, Elasticsearch, Splunk, and other logs data sources by configuring an internal link.

To configure this feature, see the Derived fields section of the Loki data source docs or the Data links section of the Elasticsearch or Splunk data source docs.

Link to a trace ID from metrics

You can link to Zipkin traces from metrics in Prometheus data sources by configuring an exemplar.

To configure this feature, see the introduction to exemplars documentation.

Dashboards

Note: Looking for prebuilt Grafana dashboards? Check out our full library of dashboards and more →

A dashboard is a set of one or more panels organized and arranged into one or more rows. Grafana ships with a variety of panels making it easy to construct the right queries, and customize the visualization so that you can create the perfect dashboard for your need. Each panel can interact with data from any configured Grafana data source.

Dashboard snapshots are static. Queries and expressions cannot be re-executed from snapshots. As a result, if you update any variables in your query or expression, it will not change your dashboard data.

Before you begin, ensure that you have configured a data source. See also:

- Use dashboards
- Build dashboards
- Create dashboard folders
- Manage dashboards
- Public dashboards
- Annotations
- Playlist
- Reporting
- Version history
- Import
- Export and share

- JSON model

Use dashboards

This topic provides an overview of dashboard features and shortcuts, and describes how to use dashboard search.

Dashboard feature overview

The dashboard user interface provides a number of features that you can use to customize the presentation of your data.

The following image and descriptions highlight all dashboard features.

- (1) **Grafana home**: Click **Home** in the breadcrumb to be redirected to the home page configured in the Grafana instance.
- (2) **Dashboard title**: When you click the dashboard title, you can search for dashboards contained in the current folder.
- (3) **Share dashboard or panel**: Use this option to share the current dashboard or panel using a link or snapshot. You can also export the dashboard definition from the share modal.
- (4) **Add**: Use this option to add a panel, dashboard row, or library panel to the current dashboard.
- (5) **Save dashboard**: Click to save changes to your dashboard.
- (6) **Dashboard insights**: Click to view analytics about your dashboard including information about users, activity, query counts. Learn more about dashboard analytics.
- (7) **Dashboard settings**: Use this option to change dashboard name, folder, and tags and manage variables and annotation queries. Learn more about dashboard settings.
- (8) **Time picker dropdown**: Click to select relative time range options and set custom absolute time ranges.
  - You can change the **Timezone** and **fiscal year** settings from the time range controls by clicking the **Change time settings** button.
  - Time settings are saved on a per-dashboard basis.
- (9) **Zoom out time range**: Click to zoom out the time range. Learn more about how to use common time range controls.
- (10) **Refresh dashboard**: Click to immediately trigger queries and refresh dashboard data.
- (11) **Refresh dashboard time interval**: Click to select a dashboard auto refresh time interval.
- (12) **View mode**: Click to display the dashboard on a large screen such as a TV or a kiosk. View mode hides irrelevant information such as navigation menus. Learn more about view mode in our How to Create Kiosks to Display Dashboards on a TV blog post.
- (13) **Dashboard panel**: The primary building block of a dashboard is the panel. To add a new panel, dashboard row, or library panel, click **Add panel**.
  - Library panels can be shared among many dashboards.
  - To move a panel, drag the panel header to another location.
  - To resize a panel, click and drag the lower right corner of the panel.

- (14) **Graph legend**: Change series colors, y-axis and series visibility directly from the legend.
- (15) **Dashboard row**: A dashboard row is a logical divider within a dashboard that groups panels together.
  - Rows can be collapsed or expanded allowing you to hide parts of the dashboard.
  - Panels inside a collapsed row do not issue queries.
  - Use repeating rows to dynamically create rows based on a template variable.

Keyboard shortcuts

Grafana has a number of keyboard shortcuts available. Press `?` or `h` on your keyboard to display all keyboard shortcuts available in your version of Grafana.

- `Ctrl+S`: Saves the current dashboard.
- `f`: Opens the dashboard finder / search.
- `d+k`: Toggle kiosk mode (hides the menu).
- `d+e`: Expand all rows.
- `d+s`: Dashboard settings.
- `Ctrl+K`: Opens the command palette.
- `Esc`: Exits panel when in fullscreen view or edit mode. Also returns you to the dashboard from dashboard settings.

Focused panel

By hovering over a panel with the mouse you can use some shortcuts that will target that panel.

- `e`: Toggle panel edit view
- `v`: Toggle panel fullscreen view
- `ps`: Open Panel Share Modal
- `pd`: Duplicate Panel
- `pr`: Remove Panel
- `pl`: Toggle panel legend

Set dashboard time range

Grafana provides several ways to manage the time ranges of the data being visualized, for dashboard, panels and also for alerting.

This section describes supported time units and relative ranges, the common time controls, dashboard-wide time settings, and panel-specific time settings.

Time units and relative ranges

Grafana supports the following time units: `s (seconds)`, `m (minutes)`, `h (hours)`, `d (days)`, `w (weeks)`, `M (months)`, `Q (quarters)` and `y (years)`.

The minus operator enables you to step back in time, relative to the current date and time, or `now`. If you want to display the full period of the unit (day, week, month, etc...), append `/<time unit>` to the end. To view fiscal periods, use `fQ (fiscal quarter)` and `fy (fiscal year)` time units.

The plus operator enables you to step forward in time, relative to now. For example, you can use this feature to look at predicted data in the future.

The following table provides example relative ranges:

Example relative range	From:	To:
Last 5 minutes	`now-5m`	`now`
The day so far	`now/d`	`now`
This week	`now/w`	`now/w`
This week so far	`now/w`	`now`
This month	`now/M`	`now/M`
This month so far	`now/M`	`now`
Previous Month	`now-1M/M`	`now-1M/M`
This year so far	`now/Y`	`now`
This Year	`now/Y`	`now/Y`
Previous fiscal year	`now-1y/fy`	`now-1y/fy`

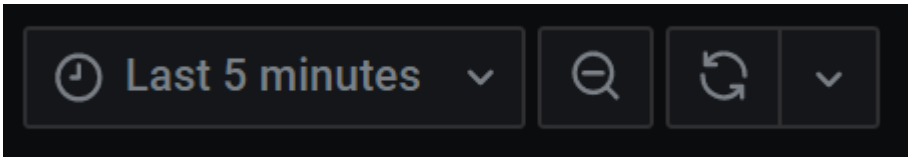
NOTE

Grafana Alerting does not support the following syntaxes at this time:

- now+n for future timestamps.
- now-1n/n for “start of n until end of n” because this is an absolute timestamp.

Common time range controls

The dashboard and panel time controls have a common UI.

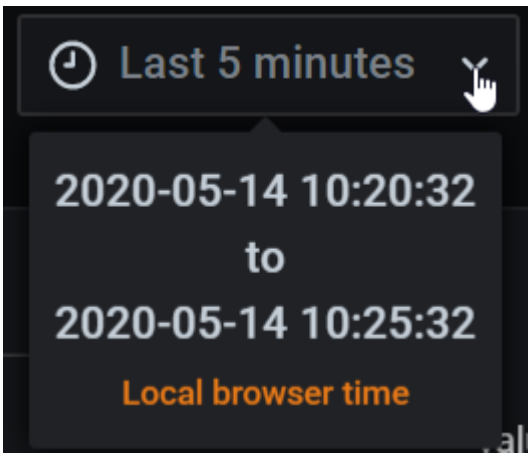


The following sections define common time range controls.

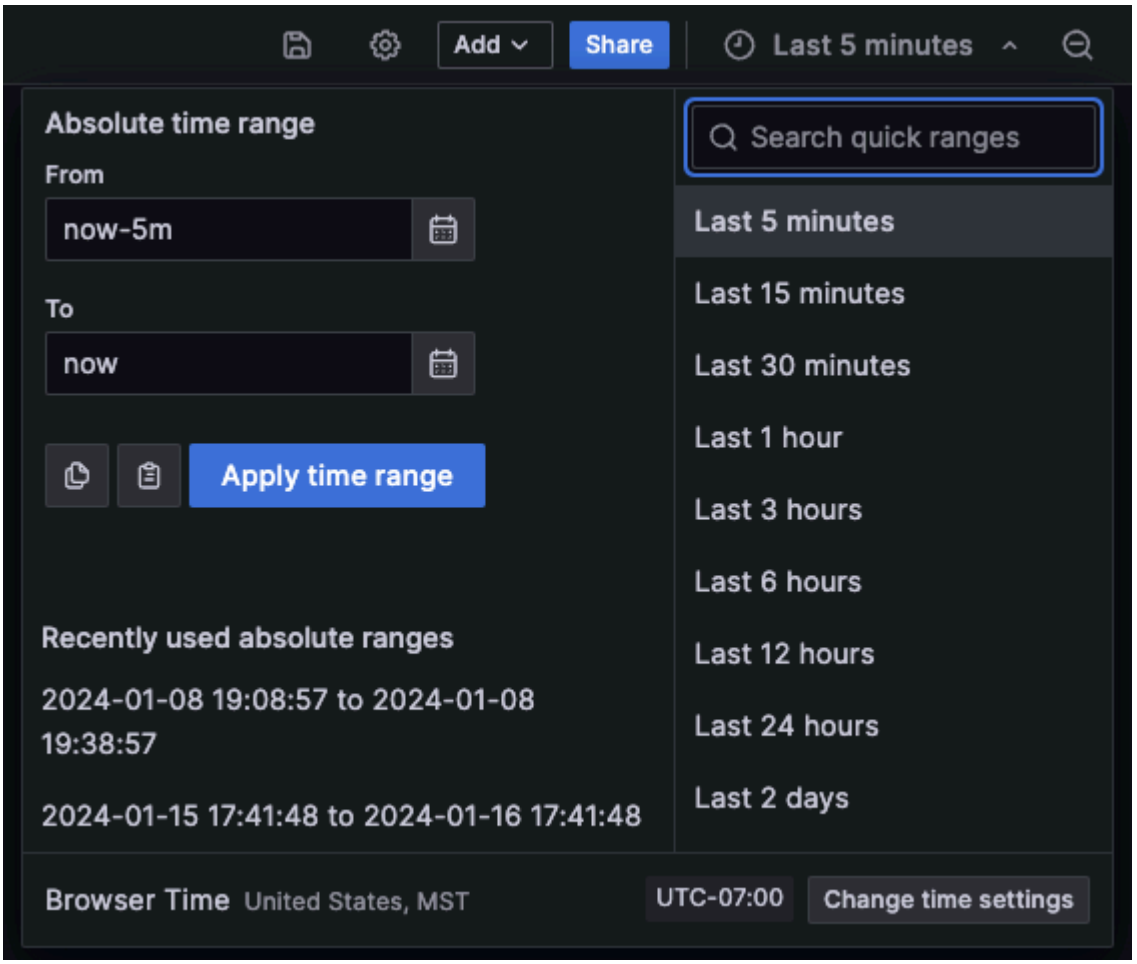
Current time range

The current time range, also called the *time picker*, shows the time range currently displayed in the dashboard or panel you are viewing.

Hover your cursor over the field to see the exact time stamps in the range and their source (such as the local browser).



Click the current time range to change it. You can change the current time using a *relative time range*, such as the last 15 minutes, or an *absolute time range*, such as `2020-05-14 00:00:00 to 2020-05-15 23:59:59`.



Relative time range

Select the relative time range from the **Relative time ranges** list. You can filter the list using the input field at the top. Some examples of time ranges include:

- Last 30 minutes

- Last 12 hours
- Last 7 days
- Last 2 years
- Yesterday
- Day before yesterday
- This day last week
- Today so far
- This week so far
- This month so far

Absolute time range

You can set an absolute time range in the following ways:

- Type values into the **From** and **To** fields. You can type exact time values or relative values, such as `now-24h`, and then click **Apply time range**.
- Click in the **From** or **To** field. Grafana displays a calendar. Click the day or days you want to use as the current time range and then click **Apply time range**.

This section also displays recently used absolute ranges.

Semi-relative time range

> **NOTE**
>
> Grafana Alerting does not support semi-relative time ranges.

You can also use the absolute time range settings to set a semi-relative time range. Semi-relative time range dashboards are useful when you need to monitor the progress of something over time, but you also want to see the entire history from a starting point.

Set a semi-relative time range by setting the start time to an absolute timestamp and the end time to a “now” that is relative to the current time. For example:

Start time: `2023-05-01 00:00:00`

End time: `now`

If you wanted to track the progress of something during business hours, you could set a time range that covers the current day, but starting at 8am, like so:

Start time: `now/d+8h`

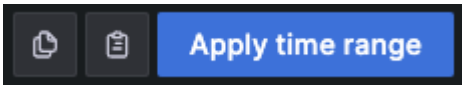
End time: `now`

This is equivalent to the **Today so far** time range preset, but it starts at 8:00am instead of 12:00am by appending +8h to the periodic start time.

Using a semi-relative time range, as time progresses, your dashboard will automatically and progressively zoom out to show more history and fewer details. At the same rate, as high data resolution decreases, historical trends over the entire time period will become more clear.

Copy and paste time range

You can copy and paste the time range from a dashboard to **Explore** and vice versa, or from one dashboard to another. Click the **Copy time range to clipboard** icon to copy the current time range to the clipboard. Then paste the time range into **Explore** or another dashboard.



You can also copy and paste a time range using the keyboard shortcuts `t+c` and `t+v` respectively.

Zoom out (Cmd+Z or Ctrl+Z)

Click the **Zoom out** icon to view a larger time range in the dashboard or panel visualization.

Zoom in (only applicable to graph visualizations)

Click and drag to select the time range in the visualization that you want to view.

Refresh dashboard

Click the **Refresh dashboard** icon to immediately run every query on the dashboard and refresh the visualizations. Grafana cancels any pending requests when you trigger a refresh.

By default, Grafana does not automatically refresh the dashboard. Queries run on their own schedule according to the panel settings. However, if you want to regularly refresh the dashboard, click the down arrow next to the **Refresh dashboard** icon, and then select a refresh interval.

Selecting the **Auto** interval schedules a refresh based on the query time range and browser window width. Short time ranges update frequently, while longer ones update infrequently. There is no need to refresh more often then the pixels available to draw any updates.

Control the time range using a URL

You can control the time range of a dashboard by providing the following query parameters in the dashboard URL:

- `from` - Defines the lower limit of the time range, specified in ms, epoch, or relative time.
- `to` - Defines the upper limit of the time range, specified in ms, epoch, or relative time.
- `time` and `time.window` - Defines a time range from `time-time.window/2` to `time+time.window/2`. Both parameters should be specified in `ms`. For example `?time=1500000000000&time.window=10000` results in a 10-second time range from 1499999995000 to 1500000005000`.
- `timezone` - Defines the time zone. For example `timezone=Europe/Madrid`.

Since these aren't variables, they don't require the `var-` prefix.

The following example shows a dashboard with the time range of the last five minutes:

```
https://${your-domain}/path/to/your/dashboard?from=now-5m&to=now
```

Copy

Build dashboards

This section includes the following topics:

Dynamic dashboards

You can create more interactive and dynamic dashboards by adding and using variables. Instead of hard-coding things like server, application, and sensor names in your metric queries, you can use variables in their place. Read more about variables here.

Import dashboards

You can import preconfigured dashboards into your Grafana instance or Cloud stack using the UI or the HTTP API.

Import a dashboard

To import a dashboard, follow these steps:

1. Click **Dashboards** in the primary menu.
2. Click **New** and select **Import** in the drop-down menu.
3. Perform one of the following steps:
   - Upload a dashboard JSON file.
   - Paste a Grafana.com dashboard URL or ID into the field provided.
   - Paste dashboard JSON text directly into the text area.
4. (Optional) Change the dashboard name, folder, or UID, and specify metric prefixes, if the dashboard uses any.
5. Select a data source, if required.
6. Click **Import**.
7. Save the dashboard.

Discover dashboards on grafana.com

The Dashboards page on grafana.com provides you with dashboards for common server applications. Browse our library of official and community-built dashboards and import them to quickly get up and running.

You can also add to this library by exporting one of your own dashboards. For more information, refer to Share dashboards and panels.

More examples

Your Grafana Cloud stack comes with several default dashboards in the **Grafana Cloud** folder in **Dashboards**. If you're running your own installation of Grafana, you can find more example dashboards in the `public/dashboards/` directory.

Dashboard URL variables

Dashboard URL variables allow you to provide more context when you share a dashboard URL.

For example, you could share a basic URL to your dashboard that looks like this:

```
https://${your-domain}/path/to/your/dashboard
```

Copy

This allows someone to navigate to the dashboard, but doesn't provide any helpful context that might be available.

Instead, you can add dashboard variables, passed as query parameters in the dashboard URL, to provide a URL like this:

```
https://${your-domain}/path/to/your/dashboard?var-example=value
```

Copy

This allows you to provide added context to the dashboard when someone navigates to it.

Variables as query parameters

Grafana interprets query string parameters prefixed with `var-` as variables in the given dashboard.

For example:

```
https://${your-domain}/path/to/your/dashboard?var-example=value
```

Copy

In this URL, the query parameter `var-example=value` represents the dashboard variable `example` with a value of `value`.

Multiple values for a variable

To pass multiple values, repeat the variable parameter once for each value:

```
https://${your-domain}/path/to/your/dashboard?var-example=value1&var-example=value2
```
Copy

Grafana interprets `var-example=value1&var-example=value2` as the dashboard variable `example` with two values: `value1` and `value2`.

Example

This dashboard in Grafana Play passes the variable `server` with multiple values, and the variables `app` and `interval` with a single value each.

Ad hoc filters

Ad hoc filters apply key/value filters to all metric queries that use the specified data source. For more information, refer to Add ad hoc filters.

To pass an ad hoc filter as a query parameter, use the variable syntax to pass the ad hoc filter variable. Then provide the key, operator, and value as a pipe-separated list.

For example:

```
https://${your-domain}/path/to/your/dashboard?var-adhoc=example_key|=|example_value
```
Copy

In this URL, the query parameter `var-adhoc=key|=|value` applies the ad hoc filter configured as the `adhoc` dashboard variable using the `example_key` key, the `=` operator, and the `example_value` value.

> **NOTE**
>
> When sharing URLs with ad hoc filters, remember to encode the URL. In the preceding example, replace the pipes (`|`) with `%7C` and the equality operator (`=`) with `%3D`.

Example

This dashboard in Grafana Play passes the ad hoc filter variable `adhoc` with the filter value `datacenter = America`.

Time range control using the URL

You can control the time range of a dashboard by providing the following query parameters in the dashboard URL:

- `from` - Defines the lower limit of the time range, specified in ms, epoch, or relative time.
- `to` - Defines the upper limit of the time range, specified in ms, epoch, or relative time.
- `time` and `time.window` - Defines a time range from `time-time.window/2` to `time+time.window/2`. Both parameters should be specified in `ms`. For example `?time=1500000000000&time.window=10000` results in a 10-second time range from 1499999995000 to 1500000005000`.

- `timezone` - Defines the time zone. For example `timezone=Europe/Madrid`.

Since these aren't variables, they don't require the `var-` prefix.

The following example shows a dashboard with the time range of the last five minutes:

```
https://${your-domain}/path/to/your/dashboard?from=now-5m&to=now
```

Copy

Variables in dashboard links

When you create dashboard links the dashboard settings, you can have current dashboard variables included in the link by selecting that option:

For steps to add variables to dashboard links, refer to Manage dashboard links.

Manage library panels

A library panel is a reusable panel that you can use in any dashboard. When you make a change to a library panel, that change propagates to all instances of where the panel is used. Library panels streamline reuse of panels across multiple dashboards.

You can save a library panel in a folder alongside saved dashboards.

Role-based access control

You can control permissions for library panels using role-based access control (RBAC). RBAC provides a standardized way of granting, changing, and revoking access when it comes to viewing and modifying Grafana resources, such as dashboards, reports, and administrative settings.

Create a library panel

When you create a library panel, the panel on the source dashboard is converted to a library panel as well. You need to save the original dashboard once a panel is converted.

1. Open a panel in edit mode.
2. In the panel display options, click the down arrow option to bring changes to the visualization.

3. Click **Library panels**, and then click **+ Create library panel** to open the create dialog.

4. In **Library panel name**, enter the name.
5. In **Save in folder**, select the folder to save the library panel.
6. Click **Create library panel** to save your changes.
7. Save the dashboard.

Once created, you can modify the library panel using any dashboard on which it appears. After you save the changes, all instances of the library panel reflect these modifications.

You can also create a library panel directly from the edit menu of any panel.

Add a library panel to a dashboard

Add a Grafana library panel to a dashboard when you want to provide visualizations to other dashboard users.

1. Click **Dashboards** in the left-side menu.

2. Click **New** and select **New Dashboard** in the dropdown.
3. On the empty dashboard, click **+ Import library panel**.

   You will see a list of your library panels.
4. Filter the list or search to find the panel you want to add.
5. Click a panel to add it to the dashboard.

Unlink a library panel

Unlink a library panel when you want to make a change to the panel and not affect other instances of the library panel.

1. Click **Dashboards** in the left-side menu.
2. Click **Library panels**.
3. Select a library panel that is being used in different dashboards.
4. Select the panel you want to unlink.
5. Hover over any part of the panel to display the actions menu on the top right corner.
6. Click the menu and select **Edit**.
7. Click **Unlink** on the top right corner of the page.
8. Click **Yes, unlink**.

View a list of library panels

You can view a list of available library panels and search for a library panel.

1. Click **Dashboards** in the left-side menu.
2. Click **Library panels**.

   You can see a list of previously defined library panels.

3. Search for a specific library panel if you know its name.

   You can also filter the panels by folder or type.

Delete a library panel

Delete a library panel when you no longer need it.

1. Click **Dashboards** in the left-side menu.
2. Click **Library panels**.
3. Click the delete icon next to the library panel name.

Manage dashboard version history

Whenever you save a version of your dashboard, a copy of that version is saved so that previous versions of your dashboard are never lost. A list of these versions is available by entering the dashboard settings and then selecting “Versions” in the left side menu.

The dashboard version history feature lets you compare and restore to previously saved dashboard versions.

Comparing two dashboard versions

To compare two dashboard versions, select the two versions from the list that you wish to compare. Once selected, the “Compare versions” button will become clickable. Click the button to view the diff between the two versions.

Upon clicking the button, you'll be brought to the diff view. By default, you'll see a textual summary of the changes, like in the image below.

If you want to view the diff of the raw JSON that represents your dashboard, you can do that as well by clicking the expand icon for the View JSON Diff section at the bottom.

Restoring to a previously saved dashboard version

If you need to restore to a previously saved dashboard version, you can do so by either clicking the "Restore" button on the right of a row in the dashboard version list, or by clicking the **Restore to version <x>** button appearing in the diff view. Clicking the button will bring up the following popup prompting you to confirm the restoration.

After restoring to a previous version, a new version will be created containing the same exact data as the previous version, only with a different version number. This is indicated in the “Notes column” for the row in the new dashboard version. This is done simply to ensure your previous dashboard versions are not affected by the change.

Manage dashboard links

You can use links to navigate between commonly-used dashboards or to connect others to your visualizations. Links let you create shortcuts to other dashboards, panels, and even external websites.

Grafana supports dashboard links, panel links, and data links. Dashboard links are displayed at the top of the dashboard. Panel links are accessible by clicking the icon next to the panel title.

Which link should you use?

Start by figuring out how you're currently navigating between dashboards. If you're often jumping between a set of dashboards and struggling to find the same context in each, links can help optimize your workflow.

The next step is to figure out which link type is right for your workflow. Even though all the link types in Grafana are used to create shortcuts to other dashboards or external websites, they work in different contexts.

- If the link relates to most if not all of the panels in the dashboard, use dashboard links.
- If you want to drill down into specific panels, use panel links.
- If you want to link to an external site, you can use either a dashboard link or a panel link.
- If you want to drill down into a specific series, or even a single measurement, use data links.

Controlling time range using the URL

To control the time range of a panel or dashboard, you can provide query parameters in the dashboard URL:

- `from` - defines lower limit of the time range, specified in ms epoch
- `to` - defines upper limit of the time range, specified in ms epoch

- `time` and `time.window` - defines a time range from `time-time.window/2` to `time+time.window/2`. Both params should be specified in ms. For example `?time=1500000000000&time.window=10000` will result in 10s time range from 1499999995000 to 1500000005000

Dashboard links

When you create a dashboard link, you can include the time range and current template variables to directly jump to the same context in another dashboard. This way, you don't have to worry whether the person you send the link to is looking at the right data. For other types of links, refer to Data link variables.

Dashboard links can also be used as shortcuts to external systems, such as submitting a GitHub issue with the current dashboard name.

To see an example of dashboard links in action, check out:

- Dashboard links with variables
- Prometheus repeat

Once you've added a dashboard link, it appears in the upper right corner of your dashboard.

Add links to dashboards

Add links to other dashboards at the top of your current dashboard.

1. While viewing the dashboard you want to link, click the gear at the top of the screen to open **Dashboard settings**.
2. Click **Links** and then click **Add Dashboard Link** or **New**.
3. In **Type**, select **dashboards**.
4. Select link options:
    - **With tags** – Enter tags to limit the linked dashboards to only the ones with the tags you enter. Otherwise, Grafana includes links to all other dashboards.
    - **As dropdown** – If you are linking to lots of dashboards, then you probably want to select this option and add an optional title to the dropdown. Otherwise, Grafana displays the dashboard links side by side across the top of your dashboard.
    - **Time range** – Select this option to include the dashboard time range in the link. When the user clicks the link, the linked dashboard opens with the indicated time range already set. **Example:** https://play.grafana.org/d/000000010/annotations?orgId=1&from=now-3h&to=now
    - **Variable values** – Select this option to include template variables currently used as query parameters in the link. When the user clicks the link, any matching templates in the linked dashboard are set to the values from the link. For more information, see Dashboard URL variables.

- **Open in new tab** – Select this option if you want the dashboard link to open in a new tab or window.

5. Click **Add**.

Add a URL link to a dashboard

Add a link to a URL at the top of your current dashboard. You can link to any available URL, including dashboards, panels, or external sites. You can even control the time range to ensure the user is zoomed in on the right data in Grafana.

1. While viewing the dashboard you want to link, click the gear at the top of the screen to open **Dashboard settings**.
2. Click **Links** and then click **Add Dashboard Link** or **New**.
3. In **Type**, select **link**.
4. Select link options:
   - **Url** – Enter the URL you want to link to. Depending on the target, you might want to include field values. **Example:** https://github.com/grafana/grafana/issues/new?title=Dashboard%3A%20HTTP%20Requests
   - **Title** – Enter the title you want the link to display.
   - **Tooltip** – Enter the tooltip you want the link to display when the user hovers their mouse over it.
   - **Icon** – Choose the icon you want displayed with the link.
   - **Time range** – Select this option to include the dashboard time range in the link. When the user clicks the link, the linked dashboard opens with the indicated time range already set. **Example:** https://play.grafana.org/d/000000010/annotations?orgId=1&from=now-3h&to=now
     - `from` - Defines the lower limit of the time range, specified in ms epoch.
     - `to` - Defines the upper limit of the time range, specified in ms epoch.
     - `time` and `time.window` - Define a time range from `time-time.window/2` to `time+time.window/2`. Both params should be specified in ms. For example `?time=150000000000&time.window=10000` will result in 10s time range from 1499999995000 to 1500000005000.
   - **Variable values** – Select this option to include template variables currently used as query parameters in the link. When the user clicks the link, any matching templates in the linked dashboard are set to the values from the link. Here is the variable format: `https://${you-domain}/path/to/your/dashboard?var-${template-varable1}=value1&var-{template-variable2}=value2` **Example:** https://play.grafana.org/d/000000074/alerting?var-app=backend&var-server=backend_01&var-server=backend_03&var-interval=1h
   - **Open in new tab** – Select this option if you want the dashboard link to open in a new tab or window.
5. Click **Add**.

Update a dashboard link

To change or update an existing dashboard link, follow this procedure.

1. In Dashboard Settings, on the Links tab, click the existing link that you want to edit.
2. Change the settings and then click **Update**.

Duplicate a dashboard link

To duplicate an existing dashboard link, click the duplicate icon next to the existing link that you want to duplicate.

Delete a dashboard link

To delete an existing dashboard link, click the trash icon next to the duplicate icon that you want to delete.

Panel links

Each panel can have its own set of links that are shown in the upper left of the panel after the panel title. You can link to any available URL, including dashboards, panels, or external sites. You can even control the time range to ensure the user is zoomed in on the right data in Grafana.

Click the icon next to the panel title to see available panel links.

Add a panel link

1. Hover over any part of the panel to which you want to add the link to display the actions menu on the top right corner.
2. Click the menu and select **Edit**.

To use a keyboard shortcut to open the panel, hover over the panel and press `e`.

3. Expand the **Panel options** section, scroll down to Panel links.
4. Click **Add link**.
5. Enter a **Title**. **Title** is a human-readable label for the link that will be displayed in the UI.
6. Enter the **URL** you want to link to. You can even add one of the template variables defined in the dashboard. Press Ctrl+Space or Cmd+Space and click in the **URL** field to see the available variables. By adding template variables to your panel link, the link sends the user to the right context, with the relevant variables already set. You can also use time variables:
   - `from` - Defines the lower limit of the time range, specified in ms epoch.
   - `to` - Defines the upper limit of the time range, specified in ms epoch.
   - `time` and `time.window` - Define a time range from `time-time.window/2` to `time+time.window/2`. Both params should be specified in ms. For example `?time=1500000000000&time.window=10000` will result in 10s time range from 1499999995000 to 1500000005000.
7. If you want the link to open in a new tab, then select **Open in a new tab**.
8. Click **Save** to save changes and close the window.
9. Click **Save** in the upper right to save your changes to the dashboard.

Update a panel link

1. Hover over any part of the panel to display the actions menu on the top right corner.
2. Click the menu and select **Edit**.

   To use a keyboard shortcut to open the panel, hover over the panel and press `e`.
3. Expand the **Panel options** section, scroll down to Panel links.
4. Find the link that you want to make changes to.
5. Click the Edit (pencil) icon to open the Edit link window.
6. Make any necessary changes.
7. Click **Save** to save changes and close the window.
8. Click **Save** in the upper right to save your changes to the dashboard.

Delete a panel link

1. Hover over any part of the panel to display the actions menu on the top right corner.
2. Click the menu and select **Edit**.

   To use a keyboard shortcut to open the panel, hover over the panel and press `e`.

3. Expand the **Panel options** section, scroll down to Panel links.
4. Find the link that you want to delete.
5. Click the **X** icon next to the link you want to delete.
6. Click **Save** in the upper right to save your changes to the dashboard.

Annotate visualizations

Annotations provide a way to mark points on a visualization with rich events. They are visualized as vertical lines and icons on all graph panels. When you hover over an annotation, you can get event description and event tags. The text field can include links to other systems with more detail.

Give it a try using Grafana Play

With Grafana Play, you can explore and see how it works, learning from practical examples to accelerate your development. This feature can be seen on Annotations.

You can annotate visualizations in three ways:

- Directly in the panel, using the built-in annotations query
- Using the HTTP API
- Configuring annotation queries in the dashboard settings

In the first two cases, you're creating new annotations, while in the last you're querying existing annotations from data sources. The built-in annotation query also supports this.

This page explains the first and third options; for information about using the HTTP API, refer to Annotations API.

Annotations are supported for the following visualization types:

- Time series
- State timeline
- Candlestick

Create annotations in panels

Grafana comes with the ability to add annotation events directly from a panel using the built-in annotation query that exists on all dashboards. Annotations that you create this way are stored in Grafana.

To add annotations directly in the panel:

- The dashboard must already be saved.
- The built-in query must be enabled. Learn more in Built-in query.

Watch the following video for a quick tutorial on creating annotations:

Add an annotation

To add an annotation, complete the following steps:

1. In the dashboard click the panel to which you're adding the annotation. A context menu will appear.

2. In the context menu, click **Add annotation**.

3. Add an annotation description and tags (optional).
4. Click **Save**.

Alternatively, to add an annotation, press Ctrl/Cmd and click the panel, and the **Add annotation** popover will appear.

Add a region annotation

1. In the dashboard press Ctrl/Cmd and click and drag on the panel.

2. Add an annotation description and tags (optional).
3. Click **Save**.

Edit an annotation

1. In the dashboard, hover over an annotation indicator on the Time series panel.
2. Click on the pencil icon in the annotation tooltip.
3. Modify the description and/or tags.
4. Click save.

Delete an annotation

1. In the dashboard hover over an annotation indicator on a panel.
2. Click on the trash icon in the annotation tooltip.

Fetch annotations through dashboard settings

In the dashboard settings, under **Annotations**, you can add new queries to fetch annotations using any data source, including the built-in data annotation data source. Annotation queries return events that can be visualized as event markers in graphs across the dashboard.

Check out the video below for a quick tutorial.

Add new annotation queries

To add a new annotation query to a dashboard, take the following steps:

1. Click the dashboard settings (gear) icon in the dashboard header to open the settings menu.
2. Select **Annotations**.
3. Click **Add annotation query**.

   If you've added a query before, the **+ New query** button is displayed.
4. Enter a name for the annotation query.

   This name is given to the toggle (checkbox) that will allow you to enable/disable showing annotation events from this query.
5. Select the data source for the annotations.

   You can also click **Open advanced data source picker** to see more options, including adding a data source (Admins only).
6. If you don't want to use the annotation query right away, clear the **Enabled** checkbox.
7. If you don't want the annotation query toggle to be displayed in the dashboard, select the **Hidden** checkbox.

8. Select a color for the event markers.
9. In the **Show in** drop-down, choose one of the following options:
   - **All panels** - The annotations are displayed on all panels that support annotations.
   - **Selected panels** - The annotations are displayed on all the panels you select.
   - **All panels except** - The annotations are displayed on all panels except the ones you select.

Annotation filtering

10. Configure the query.

    The annotation query options are different for each data source. For information about annotations in a specific data source, refer to the specific data source topic.

Built-in query

After you add an annotation, they will still be visible. This is due to the built-in annotation query that exists on all dashboards. This annotation query will fetch all annotation events that originate from the current dashboard, which are stored in Grafana, and show them on the panel where they were created. This includes alert state history annotations.

By default, the built-in annotation query uses the `-- Grafana --` special data source, and manual annotations are only supported using this data source. You can use another data source in the built-in annotation query, but you'll only be able to create automated annotations using the query editor for that data source.

To add annotations directly to the dashboard, this query must be enabled.

To confirm if the built-in query is enabled, take the following steps:

1. Click the dashboard settings (gear) icon in the dashboard header to open the dashboard settings menu.
2. Click **Annotations**.
3. Find the **Annotations & Alerts (Built-in)** query.

   If it says **Disabled** before the name of the query, then you'll need to click the query name to open it and update the setting.

You can stop annotations from being fetched and drawn by taking the following steps:

1. Click the dashboard settings (gear) icon in the dashboard header to open the settings menu.
2. Click **Annotations**.
3. Find and click the **Annotations & Alerts (Built-in)** query to open it.
4. Click the **Enabled** toggle to turn it off.

When you copy a dashboard using the **Save As** feature it will get a new dashboard id, so annotations created on the source dashboard will no longer be visible on the copy. You can still show them if you add a new **Annotation Query** and filter by tags. However, this only works if the annotations on the source dashboard had tags to filter by.

Following are some query options specific to the built-in annotation query.

Filter queries by tag

You can create new queries to fetch annotations from the built-in annotation query using the `-- Grafana --` data source by setting *Filter by* to `Tags`.

Grafana v8.1 and later versions also support typeahead of existing tags, provide at least one tag.

For example, create an annotation query name `outages` and specify a tag `outage`. This query will show all annotations (from any dashboard or via API) with the `outage` tag. If multiple tags are defined in an annotation query, then Grafana will only show annotations matching all the tags. To modify the behavior, enable `Match any`, and Grafana will show annotations that contain any one of the tags you provided.

You can also use template variables in the tag query. This means if you have a dashboard showing stats for different services and a template variable that dictates which services to show, you can use the same template variable in your annotation query to only show annotations for those services.

Add time regions

When adding or editing an annotation, you can define a repeating time region by setting **Query type** to **Time regions**. Then, define the **From** and **To** sections with the preferred days of the week and time. You also have the option to change the timezone, which is set to the dashboard's timezone, by default.

The above configuration will produce the following result in the Time series panel:

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Dashboard JSON model

A dashboard in Grafana is represented by a JSON object, which stores metadata of its dashboard. Dashboard metadata includes dashboard properties, metadata from panels, template variables, panel queries, etc.

To view the JSON of a dashboard:

1. Navigate to a dashboard.
2. In the top navigation menu, click the **Dashboard settings** (gear) icon.
3. Click **JSON Model**.

JSON fields

When a user creates a new dashboard, a new dashboard JSON object is initialized with the following fields:

> **NOTE**
>
> In the following JSON, id is shown as null which is the default value assigned to it until a dashboard is saved. Once a dashboard is saved, an integer value is assigned to the `id` field.

json Copy

```
{
  "id": null,
  "uid": "cLV5GDCkz",
  "title": "New dashboard",
  "tags": [],
  "timezone": "browser",
```

```
  "editable": true,
  "graphTooltip": 1,
  "panels": [],
  "time": {
    "from": "now-6h",
    "to": "now"
  },
  "timepicker": {
    "time_options": [],
    "refresh_intervals": []
  },
  "templating": {
```

Expand code

Each field in the dashboard JSON is explained below with its usage:

Name	Usage
id	unique numeric identifier for the dashboard. (generated by the db)
uid	unique dashboard identifier that can be generated by anyone. string (8-40)
title	current title of dashboard
tags	tags associated with dashboard, an array of strings
style	theme of dashboard, i.e. `dark` or `light`
timezone	timezone of dashboard, i.e. `utc` or `browser`
editable	whether a dashboard is editable or not
graphTooltip	0 for no shared crosshair or tooltip (default), 1 for shared crosshair, 2 for shared crosshair AND shared tooltip
time	time range for dashboard, i.e. last 6 hours, last 7 days, etc
timepicker	timepicker metadata, see timepicker section for details
templating	templating metadata, see templating section for details
annotations	annotations metadata, see annotations for how to add them
refresh	auto-refresh interval
schemaVersion	version of the JSON schema (integer), incremented each time a Grafana update brings changes to said schema
version	version of the dashboard (integer), incremented each time the dashboard is updated
panels	panels array, see below for detail.

Panels

Panels are the building blocks of a dashboard. It consists of data source queries, type of graphs, aliases, etc. Panel JSON consists of an array of JSON objects, each representing a different panel. Most of the fields are common for all panels but some fields depend on the panel type. Following is an example of panel JSON of a text panel.

```json
"panels": [
  {
    "type": "text",
    "title": "Panel Title",
    "gridPos": {
      "x": 0,
      "y": 0,
      "w": 12,
      "h": 9
    },
    "id": 4,
    "mode": "markdown",
    "content": "# title"
  }
```

Panel size and position

The gridPos property describes the panel size and position in grid coordinates.

- `w` 1-24 (the width of the dashboard is divided into 24 columns)
- `h` In grid height units, each represents 30 pixels.
- `x` The x position, in same unit as `w`.
- `y` The y position, in same unit as `h`.

The grid has a negative gravity that moves panels up if there is empty space above a panel.

timepicker

```json
"timepicker": {
    "collapse": false,
    "enable": true,
```

```
    "notice": false,
    "now": true,
    "hidden": false,
    "nowDelay": "",
    "time_options": [
      "5m",
      "15m",
      "1h",
      "6h",
      "12h",
      "24h",
      "2d",
      "7d",
      "30d"
    ],
```

Expand code

Usage of the fields is explained below:

Name	Usage
collapse	whether timepicker is collapsed or not
enable	whether timepicker is enabled or not
notice	
now	
hidden	whether timepicker is hidden or not
nowDelay	override the now time by entering a time delay. Use this option to accommodate known delays in data aggregation to avoid null values.
time_options	options available in the time picker dropdown
refresh_intervals	interval options available in the refresh picker dropdown
status	
type	

templating

The `templating` field contains an array of template variables with their saved values along with some other metadata, for example:

```
json
```

Copy

```
"templating": {
   "enable": true,
   "list": [
     {
       "allFormat": "wildcard",
       "current": {
         "tags": [],
         "text": "prod",
         "value": "prod"
       },
       "datasource": null,
       "includeAll": true,
       "name": "env",
       "options": [
         {
           "selected": false,
           "text": "All",
           "value": "*"
         },
```

Expand code

Usage of the above mentioned fields in the templating section is explained below:

Name	Usage
enable	whether templating is enabled or not
list	an array of objects each representing one template variable
allFormat	format to use while fetching all values from data source, eg: `wildcard`, `glob`, `regex`, `pipe`, etc.
current	shows current selected variable text/value on the dashboard
data source	shows data source for the variables
includeAll	whether all value option is available or not
multi	whether multiple values can be selected or not from variable value list
multiFormat	format to use while fetching timeseries from data source
name	name of variable
options	array of variable text/value pairs available for selection on dashboard
query	data source query used to fetch values for a variable
refresh	configures when to refresh a variable
regex	extracts part of a series name or metric node segment

Name	Usage
type	type of variable, i.e. `custom`, `query` or `interval`

Grafana dashboard best practices

This section provides information about best practices for intermediate Grafana administrators and users about how to build and maintain Grafana dashboards.

For more information about the different kinds of dashboards you can create, refer to Grafana dashboards: A complete guide to all the different types you can build.

Common observability strategies

When you have a lot to monitor, like a server farm, you need a strategy to decide what is important enough to monitor. This page describes several common methods for choosing what to monitor.

A logical strategy allows you to make uniform dashboards and scale your observability platform more easily.

Guidelines for usage

- The USE method tells you how happy your machines are, the RED method tells you how happy your users are.
- USE reports on causes of issues.
- RED reports on user experience and is more likely to report symptoms of problems.
- The best practice of alerting is to alert on symptoms rather than causes, so alerting should be done on RED dashboards.

USE method

USE stands for:

- **Utilization -** Percent time the resource is busy, such as node CPU usage
- **Saturation -** Amount of work a resource has to do, often queue length or node load

- **Errors -** Count of error events

This method is best for hardware resources in infrastructure, such as CPU, memory, and network devices. For more information, refer to The USE Method.

RED method

RED stands for:

- **Rate -** Requests per second
- **Errors -** Number of requests that are failing
- **Duration -** Amount of time these requests take, distribution of latency measurements

This method is most applicable to services, especially a microservices environment. For each of your services, instrument the code to expose these metrics for each component. RED dashboards are good for alerting and SLAs. A well-designed RED dashboard is a proxy for user experience.

For more information, refer to Tom Wilkie's blog post The RED method: How to instrument your services.

The Four Golden Signals

According to the Google SRE handbook, if you can only measure four metrics of your user-facing system, focus on these four.

This method is similar to the RED method, but it includes saturation.

- **Latency -** Time taken to serve a request
- **Traffic -** How much demand is placed on your system
- **Errors -** Rate of requests that are failing
- **Saturation -** How "full" your system is

> **Give it a try using Grafana Play**
>
> With Grafana Play, you can explore and see how it works, learning from practical examples to accelerate your development. This feature can be seen on The Four Golden Signals.

Dashboard management maturity model

Dashboard management maturity refers to how well-designed and efficient your dashboard ecosystem is. We recommend periodically reviewing your dashboard setup to gauge where you are and how you can improve.

Broadly speaking, dashboard maturity can be defined as low, medium, or high.

Much of the content for this topic was taken from the KubeCon 2019 talk Fool-Proof Kubernetes Dashboards for Sleep-Deprived Oncalls.

Low - default state

At this stage, you have no coherent dashboard management strategy. Almost everyone starts here.

How can you tell you are here?

- Everyone can modify your dashboards.
- Lots of copied dashboards, little to no dashboard reuse.
- One-off dashboards that hang around forever.
- No version control (dashboard JSON in version control).
- Lots of browsing for dashboards, searching for the right dashboard. This means lots of wasted time trying to find the dashboard you need.
- Not having any alerts to direct you to the right dashboard.

Medium - methodical dashboards

At this stage, you are starting to manage your dashboard use with methodical dashboards. You might have laid out a strategy, but there are some things you could improve.

How can you tell you are here?

- Prevent sprawl by using template variables. For example, you don't need a separate dashboard for each node, you can use query variables. Even better, you can make the data source a template variable too, so you can reuse the same dashboard across different clusters and monitoring backends.

  Refer to the list of Variable examples if you want some ideas.
- Methodical dashboards according to an observability strategy.
- Hierarchical dashboards with drill-downs to the next level.
- Dashboard design reflects service hierarchies. The example shown below uses the RED method (request and error rate on the left, latency duration on the right) with one row per service. The row order reflects the data flow.
- Compare like to like: split service dashboards when the magnitude differs. Make sure aggregated metrics don't drown out important information.
- Expressive charts with meaningful use of color and normalizing axes where you can.
  - Example of meaningful color: Blue means it's good, red means it's bad. Thresholds can help with that.
  - Example of normalizing axes: When comparing CPU usage, measure by percentage rather than raw number, because machines can have a different number of cores. Normalizing CPU usage by the number of cores reduces cognitive load because the viewer can trust that at 100% all cores are being used, without having to know the number of CPUs.
- Directed browsing cuts down on "guessing."

Example of using drill-down

  - Template variables make it harder to “just browse” randomly or aimlessly.
  - Most dashboards should be linked to by alerts.
  - Browsing is directed with links. For more information, refer to Manage dashboard links.
- Version-controlled dashboard JSON.

High - optimized use

At this stage, you have optimized your dashboard management use with a consistent and thoughtful strategy. It requires maintenance, but the results are worth it.

- Actively reducing sprawl.
  - Regularly review existing dashboards to make sure they are still relevant.
  - Only approved dashboards added to master dashboard list.
  - Tracking dashboard use. If you’re an Enterprise user, you can take advantage of Usage insights.
- Consistency by design.
- Use scripting libraries to generate dashboards, ensure consistency in pattern and style.
  - grafonnet (Jsonnet)
  - grafanalib (Python)
- No editing in the browser. Dashboard viewers change views with variables.
- Browsing for dashboards is the exception, not the rule.

Example of a service hierarchy

- Perform experimentation and testing in a separate Grafana instance dedicated to that purpose, not your production instance. When a dashboard in the test environment is proven useful, then add that dashboard to your main Grafana instance.

Best practices for creating dashboards

This page outlines some best practices to follow when creating Grafana dashboards.

Before you begin

Here are some principles to consider before you create a dashboard.

A dashboard should tell a story or answer a question

What story are you trying to tell with your dashboard? Try to create a logical progression of data, such as large to small or general to specific. What is the goal for this dashboard? (Hint: If the dashboard doesn't have a goal, then ask yourself if you really need the dashboard.)

Keep your graphs simple and focused on answering the question that you are asking. For example, if your question is "which servers are in trouble?", then maybe you don't need to show all the server data. Just show data for the ones in trouble.

Dashboards should reduce cognitive load, not add to it

Cognitive load is basically how hard you need to think about something in order to figure it out. Make your dashboard easy to interpret. Other users and future you (when you're trying to figure out what broke at 2AM) will appreciate it.

Ask yourself:

- Can I tell what exactly each graph represents? Is it obvious, or do I have to think about it?
- If I show this to someone else, how long will it take them to figure it out? Will they get lost?

Have a monitoring strategy

It's easy to make new dashboards. It's harder to optimize dashboard creation and adhere to a plan, but it's worth it. This strategy should govern both your overall dashboard scheme and enforce consistency in individual dashboard design.

Refer to Common observability strategies and Dashboard management maturity levels for more information.

Write it down

Once you have a strategy or design guidelines, write them down to help maintain consistency over time. Check out this Wikimedia runbook example.

Best practices to follow

- When creating a new dashboard, make sure it has a meaningful name.
  - If you are creating a dashboard to play or experiment, then put the word `TEST` or `TMP` in the name.
  - Consider including your name or initials in the dashboard name or as a tag so that people know who owns the dashboard.
  - Remove temporary experiment dashboards when you are done with them.
- If you create many related dashboards, think about how to cross-reference them for easy navigation. Refer to Best practices for managing dashboards for more information.

- Grafana retrieves data from a data source. A basic understanding of data sources in general and your specific is important.
- Avoid unnecessary dashboard refreshing to reduce the load on the network or backend. For example, if your data changes every hour, then you don't need to set the dashboard refresh rate to 30 seconds.
- Use the left and right Y-axes when displaying time series with different units or ranges.
- Add documentation to dashboards and panels.
  - To add documentation to a dashboard, add a Text panel visualization to the dashboard. Record things like the purpose of the dashboard, useful resource links, and any instructions users might need to interact with the dashboard. Check out this Wikimedia example.
  - To add documentation to a panel, edit the panel settings and add a description. Any text you add will appear if you hover your cursor over the small `i` in the top left corner of the panel.
- Reuse your dashboards and enforce consistency by using templates and variables.
- Be careful with stacking graph data. The visualizations can be misleading, and hide important data. We recommend turning it off in most cases.

Best practices for managing dashboards

This page outlines some best practices to follow when managing Grafana dashboards.

Before you begin

Here are some principles to consider before you start managing dashboards.

Strategic observability

There are several common observability strategies. You should research them and decide whether one of them works for you or if you want to come up with your own. Either way, have a plan, write it down, and stick to it.

Adapt your strategy to changing needs as necessary.

Maturity level

What is your dashboard maturity level? Analyze your current dashboard setup and compare it to the Dashboard management maturity model. Understanding where you are can help you decide how to get to where you want to be.

Best practices to follow

- Avoid dashboard sprawl, meaning the uncontrolled growth of dashboards. Dashboard sprawl negatively affects time to find the right dashboard. Duplicating dashboards and changing "one thing" (worse: keeping original tags) is the easiest kind of sprawl.
  - Periodically review the dashboards and remove unnecessary ones.

    - If you create a temporary dashboard, perhaps to test something, prefix the name with `TEST:` . Delete the dashboard when you are finished.
- Copying dashboards with no significant changes is not a good idea.
    - You miss out on updates to the original dashboard, such as documentation changes, bug fixes, or additions to metrics.
    - In many cases copies are being made to simply customize the view by setting template parameters. This should instead be done by maintaining a link to the master dashboard and customizing the view with URL parameters.
- When you must copy a dashboard, clearly rename it and *do not* copy the dashboard tags. Tags are important metadata for dashboards that are used during search. Copying tags can result in false matches.
- Maintain a dashboard of dashboards or cross-reference dashboards. This can be done in several ways:
    - Create dashboard links, panel, or data links. Links can go to other dashboards or to external systems. For more information, refer to Manage dashboard links.
    - Add a Dashboard list panel. You can then customize what you see by doing tag or folder searches.
    - Add a Text panel and use markdown to customize the display.

Manage dashboards

On the **Dashboards** page, you can perform dashboard management tasks such as:

- Browsing and creating dashboard folders
- Managing folder permissions
- Adding generative AI features to dashboards

For more information about creating dashboards, refer to Build dashboards.

Browse dashboards

On the **Dashboards** page, you can browse and manage folders and dashboards. This includes the options to:

- Create folders and dashboards.
- Move dashboards between folders.
- Delete multiple dashboards and folders.
- Navigate to a folder.
- Manage folder permissions. For more information, refer to Dashboard permissions.

The page lists all the dashboards to which you have access, grouped into folders. Dashboards without a folder are displayed at the top level alongside folders.

Shared with me

The **Shared with me** section displays folders and dashboards that are directly shared with you. These folders and dashboards aren't shown in the main list because you don't have access to one or more of their parent folders.

If you have permission to view all folders, you won't see a **Shared with me**.

Create a dashboard folder

Folders help you organize and group dashboards, which is useful when you have many dashboards or multiple teams using the same Grafana instance.

Before you begin: Ensure you have Editor permissions or greater to create folders. For more information about dashboard permissions, refer to Dashboard permissions.

To create a dashboard folder:

1. Click **Dashboards** in the primary menu.
2. Do one of the following:
   - On the **Dashboards** page, click **New** and select **New folder** in the drop-down.
   - Click an existing folder and on the folder's page, click **New** and select **New folder** in the drop-down.
3. Enter a unique name and click **Create**.

When you nest folders, you can do so up to four levels deep.

When you save a dashboard, you can optionally select a folder to save the dashboard in.

> **NOTE**
>
> Alerts can't be placed in folders with slashes (\ /) in the name. If you wish to place alerts in the folder, don't use slashes in the folder name.

To edit the name of a folder:

1. Click **Dashboards** in the primary menu.
2. Navigate to the folder by selecting it in the list, or searching for it.
3. Click the **Edit title** icon (pencil) in the header and update the name of the folder.

The new folder name is automatically saved.

Folder permissions

You can assign permissions to a folder. Dashboards in the folder inherit any permissions that you've assigned to the folder. You can assign permissions to organization roles, teams, and users.

To modify permissions for a folder:

1. Click **Dashboards** in the primary menu.
2. Navigate to the folder by selecting it in the list, or searching for it.

3. On the folder's page, click **Folder actions** and select **Manage permissions** in the drop-down.
4. Update the permissions as desired.

Changes are saved automatically.

For more information about dashboard permissions, refer to Dashboard permissions.

Set up generative AI features for dashboards

> **NOTE**
>
> Generative AI in dashboards is currently in public preview. Grafana Labs offers limited support, and breaking changes might occur prior to the feature being made generally available. Enable the `dashgpt` feature toggle in Grafana to use this feature. Contact Grafana Support to enable this feature in Grafana Cloud.

You can use generative AI to help you with the following tasks:

- **Generate panel and dashboard titles and descriptions**: Generate a title and description based on the data you've added for your panel or dashboard. This is useful when you want to visualize your data quickly and don't want to spend time coming up with a title or description.
- **Generate dashboard save changes summary**: Generate a summary of the changes you've made to a dashboard when you save it. This is great for easily tracking the history of a dashboard.

To access these features, enable the `dashgpt` feature toggle. Then install and configure Grafana's Large Language Model (LLM) app plugin. For more information, refer to the Grafana LLM plugin documentation.

When enabled, the ✨ **Auto generate** option displays next to the **Title** and **Description** fields in your panels and dashboards, or when you press the **Save** button.

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Grafana Cloud | Enterprise | Open source

Public dashboards

WARNING

Making your dashboard public could result in a large number of queries to the data sources used by your dashboard. This can be mitigated by utilizing the enterprise caching and/or rate limiting features.

Public dashboards allow you to share your Grafana dashboard with anyone. This is useful when you want to make your dashboard available to the world without requiring access to your Grafana organization. This differs from dashboard sharing, which either requires recipients to be users in the same Grafana organization or provides limited information, as with a snapshot.

You can see a list of all your public dashboards in one place by navigating to **Dashboards > Public dashboards**. For each dashboard in the list, the page displays the status, a link to view the dashboard, a link to the public dashboard configuration, and the option to revoke the public URL.

Security implications of making your dashboard public

- Anyone with the URL can access the dashboard.
- Public dashboards are read-only.
- Arbitrary queries **cannot** be run against your data sources through public dashboards. Public dashboards can only execute the queries stored on the original dashboard.

Make a dashboard public

1. Click the sharing icon in the dashboard header.
2. Click the **Public dashboard** tab.

3. Acknowledge the implications of making the dashboard public by selecting all the checkboxes.
4. Click **Generate public URL** to make the dashboard public and make your link live.
5. Copy the public dashboard link if you'd like to share it. You can always come back later for it.

Once you've made the dashboard public, a **Public** tag is displayed in the header of the dashboard.

Pause access

1. Click the sharing icon in the dashboard header.
2. Click the **Public dashboard** tab.
3. Enable the **Pause sharing dashboard** toggle.

The dashboard is no longer accessible, even with the link, until you make it shareable again.

Revoke access

1. Click the sharing icon in the dashboard header.
2. Click the **Public dashboard** tab.
3. Click **Revoke public URL** to delete the public dashboard.

The link no longer works. You must create a new public URL, as in Make a dashboard public.

Email sharing

> **NOTE**
>
> Available in private preview in Grafana Cloud. This feature will have a cost by active users after being promoted into general availability.
>
> Please contact support to have the feature enabled.

Email sharing allows you to share your public dashboard with only specific people by email, instead of having it accessible to anyone with the URL. When you use email sharing, recipients receive a one-time use link that's valid for *one hour*. Once the link is used, the viewer has access to the public dashboard for *30 days*.

Invite a viewer

1. Click the sharing icon in the dashboard header.

2. Click the **Public dashboard** tab.
3. Acknowledge the implications of making the dashboard public by selecting all the checkboxes.
4. Click **Generate public URL** to make the dashboard public and make your link live.
5. Under Can view dashboard, click **Only specified people**.
6. Enter the email you want to share the public dashboard with.
7. Click **Invite**.
8. The recipient will receive an email with a one-time use link.

Viewers requesting access

If a viewer without access tries to navigate to the public dashboard, they'll be asked to request access by providing their email. They will receive an email with a new one-time use link if the email they provided has already been invited to view the public dashboard and has not been revoked.

If the viewer doesn't have an invitation or it's been revoked, you won't be notified and no link is sent.

Revoke access for a viewer

1. Click the sharing icon in the dashboard header.
2. Click the **Public dashboard** tab.
3. Click **Revoke** on the viewer you'd like to revoke access for.

Immediately, the viewer no longer has access to the public dashboard, nor can they use any existing one-time use links they may have.

Reinvite a viewer

1. Click the sharing icon in the dashboard header.
2. Click the **Public dashboard** tab.
3. Click **Resend** on the viewer you'd like to re-share the public dashboard with.

The viewer will receive an email with a new one-time use link. This will invalidate all previously issued links for that viewer.

View public dashboard users

To see a list of users who have accessed your dashboard by way of email sharing, take the following steps:

1. In the main sidebar navigation, click **Administration**.

2. Click **Users**.
3. Click the **Public dashboard users** tab.

From here, you can see the earliest time a user has been active in a dashboard, which public dashboards they have access to, and their role.

Access limitations

One-time use links use browser cookies, so when a viewer is granted access through one of these links, they will only have access on the browser they used to claim the link.

Multiple valid one-time use links can't be generated for a single viewer. When a new one-time use link is issued for a viewer, all previous ones are invalidated.

If a Grafana user has read access to the parent dashboard, they can view the public dashboard without needing to have access granted.

Assess public dashboard usage

> **NOTE**
>
> Available in Grafana Enterprise and Grafana Cloud.

You can check usage analytics about your public dashboard by clicking the insights icon in the dashboard header:

Learn more about the kind of information provided in the dashboard insights documentation.

Supported data sources

Public dashboards *should* work with any data source that has the properties `backend` and `alerting` both set to true in its `plugin.json`. However, this can't always be guaranteed because plugin developers can override this functionality. The following lists include data sources confirmed to work with public dashboards and data sources that should work, but have not been confirmed as compatible.

Confirmed:

- ClickHouse
- MongoDB

- CloudWatch
- Elasticsearch
- Infinity
- InfluxDB
- Loki
- Microsoft SQL Server
- MySQL
- Oracle Database
- PostgreSQL
- Prometheus
- Redis
- SQLite

Unsupported:

- Graphite

Unconfirmed:

- Altinity plugin for ClickHouse
- Amazon Athena
- Amazon Redshift
- Amazon Timestream
- Apache Cassandra
- AppDynamics
- Azure Data Explorer Datasource
- Azure Monitor
- CSV
- DB2 Datasource
- Databricks
- Datadog
- Dataset
- Druid
- Dynatrace
- GitHub
- Google BigQuery
- Grafana for YNAB
- Honeycomb
- Jira
- Mock
- Neo4j Datasource
- New Relic
- OPC UA (Unified Architecture)
- Open Distro for Elasticsearch
- OpenSearch
- OpenTSDB
- Orbit
- SAP HANA®
- Salesforce
- Sentry
- ServiceNow
- Snowflake
- Splunk
- Splunk Infrastructure Monitoring
- Sqlyze data source
- TDengine
- Vertica
- Wavefront
- X-Ray
- kdb+
- simple grpc data source

Limitations

- Panels that use frontend data sources will fail to fetch data.
- Template variables are not supported.
- Exemplars will be omitted from the panel.
- Only annotations that query the `-- Grafana --` data source are supported.
- Organization annotations are not supported.
- Grafana Live and real-time event streams are not supported.
- Library panels are not supported.
- Data sources using Reverse Proxy functionality are not supported.

Custom branding

If you're a Grafana Enterprise customer, you can use custom branding to change the appearance of a public dashboard footer. For more information, refer to Configure custom branding.

Manage playlists

A *playlist* is a list of dashboards that are displayed in a sequence. You might use a playlist to build situational awareness or to present your metrics to your team or visitors.

Grafana automatically scales dashboards to any resolution, which makes them perfect for big screens.

You can access the Playlist feature from Grafana's side menu, in the Dashboards submenu.

> **NOTE**
>
> You must have at least Editor role permissions to create and manage playlists.

Access, share, and control a playlist

Use the information in this section to access existing playlists. Start and control the display of a playlist using one of the five available modes.

Access a playlist

1. Click **Dashboards** in the left-side menu.
2. Click **Playlists** to see a list of existing playlists.

Start a playlist

You can start a playlist in five different view modes. View modes determine how the menus and navigation bar appear on the dashboards.

By default, each dashboard is displayed for the amount of time entered in the Interval field, which you set when you create or edit a playlist. After you start a playlist, you can control it with the navbar at the top of the page.

1. Click **Dashboards** in the left-side menu.
2. Click **Playlists** to see a list of existing playlists.
3. Find the playlist you want to start, then click **Start playlist**.
4. In the modal that opens, select one of the five playlist modes available, based on the information in the table below.
5. Click **Start <playlist name>**.

The playlist displays each dashboard for the time specified in the **Interval** field, set when creating or editing a playlist. Once a playlist starts, you can control it using the navbar at the top of your screen.

Mode	Description
Normal mode	• The side menu remains visible. • The navbar, row, and panel controls appear at the top of the screen.
TV mode	• The side menu and dashboard submenu (including variable drop-downs and dashboard links) are hidden or removed. • The navbar, row, and panel controls appear at the top of the screen. • Enabled automatically after one minute of user inactivity. • Enable it manually using the `d v` sequence shortcut, or by appending the parameter `?inactive` to the dashboard URL. • Disable it with any mouse movement or keyboard action.
TV mode (with auto fit panels)	• The side menu and dashboard submenu (including variable drop-downs and dashboard links) are hidden or removed. • The navbar, row and panel controls appear at the top of the screen. • Dashboard panels automatically adjust to optimize space on screen.
Kiosk mode	• The side menu, navbar, ro and panel controls are completely hidden/removed from view.

Mode	Description
	• You can enable it manually using the `d v` sequence shortcut after the playlist has started. • You can disable it manually with the same shortcut.
Kiosk mode (with auto fit panels)	• The side menu, navbar, row, and panel controls are completely hidden/removed from view. • Dashboard panels automatically adjust to optimize space on screen.

Control a playlist

You can control a playlist in **Normal** or **TV** mode after it's started, using the navigation bar at the top of your screen. Press the Esc key in your keyboard to stop the playlist.

Button	Result
Next (double-right arrow)	Advances to the next dashboard.
Back (left arrow)	Returns to the previous dashboard.
Stop (square)	Ends the playlist, and exits to the current dashboard.
Cycle view mode (monitor icon)	Rotates the display of the dashboards in different view modes.
Time range	Displays data within a time range. It can be set to display the last 5 minutes up to 5 years ago, or a custom time range, using the down arrow.
Refresh (circle arrow)	Reloads the dashboard, to display the current data. It can be set to reload automatically every 5 seconds to 1 day, using the drop-down arrow.

Create a playlist

You can create a playlist to present dashboards in a sequence, with a set order and time interval between dashboards.

1. Click **Dashboards** in the left-side menu.
2. Click **Playlists** to see a list of existing playlists.
3. Click **New playlist**. The New playlist page opens.
4. In the **Name** text box, enter a descriptive name.

5. In the **Interval** text box, enter a time interval. Grafana displays a particular dashboard for the interval of time specified here before moving on to the next dashboard.
6. In Dashboards, add existing dashboards to the playlist using **Add by title** and **Add by tag** drop-down options. The dashboards you add are listed in a sequential order.
7. If needed:
   - Search for a dashboard by its name, a regular expression, or a tag.
   - Filter your results by starred status or tags.
8. If needed, rearrange the order of the dashboard you have added using the up and down arrow icon.
9. Optionally, remove a dashboard from the playlist by clicking the x icon beside dashboard.
10. Click **Save**.

Save a playlist

You can save a playlist and add it to your **Playlists** page, where you can start it. Be sure that all the dashboards you want to appear in your playlist are added when creating or editing the playlist before saving it.

1. Click **Dashboards** in the left-side menu.
2. Click **Playlists** to see a list of existing playlists.
3. Click on the playlist.
4. Edit the playlist.
5. Ensure that your playlist has a **Name**, **Interval**, and at least one **Dashboard** added to it.
6. Click **Save**.

Edit or delete a playlist

You can edit a playlist by updating its name, interval time, and by adding, removing, and rearranging the order of dashboards. On the rare occasion when you no longer need a playlist, you can delete it.

Edit a playlist

1. Click **Dashboards** in the left-side menu.
2. Click **Playlists** to see a list of existing playlists.
3. Find the playlist you want to update and click **Edit playlist**.
4. Update the name and time interval, then add or remove dashboards from the playlist using instructions in Create a playlist.

5. Click **Save** to save your changes.

Delete a playlist

1. Click **Dashboards** in the left-side menu.
2. Click **Playlists** to see a list of existing playlists.
3. Find the playlist you want to remove.
4. Click **Delete playlist**.

Rearrange dashboard order

1. Click **Dashboards** in the left-side menu.
2. Click **Playlists** to see a list of existing playlists.
3. Find the playlist you want to update and click **Edit playlist**.
4. Click and drag the dashboards into your desired order.
5. Click **Save** to save your changes.

Remove a dashboard

1. Click **Dashboards** in the left-side menu.
2. Click **Playlists** to see a list of existing playlists.
3. Find the playlist you want to update and click **Edit playlist**.
4. Click **[x]** on the name of the dashboard you want to remove from the playlist.
5. Click **Save** to save your changes.

Share a playlist in a view mode

You can share a playlist by copying the link address on the view mode you prefer, and pasting the URL to your destination.

1. Click **Dashboards** in the left-side menu.
2. Click **Playlists** to see a list of existing playlists.
3. Click the share icon of the playlist you want to share.
4. Select the view mode you prefer.
5. Click **Copy** next to the Link URL to copy it to your clipboard.

For example, the URL for the first playlist on the Grafana Play site in Kiosk mode will look like this:

https://play.grafana.org/playlists/play/1?kiosk.

6. Paste the URL to your destination.

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Grafana Cloud | Enterprise

Create and manage reports

Reporting enables you to automatically generate PDFs from any of your dashboards and have Grafana email them to interested parties on a schedule. This is available in Grafana Cloud and in Grafana Enterprise.

If you have Role-based access control enabled, for some actions you would need to have relevant permissions. Refer to specific guides to understand what permissions are required.

Any changes you make to a dashboard used in a report are reflected the next time the report is sent. For example, if you change the time range in the dashboard, then the time range in the report also changes, unless you've configured a custom time range.

For information about recent improvements to the reporting UI, refer to Grafana reporting: How we improved the UX in Grafana.

Requirements

- SMTP must be configured for reports to be sent. Refer to SMTP in Configuration for more information.
- The Image Renderer plugin must be installed or the remote rendering service must be set up. Refer to Image rendering for more information.

Access control

When RBAC is enabled, you need to have the relevant [Permissions][] to create and manage reports.

Create or update a report

Only organization administrators can create reports by default. You can customize who can create reports with Role-based access control.

1. Click **Dashboards > Reports** in the side navigation menu.

   The Reports page allows you to view, create, and update your reports. The report form has a multi-step layout. The steps do not need to be completed in succession and can be skipped over by clicking a step name.

2. Click **+ Create a new report**.

3. Select report dashboard.

   - **Source dashboard:** Select the dashboard from which you want to generate the report.
   - **Time range:** (optional) Use custom time range for the report. For more information, refer to Report time range.
   - **Add another dashboard:** Add more than one dashboard to the report.

4. Format the report.

   - **Choose format options for the report:** Select at least one option. Attach report as PDF, embed dashboard as an image, or attach CSV file of table panel data.
   - If you selected the PDF format option:
     - Select an orientation for the report: **Portrait** or **Landscape**.
     - Select a layout for the generated report: **Simple** or **Grid**. The simple layout renders each panel as full-width across the PDF. The grid layout renders the PDF with the same panel arrangement and width as the source dashboard.
     - Click **Preview PDF** to view a rendered PDF with the options you selected.

5. Schedule report.

   - Enter scheduling information. Options vary depending on the frequency selected.

6. Enter report information. All fields are required unless otherwise indicated.

   - **Report name:** Name of the report as you want it to appear in the **Reports** list. The report name populates the email subject line.
   - **Recipients:** Enter the emails of the people or teams that you want to receive the report, separated by commas or semicolons.
   - **Reply to:** (optional) The address that appears in the **Reply to** field of the email.
   - **Message:** (optional) Message body in the email with the report.
   - **Include a dashboard link:** Include a link to the dashboard from within the report email.
   - **Send test email:** To verify that the configuration works as expected. You can choose to send this email to the recipients configured for the report, or to a different set of email addresses only used for testing.

7 Preview and save the report.

Save as draft

Note: Available in Grafana Enterprise version 9.1.0 and later and Grafana Cloud.

You can save a report as a draft at any point during the report creation or update process. You can save a report as a draft even if it's missing required fields. Also, the report won't be sent according to its schedule while it's a draft.

Choose template variables

Note: Available in Grafana Enterprise version 7.5 and later behind the `reportVariables` feature flag, Grafana Enterprise version 8.0 and later without a feature flag, and Grafana Cloud.

You can configure report-specific template variables for the dashboard on the report page. The variables that you select will override the variables from the dashboard, and they are used when rendering a PDF file of the report. For detailed information about using template variables, refer to the Templates and variables section.

NOTE

The query variables saved with a report might become of date if the results of that query change. For example, if your template variable queries for a list of hostnames and a new hostname is added, then it will not be included in the report. If that occurs, the selected variables must be manually updated in the report. If you select the `All` value for the template variable or if you keep the dashboard's original variable selection, then the report stays up-to-date as new values are added.

Render a report with panels or rows set to repeat by a variable

Note: Available in Grafana Enterprise version 8.0 and later, and Grafana Cloud.

You can include dynamic dashboards with panels or rows, set to repeat by a variable, into reports. For detailed information about setting up repeating panels or rows in dashboards, refer to Repeat panels or rows.

Caveats

- Rendering repeating panels for dynamic variable types (for example, `query` variables) with selected `All` value is currently not supported. As a workaround, select all the values.
- If you select different template variables in a report for a dashboard with repeating rows, you might see empty space or missing values at the bottom of the report. This is because the dimensions of the panels from the dashboard are used to generate the report. To avoid this issue
  - use the dashboard's original template variables for the report, or make a copy of the dashboard
  - select the new set of template variables
  - generate a report based on the copied dashboard.
- Rendering of the repeating panels inside collapsed rows in reports is not supported.

Report time range

> **Note:** You can set custom report time ranges in Grafana Enterprise 7.2+ and Grafana Cloud.

By default, reports use the saved time range of the dashboard. You can change the time range of the report by:

- Saving a modified time range to the dashboard. Changing the dashboard time range without saving it doesn't change the time zone of the report.
- Setting a time range via the **Time range** field in the report form. If specified, the custom time range overrides the time range from the report's dashboard.

The page header of the report displays the time range for the dashboard's data queries.

Report time zones

Reports use the time zone of the dashboard from which they're generated. You can control the time zone for your reports by setting the dashboard to a specific time zone. Note that this affects the display of the dashboard for all users.

If a dashboard has the **Browser Time** setting, the reports generated from that dashboard use the time zone of the Grafana server. As a result, this time zone might not match the time zone of users creating or receiving the report.

If the time zone is set differently between your Grafana server and its remote image renderer, then the time ranges in the report might be different between the page header and the time axes in the panels. To avoid this, set the time zone to UTC for dashboards when using a remote renderer. Each dashboard's time zone setting is visible in the time range controls.

Layout and orientation

Layout	Orientation	Support	Description	Preview
Simple	Portrait	v6.4+	Generates an A4 page in portrait mode with three panels per page.	
Simple	Landscape	v6.7+	Generates an A4 page in landscape mode with a single panel per page.	
Grid	Portrait	v7.2+	Generates an A4 page in portrait mode with panels arranged in the same way as at the original dashboard.	
Grid	Landscape	v7.2+	Generates an A4 page in landscape mode with panels arranged in the same way as in the original dashboard.	

CSV export

Note: Available in Grafana Enterprise 8+ with the Grafana image renderer plugin v3.0+, and Grafana Cloud.

You can attach a CSV file to the report email for each table panel on the selected dashboard, along with the PDF report. By default, CSVs larger than 10Mb are not sent which keeps email servers from rejecting the email. You can increase or decrease this limit in the reporting configuration.

This feature relies on the same plugin that supports the image rendering features.

When the CSV file is generated, it is temporarily written to the `csv` folder in the Grafana `data` folder.

A background job runs every 10 minutes and removes temporary CSV files. You can configure how long a CSV file should be stored before being removed by configuring the temp-data-lifetime setting. This setting also affects how long a renderer PNG file should be stored.

Table data in PDF

> **NOTE**
>
> Available in public preview (`pdfTables` feature toggle) in Grafana Enterprise v10.3+ with the Grafana image renderer plugin v3.0+, and Grafana Cloud.

When there's more data in your table visualizations than can be shown in the dashboard PDF, you can select one of these two options to access all table visualization data as PDF in your reports:

- **Include table data as PDF appendix** - Adds an appendix to the main dashboard PDF.
- **Attach a separate PDF of table data** - Generates a separate PDF file.

This feature relies on the same plugin that supports the image rendering features.

Scheduling

Note: Available in Grafana Enterprise version 8.0 and later, and Grafana Cloud. The scheduler was significantly changed in Grafana Enterprise version 8.1.

Scheduled reports can be sent once, or repeated on an hourly, daily, weekly, or monthly basis, or sent at custom intervals. You can also disable scheduling by selecting **Never**, for example to send the report via the API.

Send now or schedule for later

- **Send now** sends the report immediately after you save it. To stop sending the report at some point in the future, add an end date. If you leave the end date empty, the report is sent out indefinitely.
- **Send later** schedules a report for a later date. Thus, the start date and time are required fields. If you leave the end date empty, the report is sent out indefinitely.

Send only from Monday to Friday

For reports that have an hourly or daily frequency, you can choose to send them only from Monday to Friday.

Send on the last day of the month

When you schedule a report with a monthly frequency, and set the start date between the 29th and the 31st of the month, the report is only sent during the months that have those dates. If you want the report to be sent every month, select the **Send on the last day of the month** option instead. This way, the report is sent on the last day of every month regardless of how many days there are in any given month.

Send a test email

Note: Available in Grafana Enterprise version 7.0 and later, and Grafana Cloud.

1. In the report, click **Send test email**.
2. In the **Email** field, enter the email address or addresses that you want to test, separated by a semicolon. If you want to use email addresses from the report, then select the **Use emails from report** check box.
3. Click **Send**.

The last saved version of the report will be sent to selected emails. You can use this to verify emails are working and to make sure the report is generated and displayed as you expect.

Pause a report

Note: Available in Grafana Enterprise version 8.0 and later, and Grafana Cloud.

You can pause sending reports from the report list view by clicking the pause icon. The report will not be sent according to its schedule until it is resumed by clicking the resume button on the report row.

Add multiple dashboards to a report

Note: Available in Grafana Enterprise version 9.0 and later, and Grafana Cloud.

You can add more than one dashboard to a report. Additional dashboards will be rendered as new pages in the same PDF file, or additional images if you chose to embed images in your report email. You cannot add the same dashboard to a report multiple times.

Embed a dashboard as an image into a report

Note: Available in Grafana Enterprise version 9.0 and later, and Grafana Cloud.

You can send a report email with an image of the dashboard embedded in the email instead of attached as a PDF. In this case, the email recipients can see the dashboard at a glance instead of having to open the PDF.

Export dashboard as PDF

You can generate and save PDF files of any dashboard.

Note: Available in Grafana Enterprise version 6.7 and later, and Grafana Cloud.

1. In the dashboard that you want to export as PDF, click the **Share** button.
2. On the PDF tab, select a layout option for the exported dashboard: **Portrait** or **Landscape**.
3. Click **Save as PDF** to render the dashboard as a PDF file.

   Grafana opens the PDF file in a new window or browser tab.

Send a report via the API

You can send reports programmatically with the send report endpoint in the HTTP APIs.

Rendering configuration

When generating reports, each panel renders separately before being collected in a PDF. You can configure the per-panel rendering timeout and number of concurrently rendered panels.

To make a panel more legible, you can set a scale factor for the rendered images. However, a higher scale factor increases the file size of the generated PDF.

You can also specify custom fonts that support different Unicode scripts. The DejaVu font is the default used for PDF rendering.

These options are available in the configuration file.

ini

Copy

```
[reporting]
# Use this option to enable or disable the reporting feature. When disabled, no reports are genera
enabled = true
# Set timeout for each panel rendering request
rendering_timeout = 10s
```

```
# Set maximum number of concurrent calls to the rendering service
concurrent_render_limit = 4
# Set the scale factor for rendering images. 2 is enough for monitor resolutions
# 4 would be better for printed material. Setting a higher value affects performance and memory
image_scale_factor = 2
# Set the maximum file size in megabytes for the CSV attachments
max_attachment_size_mb = 10
# Path to the directory containing font files
fonts_path =
# Name of the TrueType font file with regular style
font_regular = DejaVuSansCondensed.ttf
# Name of the TrueType font file with bold style
font_bold = DejaVuSansCondensed-Bold.ttf
```

Expand code

Report settings

Note: Available in Grafana Enterprise version 7.2 and later, and Grafana Cloud.

You can configure organization-wide report settings in the **Settings** under **Dashboards > Reporting**. Settings are applied to all the reports for current organization.

You can customize the branding options.

Report branding:

- **Company logo:** Company logo displayed in the report PDF. It can be configured by specifying a URL, or by uploading a file. The maximum file size is 16 MB. Defaults to the Grafana logo.

Email branding:

- **Company logo:** Company logo displayed in the report email. It can be configured by specifying a URL, or by uploading a file. The maximum file size is 16 MB. Defaults to the Grafana logo.
- **Email footer:** Toggle to enable the report email footer. Select **Sent by** or **None**.
- **Footer link text:** Text of the link in the report email footer. Defaults to `Grafana`.
- **Footer link URL:** Link of the report email footer.

Currently, the API does not allow for the simultaneous upload of files with identical names for both the email logo and report logo. You can still upload the same file for each logo separately in two distinct steps.

Troubleshoot reporting

To troubleshoot and get more log information, enable debug logging in the configuration file. Refer to Configuration for more information.

bash Copy

```
[log]
filters = report:debug
```

Share dashboards and panels

Grafana enables you to share dashboards and panels with other users within an organization and in certain situations, publicly on the Web. You can share using:

- A direct link
- A Snapshot
- An embedded link (for panels only)
- An export link (for dashboards only)

You must have an authorized viewer permission to see an image rendered by a direct link.

The same permission is also required to view embedded links unless you have anonymous access permission enabled for your Grafana instance.

> **NOTE**
>
> As of Grafana 8.0, anonymous access permission is not available in Grafana Cloud.

When you share a panel or dashboard as a snapshot, a snapshot (which is a panel or dashboard at the moment you take the snapshot) is publicly available on the web. Anyone with a link to it can access it. Because snapshots do not require any authorization to view, Grafana removes information related to the account it came from, as well as any sensitive data from the snapshot.

Share a dashboard

You can share a dashboard as a direct link or as a snapshot. You can also export a dashboard.

> **NOTE**
>
> If you change a dashboard, ensure that you save the changes before sharing.

1. Click **Dashboards** in the left-side menu.
2. Click the dashboard you want to share.
3. Click the **Share** button at the top right of the screen.

   The share dialog opens and shows the Link tab.

Share a direct link

The **Link** tab shows the current time range, template variables, and the default theme. You can also share a shortened URL.

1. Click **Copy**.

   This action copies the default or the shortened URL to the clipboard.
2. Send the copied URL to a Grafana user with authorization to view the link.

Publish a snapshot

A dashboard snapshot shares an interactive dashboard publicly. Grafana strips sensitive data such as queries (metric, template and annotation) and panel links, leaving only the visible metric data and series names embedded in the dashboard. Dashboard snapshots can be accessed by anyone with the link.

You can publish snapshots to your local instance or to snapshots.raintank.io. The latter is a free service provided by Grafana Labs that enables you to publish dashboard snapshots to an external Grafana instance. Anyone with the link can view it. You can set an expiration time if you want the snapshot removed after a certain time period.

1. Click the **Snapshot** tab.
2. Click **Publish to snapshots.raintank.io** or **Local Snapshot**.

   Grafana generates a link of the snapshot.
3. Copy the snapshot link, and share it either within your organization or publicly on the web.

If you created a snapshot by mistake, click **Delete snapshot** in the dialog box to remove the snapshot from your Grafana instance.

Delete a snapshot

To delete existing snapshots, follow these steps:

1. In the primary menu, click **Dashboards**.
2. Click **Snapshots** to go to the snapshots management page.
3. Click the red **x** next to the snapshot URL that you want to delete.

The snapshot is immediately deleted. You may need to clear your browser cache or use a private or incognito browser to confirm this.

Export a dashboard as JSON

The dashboard export action creates a Grafana JSON file that contains everything you need, including layout, variables, styles, data sources, queries, and so on, so that you can later import the dashboard.

1. Click **Dashboards** in the main menu.
2. Open the dashboard you want to export.
3. Click the **Share** icon in the top navigation bar.
4. Click **Export**.

   If you're exporting the dashboard to use in another instance, with different data source UIDs, enable the **Export for sharing externally** switch.
5. Click **Save to file**.

Grafana downloads a JSON file to your local machine.

Make a dashboard portable

If you want to export a dashboard for others to use, you can add template variables for things like a metric prefix (use a constant variable) and server name.

A template variable of the type `Constant` is automatically hidden in the dashboard, and is also added as a required input when the dashboard is imported.

Export dashboard as PDF

You can generate and save PDF files of any dashboard.

> **Note:** Available in Grafana Enterprise and Grafana Cloud.

1. Click **Dashboards** in the left-side menu.
2. Click the dashboard you want to share.
3. Click the **Share** button at the top right of the screen.
4. On the PDF tab, select a layout option for the exported dashboard: **Portrait** or **Landscape**.
5. Click **Save as PDF** to render the dashboard as a PDF file.

   Grafana opens the PDF file in a new window or browser tab.

Share a panel

You can share a panel as a direct link, as a snapshot, or as an embedded link. You can also create library panels using the **Share** option on any panel.

1. Hover over any part of the panel to display the actions menu on the top right corner.
2. Click the menu and select **Share**.

   The share dialog opens and shows the **Link** tab.

Use direct link

The **Link** tab shows the current time range, template variables, and the default theme. You can optionally enable a shortened URL to share.

1. Click **Copy**.

   This action copies the default or the shortened URL to the clipboard.
2. Send the copied URL to a Grafana user with authorization to view the link.
3. You also optionally click **Direct link rendered image** to share an image of the panel.

For more information, refer to Image rendering.

The following example shows a link to a server-side rendered PNG:

```bash
https://play.grafana.org/d/000000012/grafana-play-home?orgId=1&from=1568719680173&to=1568726880174
```

Query string parameters for server-side rendered images

- **width:** Width in pixels. Default is 800.
- **height:** Height in pixels. Default is 400.
- **tz:** Timezone in the format `UTC%2BHH%3AMM` where HH and MM are offset in hours and minutes after UTC
- **timeout:** Number of seconds. The timeout can be increased if the query for the panel needs more than the default 30 seconds.
- **scale:** Numeric value to configure device scale factor. Default is 1. Use a higher value to produce more detailed images (higher DPI). Supported in Grafana v7.0+.

Publish a snapshot

A panel snapshot shares an interactive panel publicly. Grafana strips sensitive data leaving only the visible metric data and series names embedded in the dashboard. Panel snapshots can be accessed by anyone with the link.

You can publish snapshots to your local instance or to snapshots.raintank.io. The latter is a free service provided by Grafana Labs, that enables you to publish dashboard snapshots to an external Grafana instance. You can optionally set an expiration time if you want the snapshot to be removed after a certain time period.

1. In the **Share Panel** dialog, click **Snapshot** to go to the tab.
2. Click **Publish to snapshots.raintank.io** or **Local Snapshot**.

   Grafana generates the link of the snapshot.
3. Copy the snapshot link, and share it either within your organization or publicly on the web.

If you created a snapshot by mistake, click **Delete snapshot** in the dialog box to remove the snapshot from your Grafana instance.

Delete a snapshot

To delete existing snapshots, follow these steps:

1. In the primary menu, click **Dashboards**.
2. Click **Snapshots** to go to the snapshots management page.
3. Click the red **x** next to the snapshot URL that you want to delete.

The snapshot is immediately deleted. You may need to clear your browser cache or use a private or incognito browser to confirm this.

Embed panel

You can embed a panel using an iframe on another web site. A viewer must be signed into Grafana to view the graph.

> Note: As of Grafana 8.0, anonymous access permission is no longer available for Grafana Cloud.

Here is an example of the HTML code:

```html
<iframe
  src="https://snapshots.raintank.io/dashboard-solo/snapshot/y7zwi2bZ7FcoTlB93WN7yWO4aMiz3pZb?from=1493369923321
  width="650"
  height="300"
  frameborder="0"
>
```

The result is an interactive Grafana graph embedded in an iframe.

Library panel

To create a library panel from the **Share Panel** dialog:

1. Click **Library panel**.
2. In **Library panel name**, enter the name.
3. In **Save in folder**, select the folder in which to save the library panel. By default, the root level is selected.
4. Click **Create library panel** to save your changes.
5. Save the dashboard.

Variables

The following topics describe how to add and manage variables in your dashboards:

- Add variables
- Manage and inspect variables
- Variable syntax

A variable is a placeholder for a value. You can use variables in metric queries and in panel titles. So when you change the value, using the dropdown at the top of the dashboard, your panel's metric queries will change to reflect the new value.

Variables allow you to create more interactive and dynamic dashboards. Instead of hard-coding things like server, application, and sensor names in your metric queries, you can use variables in

their place. Variables are displayed as dropdown lists at the top of the dashboard. These dropdowns make it easy to change the data being displayed in your dashboard.

Give it a try using Grafana Play

With Grafana Play, you can explore and see how it works, learning from practical examples to accelerate your development. This feature can be seen on Templating - Global variables and interpolation.

Variables are useful for administrators who want to allow Grafana viewers to adjust visualizations without giving them full editing permissions. Grafana viewers can use variables.

Variables and templates also allow you to single-source dashboards. If you have multiple identical data sources or servers, you can make one dashboard and use variables to change what you are viewing. This simplifies maintenance and upkeep enormously.

Templates

A *template* is any query that contains a variable.

For example, if you were administering a dashboard to monitor several servers, you *could* make a dashboard for each server. Or you could create one dashboard and use panels with template queries like this one:

```
wmi_system_threads{instance=~"$server"}
```

Copy

Variable values are always synced to the URL using the syntax `var-<varname>=value`.

Additional Examples

Variables are listed in drop-down lists across the top of the screen. Select different variables to see how the visualizations change.

To see variable settings, navigate to **Dashboard Settings > Variables**. Click a variable in the list to see its settings.

Variables can be used in titles, descriptions, text panels, and queries. Queries with text that starts with `$` are templates. Not all panels will have template queries.

The following dashboards in Grafana Play provide examples of template variables:

- Templating, repeated panels - Using query variables to control how many panels appear.
- Templated Dynamic Dashboard - Uses query variables, chained query variables, an interval variable, and a repeated panel.
- Templating - Nested Variables Drilldown

Variable best practices

- Variable drop-down lists are displayed in the order they are listed in the variable list in Dashboard settings.
- Put the variables that you will change often at the top, so they will be shown first (far left on the dashboard).
- By default, variables don't have a default value. This means that the topmost value in the drop-down is always preselected. If you want to pre-populate a variable with an empty value, you can use the following workaround in the variable settings:
  1. Select the **Include All Option** checkbox.
  2. In the **Custom all value** field, enter a value like `+`.

Add variables

The following table lists the types of variables shipped with Grafana.

Variable type	Description
Query	Query-generated list of values such as metric names, server names, sensor IDs, data centers, and so on. Add a query variable.
Custom	Define the variable options manually using a comma-separated list. Add a custom variable.
Text box	Display a free text input field with an optional default value. Add a text box variable.
Constant	Define a hidden constant. Add a constant variable.
Data source	Quickly change the data source for an entire dashboard. Add a data source variable.
Interval	Interval variables represent time spans. Add an interval variable.
Ad hoc filters	Key/value filters that are automatically added to all metric queries for a data source (Prometheus, Loki, InfluxDB, and Elasticsearch only). Add ad hoc filters.
Global variables	Built-in variables that can be used in expressions in the query editor. Refer to Global variables.
Chained variables	Variable queries can contain other variables. Refer to Chained variables.

Enter General options

You must enter general options for any type of variable that you create.

1. Navigate to the dashboard you want to make a variable for and click the **Dashboard settings** (gear) icon at the top of the page.

2. On the **Variables** tab, click **New variable**.
3. Enter a **Name** for the variable.
4. In the **Type** list, select **Query**.
5. (Optional) In **Label**, enter the display name of the variable dropdown.

   If you don't enter a display name, then the dropdown label is the variable name.
6. Choose a **Hide** option:
   - **No selection (blank):** The variable dropdown displays the variable **Name** or **Label** value. This is the default.
   - **Label:** The variable dropdown only displays the selected variable value and a down arrow.
   - **Variable:** No variable dropdown is displayed on the dashboard.

Add a query variable

Query variables enable you to write a data source query that can return a list of metric names, tag values, or keys. For example, a query variable might return a list of server names, sensor IDs, or data centers. The variable values change as they dynamically fetch options with a data source query.

Query variables are generally only supported for strings. If your query returns numbers or any other data type, you might need to convert them to strings in order to use them as variables. For the Azure data source, for example, you can use the tostring function for this purpose.

Query expressions can contain references to other variables and in effect create linked variables. Grafana detects this and automatically refreshes a variable when one of its linked variables change.

Note: Query expressions are different for each data source. For more information, refer to the documentation for your data source.

1. Enter general options.
2. In the **Data source** list, select the target data source for the query. For more information about data sources, refer to Add a data source.
3. In the **Refresh** list, select when the variable should update options.
   - **On Dashboard Load:** Queries the data source every time the dashboard loads. This slows down dashboard loading, because the variable query needs to be completed before dashboard can be initialized.
   - **On Time Range Change:** Queries the data source every time the dashboard loads and when the dashboard time range changes. Use this option if your variable options query contains a time range filter or is dependent on the dashboard time range.

4. In the **Query** field, enter a query.
   - The query field varies according to your data source. Some data sources have custom query editors.
   - Each data source defines how the variable values are extracted. The typical implementation uses every string value returned from the data source response as a variable value. Make sure to double-check the documentation for the data source.
   - Some data sources let you provide custom "display names" for the values. For instance, the PostgreSQL, MySQL, and Microsoft SQL Server plugins handle this by looking for fields named `__text` and `__value` in the result. Other data sources may look for `text` and `value` or use a different approach. Always remember to double-check the documentation for the data source.
   - If you need more room in a single input field query editor, then hover your cursor over the lines in the lower right corner of the field and drag downward to expand.
5. (Optional) In the **Regex** field, type a regex expression to filter or capture specific parts of the names returned by your data source query. To see examples, refer to Filter variables with regex.
6. In the **Sort** list, select the sort order for values to be displayed in the dropdown list. The default option, **Disabled**, means that the order of options returned by your data source query will be used.
7. (Optional) Enter Selection Options.
8. In **Preview of values**, Grafana displays a list of the current variable values. Review them to ensure they match what you expect.
9. Click **Add** to add the variable to the dashboard.

Add a custom variable

Use a *custom* variable for a value that does not change, such as a number or a string.

For example, if you have server names or region names that never change, then you might want to create them as custom variables rather than query variables. Because they do not change, you might use them in chained variables rather than other query variables. That would reduce the number of queries Grafana must send when chained variables are updated.

1. Enter general options.
2. In the **Values separated by comma** list, enter the values for this variable in a comma-separated list. You can include numbers, strings, or key/value pairs separated by a space and a colon. For example, `key1 : value1,key2 : value2`.
3. (Optional) Enter Selection Options.
4. In **Preview of values**, Grafana displays a list of the current variable values. Review them to ensure they match what you expect.
5. Click **Add** to add the variable to the dashboard.

Add a text box variable

Text box variables display a free text input field with an optional default value. This is the most flexible variable, because you can enter any value. Use this type of variable if you have metrics with high cardinality or if you want to update multiple panels in a dashboard at the same time.

For more information about cardinality, refer to What are cardinality spikes and why do they matter?

1. Enter general options.
2. (Optional) In the **Default value** field, select the default value for the variable. If you do not enter anything in this field, then Grafana displays an empty text box for users to type text into.
3. In **Preview of values**, Grafana displays a list of the current variable values. Review them to ensure they match what you expect.
4. Click **Add** to add the variable to the dashboard.

Add a constant variable

Constant variables enable you to define a hidden constant. This is useful for metric path prefixes for dashboards you want to share. When you export a dashboard, constant variables are converted to import options.

Constant variables are *not* flexible. Each constant variable only holds one value, and it cannot be updated unless you update the variable settings.

Constant variables are useful when you have complex values that you need to include in queries but don't want to retype in every query. For example, if you had a server path called `i-0b6a61efe2ab843gg`, then you could replace it with a variable called `$path_gg`.

1. Enter general options.
2. In the **Value** field, enter the variable value. You can enter letters, numbers, and symbols. You can even use wildcards if you use raw format.
3. In **Preview of values**, Grafana displays the current variable value. Review it to ensure it matches what you expect.
4. Click **Add** to add the variable to the dashboard.

Add a data source variable

Data source variables enable you to quickly change the data source for an entire dashboard. They are useful if you have multiple instances of a data source, perhaps in different environments.

1. Enter general options.

2. In the **Type** list, select the target data source for the variable.

   You can also click **Open advanced data source picker** to see more options, including adding a data source (Admins only). For more information about data sources, refer to Add a data source.

3. (Optional) In **Instance name filter**, enter a regex filter for which data source instances to choose from in the variable value drop-down list. Leave this field empty to display all instances.

4. (Optional) Enter Selection Options.

5. In **Preview of values**, Grafana displays a list of the current variable values. Review them to ensure they match what you expect.

6. Click **Add** to add the variable to the dashboard.

Add an interval variable

Use an *interval* variable to represents time spans such as `1m`, `1h`, `1d`. You can think of them as a dashboard-wide "group by time" command. Interval variables change how the data is grouped in the visualization. You can also use the Auto Option to return a set number of data points per time span.

You can use an interval variable as a parameter to group by time (for InfluxDB), date histogram interval (for Elasticsearch), or as a summarize function parameter (for Graphite).

1. Enter general options.

2. In the **Values** field, enter the time range intervals that you want to appear in the variable drop-down list. The following time units are supported: `s (seconds)`, `m (minutes)`, `h (hours)`, `d (days)`, `w (weeks)`, `M (months)`, and `y (years)`. You can also accept or edit the default values: `1m,10m,30m,1h,6h,12h,1d,7d,14d,30d`.

3. (Optional) Turn on the **Auto Option** if you want to add the `auto` option to the list. This option allows you to specify how many times the current time range should be divided to calculate the current `auto` time span. If you turn it on, then two more options appear:

   - **Step count -** Select the number of times the current time range will be divided to calculate the value, similar to the **Max data points** query option. For example, if the current visible time range is 30 minutes, then the `auto` interval groups the data into 30 one-minute increments. The default value is 30 steps.
   - **Min Interval -** The minimum threshold below which the step count intervals will not divide the time. To continue the 30 minute example, if the minimum interval is set to 2m, then Grafana would group the data into 15 two-minute increments.

4. In **Preview of values**, Grafana displays a list of the current variable values. Review them to ensure they match what you expect.

5. Click **Add** to add the variable to the dashboard.

Interval variable examples

The following example shows a template variable `myinterval` in a Graphite function:

```
summarize($myinterval, sum, false)
```

Copy

The following example shows a more complex Graphite example, from the Graphite Template Nested Requests panel:

```
groupByNode(summarize(movingAverage(apps.$app.$server.counters.requests.count, 5), '
```

Copy

Add ad hoc filters

Ad hoc filters enable you to add key/value filters that are automatically added to all metric queries that use the specified data source. Unlike other variables, you do not use ad hoc filters in queries. Instead, you use ad hoc filters to write filters for existing queries.

> **NOTE**
>
> Not all data sources support ad hoc filters. Examples of those that do include Prometheus, Loki, InfluxDB, and Elasticsearch.

1. Enter general options.
2. In the **Data source** list, select the target data source.

   You can also click **Open advanced data source picker** to see more options, including adding a data source (Admins only). For more information about data sources, refer to Add a data source.
3. Click **Add** to add the variable to the dashboard.

Create ad hoc filters

Ad hoc filters are one of the most complex and flexible variable options available. Instead of a regular list of variable options, this variable allows you to build a dashboard-wide ad hoc query. Filters you apply in this manner are applied to all panels on the dashboard.

Configure variable selection options

Selection Options are a feature you can use to manage variable option selections. All selection options are optional, and they are off by default.

Multi-value variables

Interpolating a variable with multiple values selected is tricky as it is not straight forward how to format the multiple values into a string that is valid in the given context where the variable is used. Grafana tries to solve this by allowing each data source plugin to inform the templating interpolation engine what format to use for multiple values.

> **NOTE**
>
> The **Custom all value** option on the variable must be blank for Grafana to format all values into a single string. If it is left blank, then Grafana concatenates (adds together) all the values in the query. Something like `value1,value2,value3`. If a custom `all` value is used, then instead the value will be something like `*` or `all`.

Multi-value variables with a Graphite data source

Graphite uses glob expressions. A variable with multiple values would, in this case, be interpolated as `{host1,host2,host3}` if the current variable value was *host1*, *host2*, and *host3*.

Multi-value variables with a Prometheus or InfluxDB data source

InfluxDB and Prometheus use regex expressions, so the same variable would be interpolated as `(host1|host2|host3)`. Every value would also be regex escaped. If not, a value with a regex control character would break the regex expression.

Multi-value variables with an Elastic data source

Elasticsearch uses lucene query syntax, so the same variable would be formatted as `("host1" OR "host2" OR "host3")`. In this case, every value must be escaped so that the value only contains lucene control words and quotation marks.

Troubleshoot multi-value variables

Automatic escaping and formatting can cause problems and it can be tricky to grasp the logic behind it. Especially for InfluxDB and Prometheus where the use of regex syntax requires that the variable is used in regex operator context.

If you do not want Grafana to do this automatic regex escaping and formatting, then you must do one of the following:

- Turn off the **Multi-value** or **Include All option** options.
- Use the raw variable format.

Include All option

Grafana adds an `All` option to the variable dropdown list. If a user selects this option, then all variable options are selected.

Custom all value

This option is only visible if the **Include All option** is selected.

Enter regex, globs, or lucene syntax in the **Custom all value** field to define the value of the `All` option.

By default the `All` value includes all options in combined expression. This can become very long and can have performance problems. Sometimes it can be better to specify a custom all value, like a wildcard regex.

In order to have custom regex, globs, or lucene syntax in the **Custom all value** option, it is never escaped so you will have to think about what is a valid value for your data source.

Global variables

Grafana has global built-in variables that can be used in expressions in the query editor. This topic lists them in alphabetical order and defines them. These variables are useful in queries, dashboard links, panel links, and data links.

$__dashboard

> Only available in Grafana v6.7+. In Grafana 7.1, the variable changed from showing the UID of the current dashboard to the name of the current dashboard.

This variable is the name of the current dashboard.

$__from and $__to

Grafana has two built-in time range variables: `$__from` and `$__to`. They are currently always interpolated as epoch milliseconds by default, but you can control date formatting.

> **NOTE**
>
> This special formatting syntax is only available in Grafana 7.1.2+

Syntax	Example result	Description
`${__from}`	1594671549254	Unix millisecond epoch
`${__from:date}`	2020-07-13T20:19:09.254Z	No args, defaults to ISO 8601/RFC 3339

Syntax	Example result	Description
`${__from:date:iso}`	2020-07-13T20:19:09.254Z	ISO 8601/RFC 3339
`${__from:date:seconds}`	1594671549	Unix seconds epoch
`${__from:date:YYYY-MM}`	2020-07	Any custom date format that does not include the `:` character. Uses browser time. Use `:date` or `:date:iso` for UTC

The syntax above also works with `${__to}`.

You can use this variable in URLs, as well. For example, you can send a user to a dashboard that shows a time range from six hours ago until now: https://play.grafana.org/d/000000012/grafana-play-home?viewPanel=2&orgId=1?from=now-6h&to=now

$__interval

You can use the `$__interval` variable as a parameter to group by time (for InfluxDB, MySQL, Postgres, MSSQL), Date histogram interval (for Elasticsearch), or as a *summarize* function parameter (for Graphite).

Grafana automatically calculates an interval that can be used to group by time in queries. When there are more data points than can be shown on a graph, then queries can be made more efficient by grouping by a larger interval. It is more efficient to group by 1 day than by 10s when looking at 3 months of data and the graph will look the same and the query will be faster. The `$__interval` is calculated using the time range and the width of the graph (the number of pixels).

Approximate Calculation: `(to - from) / resolution`

For example, when the time range is 1 hour and the graph is full screen, then the interval might be calculated to `2m` - points are grouped in 2 minute intervals. If the time range is 6 months and the graph is full screen, then the interval might be `1d` (1 day) - points are grouped by day.

In the InfluxDB data source, the legacy variable `$interval` is the same variable. `$__interval` should be used instead.

The InfluxDB and Elasticsearch data sources have `Group by time interval` fields that are used to hard code the interval or to set the minimum limit for the `$__interval` variable (by using the `>` syntax → `>10m`).

$__interval_ms

This variable is the `$__interval` variable in milliseconds, not a time interval formatted string. For example, if the `$__interval` is `20m` then the `$__interval_ms` is `1200000`.

$__name

This variable is only available in the Singlestat panel and can be used in the prefix or suffix fields on the Options tab. The variable will be replaced with the series name or alias.

> **NOTE**
>
> The Singlestat panel is no longer available from Grafana 8.0.

$__org

This variable is the ID of the current organization. `${__org.name}` is the name of the current organization.

$__user

> Only available in Grafana v7.1+

`${__user.id}` is the ID of the current user. `${__user.login}` is the login handle of the current user. `${__user.email}` is the email for the current user.

$__range

Currently only supported for Prometheus and Loki data sources. This variable represents the range for the current dashboard. It is calculated by `to - from`. It has a millisecond and a second representation called `$__range_ms` and `$__range_s`.

$__rate_interval

Currently only supported for Prometheus data sources. The `$__rate_interval` variable is meant to be used in the rate function. Refer to Prometheus query variables for details.

$__rate_interval_ms

This variable is the `$__rate_interval` variable in milliseconds, not a time-interval-formatted string. For example, if the `$__rate_interval` is `20m` then the `$__rate_interval_ms` is `1200000`.

$timeFilter or $__timeFilter

The `$timeFilter` variable returns the currently selected time range as an expression. For example, the time range interval `Last 7 days` expression is `time > now() - 7d`.

This is used in several places, including:

- The WHERE clause for the InfluxDB data source. Grafana adds it automatically to InfluxDB queries when in Query Editor mode. You can add it manually in Text Editor mode: `WHERE $timeFilter`.
- Log Analytics queries in the Azure Monitor data source.

- SQL queries in MySQL, Postgres, and MSSQL.
- The `$__timeFilter` variable is used in the MySQL data source.

$__timezone

The `$__timezone` variable returns the currently selected time zone, either `utc` or an entry of the IANA time zone database (for example, `America/New_York`).

If the currently selected time zone is *Browser Time*, Grafana will try to determine your browser time zone.

Chained variables

Chained variables, also called *linked variables* or *nested variables*, are query variables with one or more other variables in their variable query. This section explains how chained variables work and provides links to example dashboards that use chained variables.

Chained variable queries are different for every data source, but the premise is the same for all. You can use chained variable queries in any data source that allows them.

Extremely complex linked templated dashboards are possible, 5 or 10 levels deep. Technically, there is no limit to how deep or complex you can go, but the more links you have, the greater the query load.

Grafana Play dashboard examples

The following Grafana Play dashboards contain fairly simple chained variables, only two layers deep. To view the variables and their settings, click **Dashboard settings** (gear icon) and then click **Variables**. Both examples are expanded in the following section.

- Graphite Templated Nested
- InfluxDB Templated

Examples explained

Variables are useful to reuse dashboards and dynamically change what is shown in dashboards. Chained variables are especially useful to filter what you see.

Create parent/child relationship in a variable, sort of a tree structure where you can select different levels of filters.

The following sections explain the linked examples in the dashboards above in depth and builds on them. While the examples are data source-specific, the concepts can be applied broadly.

Graphite example

In this example, there are several applications. Each application has a different subset of servers. It is based on the Graphite Templated Nested.

Now, you could make separate variables for each metric source, but then you have to know which server goes with which app. A better solution is to use one variable to filter another. In this example, when the user changes the value of the `app` variable, it changes the dropdown options returned by the `server` variable. Both variables use the **Multi-value** option and **Include all option**, enabling users to select some or all options presented at any time.

app variable

The query for this variable basically says, "Give me all the applications that exist."

```
apps.*
```

Copy

The values returned are `backend`, `country`, `fakesite`, and `All`.

server variable

The query for this variable basically says, "Give me all servers for the currently chosen application."

```
apps.$app.*
```

Copy

If the user selects `backend`, then the query changes to:

```
apps.backend.*
```

Copy

The query returns all servers associated with `backend`, including `backend_01`, `backend_02`, and so on.

If the user selects `fakesite`, then the query changes to:

```
apps.fakesite.*
```

Copy

The query returns all servers associated with `fakesite`, including `web_server_01`, `web_server_02`, and so on.

More variables

> **NOTE**
>
> This example is theoretical. The Graphite server used in the example does not contain CPU metrics.

The dashboard stops at two levels, but you could keep going. For example, if you wanted to get CPU metrics for selected servers, you could copy the `server` variable and extend the query so that it reads:

```
apps.$app.$server.cpu.*
```

Copy

This query basically says, "Show me the CPU metrics for the selected server."

Depending on what variable options the user selects, you could get queries like:

```
apps.backend.backend_01.cpu.*
apps.{backend.backend_02,backend_03}.cpu.*
apps.fakesite.web_server_01.cpu.*
```

Copy

InfluxDB example

In this example, you have several data centers. Each data center has a different subset of hosts. It is based on the InfluxDB Templated dashboard.

In this example, when the user changes the value of the `datacenter` variable, it changes the dropdown options returned by the `host` variable. The `host` variable uses the **Multi-value** option and **Include all option**, allowing users to select some or all options presented at any time. The `datacenter` does not use either option, so you can only select one data center at a time.

datacenter variable

The query for this variable basically says, “Give me all the data centers that exist.”

```
SHOW TAG VALUES WITH KEY = "datacenter"
```

Copy

The values returned are `America`, `Africa`, `Asia`, and `Europe`.

host variable

The query for this variable basically says, “Give me all hosts for the currently chosen data center.”

```
SHOW TAG VALUES WITH KEY = "hostname" WHERE "datacenter" =~ /^$datacenter$/
```

Copy

If the user selects `America`, then the query changes to:

```
SHOW TAG VALUES WITH KEY = "hostname" WHERE "datacenter" =~ /^America/
```

Copy

The query returns all servers associated with `America`, including `server1`, `server2`, and so on.

If the user selects `Europe`, then the query changes to:

```
SHOW TAG VALUES WITH KEY = "hostname" WHERE "datacenter" =~ /^Europe/
```

Copy

The query returns all servers associated with `Europe`, including `server3`, `server4`, and so on.

More variables

> **NOTE**
>
> This example is theoretical. The InfluxDB server used in the example does not contain CPU metrics.

The dashboard stops at two levels, but you could keep going. For example, if you wanted to get CPU metrics for selected hosts, you could copy the `host` variable and extend the query so that it

reads:

```
SHOW TAG VALUES WITH KEY = "cpu" WHERE "datacenter" =~ /^$datacenter$/ AND "host" =~
```

Copy

This query basically says, “Show me the CPU metrics for the selected host.”

Depending on what variable options the user selects, you could get queries like:

bash

Copy

```
SHOW TAG VALUES WITH KEY = "cpu" WHERE "datacenter" =~ /^America/ AND "host" =~ /^server2/
SHOW TAG VALUES WITH KEY = "cpu" WHERE "datacenter" =~ /^Africa/ AND "host" =~ /^server/7/
SHOW TAG VALUES WITH KEY = "cpu" WHERE "datacenter" =~ /^Europe/ AND "host" =~ /^server3+server4/
```

Best practices and tips

The following practices will make your dashboards and variables easier to use.

Creating new linked variables

- Chaining variables create parent/child dependencies. You can envision them as a ladder or a tree.
- The easiest way to create a new chained variable is to copy the variable that you want to base the new one on. In the variable list, click the **Duplicate variable** icon to the right of the variable entry to create a copy. You can then add on to the query for the parent variable.
- New variables created this way appear at the bottom of the list. You might need to drag it to a different position in the list to get it into a logical order.

Variable order

You can change the orders of variables in the dashboard variable list by clicking the up and down arrows on the right side of each entry. Grafana lists variable dropdowns left to right according to this list, with the variable at the top on the far left.

- List variables that do not have dependencies at the top, before their child variables.
- Each variable should follow the one it is dependent on.
- Remember there is no indication in the UI of which variables have dependency relationships. List the variables in a logical order to make it easy on other users (and yourself).

Complexity consideration

The more layers of dependency you have in variables, the longer it will take to update dashboards after you change variables.

For example, if you have a series of four linked variables (country, region, server, metric) and you change a root variable value (country), then Grafana must run queries for all the dependent variables before it updates the visualizations in the dashboard.

Filter variables with regex

Using the Regex Query option, you filter the list of options returned by the variable query or modify the options returned.

This page shows how to use regex to filter/modify values in the variable dropdown.

Using the Regex Query Option, you filter the list of options returned by the Variable query or modify the options returned. For more information, refer to the Mozilla guide on Regular expressions.

Examples of filtering on the following list of options:

```text
backend_01
backend_02
backend_03
backend_04
```

Filter so that only the options that end with `01` or `02` are returned:

Regex:

```regex
/
(
01|02
)
$/
```

Result:

```text
backend_01
backend_02
```

Filter and modify the options using a regex capture group to return part of the text:

Regex:

```regex
/.*
(
01|02
)
/
```

Result:

```text
01
02
```

Filter and modify - Prometheus Example

List of options:

```text
up{instance="demo.robustperception.io:9090",job="prometheus"} 1 1521630638000
up{instance="demo.robustperception.io:9093",job="alertmanager"} 1 1521630638000
up{instance="demo.robustperception.io:9100",job="node"} 1 1521630638000
```

Regex:

```regex
/.*instance="
(
[^"]*
)
.*/
```

Result:

```text
```

```
demo.robustperception.io:9090
demo.robustperception.io:9093
demo.robustperception.io:9100
```

Filter and modify using named text and value capture groups

> **NOTE**
>
> This feature is available in Grafana 7.4+.

Using named capture groups, you can capture separate 'text' and 'value' parts from the options returned by the variable query. This allows the variable drop-down list to contain a friendly name for each value that can be selected.

For example, when querying the `node_hwmon_chip_names` Prometheus metric, the `chip_name` is a lot friendlier than the `chip` value. So the following variable query result:

text Copy

```
node_hwmon_chip_names{chip="0000:d7:00_0_0000:d8:00_0",chip_name="enp216s0f0np0"} 1
node_hwmon_chip_names{chip="0000:d7:00_0_0000:d8:00_1",chip_name="enp216s0f0np1"} 1
node_hwmon_chip_names{chip="0000:d7:00_0_0000:d8:00_2",chip_name="enp216s0f0np2"} 1
node_hwmon_chip_names{chip="0000:d7:00_0_0000:d8:00_3",chip_name="enp216s0f0np3"} 1
```

Passed through the following Regex:

regex Copy

```
/chip_name="
(?<text>
[^"]+
)
|chip="
(?<value>
[^"]+
)
/g
```

Would produce the following drop-down list:

text Copy

```
Display Name          Value
------------          -------------------------
enp216s0f0np0         0000:d7:00_0_0000:d8:00_0
enp216s0f0np1         0000:d7:00_0_0000:d8:00_1
enp216s0f0np2         0000:d7:00_0_0000:d8:00_2
enp216s0f0np3         0000:d7:00_0_0000:d8:00_3
```

Note: Only `text` and `value` capture group names are supported.

Variable syntax

Panel titles and metric queries can refer to variables using two different syntaxes:

- `$varname` This syntax is easy to read, but it does not allow you to use a variable in the middle of a word. **Example:** apps.frontend.$server.requests.count
- `${var_name}` Use this syntax when you want to interpolate a variable in the middle of an expression.
- `${var_name:<format>}` This format gives you more control over how Grafana interpolates values. Refer to Advanced variable format options for more detail on all the formatting types.
- `[[varname]]` Do not use. Deprecated old syntax, will be removed in a future release.

Before queries are sent to your data source the query is *interpolated*, meaning the variable is replaced with its current value. During interpolation, the variable value might be *escaped* in order to conform to the syntax of the query language and where it is used. For example, a variable used in a regex expression in an InfluxDB or Prometheus query will be regex escaped. Read the data source specific documentation topic for details on value escaping during interpolation.

For advanced syntax to override data source default formatting, refer to Advanced variable format options.

Advanced variable format options

The formatting of the variable interpolation depends on the data source, but there are some situations where you might want to change the default formatting.

For example, the default for the MySql data source is to join multiple values as comma-separated with quotes: `'server01','server02'`. In some cases, you might want to have a comma-separated string without quotes: `server01,server02`. You can make that happen with advanced variable formatting options listed below.

General syntax

Syntax: `${var_name:option}`

Test the formatting options on the Grafana Play site.

If any invalid formatting option is specified, then `glob` is the default/fallback option.

An alternative syntax (that might be deprecated in the future) is `[[var_name:option]]`.

CSV

Formats variables with multiple values as a comma-separated string.

```bash
servers = ['test1', 'test2']
String to interpolate: '${servers:csv}'
Interpolation result: 'test1,test2'
```

Distributed - OpenTSDB

Formats variables with multiple values in custom format for OpenTSDB.

```bash
servers = ['test1', 'test2']
String to interpolate: '${servers:distributed}'
Interpolation result: 'test1,servers=test2'
```

Doublequote

Formats single- and multi-valued variables into a comma-separated string, escapes `"` in each value by `\"` and quotes each value with `"`.

```bash
servers = ['test1', 'test2']
String to interpolate: '${servers:doublequote}'
Interpolation result: '"test1","test2"'
```

Glob - Graphite

Formats variables with multiple values into a glob (for Graphite queries).

```bash
```

```
servers = ['test1', 'test2']
String to interpolate: '${servers:glob}'
Interpolation result: '{test1,test2}'
```

JSON

Formats variables with multiple values as a comma-separated string.

bash Copy

```
servers = ['test1', 'test2']
String to interpolate: '${servers:json}'
Interpolation result: '["test1", "test2"]'
```

Lucene - Elasticsearch

Formats variables with multiple values in Lucene format for Elasticsearch.

bash Copy

```
servers = ['test1', 'test2']
String to interpolate: '${servers:lucene}'
Interpolation result: '("test1" OR "test2")'
```

Percentencode

Formats single and multi valued variables for use in URL parameters.

bash Copy

```
servers = ['foo()bar BAZ', 'test2']
String to interpolate: '${servers:percentencode}'
Interpolation result: 'foo%28%29bar%20BAZ%2Ctest2'
```

Pipe

Formats variables with multiple values into a pipe-separated string.

bash Copy

```
servers = ['test1.', 'test2']
String to interpolate: '${servers:pipe}'
```

```
Interpolation result: 'test1.|test2'
```

Raw

The raw format for a data source variable returns the UID (unique identifier) of the data source, rather than its name.

```bash
datasourceVariable = 'd7bbe725-9e48-4af8-a0cb-6cb255d873a3'
String to interpolate: '${datasourceVariable:raw}'
Interpolation result: 'd7bbe725-9e48-4af8-a0cb-6cb255d873a3'
```

Regex

Formats variables with multiple values into a regex string.

```bash
servers = ['test1.', 'test2']
String to interpolate: '${servers:regex}'
Interpolation result: '(test1\.|test2)'
```

Singlequote

Formats single- and multi-valued variables into a comma-separated string, escapes `'` in each value by `\'` and quotes each value with `'`.

```bash
servers = ['test1', 'test2']
String to interpolate: '${servers:singlequote}'
Interpolation result: "'test1','test2'"
```

Sqlstring

Formats single- and multi-valued variables into a comma-separated string, escapes `'` in each value by `''` and quotes each value with `'`.

```bash
```

```
servers = ["test'1", "test2"]
String to interpolate: '${servers:sqlstring}'
Interpolation result: "'test''1','test2'"
```

Text

Formats single- and multi-valued variables into their text representation. For a single variable it will just return the text representation. For multi-valued variables it will return the text representation combined with `+`.

bash Copy

```
servers = ["test1", "test2"]
String to interpolate: '${servers:text}'
Interpolation result: "test1 + test2"
```

Query parameters

Formats single- and multi-valued variables into their query parameter representation. Example: `var-foo=value1&var-foo=value2`

bash Copy

```
servers = ["test1", "test2"]
String to interpolate: '${servers:queryparam}'
Interpolation result: "var-servers=test1&var-servers=test2"
```

Assess dashboard usage

Usage insights enables you to have a better understanding of how your Grafana instance is used.

Note: Available in Grafana Enterprise and Grafana Cloud. Grafana Cloud insights logs include additional fields with their own dashboards. Read more in the Grafana Cloud documentation.

The usage insights feature collects a number of aggregated data and stores them in the database:

- Dashboard views (aggregated and per user)
- Data source errors
- Data source queries

The aggregated data provides you access to several features:

- Dashboard and data source insights
- Presence indicator
- Sort dashboards by using insights data
- Visualize usage insight data in a dashboard

This feature also generates detailed logs that can be exported to Loki. Refer to Export logs of usage insights.

Dashboard and data source insights

For every dashboard and data source, you can access usage information.

Dashboard insights

To see dashboard usage information, click the dashboard insights icon in the header.

Dashboard insights show the following information:

- **Stats:** The number of daily queries and errors for the past 30 days.
- **Users & activity:** The daily view count for the last 30 days; last activities on the dashboard and recent users (with a limit of 20).

If public dashboards are enabled, you'll also see a **Public dashboards** tab in your analytics.

Data source insights

Data source insights provides information about how a data source has been used in the past 30 days, such as:

- Queries per day

- Errors per day
- Query load time per day (averaged in ms)

To find data source insights:

1. Click **Connections** in the main navigation.
2. Under Your connections, click **Data sources**.
3. Click a data source.
4. Click the **Insights** tab.

Presence indicator

When you are signed in and looking at a dashboard, you can know who is looking at the same dashboard as you are via a presence indicator, which displays avatars of users who have recently interacted with the dashboard. The default time frame is 10 minutes. To see the user's name, hover over the user's avatar. The avatars come from Gravatar based on the user's email.

When there are more active users on a dashboard than can fit within the presence indicator, click the **+X** icon. Doing so opens dashboard insights, which contains more details about recent user activity.

To change *recent* to something other than the past 10 minutes, edit the configuration file:

```ini
[analytics.views]

# Set age for recent active users to 10 minutes
recent_users_age = 10m
```

To disable the presence indicator, edit the configuration file as follows:

```ini
[analytics.views]



# Disables the presence indicator
recent_users_age = 0
```

The dashboard won't show any avatars and thus no recent user activity.

Sort dashboards by using insights data

In the search view, you can use insights data to help you find most-used, broken, and unused dashboards.

You can sort the dashboards by:

- Errors total
- Errors 30 days (most and least)
- Views total
- Views 30 days (most and least)

Visualize usage insights data

If you set up your installation to export logs of usage insights, we've created two dashboards to help you take advantage of this data.

1. Usage Insights overview provides a top-level perspective of user activity.
2. Data source details dashboard provides a view of data source activity and health.

You can click the previous links to download the respective dashboard JSON, then import into your Grafana installation.

Troubleshoot dashboards

Use the following strategies to help you troubleshoot common dashboard problems.

Dashboard is slow

- Are you trying to render dozens (or hundreds or thousands) of time series on a graph? This can cause the browser to lag. Try using functions like `highestMax` (in Graphite) to reduce the number of returned series.
- Sometimes series names can be very large. This causes larger response sizes. Try using `alias` to reduce the size of the returned series names.
- Are you querying many time series or a long time range? Both of these conditions can cause Grafana or your data source to pull in a lot of data, which may slow the dashboard down. Try reducing one or both of these.
- There could be high load on your network infrastructure. If the slowness isn't consistent, this may be the problem.

Dashboard refresh rate issues

By default, Grafana queries your data source every 30 seconds. However, setting a low refresh rate on your dashboards puts unnecessary stress on the backend. In many cases, querying this frequently isn't necessary because the data source isn't sending data often enough for there to be changes every 30 seconds.

We recommend the following:

- Only enable auto-refreshing on dashboards, panels, or variables if necessary. Users can refresh their browser manually.
- If you require auto-refreshing, then set the refresh rate to a longer time period that makes sense, such as once a minute, every 10 minutes, or every hour.

- Check the time range of your dashboard. If your dashboard has a longer time range, such as a week, then you really don't need automated refreshing and you should disable it.

Handling or rendering null data is wrong or confusing

Some applications publish data intermittently; for example, they only post a metric when an event occurs. By default, Grafana graphs connect lines between the data points, but this can be deceptive.

The graph in the following image has:

- Points and 3-point radius enabled to highlight where data points are actually present.
- **Connect null values** set to **Always**.

The graph in this next image shows bars instead of lines and has the **No value** option under **Standard options** set to **0**.

As you can see, there's a significant difference in the visualizations.

Panels and visualizations

The *panel* is the basic visualization building block in Grafana. Each panel has a query editor specific to the data source selected in the panel. The query editor allows you to build a query that returns the data you want to visualize.

There are a wide variety of styling and formatting options for each panel. Panels can be dragged, dropped, and resized to rearrange them on the dashboard.

Before you add a panel, ensure that you have configured a data source.

- For details about using data sources, refer to Data sources.
- For more information about managing data sources as an administrator, refer to Data source management.

> **NOTE**
>
> Data source management is only available in Grafana Enterprise and Grafana Cloud.

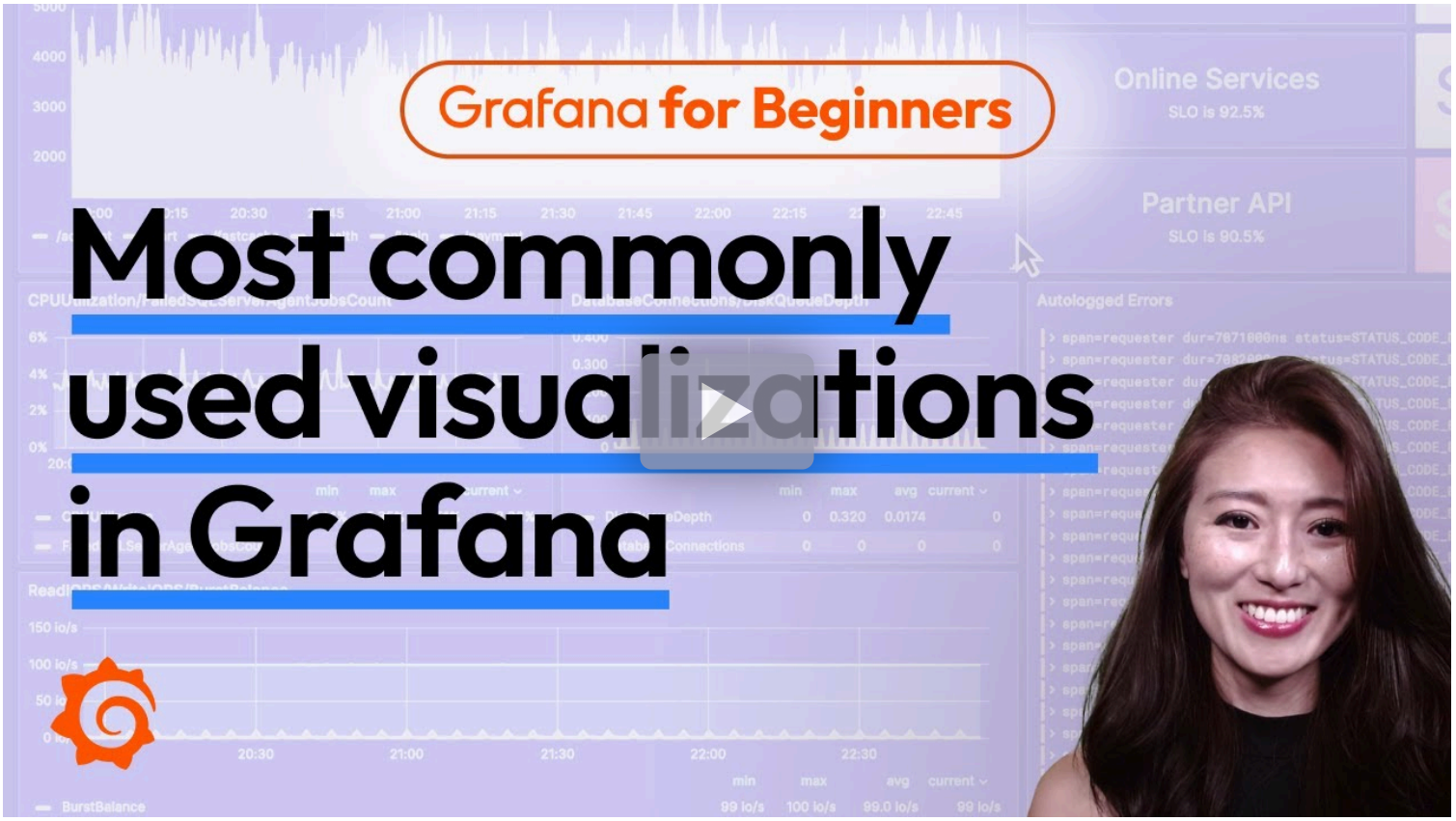
This section includes the following sub topics:

- Configure data links
- Configure value mappings
- Configure thresholds
- Configure field overrides

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Visualizations

Grafana offers a variety of visualizations to support different use cases. This section of the documentation highlights the built-in visualizations, their options and typical usage.



NOTE

If you are unsure which visualization to pick, Grafana can provide visualization suggestions based on the panel query. When you select a visualization, Grafana will show a preview with that visualization applied.

- Graphs & charts
  - Time series is the default and main Graph visualization.

    - State timeline for state changes over time.
    - Status history for periodic state over time.
    - Bar chart shows any categorical data.
    - Histogram calculates and shows value distribution in a bar chart.
    - Heatmap visualizes data in two dimensions, used typically for the magnitude of a phenomenon.
    - Pie chart is typically used where proportionality is important.
    - Candlestick is typically for financial data where the focus is price/data movement.
    - Gauge is the traditional rounded visual showing how far a single metric is from a threshold.
    - Trend for datasets that have a sequential, numeric x that is not time.
- Stats & numbers
    - Stat for big stats and optional sparkline.
    - Bar gauge is a horizontal or vertical bar gauge.
- Misc
    - Table is the main and only table visualization.
    - Logs is the main visualization for logs.
    - Node graph for directed graphs or networks.
    - Traces is the main visualization for traces.
    - Flame graph is the main visualization for profiling.
    - Canvas allows you to explicitly place elements within static and dynamic layouts.
    - Geomap helps you visualize geospatial data.
    - Datagrid allows you to create and manipulate data, and act as data source for other panels.
- Widgets
    - Dashboard list can list dashboards.
    - Alert list can list alerts.
    - Annotations list can list available annotations.
    - Text can show markdown and html.
    - News can show RSS feeds.

The following video shows you how to create gauge, time series line graph, stats, logs, and node graph visualizations:

Get more

You can add more visualization types by installing panel plugins.

Examples

Below you can find some good examples for how all the visualizations in Grafana can be configured. You can also explore play.grafana.org which has a large set of demo dashboards that showcase all the different visualizations.

Graphs

For time based line, area and bar charts we recommend the default time series visualization. This public demo dashboard contains many different examples for how this visualization can be configured and styled.

Time series

For categorical data use a bar chart.

Bar chart

Big numbers & stats

A stat shows one large stat value with an optional graph sparkline. You can control the background or value color using thresholds or color scales.

Stat

Gauge

If you want to present a value as it relates to a min and max value you have two options. First a standard radial gauge shown below.

Secondly Grafana also has a horizontal or vertical bar gauge with three different distinct display modes.

Table

To show data in a table layout, use a table.

Table visualization

Pie chart

To display reduced series, or values in a series, from one or more queries, as they relate to each other, use a pie chart.

Pie chart

Heatmaps

To show value distribution over, time use a heatmap.

Heatmap

State timeline

A state timeline shows discrete state changes over time. When used with time series, the thresholds are used to turn the numerical values into discrete state regions.

State timeline with string states

Time series

Time series visualizations are the default way to show the variations of a set of data values over time. Each data point is matched to a timestamp and this *time series* is displayed as a graph. The visualization can render series as lines, points, or bars and it's versatile enough to display almost any type of time-series data.

NOTE

You can migrate from the legacy Graph visualization to the time series visualization. To migrate, open the panel and click the **Migrate** button in the side pane.

A time series visualization displays an x-y graph with time progression on the x-axis and the magnitude of the values on the y-axis. This visualization is ideal for displaying large numbers of timed data points that would be hard to track in a table or list.

You can use the time series visualization if you need track:

- Temperature variations throughout the day
- The daily progress of your retirement account
- The distance you jog each day over the course of a year

Configure a time series visualization

The following video guides you through the creation steps and common customizations of time series visualizations, and is great for beginners:

Give it a try using Grafana Play

With Grafana Play, you can explore and see how it works, learning from practical examples to accelerate your development. This feature can be seen on Time Series Visualizations in Grafana.

Supported data formats

Time series visualizations require time-series data—a sequence of measurements, ordered in time, and formatted as a table—where every row in the table represents one individual measurement at a specific time. Learn more about time-series data.

The dataset must contain at least one numeric field, and in the case of multiple numeric fields, each one is plotted as a new line, point, or bar labeled with the field name in the tooltip.

Example 1

In the following example, there are three numeric fields represented by three lines in the chart:

Time	value1	value2	value3
2022-11-01 10:00:00	1	2	3
2022-11-01 11:00:00	4	5	6
2022-11-01 12:00:00	7	8	9
2022-11-01 13:00:00	4	5	6

If the time field isn't automatically detected, you might need to convert the data to a time format using a data transformation.

Example 2

The time series visualization also supports multiple datasets. If all datasets are in the correct format, the visualization plots the numeric fields of all datasets and labels them using the column name of the field.

Query1

Time	value1	value2	value3
2022-11-01 10:00:00	1	2	3
2022-11-01 11:00:00	4	5	6
2022-11-01 12:00:00	7	8	9

Query2

timestamp	number1	number2	number3
2022-11-01 10:30:00	11	12	13
2022-11-01 11:30:00	14	15	16
2022-11-01 12:30:00	17	18	19

timestamp	number1	number2	number3
2022-11-01 13:30:00	14	15	16

Example 3

If you want to more easily compare events between different, but overlapping, time frames, you can do this by using a time offset while querying the compared dataset:

Query1

Time	value1	value2	value3
2022-11-01 10:00:00	1	2	3
2022-11-01 11:00:00	4	5	6
2022-11-01 12:00:00	7	8	9

Query2

timestamp(-30min)	number1	number2	number3
2022-11-01 10:30:00	11	12	13
2022-11-01 11:30:00	14	15	16
2022-11-01 12:30:00	17	18	19
2022-11-01 13:30:00	14	15	16

When you add the offset, the resulting visualization makes the datasets appear to be occurring at the same time so that you can compare them more easily.

Alert rules

You can link alert rules to time series visualizations in the form of annotations to observe when alerts fire and are resolved. In addition, you can create alert rules from the **Alert** tab within the panel editor.

Special overrides

The following overrides help you further refine a time series visualization.

Transform override property

Use the **Graph styles > Transform** override property to transform series values without affecting the values shown in the tooltip, context menu, or legend. Choose from the following transform options:

- **Constant** - Show the first value as a constant line.
- **Negative Y transform** - Flip the results to negative values on the y-axis.

Fill below to override property

The **Graph styles > Fill below to** override property fills the area between two series. When you configure the property, select the series for which you want the fill to stop.

The following example shows three series: Min, Max, and Value. The Min and Max series have **Line width** set to 0. Max has a **Fill below to** override set to Min, which fills the area between Max and Min with the Max line color.

Multiple y-axes

In some cases, you might want to display multiple y-axes. For example, if you have a dataset showing both temperature and humidity over time, you might want to show two y-axes with different units for the two series.

You can configure multiple y-axes and control where they're displayed in the visualization by adding field overrides. This example of a dataset that includes temperature and humidity describes how you can configure that. Repeat the steps for every y-axis you wish to display.

Configuration options

The following section describes the configuration options available in the panel editor pane for this visualization. These options are, as much as possible, ordered as they appear in Grafana.

Panel options

In the **Panel options** section of the panel editor pane, set basic options like panel title and description, as well as panel links. To learn more, refer to Configure panel options.

Tooltip options

Tooltip options control the information overlay that appears when you hover over data points in the visualization.

Option	Description
Tooltip mode	When you hover your cursor over the visualization, Grafana can display tooltips. Choose how tooltips behave.
Values sort order	This option controls the order in which values are listed in a tooltip.
Hover proximity	Set the hover proximity (in pixels) to control how close the cursor must be to a data point to trigger the tooltip to display.
Max width	Set the maximum width of the tooltip box.

Option	Description
Max height	Set the maximum height of the tooltip box. The default is 600 pixels.

Tooltip mode

When you hover your cursor over the visualization, Grafana can display tooltips. Choose how tooltips behave.

- **Single -** The hover tooltip shows only a single series, the one that you are hovering over on the visualization.
- **All -** The hover tooltip shows all series in the visualization. Grafana highlights the series that you are hovering over in bold in the series list in the tooltip.
- **Hidden -** Do not display the tooltip when you interact with the visualization.

Use an override to hide individual series from the tooltip.

Values sort order

When you set the **Tooltip mode** to **All**, the **Values sort order** option is displayed. This option controls the order in which values are listed in a tooltip. Choose from the following:

- **None** - Grafana automatically sorts the values displayed in a tooltip.
- **Ascending** - Values in the tooltip are listed from smallest to largest.
- **Descending** - Values in the tooltip are listed from largest to smallest.

Hover proximity

Set the hover proximity (in pixels) to control how close the cursor must be to a data point to trigger the tooltip to display.

Legend options

Legend options control the series names and statistics that appear under or to the right of the graph. For more information about the legend, refer to Configure a legend.

Option	Description
Visibility	Toggle the switch to turn the legend on or off.
Mode	Use these settings to define how the legend appears in your visualization. **List** displays the legend as a list. This is a default display mode of the legend. **Table** displays the legend as a table.
Placement	Choose where to display the legend. **Bottom** places the legend below the graph. **Right** places the legend to the right of the graph.
Width	Control how wide the legend is when placed on the right side of the visualization. This option is only displayed if you set the legend placement to **Right**.
Values	Choose which of the standard calculations to show in the legend. You can have more than one.

Axis options

Options under the **Axis** section control how the x- and y-axes are rendered. Some options don't take effect until you click outside of the field option box you're editing. You can also press `Enter`.

Option	Description
Time zone	Set the desired time zones to display along the x-axis.
Placement	Select the placement of the y-axis.
Label	Set a y-axis text label. If you have more than one y-axis, then you can assign different labels using an override.
Width	Set a fixed width of the axis. By default, Grafana dynamically calculates the width of an axis. By setting the width of the axis, data with different axes types can share the same display proportions. This setting makes it easier for you to compare more than one graph's worth of data because the axes aren't shifted or stretched within visual proximity to each other.
Show grid lines	Set the axis grid line visibility.
Color	Set the color of the axis.
Show border	Set the axis border visibility.
Scale	Set the y-axis values scale.

Option	Description
Centered zero	Set the y-axis so it's centered on zero.
Soft min	Set a soft min to better control the y-axis limits. zero.
Soft max	Set a soft max to better control the y-axis limits. zero.

Placement

Select the placement of the y-axis. Choose from the following:

- **Auto** - Automatically assigns the y-axis to the series. When there are two or more series with different units, Grafana assigns the left axis to the first unit and the right axis to the units that follow.
- **Left** - Display all y-axes on the left side.
- **Right** - Display all y-axes on the right side.
- **Hidden** - Hide all axes. To selectively hide axes, Add a field override that targets specific fields.

Soft min and soft max

Set a **Soft min** or **soft max** option for better control of y-axis limits. By default, Grafana sets the range for the y-axis automatically based on the dataset.

Soft min and **soft max** settings can prevent small variations in the data from being magnified when it's mostly flat. In contrast, hard min and max values help prevent obscuring useful detail in the data by clipping intermittent spikes past a specific point.

To define hard limits of the y-axis, set standard min/max options. For more information, refer to Configure standard options.

Graph styles options

The options under the **Graph styles** section let you control the general appearance of the graph, excluding color.

Option	Description
Style	Choose whether to display your time-series data as lines, bars, or points.
Line interpolation	Choose how the graph interpolates the series line.
Line width	Set the thickness of the series lines or the outline for bars using the **Line width** slider.
Fill opacity	Set the series area fill color using the **Fill opacity** slider.
Gradient mode	Choose a gradient mode to control the gradient fill, which is based on the series color.
Line style	Choose a solid, dashed, or dotted line style.
Connect null values	Choose how null values, which are gaps in the data, appear on the graph.
Disconnect values	Choose whether to set a threshold above which values in the data should be disconnected.
Show points	Set whether to show data points to lines or bars.
Point size	Set the size of the points, from 1 to 40 pixels in diameter.
Stack series	Set whether Grafana displays series on top of each other.
Bar alignment	Set the position of the bar relative to a data point.
Bar width factor	Set the width of the bar relative to minimum space between data points. A factor of 0.5 means that the bars take up half of the available space between data points. A factor of 1.0 means that the bars take up all available space.

Style

Choose whether to display your time-series data as lines, bars, or points. You can use overrides to combine multiple styles in the same graph. Choose from the following:

Line interpolation

Choose how the graph interpolates the series line:

- **Linear** - Points are joined by straight lines.
- **Smooth** - Points are joined by curved lines that smooths transitions between points.
- **Step before** - The line is displayed as steps between points. Points are rendered at the end of the step.

- **Step after** - The line is displayed as steps between points. Points are rendered at the beginning of the step.

Line width

Set the thickness of the series lines or the outline for bars using the **Line width** slider.

Fill opacity

Set the series area fill color using the **Fill opacity** slider.

Gradient mode

Choose a gradient mode to control the gradient fill, which is based on the series color. To change the color, use the standard color scheme field option. For more information, refer to Color scheme.

- **None** - No gradient fill. This is the default setting.
- **Opacity** - An opacity gradient where the opacity of the fill increases as y-axis values increase.
- **Hue** - A subtle gradient that's based on the hue of the series color.
- **Scheme** - A color gradient defined by your Color scheme. This setting is used for the fill area and line. For more information about scheme, refer to Scheme gradient mode.

Gradient appearance is influenced by the **Fill opacity** setting. The following image shows the **Fill opacity** set to 50.

Scheme gradient mode

The **Gradient mode** option located under the **Graph styles** section has a mode called **Scheme**. When you enable **Scheme**, the line or bar receives a gradient color defined from the selected **Color scheme**.

From thresholds

If the **Color scheme** is set to **From thresholds (by value)** and **Gradient mode** is set to **Scheme**, then the line or bar color changes as it crosses the defined thresholds.

Gradient color schemes

The following image shows a line chart with the **Green-Yellow-Red (by value)** color scheme option selected.

Line style

Choose a solid, dashed, or dotted line style:

- **Solid** - Display a solid line. This is the default setting.
- **Dash** - Display a dashed line. When you choose this option, a list appears for you to select the length and gap (length, gap) for the line dashes. Dash spacing is 10, 10 by default.
- **Dots** - Display dotted lines. When you choose this option, a list appears for you to select the gap (length = 0, gap) for the dot spacing. Dot spacing is 0, 10 by default.

Connect null values

Choose how null values, which are gaps in the data, appear on the graph. Null values can be connected to form a continuous line or set to a threshold above which gaps in the data are no longer connected.

- **Never** - Time series data points with gaps in the data are never connected.
- **Always** - Time series data points with gaps in the data are always connected.
- **Threshold** - Specify a threshold above which gaps in the data are no longer connected. This can be useful when the connected gaps in the data are of a known size and/or within a known range, and gaps outside this range should no longer be connected.

Disconnect values

Choose whether to set a threshold above which values in the data should be disconnected.

- **Never** - Time series data points in the data are never disconnected.
- **Threshold** - Specify a threshold above which values in the data are disconnected. This can be useful when desired values in the data are of a known size and/or within a known range, and values outside this range should no longer be connected.

To change the color, use the standard color scheme field option.

Show points

Set whether to show data points as lines or bars. Choose from the following:

- **Auto** - Grafana determines a point's visibility based on the density of the data. If the density is low, then points appear.
- **Always** - Show the points regardless of how dense the data set is.
- **Never** - Don't show points.

Stack series

Set whether Grafana stacks or displays series on top of each other. Be cautious when using stacking because it can create misleading graphs. To read more about why stacking might not be the best approach, refer to The issue with stacking. Choose from the following:

- **Off** - Turns off series stacking. When **Off**, all series share the same space in the visualization.
- **Normal** - Stacks series on top of each other.
- **100%** - Stack by percentage where all series add up to 100%.

Stack series in groups

The stacking group option is only available as an override. For more information about creating an override, refer to Configure field overrides.

1. Edit the panel and click **Overrides**.
2. Create a field override for the **Stack series** option.
3. In stacking mode, click **Normal**.
4. Name the stacking group in which you want the series to appear.

   The stacking group name option is only available when you create an override.

Bar alignment

Set the position of the bar relative to a data point. In the examples below, **Show points** is set to **Always** which makes it easier to see the difference this setting makes. The points don't change, but the bars change in relationship to the points. Choose from the following:

- **Before** The bar is drawn before the point. The point is placed on the trailing corner of the bar.
- **Center** The bar is drawn around the point. The point is placed in the center of the bar. This is the default.
- **After** The bar is drawn after the point. The point is placed on the leading corner of the bar.

Standard options

Standard options in the panel editor pane let you change how field data is displayed in your visualizations. When you set a standard option, the change is applied to all fields or series. For more granular control over the display of fields, refer to Configure overrides.

Option	Description
Unit	Choose which unit a field should use.
Min/Max	Set the minimum and maximum values used in percentage threshold calculations or leave these field empty for them to be calculated automatically.
Field min/max	Enable **Field min/max** to have Grafana calculate the min or max of each field individually, based on the minimum or maximum value of the field.
Decimals	Specify the number of decimals Grafana includes in the rendered value.
Display name	Set the display title of all fields. You can use variables in the field title.
Color scheme	Set single or multiple colors for your entire visualization.
No value	Enter what Grafana should display if the field value is empty or null. The default value is a hyphen (-).

To learn more, refer to Configure standard options.

Data links

Data links allow you to link to other panels, dashboards, and external resources while maintaining the context of the source panel. You can create links that include the series name or even the value under the cursor.

For each data link, set the following options:

- **Title**
- **URL**
- **Open in new tab**

To learn more, refer to Configure data links.

Value mappings

Value mapping is a technique you can use to change how data appears in a visualization.

For each value mapping, set the following options:

- **Condition** - Choose what's mapped to the display text and (optionally) color:
  - **Value** - Specific values
  - **Range** - Numerical ranges
  - **Regex** - Regular expressions
  - **Special** - Special values like `Null`, `NaN` (not a number), or boolean values like `true` and `false`
- **Display text**

- **Color** (Optional)
- **Icon** (Canvas only)

To learn more, refer to Configure value mappings.

Thresholds

A threshold is a value or limit you set for a metric that's reflected visually when it's met or exceeded. Thresholds are one way you can conditionally style and color your visualizations based on query results.

For each threshold, set the following options:

Option	Description
Value	Set the value for each threshold.
Thresholds mode	Choose from **Absolute** and **Percentage**.
Show thresholds	Choose from a variety of display options including not displaying thresholds at all.

To learn more, refer to Configure thresholds.

Field overrides

Overrides allow you to customize visualization settings for specific fields or series. When you add an override rule, it targets a particular set of fields and lets you define multiple options for how that field is displayed.

Choose from the following override options:

Option	Description
Fields with name	Select a field from the list of all available fields.
Field with name matching regex	Specify fields to override with a regular expression.
Fields with type	Select fields by type, such as string, numeric, or time.
Fields returned by query	Select all fields returned by a specific query, such as A, B, or C.
Fields with values	Select all fields returned by your defined reducer condition, such as **Min**, **Max**, **Count**, **Total**.

To learn more, refer to Configure field overrides.

Alert list

Alert lists allow you to display a list of important alerts that you want to track. You can configure the alert list to show the current state of your alert, such as firing, pending, or normal. Learn more about alerts in Grafana Alerting overview.

On each dashboard load, this visualization queries the alert list, always providing the most up-to-date results.

Give it a try using Grafana Play

With Grafana Play, you can explore and see how it works, learning from practical examples to accelerate your development. This feature can be seen on Alert List.

Configure an alert list

Once you've created a dashboard, the following video shows you how to configure an alert list visualization:

Panel options

In the **Panel options** section of the panel editor pane, set basic options like panel title and description, as well as panel links. To learn more, refer to Configure panel options.

Options

Use the following options to refine your alert list visualization.

View mode

Choose between **List** to display alerts in a detailed list format with comprehensive information, or **Stat** to show alerts as a summarized single-value statistic.

Group mode

Choose between **Default grouping** to show alert instances grouped by their alert rule, or **Custom grouping** to show alert instances grouped by a custom set of labels.

Max items

Sets the maximum number of alerts to list. By default, Grafana sets this value to 10.

Sort order

Select how to order the alerts displayed. Choose from:

- **Alphabetical (asc) -** Alphabetical order.
- **Alphabetical (desc) -** Reverse alphabetical order.
- **Importance -** By importance according to the following values, with 1 being the highest:
  - alerting: 1
  - firing: 1
  - no_data: 2
  - pending: 3
  - ok: 4
  - paused: 5
  - inactive: 5
- **Time (asc) -** Newest active alert instances first.
- **Time (desc) -** Oldest active alert instances first.

Alerts linked to this dashboard

Toggle the switch on to only show alerts from the dashboard the alert list is in.

Filter

These options allow you to limit alerts shown to only those that match the query, folder, or tags you choose.

Alert name

Filter alerts by name.

Alert instance label

Filter alert instances using label querying. For example, `{severity="critical", instance=~"cluster-us-.+"}`.

Datasource

Filter alerts from the selected data source.

Folder

Filter alerts by the selected folder. Only alerts from dashboards in this folder are displayed.

Alert state filter

Choose which alert states to display in this visualization.

- **Alerting / Firing -** Shows alerts that are currently active and triggering an alert condition.
- **Pending -** Shows alerts that are in a transitional state, waiting for conditions to be met before triggering.
- **No Data -** Shows alerts where the data source is not returning any data, which could indicate an issue with data collection.
- **Normal -** Shows alerts that are in a normal or resolved state, where no alert condition is currently met.
- **Error -** Shows alerts where an error has occurred, typically related to an issue in the alerting process.

Annotations list

The annotations list shows a list of available annotations you can use to view annotated data. Various options are available to filter the list based on tags and on the current dashboard.

Panel options

In the **Panel options** section of the panel editor pane, set basic options like panel title and description, as well as panel links. To learn more, refer to Configure panel options.

Annotation query

The following options control the source query for the list of annotations.

Query Filter

Use the query filter to create a list of annotations from all dashboards in your organization or the current dashboard in which this panel is located. It has the following options:

- All dashboards - List annotations from all dashboards in the current organization.
- This dashboard - Limit the list to the annotations on the current dashboard.

Time Range

Use the time range option to specify whether the list should be limited to the current time range. It has the following options:

- None - no time range limit for the annotations query.
- This dashboard - Limit the list to the time range of the dashboard where the annotations list is available.

Tags

Use the tags option to filter the annotations by tags. You can add multiple tags in order to refine the list.

> **NOTE**
>
> Optionally, leave the tag list empty and filter on the fly by selecting tags that are listed as part of the results on the panel itself.

Limit

Use the limit option to limit the number of results returned.

Display

These options control additional meta-data included in the annotations list display.

Show user

Use this option to show or hide which user created the annotation.

Show time

Use this option to show or hide the time the annotation creation time.

Show Tags

Use this option to show or hide the tags associated with an annotation. *NB*: You can use the tags to live-filter the annotations list on the visualization itself.

Link behavior

Link target

Use this option to chose how to view the annotated data. It has the following options.

- Panel - This option will take you directly to a full-screen view of the panel with the corresponding annotation
- Dashboard - This option will focus the annotation in the context of a complete dashboard

Time before

Use this option to set the time range before the annotation. Use duration string values like “1h” = 1 hour, “10m” = 10 minutes, etc.

Time after

Use this option to set the time range after the annotation.

Bar chart

A bar chart is a visual representation that uses rectangular bars, where the length of each bar represents each value. You can use the bar chart visualization when you want to compare values over different categories or time periods. The visualization can display the bars horizontally or vertically, and can be customized to group or stack bars for more complex data analysis.

You can use the bar chart visualization if you need to show:

- Population distribution by age or location
- CPU usage per application
- Sales per division
- Server cost distribution

Configure a bar chart

The following video shows you how to create and configure a bar chart visualization:

Give it a try using Grafana Play

With Grafana Play, you can explore and see how it works, learning from practical examples to accelerate your development. This feature can be seen on Grafana Bar Charts and Pie Charts.

Supported data formats

To create a bar chart visualization, you need a dataset containing one string or time field (or column) and at least one numeric field, though preferably more than one to make best use of the visualization.

The text or time field is used to label the bars or values in each row of data and the numeric fields are represented by proportionally sized bars.

Example 1

Group	Value1	Value2	Value3
uno	5	3	2

If you have more than one text or time field, by default, the visualization uses the first one, but you can change this in the x-axis option as described in the Bar chart options section.

Example 2

If your dataset contains multiple rows, the visualization displays multiple bar chart groups where each group contains multiple bars representing all the numeric values for a row.

Group	Value1	Value2	Value3
uno	5	3	2
dos	10	6	4
tres	20	8	2

While the first field can be time-based and you can use a bar chart to plot time-series data, for large amounts of time-series data, we recommend that you use the time series visualization and configure it to be displayed as bars.

We recommend that you only use one dataset in a bar chart because using multiple datasets can result in unexpected behavior.

Panel options

In the **Panel options** section of the panel editor pane, set basic options like panel title and description, as well as panel links. To learn more, refer to Configure panel options.

Bar chart options

Use these options to refine your visualization.

X Axis

Specify which field is used for the x-axis.

Orientation

- **Auto** - Grafana decides the bar orientation based on what the panel dimensions.
- **Horizontal** - Will make the X axis the category axis.
- **Vertical** - Will make the Y axis the category axis.

Rotate x-axis tick labels

When the graph is vertically oriented, this setting rotates the labels under the bars. This setting is useful when bar chart labels are long and overlap.

X-axis tick label maximum length

Sets the maximum length of bar chart labels. Labels longer than the maximum length are truncated, and appended with `...`.

Bar labels minimum spacing

Sets the minimum spacing between bar labels.

Show values

This controls whether values are shown on top or to the left of bars.

- **Auto** Values will be shown if there is space
- **Always** Always show values.
- **Never** Never show values.

Stacking

Controls bar chart stacking.

- **Off**: Bars will not be stacked.
- **Normal**: Bars will be stacked on each other.

- **Percent**: Bars will be stacked on each other, and the height of each bar is the percentage of the total height of the stack.

Group width

Controls the width of groups. 1 = max with, 0 = min width.

Bar width

Controls the width of bars. 1 = Max width, 0 = Min width.

Bar radius

Controls the radius of the bars.

- 0 = Minimum radius
- 0.5 = Maximum radius

Highlight full area on cover

Controls if the entire surrounding area of the bar is highlighted when you hover over the bar.

Line width

Controls line width of the bars.

Fill opacity

Controls the fill opacity bars.

Gradient mode

Set the mode of the gradient fill. Fill gradient is based on the line color. To change the color, use the standard color scheme field option.

Gradient appearance is influenced by the **Fill opacity** setting.

None

No gradient fill. This is the default setting.

Opacity

Transparency of the gradient is calculated based on the values on the y-axis. Opacity of the fill is increasing with the values on the Y-axis.

Hue

Gradient color is generated based on the hue of the line color.

Scheme gradient mode

The **Gradient mode** option located under the **Graph styles** has a mode named **Scheme**. When you enable **Scheme**, the bar receives a gradient color defined from the selected **Color scheme**.

From thresholds

If the **Color scheme** is set to **From thresholds (by value)** and **Gradient mode** is set to **Scheme**, then the bar color changes as they cross the defined thresholds.

Color scheme: From thresholds

Gradient color schemes

The following image shows a bar chart with the **Green-Yellow-Red (by value)** color scheme option selected.

Color scheme: Green-Yellow-Red

Tooltip options

Tooltip options control the information overlay that appears when you hover over data points in the visualization.

Tooltip mode

When you hover your cursor over the visualization, Grafana can display tooltips. Choose how tooltips behave.

- **Single -** The hover tooltip shows only a single series, the one that you are hovering over on the visualization.
- **All -** The hover tooltip shows all series in the visualization. Grafana highlights the series that you are hovering over in bold in the series list in the tooltip.
- **Hidden -** Do not display the tooltip when you interact with the visualization.

Use an override to hide individual series from the tooltip.

Values sort order

When you set the **Tooltip mode** to **All**, the **Values sort order** option is displayed. This option controls the order in which values are listed in a tooltip. Choose from the following:

- **None** - Grafana automatically sorts the values displayed in a tooltip.
- **Ascending** - Values in the tooltip are listed from smallest to largest.
- **Descending** - Values in the tooltip are listed from largest to smallest.

Max height

Set the maximum height of the tooltip box. The default is 600 pixels.

Legend options

Legend options control the series names and statistics that appear under or to the right of the graph. For more information about the legend, refer to Configure a legend.

Option	Description
Visibility	Toggle the switch to turn the legend on or off.
Mode	Use these settings to define how the legend appears in your visualization. **List** displays the legend as a list. This is a default display mode of the legend. **Table** displays the legend as a table.
Placement	Choose where to display the legend. **Bottom** places the legend below the graph. **Right** places the legend to the right of the graph.
Width	Control how wide the legend is when placed on the right side of the visualization. This option is only displayed if you set the legend placement to **Right**.

Option	Description
Values	Choose which of the standard calculations to show in the legend. You can have more than one.

Text size

Enter a **Value** to change the size of the text on your bar chart.

Axis

Use the following field settings to refine how your axes display.

Some field options will not affect the visualization until you click outside of the field option box you are editing or press Enter.

Placement

Select the placement of the Y-axis.

Auto

Grafana automatically assigns Y-axis to the series. When there are two or more series with different units, then Grafana assigns the left axis to the first unit and right to the following units.

Left

Display all Y-axes on the left side.

Right

Display all Y-axes on the right side.

Hidden

Hide all axes.

To selectively hide axes, Add a field override that targets specific fields.

Label

Set a Y-axis text label.

If you have more than one Y-axis, then you can give assign different labels with an override.

Width

Set a fixed width of the axis. By default, Grafana dynamically calculates the width of an axis.

By setting the width of the axis, data whose axes types are different can share the same display proportions. This makes it easier to compare more than one graph's worth of data because the axes are not shifted or stretched within visual proximity of each other.

Soft min and soft max

Set a **Soft min** or **soft max** option for better control of Y-axis limits. By default, Grafana sets the range for the Y-axis automatically based on the dataset.

Soft min and **soft max** settings can prevent blips from turning into mountains when the data is mostly flat, and hard min or max derived from standard min and max field options can prevent intermittent spikes from flattening useful detail by clipping the spikes past a defined point.

You can set standard min/max options to define hard limits of the Y-axis. For more information, refer to Standard options definitions.

Multiple y-axes

In some cases, you might want to display multiple y-axes. For example, if you have a dataset showing both temperature and humidity over time, you might want to show two y-axes with different units for the two series.

You can configure multiple y-axes and control where they're displayed in the visualization by adding field overrides. This example of a dataset that includes temperature and humidity describes how you can configure that. Repeat the steps for every y-axis you wish to display.

Standard options

Standard options in the panel editor pane let you change how field data is displayed in your visualizations. When you set a standard option, the change is applied to all fields or series. For more granular control over the display of fields, refer to Configure overrides.

Option	Description
Unit	Choose which unit a field should use.
Min/Max	Set the minimum and maximum values used in percentage threshold calculations or leave these field empty for them to be calculated automatically.
Field min/max	Enable **Field min/max** to have Grafana calculate the min or max of each field individually, based on the minimum or maximum value of the field.
Decimals	Specify the number of decimals Grafana includes in the rendered value.
Display name	Set the display title of all fields. You can use variables in the field title.
Color scheme	Set single or multiple colors for your entire visualization.

Option	Description
No value	Enter what Grafana should display if the field value is empty or null. The default value is a hyphen (-).

To learn more, refer to Configure standard options.

Data links

Data links allow you to link to other panels, dashboards, and external resources while maintaining the context of the source panel. You can create links that include the series name or even the value under the cursor.

For each data link, set the following options:

- **Title**
- **URL**
- **Open in new tab**

To learn more, refer to Configure data links.

Thresholds

A threshold is a value or limit you set for a metric that's reflected visually when it's met or exceeded. Thresholds are one way you can conditionally style and color your visualizations based on query results.

For each threshold, set the following options:

Option	Description
Value	Set the value for each threshold.
Thresholds mode	Choose from **Absolute** and **Percentage**.
Show thresholds	Choose from a variety of display options including not displaying thresholds at all.

To learn more, refer to Configure thresholds.

Value mappings

Value mapping is a technique you can use to change how data appears in a visualization.

For each value mapping, set the following options:

- **Condition** - Choose what's mapped to the display text and (optionally) color:
  - **Value** - Specific values

    - **Range** - Numerical ranges
    - **Regex** - Regular expressions
    - **Special** - Special values like `Null`, `NaN` (not a number), or boolean values like `true` and `false`
- **Display text**
- **Color** (Optional)
- **Icon** (Canvas only)

To learn more, refer to Configure value mappings.

Field overrides

Overrides allow you to customize visualization settings for specific fields or series. When you add an override rule, it targets a particular set of fields and lets you define multiple options for how that field is displayed.

Choose from the following override options:

Option	Description
Fields with name	Select a field from the list of all available fields.
Field with name matching regex	Specify fields to override with a regular expression.
Fields with type	Select fields by type, such as string, numeric, or time.
Fields returned by query	Select all fields returned by a specific query, such as A, B, or C.
Fields with values	Select all fields returned by your defined reducer condition, such as **Min**, **Max**, **Count**, **Total**.

To learn more, refer to Configure field overrides.

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Bar gauge

Bar gauges simplify your data by reducing every field to a single value. You choose how Grafana calculates the reduction. This visualization can show one or more bar gauges depending on how many series, rows, or columns your query returns.

The bar gauge visualization displays values as bars with various lengths or fills proportional to the values they represent. They differ from traditional bar charts in that they act as gauges displaying metrics between ranges. One example is a thermometer displaying body temperature in a bar filling up.

You can use a bar gauge visualization when you need to show:

- Key performance indicators (KPIs)
- System health
- Savings goals
- Attendance
- Process completion rates

Give it a try using Grafana Play

With Grafana Play, you can explore and see how it works, learning from practical examples to accelerate your development. This feature can be seen on Bar Gauge.

Supported data formats

To create a bar gauge visualization, you need a dataset querying at least one numeric field. Every numeric field in the dataset is displayed as a bar gauge. Text or time fields aren't required but if they're present, they're used for labeling.

Example 1

Label	Value1	Value2	Value3
Row1	5	3	2

The minimum and maximum range for the bar gauges is automatically pulled from the largest and smallest numeric values in the dataset. You can also manually define the minimum and maximum values as indicated in the Standard options section.

You can also define the minimum and maximum from the dataset provided.

Example 2

Label	Value	Max	Min
Row1	3	6	1

If you don’t want to show gauges for the min and max values, you can configure only one field to be displayed as described in the Value options section.

Even if the min and max aren’t displayed, the visualization still pulls the range from the data set.

Example 3

The bar gauge visualization also supports multiple records (rows) in the dataset.

Label	Value1	Value2	Value3
Row1	5	3	2
Row2	10	6	4
Row3	20	8	2

By default, the visualization is configured to calculate a single value per column or series and to display only the last set of data. However, it derives the minimum and maximum from the full dataset even if those values aren’t visible. In this example, that means only the last row of data is

displayed in the gauges and the minimum and maximum values are defined as 2 and 20, pulled from the whole dataset.

If you want to show one gauge per cell you can change the Show setting from Calculate to All values and each bar is labeled by concatenating the text column with each value's column name.

Panel options

In the **Panel options** section of the panel editor pane, set basic options like panel title and description, as well as panel links. To learn more, refer to Configure panel options.

Value options

Use the following options to refine how your visualization displays the value:

Show

Choose how Grafana displays your data.

Calculate

Show a calculated value based on all rows.

- **Calculation -** Select a reducer function that Grafana will use to reduce many fields to a single value. For a list of available calculations, refer to Calculation types.
- **Fields -** Select the fields display in the panel.

All values

Show a separate stat for every row. If you select this option, then you can also limit the number of rows to display.

- **Limit -** The maximum number of rows to display. Default is 5,000.
- **Fields -** Select the fields display in the panel.

Bar gauge options

Adjust how the bar gauge is displayed.

Orientation

Choose a stacking direction.

- **Auto -** Grafana determines the best orientation.
- **Horizontal -** Bars stretch horizontally, left to right.
- **Vertical -** Bars stretch vertically, bottom to top.

Display mode

Choose a display mode.

- **Gradient -** Threshold levels define a gradient.
- **Retro LCD -** The gauge is split into small cells that are lit or unlit.
- **Basic -** Single color based on the matching threshold.

Value display

Choose a value display mode.

- **Value color -** Value color is determined by value.
- **Text color -** Value color is default text color.
- **Hidden -** Values are hidden.

Name placement

Choose a name placement mode.

> **NOTE**
>
> This option only applies when the orientation of the bar gauge is horizontal. When the bar gauge is in the vertical orientation, names are always placed at the bottom of each bar gauge.

- **Auto -** Grafana determines the best placement.
- **Top -** Names are placed on top of each bar gauge.
- **Left -** Names are placed to the left of each bar gauge.

Show unfilled area

Select this if you want to render the unfilled region of the bars as dark gray. Not applicable to Retro LCD display mode.

Bar size

Choose a bar size mode.

- **Auto -** Grafana determines the best bar gauge size.
- **Manual -** Manually configure the bar gauge size.

Min width

Limit the minimum width of the bar column when the gauge is oriented vertically.

Automatically show x-axis scrollbar when there's a large amount of data.

> **NOTE**
> This option only applies when bar size is set to manual.

Min height

Limit the minimum height of the bar row when the gauge is oriented horizontally.

Automatically show y-axis scrollbar when there's a large amount of data.

> **NOTE**
> This option only applies when bar size is set to manual.

Max height

Limit the maximum height of the bar row when the gauge is oriented horizontally.

Automatically show y-axis scrollbar when there's a large amount of data.

> **NOTE**
> This option only applies when bar size is set to manual.

Standard options

Standard options in the panel editor pane let you change how field data is displayed in your visualizations. When you set a standard option, the change is applied to all fields or series. For more granular control over the display of fields, refer to Configure overrides.

Option	Description
Unit	Choose which unit a field should use.
Min/Max	Set the minimum and maximum values used in percentage threshold calculations or leave these field empty for them to be calculated automatically.
Field min/max	Enable **Field min/max** to have Grafana calculate the min or max of each field individually, based on the minimum or maximum value of the field.
Decimals	Specify the number of decimals Grafana includes in the rendered value.
Display name	Set the display title of all fields. You can use variables in the field title.
Color scheme	Set single or multiple colors for your entire visualization.
No value	Enter what Grafana should display if the field value is empty or null. The default value is a hyphen (-).

To learn more, refer to Configure standard options.

Data links

Data links allow you to link to other panels, dashboards, and external resources while maintaining the context of the source panel. You can create links that include the series name or even the value under the cursor.

For each data link, set the following options:

- **Title**
- **URL**
- **Open in new tab**

To learn more, refer to Configure data links.

Value mappings

Value mapping is a technique you can use to change how data appears in a visualization.

For each value mapping, set the following options:

- **Condition** - Choose what's mapped to the display text and (optionally) color:
  - **Value** - Specific values
  - **Range** - Numerical ranges
  - **Regex** - Regular expressions
  - **Special** - Special values like `Null`, `NaN` (not a number), or boolean values like `true` and `false`

- **Display text**
- **Color** (Optional)
- **Icon** (Canvas only)

To learn more, refer to Configure value mappings.

Thresholds

A threshold is a value or limit you set for a metric that's reflected visually when it's met or exceeded. Thresholds are one way you can conditionally style and color your visualizations based on query results.

Set the following options:

- **Value** - Set the value for each threshold.
- **Thresholds mode** - Choose from:
  - **Absolute**
  - **Percentage**

To learn more, refer to Configure thresholds.

Last, colors of the bar gauge thresholds can be configured as described above.

Field overrides

Overrides allow you to customize visualization settings for specific fields or series. When you add an override rule, it targets a particular set of fields and lets you define multiple options for how that field is displayed.

Choose from the following override options:

Option	Description
Fields with name	Select a field from the list of all available fields.
Field with name matching regex	Specify fields to override with a regular expression.

Option	Description
Fields with type	Select fields by type, such as string, numeric, or time.
Fields returned by query	Select all fields returned by a specific query, such as A, B, or C.
Fields with values	Select all fields returned by your defined reducer condition, such as **Min**, **Max**, **Count**, **Total**.

To learn more, refer to Configure field overrides.

Candlestick

The candlestick visualization allows you to visualize data that includes a number of consistent dimensions focused on price movements, such as stock prices. The candlestick visualization includes an Open-High-Low-Close (OHLC) mode, as well as support for additional dimensions based on time series data.

Candlestick visualizations build upon the foundation of the time series visualization and include many common configuration settings.

You can use a candlestick if you want to visualize, at a glance, how a price moved over time, whether it went up, down, or stayed the same, and how much it fluctuated:

Each candlestick is represented as a rectangle, referred to as the *candlestick body*. The candlestick body displays the opening and closing prices during a time period. Green candlesticks

represent when the price appreciated while the red candlesticks represent when the price depreciated. The lines sticking out the candlestick body are referred to as *wicks* or *shadows*, which represent the highest and lowest prices during the time period.

Use a candlestick when you need to:

- Monitor and identify trends in price movements of specific assets such as stocks, currencies, or commodities.
- Analyze any volatility in the stock market.
- Provide data analysis to help with trading decisions.

Configure a candlestick

Once you've created a dashboard, the following video shows you how to configure a candlestick visualization:

Give it a try using Grafana Play

With Grafana Play, you can explore and see how it works, learning from practical examples to accelerate your development. This feature can be seen on Candlestick.

Supported data formats

The candlestick visualization works best with price movement data for an asset. The data must include:

- **Timestamps** - The time at which each price movement occurred.
- **Opening price** - The price of the asset at the beginning of the time period.
- **Closing price** - The price of the asset at the end of the time period.
- **Highest price** - The highest price the asset reached during the time period.
- **Lowest price** - The lowest price the asset reached during the time period.

Example

Timestamps	Open	High	Low	Close
2024-03-13 10:05:00	0.200	0.205	0.201	0.203
2024-03-14 10:10:10	0.204	0.205	0.201	0.200
2024-03-15 10:15:10	0.204	0.205	0.201	0.200
2024-03-16 10:20:11	0.203	0.203	0.202	0.203
2024-03-17 10:25:11	0.203	0.203	0.202	0.203
2024-03-18 10:30:12	0.202	0.202	0.201	0.201

The data is converted as follows:

Panel options

In the **Panel options** section of the panel editor pane, set basic options like panel title and description, as well as panel links. To learn more, refer to Configure panel options.

Mode

The mode options allow you to toggle which dimensions are used for the visualization.

- **Candles** limits the panel dimensions to the open, high, low, and close dimensions used by candlestick visualizations.

- **Volume** limits the panel dimension to the volume dimension.
- **Both** is the default behavior for the candlestick visualization. It includes both candlestick and volume visualizations.

Candle style

- **Candles** is the default display style and creates candle-style visualizations between the open and close dimensions.
- **OHLC Bars** displays the four core dimensions open, high, low, and close values.

Color strategy

- **Since Open** is the default behavior. This mode will utilize the *Up* color (below) if the intra-period price movement is positive. In other words, if the value on close is greater or equal to the value on open, the *Up* color is used.
- **Since Prior Close** is an alternative display method based where the color of the candle is based on the inter-period price movement or change in value. In other words, if the value on open is greater than the previous value on close, the *Up* color is used. If the value on open is lower than the previous value on close, the *Down* color is used. *This option also triggers the hollow candlestick visualization mode*. Hollow candlesticks indicate that the intra-period movement is positive (value is higher on close than on open), filled candlesticks indicate the intra-period change is negative (value is lower on close than on open). To learn more, see the explanation of the differences.

Up & Down Colors

The **Up color** and **Down color** options select which colors are used when the price movement is up or down. Please note that the *Color strategy* above will determine if intra-period or inter-period price movement is used to select the candle or OHLC bar color.

Open, High, Low, Close

The candlestick visualization will attempt to map fields from your data to the appropriate dimension:

- **Open** corresponds to the starting value of the given period.
- **High** corresponds to the highest value of the given period.
- **Low** corresponds to the lowest value of the given period.
- **Close** corresponds to the final (end) value of the given period.
- **Volume** corresponds to the sample count in the given period. (for example, number of trades)

NOTE

The candlestick visualization legend doesn't display these values.

If your data can't be mapped to these dimensions for some reason (for example, because the column names aren't the same), you can map them manually using the **Open**, **High**, **Low**, and **Close** fields under the **Candlestick** options in the panel editor:

Additional fields

The candlestick visualization is based on the time series visualization. It can visualize additional data dimensions beyond open, high, low, close, and volume The **Include** and **Ignore** options allow it to visualize other included data such as simple moving averages, Bollinger bands and more, using the same styles and configurations available in the time series visualization.

Standard options

Standard options in the panel editor pane let you change how field data is displayed in your visualizations. When you set a standard option, the change is applied to all fields or series. For more granular control over the display of fields, refer to Configure overrides.

Option	Description
Unit	Choose which unit a field should use.
Min/Max	Set the minimum and maximum values used in percentage threshold calculations or leave these field empty for them to be calculated automatically.
Field min/max	Enable **Field min/max** to have Grafana calculate the min or max of each field individually, based on the minimum or maximum value of the field.

Option	Description
Decimals	Specify the number of decimals Grafana includes in the rendered value.
Display name	Set the display title of all fields. You can use variables in the field title.
Color scheme	Set single or multiple colors for your entire visualization.
No value	Enter what Grafana should display if the field value is empty or null. The default value is a hyphen (-).

To learn more, refer to Configure standard options.

Legend options

Legend options control the series names and statistics that appear under or to the right of the graph. For more information about the legend, refer to Configure a legend.

Option	Description
Visibility	Toggle the switch to turn the legend on or off.
Mode	Use these settings to define how the legend appears in your visualization. **List** displays the legend as a list. This is a default display mode of the legend. **Table** displays the legend as a table.
Placement	Choose where to display the legend. **Bottom** places the legend below the graph. **Right** places the legend to the right of the graph.
Width	Control how wide the legend is when placed on the right side of the visualization. This option is only displayed if you set the legend placement to **Right**.
Values	Choose which of the standard calculations to show in the legend. You can have more than one.

Tooltip options

Tooltip options control the information overlay that appears when you hover over data points in the visualization.

Option	Description
Tooltip mode	When you hover your cursor over the visualization, Grafana can display tooltips. Choose how tooltips behave.
Values sort order	This option controls the order in which values are listed in a tooltip.

Option	Description
Hover proximity	Set the hover proximity (in pixels) to control how close the cursor must be to a data point to trigger the tooltip to display.
Max width	Set the maximum width of the tooltip box.
Max height	Set the maximum height of the tooltip box. The default is 600 pixels.

Tooltip mode

When you hover your cursor over the visualization, Grafana can display tooltips. Choose how tooltips behave.

- **Single -** The hover tooltip shows only a single series, the one that you are hovering over on the visualization.
- **All -** The hover tooltip shows all series in the visualization. Grafana highlights the series that you are hovering over in bold in the series list in the tooltip.
- **Hidden -** Do not display the tooltip when you interact with the visualization.

Use an override to hide individual series from the tooltip.

Values sort order

When you set the **Tooltip mode** to **All**, the **Values sort order** option is displayed. This option controls the order in which values are listed in a tooltip. Choose from the following:

- **None** - Grafana automatically sorts the values displayed in a tooltip.
- **Ascending** - Values in the tooltip are listed from smallest to largest.
- **Descending** - Values in the tooltip are listed from largest to smallest.

Hover proximity

Set the hover proximity (in pixels) to control how close the cursor must be to a data point to trigger the tooltip to display.

Data links

Data links allow you to link to other panels, dashboards, and external resources while maintaining the context of the source panel. You can create links that include the series name or even the value under the cursor.

For each data link, set the following options:

- **Title**
- **URL**
- **Open in new tab**

To learn more, refer to Configure data links.

Thresholds

A threshold is a value or limit you set for a metric that's reflected visually when it's met or exceeded. Thresholds are one way you can conditionally style and color your visualizations based on query results.

For each threshold, set the following options:

Option	Description
Value	Set the value for each threshold.
Thresholds mode	Choose from **Absolute** and **Percentage**.
Show thresholds	Choose from a variety of display options including not displaying thresholds at all.

To learn more, refer to Configure thresholds.

Value mappings

Value mapping is a technique you can use to change how data appears in a visualization.

For each value mapping, set the following options:

- **Condition** - Choose what's mapped to the display text and (optionally) color:
  - **Value** - Specific values
  - **Range** - Numerical ranges
  - **Regex** - Regular expressions
  - **Special** - Special values like `Null`, `NaN` (not a number), or boolean values like `true` and `false`
- **Display text**
- **Color** (Optional)
- **Icon** (Canvas only)

To learn more, refer to Configure value mappings.

Field overrides

Overrides allow you to customize visualization settings for specific fields or series. When you add an override rule, it targets a particular set of fields and lets you define multiple options for how that field is displayed.

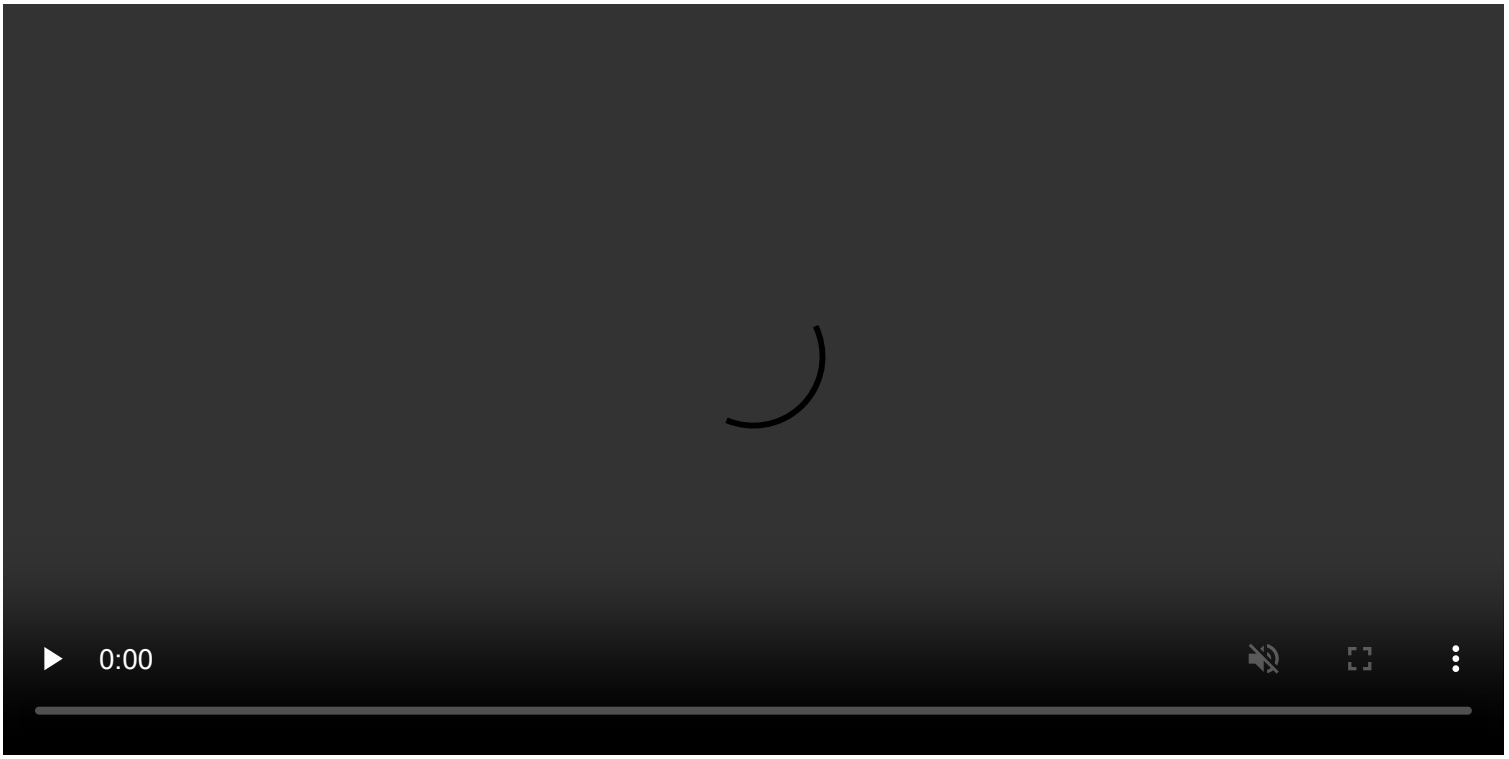
Choose from the following override options:

Option	Description
Fields with name	Select a field from the list of all available fields.
Field with name matching regex	Specify fields to override with a regular expression.
Fields with type	Select fields by type, such as string, numeric, or time.
Fields returned by query	Select all fields returned by a specific query, such as A, B, or C.
Fields with values	Select all fields returned by your defined reducer condition, such as **Min**, **Max**, **Count**, **Total**.

To learn more, refer to Configure field overrides.

Canvas

Canvases combine the power of Grafana with the flexibility of custom elements. They are extensible visualizations that allow you to add and arrange elements wherever you want within unstructured static and dynamic layouts. This lets you design custom visualizations and overlay data in ways that aren't possible with standard Grafana visualizations, all within the Grafana UI.



If you've used popular UI and web design tools, then designing canvases will feel very familiar. With all of these dynamic elements, there's almost no limit to what a canvas can display.

NOTE

We'd love your feedback on the canvas visualization. Please check out the open Github issues and submit a new feature request as needed.

Supported data formats

The canvas visualization is unique in that it doesn't have any specific data requirements. You can even start adding and configuring visual elements without providing any data. However, any data you plan to consume should be accessible through supported Grafana data sources and structured in a way that ensures smooth integration with your custom elements.

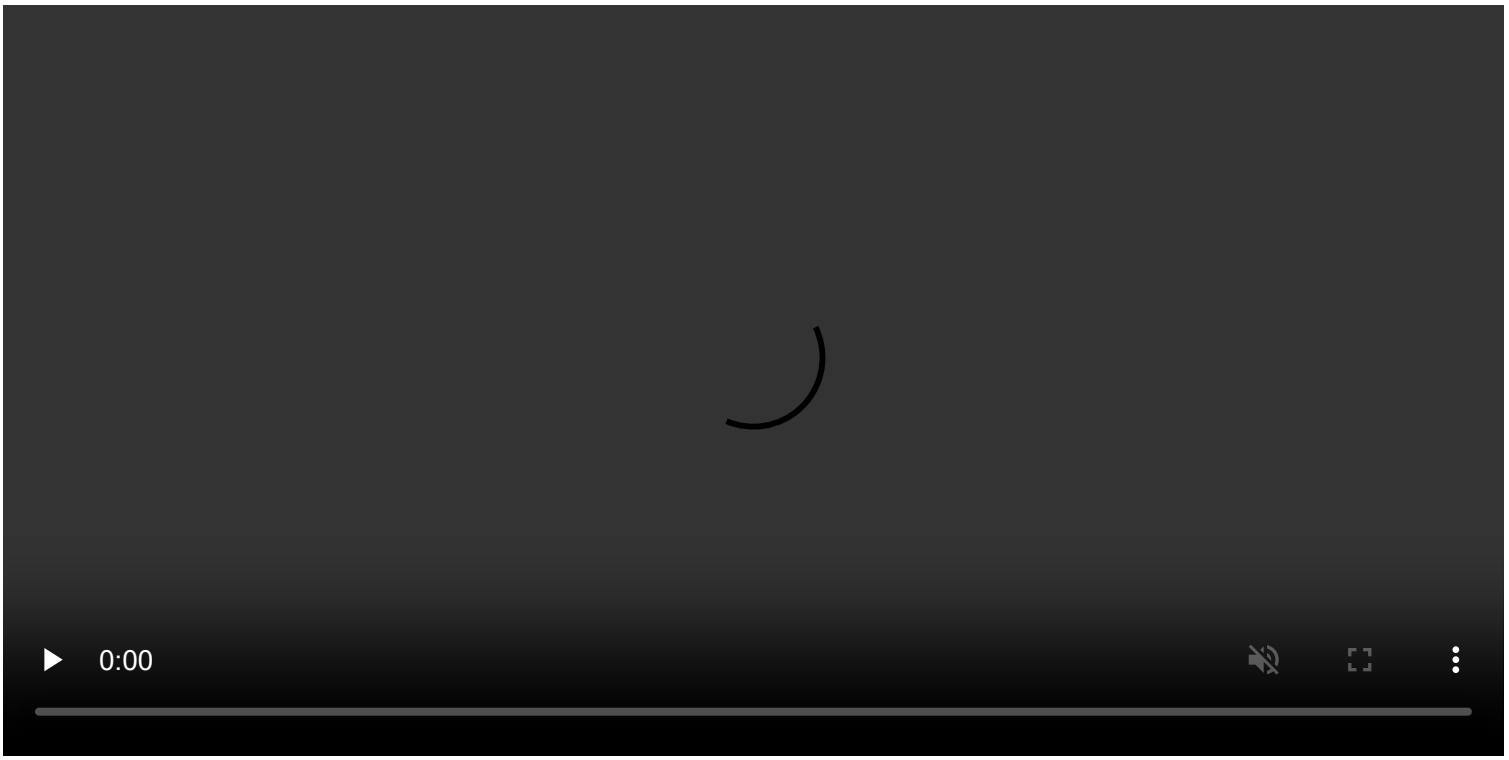
If your canvas is going to update in real time, your data should support refreshes at your desired intervals without degrading the user experience.

You can tie Elements and Connections to data through options like text, colors, and background pattern images, etc. available in the canvas visualization.

Elements

Metric value

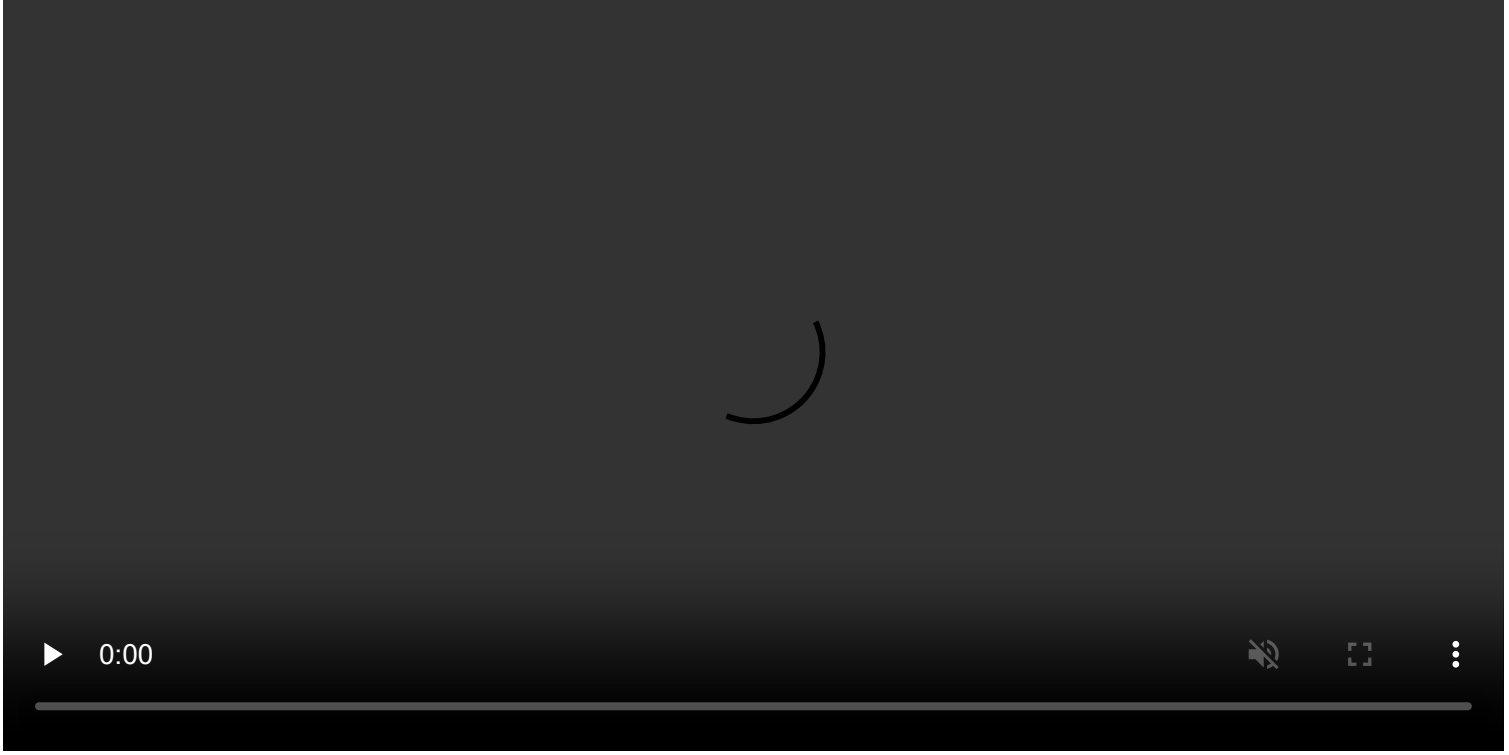
The metric value element lets you easily select the data you want to display on a canvas. This element has a unique "edit" mode that can be triggered either through the context menu "Edit" option or by double clicking. When in edit mode you can select which field data that you want to display.



Metric value element demo

Text

The text element lets you easily add text to the canvas. The element also supports an editing mode, triggered via either double clicking or the edit menu option in the context menu.



Text element demo

Ellipse

The ellipse element lets you add a basic ellipse to the canvas. An ellipse element can display text (both fixed and field data) and its background color can be changed based on data thresholds.

Rectangle

The rectangle element lets you add a basic rectangle to the canvas. A rectangle element can display text (both fixed and field data) and its background color can be changed based on data thresholds.

Icon

The icon element lets you add a supported icon to the canvas. Icons can have their color set based on thresholds / value mappings.

Server

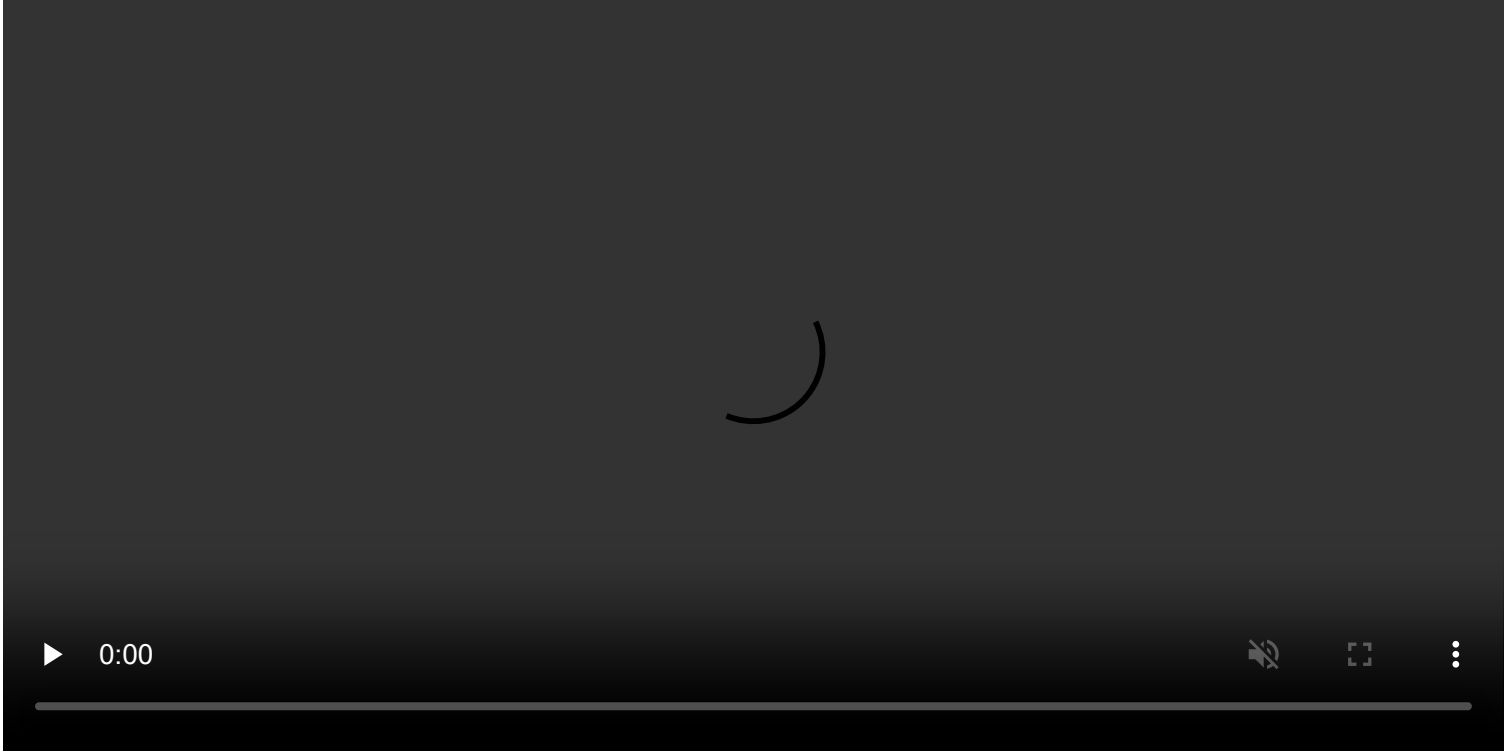
The server element lets you easily represent a single server, a stack of servers, a database, or a terminal. Server elements support status color, bulb color, and a bulb blink rate all configurable by fixed or field values.

Canvas server element

Button

The button element lets you add a basic button to the canvas. Button elements support triggering basic, unauthenticated API calls. API settings are found in the button element editor. You can also pass template variables in the API editor.

> **NOTE**
>
> A button click will only trigger an API call when inline editing is disabled.



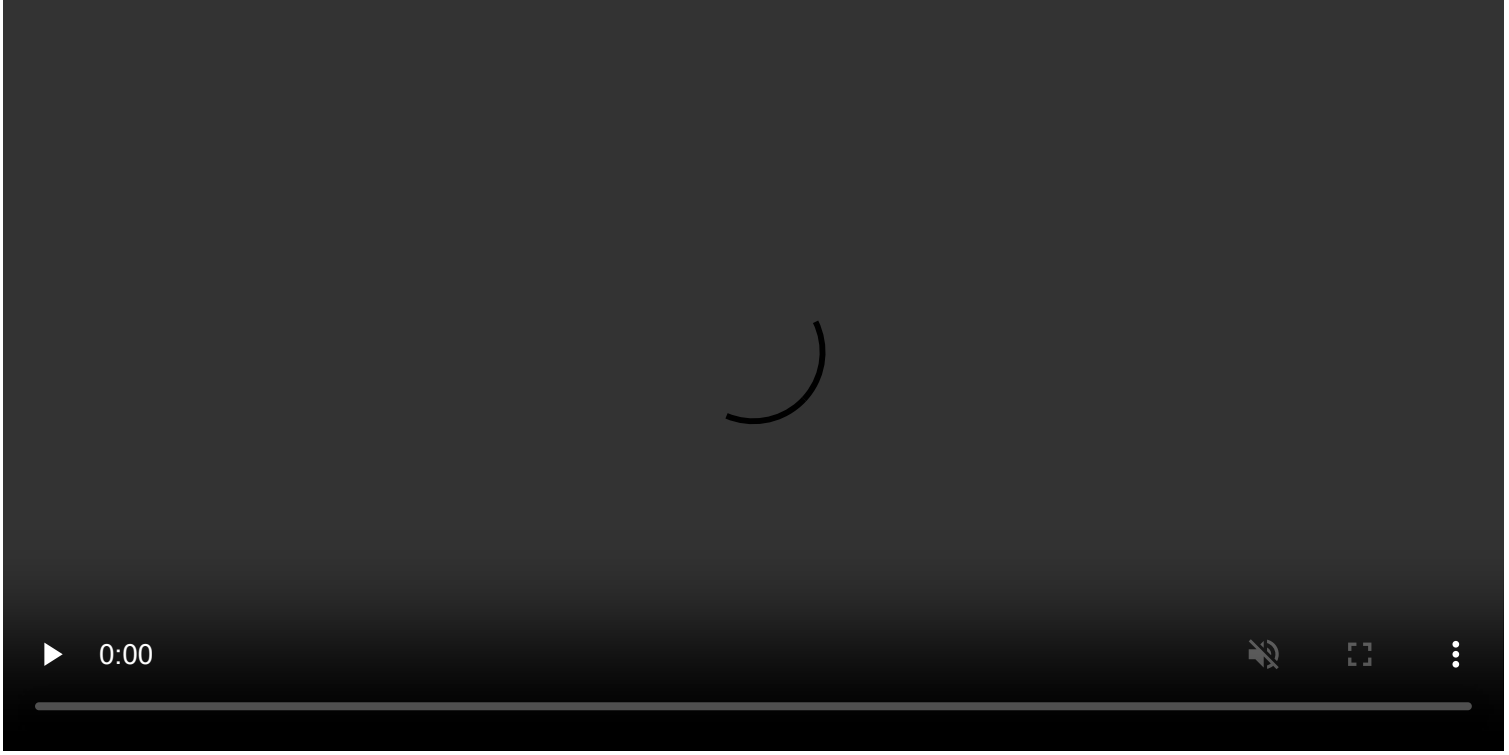
Canvas button element demo

Give it a try using Grafana Play

With Grafana Play, you can explore and see how it works, learning from practical examples to accelerate your development. This feature can be seen on Canvas Visualization: Buttons.

Connections

When building a canvas, you can connect elements together to create more complex visualizations. Connections are created by dragging from the connection anchor of one element to the connection anchor of another element. You can also create connections to the background of the canvas. Connection anchors are displayed when you hover over an element and inline editing is turned on. To remove a connection, simply click on the connection directly and then press the “Delete” or “Backspace” key.



Canvas connections demo

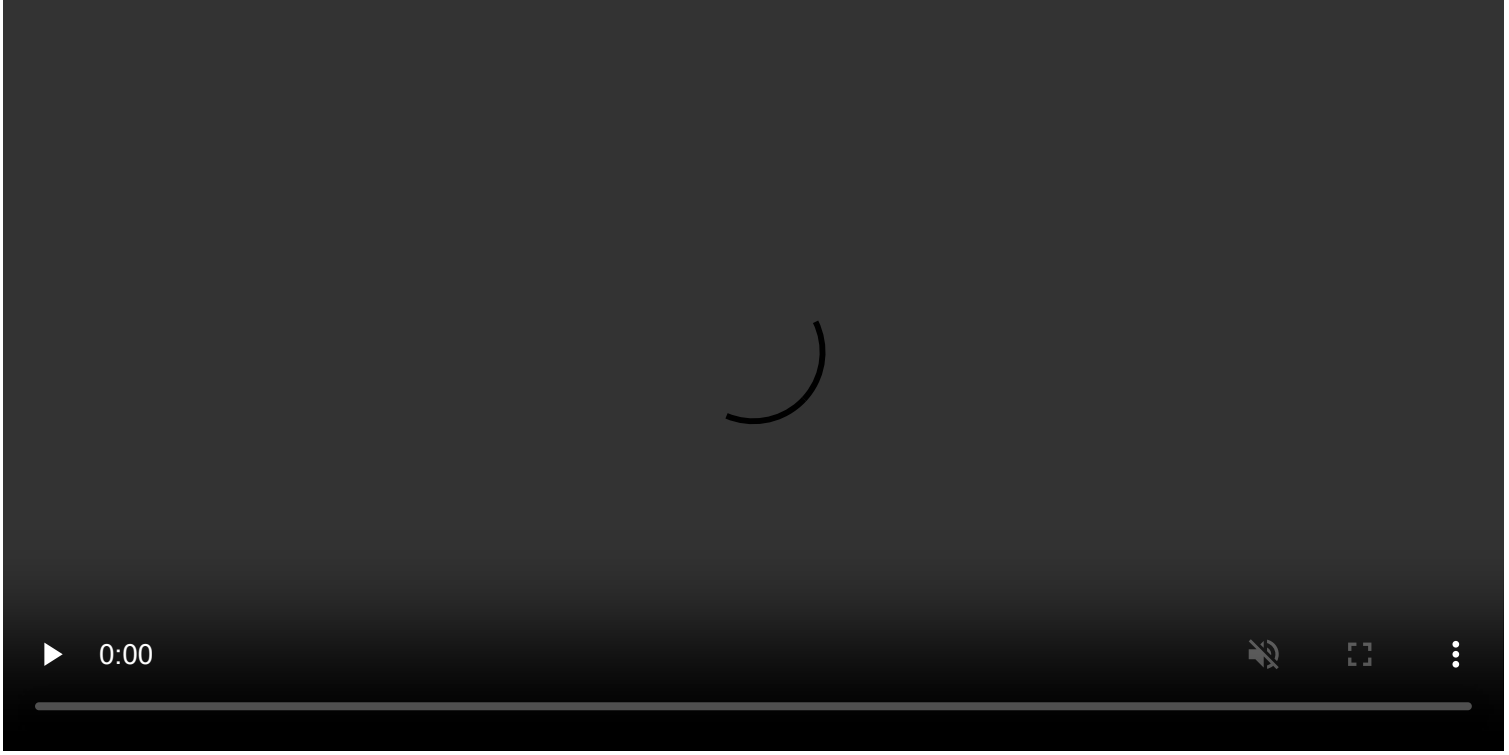
You can set both the size and color of connections based on fixed or field values. To do so, enter into panel edit mode, select the connection, and modify the connection's properties in the panel editor.

Canvas service graph

Canvas editing

Inline editor

You can edit your canvas inline while in the context of dashboard mode.

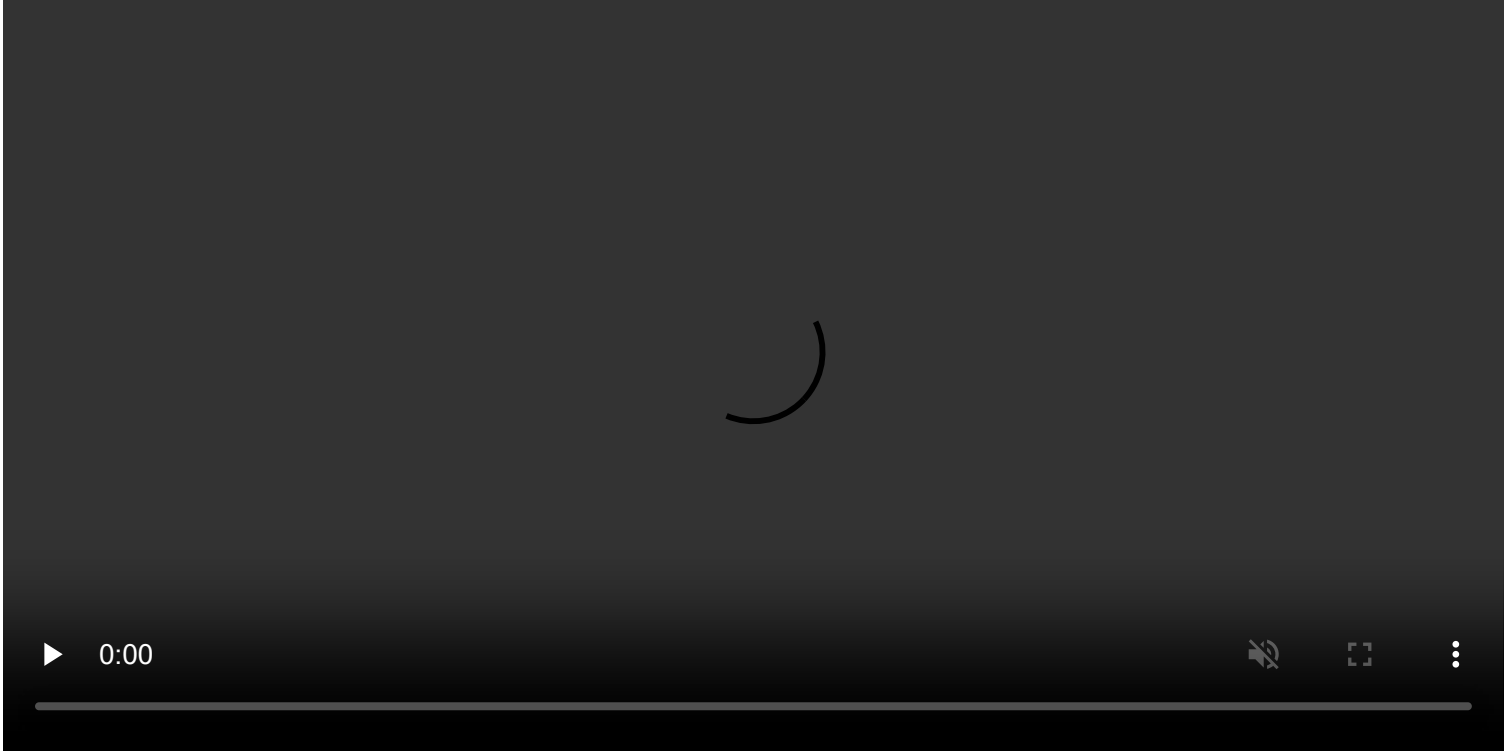


Inline editor demo

Pan and zoom

You can enable panning and zooming in a canvas. This allows you to both create and navigate more complex designs.

> **NOTE**
>
> Canvas pan and zoom is currently in public preview. Grafana Labs offers limited support, and breaking changes might occur prior to the feature being made generally available. Enable the `canvasPanelPanZoom` feature toggle in Grafana to use this feature. Contact Grafana Support to enable this feature in Grafana Cloud.



Canvas pan and zoom enablement video

Infinite panning

You can enable infinite panning in a canvas when pan and zoom is enabled. This allows you to pan and zoom the canvas and uncover larger designs.

> **NOTE**
>
> Infinite panning is an experimental feature that may not work as expected in all scenarios. For example, elements that are not top-left constrained may experience unexpected movement when panning.

Context menu

The context menu lets you perform common tasks quickly and efficiently. Supported functionality includes opening / closing the inline editor, duplicating an element, deleting an element, and more.

The context menu is triggered by a right click action over the panel / over a given canvas element. When right clicking the panel, you are able to set a background image and easily add elements to the canvas.

Canvas panel context menu

When right clicking an element, you are able to edit, delete, duplicate, and modify the element’s layer positioning.

Canvas element context menu

Element snapping and alignment

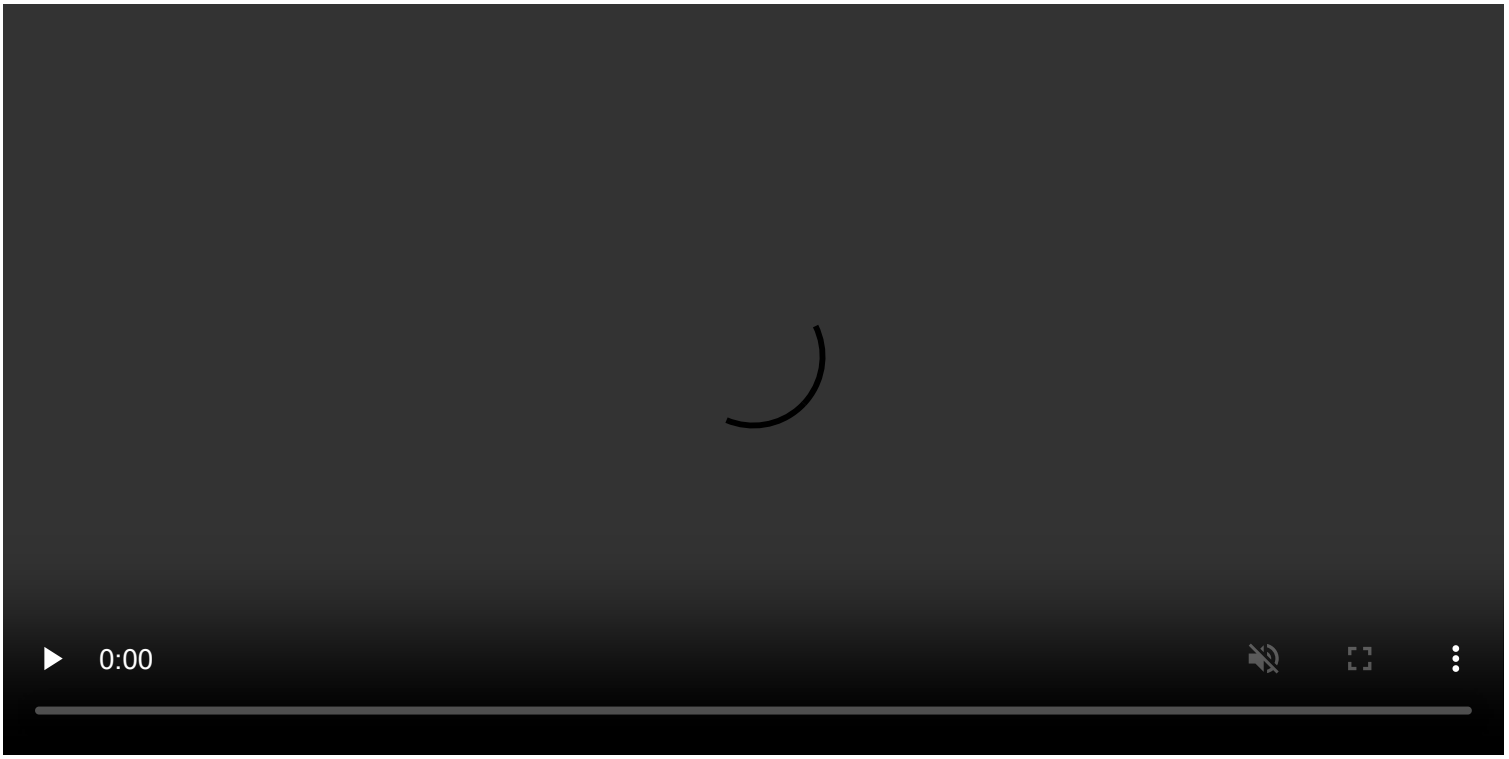
When you're moving elements around the canvas, snapping and alignment guides help you create more precise layouts.

> **NOTE**
>
> Currently, element snapping and alignment only works when the canvas is not zoomed in.

Canvas options

Inline editing

The inline editing toggle lets you lock or unlock the canvas. When turned off the canvas becomes "locked", freezing elements in place and preventing unintended modifications.



Inline editing toggle demo

Data links

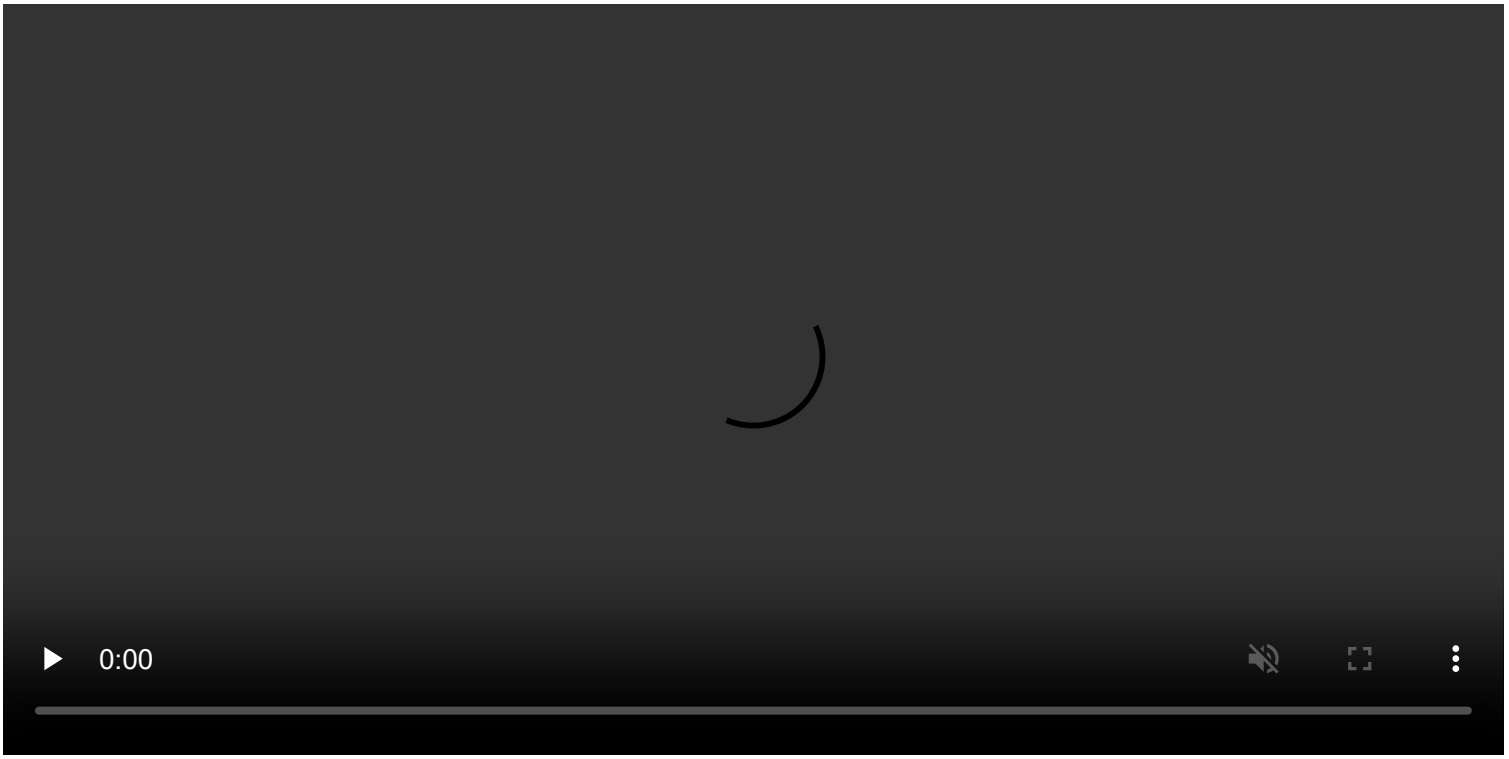
Canvases support data links for all elements except drone and button elements. You can add a data link by following these steps:

1. Set an element to be tied to a field value.
2. Turn off the inline editing toggle.
3. Create an override for **Fields with name** and select the element field name from the list.
4. Click the **+ Add override property** button.
5. Select `Datalinks > Datalinks` from the list.

6. Click **+Add link** add a title and URL for the data link.
7. Hover over the element to display the data link tooltip.
8. Click on the element to be able to open the data link.

If multiple elements use the same field name, and you want to control which elements display the data link, you can create a unique field name using the add field from calculation transform. The alias you create in the transformation will appear as a field you can use with an element.

1. In the panel editor for the canvas, click the **Transform** tab.
2. Select **Add field from calculation** from the list of transformations, or click **+ Add transformation** to display the list first.
3. Choose **Reduce row** from the dropdown and click the field name that you want to use for the element.
4. Select **All Values** from the **Calculation** dropdown.
5. Add an alias for the field name.
6. Reference the new unique field alias to create the element and field override.



Data links demo

Panel options

In the **Panel options** section of the panel editor pane, set basic options like panel title and description, as well as panel links. To learn more, refer to Configure panel options.

Standard options

Standard options in the panel editor pane let you change how field data is displayed in your visualizations. When you set a standard option, the change is applied to all fields or series. For more granular control over the display of fields, refer to Configure overrides.

Option	Description
Unit	Choose which unit a field should use.
Min/Max	Set the minimum and maximum values used in percentage threshold calculations or leave these field empty for them to be calculated automatically.
Field min/max	Enable **Field min/max** to have Grafana calculate the min or max of each field individually, based on the minimum or maximum value of the field.
Decimals	Specify the number of decimals Grafana includes in the rendered value.
Display name	Set the display title of all fields. You can use variables in the field title.
Color scheme	Set single or multiple colors for your entire visualization.
No value	Enter what Grafana should display if the field value is empty or null. The default value is a hyphen (-).

To learn more, refer to Configure standard options.

Thresholds

A threshold is a value or limit you set for a metric that's reflected visually when it's met or exceeded. Thresholds are one way you can conditionally style and color your visualizations based on query results.

Set the following options:

- **Value** - Set the value for each threshold.
- **Thresholds mode** - Choose from:
  - **Absolute**
  - **Percentage**

To learn more, refer to Configure thresholds.

Value mappings

Value mapping is a technique you can use to change how data appears in a visualization.

For each value mapping, set the following options:

- **Condition** - Choose what's mapped to the display text and (optionally) color:
  - **Value** - Specific values

    - **Range** - Numerical ranges
    - **Regex** - Regular expressions
    - **Special** - Special values like `Null`, `NaN` (not a number), or boolean values like `true` and `false`
- **Display text**
- **Color** (Optional)
- **Icon** (Canvas only)

To learn more, refer to Configure value mappings.

Field overrides

Overrides allow you to customize visualization settings for specific fields or series. When you add an override rule, it targets a particular set of fields and lets you define multiple options for how that field is displayed.

Choose from the following override options:

Option	Description
Fields with name	Select a field from the list of all available fields.
Field with name matching regex	Specify fields to override with a regular expression.
Fields with type	Select fields by type, such as string, numeric, or time.
Fields returned by query	Select all fields returned by a specific query, such as A, B, or C.
Fields with values	Select all fields returned by your defined reducer condition, such as **Min**, **Max**, **Count**, **Total**.

To learn more, refer to Configure field overrides.

Dashboard list

Dashboard lists allow you to display dynamic links to other dashboards. You can configure the list to use starred dashboards, recently viewed dashboards, a search query, and dashboard tags.

On each dashboard load, this panel queries the dashboard list, always providing the most up-to-date results.

You can use a dashboard list visualization to display a list of important dashboards that you want to track.

Configure a dashboard list visualization

Once you've created a dashboard, the following video shows you how to configure a dashboard list visualization:

Give it a try using Grafana Play

With Grafana Play, you can explore and see how it works, learning from practical examples to accelerate your development. This feature can be seen on Dashboard List Visualization.

Panel options

In the **Panel options** section of the panel editor pane, set basic options like panel title and description, as well as panel links. To learn more, refer to Configure panel options.

Dashboard list options

Use the following options to refine your dashboard list visualization.

Include current time range

Select this option to propagate the time range of the current dashboard to the dashboard links. When you click a link, the linked dashboard opens with the indicated time range already set.

Include current template variable values

Select this option to include template variables that are being used as query parameters in a link. When you click the link, any matching templates in the linked dashboard are set to the values from the link. Learn more in Dashboard URL variables.

Starred

Display starred dashboards in alphabetical order.

Recently viewed

Display recently viewed dashboards in alphabetical order.

Search

Display dashboards by search query or tags. You must enter at least one value in **Query** or **Tags**. For the **Query** and **Tags** fields, variable interpolation is supported. For example, `$my_var` or `${my_var}`. Learn more in Search option.

Show headings

The selected list section is shown as a heading:

- **Starred**
- **Recently viewed**
- **Search**

Max items

Sets the maximum number of items to list per section. For example, if you leave this at the default value of 10 and select **Starred** and **Recently viewed** dashboards, then the panel displays up to 20 total dashboards, 10 in each section.

Search options

These options only apply if you select the **Search** option.

Query

Use this field to search by dashboard name. Query terms are case-insensitive and partial values are accepted. For example, if you have dashboards called “Indoor Temps” and “Outdoor temp”, entering the word “temp” would return both results.

Folder

Select the dashboard folders that you want to display.

Tags

Enter tags by which you want to search. Note that tags don't appear as you type, and they're case sensitive. Tag search uses an `OR` condition, so if a dashboard has one of the defined tags, it's included in the list.

> **NOTE**
>
> When multiple tags and strings appear, the dashboard list displays those matching *all* conditions.

Datagrid

NOTE

The Grafana datagrid is experimental. This feature is supported by the engineering team on a best-effort basis, and breaking changes may occur without notice prior to general availability.

Datagrids offer you the ability to create, edit, and fine-tune data within Grafana. As such, this panel can act as a data source for other panels inside a dashboard.

Through it, you can manipulate data queried from any data source, you can start from a blank slate, or you can pull data from a dragged and dropped file. You can then use the panel as a simple tabular visualization, or you can modify the data—and even remove it altogether—to create a blank slate.

Editing the dataset changes the data source to use the inbuilt `-- Grafana --` data source, thus replacing the old data source settings and related queries, while also copying the current dataset into the dashboard model.

You can then use the panel as a data source for other panels, by using the inbuilt `-- Dashboard --` data source to pull the datagrid data. This allows for an interactive dashboard experience, where you can modify the data and see the changes reflected in other panels.

Learn more about the inbuilt `-- Grafana --` and `-- Dashboard --` data sources in the special data sources documentation.

Context menu

To provide a more streamlined experience, the datagrid has a context menu that can be accessed by right-clicking on a cell, column header, or row selector. Depending on the state of your datagrid, the context menu offers different options including:

- Delete or clear rows and columns.
- Remove all existing data (rendering your datagrid blank).
- Trigger search functionality, which allows you to find keywords within the dataset.

Deleting a row or column will remove the data from the datagrid, while clearing a row or column will only remove the data from the cells, leaving the row or column intact.

Header menu

You can also access a header menu by clicking the dropdown icon next to the header title. From here, you can not only delete or clear a column, but also rename it, freeze it, or convert the field type of the column.

Selecting series

If there are multiple series, you can set the datagrid to display the preferred dataset using the **Select series** dropdown in the panel options.

Using datagrids

Datagrids offer various ways of interacting with your data. You can add, edit, move, clear, and remove rows and columns; use the inbuilt search functionality to find specific data; and convert field types or freeze horizontal scroll on a specific column.

Add data

You can add data to a datagrid by creating a new column or row.

To create a new column, take the following steps:

1. In an existing panel, click the **+** button in the table header after the last column.
2. When prompted, add a name for the new column.
3. Click anywhere outside the field or press the Enter key to save the column.

Now you can add data in each cell.

To add a new row, click a **+** button after the last row. The button is present in each cell after the last row, and clicking it triggers the creation of a new row while also activating the cell that you clicked.

Edit data

You can edit data by taking the following steps:

1. Double-click on the cell that needs to be modified. This will activate the cell and allow you to edit the data.

2. After editing the data, click anywhere outside the cell or press the Enter key to finalize the edit.

To easily clear a cell of data, you can click on a cell to focus it and then press the Delete key.

Move data

You can move columns and rows as needed.

To move a column, take the following steps:

1. Click and hold the header of the column that needs to be moved.
2. Drag the column to the desired location.
3. Release the mouse button to finalize the move.

To move a row, click and hold on the row selector from the number column situated on the far left side of the grid, and drag it to the desired location. Releasing the mouse button finalizes the move.

Select multiple cells

You can select multiple cells by clicking on a single cell and dragging the mouse across others. This selection can be used to copy the data from the selected cells or to delete them using the Delete key.

Delete/clear multiple rows or columns

To delete or clear multiple rows, take the following steps:

1. Hover over the number column (to the left of the first column in the grid) to display row checkbox.
2. Select the checkboxes for the rows you want to work with. To select multiple consecutive rows, press and hold the Shift key while clicking on the first and last row. To select non-consecutive rows, press and hold the Ctrl (or Cmd) key while clicking the desired rows.
3. Right-click to access the context menu.
4. Select **Delete rows** or **Clear rows**.

The same rules apply to columns by clicking the column headers.

To delete all rows, use the “select all” checkbox at the top left corner of the datagrid. This selects all rows and allows you to delete them using the context menu.

Panel options

In the **Panel options** section of the panel editor pane, set basic options like panel title and description, as well as panel links. To learn more, refer to Configure panel options.

Flame graph

Flame graphs let you visualize profiling data. Using this visualization, a profile can be represented as a flame graph, top table, or both.

For example, if you want to understand which parts of a program consume the most resources, such as CPU time, memory, or I/O operations, you can use a flame graph to visualize and analyze where potential performance issues are:

Panel options

In the **Panel options** section of the panel editor pane, set basic options like panel title and description, as well as panel links. To learn more, refer to Configure panel options.

Flame graph mode

You can use a flame graph visualization if you need to:

- Identify any performance hotspots to find where code optimizations may be needed.
- Diagnose the root cause of any performance degradation.
- Analyze the behavior of complex systems, including distributed systems or microservices architectures.

To learn more about how Grafana Pyroscope visualizes flame graphs, refer to Flame graphs: Visualizing performance data.

Configure a flame graph visualization

Once you've created a dashboard, the following video shows you how to configure a flame graph visualization:

Give it a try using Grafana Play

With Grafana Play, you can explore and see how it works, learning from practical examples to accelerate your development. This feature can be seen on Flame Graphs.

Supported data formats

To render a flame graph, you must format the data frame data using a *nested set model*.

A nested set model ensures each item of a flame graph is encoded by its nesting level as an integer value, its metadata, and by its order in the data frame. This means that the order of items is significant and needs to be correct. The ordering is a depth-first traversal of the items in the flame

graph which recreates the graph without needing variable-length values in the data frame like in a children's array.

Required fields:

Field name	Type	Description
level	number	The nesting level of the item. In other words how many items are between this item and the top item of the flame graph.
value	number	The absolute or cumulative value of the item. This translates to the width of the item in the graph.
label	string	Label to be shown for the particular item.
self	number	Self value, which is usually the cumulative value of the item minus the sum of cumulative values of its immediate children.

Example

The following table is an example of the type of data you need for a flame graph visualization and how it should be formatted:

level	value	self	label
0	16.5 Bil	16.5 K	total
1	4.10 Bil	4.10 k	test/pkg/agent.(*Target).start.func1
2	4.10 Bil	4.10 K	test/pkg/agent.(*Target).start.func1
3	3.67 Bil	3.67 K	test/pkg/distributor.(*Distributor).Push
4	1.13 Bil	1.13 K	compress/gzip.(*Writer).Write
5	1.06 Bil	1.06 K	compress/flat.(*compressor).write

Modes

Flame graph mode

A flame graph takes advantage of the hierarchical nature of profiling data. It condenses data into a format that allows you to easily see which code paths are consuming the most system resources, such as CPU time, allocated objects, or space when measuring memory. Each block in the flame graph represents a function call in a stack and its width represents its value.

Grayed-out sections are a set of functions that represent a relatively small value and they are collapsed together into one section for performance reasons.

You can hover over a specific function to view a tooltip that shows you additional data about that function, like the function's value, percentage of total value, and the number of samples with that function.

Drop-down actions

You can click a function to show a drop-down menu with additional actions:

- Focus block
- Copy function name
- Sandwich view

Focus block

When you click **Focus block**, the block, or function, is set to 100% of the flame graph's width and all its child functions are shown with their widths updated relative to the width of the parent function. This makes it easier to drill down into smaller parts of the flame graph.

Copy function name

When you click **Copy function name**, the full name of the function that the block represents is copied.

Sandwich view

The sandwich view allows you to show the context of the clicked function. It shows all the function's callers on the top and all the callees at the bottom. This shows the aggregated context of the function so if the function exists in multiple places in the flame graph, all the contexts are shown and aggregated in the sandwich view.

Status bar

The status bar shows metadata about the flame graph and currently applied modifications, like what part of the graph is in focus or what function is shown in sandwich view. Click the **X** in the status bar pill to remove that modification.

Top table mode

The top table shows the functions from the profile in table format. The table has three columns: symbols, self, and total. The table is sorted by self time by default, but can be reordered by total time or symbol name by clicking the column headers. Each row represents aggregated values for the given function if the function appears in multiple places in the profile.

There are also action buttons on the left-most side of each row. The first button searches for the function name while second button shows the sandwich view of the function.

Toolbar

Search

You can use the search field to find functions with a particular name. All the functions in the flame graph that match the search will remain colored while the rest of the functions are grayed-out.

Color schema picker

You can switch between coloring functions by their value or by their package name to visually tie functions from the same package together.

Text align

Align text either to the left or to the right to show more important parts of the function name when it does not fit into the block.

Visualization picker

You can choose to show only the flame graph, only table, or both at the same time

Gauge

Gauges are single-value visualizations that allow you to quickly visualize where a value falls within a defined or calculated min and max range. With repeat options, you can display multiple gauges, each corresponding to a different series, column, or row.

You can use gauges if you need to track:

- Service level objectives (SLOs)
- How full a piece of equipment is
- How fast a vehicle is moving within a set of limits
- Network latency
- Equipment state with setpoint and alarm thresholds
- CPU consumption (0-100%)
- RAM availability

Configure a time series visualization

The following video provides beginner steps for creating gauge panels. You'll learn the data requirements and caveats, special customizations, and much more:

Give it a try using Grafana Play

With Grafana Play, you can explore and see how it works, learning from practical examples to accelerate your development. This feature can be seen on Grafana Gauge Visualization.

Supported data formats

To create a gauge visualization you need a dataset containing at least one numeric field. These values are identified by the field name. Additional text fields aren't required but can be used for identification and labeling.

Example - One value

GaugeName	GaugeValue
MyGauge	5

This dataset generates a visualization with one empty gauge showing the numeric value. This is because the gauge visualization automatically defines the upper and lower range from the minimum and maximum values in the dataset. This dataset has only one value, so it's set as both minimum and maximum.

If you only have one value, but you want to define a different minimum and maximum, you can set them manually in the Standard options settings to generate a more typical looking gauge.

Example - One row, multiple values

The gauge visualization can support multiple fields in a dataset.

Identifier	value1	value2	value3
Gauges	5	3	10

When there are multiple values in the dataset, the visualization displays multiple gauges and automatically defines the minimum and maximum. In this case, those are 3 and 10. Because the minimum and maximum values are defined, each gauge is shaded in to show that value in relation to the minimum and maximum.

Example - Multiple rows and values

The gauge visualization can display datasets with multiple rows of data or even multiple datasets.

Identifier	value1	value2	value3
Gauges	5	3	10
Indicators	6	9	15
Defaults	1	4	8

By default, the visualization is configured to calculate a single value per column or series and to display only the last row of data. However, it derives the minimum and maximum from the full dataset, even if those values aren't visible.

In this example, that means only the last row of data is displayed in the gauges and the minimum and maximum values are 1 and 10. The value 1 is displayed because it's in the last row, while 10 is not.

If you want to show one gauge per table cell, you can change the **Show** setting from **Calculate** to **All values**, and each gauge is labeled by concatenating the text column with each value's column name.

Example - Defined min and max

You can also define minimum and maximum values as part of the dataset.

Identifier	value	max	min
Gauges	5	10	2

If you don't want to display gauges for the `min` and `max` values, you can configure only one field to be displayed as described in the value options section.

Even when minimum and maximum values aren't displayed, the visualization still pulls the range from them.

Panel options

In the **Panel options** section of the panel editor pane, set basic options like panel title and description, as well as panel links. To learn more, refer to Configure panel options.

Value options

Use the following options to refine how your visualization displays the value:

Show

Choose how Grafana displays your data.

Calculate

Show a calculated value based on all rows.

- **Calculation -** Select a reducer function that Grafana will use to reduce many fields to a single value. For a list of available calculations, refer to Calculation types.

- **Fields -** Select the fields display in the panel.

All values

Show a separate stat for every row. If you select this option, then you can also limit the number of rows to display.

- **Limit -** The maximum number of rows to display. Default is 5,000.
- **Fields -** Select the fields display in the panel.

Gauge

Adjust how the gauge is displayed.

Orientation

Choose a stacking direction.

- **Auto -** Gauges display in rows and columns.
- **Horizontal -** Gauges display top to bottom.
- **Vertical -** Gauges display left to right.

Show threshold labels

Controls if threshold values are shown.

Show threshold markers

Controls if a threshold band is shown outside the inner gauge value band.

Gauge size

Choose a gauge size mode.

- **Auto -** Grafana determines the best gauge size.
- **Manual -** Manually configure the gauge size.

Min width

Set the minimum width of vertically-oriented gauges.

If you set a minimum width, the x-axis scrollbar is automatically displayed when there's a large amount of data.

> **NOTE**
>
> This option only applies when gauge size is set to manual.

Min height

Set the minimum height of horizontally-oriented gauges.

If you set a minimum height, the y-axis scrollbar is automatically displayed when there's a large amount of data.

> **NOTE**
>
> This option only applies when gauge size is set to manual.

Neutral

Set the starting value from which every gauge will be filled.

Text size

Adjust the sizes of the gauge text.

- **Title -** Enter a numeric value for the gauge title size.
- **Value -** Enter a numeric value for the gauge value size.

Standard options

Standard options in the panel editor pane let you change how field data is displayed in your visualizations. When you set a standard option, the change is applied to all fields or series. For more granular control over the display of fields, refer to Configure overrides.

Option	Description
Unit	Choose which unit a field should use.
Min/Max	Set the minimum and maximum values used in percentage threshold calculations or leave these field empty for them to be calculated automatically.
Field min/max	Enable **Field min/max** to have Grafana calculate the min or max of each field individually, based on the minimum or maximum value of the field.
Decimals	Specify the number of decimals Grafana includes in the rendered value.

Option	Description
Display name	Set the display title of all fields. You can use variables in the field title.
Color scheme	Set single or multiple colors for your entire visualization.
No value	Enter what Grafana should display if the field value is empty or null. The default value is a hyphen (-).

To learn more, refer to Configure standard options.

Data links

Data links allow you to link to other panels, dashboards, and external resources while maintaining the context of the source panel. You can create links that include the series name or even the value under the cursor.

For each data link, set the following options:

- **Title**
- **URL**
- **Open in new tab**

To learn more, refer to Configure data links.

Value mappings

Value mapping is a technique you can use to change how data appears in a visualization.

For each value mapping, set the following options:

- **Condition** - Choose what's mapped to the display text and (optionally) color:
  - **Value** - Specific values
  - **Range** - Numerical ranges
  - **Regex** - Regular expressions
  - **Special** - Special values like `Null`, `NaN` (not a number), or boolean values like `true` and `false`
- **Display text**
- **Color** (Optional)
- **Icon** (Canvas only)

To learn more, refer to Configure value mappings.

Thresholds

A threshold is a value or limit you set for a metric that's reflected visually when it's met or exceeded. Thresholds are one way you can conditionally style and color your visualizations based on query results.

Set the following options:

- **Value** - Set the value for each threshold.
- **Thresholds mode** - Choose from:
  - **Absolute**
  - **Percentage**

To learn more, refer to Configure thresholds.

Last, gauge colors and thresholds (the outer bar markers) of the gauge can be configured as described above.

Field overrides

Overrides allow you to customize visualization settings for specific fields or series. When you add an override rule, it targets a particular set of fields and lets you define multiple options for how that field is displayed.

Choose from the following override options:

Option	Description
Fields with name	Select a field from the list of all available fields.
Field with name matching regex	Specify fields to override with a regular expression.
Fields with type	Select fields by type, such as string, numeric, or time.
Fields returned by query	Select all fields returned by a specific query, such as A, B, or C.
Fields with values	Select all fields returned by your defined reducer condition, such as **Min**, **Max**, **Count**, **Total**.

To learn more, refer to Configure field overrides.

Geomap

Geomaps allow you to view and customize the world map using geospatial data. It's the ideal visualization if you have data that includes location information and you want to see it displayed in a map.

You can configure and overlay map layers, like heatmaps and networks, and blend included basemaps or your own custom maps. This helps you to easily focus on the important location-based characteristics of the data.

When a geomap is in focus, in addition to typical mouse controls, you can pan around using the arrow keys or zoom in and out using the plus (`+`) and minus (`-`) keys or icons.

Geomaps are also useful when you have location data that's changing in real time and you want to visualize where an element is moving, using auto-refresh.

You can use a geomap visualization if you need to:

- Track your fleet of vehicles and associated metrics
- Show the locations and statuses of data centers or other connected assets in a network
- Display geographic trends in a heatmap

- Visualize the relationship of your locations' HVAC consumption or solar production with the sun's location

> **NOTE**
>
> We'd love your feedback on the geomap visualization. Please check out the open Github issues and submit a new feature request as needed.

Configure a geomap visualization

The following video provides beginner steps for creating geomap visualizations. You'll learn the data requirements and caveats, special customizations, preconfigured displays and much more:

> **Give it a try using Grafana Play**
>
> With Grafana Play, you can explore and see how it works, learning from practical examples to accelerate your development. This feature can be seen on Geomap Examples.

Supported data formats

To create a geomap visualization, you need datasets containing fields with location information.

The supported location formats are:

- Latitude and longitude

- Geohash
- Lookup codes: country, US states, or airports

To learn more, refer to Location mode.

Geomaps also support additional fields with various data types to define things like labels, numbers, heat sizes, and colors.

Example - Latitude and longitude

If you plan to use latitude and longitude coordinates, the dataset must include at least two fields (or columns): one called `latitude` (you can also use `lat`), and one called `longitude` (also `lon` or `lng`). When you use this naming convention, the visualization automatically detects the fields and displays the elements. The order of the fields doesn't matter as long as there is one latitude and one longitude.

Name	latitude	longitude	value
Disneyland	33.8121	-117.9190	4
DisneyWorld	28.3772	-81.5707	10
EuroDisney	48.867374	2.784018	3
Tokyo Disney	35.6329	139.8804	70
Shanghai Disney	31.1414	121.6682	1

If your latitude and longitude fields are named differently, you can specify them, as indicated in the Location mode section.

Example - Geohash

If your location data is in geohash format, the visualization requires at least one field (or column) containing location data.

If the field is named `geohash`, the visualization automatically detects the location and displays the elements. The order of the fields doesn't matter and the data set can have multiple other numeric, text, and time fields.

Name	geohash	trips
Cancun	d5f21	8
Honolulu	87z9ps	0
Palm Cove	rhzxudynb014	1

Name	geohash	trips
Mykonos	swdj02ey9gyx	3

If your field containing geohash location data is not named as above, you can configure the visualization to use geohash and specify which field to use, as explained in the Location mode section.

Example - Lookup codes

The geomap visualization can identify locations based on country, airport, or US state codes.

For this configuration, the dataset must contain at least one field (or column) containing the location code.

If the field is named `lookup`, the visualization automatically detects it and displays points based on country codes.

Year	lookup	gdp
2016	MEX	104171935
2016	DEU	94393454
2016	FRA	83654250
2016	BRA	80921527
2016	CAN	79699762

The other location types— airport codes or US state codes—aren't automatically detected.

If you want to use other codes or give the field a custom name, you can follow the steps in the Location mode section.

Panel options

In the **Panel options** section of the panel editor pane, set basic options like panel title and description, as well as panel links. To learn more, refer to Configure panel options.

Map View

The map view controls the initial view of the map when the dashboard loads.

Initial View

The initial view configures how the geomap renders when the panel is first loaded.

- **View** sets the center for the map when the panel first loads.

- **Fit to data** fits the map view based on the data extents of Map layers and updates when data changes.
  - **Data** option allows selection of extent based on data from “All layers”, a single “Layer”, or the “Last value” from a selected layer.
  - **Layer** can be selected if fitting data from a single “Layer” or the “Last value” of a layer.
  - **Padding** sets padding in relative percent beyond data extent (not available when looking at “Last value” only).
  - **Max Zoom** sets the maximum zoom level when fitting data.
- **Coordinates** sets the map view based on:
  - **Latitude**
  - **Longitude**
- Default Views are also available including:
  - **(0°, 0°)**
  - **North America**
  - **South America**
  - **Europe**
  - **Africa**
  - **West Asia**
  - **South Asia**
  - **South-East Asia**
  - **East Asia**
  - **Australia**
  - **Oceania**
- **Zoom** sets the initial zoom level.

Share view

The **Share view** option allows you to link the movement and zoom actions of multiple map visualizations within the same dashboard. The map visualizations that have this option enabled act in tandem when one of them is moved or zoomed, leaving the other ones independent.

> **NOTE**
>
> You might need to reload the dashboard for this feature to work.

Map layers

Geomaps support showing multiple layers. Each layer determines how you visualize geospatial data on top of the base map.

Types

There are seven map layer types to choose from in a geomap.

- Markers renders a marker at each data point.
- Heatmap visualizes a heatmap of the data.
- GeoJSON renders static data from a GeoJSON file.
- Night / Day renders a night / day region.
- Route (Beta) render data points as a route.
- Photos (Beta) renders a photo at each data point.
- Network (Beta) visualizes a network graph from the data.

> **NOTE**
>
> Beta is equivalent to the public preview release stage.

There are also two experimental (or alpha) layer types.

- **Icon at last point (alpha)** renders an icon at the last data point.
- **Dynamic GeoJSON (alpha)** styles a GeoJSON file based on query results.

> **NOTE**
>
> To enable experimental layers: Set `enable_alpha` to `true` in your configuration file:

```
[panels]
enable_alpha = true
```

Copy

> To enable the experimental layers using Docker, run the following command:

```
docker run -p 3000:3000 -e "GF_PANELS_ENABLE_ALPHA=true" grafana/grafana:<VERSI
```

Copy

> **NOTE**
>
> Basemap layer types can also be added as layers. You can specify an opacity.

Layer Controls

The layer controls allow you to create layers, change their name, reorder and delete layers.

- **Add layer** creates an additional, configurable data layer for the geomap. When you add a layer, you are prompted to select a layer type. You can change the layer type at any point during panel configuration. See the **Layer Types** section above for details on each layer type.
- The layer controls allow you to rename, delete, and reorder the layers of the visualization.
  - **Edit layer name (pencil icon)** renames the layer.
  - **Trash Bin** deletes the layer.
  - **Reorder (six dots/grab handle)** allows you to change the layer order. Data on higher layers will appear above data on lower layers. The visualization will update the layer order as you drag and drop to help simplify choosing a layer order.

You can add multiple layers of data to a single geomap in order to create rich, detailed visualizations.

Data

Geomaps need a source of geographical data gathered from a data source query which can return multiple datasets. By default Grafana picks the first dataset, but this drop-down allows you to pick other datasets if the query returns more than one.

Location mode

There are four options to map the data returned by the selected query:

- **Auto** automatically searches for location data. Use this option when your query is based on one of the following names for data fields.
  - geohash: “geohash”
  - latitude: “latitude”, “lat”
  - longitude: “longitude”, “lng”, “lon”
  - lookup: “lookup”
- **Coords** specifies that your query holds coordinate data. You will get prompted to select numeric data fields for latitude and longitude from your database query.
- **Geohash** specifies that your query holds geohash data. You will be prompted to select a string data field for the geohash from your database query.
- **Lookup** specifies that your query holds location name data that needs to be mapped to a value. You will be prompted to select the lookup field from your database query and a gazetteer. The gazetteer is the directory that is used to map your queried data to a geographical point.

Basemap layer

A basemap layer provides the visual foundation for a mapping application. It typically contains data with global coverage. Several base layer options are available each with specific configuration

options to style the base map.

Types

There are four basemap layer types to choose from in a geomap.

- Open Street Map adds a map from a collaborative free geographic world database.
- CARTO adds a layer from CARTO Raster basemaps.
- ArcGIS adds a layer from an ESRI ArcGIS MapServer.
- XYZ adds a map from a generic tile layer.

Default

The default base layer uses the CARTO map. You can define custom default base layers in the `.ini` configuration file.

Configure the default base layer with provisioning

You can configure the default base map using config files with Grafana's provisioning system. For more information on all the settings, refer to the provisioning docs page.

Use the JSON configuration option `default_baselayer_config` to define the default base map. There are currently four base map options to choose from: `carto`, `esri-xyz`, `osm-standard`, `xyz`. Here are some provisioning examples for each base map option.

- **carto** loads the CartoDB tile server. You can choose from `auto`, `dark`, and `light` theme for the base map and can be set as shown below. The `showLabels` tag determines whether or not Grafana shows the Country details on top of the map. Here is an example:

ini

Copy

```
geomap_default_baselayer = `{
  "type": "carto",
  "config": {
    "theme": "auto",
    "showLabels": true
```

```
  }
}`
```

- **esri-xyz** loads the ESRI tile server. There are already multiple server instances implemented to show the various map styles: `world-imagery`, `world-physical`, `topo`, `usa-topo`, and `ocean`. The `custom` server option allows you to configure your own ArcGIS map server. Here are some examples:

ini Copy

```
geomap_default_baselayer = `{
  "type": "esri-xyz",
  "config": {
    "server": "world-imagery"
  }
}`
```

ini Copy

```
geomap_default_baselayer = `{
  "type": "esri-xyz",
  "config": {
    "server": "custom",
    "url": "[tile server url]",
    "attribution": "[tile server attribution]"
  }
}`
```

- **osm-standard** loads the OpenStreetMap tile server. There are no additional configurations needed and the `config` fields can be left blank. Here is an example:

ini Copy

```
default_baselayer_config = `{
  "type": "osm-standard",
  "config": {}
}`
```

- **xyz** loads a custom tile server defined by the user. Set a valid tile server `url`, with {z}/{x}/{y} for this option in order to properly load a default base map. Here is an example:

ini Copy

```
default_baselayer_config = `{
  "type": "xyz",
  "config": {
    "attribution": "Open street map",
    "url": "https://tile.openstreetmap.org/{z}/{x}/{y}.png"
  }
}`
```

`enable_custom_baselayers` allows you to enable or disable custom open source base maps that are already implemented. The default is `true`.

Markers layer

The markers layer allows you to display data points as different marker shapes such as circles, squares, triangles, stars, and more.

- **Data** and **Location mode** configure the data settings for the layer. For more information, refer to Data and Location mode.
- **Size** configures the size of the markers. The default is `Fixed size`, which makes all marker sizes the same regardless of the data; however, there is also an option to size the markers based on data corresponding to a selected field. `Min` and `Max` marker sizes have to be set such that the markers can scale within this range.
- **Symbol** allows you to choose the symbol, icon, or graphic to aid in providing additional visual context to your data. Choose from assets that are included with Grafana such as simple symbols or the Unicon library. You can also specify a URL containing an image asset. The image must be a scalable vector graphic (SVG).
- **Symbol Vertical Align** configures the vertical alignment of the symbol relative to the data point. Note that the symbol's rotation angle is applied first around the data point, then the vertical alignment is applied relative to the rotation of the symbol.

- **Symbol Horizontal Align** configures the horizontal alignment of the symbol relative to the data point. Note that the symbol's rotation angle is applied first around the data point, then the horizontal alignment is applied relative to the rotation of the symbol.
- **Color** configures the color of the markers. The default `Fixed color` sets all markers to a specific color. There is also an option to have conditional colors depending on the selected field data point values and the color scheme set in the `Standard options` section.
- **Fill opacity** configures the transparency of each marker.
- **Rotation angle** configures the rotation angle of each marker. The default is `Fixed value`, which makes all markers rotate to the same angle regardless of the data; however, there is also an option to set the rotation of the markers based on data corresponding to a selected field.
- **Text label** configures a text label for each marker.
- **Show legend** allows you to toggle the legend for the layer.
- **Display tooltip** allows you to toggle tooltips for the layer.

Heatmap layer

The heatmap layer clusters various data points to visualize locations with different densities. To add a heatmap layer:

Click on the drop-down menu under Data Layer and choose `Heatmap`.

Similar to `Markers`, you are prompted with various options to determine which data points to visualize and how you want to visualize them.

- **Data** and **Location mode** configure the data settings for the layer. For more information, refer to Data and Location mode.
- **Weight values** configure the intensity of the heatmap clusters. `Fixed value` keeps a constant weight value throughout all data points. This value should be in the range of 0~1. Similar to Markers, there is an alternate option in the drop-down to automatically scale the weight values depending on data values.
- **Radius** configures the size of the heatmap clusters.
- **Blur** configures the amount of blur on each cluster.
- **Opacity** configures the opacity of each cluster.
- **Display tooltip** allows you to toggle tooltips for the layer.

GeoJSON layer

The GeoJSON layer allows you to select and load a static GeoJSON file from the filesystem.

- **GeoJSON URL** provides a choice of GeoJSON files that ship with Grafana.
- **Default Style** controls which styles to apply when no rules above match.
  - **Color** configures the color of the default style
  - **Opacity** configures the default opacity
- **Style Rules** apply styles based on feature properties
  - **Rule** allows you to select a *feature*, *condition*, and *value* from the GeoJSON file in order to define a rule. The trash bin icon can be used to delete the current rule.
  - **Color** configures the color of the style for the current rule
  - **Opacity** configures the transparency level for the current rule
- **Add style rule** creates additional style rules.
- **Display tooltip** allows you to toggle tooltips for the layer.

> **NOTE**
>
> Styles can be set within the "properties" object of the GeoJSON with support for the following geometries:
>
> - Polygon, MultiPolygon
>   - **"fill"** - The color of the interior of the polygon(s)
>   - **"fill-opacity"** - The opacity of the interior of the polygon(s)
>   - **"stroke-width"** - The width of the line component of the polygon(s)
> - Point, MultiPoint
>   - **"marker-color"** - The color of the point(s)
>   - **"marker-size"** - The size of the point(s)
> - LineString, MultiLineString
>   - **"stroke"** - The color of the line(s)
>   - **"stroke-width"** - The width of the line(s)

Night / Day layer

The Night / Day layer displays night and day regions based on the current time range.

Geomap panel Night / Day

Options

- **Data** configures the data set for the layer. For more information, refer to Data.
- **Show** toggles the time source from panel time range.
- **Night region color** picks the color for the night region.
- **Display sun** toggles the sun icon.
- **Opacity** set the opacity from `0` (transparent) to `1` (opaque).
- **Display tooltip** allows you to toggle tooltips for the layer.

Geomap panel Night / Day options

More information

- **Extensions for OpenLayers - DayNight**

Route layer (Beta)

> **CAUTION**
>
> The Route layer is currently in public preview. Grafana Labs offers limited support, and breaking changes might occur prior to the feature being made generally available.

The Route layer renders data points as a route.

Geomap panel Route

Options

- **Data** and **Location mode** configure the data settings for the layer. For more information, refer to Data and Location mode.
- **Size** sets the route thickness. Fixed value by default. When field data is selected you can set the Min and Max range in which field data can scale.
- **Color** sets the route color. Set to `Fixed color` by default. You can also tie the color to field data.
- **Fill opacity** configures the opacity of the route.
- **Text label** configures a text label for each route.

- **Arrow** sets the arrow styling to display along route, in order of data.
  - **None**
  - **Forward**
  - **Reverse**
- **Display tooltip** allows you to toggle tooltips for the layer.

Geomap panel Route arrows with size

More information

- **Extensions for OpenLayers - Flow Line Style**

Photos layer (Beta)

> **CAUTION**
>
> The Photos layer is currently in public preview. Grafana Labs offers limited support, and breaking changes might occur prior to the feature being made generally available.

The Photos layer renders a photo at each data point.

Geomap panel Photos

Options

- **Data** and **Location mode** configure the data settings for the layer. For more information, refer to Data and Location mode.
- **Image Source field** allows you to select a string field containing image data in either of the following formats:
  - **Image URLs**
  - **Base64 encoded** - Image binary (“...”)
- **Kind** sets the frame style around the images. Choose from:
  - **Square**
  - **Circle**
  - **Anchored**
  - **Folio**
- **Crop** toggles whether the images are cropped to fit.
- **Shadow** toggles a box shadow behind the images.
- **Border** sets the border size around images.
- **Border color** sets the border color around images.
- **Radius** sets the overall size of images in pixels.

- **Display tooltip** allows you to toggle tooltips for the layer.

Geomap panel Photos options

More information

- **Extensions for OpenLayers - Image Photo Style**

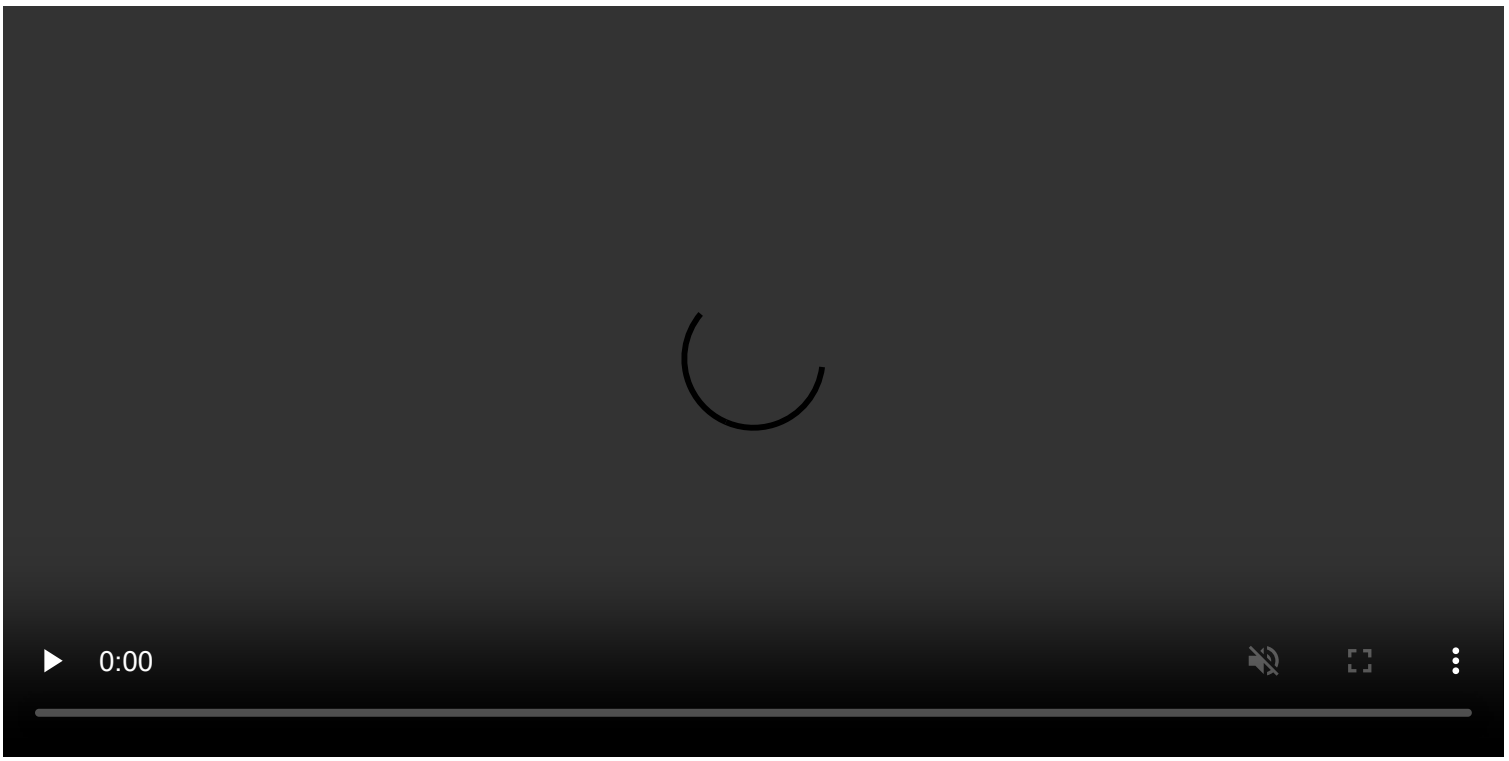
Network layer (Beta)

CAUTION

The Network layer is currently in public preview. Grafana Labs offers limited support, and breaking changes might occur prior to the feature being made generally available.

The Network layer renders a network graph. This layer supports the same data format supported by the node graph visualization with the addition of geospatial data included in the nodes data. The geospatial data is used to locate and render the nodes on the map.

Geomap network layer



Node graph to Geomap network layer

Options

- **Data** and **Location mode** configure the data settings for the layer. For more information, refer to Data and Location mode.
- **Arrow** sets the arrow direction to display for each edge, with forward meaning source to target. Choose from:
  - **None**
  - **Forward**
  - **Reverse**
  - **Both**
- **Show legend** allows you to toggle the legend for the layer. **Note:** The legend currently only supports node data.
- **Display tooltip** allows you to toggle tooltips for the layer.

Node styles

- **Size** configures the size of the nodes. The default is `Fixed size`, which makes all node sizes the same regardless of the data; however, there is also an option to size the nodes based on data corresponding to a selected field. `Min` and `Max` node sizes have to be set such that the nodes can scale within this range.
- **Symbol** allows you to choose the symbol, icon, or graphic to aid in providing additional visual context to your data. Choose from assets that are included with Grafana such as simple symbols or the Unicon library. You can also specify a URL containing an image asset. The image must be a scalable vector graphic (SVG).
- **Color** configures the color of the nodes. The default `Fixed color` sets all nodes to a specific color. There is also an option to have conditional colors depending on the selected field data point values and the color scheme set in the `Standard options` section.
- **Fill opacity** configures the transparency of each node.
- **Rotation angle** configures the rotation angle of each node. The default is `Fixed value`, which makes all nodes rotate to the same angle regardless of the data; however, there is also an option to set the rotation of the nodes based on data corresponding to a selected field.
- **Text label** configures a text label for each node.

Edge styles

- **Size** configures the line width of the edges. The default is `Fixed size`, which makes all edge line widths the same regardless of the data; however, there is also an option to size the edges based on data corresponding to a selected field. `Min` and `Max` eges sizes have to be set such that the edges can scale within this range.

- **Color** configures the color of the edges. The default `Fixed color` sets all edges to a specific color. There is also an option to have conditional colors depending on the selected field data point values and the color scheme set in the `Standard options` section.
- **Fill opacity** configures the transparency of each edge.
- **Text label** configures a text label for each edge.

CARTO layer

A CARTO layer is from CARTO Raster basemaps.

Options

- **Theme**
  - Auto
  - Light

Geomap panel CARTO light example

  - Dark

Geomap panel CARTO dark example

- **Show labels** shows the Country details on top of the map.
- **Opacity** from 0 (transparent) to 1 (opaque)

Geomap panel CARTO options

More Information

- **About CARTO**

XYZ tile layer

The XYZ tile layer is a map from a generic tile layer.

Geomap panel xyz example

Options

- **URL template**

  > **Note:** Set a valid tile server url, with {z}/{x}/{y} for example: https://tile.openstreetmap.org/{z}/{x}/{y}.png

- **Attribution** sets the reference string for the layer if displayed in map controls
- **Opacity** from 0 (transparent) to 1 (opaque)

Geomap panel xyz options

More information

- **Tiled Web Map Wikipedia**
- **List of Open Street Map Tile Servers**

Open Street Map layer

A map from a collaborative free geographic world database.

Geomap panel Open Street Map

Options

- **Opacity** from 0 (transparent) to 1 (opaque)

Geomap panel Open Street Map options

More Information

- About Open Street Map

ArcGIS layer

An ArcGIS layer is a layer from an ESRI ArcGIS MapServer.

Options

- **Server Instance** to select the map type.
  - World Street Map

Geomap panel ArcGIS World Street Map

  - World Imagery

Geomap panel ArcGIS World Imagery

- World Physical

Geomap panel ArcGIS World Physical

- Topographic

Geomap panel ArcGIS Topographic

- USA Topographic

Geomap panel ArcGIS USA Topographic

- World Ocean

Geomap panel ArcGIS World Ocean

- Custom MapServer (see XYZ for formatting)
  - URL template
  - Attribution

- **Opacity** from 0 (transparent) to 1 (opaque)

Geomap panel ArcGIS options

More Information

- **ArcGIS Services**
- **About ESRI**

Map Controls

The map controls section contains various options for map information and tool overlays.

Geomap panel map controls

Zoom

This section describes each of the zoom controls.

Show zoom control

Displays zoom controls in the upper left corner. This control can be useful when using systems that don’t have a mouse.

Geomap panel zoom

Mouse wheel zoom

Enables the mouse wheel to be used for zooming in or out.

Show attribution

Displays attribution for basemap layers.

Geomap panel attribution

Show scale

Displays scale information in the bottom left corner.

Geomap panel scale

NOTE

Currently only displays units in [m]/[km].

Show measure tools

Displays measure tools in the upper right corner. Measurements appear only when this control is open.

Geomap panel measure

- **Click** to start measuring
- **Continue clicking** to continue measurement
- **Double-click** to end measurement

> **NOTE**
>
> When you change measurement type or units, the previous measurement is removed from the map. If the control is closed and then re-opened, the most recent measurement is displayed. A measurement can be modified by clicking and dragging on it.

Length

Get the spherical length of a geometry. This length is the sum of the great circle distances between coordinates. For multi-part geometries, the length is the sum of the length of each part.

Geometries are assumed to be in ‘EPSG:3857’.

- **Metric (m/km)**
- **Feet (ft)**
- **Miles (mi)**
- **Nautical miles (nmi)**

Geomap panel measure length

Area

Get the spherical area of a geometry. This area is calculated assuming that polygon edges are segments of great circles on a sphere. Geometries are assumed to be in ‘EPSG:3857’.

- **Square Meters (m^2)**

- **Square Kilometers (km²)**
- **Square Feet (ft²)**
- **Square Miles (mi²)**
- **Acres (acre)**
- **Hectare (ha)**

Geomap panel measure area

Show debug

Displays debug information in the upper right corner. This can be useful for debugging or validating a data source.

- **Zoom** displays current zoom level of the map.
- **Center** displays the current **longitude**, **latitude** of the map center.

Geomap panel debug

Tooltip

- **None** displays tooltips only when a data point is clicked.
- **Details** displays tooltips when a mouse pointer hovers over a data point.

Standard options

Standard options in the panel editor pane let you change how field data is displayed in your visualizations. When you set a standard option, the change is applied to all fields or series. For more granular control over the display of fields, refer to Configure overrides.

Option	Description
Unit	Choose which unit a field should use.
Min/Max	Set the minimum and maximum values used in percentage threshold calculations or leave these field empty for them to be calculated automatically.
Field min/max	Enable **Field min/max** to have Grafana calculate the min or max of each field individually, based on the minimum or maximum value of the field.
Decimals	Specify the number of decimals Grafana includes in the rendered value.
Display name	Set the display title of all fields. You can use variables in the field title.
Color scheme	Set single or multiple colors for your entire visualization.
No value	Enter what Grafana should display if the field value is empty or null. The default value is a hyphen (-).

To learn more, refer to Configure standard options.

Data links

Data links allow you to link to other panels, dashboards, and external resources while maintaining the context of the source panel. You can create links that include the series name or even the value under the cursor.

For each data link, set the following options:

- **Title**
- **URL**
- **Open in new tab**

To learn more, refer to Configure data links.

Value mappings

Value mapping is a technique you can use to change how data appears in a visualization.

For each value mapping, set the following options:

- **Condition** - Choose what's mapped to the display text and (optionally) color:
  - **Value** - Specific values
  - **Range** - Numerical ranges
  - **Regex** - Regular expressions
  - **Special** - Special values like `Null`, `NaN` (not a number), or boolean values like `true` and `false`
- **Display text**
- **Color** (Optional)

- **Icon** (Canvas only)

To learn more, refer to Configure value mappings.

Thresholds

A threshold is a value or limit you set for a metric that's reflected visually when it's met or exceeded. Thresholds are one way you can conditionally style and color your visualizations based on query results.

Set the following options:

- **Value** - Set the value for each threshold.
- **Thresholds mode** - Choose from:
  - **Absolute**
  - **Percentage**

To learn more, refer to Configure thresholds.

Field overrides

Overrides allow you to customize visualization settings for specific fields or series. When you add an override rule, it targets a particular set of fields and lets you define multiple options for how that field is displayed.

Choose from the following override options:

Option	Description
Fields with name	Select a field from the list of all available fields.
Field with name matching regex	Specify fields to override with a regular expression.
Fields with type	Select fields by type, such as string, numeric, or time.
Fields returned by query	Select all fields returned by a specific query, such as A, B, or C.
Fields with values	Select all fields returned by your defined reducer condition, such as **Min**, **Max**, **Count**, **Total**.

To learn more, refer to Configure field overrides.

Heatmap

Heatmaps allow you to view histograms over time. While histograms display the data distribution that falls in a specific value range, heatmaps allow you to identify patterns in the histogram data distribution over time. For more information about heatmaps, refer to Introduction to histograms and heatmaps.

For example, if you want to understand the temperature changes for the past few years, you can use a heatmap visualization to identify trends in your data:

Give it a try using Grafana Play

With Grafana Play, you can explore and see how it works, learning from practical examples to accelerate your development. This feature can be seen on Grafana Heatmaps.

You can use a heatmap visualization if you need to:

- Visualize a large density of your data distribution.
- Condense large amounts of data through various color schemes that are easier to interpret.

- Identify any outliers in your data distribution.
- Provide statistical analysis to see how values or trends change over time.

Configure a heatmap visualization

Once you've created a dashboard, the following video shows you how to configure a heatmap visualization:

Supported data formats

Heatmaps support time series data.

Example

The table below is a simplified output of random walk distribution over time:

Time	Walking (km)
2023-06-25 21:13:09	10
2023-08-25 21:13:10	8
2023-08-30 21:13:10	10
2023-10-08 21:13:11	12

Time	Walking (km)
2023-12-25 21:13:11	14
2024-01-05 21:13:12	13
2024-02-22 21:13:13	10

The data is converted as follows:

Panel options

In the **Panel options** section of the panel editor pane, set basic options like panel title and description, as well as panel links. To learn more, refer to Configure panel options.

Heatmap options

Calculate from data

This setting determines if the data is already a calculated heatmap (from the data source/transformer), or one that should be calculated in the panel.

X Bucket

This setting determines how the X-axis is split into buckets. You can specify a time interval in the **Size** input. For example, a time range of `1h` makes the cells 1-hour wide on the X-axis.

Y Bucket

This setting determines how the Y-axis is split into buckets.

Y Bucket scale

Select one of the following Y-axis value scales:

- **linear -** Linear scale.
- **log (base 2) -** Logarithmic scale with base 2.
- **log (base 10) -** Logarithmic scale with base 10.
- **symlog -** Symlog scale.

Y Axes

Defines how the Y axis is displayed

Placement

- **Left** On the left
- **Right** On the right
- **Hidden** Hidden

Unit

Unit configuration

Decimals

This setting determines decimal configuration.

Min/Max value

This setting configures the axis range.

Axis width

This setting configures the width for the axis.

Axis value

This setting configures the axis value.

Reverse

When selected, the axis appears in reverse order.

Multiple y-axes

In some cases, you might want to display multiple y-axes. For example, if you have a dataset showing both temperature and humidity over time, you might want to show two y-axes with different units for the two series.

You can configure multiple y-axes and control where they're displayed in the visualization by adding field overrides. This example of a dataset that includes temperature and humidity describes how you can configure that. Repeat the steps for every y-axis you wish to display.

Colors

The color spectrum controls the mapping between value count (in each bucket) and the color assigned to each bucket. The leftmost color on the spectrum represents the minimum count and the color on the right most side represents the maximum count. Some color schemes are automatically inverted when using the light theme.

You can also change the color mode to Opacity. In this case, the color will not change but the amount of opacity will change with the bucket count

- **Mode**
  - **Scheme -** Bucket value represented by cell color.
    - **Scheme -** If the mode is **scheme**, then select a color scheme.
  - **opacity -** Bucket value represented by cell opacity. Opaque cell means maximum value.
    - **Color -** Cell base color.
    - **Scale -** Scale for mapping bucket values to the opacity.
      - **linear -** Linear scale. Bucket value maps linearly to the opacity.
      - **sqrt -** Power scale. Cell opacity calculated as `value ^ k`, where `k` is a configured **Exponent** value. If exponent is less than `1`, you will get a logarithmic scale. If exponent is greater than `1`, you will get an exponential scale. In case of `1`, scale will be the same as linear.
    - **Exponent -** value of the exponent, greater than `0`.

Start/end color from value

By default, Grafana calculates cell colors based on minimum and maximum bucket values. With Min and Max you can overwrite those values. Consider a bucket value as a Z-axis and Min and Max as Z-Min and Z-Max, respectively.

- **Start -** Minimum value using for cell color calculation. If the bucket value is less than Min, then it is mapped to the "minimum" color. The series min value is the default value.
- **End -** Maximum value using for cell color calculation. If the bucket value is greater than Max, then it is mapped to the "maximum" color. The series max value is the default value.

Cell display

Use these settings to refine your visualization.

Additional display options

Tooltip

Tooltip mode

When you hover your cursor over the visualization, Grafana can display tooltips. Choose how tooltips behave.

- **Single -** The hover tooltip shows only a single series, the one that you are hovering over on the visualization.
- **All -** The hover tooltip shows all series in the visualization. Grafana highlights the series that you are hovering over in bold in the series list in the tooltip.
- **Hidden -** Do not display the tooltip when you interact with the visualization.

Use an override to hide individual series from the tooltip.

Show histogram (Y axis)

When you set the **Tooltip mode** to **Single**, this option is displayed. This option controls whether or not the tooltip includes a histogram representing the y-axis.

Show color scale

When you set the **Tooltip mode** to **Single**, this option is displayed. This option controls whether or not the tooltip includes the color scale that's also represented in the legend. When the color scale is included in the tooltip, it shows the hovered value on the scale:

Legend

Choose whether you want to display the heatmap legend on the visualization by toggling the **Show legend** switch.

Data links

Data links allow you to link to other panels, dashboards, and external resources while maintaining the context of the source panel. You can create links that include the series name or even the value under the cursor.

For each data link, set the following options:

- **Title**
- **URL**
- **Open in new tab**

To learn more, refer to Configure data links.

Field overrides

Overrides allow you to customize visualization settings for specific fields or series. When you add an override rule, it targets a particular set of fields and lets you define multiple options for how that field is displayed.

Choose from the following override options:

Option	Description
Fields with name	Select a field from the list of all available fields.
Field with name matching regex	Specify fields to override with a regular expression.
Fields with type	Select fields by type, such as string, numeric, or time.
Fields returned by query	Select all fields returned by a specific query, such as A, B, or C.
Fields with values	Select all fields returned by your defined reducer condition, such as **Min**, **Max**, **Count**, **Total**.

To learn more, refer to Configure field overrides.

Exemplars

Set the color used to show exemplar data.

Histogram

Histograms calculate the distribution of values and present them as a bar chart. Each bar represents a bucket; the y-axis and the height of each bar represent the count of values that fall into each bucket, and the x-axis represents the value range.

For example, if you want to understand the distribution of people's heights, you can use a histogram visualization to identify patterns or insights in the data distribution:

You can use a histogram visualization if you need to:

- Visualize and analyze data distributions over a specific time range to see how frequently certain values occur.
- Identify any outliers in your data distribution.
- Provide statistical analysis to help with decision-making

Configure a histogram visualization

Once you've created a dashboard, the following video shows you how to configure a histogram visualization:

Give it a try using Grafana Play

With Grafana Play, you can explore and see how it works, learning from practical examples to accelerate your development. This feature can be seen on Histogram Examples.

Supported data formats

Histograms support time series and any table results with one or more numerical fields.

Examples

The following tables are examples of the type of data you need for a histogram visualization and how it should be formatted.

Time-series table

Time	Walking (km)
2024-03-25 21:13:09	37.2

Time	Walking (km)
2024-03-25 21:13:10	37.1
2024-03-25 21:13:10	37.0
2024-03-25 21:13:11	37.2
2024-03-25 21:13:11	36.9
2024-03-25 21:13:12	36.7
2024-03-25 21:13:13	36.3

The data is converted as follows:

Basic numerical table

Gender	Height (kg)	Weight (lbs)
Male	73.8	242
Male	68.8	162
Male	74.1	213
Male	71.7	220
Male	69.9	206
Male	67.3	152
Male	68.8	184

The data is converted as follows:

Panel options

In the **Panel options** section of the panel editor pane, set basic options like panel title and description, as well as panel links. To learn more, refer to Configure panel options.

Histogram options

Use the following options to refine your histogram visualization.

Bucket count

Specifies the number of bins used to group your data in the histogram, affecting the granularity of the displayed distribution. Leave this empty for automatic bucket count of 30.

Bucket size

The size of the buckets. Leave this empty for automatic bucket sizing (~10% of the full range).

Bucket offset

If the first bucket should not start at zero. A non-zero offset has the effect of shifting the aggregation window. For example, 5-sized buckets that are 0-5, 5-10, 10-15 with a default 0 offset would become 2-7, 7-12, 12-17 with an offset of 2; offsets of 0, 5, or 10, in this case, would effectively do nothing. Typically, this option would be used with an explicitly defined bucket size rather than automatic. For this setting to affect, the offset amount should be greater than 0 and less than the bucket size; values outside this range will have the same effect as values within this range.

Combine series

This will merge all series and fields into a combined histogram.

Stacking

Controls how multiple series are displayed in the histogram. Choose from the following:

- **Off** - Series are not stacked, but instead shown side by side.
- **Normal** - Series are stacked on top of each other, showing cumulative values.
- **100%** - Series are stacked to fill 100% of the chart, showing the relative proportion of each series.

Line width

Controls line width of the bars.

Fill opacity

Controls the fill opacity bars.

Gradient mode

Set the mode of the gradient fill. Fill gradient is based on the line color. To change the color, use the standard color scheme field option.

Gradient display is influenced by the **Fill opacity** setting.

Choose from the following:

- **None** - No gradient fill. This is the default setting.
- **Opacity** - Transparency of the gradient is calculated based on the values on the Y-axis. The opacity of the fill is increasing with the values on the Y-axis.
- **Hue** - Gradient color is generated based on the hue of the line color.
- **Scheme** - The selected color palette is applied to the histogram bars.

Standard options

Standard options in the panel editor pane let you change how field data is displayed in your visualizations. When you set a standard option, the change is applied to all fields or series. For more granular control over the display of fields, refer to Configure overrides.

Option	Description
Unit	Choose which unit a field should use.
Min/Max	Set the minimum and maximum values used in percentage threshold calculations or leave these field empty for them to be calculated automatically.

Option	Description
Field min/max	Enable **Field min/max** to have Grafana calculate the min or max of each field individually, based on the minimum or maximum value of the field.
Decimals	Specify the number of decimals Grafana includes in the rendered value.
Display name	Set the display title of all fields. You can use variables in the field title.
Color scheme	Set single or multiple colors for your entire visualization.
No value	Enter what Grafana should display if the field value is empty or null. The default value is a hyphen (-).

To learn more, refer to Configure standard options.

Legend options

Legend options control the series names and statistics that appear under or to the right of the graph. For more information about the legend, refer to Configure a legend.

Option	Description
Visibility	Toggle the switch to turn the legend on or off.
Mode	Use these settings to define how the legend appears in your visualization. **List** displays the legend as a list. This is a default display mode of the legend. **Table** displays the legend as a table.
Placement	Choose where to display the legend. **Bottom** places the legend below the graph. **Right** places the legend to the right of the graph.
Width	Control how wide the legend is when placed on the right side of the visualization. This option is only displayed if you set the legend placement to **Right**.
Values	Choose which of the standard calculations to show in the legend. You can have more than one.

Value mappings

Value mapping is a technique you can use to change how data appears in a visualization.

For each value mapping, set the following options:

- **Condition** - Choose what's mapped to the display text and (optionally) color:
  - **Value** - Specific values
  - **Range** - Numerical ranges
  - **Regex** - Regular expressions

  - **Special** - Special values like `Null`, `NaN` (not a number), or boolean values like `true` and `false`
- **Display text**
- **Color** (Optional)
- **Icon** (Canvas only)

To learn more, refer to Configure value mappings.

Thresholds

A threshold is a value or limit you set for a metric that's reflected visually when it's met or exceeded. Thresholds are one way you can conditionally style and color your visualizations based on query results.

Set the following options:

- **Value** - Set the value for each threshold.
- **Thresholds mode** - Choose from:
  - **Absolute**
  - **Percentage**

To learn more, refer to Configure thresholds.

Data links

Data links allow you to link to other panels, dashboards, and external resources while maintaining the context of the source panel. You can create links that include the series name or even the value under the cursor.

For each data link, set the following options:

- **Title**
- **URL**
- **Open in new tab**

To learn more, refer to Configure data links.

Field overrides

Overrides allow you to customize visualization settings for specific fields or series. When you add an override rule, it targets a particular set of fields and lets you define multiple options for how that field is displayed.

Choose from the following override options:

Option	Description
Fields with name	Select a field from the list of all available fields.
Field with name matching regex	Specify fields to override with a regular expression.
Fields with type	Select fields by type, such as string, numeric, or time.
Fields returned by query	Select all fields returned by a specific query, such as A, B, or C.
Fields with values	Select all fields returned by your defined reducer condition, such as **Min**, **Max**, **Count**, **Total**.

To learn more, refer to Configure field overrides.

Logs

Logs are structured records of events or messages generated by a system or application—that is, a series of text records with status updates from your system or app. They generally include timestamps, messages, and context information like the severity of the logged event.

The logs visualization displays these records from data sources that support logs, such as Elastic, Influx, and Loki. The logs visualization has colored indicators of log status, as well as collapsible log events that help you analyze the information generated.

Give it a try using Grafana Play

With Grafana Play, you can explore and see how it works, learning from practical examples to accelerate your development. This feature can be seen on Logs Panel.

Typically, you use logs with a graph visualization to display the log output of a related process. If you have an incident in your application or systems, such as a website disruption or code failure, you can use the logs visualization to help you figure out what went wrong, when, and even why.

Configure a log visualization

The following video provides a walkthrough of creating a logs visualization. You'll also learn how to customize some settings and log visualization caveats:

Supported data formats

The logs visualization works best with log-type datasets such as queries from data sources like Loki, Elastic, and InlfuxDB.

You can also build log-formatted data from other data sources as long as the first field is a time type followed by string, number, and time fields. The leading time field is used to sort and timestamp the logs and if the data contains other time-type fields, they're included as elements of the logged record.

The second field is used as the log record title regardless of whether it's a time, numeric, or string field. Usually the second field is a text field containing multiple string elements, but if the message level (or `lvl`) is present, the visualization uses the values in it to add colors to the record, as described in Log levels integration.

Subsequent fields are collapsed inside of each log record and you can open them by clicking the expand (`>`) icon.

To limit the number of log lines rendered in the visualization, you can use the **Max data points** setting in the panel **Query options**. If that option isn't set, then the data source typically enforces its own default limit.

Example

Time	TitleMessage	Element1	Element2	Element3
2023-02-01 12:00:00	title=Log1 lvl=info	1	server2	2023-02-01 11:00:00
2023-02-01 11:30:00	title=Log1 lvl=error	1	server2	2023-02-01 11:00:00
2023-02-01 11:00:00	title=Log1 lvl=trace	1	server2	2023-02-01 11:00:00

Panel options

In the **Panel options** section of the panel editor pane, set basic options like panel title and description, as well as panel links. To learn more, refer to Configure panel options.

Log level

For logs where a **level** label is specified, we use the value of the label to determine the log level and update color accordingly. If the log doesn't have a level label specified, we try to find out if its content matches any of the supported expressions (see below for more information). The log level is always determined by the first match. In case Grafana is not able to determine a log level, it will be visualized with **unknown** log level. See supported log levels and mappings of log level abbreviation and expressions.

Log details

Each log row has an extendable area with its labels and detected fields, for more robust interaction. Each field or label has a stats icon to display ad-hoc statistics in relation to all displayed logs.

Display options

Use these settings to refine your visualization:

- **Time -** Show or hide the time column. This is the timestamp associated with the log line as reported from the data source.
- **Unique labels -** Show or hide the unique labels column, which shows only non-common labels.
- **Common labels -** Show or hide the common labels.
- **Wrap lines -** Toggle line wrapping.

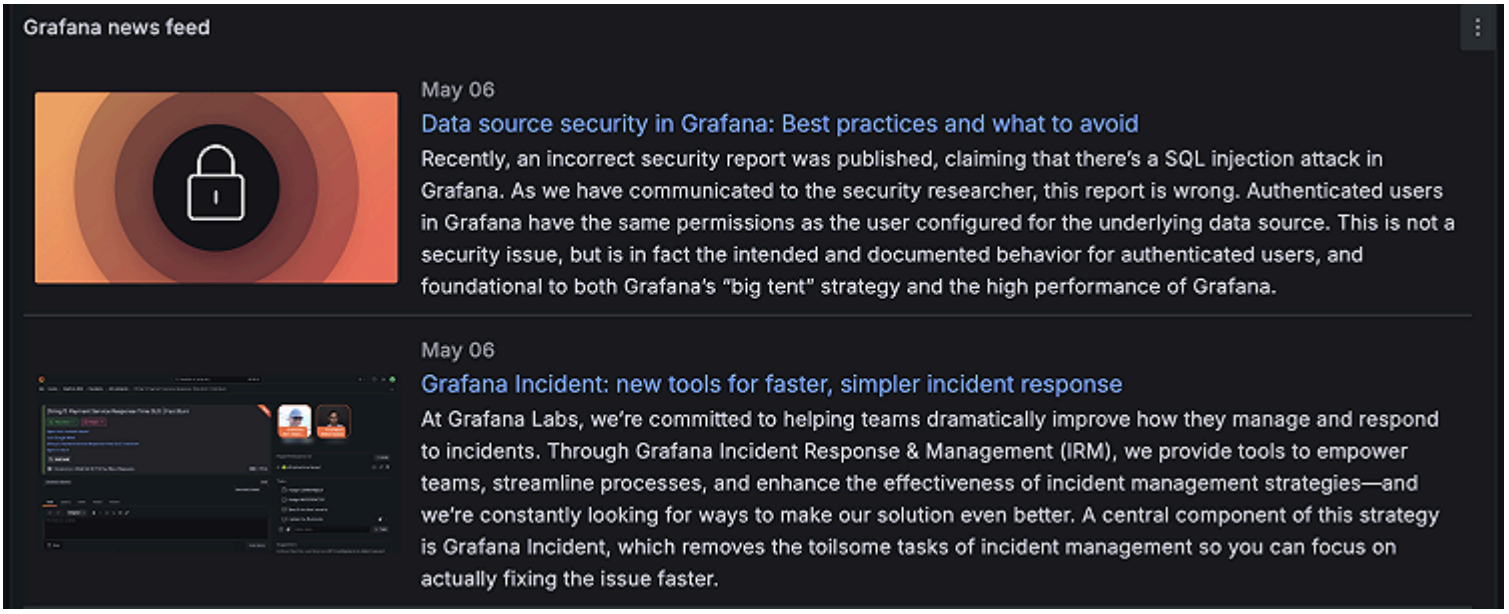
- **Prettify JSON -** Set this to `true` to pretty print all JSON logs. This setting does not affect logs in any format other than JSON.
- **Enable log details -** Toggle option to see the log details view for each log row. The default setting is true.
- **Deduplication -** Hides log messages that are duplicates of others shown according to your selected criteria. Choose from: **Exact** (ignoring ISO datetimes), **Numerical** (ignoring only those that differ by numbers such as IPs or latencies), or **Signatures** (removing successive lines with identical punctuation and white space).
- **Order -** Display results in descending or ascending time order. The default is **Descending**, showing the newest logs first. Set to **Ascending** to show the oldest log lines first.

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Grafana Cloud | Enterprise | Open source

News

The news visualization displays an RSS feed. By default, it displays articles from the Grafana Labs blog, but you can change this by entering a different RSS feed URL.



NOTE

In version 8.5, we discontinued the "Use Proxy" option for Grafana news visualizations. As a result, RSS feeds that are not configured for request by Grafana's frontend (with the appropriate CORS headers) may not load.

You can use the news visualization to provide regular news and updates to your users.

Give it a try using Grafana Play

With Grafana Play, you can explore and see how it works, learning from practical examples to accelerate your development. This feature can be seen on News Panel.

Configure a news visualization

Once you've created a dashboard, enter the URL of an RSS in the URL field in the **News** section. This visualization type doesn't accept any other queries, and you shouldn't expect to be able to filter or query the RSS feed data in any way using this visualization.

If you're having trouble loading an RSS feed, you can try rehosting the feed on a different server or using a CORS proxy. A CORS proxy is a tool that allows you to bypass CORS restrictions by making requests to the RSS feed on your behalf. You can find more information about using CORS proxies online.

If you're unable to display an RSS feed using the news visualization, you can try using the community RSS/Atom data source plugin RSS/Atom data source in combination with the Dynamic text community panel Dynamic text. This will allow you to display the RSS feed in a different way.

Supported data formats

The news visualization supports RSS and Atom feeds.

Panel options

In the **Panel options** section of the panel editor pane, set basic options like panel title and description, as well as panel links. To learn more, refer to Configure panel options.

News options

Use the following options to refine your news visualization.

URL

The URL of the RSS or Atom feed.

Show image

Controls if the news social image is displayed beside the text content.

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Grafana Cloud | Enterprise | Open source

Node graph

Node graphs are useful when you need to visualize elements that are related to each other. This is done by displaying circles—or *nodes*—for each element you want to visualize, connected by lines—or *edges*. The visualization uses a directed force layout that positions the nodes into a network of connected circles.

Node graphs display useful information about each node, as well as the relationships between them, allowing you to visualize complex infrastructure maps, hierarchies, or execution diagrams.

The appearance of nodes and edges can also be customized in several ways including color, borders, and line style.

You can use a node graph visualization if you need to show:

- Solution topologies
- Networks
- Infrastructure
- Organizational charts
- Critical path diagrams
- Family trees
- Mind maps

Configure a node graph visualization

The following video provides beginner steps for creating node panel visualizations. You'll learn the data requirements and caveats, special customizations, and much more:

Give it a try using Grafana Play

With Grafana Play, you can explore and see how it works, learning from practical examples to accelerate your development. This feature can be seen on Node graph panel.

Supported data formats

To create node graphs, you need two datasets: one containing the records for the displayed elements (nodes) and one dataset containing the records for the connections between those elements (edges).

Nodes dataset

The nodes dataset must contain one alphanumeric ID field that gives each element a unique identifier. The visualization also accepts other options fields for titles, subtitles, main and secondary stats, arc information for how much of the circle border to paint, details, colors, icons, node size, and indicators for element highlighting. For more information and naming conventions for these fields, refer to the Nodes data frame structure section.

Example

id	title	subtitle	mainstat	secondarystat	color	icon	highlighted
node1	PC	Windows	AMD	16gbRAM	blue		true
node2	PC	Linux	Intel	32gbRAM	green	eye	false
node3	Mac	MacOS	M3	16gbRAM	gray	apps	false
node4	Alone	SoLonely	JustHere	NotConnected	red		false

If the icon field contains a value, it's displayed instead of the title and subtitle. For a list of of available icons, refer to Icons Overview.

Edges dataset

Similar to the nodes dataset, the edges dataset needs one unique ID field for each relationship, followed by two fields containing the source and the target nodes of the edge; that is, the nodes the edge connects. Other optional fields are main and secondary stats, context menu elements, line thickness, highlight indications, line colors, and configurations to turn the connection into a dashed line. For more information and naming conventions for these fields, refer to the Edges data frame structure section.

Example

id	source	target	mainstat	seconddarystat	thickness	highlighted	cc
edge1	node1	node2	TheMain	TheSub	3	true	cy

id	source	target	mainstat	seconddarystat	thickness	highlighted	cc
edge2	node3	node2	Main2	Sub2	1	false	or

If a node lacks edge connections, it's displayed on its own outside of the network.

Panel options

In the **Panel options** section of the panel editor pane, set basic options like panel title and description, as well as panel links. To learn more, refer to Configure panel options.

Nodes options

The **Nodes** options section provides configurations for node behaviors.

- **Main stat unit** - Choose which unit the main stat displays in the graph's nodes.
- **Secondary stat unit** - Choose which unit the secondary stat displays in the graph's nodes.
- **Arc sections** - Configure which fields define the size of the colored circle around the node and select a color for each. You can add multiple fields.

> **NOTE**
>
> Defining arc sections overrides the automatic detection of `arc__*` and `color` fields described in the **Optional fields** section of Nodes data frame structure.

Edges options

The **Edges** options section provides configurations for node edges behaviors.

- **Main stat unit** - Choose which unit the main stat displays in the graph's edges.
- **Secondary stat unit** - Choose which unit the secondary stat displays in the graph's edges.

Data requirements

A node graph requires a specific shape of the data to be able to display its nodes and edges. This means not every data source or query can be visualized with this graph. If you want to use this as a data source developer see the section about data API.

A node graph consists of *nodes* and *edges*.

- A *node* is displayed as a circle. A node might represent an application, a service, or anything else that is relevant from an application perspective.
- An *edge* is displayed as a line that connects two nodes. The connection might be a request, an execution, or some other relationship between the two nodes.

Both nodes and edges can have associated metadata or statistics. The data source defines what information and values is shown, so different data sources can show different type of values or not show some values.

Nodes

> **NOTE**
>
> Node graphs can show only 1,500 nodes. If this limit is crossed a warning will be visible in upper right corner, and some nodes will be hidden. You can expand hidden parts of the graph by clicking on the “Hidden nodes” markers in the graph.

Usually, nodes show two statistical values inside the node and two identifiers just below the node, usually name and type. Nodes can also show another set of values as a color circle around the node, with sections of different color represents different values that should add up to 1.

For example, you can have the percentage of errors represented by a red portion of the circle. Additional details can be displayed in a context menu which is displayed when you click on the node. There also can be additional links in the context menu that can target either other parts of Grafana or any external link.

Edges

Edges can also show statistics when you hover over the edge. Similar to nodes, you can open a context menu with additional details and links by clicking on the edge.

The first data source supporting this visualization is X-Ray data source for its Service map feature. For more information, refer to the X-Ray plugin documentation.

Navigating the node graph

You can pan and zoom in or out a node graph.

Pan

You can pan the view by clicking outside any node or edge and dragging your mouse.

Zoom in or out

Use the buttons in the upper left corner or use the mouse wheel, touchpad scroll, together with either Ctrl or Cmd key to zoom in or out.

Explore hidden nodes

The number of nodes shown at a given time is limited to maintain a reasonable visualization performance. Nodes that are not currently visible are hidden behind clickable markers that show an approximate number of hidden nodes that are connected by a particular edge. You can click on the marker to expand the graph around that node.

Grid view

You can switch to the grid view to have a better overview of the most interesting nodes in the graph. Grid view shows nodes in a grid without edges and can be sorted by stats shown inside the node or by stats represented by the a colored border of the nodes.

To sort the nodes, click on the stats inside the legend. The marker next to the stat name shows which stat is currently used for sorting and sorting direction.

Click on the node and select “Show in Graph layout” option to switch back to graph layout and focus on the selected node, to show it in context of the full graph.

Data API

This visualization needs a specific shape of the data to be returned from the data source in order to correctly display it.

Node graphs, at minimum, require a data frame describing the edges of the graph. By default, node graphs will compute the nodes and any stats based on this data frame. Optionally a second data frame describing the nodes can be sent in case there is need to show more node specific metadata. You have to set `frame.meta.preferredVisualisationType = 'nodeGraph'` on both data frames or name them `nodes` and `edges` respectively for the node graph to render.

Edges data frame structure

Required fields:

Field name	Type	Description
id	string	Unique identifier of the edge.
source	string	Id of the source node.
target	string	Id of the target.

Optional fields:

Field name	Type	Description
mainstat	string/number	First stat shown in the overlay when hovering over the edge. It can be a string showing the value as is or it can be a number. If it is a number, any unit associated with that field is also shown
secondarystat	string/number	Same as mainStat, but shown right under it.
detail__*	string/number	Any field prefixed with `detail__` will be shown in the header of context menu when clicked on the edge. Use `config.displayName` for more human readable label.
thickness	number	The thickness of the edge. Default: `1`
highlighted	boolean	Sets whether the edge should be highlighted. Useful, for example, to represent a specific path in the graph by highlighting several nodes and edges. Default: `false`

Nodes data frame structure

Required fields:

Field name	Type	Description
id	string	Unique identifier of the node. This ID is referenced by edge in its source and target field.

Optional fields:

Field name	Type	Description
title	string	Name of the node visible in just under the node.
subtitle	string	Additional, name, type or other identifier shown under the title.
mainstat	string/number	First stat shown inside the node itself. It can either be a string showing the value as is or a number. If it is a number, any unit associated with that field is also shown.
secondarystat	string/number	Same as mainStat, but shown under it inside the node.
arc__*	number	Any field prefixed with `arc__` will be used to create the color circle around the node. All values in these fields should add up to 1. You can specify color using `config.color.fixedColor`.
detail__*	string/number	Any field prefixed with `detail__` will be shown in the header of context menu when clicked on the node. Use `config.displayName` for more human readable label.
color	string/number	Can be used to specify a single color instead of using the `arc__` fields to specify color sections. It can be either a string which

Field name	Type	Description
		should then be an acceptable HTML color string or it can be a number in which case the behaviour depends on `field.config.color.mode` setting. This can be for example used to create gradient colors controlled by the field value.
icon	string	Name of the icon to show inside the node instead of the default stats. Only Grafana built in icons are allowed (see the available icons here).
nodeRadius	number	Radius value in pixels. Used to manage node size.
highlighted	boolean	Sets whether the node should be highlighted. Useful for example to represent a specific path in the graph by highlighting several nodes and edges. Default: `false`

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Pie chart

A pie chart is a graph that displays data as segments of a circle proportional to the whole, making it look like a sliced pie. Each slice corresponds to a value or measurement.

The pie chart visualization is ideal when you have data that adds up to a total and you want to show the proportion of each value compared to other slices, as well as to the whole of the pie.

You can use a pie chart if you need to compare:

- Browser share distribution in the market
- Incident causes per category
- Network traffic sources
- User demographics

Configure a pie chart visualization

The following video guides you through the creation steps and common customizations of pie chart visualizations and is great for beginners:

Give it a try using Grafana Play

With Grafana Play, you can explore and see how it works, learning from practical examples to accelerate your development. This feature can be seen on Grafana Bar Charts and Pie Charts.

Supported data formats

The pie chart is different from other visualizations in that it will only display one pie, regardless of the number of datasets, fields, or records queried in it.

To create a pie chart visualization, you need a dataset containing a set of numeric values either in rows, columns, or both.

Example - One row

The easiest way to provide data for a pie chart visualization is in a dataset with a single record (or row) containing the fields (or columns) that you want in the pie, as in the following example. The default settings of the pie chart visualization automatically display each column as a slice of the pie.

Value1	Value2	Value3	Optional
5	3	2	Sums10

Example - Multiple rows

If you need to use numeric data that's in multiple rows, the default **Show** parameter of the visualization Value options is set to **Calculate** and use data from the last row.

Value	Label
5	Value1
3	Value2
2	Value3

By default, the visualization is configured to calculate a single value per column or series and to display only the last row of data.

To allow values in multiple rows to be displayed, change the **Show** setting in the Value options from **Calculate** to **All values**.

Example - Multiple rows and columns

If your dataset contains multiple rows and columns with numeric data, by default only the last row's values are summed.

Value1	Value2	Value3	Optional
5	3	2	Sums10
10	6	4	Sums20
20	8	2	Sums30

If you want to display all the cells, change the **Show** setting in the Value options from **Calculate** to **All values**. This also labels the elements by concatenating all the text fields (if you have any) with the column name.

If you want to display only the values from a given field (or column), once the **Show** setting in the Value options is set to **All values**, set the **Fields** option to the column you wish to sum in the display. The value labels are also concatenated as indicated before.

Panel options

In the **Panel options** section of the panel editor pane, set basic options like panel title and description, as well as panel links. To learn more, refer to Configure panel options.

Value options

Use the following options to refine the value in your visualization.

Show

Choose how much information to show.

- **Calculate -** Reduces each value to a single value per series.
- **All values -** Displays every value from a single series.

Calculation

Select a calculation to reduce each series when Calculate has been selected. For information about available calculations, refer to Calculation types.

Limit

When displaying every value from a single series, this limits the number of values displayed.

Fields

Select which field or fields to display in the visualization. Each field name is available on the list, or you can select one of the following options:

- **Numeric fields -** All fields with numerical values.
- **All fields -** All fields that are not removed by transformations.
- **Time -** All fields with time values.

Pie chart options

Use these options to refine how your visualization looks.

Pie chart type

Select the pie chart display style.

Pie

Donut

Labels

Select labels to display on the pie chart. You can select more than one.

- **Name -** The series or field name.
- **Percent -** The percentage of the whole.
- **Value -** The raw numerical value.

Labels are displayed in white over the body of the chart. You might need to select darker chart colors to make them more visible. Long names or numbers might be clipped.

The following example shows a pie chart with **Name** and **Percent** labels displayed.

Tooltip options

Tooltip options control the information overlay that appears when you hover over data points in the visualization.

Tooltip mode

When you hover your cursor over the visualization, Grafana can display tooltips. Choose how tooltips behave.

- **Single -** The hover tooltip shows only a single series, the one that you are hovering over on the visualization.
- **All -** The hover tooltip shows all series in the visualization. Grafana highlights the series that you are hovering over in bold in the series list in the tooltip.
- **Hidden -** Do not display the tooltip when you interact with the visualization.

Use an override to hide individual series from the tooltip.

Values sort order

When you set the **Tooltip mode** to **All**, the **Values sort order** option is displayed. This option controls the order in which values are listed in a tooltip. Choose from the following:

- **None** - Grafana automatically sorts the values displayed in a tooltip.
- **Ascending** - Values in the tooltip are listed from smallest to largest.
- **Descending** - Values in the tooltip are listed from largest to smallest.

Max height

Set the maximum height of the tooltip box. The default is 600 pixels.

Legend options

Use these settings to define how the legend appears in your visualization. For more information about the legend, refer to Configure a legend.

Visibility

Toggle the switch to turn the legend on or off.

Mode

Use these settings to define how the legend appears in your visualization.

- **List -** Displays the legend as a list. This is a default display mode of the legend.
- **Table -** Displays the legend as a table.

Placement

Choose where to display the legend.

- **Bottom -** Below the graph.
- **Right -** To the right of the graph.

Width

Control how wide the legend is when placed on the right side of the visualization. This option is only displayed if you set the legend placement to **Right**.

Values

Select values to display in the legend. You can select more than one.

- **Percent:** The percentage of the whole.
- **Value:** The raw numerical value.

Standard options

Standard options in the panel editor pane let you change how field data is displayed in your visualizations. When you set a standard option, the change is applied to all fields or series. For more granular control over the display of fields, refer to Configure overrides.

Option	Description
Unit	Choose which unit a field should use.
Min/Max	Set the minimum and maximum values used in percentage threshold calculations or leave these field empty for them to be calculated automatically.
Field min/max	Enable **Field min/max** to have Grafana calculate the min or max of each field individually, based on the minimum or maximum value of the field.
Decimals	Specify the number of decimals Grafana includes in the rendered value.
Display name	Set the display title of all fields. You can use variables in the field title.
Color scheme	Set single or multiple colors for your entire visualization.
No value	Enter what Grafana should display if the field value is empty or null. The default value is a hyphen (-).

To learn more, refer to Configure standard options.

Data links

Data links allow you to link to other panels, dashboards, and external resources while maintaining the context of the source panel. You can create links that include the series name or even the value under the cursor.

For each data link, set the following options:

- **Title**
- **URL**
- **Open in new tab**

To learn more, refer to Configure data links.

Value mappings

Value mapping is a technique you can use to change how data appears in a visualization.

For each value mapping, set the following options:

- **Condition** - Choose what's mapped to the display text and (optionally) color:
  - **Value** - Specific values
  - **Range** - Numerical ranges
  - **Regex** - Regular expressions
  - **Special** - Special values like `Null`, `NaN` (not a number), or boolean values like `true` and `false`

- **Display text**
- **Color** (Optional)
- **Icon** (Canvas only)

To learn more, refer to Configure value mappings.

Field overrides

Overrides allow you to customize visualization settings for specific fields or series. When you add an override rule, it targets a particular set of fields and lets you define multiple options for how that field is displayed.

Choose from the following override options:

Option	Description
Fields with name	Select a field from the list of all available fields.
Field with name matching regex	Specify fields to override with a regular expression.
Fields with type	Select fields by type, such as string, numeric, or time.
Fields returned by query	Select all fields returned by a specific query, such as A, B, or C.
Fields with values	Select all fields returned by your defined reducer condition, such as **Min**, **Max**, **Count**, **Total**.

To learn more, refer to Configure field overrides.

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Stat

NOTE

This visualization replaces the Singlestat visualization, which was deprecated in Grafana 7.0 and removed in Grafana 8.0.

A stat visualization displays your data in single values of interest—such as the latest or current value of a series—with an optional graph sparkline. A graph sparkline, which is only available in stat visualizations, is a small time-series graph shown in the background of each value in the visualization.

For example, if you're monitoring the utilization of various services, you can use a stat visualization to show their latest usage:

Use a stat visualization when you need to:

- Monitor key metrics at a glance, such as the latest health of your application, number of high priority bugs in your application, or total number of sales.
- Display aggregated data, such as the average response time of your services.
- Highlight values above your normal thresholds to quickly identify if any metrics are outside your expected range.

Give it a try using Grafana Play

With Grafana Play, you can explore and see how it works, learning from practical examples to accelerate your development. This feature can be seen on Stat Visualizations in Grafana.

Configure a stat visualization

Once you've created a dashboard, the following video shows you how to configure a stat visualization:

Alternatively, refer to this blog post on how to easily retrieve values from a range in Grafana using a stat visualization.

Supported data formats

The stat visualization supports a variety of formats for displaying data. Supported formats include:

- **Single values** - The most common format and can be numerical, strings, or boolean values.
- **Time-series data** - Calculation types can be applied to your time-series data to display single values over a specified time range.

Examples

The following tables are examples of the type of data you need for a stat visualization and how it should be formatted.

Single numerical values

Number of high priority bugs
80
52
59
40

The data is visualized as follows, with the last value displayed, along with a sparkline and percentage change:

Time-series data

Time	Cellar	Living room	Porch	Bedroom	Guest room	Kitchen
2024-03-20 06:34:40	12.3	18.3	18.8	15.9	9.29	9.61
2024-03-20 06:41:40	16.8	17.1	21.5	14.1	10.5	17.5
2024-03-20 06:48:40	16.7	18.0	21.0	9.51	13.6	20.1
2024-03-20 06:55:40	14.3	18.7	16.5	9.11	14.8	12.5
2024-03-20 07:02:40	12.8	15.2	21.1	15.6	7.98	13.0

The data is visualized as follows, with the mean value displayed for each room, along with the room name, sparkline, and unit of measurement:

By default, a stat displays one of the following:

- Just the value for a single series or field.
- Both the value and name for multiple series or fields.

You can use the **Text mode** to control how the text is displayed.

Automatic layout adjustment

The panel automatically adjusts the layout depending on available width and height in the dashboard. It automatically hides the graph (sparkline) if the panel becomes too small.

Panel options

In the **Panel options** section of the panel editor pane, set basic options like panel title and description, as well as panel links. To learn more, refer to Configure panel options.

Value options

Use the following options to refine how your visualization displays its values:

Show

Display a single value per column or series, or show values for each row.

Calculate

Display a calculated value based on all rows.

- **Calculation -** Select a reducer function that Grafana will use to reduce many fields to a single value. For a list of available calculations, refer to Calculation types.
- **Fields -** Select the fields display in the visualization.

All values

Show a separate stat for every row. If you select this option, then you can also limit the number of rows to display.

- **Limit -** The maximum number of rows to display. Default is 5,000.
- **Fields -** Select the fields display in the visualization.

Stat styles

Style your visualization.

Orientation

Choose a stacking direction.

- **Auto -** Grafana selects what it thinks is the best orientation.
- **Horizontal -** Bars stretch horizontally, left to right.
- **Vertical -** Bars stretch vertically, top to bottom.

Text mode

You can use the Text mode option to control what text the visualization renders. If the value is not important, only the name and color is, then change the **Text mode** to **Name**. The value will still be used to determine color and is displayed in a tooltip.

- **Auto -** If the data contains multiple series or fields, show both name and value.
- **Value -** Show only value, never name. Name is displayed in the hover tooltip instead.
- **Value and name -** Always show value and name.
- **Name -** Show name instead of value. Value is displayed in the hover tooltip.
- **None -** Show nothing (empty). Name and value are displayed in the hover tooltip.

Wide layout

Set whether wide layout is enabled or not. Wide layout is enabled by default.

- **On -** Wide layout is enabled.
- **Off -** Wide layout is disabled.

> **NOTE**
>
> This option is only applicable when **Text mode** is set to **Value and name**. When wide layout is enabled, the value and name are displayed side-by-side with the value on the right, if the panel is wide enough. When wide layout is disabled, the value is always rendered underneath the name.

Color mode

Select a color mode.

- **None -** No color applied to the value.
- **Value -** Applies color to the value and graph area.
- **Background Gradient -** Applies color to the value, graph area, and background, with a slight background gradient.
- **Background Solid -** Applies color to the value, graph area, and background, with a solid background color.

Graph mode

Select a graph and sparkline mode.

- **None -** Hides the graph and only shows the value.
- **Area -** Shows the area graph below the value. This requires that your query returns a time column.

Text alignment

Choose an alignment mode.

- **Auto -** If only a single value is shown (no repeat), then the value is centered. If multiple series or rows are shown, then the value is left-aligned.
- **Center -** Stat value is centered.

Show percent change

Set whether percent change is displayed or not. Disabled by default.

> **NOTE**
>
> This option is not applicable when the **Show** setting, under **Value options**, is set to **All values**.

Text size

Adjust the sizes of the gauge text.

- **Title -** Enter a numeric value for the gauge title size.
- **Value -** Enter a numeric value for the gauge value size.

Standard options

Standard options in the panel editor pane let you change how field data is displayed in your visualizations. When you set a standard option, the change is applied to all fields or series. For more granular control over the display of fields, refer to Configure overrides.

Option	Description
Unit	Choose which unit a field should use.
Min/Max	Set the minimum and maximum values used in percentage threshold calculations or leave these field empty for them to be calculated automatically.
Field min/max	Enable **Field min/max** to have Grafana calculate the min or max of each field individually, based on the minimum or maximum value of the field.
Decimals	Specify the number of decimals Grafana includes in the rendered value.
Display name	Set the display title of all fields. You can use variables in the field title.
Color scheme	Set single or multiple colors for your entire visualization.
No value	Enter what Grafana should display if the field value is empty or null. The default value is a hyphen (-).

To learn more, refer to Configure standard options.

Data links

Data links allow you to link to other panels, dashboards, and external resources while maintaining the context of the source panel. You can create links that include the series name or even the value under the cursor.

For each data link, set the following options:

- **Title**
- **URL**
- **Open in new tab**

To learn more, refer to Configure data links.

Value mappings

Value mapping is a technique you can use to change how data appears in a visualization.

For each value mapping, set the following options:

- **Condition** - Choose what's mapped to the display text and (optionally) color:
  - **Value** - Specific values
  - **Range** - Numerical ranges
  - **Regex** - Regular expressions
  - **Special** - Special values like `Null`, `NaN` (not a number), or boolean values like `true` and `false`
- **Display text**
- **Color** (Optional)
- **Icon** (Canvas only)

To learn more, refer to Configure value mappings.

Thresholds

A threshold is a value or limit you set for a metric that's reflected visually when it's met or exceeded. Thresholds are one way you can conditionally style and color your visualizations based on query results.

Set the following options:

- **Value** - Set the value for each threshold.
- **Thresholds mode** - Choose from:
  - **Absolute**
  - **Percentage**

To learn more, refer to Configure thresholds.

Field overrides

Overrides allow you to customize visualization settings for specific fields or series. When you add an override rule, it targets a particular set of fields and lets you define multiple options for how that field is displayed.

Choose from the following override options:

Option	Description
Fields with name	Select a field from the list of all available fields.
Field with name matching regex	Specify fields to override with a regular expression.
Fields with type	Select fields by type, such as string, numeric, or time.
Fields returned by query	Select all fields returned by a specific query, such as A, B, or C.

Option	Description
Fields with values	Select all fields returned by your defined reducer condition, such as **Min**, **Max**, **Count**, **Total**.

To learn more, refer to Configure field overrides.

State timeline

A state timeline visualization displays data in a way that shows state changes over time. In a state timeline, the data is presented as a series of bars or bands called *state regions*. State regions can be rendered with or without values, and the region length indicates the duration or frequency of a state within a given time range.

For example, if you're monitoring the CPU usage of a server, you can use a state timeline to visualize the different states, such as "LOW," "NORMAL," "HIGH," or "CRITICAL," over time. Each state is represented by a different color and the lengths represent the duration of time that the server remained in that state:

The state timeline visualization is useful when you need to monitor and analyze changes in states or statuses of various entities over time. You can use one when you need to:

- Monitor the status of a server, application, or service to know when your infrastructure is experiencing issues over time.
- Identify operational trends over time.
- Spot any recurring issues with the health of your applications.

Configure a state timeline

Give it a try using Grafana Play

With Grafana Play, you can explore and see how it works, learning from practical examples to accelerate your development. This feature can be seen on Grafana State Timeline & Status History.

Supported data formats

The state timeline visualization works best if you have data capturing the various states of entities over time, formatted as a table. The data must include:

- **Timestamps** - Indicate when each state change occurred. This could also be the start time for the state change. You can also add an optional timestamp to indicate the end time for the state change.
- **Entity name/identifier** - Represents the name of the entity you're trying to monitor.
- **State value** - Represents the state value of the entity you're monitoring. These can be string, numerical, or boolean states.

Each state ends when the next state begins or when there is a `null` value.

Examples

The following tables are examples of the type of data you need for a state timeline visualization and how it should be formatted.

Single time column with null values

Timestamps	Server A	Server B
2024-02-29 8:00:00	Up	Up
2024-02-29 8:15:00	null	Up
2024-02-29 8:30:00	Down	null
2024-02-29 8:45:00		Up
2024-02-29 9:00:00	Up	
2024-02-29 9:15:00	Up	Down
2024-02-29 9:30:00	Up	Down
2024-02-29 10:00:00	Down	Down
2024-02-29 10:30:00	Warning	Down

The data is converted as follows, with the null and empty values visualized as gaps in the state timeline:

Two time columns without null values

Start time	End time	Server A	Server B
2024-02-29 8:00:00	2024-02-29 8:15:00	Up	Up
2024-02-29 8:15:00	2024-02-29 8:30:00	Up	Up
2024-02-29 8:45:00	2024-02-29 9:00:00	Down	Up
2024-02-29 9:00:00	2024-02-29 9:15:00	Down	Up
2024-02-29 9:30:00	2024-02-29 10:00:00	Down	Down
2024-02-29 10:00:00	2024-02-29 10:30:00	Warning	Down

The data is converted as follows:

If your query results aren't in a table format like the preceding examples, especially for time-series data, you can apply specific transformations to achieve this.

Panel options

In the **Panel options** section of the panel editor pane, set basic options like panel title and description, as well as panel links. To learn more, refer to Configure panel options.

State timeline options

Use these options to refine the visualization.

Merge equal consecutive values

Controls whether Grafana merges identical values if they are next to each other.

Show values

Controls whether values are rendered inside the state regions. Auto will render values if there is sufficient space.

Align values

Controls value alignment inside state regions.

Row height

Controls how much space between rows there are. 1 = no space = 0.5 = 50% space.

Line width

Controls line width of state regions.

Fill opacity

Controls the opacity of state regions.

Connect null values

Choose how null values, which are gaps in the data, appear on the graph. Null values can be connected to form a continuous line or set to a threshold above which gaps in the data are no longer connected.

- **Never** - Time series data points with gaps in the data are never connected.
- **Always** - Time series data points with gaps in the data are always connected.
- **Threshold** - Specify a threshold above which gaps in the data are no longer connected. This can be useful when the connected gaps in the data are of a known size and/or within a known range, and gaps outside this range should no longer be connected.

Disconnect values

Choose whether to set a threshold above which values in the data should be disconnected.

- **Never** - Time series data points in the data are never disconnected.
- **Threshold** - Specify a threshold above which values in the data are disconnected. This can be useful when desired values in the data are of a known size and/or within a known range, and values outside this range should no longer be connected.

Time series data with thresholds

The visualization can be used with time series data as well. In this case, the thresholds are used to turn the time series into discrete colored state regions.

state timeline with time series

Standard options

Standard options in the panel editor pane let you change how field data is displayed in your visualizations. When you set a standard option, the change is applied to all fields or series. For

more granular control over the display of fields, refer to Configure overrides.

Option	Description
Unit	Choose which unit a field should use.
Min/Max	Set the minimum and maximum values used in percentage threshold calculations or leave these field empty for them to be calculated automatically.
Field min/max	Enable **Field min/max** to have Grafana calculate the min or max of each field individually, based on the minimum or maximum value of the field.
Decimals	Specify the number of decimals Grafana includes in the rendered value.
Display name	Set the display title of all fields. You can use variables in the field title.
Color scheme	Set single or multiple colors for your entire visualization.
No value	Enter what Grafana should display if the field value is empty or null. The default value is a hyphen (-).

To learn more, refer to Configure standard options.

Legend options

When the legend option is enabled it can show either the value mappings or the threshold brackets. To show the value mappings in the legend, it's important that the **Color scheme** as referenced in Color scheme is set to **Single color** or **Classic palette**. To see the threshold brackets in the legend set the **Color scheme** to **From thresholds**.

For more information about the legend, refer to Configure a legend.

Visibility

Toggle the switch to turn the legend on or off.

Mode

Use these settings to define how the legend appears in your visualization.

- **List -** Displays the legend as a list. This is a default display mode of the legend.
- **Table -** Displays the legend as a table.

Placement

Choose where to display the legend.

- **Bottom -** Below the graph.

- **Right -** To the right of the graph.

Width

Control how wide the legend is when placed on the right side of the visualization. This option is only displayed if you set the legend placement to **Right**.

Tooltip options

Tooltip options control the information overlay that appears when you hover over data points in the visualization.

Tooltip mode

When you hover your cursor over the visualization, Grafana can display tooltips. Choose how tooltips behave.

- **Single -** The hover tooltip shows only a single series, the one that you are hovering over on the visualization.
- **All -** The hover tooltip shows all series in the visualization. Grafana highlights the series that you are hovering over in bold in the series list in the tooltip.
- **Hidden -** Do not display the tooltip when you interact with the visualization.

Use an override to hide individual series from the tooltip.

Values sort order

When you set the **Tooltip mode** to **All**, the **Values sort order** option is displayed. This option controls the order in which values are listed in a tooltip. Choose from the following:

- **None** - Grafana automatically sorts the values displayed in a tooltip.
- **Ascending** - Values in the tooltip are listed from smallest to largest.
- **Descending** - Values in the tooltip are listed from largest to smallest.

Max height

Set the maximum height of the tooltip box. The default is 600 pixels.

Data links

Data links allow you to link to other panels, dashboards, and external resources while maintaining the context of the source panel. You can create links that include the series name or even the value under the cursor.

For each data link, set the following options:

- **Title**
- **URL**
- **Open in new tab**

To learn more, refer to Configure data links.

Value mappings

Value mapping is a technique you can use to change how data appears in a visualization.

For each value mapping, set the following options:

- **Condition** - Choose what's mapped to the display text and (optionally) color:
  - **Value** - Specific values
  - **Range** - Numerical ranges
  - **Regex** - Regular expressions
  - **Special** - Special values like `Null`, `NaN` (not a number), or boolean values like `true` and `false`
- **Display text**
- **Color** (Optional)
- **Icon** (Canvas only)

To learn more, refer to Configure value mappings.

Value mappings side editor

Thresholds

A threshold is a value or limit you set for a metric that's reflected visually when it's met or exceeded. Thresholds are one way you can conditionally style and color your visualizations based on query results.

Set the following options:

- **Value** - Set the value for each threshold.
- **Thresholds mode** - Choose from:
  - **Absolute**

- **Percentage**

To learn more, refer to Configure thresholds.

Field overrides

Overrides allow you to customize visualization settings for specific fields or series. When you add an override rule, it targets a particular set of fields and lets you define multiple options for how that field is displayed.

Choose from the following override options:

Option	Description
Fields with name	Select a field from the list of all available fields.
Field with name matching regex	Specify fields to override with a regular expression.
Fields with type	Select fields by type, such as string, numeric, or time.
Fields returned by query	Select all fields returned by a specific query, such as A, B, or C.
Fields with values	Select all fields returned by your defined reducer condition, such as **Min**, **Max**, **Count**, **Total**.

To learn more, refer to Configure field overrides.

Status history

A status history visualization displays data in a way that shows periodic states over time. In a status history, each field or series is rendered as a horizontal row, with multiple boxes showing the different statuses. This provides you with a centralized view for the status of a component or service.

For example, if you're monitoring the health status of different services, you can use a status history to visualize the different statuses, such as "OK," "WARN," or "BAD," over time. Each status is represented by a different color:

NOTE

A status history is similar to a state timeline, but has different configuration options. Unlike state timelines, status histories don't merge consecutive values.

Use a status history when you need to:

- Monitor the status of a server, application, or service to know when your infrastructure is experiencing issues over time.
- Identify operational trends over time.

- Spot any recurring issues with the health of your applications.

Configure a status history

Once you've created a dashboard, you can use the following state timeline video as a reference for how to configure a status history:

Give it a try using Grafana Play

With Grafana Play, you can explore and see how it works, learning from practical examples to accelerate your development. This feature can be seen on Grafana State Timeline & Status History.

Supported data formats

The status history visualization works best if you have data capturing the various status of entities over time, formatted as a table. The data must include:

- **Timestamps** - Indicate when each status change occurred. This could also be the start time for the status change. You can also add an optional timestamp to indicate the end time for the status change.
- **Entity name/identifier** - Represents the name of the entity you're trying to monitor.
- **Status value** - Represents the state value of the entity you're monitoring. These can be string, numerical, or boolean states.

Examples

The following tables are examples of the type of data you need for a status history visualization and how it should be formatted.

Single time column with null values

Timestamps	Backend_01	Backend_02
2024-02-29 8:00:00	OK	WARN
2024-02-29 8:15:00	WARN	
2024-02-29 8:18:00		WARN
2024-02-29 8:30:00	BAD	
2024-02-29 8:36:00		OK
2024-02-29 8:45:00	OK	

The data is converted as follows, with the null and empty values visualized as gaps in the status history:

Two time columns without null values

Start time	End time	Backend_01	Backend_02
2024-02-29 8:00:00	2024-02-29 8:15:00	OK	OK
2024-02-29 8:15:00	2024-02-29 8:30:00	OK	OK
2024-02-29 8:30:00	2024-02-29 8:45:00	OK	OK

Start time	End time	Backend_01	Backend_02
2024-02-29 8:45:00	2024-02-29 9:00:00	BAD	WARN
2024-02-29 9:00:00	2024-02-29 9:15:00	OK	WARN

The data is converted as follows:

Panel options

In the **Panel options** section of the panel editor pane, set basic options like panel title and description, as well as panel links. To learn more, refer to Configure panel options.

Status history options

Use these options to refine the visualization.

Show values

Controls whether values are rendered inside the value boxes. Auto will render values if there is sufficient space.

Row height

Controls the height of boxes. 1 = maximum space and 0 = minimum space.

Column width

Controls the width of boxes. 1 = maximum space and 0 = minimum space.

Line width

Controls line width of state regions.

Fill opacity

Controls the opacity of state regions.

Legend options

When the legend option is enabled it can show either the value mappings or the threshold brackets. To show the value mappings in the legend, it's important that the **Color scheme** as referenced in Color scheme is set to **Single color** or **Classic palette**. To see the threshold brackets in the legend set the **Color scheme** to **From thresholds**.

For more information about the legend, refer to Configure a legend.

Visibility

Toggle the switch to turn the legend on or off.

Mode

Use these settings to define how the legend appears in your visualization.

- **List -** Displays the legend as a list. This is a default display mode of the legend.
- **Table -** Displays the legend as a table.

Placement

Choose where to display the legend.

- **Bottom -** Below the graph.
- **Right -** To the right of the graph.

Width

Control how wide the legend is when placed on the right side of the visualization. This option is only displayed if you set the legend placement to **Right**.

Tooltip options

Tooltip options control the information overlay that appears when you hover over data points in the visualization.

Tooltip mode

When you hover your cursor over the visualization, Grafana can display tooltips. Choose how tooltips behave.

- **Single -** The hover tooltip shows only a single series, the one that you are hovering over on the visualization.
- **All -** The hover tooltip shows all series in the visualization. Grafana highlights the series that you are hovering over in bold in the series list in the tooltip.
- **Hidden -** Do not display the tooltip when you interact with the visualization.

Use an override to hide individual series from the tooltip.

Values sort order

When you set the **Tooltip mode** to **All**, the **Values sort order** option is displayed. This option controls the order in which values are listed in a tooltip. Choose from the following:

- **None** - Grafana automatically sorts the values displayed in a tooltip.
- **Ascending** - Values in the tooltip are listed from smallest to largest.
- **Descending** - Values in the tooltip are listed from largest to smallest.

Max height

Set the maximum height of the tooltip box. The default is 600 pixels.

Standard options

Standard options in the panel editor pane let you change how field data is displayed in your visualizations. When you set a standard option, the change is applied to all fields or series. For more granular control over the display of fields, refer to Configure overrides.

Option	Description
Unit	Choose which unit a field should use.
Min/Max	Set the minimum and maximum values used in percentage threshold calculations or leave these field empty for them to be calculated automatically.
Field min/max	Enable **Field min/max** to have Grafana calculate the min or max of each field individually, based on the minimum or maximum value of the field.
Decimals	Specify the number of decimals Grafana includes in the rendered value.
Display name	Set the display title of all fields. You can use variables in the field title.
Color scheme	Set single or multiple colors for your entire visualization.
No value	Enter what Grafana should display if the field value is empty or null. The default value is a hyphen (-).

To learn more, refer to Configure standard options.

Data links

Data links allow you to link to other panels, dashboards, and external resources while maintaining the context of the source panel. You can create links that include the series name or even the value under the cursor.

For each data link, set the following options:

- **Title**
- **URL**
- **Open in new tab**

To learn more, refer to Configure data links.

Value mappings

Value mapping is a technique you can use to change how data appears in a visualization.

For each value mapping, set the following options:

- **Condition** - Choose what's mapped to the display text and (optionally) color:
  - **Value** - Specific values
  - **Range** - Numerical ranges
  - **Regex** - Regular expressions
  - **Special** - Special values like `Null`, `NaN` (not a number), or boolean values like `true` and `false`
- **Display text**
- **Color** (Optional)
- **Icon** (Canvas only)

To learn more, refer to Configure value mappings.

Value mappings side editor

Thresholds

A threshold is a value or limit you set for a metric that's reflected visually when it's met or exceeded. Thresholds are one way you can conditionally style and color your visualizations based on query results.

Set the following options:

- **Value** - Set the value for each threshold.
- **Thresholds mode** - Choose from:
  - **Absolute**
  - **Percentage**

To learn more, refer to Configure thresholds.

Field overrides

Overrides allow you to customize visualization settings for specific fields or series. When you add an override rule, it targets a particular set of fields and lets you define multiple options for how that field is displayed.

Choose from the following override options:

Option	Description
Fields with name	Select a field from the list of all available fields.
Field with name matching regex	Specify fields to override with a regular expression.
Fields with type	Select fields by type, such as string, numeric, or time.
Fields returned by query	Select all fields returned by a specific query, such as A, B, or C.
Fields with values	Select all fields returned by your defined reducer condition, such as **Min**, **Max**, **Count**, **Total**.

To learn more, refer to Configure field overrides.

Table

Tables are a highly flexible visualization designed to display data in columns and rows. They support various data types, including tables, time series, annotations, and raw JSON data. The table visualization can even take multiple data sets and provide the option to switch between them. With this versatility, it's the preferred visualization for viewing multiple data types, aiding in your data analysis needs.

You can use a table visualization to show datasets such as:

- Common database queries like logs, traces, metrics
- Financial reports
- Customer lists
- Product catalogs

Any information you might want to put in a spreadsheet can often be best visualized in a table.

Tables also provide different styles to visualize data inside the table cells such as colored text and cell backgrounds, gauges, sparklines, data links, JSON code, and images.

Configure a table visualization

The following video provides a visual walkthrough of the options you can set in a table visualization. If you want to see a configuration in action, check out the video:

Give it a try using Grafana Play

With Grafana Play, you can explore and see how it works, learning from practical examples to accelerate your development. This feature can be seen on Table Visualizations in Grafana.

NOTE

Annotations and alerts are not currently supported for tables.

Supported data formats

The table visualization supports any data that has a column-row structure.

Example

```
Column1, Column2, Column3
value1 , value2 , value3
```

Copy

```
value4 , value5 , value6
value7 , value8 , value9
```

If a cell is missing or the table cell-row structure is not complete, the table visualization won't display any of the data:

```
Column1, Column2, Column3
value1 , value2 , value3
gap1   , gap2
value4 , value5 , value6
```

Copy

If you need to hide columns, you can do so using data transformations, field overrides, or by building a query that returns only the needed columns.

If you're using a cell type such as sparkline or JSON, the data requirements may differ in a way that's specific to that type. For more info refer to Cell type.

Debugging in tables

The table visualization helps with debugging when you need to know exactly what results your query is returning and why other visualizations might not be working. This functionality is also accessible in most visualizations by toggling on the **Table view** switch at the top of the panel:

Sort column

Click a column title to change the sort order from default to descending to ascending. Each time you click, the sort order changes to the next option in the cycle. You can sort multiple columns by holding the `shift` key and clicking the column name.

Data set selector

If the data queried contains multiple data sets, a table displays a drop-down list at the bottom, so you can select the data set you want to visualize.

Panel options

In the **Panel options** section of the panel editor pane, set basic options like panel title and description, as well as panel links. To learn more, refer to Configure panel options.

Table options

> **NOTE**
>
> If you are using a table created before Grafana 7.0, then you need to migrate to the new table version in order to see these options. To migrate, on the Panel tab, click **Table** visualization. Grafana updates the table version and you can then access all table options.

Show header

Show or hide column names imported from your data source.

Column width

By default, Grafana automatically calculates the column width based on the table size and the minimum column width. This field option can override the setting and define the width for all columns in pixels.

For example, if you enter `100` in the field, then when you click outside the field, all the columns will be set to 100 pixels wide.

Minimum column width

By default, the minimum width of the table column is 150 pixels. This field option can override that default and will define the new minimum column width for the table in pixels.

For example, if you enter `75` in the field, then when you click outside the field, all the columns will scale to no smaller than 75 pixels wide.

For small-screen devices, such as smartphones or tablets, reduce the default `150` pixel value to `50` to allow table-based panels to render correctly in dashboards.

Column alignment

Choose how Grafana should align cell contents:

- Auto (default)
- Left
- Center
- Right

Cell type

By default, Grafana automatically chooses display settings. You can override the settings by choosing one of the following options to set the default for all fields. Additional configuration is available for some cell types.

> **NOTE**
>
> If you set these in the Field tab, then the type will apply to all fields, including the time field. Many options will work best if you set them in the Override tab so that they can be restricted to one or more fields.

Color text

If thresholds are set, then the field text is displayed in the appropriate threshold color.

Color background (gradient or solid)

If thresholds are set, then the field background is displayed in the appropriate threshold color.

Color background

Gauge

Cells can be displayed as a graphical gauge, with several different presentation types.

> **NOTE**
>
> The maximum and minimum values of the gauges are configured automatically from the smallest and largest values in your whole data set. If you don't want the max/min values to be pulled from the whole data set, you can configure them for each column with field overrides.

Basic

The basic mode will show a simple gauge with the threshold levels defining the color of gauge.

Gradient gauge

Gradient

The threshold levels define a gradient.

Gradient gauge

LCD

The gauge is split up in small cells that are lit or unlit.

LCD gauge

Label Options

Additionally, labels displayed alongside of the gauges can be set to be colored by value, match the theme text color, or be hidden.

Value Color

Color Label by Value

Text Color

Color Label by theme color

Hidden

Hide Label

Data links

If you’ve configured data links, when the cell type is **Auto** mode, the cell text becomes clickable. If you change the cell type to **Data links**, the cell text reflects the titles of the configured data links. To control the application of data link text more granularly use a **Cell option > Cell type > Data links** field override.

JSON view

Shows value formatted as code. If a value is an object the JSON view allowing browsing the JSON object will appear on hover.

JSON view

Image

Only available in Grafana 7.3+

If you have a field value that is an image URL or a base64 encoded image you can configure the table to display it as an image.

Table hover

Sparkline

Shows value rendered as a sparkline. Requires time series to table data transform.

Sparkline

Cell value inspect

Enables value inspection from table cell. The raw value is presented in a modal window.

NOTE

Cell value inspection is only available when cell display mode is set to Auto, Color text, Color background or JSON View.

Column filter

You can temporarily change how column data is displayed. For example, you can order values from highest to lowest or hide specific values. For more information, refer to Filter table columns.

Pagination

Use this option to enable or disable pagination. It is a front-end option that does not affect queries. When enabled, the page size automatically adjusts to the height of the table.

Filter table columns

If you turn on the **Column filter**, then you can filter table options.

Turn on column filtering

1. In Grafana, navigate to the dashboard with the table with the columns that you want to filter.
2. On the table panel you want to filter, open the panel editor.
3. Click the **Field** tab.
4. In Table options, turn on the **Column filter** option.

A filter icon appears next to each column title.

Column filtering turned on

Filter column values

To filter column values, click the filter (funnel) icon next to a column title. Grafana displays the filter options for that column.

Filter column values

Click the check box next to the values that you want to display. Enter text in the search field at the top to show those values in the display so that you can select them rather than scroll to find them.

Choose from several operators to display column values:

- **Contains** - Matches a regex pattern (operator by default).
- **Expression** - Evaluates a boolean expression. The character `$` represents the column value in the expression (for example, “$ >= 10 && $ <= 12”).
- The typical comparison operators: `=`, `!=`, `<`, `<=`, `>`, `>=`.

Click the check box above the **Ok** and **Cancel** buttons to add or remove all displayed values to/from the filter.

Clear column filters

Columns with filters applied have a blue funnel displayed next to the title.

Filtered column

To remove the filter, click the blue funnel icon and then click **Clear filter**.

Table footer

You can use the table footer to show calculations on fields.

After you enable the table footer:

1. Select the **Calculation**
2. Select the **Fields** that you want to calculate

The system applies the calculation to all numeric fields if you do not select a field.

Count rows

If you want to show the number of rows in the dataset instead of the number of values in the selected fields, select the **Count** calculation and enable **Count rows**.

Standard options

Standard options in the panel editor pane let you change how field data is displayed in your visualizations. When you set a standard option, the change is applied to all fields or series. For more granular control over the display of fields, refer to Configure overrides.

Option	Description
Unit	Choose which unit a field should use.
Min/Max	Set the minimum and maximum values used in percentage threshold calculations or leave these field empty for them to be calculated automatically.
Field min/max	Enable **Field min/max** to have Grafana calculate the min or max of each field individually, based on the minimum or maximum value of the field.
Decimals	Specify the number of decimals Grafana includes in the rendered value.
Display name	Set the display title of all fields. You can use variables in the field title.
Color scheme	Set single or multiple colors for your entire visualization.
No value	Enter what Grafana should display if the field value is empty or null. The default value is a hyphen (-).

To learn more, refer to Configure standard options.

Data links

Data links allow you to link to other panels, dashboards, and external resources while maintaining the context of the source panel. You can create links that include the series name or even the value under the cursor.

For each data link, set the following options:

- **Title**
- **URL**
- **Open in new tab**

To learn more, refer to Configure data links.

Value mappings

Value mapping is a technique you can use to change how data appears in a visualization.

For each value mapping, set the following options:

- **Condition** - Choose what's mapped to the display text and (optionally) color:
  - **Value** - Specific values
  - **Range** - Numerical ranges

    - **Regex** - Regular expressions
    - **Special** - Special values like `Null`, `NaN` (not a number), or boolean values like `true` and `false`
- **Display text**
- **Color** (Optional)
- **Icon** (Canvas only)

To learn more, refer to Configure value mappings.

Thresholds

A threshold is a value or limit you set for a metric that's reflected visually when it's met or exceeded. Thresholds are one way you can conditionally style and color your visualizations based on query results.

Set the following options:

- **Value** - Set the value for each threshold.
- **Thresholds mode** - Choose from:
    - **Absolute**
    - **Percentage**

To learn more, refer to Configure thresholds.

Field overrides

Overrides allow you to customize visualization settings for specific fields or series. When you add an override rule, it targets a particular set of fields and lets you define multiple options for how that field is displayed.

Choose from the following override options:

Option	Description
Fields with name	Select a field from the list of all available fields.
Field with name matching regex	Specify fields to override with a regular expression.
Fields with type	Select fields by type, such as string, numeric, or time.
Fields returned by query	Select all fields returned by a specific query, such as A, B, or C.
Fields with values	Select all fields returned by your defined reducer condition, such as **Min**, **Max**, **Count**, **Total**.

To learn more, refer to Configure field overrides.

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Text

Text visualizations enable you to directly include text or HTML in your dashboards. This can be used to add contextual information and descriptions or embed complex HTML.

For example, if you want to display important links to your dashboard, you can use a text visualization to add these links:

Give it a try using Grafana Play

With Grafana Play, you can explore and see how it works, learning from practical examples to accelerate your development. This feature can be seen on Text Panel.

Use a text visualization when you need to:

- Add important links or useful annotations.
- Provide instructions or guidance on how to interpret different panels, configure settings, or take specific actions based on the displayed data.
- Announce any scheduled maintenance or downtime that might impact your dashboards.

Panel options

In the **Panel options** section of the panel editor pane, set basic options like panel title and description, as well as panel links. To learn more, refer to Configure panel options.

Text options

Use the following options to refine your text visualization.

Mode

Mode determines how embedded content appears.

> **NOTE**
>
> To allow embedding of iframes and other websites, you need set `allow_embedding = true` in your Grafana `config.ini` or environment variables (depending on your employment).

Markdown

This option formats the content as markdown.

HTML

This setting renders the content as sanitized HTML. If you require more direct control over the output, you can set the disable_sanitize_html flag which enables you to directly enter HTML.

Code

This setting renders content inside a read-only code editor. Select an appropriate language to apply syntax highlighting to the embedded text.

Variables

Variables in the content will be expanded for display.

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Grafana Cloud | Enterprise | Open source

Traces

Traces visualizations let you follow a request as it traverses the services in your infrastructure. The traces visualization displays traces data in a diagram that allows you to easily interpret it.

For more information about traces and how to use them, refer to the following documentation:

- Tracing in Explore
- Tempo data source
- Getting started with Tempo

Screenshot of the trace view

Give it a try using Grafana Play

With Grafana Play, you can explore and see how it works, learning from practical examples to accelerate your development. This feature can be seen on Traces Panel.

Add a panel with tracing visualizations

Once you have tracing data available in your Grafana stack, you can add tracing panels to your Grafana dashboards.

Using a dashboard variable, `traceID`, lets you create a query to show specific traces for a given trace ID. For more information about dashboard variables, refer to the Variables documentation.

Before you begin

To use this procedure, you need:

- A Grafana instance
- A Tempo data source connected to your Grafana instance

Add the traces visualization query

To view and analyze traces data in a dashboard, you need to add the traces visualization to your dashboard and define a query using the panel editor. The query determines the data that is displayed in the visualization. For more information on the panel editor, refer to the Panel editor documentation.

This procedure uses dashboard variables and templates to allow you to enter trace IDs which can then be visualized. You'll use a variable called `traceId` and add it as a template query.

1. From your Grafana stack, create a new dashboard or go to an existing dashboard where you'd like to add traces visualizations.
2. Select **Add visualization** from a new dashboard or select **Add Panel** on an existing dashboard.
3. Search for and select the appropriate tracing data source.
4. In the top-right of the panel editor, select the **Visualizations** tab, search for, and select **Traces**.
5. Under the **Panel options**, enter a **Title** for your trace panel. For more information on the panel editor, refer to the Configure panel options documentation.
6. In the query editor, select the **TraceQL** query type tab.
7. Enter `${traceId}` in the TraceQL query field to create a dashboard variable. This variable is used as the template query.

Add a template query

8. Select **Apply** in the panel editor to add the panel to the dashboard.
9. Go to the dashboard **Settings** and add a new variable called `traceId`, of variable type **Custom**, giving it a label if required. Select **Apply** to add the variable to the dashboard.

Add a Custom variable

10. Verify that the panel works by using a valid trace ID for the data source used for the trace panel and editing the ID in the dashboard variable.

Results of query in trace panel

Add TraceQL with table visualizations

While you can add a trace visualization to a dashboard, having to manually add trace IDs as a dashboard variable is cumbersome. It's more useful to instead be able to use TraceQL queries to search for specific types of traces and then select appropriate traces from matching results.

1. In the same dashboard where you added the trace visualization, select **Add panel** to add a new visualization panel.
2. Select the same trace data source you used in the previous section.
3. In the top-right of the panel editor, select the **Visualizations** tab, search for, and select **Table**.
4. In the query editor, select the **TraceQL** tab.
5. Under the **Panel options**, enter a **Title** for your trace panel.
6. Add an appropriate TraceQL query to search for traces that you would like to visualize in the dashboard. This example uses a simple, static query. You can write the TraceQL query as a template query to take advantage of other dashboard variables, if they exist. This lets you create dynamic queries based on these variables.

Create a dynamic query

When results are returned from a query, the results are rendered in the panel's table.

Results of a returned query in the panel table

Use a variable to add other links to traces

The results in the traces visualization include links to the **Explore** page that renders the trace. You can add other links to traces in the table that fill in the `traceId` dashboard variable when selected, so that the trace is visualized in the same dashboard.

To create a set of data links in the panel, use the following steps:

1. In the right-side menu, under **Data links**, select **Add link**.
2. Add a **Title** for the data link.
3. Find the UUID of the dashboard by looking in your browser's address bar when the full dashboard is being rendered. Because this is a link to a dashboard in the same Grafana stack, only the path of the dashboard is required.

   Unique identifier for the dashboard
4. In the **URL** field, make a self-reference to the dashboard that contains both of the panels. This self-reference uses the value of the selected trace in the table to fill in the dashboard variable. Use the path for the dashboard from the previous step and then fill in the value of `traceId` using the selected results from the TraceQL table. The trace ID is exposed using the `traceID` data field in the returned results, so use that as the value for the dashboard variable.

Edit link and add the Trace link

5. Select **Save** to save the data link.
6. Select **Apply** from the panel editor to apply the panel to the dashboard.
7. Save the dashboard.

You should now see a list of matching traces in the table visualization. While selecting the **TraceID** or **SpanID** fields will give you the option to either open the **Explore** page to visualize the trace or following the data link, selecting any other field (such as **Start time**, **Name** or **Duration**) automatically follows the data link, filling in the `traceId` dashboard variable, and then shows the relevant trace in the trace panel.

Selecting the trace link

Follow the trace link populates the trace ID and displays the traces view

Panel options

In the **Panel options** section of the panel editor pane, set basic options like panel title and description, as well as panel links. To learn more, refer to Configure panel options.

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Trend

Trend visualizations should be used for datasets that have a sequential, numeric X that is not time. Some examples are function graphs, rpm/torque curves, supply/demand relationships, and elevation or heart rate plots along a race course (with x as distance or duration from start).

Trend visualizations support all visual styles and options available in the time series visualization with these exceptions:

- No annotations or time regions
- No shared cursor/crosshair
- No multi-timezone x axis
- No ability to change the dashboard time range via drag-selection

X Field selection

Use this option to select a field that contains increasing numeric values.

Trend x value selection

For example, you could represent engine power and torque versus speed where speed is plotted on the x axis and power and torque are plotted on the y axes.

Trend engine power and torque curves

Panel options

In the **Panel options** section of the panel editor pane, set basic options like panel title and description, as well as panel links. To learn more, refer to Configure panel options.

Standard options

Standard options in the panel editor pane let you change how field data is displayed in your visualizations. When you set a standard option, the change is applied to all fields or series. For more granular control over the display of fields, refer to Configure overrides.

Option	Description
Unit	Choose which unit a field should use.
Min/Max	Set the minimum and maximum values used in percentage threshold calculations or leave these field empty for them to be calculated automatically.
Field min/max	Enable **Field min/max** to have Grafana calculate the min or max of each field individually, based on the minimum or maximum value of the field.
Decimals	Specify the number of decimals Grafana includes in the rendered value.
Display name	Set the display title of all fields. You can use variables in the field title.
Color scheme	Set single or multiple colors for your entire visualization.
No value	Enter what Grafana should display if the field value is empty or null. The default value is a hyphen (-).

To learn more, refer to Configure standard options.

Legend options

Legend options control the series names and statistics that appear under or to the right of the graph. For more information about the legend, refer to Configure a legend.

Option	Description
Visibility	Toggle the switch to turn the legend on or off.
Mode	Use these settings to define how the legend appears in your visualization. **List** displays the legend as a list. This is a default display mode of the legend. **Table** displays the legend as a table.
Placement	Choose where to display the legend. **Bottom** places the legend below the graph. **Right** places the legend to the right of the graph.
Width	Control how wide the legend is when placed on the right side of the visualization. This option is only displayed if you set the legend placement to **Right**.

Option	Description
Values	Choose which of the standard calculations to show in the legend. You can have more than one.

Tooltip options

Tooltip options control the information overlay that appears when you hover over data points in the visualization.

Option	Description
Tooltip mode	When you hover your cursor over the visualization, Grafana can display tooltips. Choose how tooltips behave.
Values sort order	This option controls the order in which values are listed in a tooltip.
Hover proximity	Set the hover proximity (in pixels) to control how close the cursor must be to a data point to trigger the tooltip to display.
Max width	Set the maximum width of the tooltip box.
Max height	Set the maximum height of the tooltip box. The default is 600 pixels.

Tooltip mode

When you hover your cursor over the visualization, Grafana can display tooltips. Choose how tooltips behave.

- **Single -** The hover tooltip shows only a single series, the one that you are hovering over on the visualization.
- **All -** The hover tooltip shows all series in the visualization. Grafana highlights the series that you are hovering over in bold in the series list in the tooltip.
- **Hidden -** Do not display the tooltip when you interact with the visualization.

Use an override to hide individual series from the tooltip.

Values sort order

When you set the **Tooltip mode** to **All**, the **Values sort order** option is displayed. This option controls the order in which values are listed in a tooltip. Choose from the following:

- **None** - Grafana automatically sorts the values displayed in a tooltip.
- **Ascending** - Values in the tooltip are listed from smallest to largest.
- **Descending** - Values in the tooltip are listed from largest to smallest.

Hover proximity

Set the hover proximity (in pixels) to control how close the cursor must be to a data point to trigger the tooltip to display.

Data links

Data links allow you to link to other panels, dashboards, and external resources while maintaining the context of the source panel. You can create links that include the series name or even the value under the cursor.

For each data link, set the following options:

- **Title**
- **URL**
- **Open in new tab**

To learn more, refer to Configure data links.

Thresholds

A threshold is a value or limit you set for a metric that's reflected visually when it's met or exceeded. Thresholds are one way you can conditionally style and color your visualizations based on query results.

For each threshold, set the following options:

Option	Description
Value	Set the value for each threshold.

Option	Description
Thresholds mode	Choose from **Absolute** and **Percentage**.
Show thresholds	Choose from a variety of display options including not displaying thresholds at all.

To learn more, refer to Configure thresholds.

Value mappings

Value mapping is a technique you can use to change how data appears in a visualization.

For each value mapping, set the following options:

- **Condition** - Choose what's mapped to the display text and (optionally) color:
  - **Value** - Specific values
  - **Range** - Numerical ranges
  - **Regex** - Regular expressions
  - **Special** - Special values like `Null`, `NaN` (not a number), or boolean values like `true` and `false`
- **Display text**
- **Color** (Optional)
- **Icon** (Canvas only)

To learn more, refer to Configure value mappings.

Field overrides

Overrides allow you to customize visualization settings for specific fields or series. When you add an override rule, it targets a particular set of fields and lets you define multiple options for how that field is displayed.

Choose from the following override options:

Option	Description
Fields with name	Select a field from the list of all available fields.
Field with name matching regex	Specify fields to override with a regular expression.
Fields with type	Select fields by type, such as string, numeric, or time.
Fields returned by query	Select all fields returned by a specific query, such as A, B, or C.
Fields with values	Select all fields returned by your defined reducer condition, such as **Min**, **Max**, **Count**, **Total**.

To learn more, refer to Configure field overrides.

Panel overview

A Grafana panel is a visual representation of data composed of a query and a visualization. Within panels, you can apply transformations, which process the results of a query before they're passed on for visualization. You can also further customize a panel by formatting data and configuring visualization options.

Each panel has a query editor specific to the data source selected in the panel. The query editor allows you to build a query that returns the data you want to visualize.

Panels offer a wide variety of formatting and styling options, from applying colors based on field values to creating custom units. Each visualization also comes with options specific to it that give you further control over how your data is displayed. Panels can also be dragged, dropped, and resized to rearrange them on the dashboard.

To get started adding panels, ensure that you have configured a data source:

- For details about using data sources, refer to Data sources.
- For more information about managing data sources as an administrator, refer to Data source management.

> **NOTE**
>
> Data source management is only available in Grafana Enterprise and Grafana Cloud.

Panel feature overview

The following image and descriptions highlight the panel features:

1. **Panel title** - You can create your own panel titles or have Grafana create them for you using generative AI features.
2. **Panel description** - You can create your own panel descriptions or have Grafana create them for you using generative AI features
3. **Links** - Add panel links to other dashboards, panels, or external sites.
4. **Panel menu** - In the panel menu, access actions such as **View**, **Edit**, **Inspect**, and **Remove**.
5. **Legend** - Change series colors, y-axis, and series visibility directly from the legend.
6. **Tooltips** - View tooltips to get more information about data points.

Panel menu

To access the panel editor, hover over the top-right corner of any panel. Click the panel menu icon that appears and select **Edit**. The panel menu gives you access to the following actions:

- **View**: View the panel in full screen.
- **Edit**: Open the panel editor to edit panel and visualization options.
- **Share**: Share the panel as a link, embed, or library panel.
- **Explore**: Open the panel in **Explore**, where you can focus on your query.
- **Inspect**: Open the **Inspect** drawer, where you can review the panel data, stats, metadata, JSON, and query.
  - **Data**: Open the **Inspect** drawer in the **Data** tab.
  - **Query**: Open the **Inspect** drawer in the **Query** tab.
  - **Panel JSON**: Open the **Inspect** drawer in the **JSON** tab.
- **Extensions**: Access other actions provided by installed applications, such as declaring an incident. Note that this option doesn't appear unless you have app plugins installed which contribute an extension to the panel menu.
- **More**: Access other panel actions.

- **Duplicate**: Make a copy of the panel. Duplicated panels query data separately from the original panel. You can use the special `Dashboard` data source to share the same query results across panels instead.
- **Copy**: Copy the panel to the clipboard.
- **Create library panel**: Create a panel that can be imported into other dashboards.
- **Create alert**: Open the alert rule configuration page in **Alerting**, where you can create a Grafana-managed alert based on the panel queries.
- **Hide legend**: Hide the panel legend.
- **Get help**: Send a snapshot or panel data to Grafana Labs Technical Support.

- **Remove**: Remove the panel from the dashboard.

Keyboard shortcuts

Grafana has a number of keyboard shortcuts available specifically for panels. Press `?` on your keyboard to display all keyboard shortcuts available in your version of Grafana.

By hovering over a panel with the mouse you can use some shortcuts that will target that panel.

- `e`: Toggle panel edit view
- `v`: Toggle panel fullscreen view
- `ps`: Open Panel Share Modal
- `pd`: Duplicate Panel
- `pr`: Remove Panel
- `pl`: Toggle panel legend

Add a panel

To add a panel in a new dashboard click **+ Add visualization** in the middle of the dashboard:

To add a panel to an existing dashboard, click **Add** in the dashboard header and select **Visualization** in the drop-down:

Panel configuration

To configure panels, refer to the following subtopics:

- Configure panel options
- Configure standard options
- Configure a legend
- Configure tooltips
- Configure data links
- Configure value mappings
- Configure thresholds
- Configure field overrides

Panel editor overview

In the panel editor, you can update all the elements of a visualization including the data source, queries, time range, and visualization display options.

To add a panel in a new dashboard click **+ Add visualization** in the middle of the dashboard. To add a panel to an existing dashboard, click **Add** in the dashboard header and select **Visualization** in the drop-down:

Panel menu

To access the panel editor, hover over the top-right corner of any panel. Click the panel menu icon that appears and select **Edit**. The panel menu gives you access to the following actions:

- **View**: View the panel in full screen.
- **Edit**: Open the panel editor to edit panel and visualization options.
- **Share**: Share the panel as a link, embed, or library panel.
- **Explore**: Open the panel in **Explore**, where you can focus on your query.
- **Inspect**: Open the **Inspect** drawer, where you can review the panel data, stats, metadata, JSON, and query.
  - **Data**: Open the **Inspect** drawer in the **Data** tab.
  - **Query**: Open the **Inspect** drawer in the **Query** tab.
  - **Panel JSON**: Open the **Inspect** drawer in the **JSON** tab.
- **Extensions**: Access other actions provided by installed applications, such as declaring an incident. Note that this option doesn't appear unless you have app plugins installed which contribute an extension to the panel menu.
- **More**: Access other panel actions.
  - **Duplicate**: Make a copy of the panel. Duplicated panels query data separately from the original panel. You can use the special `Dashboard` data source to share the same query results across panels instead.
  - **Copy**: Copy the panel to the clipboard.
  - **Create library panel**: Create a panel that can be imported into other dashboards.
  - **Create alert**: Open the alert rule configuration page in **Alerting**, where you can [create a Grafana-managed alert] based on the panel queries.
  - **Hide legend**: Hide the panel legend.
  - **Get help**: Send a snapshot or panel data to Grafana Labs Technical Support.
- **Remove**: Remove the panel from the dashboard.

Panel editor

This section describes the areas of the Grafana panel editor.

1. Panel header: The header section lists the dashboard in which the panel appears and the following controls:
   - **Discard:** Discards changes you have made to the panel since you last saved the dashboard.
   - **Save:** Saves changes you made to the panel.
   - **Apply:** Applies changes you made and closes the panel editor, returning you to the dashboard. You'll have to save the dashboard to persist the applied changes.
2. Visualization preview: The visualization preview section contains the following options:
   - **Table view:** Convert any visualization to a table so you can see the data. Table views are helpful for troubleshooting. This view only contains the raw data. It doesn't include transformations you might have applied to the data or the formatting options available in the Table visualization.
   - **Fill:** The visualization preview fills the available space. If you change the width of the side pane or height of the bottom pane the visualization changes to fill the available space.
   - **Actual:** The visualization preview has the exact size as the size on the dashboard. If not enough space is available, the visualization scales down preserving the aspect ratio.
   - **Time range controls: Default** is either the browser local timezone or the timezone selected at a higher level.
3. Data section: The data section contains tabs where you enter queries, transform your data, and create alert rules (if applicable).
   - **Query tab:** Select your data source and enter queries here. For more information, refer to Add a query. When you create a new dashboard, you'll be prompted to select a data source before you get to the panel editor. You set or update the data source in existing dashboards using the drop-down in the **Query** tab.
   - **Transform tab:** Apply data transformations. For more information, refer to Transform data.
   - **Alert tab:** Write alert rules. For more information, refer to the overview of Grafana Alerting.
4. Panel display options: The display options section contains tabs where you configure almost every aspect of your data visualization.

Panel inspect drawer

The inspect drawer helps you understand and troubleshoot your panels. You can view the raw data for any panel, export that data to a comma-separated values (CSV) file, view query requests, and export panel and data JSON.

To access the panel inspect drawer from the edit view, hover over any part of the panel to display the actions menu on the top right corner. Click the menu and select **Inspect**.

> **NOTE**

Not all panel types include all tabs. For example, dashboard list panels don't have raw data to inspect, so they don't display the Stats, Data, or Query tabs.

The panel inspector consists of the following options:

1. The panel inspect drawer displays opens a drawer on the right side. Click the arrow in the upper right corner to expand or reduce the drawer pane.
2. **Data tab -** Shows the raw data returned by the query with transformations applied. Field options such as overrides and value mappings aren't applied by default.
3. **Stats tab -** Shows how long your query takes and how much it returns.
4. **JSON tab -** Allows you to view and copy the panel JSON, panel data JSON, and data frame structure JSON. This is useful if you are provisioning or administering Grafana.
5. **Query tab -** Shows you the requests to the server sent when Grafana queries the data source.
6. **Error tab -** Shows the error. Only visible when query returns error.

The panel inspect view

The panel inspect view, which you can open via the panel menu, helps you understand and troubleshoot your panels. You can inspect the raw data for any Grafana panel, export that data to a comma-separated values (CSV) file, view query requests, and export panel and data JSON.

Note: Not all panel types include all tabs. For example, dashboard list panels do not have raw data to inspect, so they do not display the Stats, Data, or Query tabs.

The panel inspector consists of the following options:

1. The panel inspector displays Inspect: at the top of the pane. Click the arrow in the upper right corner to expand or reduce the pane.
2. **Data tab -** Shows the raw data returned by the query with transformations applied. Field options such as overrides and value mappings are not applied by default.
3. **Stats tab -** Shows how long your query takes and how much it returns.
4. **JSON tab -** Allows you to view and copy the panel JSON, panel data JSON, and data frame structure JSON. This is useful if you are provisioning or administering Grafana.
5. **Query tab -** Shows you the requests to the server sent when Grafana queries the data source.
6. **Error tab -** Shows the error. Only visible when query returns error.

Download raw query results

Grafana generates a CSV file that contains your data, including any transformations to that data. You can choose to view the data before or after the panel applies field options or field option overrides.

1. Edit the panel that contains the query data you want to download.
2. In the query editor, click **Query Inspector**.
3. Click **Data**.

   If your panel contains multiple queries or queries multiple nodes, then you have additional options.

   - **Select result**: Choose which result set data you want to view.
   - **Transform data**
   - **Join by time**: View raw data from all your queries at once, one result set per column. Click a column heading to reorder the data.
4. To see data before the system applies field overrides, click the **Formatted data** toggle.
5. To download a CSV file specifically formatted for Excel, click the **Download for Excel** toggle .
6. Click **Download CSV**.

Inspect query performance

The **Stats** tab displays statistics that tell you how long your query takes, how many queries you send, and the number of rows returned. This information can help you troubleshoot your queries, especially if any of the numbers are unexpectedly high or low.

1. Edit the panel that contains the query with performance you want to inspect.
2. In the query editor, click **Query Inspector**.
3. Click **Stats**.

Statistics are displayed in read-only format.

Inspect query request and response data

Inspect query request and response data when you want to troubleshoot a query that returns unexpected results, or fails to return expected results.

1. Edit the panel that contains the query you want to export.
2. In the query editor, click **Query Inspector**.
3. Click **Refresh**.

   The panel populates with response data.
4. Make adjustments, as necessary and re-run the query.
5. To download the query request and response data, click the **Copy to clipboard** icon and paste the results into another application.

Query and transform data

Grafana supports many types of data sources. Data source **queries** return data that Grafana can **transform** and visualize. Each data source uses its own query language, and data source plugins each implement a query-building user interface called a query editor.

About queries

Grafana panels communicate with data sources via queries, which retrieve data for the visualization. A query is a question written in the query language used by the data source.

You can configure query frequency and data collection limits in the panel's data source options. Grafana supports up to 26 queries per panel.

Important: You **must** be familiar with a data source's query language. For more information, refer to Data sources.

Query editors

Each data source's **query editor** provides a customized user interface that helps you write queries that take advantage of its unique capabilities.

Because of the differences between query languages, each data source query editor looks and functions differently. Depending on your data source, the query editor might provide auto-completion features, metric names, variable suggestions, or a visual query-building interface.

For example, this video demonstrates the visual Prometheus query builder:

For details on a specific data source's unique query editor features, refer to its documentation:

- For data sources included with Grafana, refer to Built-in core data sources, which links to each core data source's documentation.
- For data sources installed as plugins, refer to its own documentation.
  - Data source plugins in Grafana's plugin catalog link to or include their documentation in their catalog listings. For details about the plugin catalog, refer to Plugin management.
  - For links to Grafana Enterprise data source plugin documentation, refer to the Enterprise plugins index.

Query syntax

Each data source uses a different query languages to request data. For details on a specific data source's unique query language, refer to its documentation.

PostgreSQL example:

```
SELECT hostname FROM host WHERE region IN($region)
```

Copy

PromQL example:

```
query_result(max_over_time(<metric>[${__range_s}s]) != <state>)
```

Copy

Special data sources

Grafana also includes three special data sources: **Grafana**, **Mixed**, and **Dashboard**. For details, refer to Data sources

Navigate the Query tab

A panel's Query tab consists of the following elements:

- **Data source selector:** Selects the data source to query. For more information about data sources, refer to Data sources.
- **Query options:** Sets maximum data retrieval parameters and query execution time intervals.
- **Query inspector button:** Opens the query inspector panel, where you can view and optimize your query.
- **Query editor list:** Lists the queries you've written.
- **Expressions:** Uses the expression builder to create alert expressions. For more information about expressions, refer to Use expressions to manipulate data.

Add a query

A query returns data that Grafana visualizes in dashboard panels. When you create a panel, Grafana automatically selects the default data source.

To add a query:

1. Edit the panel to which you're adding a query.
2. Click the **Query** tab.
3. Click the **Data source** drop-down menu and select a data source.

   If you're creating a new dashboard, you'll be prompted to select a data source when you add the first panel.
4. Click **Query options** to configure the maximum number of data points you need. For more information about query options, refer to Query options.
5. Write the query using the query editor.

6 Click **Apply**.

Grafana queries the data source and visualizes the data.

Manage queries

Grafana organizes queries in collapsible query rows. Each query row contains a query editor and is identified with a letter (A, B, C, and so on).

You can:

Icon	Description
	Toggles query editor help. If supported by the data source, click this icon to display information on how to use the query editor or provide quick access to common queries.
	Copies a query. Duplicating queries is useful when working with multiple complex queries that are similar and you want to either experiment with different variants or do minor alterations.
	Hides a query. Grafana does not send hidden queries to the data source.
	Removes a query. Removing a query permanently deletes it, but sometimes you can recover deleted queries by reverting to previously saved versions of the panel.
	Reorders queries. Change the order of queries by clicking and holding the drag icon, then drag queries where desired. The order of results reflects the order of the queries, so you can often adjust your visual results based on query order.

Query options

Click **Query options** next to the data source selector to see settings for the selected data source. Changes you make here affect only queries made in this panel.

Grafana sets defaults that are shown in dark gray text. Changes are displayed in white text. To return a field to the default setting, delete the white text from the field.

Panel data source query options include:

- **Max data points:** If the data source supports it, this sets the maximum number of data points for each series returned. If the query returns more data points than the max data points setting, then the data source reduces the number of points returned by aggregating them together by average, max, or another function.

  You can limit the number of points to improve query performance or smooth the visualized line. The default value is the width (or number of pixels) of the graph, because you can only visualize as many data points as the graph panel has room to display.

  With streaming data, Grafana uses the max data points value for the rolling buffer. Streaming is a continuous flow of data, and buffering divides the stream into chunks. For example, Loki streams data in its live tailing mode.

- **Min interval:** Sets a minimum limit for the automatically calculated interval, which is typically the minimum scrape interval. If a data point is saved every 15 seconds, you don't benefit from having an interval lower than that. You can also set this to a higher minimum than the scrape interval to retrieve queries that are more coarse-grained and well-functioning.

  > **NOTE**
  >
  > The **Min interval** corresponds to the min step in Prometheus. Changing the Prometheus interval can change the start and end of the query range because Prometheus aligns the range to the interval. Refer to Min step for more details.

- **Interval:** Sets a time span that you can use when aggregating or grouping data points by time.

  Grafana automatically calculates an appropriate interval that you can use as a variable in templated queries. The variable is measured in either seconds (`$__interval`) or milliseconds (`$__interval_ms`).

  Intervals are typically used in aggregation functions like sum or average. For example, this is a Prometheus query that uses the interval variable: `rate(http_requests_total[$__interval])`.

  This automatic interval is calculated based on the width of the graph. As the user zooms out on a visualization, the interval grows, resulting in a more coarse-grained aggregation. Likewise, if the user zooms in, the interval decreases, resulting in a more fine-grained aggregation.

For more information, refer to Global variables.

- **Relative time:** Overrides the relative time range for individual panels, which causes them to be different than what is selected in the dashboard time picker in the top-right corner of the dashboard. You can use this to show metrics from different time periods or days on the same dashboard.

  Note: Panel time overrides have no effect when the dashboard's time range is absolute.

Example	Relative time field
Last 5 minutes	`now-5m`
The day so far	`now/d`
Last 5 days	`now-5d/d`
This week so far	`now/w`
Last 2 years	`now-2y/y`

Give it a try using Grafana Play

With Grafana Play, you can explore and see how it works, learning from practical examples to accelerate your development. This feature can be seen on Time range override.

- **Time shift:** Overrides the time range for individual panels by shifting its start and end relative to the time picker. For example, you can shift the time range for the panel to be two hours earlier than the dashboard time picker.

  Note: Panel time overrides have no effect when the dashboard's time range is absolute.

Example	Time shift field
Last entire week	`1w/w`
Two entire weeks ago	`2w/w`
Last entire month	`1M/M`

Example	Time shift field
This entire year	`1d/y`
Last entire year	`1y/y`

- **Cache timeout:** *(Visible only if available in the data source)* Overrides the default cache timeout if your time series store has a query cache. Specify this value as a numeric value in seconds.

Write expression queries

Server-side expressions enable you to manipulate data returned from queries with math and other operations. Expressions create new data and do not manipulate the data returned by data sources.

About expressions

Server-side expressions allow you to manipulate data returned from queries with math and other operations. Expressions create new data and do not manipulate the data returned by data sources, aside from some minor data restructuring to make the data acceptable input for expressions.

Using expressions

Expressions are most commonly used for Grafana Alerting. The processing is done server-side, so expressions can operate without a browser session. However, expressions can also be used with backend data sources and visualization.

NOTE

Expressions do not work with legacy dashboard alerts.

Expressions are meant to augment data sources by enabling queries from different data sources to be combined or by providing operations unavailable in a data source.

NOTE

When possible, you should do data processing inside the data source. Copying data from storage to the Grafana server for processing is inefficient, so expressions are targeted at lightweight data processing.

Expressions work with data source queries that return time series or number data. They also operate on multiple-dimensional data. For example, a query that returns multiple series, where each series is identified by labels or tags.

An individual expression takes one or more queries or other expressions as input and adds data to the result. Each individual expression or query is represented by a variable that is a named identifier known as its RefID (e.g., the default letter `A` or `B`).

To reference the output of an individual expression or a data source query in another expression, this identifier is used as a variable.

Types of expressions

Expressions work with two types of data.

- A collection of time series.
- A collection of numbers, where each number is an item.

Each collection is returned from a single data source query or expression and represented by the RefID. Each collection is a set, where each item in the set is uniquely identified by its dimensions which are stored as labels or key-value pairs.

Data source queries

Server-side expressions only support data source queries for backend data sources. The data is generally assumed to be labeled time series data. In the future we intend to add an assertion of the query return type (number or time series) data so expressions can handle errors better.

Data source queries, when used with expressions, are executed by the expression engine. When it does this, it restructures data to be either one time series or one number per data frame. So for example if using a data source that returns multiple series on one frame in the table view, you might notice it looks different when executed with expressions.

Currently, the only non-time series format (number) is supported when you're using data frames and you have a table response that returns a data frame with no time, string columns, and one number column:

Loc	Host	Avg_CPU
MIA	A	1
NYC	B	2

The example above will produce a number that works with expressions. The string columns become labels and the number column the corresponding value. For example `{"Loc": "MIA", "Host": "A"}` with a value of 1.

Operations

You can use the following operations in expressions: math, reduce, and resample.

Math

Math is for free-form math formulas on time series or number data. Math operations take numbers and time series as input and change them to different numbers and time series.

Data from other queries or expressions are referenced with the RefID prefixed with a dollar sign, for example `$A`. If the variable has spaces in the name, then you can use a brace syntax like `${my variable}`.

Numeric constants may be in decimal (`2.24`), octal (with a leading zero like `072`), or hex (with a leading 0x like `0x2A`). Exponentials and signs are also supported (e.g., `-0.8e-2`).

Operators

The arithmetic (`+`, binary and unary `-`, `*`, `/`, `%`, exponent `**`), relational (`<`, `>`, `==`, `!=`, `>=`, `<=`), and logical (`&&`, `||`, and unary `!`) operators are supported.

How the operation behaves with data depends on if it is a number or time series data.

With binary operations, such as `$A + $B` or `$A || $B`, the operator is applied in the following ways depending on the type of data:

- If both `$A` and `$B` are a number, then the operation is performed between the two numbers.
- If one variable is a number, and the other variable is a time series, then the operation between the value of each point in the time series and the number is performed.
- If both `$A` and `$B` are time series data, then the operation between each value in the two series is performed for each time stamp that exists in both `$A` and `$B`. The Resample operation can be used to line up time stamps. (**Note:** in the future, we plan to add options to the Math operation for different behaviors).

Summary:

- Number OP number = number
- Number OP series = series
- Series OP series = series

Because expressions work with multiple series or numbers represented by a single variable, binary operations also perform a union (join) between the two variables. This is done based on the identifying labels associated with each individual series or number.

So if you have numbers with labels like `{host=web01}` in `$A` and another number in `$B` with the same labels then the operation is performed between those two items within each variable, and the result will share the same labels. The rules for the behavior of this union are as follows:

- An item with no labels will join to anything.
- If both `$A` and `$B` each contain only one item (one series, or one number), they will join.

- If labels are exact match they will join.
- If labels are a subset of the other, for example and item in `$A` is labeled `{host=A,dc=MIA}` and item in `$B` is labeled `{host=A}` they will join.
- Currently, if within a variable such as `$A` there are different tag *keys* for each item, the join behavior is undefined.

The relational and logical operators return 0 for false 1 for true.

Math Functions

While most functions exist in the own expression operations, the math operation does have some functions similar to math operators or symbols. When functions can take either numbers or series, than the same type as the argument will be returned. When it is a series, the operation of performed for the value of each point in the series.

abs

abs returns the absolute value of its argument which can be a number or a series. For example `abs(-1)` or `abs($A)`.

is_inf

is_inf takes a number or a series and returns `1` for `Inf` values (negative or positive) and `0` for other values. For example `is_inf($A)`.

> **NOTE**
>
> If you need to specifically check for negative infinity for example, you can do a comparison like `$A == infn()`.

is_nan

is_nan takes a number or a series and returns `1` for `NaN` values and `0` for other values. For example `is_nan($A)`. This function exists because `NaN` is not equal to `NaN`.

is_null

is_null takes a number or a series and returns `1` for `null` values and `0` for other values. For example `is_null($A)`.

is_number

is_number takes a number or a series and returns `1` for all real number values and `0` for other values (which are `null`, `Inf+`, `Inf-`, and `NaN`). For example `is_number($A)`.

log

Log returns the natural logarithm of of its argument which can be a number or a series. If the value is less than 0, NaN is returned. For example `log(-1)` or `log($A)`.

inf, infn, nan, and null

The inf, infn, nan, and null functions all return a single value of the name. They primarily exist for testing. Example: `null()`.

round

Round returns a rounded integer value. For example, `round(3.123)` or `round($A)`. (This function should probably take an argument so it can add precision to the rounded value).

ceil

Ceil rounds the number up to the nearest integer value. For example, `ceil(3.123)` returns 4.

floor

Floor rounds the number down to the nearest integer value. For example, `floor(3.123)` returns 3.

Reduce

Reduce takes one or more time series returned from a query or an expression and turns each series into a single number. The labels of the time series are kept as labels on each outputted reduced number.

Fields:

- **Function -** The reduction function to use
- **Input -** The variable (refID (such as `A`)) to resample
- **Mode -** Allows control behavior of reduction function when a series contains non-numerical values (null, NaN, +-Inf)

Reduction Functions

Count

Count returns the number of points in each series.

Mean

Mean returns the total of all values in each series divided by the number of points in that series. In `strict` mode if any values in the series are null or nan, or if the series is empty, NaN is returned.

Min and Max

Min and Max return the smallest or largest value in the series respectively. In `strict` mode if any values in the series are null or nan, or if the series is empty, NaN is returned.

Sum

Sum returns the total of all values in the series. If series is of zero length, the sum will be 0. In `strict` mode if there are any NaN or Null values in the series, NaN is returned.

Last

Last returns the last number in the series. If the series has no values then returns NaN.

Reduction Modes

Strict

In Strict mode the input series is processed as is. If any values in the series are non-numeric (null, NaN or +-Inf), NaN is returned.

Drop Non-Numeric

In this mode all non-numeric values (null, NaN or +-Inf) in the input series are filtered out before executing the reduction function.

Replace Non-Numeric

In this mode all non-numeric values are replaced by a pre-defined value.

Resample

Resample changes the time stamps in each time series to have a consistent time interval. The main use case is so you can resample time series that do not share the same timestamps so math can be performed between them. This can be done by resample each of the two series, and then in a Math operation referencing the resampled variables.

Fields:

- **Input -** The variable of time series data (refID (such as `A`)) to resample
- **Resample to -** The duration of time to resample to, for example `10s`. Units may be `s` seconds, `m` for minutes, `h` for hours, `d` for days, `w` for weeks, and `y` of years.
- **Downsample -** The reduction function to use when there are more than one data point per window sample. See the reduction operation for behavior details.
- **Upsample -** The method to use to fill a window sample that has no data points.
  - **pad** fills with the last know value
  - **backfill** with next known value
  - **fillna** to fill empty sample windows with NaNs

Write an expression

If your data source supports them, then Grafana displays the **Expression** button and shows any existing expressions in the query editor list.

For more information about expressions, refer to About expressions.

1. Open the panel.
2. Below the query, click **Expression**.
3. In the **Operation** field, select the type of expression you want to write.

   For more information about expression operations, refer to About expressions.
4. Write the expression.
5. Click **Apply**.

Special cases

When any queried data source returns no series or numbers, the expression engine returns `NoData`. For example, if a request contains two data source queries that are merged by an expression, if `NoData` is returned by at least one of the data source queries, then the returned result for the entire query is `NoData`.

For more information about how Grafana Alerting processes `NoData` results, refer to No data and error handling.

In the case of using an expression on multiple queries, the expression engine requires that all of the queries return an identical timestamp. For example, if using math to combine the results of multiple SQL queries which each use `SELECT NOW() AS "time"`, the expression will only work if all queries evaluate `NOW()` to an identical timestamp; which does not always happen. To resolve this, you can replace `NOW()` with an arbitrary time, such as `SELECT 1 AS "time"`, or any other valid UNIX timestamp.

Share query results with another panel

Grafana lets you use the query result from one panel for any other panel in the dashboard. Sharing query results across panels reduces the number of queries made to your data source, which can improve the performance of your dashboard.

The Dashboard data source lets you select a panel in your dashboard that contains the queries you want to share the results for. Instead of sending a separate query for each panel, Grafana sends one query and other panels use the query results to construct visualizations.

This strategy can drastically reduce the number of queries being made when you for example have several panels visualizing the same data.

1. Create a dashboard.
2. Create a panel.
3. Change the panel title to “Source panel”. You’ll use this panel as a source for the other panels.
4. Define the query or queries that you want share.

   If you don’t have a data source available, use the **Grafana** data source, which returns a random time series that you can use for testing.
5. Add a new panel and select the **Dashboard** data source in the query editor.
6. In the **Use results from panel list**, select the first panel you created.

All queries defined in the source panel are now available to the new panel. Queries defined in the source panel can be shared with multiple panels.

You can click on any of the queries to go to the panel where they are defined.

Transform data

Transformations are a powerful way to manipulate data returned by a query before the system applies a visualization. Using transformations, you can:

- Rename fields
- Join time series data
- Perform mathematical operations across queries
- Use the output of one transformation as the input to another transformation

For users that rely on multiple views of the same dataset, transformations offer an efficient method of creating and maintaining numerous dashboards.

You can also use the output of one transformation as the input to another transformation, which results in a performance gain.

Sometimes the system cannot graph transformed data. When that happens, click the `Table view` toggle above the visualization to switch to a table view of the data. This can help you understand the final result of your transformations.

Transformation types

Grafana provides a number of ways that you can transform data. For a complete list of transformations, refer to Transformation functions.

Order of transformations

When there are multiple transformations, Grafana applies them in the order they are listed. Each transformation creates a result set that then passes on to the next transformation in the

processing pipeline.

The order in which Grafana applies transformations directly impacts the results. For example, if you use a Reduce transformation to condense all the results of one column into a single value, then you can only apply transformations to that single value.

Add a transformation function to data

The following steps guide you in adding a transformation to data. This documentation does not include steps for each type of transformation. For a complete list of transformations, refer to Transformation functions.

1. Navigate to the panel where you want to add one or more transformations.
2. Hover over any part of the panel to display the actions menu on the top right corner.
3. Click the menu and select **Edit**.
4. Click the **Transform** tab.
5. Click a transformation. A transformation row appears where you configure the transformation options. For more information about how to configure a transformation, refer to Transformation functions. For information about available calculations, refer to Calculation types.
6. To apply another transformation, click **Add transformation**. This transformation acts on the result set returned by the previous transformation.

Debug a transformation

To see the input and the output result sets of the transformation, click the bug icon on the right side of the transformation row.

The input and output results sets can help you debug a transformation.

Disable a transformation

You can disable or hide one or more transformations by clicking on the eye icon on the top right side of the transformation row. This disables the applied actions of that specific transformation and can help to identify issues when you change several transformations one after another.

Filter a transformation

If your panel uses more than one query, you can filter these and apply the selected transformation to only one of the queries. To do this, click the filter icon on the top right of the transformation row. This opens a drop-down with a list of queries used on the panel. From here, you can select the query you want to transform.

You can also filter by annotations (which includes exemplars) to apply transformations to them. When you do so, the list of fields changes to reflect those in the annotation or exemplar tooltip.

The filter icon is always displayed if your panel has more than one query or source of data (that is, panel or annotation data) but it may not work if previous transformations for merging the queries' outputs are applied. This is because one transformation takes the output of the previous one.

Delete a transformation

We recommend that you remove transformations that you don't need. When you delete a transformation, you remove the data from the visualization.

Before you begin:

- Identify all dashboards that rely on the transformation and inform impacted dashboard users.

To delete a transformation:

1. Open a panel for editing.
2. Click the **Transform** tab.
3. Click the trash icon next to the transformation you want to delete.

Transformation functions

You can perform the following transformations on your data.

Add field from calculation

Use this transformation to add a new field calculated from two other fields. Each transformation allows you to add one new field.

- **Mode** - Select a mode:
  - **Reduce row** - Apply selected calculation on each row of selected fields independently.
  - **Binary operation** - Apply basic binary operations (for example, sum or multiply) on values in a single row from two selected fields.
  - **Unary operation** - Apply basic unary operations on values in a single row from a selected field. The available operations are:
    - **Absolute value (abs)** - Returns the absolute value of a given expression. It represents its distance from zero as a positive number.
    - **Natural exponential (exp)** - Returns *e* raised to the power of a given expression.
    - **Natural logarithm (ln)** - Returns the natural logarithm of a given expression.
    - **Floor (floor)** - Returns the largest integer less than or equal to a given expression.
    - **Ceiling (ceil)** - Returns the smallest integer greater than or equal to a given expression.
  - **Cumulative functions** - Apply functions on the current row and all preceding rows.
    - **Total** - Calculates the cumulative total up to and including the current row.
    - **Mean** - Calculates the mean up to and including the current row.
  - **Window functions** - Apply window functions. The window can either be **trailing** or **centered**. With a trailing window the current row will be the last row in the window. With a centered window the window will be centered on the current row. For even window sizes, the window will be centered between the current row, and the previous row.
    - **Mean** - Calculates the moving mean or running average.

    - **Stddev** - Calculates the moving standard deviation.
    - **Variance** - Calculates the moving variance.
  - **Row index** - Insert a field with the row index.
- **Field name** - Select the names of fields you want to use in the calculation for the new field.
- **Calculation** - If you select **Reduce row** mode, then the **Calculation** field appears. Click in the field to see a list of calculation choices you can use to create the new field. For information about available calculations, refer to Calculation types.
- **Operation** - If you select **Binary operation** or **Unary operation** mode, then the **Operation** fields appear. These fields allow you to apply basic math operations on values in a single row from selected fields. You can also use numerical values for binary operations.
- **As percentile** - If you select **Row index** mode, then the **As percentile** switch appears. This switch allows you to transform the row index as a percentage of the total number of rows.
- **Alias** - (Optional) Enter the name of your new field. If you leave this blank, then the field will be named to match the calculation.
- **Replace all fields** - (Optional) Select this option if you want to hide all other fields and display only your calculated field in the visualization.

Note: Cumulative functions and **Window functions** modes are currently in public preview. Grafana Labs offers limited support, and breaking changes might occur prior to the feature being made generally available. Enable the `addFieldFromCalculationStatFunctions` feature toggle in Grafana to use this feature. Contact Grafana Support to enable this feature in Grafana Cloud.

In the example below, we added two fields together and named them Sum.

Concatenate fields

Use this transformation to combine all fields from all frames into one result.

For example, if you have separate queries retrieving temperature and uptime data (Query A) and air quality index and error information (Query B), applying the concatenate transformation yields a consolidated data frame with all relevant information in one view.

Consider the following:

Query A:

Temp	Uptime
15.4	1230233

Query B:

AQI	Errors
3.2	5

After you concatenate the fields, the data frame would be:

Temp	Uptime	AQI	Errors
15.4	1230233	3.2	5

This transformation simplifies the process of merging data from different sources, providing a comprehensive view for analysis and visualization.

Config from query results

Use this transformation to select a query and extract standard options, such as **Min**, **Max**, **Unit**, and **Thresholds**, and apply them to other query results. This feature enables dynamic visualization configuration based on the data returned by a specific query.

Options

- **Config query** - Select the query that returns the data you want to use as configuration.
- **Apply to** - Select the fields or series to which the configuration should be applied.

- **Apply to options** - Specify a field type or use a field name regex, depending on your selection in **Apply to**.

Field mapping table

Below the configuration options, you'll find the field mapping table. This table lists all fields found in the data returned by the config query, along with **Use as** and **Select** options. It provides control over mapping fields to config properties, and for multiple rows, it allows you to choose which value to select.

Example

Input[0] (From query: A, name: ServerA)

Time	Value
1626178119127	10
1626178119129	30

Input[1] (From query: B)

Time	Value
1626178119127	100
1626178119129	100

Output (Same as Input[0] but now with config on the Value field)

Time	Value (config: Max=100)
1626178119127	10
1626178119129	30

Each row in the source data becomes a separate field. Each field now has a maximum configuration option set. Options such as **Min**, **Max**, **Unit**, and **Thresholds** are part of the field configuration. If set, they are used by the visualization instead of any options manually configured in the panel editor options pane.

Value mappings

You can also transform a query result into value mappings. With this option, every row in the configuration query result defines a single value mapping row. See the following example.

Config query result:

Value	Text	Color
L	Low	blue
M	Medium	green
H	High	red

In the field mapping specify:

Field	Use as	Select
Value	Value mappings / Value	All values
Text	Value mappings / Text	All values
Color	Value mappings / Ciolor	All values

Grafana builds value mappings from your query result and applies them to the real data query results. You should see values being mapped and colored according to the config query results.

Convert field type

Use this transformation to modify the field type of a specified field.

This transformation has the following options:

- **Field** - Select from available fields
- **as** - Select the FieldType to convert to
  - **Numeric** - attempts to make the values numbers
  - **String** - will make the values strings
  - **Time** - attempts to parse the values as time
    - The input will be parsed according to the Moment.js parsing format
    - It will parse the numeric input as a Unix epoch timestamp in milliseconds. You must multiply your input by 1000 if it's in seconds.
    - Will show an option to specify a DateFormat as input by a string like yyyy-mm-dd or DD MM YYYY hh:mm:ss
  - **Boolean** - will make the values booleans
  - **Enum** - will make the values enums

    - Will show a table to manage the enums
- **Other** - attempts to parse the values as JSON

For example, consider the following query that could be modified by selecting the time field as Time and specifying Date Format as YYYY.

Sample Query

Time	Mark	Value
2017-07-01	above	25
2018-08-02	below	22
2019-09-02	below	29
2020-10-04	above	22

The result:

Transformed Query

Time	Mark	Value
2017-01-01 00:00:00	above	25
2018-01-01 00:00:00	below	22
2019-01-01 00:00:00	below	29
2020-01-01 00:00:00	above	22

This transformation allows you to flexibly adapt your data types, ensuring compatibility and consistency in your visualizations.

Extract fields

Use this transformation to select a source of data and extract content from it in different formats. This transformation has the following fields:

- **Source** - Select the field for the source of data.
- **Format** - Choose one of the following:
    - **JSON** - Parse JSON content from the source.
    - **Key+value pairs** - Parse content in the format ‘a=b’ or ‘c:d’ from the source.
    - **Auto** - Discover fields automatically.

- **Replace All Fields** - (Optional) Select this option to hide all other fields and display only your calculated field in the visualization.
- **Keep Time** - (Optional) Available only if **Replace All Fields** is true. Keeps the time field in the output.

Consider the following dataset:

Dataset Example

Timestamp	json_data
1636678740000000000	{“value”: 1}
1636678680000000000	{“value”: 5}
1636678620000000000	{“value”: 12}

You could prepare the data to be used by a Time series panel with this configuration:

- Source: json_data
- Format: JSON
  - Field: value
  - Alias: my_value
- Replace all fields: true
- Keep time: true

This will generate the following output:

Transformed Data

Timestamp	my_value
1636678740000000000	1
1636678680000000000	5
1636678620000000000	12

This transformation allows you to extract and format data in various ways. You can customize the extraction format based on your specific data needs.

Lookup fields from resource

Use this transformation to enrich a field value by looking up additional fields from an external source.

This transformation has the following fields:

- **Field** - Select a text field from your dataset.
- **Lookup** - Choose from **Countries**, **USA States**, and **Airports**.

This transformation currently supports spatial data.

For example, if you have this data:

Dataset Example

Location	Values
AL	0
AK	10
Arizona	5
Arkansas	1
Somewhere	5

With this configuration:

- Field: location
- Lookup: USA States

You'll get the following output:

Transformed Data

Location	ID	Name	Lng	Lat	Values
AL	AL	Alabama	-80.891064	12.448457	0
AK	AK	Arkansas	-100.891064	24.448457	10
Arizona					5
Arkansas					1
Somewhere					5

This transformation lets you augment your data by fetching additional information from external sources, providing a more comprehensive dataset for analysis and visualization.

Filter data by query refId

Use this transformation to hide one or more queries in panels that have multiple queries.

Grafana displays the query identification letters in dark gray text. Click a query identifier to toggle filtering. If the query letter is white, then the results are displayed. If the query letter is dark, then the results are hidden.

> **Note:** This transformation is not available for Graphite because this data source does not support correlating returned data with queries.

In the example below, the panel has three queries (A, B, C). We removed the B query from the visualization.

Filter data by values

Use this transformation to selectively filter data points directly within your visualization. This transformation provides options to include or exclude data based on one or more conditions applied to a selected field.

This transformation is very useful if your data source does not natively filter by values. You might also use this to narrow values to display if you are using a shared query.

The available conditions for all fields are:

- **Regex** - Match a regex expression.
- **Is Null** - Match if the value is null.
- **Is Not Null** - Match if the value is not null.
- **Equal** - Match if the value is equal to the specified value.
- **Different** - Match if the value is different than the specified value.

The available conditions for number fields are:

- **Greater** - Match if the value is greater than the specified value.
- **Lower** - Match if the value is lower than the specified value.

- **Greater or equal** - Match if the value is greater or equal.
- **Lower or equal** - Match if the value is lower or equal.
- **Range** - Match a range between a specified minimum and maximum, min and max included.

Consider the following dataset:

Dataset Example

Time	Temperature	Altitude
2020-07-07 11:34:23	32	101
2020-07-07 11:34:22	28	125
2020-07-07 11:34:21	26	110
2020-07-07 11:34:20	23	98
2020-07-07 10:32:24	31	95
2020-07-07 10:31:22	20	85
2020-07-07 09:30:57	19	101

If you **Include** the data points that have a temperature below 30°C, the configuration will look as follows:

- Filter Type: 'Include'
- Condition: Rows where 'Temperature' matches 'Lower Than' '30'

And you will get the following result, where only the temperatures below 30°C are included:

Transformed Data

Time	Temperature	Altitude
2020-07-07 11:34:22	28	125
2020-07-07 11:34:21	26	110
2020-07-07 11:34:20	23	98
2020-07-07 10:31:22	20	85
2020-07-07 09:30:57	19	101

You can add more than one condition to the filter. For example, you might want to include the data only if the altitude is greater than 100. To do so, add that condition to the following configuration:

- Filter type: ‘Include’ rows that ‘Match All’ conditions
- Condition 1: Rows where ‘Temperature’ matches ‘Lower’ than ‘30’
- Condition 2: Rows where ‘Altitude’ matches ‘Greater’ than ‘100’

When you have more than one condition, you can choose if you want the action (include/exclude) to be applied on rows that **Match all** conditions or **Match any** of the conditions you added.

In the example above, we chose **Match all** because we wanted to include the rows that have a temperature lower than 30°C *AND* an altitude higher than 100. If we wanted to include the rows that have a temperature lower than 30°C *OR* an altitude higher than 100 instead, then we would select **Match any**. This would include the first row in the original data, which has a temperature of 32°C (does not match the first condition) but an altitude of 101 (which matches the second condition), so it is included.

Conditions that are invalid or incompletely configured are ignored.

This versatile data filtering transformation lets you to selectively include or exclude data points based on specific conditions. Customize the criteria to tailor your data presentation to meet your unique analytical needs.

Filter fields by name

Use this transformation to selectively remove parts of your query results. There are three ways to filter field names:

- Using a regular expression
- Manually selecting included fields
- Using a dashboard variable

Use a regular expression

When you filter using a regular expression, field names that match the regular expression are included.

For example, from the input data:

Time	dev-eu-west	dev-eu-north	prod-eu-west	prod-eu-north
2023-03-04 23:56:23	23.5	24.5	22.2	20.2
2023-03-04 23:56:23	23.6	24.4	22.1	20.1

The result from using the regular expression ‘prod.*’ would be:

Time	prod-eu-west	prod-eu-north
2023-03-04 23:56:23	22.2	20.2
2023-03-04 23:56:23	22.1	20.1

The regular expression can include an interpolated dashboard variable by using the ${variableName} syntax.

Manually select included fields

Click and uncheck the field names to remove them from the result. Fields that are matched by the regular expression are still included, even if they're unchecked.

Use a dashboard variable

Enable 'From variable' to let you select a dashboard variable that's used to include fields. By setting up a dashboard variable with multiple choices, the same fields can be displayed across multiple visualizations.

Here's the table after we applied the transformation to remove the Min field.

Here is the same query using a Stat visualization.

This transformation provides flexibility in tailoring your query results to focus on the specific fields you need for effective analysis and visualization.

Format string

Use this transformation to customize the output of a string field. This transformation has the following fields:

- **Upper case** - Formats the entire string in uppercase characters.
- **Lower case** - Formats the entire string in lowercase characters.
- **Sentence case** - Formats the first character of the string in uppercase.
- **Title case** - Formats the first character of each word in the string in uppercase.
- **Pascal case** - Formats the first character of each word in the string in uppercase and doesn't include spaces between words.
- **Camel case** - Formats the first character of each word in the string in uppercase, except the first word, and doesn't include spaces between words.
- **Snake case** - Formats all characters in the string in lowercase and uses underscores instead of spaces between words.
- **Kebab case** - Formats all characters in the string in lowercase and uses dashes instead of spaces between words.
- **Trim** - Removes all leading and trailing spaces from the string.
- **Substring** - Returns a substring of the string, using the specified start and end positions.

This transformation provides a convenient way to standardize and tailor the presentation of string data for better visualization and analysis.

Note: This transformation is currently in public preview. Grafana Labs offers limited support, and breaking changes might occur prior to the feature being made generally available. Enable

the `formatString` feature toggle in Grafana to use this feature. Contact Grafana Support to enable this feature in Grafana Cloud.

Format time

Use this transformation to customize the output of a time field. Output can be formatted using Moment.js format strings. For example, if you want to display only the year of a time field, the format string 'YYYY' can be used to show the calendar year (for example, 1999 or 2012).

Before Transformation:

Timestamp	Event
1636678740000000000	System Start
1636678680000000000	User Login
1636678620000000000	Data Updated

After applying 'YYYY-MM-DD HH:mm:ss':

Timestamp	Event
2021-11-12 14:25:40	System Start
2021-11-12 14:24:40	User Login
2021-11-12 14:23:40	Data Updated

This transformation lets you tailor the time representation in your visualizations, providing flexibility and precision in displaying temporal data.

Note: This transformation is available in Grafana 10.1+ as an alpha feature.

Group by

Use this transformation to group the data by a specified field (column) value and process calculations on each group. Click to see a list of calculation choices. For information about available calculations, refer to Calculation types.

Here's an example of original data.

Time	Server ID	CPU Temperature	Server Status
2020-07-07 11:34:20	server 1	80	Shutdown
2020-07-07 11:34:20	server 3	62	OK
2020-07-07 10:32:20	server 2	90	Overload
2020-07-07 10:31:22	server 3	55	OK
2020-07-07 09:30:57	server 3	62	Rebooting
2020-07-07 09:30:05	server 2	88	OK
2020-07-07 09:28:06	server 1	80	OK
2020-07-07 09:25:05	server 2	88	OK
2020-07-07 09:23:07	server 1	86	OK

This transformation goes in two steps. First you specify one or multiple fields to group the data by. This will group all the same values of those fields together, as if you sorted them. For instance if we group by the Server ID field, then it would group the data this way:

Time	Server ID	CPU Temperature	Server Status
2020-07-07 11:34:20	**server 1**	80	Shutdown
2020-07-07 09:28:06	**server 1**	80	OK
2020-07-07 09:23:07	**server 1**	86	OK
2020-07-07 10:32:20	server 2	90	Overload
2020-07-07 09:30:05	server 2	88	OK
2020-07-07 09:25:05	server 2	88	OK
2020-07-07 11:34:20	***server 3***	62	OK
2020-07-07 10:31:22	***server 3***	55	OK
2020-07-07 09:30:57	***server 3***	62	Rebooting

All rows with the same value of Server ID are grouped together.

After choosing which field you want to group your data by, you can add various calculations on the other fields, and apply the calculation to each group of rows. For instance, we could want to calculate the average CPU temperature for each of those servers. So we can add the *mean* calculation applied on the CPU Temperature field to get the following:

Server ID	CPU Temperature (mean)
server 1	82
server 2	88.6
server 3	59.6

And we can add more than one calculation. For instance:

- For field Time, we can calculate the *Last* value, to know when the last data point was received for each server
- For field Server Status, we can calculate the *Last* value to know what is the last state value for each server
- For field Temperature, we can also calculate the *Last* value to know what is the latest monitored temperature for each server

We would then get:

Server ID	CPU Temperature (mean)	CPU Temperature (last)	Time (last)	Server Status (last)
server 1	82	80	2020-07-07 11:34:20	Shutdown
server 2	88.6	90	2020-07-07 10:32:20	Overload
server 3	59.6	62	2020-07-07 11:34:20	OK

This transformation allows you to extract essential information from your time series and present it conveniently.

Grouping to matrix

Use this transformation to combine three fields—which are used as input for the **Column**, **Row**, and **Cell value** fields from the query output—and generate a matrix. The matrix is calculated as follows:

Original data

Server ID	CPU Temperature	Server Status
server 1	82	OK
server 2	88.6	OK
server 3	59.6	Shutdown

We can generate a matrix using the values of ‘Server Status’ as column names, the ‘Server ID’ values as row names, and the ‘CPU Temperature’ as content of each cell. The content of each cell will appear for the existing column (‘Server Status’) and row combination (‘Server ID’). For the rest of the cells, you can select which value to display between: **Null**, **True**, **False**, or **Empty**.

Output

Server IDServer Status	OK	Shutdown
server 1	82	
server 2	88.6	
server 3		59.6

Use this transformation to construct a matrix by specifying fields from your query results. The matrix output reflects the relationships between the unique values in these fields. This helps you present complex relationships in a clear and structured matrix format.

Create heatmap

Use this transformation to prepare histogram data for visualizing trends over time. Similar to the heatmap visualization, this transformation converts histogram metrics into temporal buckets.

X Bucket

This setting determines how the x-axis is split into buckets.

- **Size** - Specify a time interval in the input field. For example, a time range of ‘1h’ creates cells one hour wide on the x-axis.
- **Count** - For non-time-related series, use this option to define the number of elements in a bucket.

Y Bucket

This setting determines how the y-axis is split into buckets.

- **Linear**
- **Logarithmic** - Choose between log base 2 or log base 10.
- **Symlog** - Uses a symmetrical logarithmic scale. Choose between log base 2 or log base 10, allowing for negative values.

Assume you have the following dataset:

Timestamp	Value
2023-01-01 12:00:00	5
2023-01-01 12:15:00	10
2023-01-01 12:30:00	15
2023-01-01 12:45:00	8

- With X Bucket set to ‘Size: 15m’ and Y Bucket as ‘Linear’, the histogram organizes values into time intervals of 15 minutes on the x-axis and linearly on the y-axis.
- For X Bucket as ‘Count: 2’ and Y Bucket as ‘Logarithmic (base 10)’, the histogram groups values into buckets of two on the x-axis and use a logarithmic scale on the y-axis.

Histogram

Use this transformation to generate a histogram based on input data, allowing you to visualize the distribution of values.

- **Bucket size** - The range between the lowest and highest items in a bucket (xMin to xMax).
- **Bucket offset** - The offset for non-zero-based buckets.
- **Combine series** - Create a unified histogram using all available series.

Original data

Series 1:

A	B	C
1	3	5
2	4	6
3	5	7
4	6	8
5	7	9

Series 2:

C
5
6

C
7
8
9

Output

xMin	xMax	A	B	C	C
1	2	1	0	0	0
2	3	1	0	0	0
3	4	1	1	0	0
4	5	1	1	0	0
5	6	1	1	1	1
6	7	0	1	1	1
7	8	0	1	1	1
8	9	0	0	1	1
9	10	0	0	1	1

Visualize the distribution of values using the generated histogram, providing insights into the data's spread and density.

Join by field

Use this transformation to merge multiple results into a single table, enabling the consolidation of data from different queries.

This is especially useful for converting multiple time series results into a single wide table with a shared time field.

Inner join

An inner join merges data from multiple tables where all tables share the same value from the selected field. This type of join excludes data where values do not match in every result.

Use this transformation to combine the results from multiple queries (combining on a passed join field or the first time column) into one result, and drop rows where a successful join cannot occur.

In the following example, two queries return table data. It is visualized as two separate tables before applying the inner join transformation.

Query A:

Time	Job	Uptime
2020-07-07 11:34:20	node	25260122
2020-07-07 11:24:20	postgre	123001233
2020-07-07 11:14:20	postgre	345001233

Query B:

Time	Server	Errors
2020-07-07 11:34:20	server 1	15
2020-07-07 11:24:20	server 2	5
2020-07-07 11:04:20	server 3	10

The result after applying the inner join transformation looks like the following:

Time	Job	Uptime	Server	Errors
2020-07-07 11:34:20	node	25260122	server 1	15
2020-07-07 11:24:20	postgre	123001233	server 2	5

Outer join

An outer join includes all data from an inner join and rows where values do not match in every input. While the inner join joins Query A and Query B on the time field, the outer join includes all rows that don't match on the time field.

In the following example, two queries return table data. It is visualized as two tables before applying the outer join transformation.

Query A:

Time	Job	Uptime
2020-07-07 11:34:20	node	25260122
2020-07-07 11:24:20	postgre	123001233
2020-07-07 11:14:20	postgre	345001233

Query B:

Time	Server	Errors
2020-07-07 11:34:20	server 1	15
2020-07-07 11:24:20	server 2	5
2020-07-07 11:04:20	server 3	10

The result after applying the outer join transformation looks like the following:

Time	Job	Uptime	Server	Errors
2020-07-07 11:04:20			server 3	10
2020-07-07 11:14:20	postgre	345001233		
2020-07-07 11:34:20	node	25260122	server 1	15
2020-07-07 11:24:20	postgre	123001233	server 2	5

In the following example, a template query displays time series data from multiple servers in a table visualization. The results of only one query can be viewed at a time.

I applied a transformation to join the query results using the time field. Now I can run calculations, combine, and organize the results in this new table.

Combine and analyze data from various queries with table joining for a comprehensive view of your information.

Join by labels

Use this transformation to join multiple results into a single table.

This is especially useful for converting multiple time series results into a single wide table with a shared **Label** field.

- **Join** - Select the label to join by between the labels available or common across all time series.
- **Value** - The name for the output result.

Example

Input

series1{what="Temp", cluster="A", job="J1"}

Time	Value
1	10
2	200

series2{what="Temp", cluster="B", job="J1"}

Time	Value
1	10
2	200

series3{what="Speed", cluster="B", job="J1"}

Time	Value
22	22
28	77

Config

value: "what"

Output

cluster	job	Temp	Speed
A	J1	10	
A	J1	200	
B	J1	10	22
B	J1	200	77

Combine and organize time series data effectively with this transformation for comprehensive insights.

Labels to fields

Use this transformation to convert time series results with labels or tags into a table, including each label's keys and values in the result. Display labels as either columns or row values for enhanced data visualization.

Given a query result of two time series:

- Series 1: labels Server=Server A, Datacenter=EU
- Series 2: labels Server=Server B, Datacenter=EU

In "Columns" mode, the result looks like this:

Time	Server	Datacenter	Value
2020-07-07 11:34:20	Server A	EU	1
2020-07-07 11:34:20	Server B	EU	2

In “Rows” mode, the result has a table for each series and show each label value like this:

label	value
Server	Server A
Datacenter	EU

label	value
Server	Server B
Datacenter	EU

Value field name

If you selected Server as the **Value field name**, then you would get one field for every value of the Server label.

Time	Datacenter	Server A	Server B
2020-07-07 11:34:20	EU	1	2

Merging behavior

The labels to fields transformer is internally two separate transformations. The first acts on single series and extracts labels to fields. The second is the merge transformation that joins all the results into a single table. The merge transformation tries to join on all matching fields. This merge step is required and cannot be turned off.

To illustrate this, here is an example where you have two queries that return time series with no overlapping labels.

- Series 1: labels Server=ServerA
- Series 2: labels Datacenter=EU

This will first result in these two tables:

Time	Server	Value
2020-07-07 11:34:20	ServerA	10

Time	Datacenter	Value
2020-07-07 11:34:20	EU	20

After merge:

Time	Server	Value	Datacenter
2020-07-07 11:34:20	ServerA	10	
2020-07-07 11:34:20		20	EU

Convert your time series data into a structured table format for a clearer and more organized representation.

Limit

Use this transformation to restrict the number of rows displayed, providing a more focused view of your data. This is particularly useful when dealing with large datasets.

Below is an example illustrating the impact of the **Limit** transformation on a response from a data source:

Time	Metric	Value
2020-07-07 11:34:20	Temperature	25
2020-07-07 11:34:20	Humidity	22
2020-07-07 10:32:20	Humidity	29
2020-07-07 10:31:22	Temperature	22
2020-07-07 09:30:57	Humidity	33
2020-07-07 09:30:05	Temperature	19

Here is the result after adding a Limit transformation with a value of ‘3’:

Time	Metric	Value
2020-07-07 11:34:20	Temperature	25
2020-07-07 11:34:20	Humidity	22
2020-07-07 10:32:20	Humidity	29

This transformation helps you tailor the visual presentation of your data to focus on the most relevant information.

Merge series/tables

Use this transformation to combine the results from multiple queries into a single result, which is particularly useful when using the table panel visualization. This transformation merges values into the same row if the shared fields contain the same data.

Here's an example illustrating the impact of the **Merge series/tables** transformation on two queries returning table data:

Query A:

Time	Job	Uptime
2020-07-07 11:34:20	node	25260122
2020-07-07 11:24:20	postgre	123001233

Query B:

Time	Job	Errors
2020-07-07 11:34:20	node	15
2020-07-07 11:24:20	postgre	5

Here is the result after applying the Merge transformation.

Time	Job	Errors	Uptime
2020-07-07 11:34:20	node	15	25260122
2020-07-07 11:24:20	postgre	5	123001233

This transformation combines values from Query A and Query B into a unified table, enhancing the presentation of data for better insights.

Organize fields by name

Use this transformation to provide the flexibility to rename, reorder, or hide fields returned by a single query in your panel. This transformation is applicable only to panels with a single query. If your panel has multiple queries, consider using an “Outer join” transformation or removing extra queries.

Transforming fields

Grafana displays a list of fields returned by the query, allowing you to perform the following actions:

- **Change field order** - Hover over a field, and when your cursor turns into a hand, drag the field to its new position.
- **Hide or show a field** - Use the eye icon next to the field name to toggle the visibility of a specific field.
- **Rename fields** - Type a new name in the “Rename ” box to customize field names.

Example:

Original Query Result

Time	Metric	Value
2020-07-07 11:34:20	Temperature	25
2020-07-07 11:34:20	Humidity	22
2020-07-07 10:32:20	Humidity	29

After Applying Field Overrides

Time	Sensor	Reading
2020-07-07 11:34:20	Temperature	25
2020-07-07 11:34:20	Humidity	22
2020-07-07 10:32:20	Humidity	29

This transformation lets you to tailor the display of query results, ensuring a clear and insightful representation of your data in Grafana.

Partition by values

Use this transformation to streamline the process of graphing multiple series without the need for multiple queries with different ‘WHERE’ clauses.

This is particularly useful when dealing with a metrics SQL table, as illustrated below:

Time	Region	Value
2022-10-20 12:00:00	US	1520
2022-10-20 12:00:00	EU	2936
2022-10-20 01:00:00	US	1327
2022-10-20 01:00:00	EU	912

With the **Partition by values** transformation, you can issue a single query and split the results by unique values in one or more columns (fields) of your choosing. The following example uses ‘Region’:

‘SELECT Time, Region, Value FROM metrics WHERE Time > “2022-10-20”’

Time	Region	Value
2022-10-20 12:00:00	US	1520
2022-10-20 01:00:00	US	1327

Time	Region	Value
2022-10-20 12:00:00	EU	2936
2022-10-20 01:00:00	EU	912

This transformation simplifies the process and enhances the flexibility of visualizing multiple series within the same time series visualization.

Prepare time series

Use this transformation to address issues when a data source returns time series data in a format that isn’t compatible with the desired visualization. This transformation allows you to convert time series data between wide and long formats, providing flexibility in data frame structures.

Available options

Multi-frame time series

Use this option to transform the time series data frame from the wide format to the long format. This is particularly helpful when your data source delivers time series information in a format that needs to be reshaped for optimal compatibility with your visualization.

Example: Converting from wide to long format

Timestamp	Value1	Value2
2023-01-01 00:00:00	10	20
2023-01-01 01:00:00	15	25

Transformed to:

Timestamp	Variable	Value
2023-01-01 00:00:00	Value1	10
2023-01-01 00:00:00	Value2	20
2023-01-01 01:00:00	Value1	15
2023-01-01 01:00:00	Value2	25

Wide time series

Select this option to transform the time series data frame from the long format to the wide format. If your data source returns time series data in a long format and your visualization requires a wide format, this transformation simplifies the process.

Example: Converting from long to wide format

Timestamp	Variable	Value
2023-01-01 00:00:00	Value1	10
2023-01-01 00:00:00	Value2	20
2023-01-01 01:00:00	Value1	15
2023-01-01 01:00:00	Value2	25

Transformed to:

Timestamp	Value1	Value2
2023-01-01 00:00:00	10	20
2023-01-01 01:00:00	15	25

Note: This transformation is available in Grafana 7.5.10+ and Grafana 8.0.6+.

Reduce

Use this transformation to apply a calculation to each field in the data frame and return a single value. This transformation is particularly useful for consolidating multiple time series data into a more compact, summarized format. Time fields are removed when applying this transformation.

Consider the input:

Query A:

Time	Temp	Uptime
2020-07-07 11:34:20	12.3	256122
2020-07-07 11:24:20	15.4	1230233

Query B:

Time	AQI	Errors
2020-07-07 11:34:20	6.5	15
2020-07-07 11:24:20	3.2	5

The reduce transformer has two modes:

- **Series to rows** - Creates a row for each field and a column for each calculation.
- **Reduce fields** - Keeps the existing frame structure, but collapses each field into a single value.

For example, if you used the **First** and **Last** calculation with a **Series to rows** transformation, then the result would be:

Field	First	Last
Temp	12.3	15.4
Uptime	256122	1230233
AQI	6.5	3.2
Errors	15	5

The **Reduce fields** with the **Last** calculation, results in two frames, each with one row:

Query A:

Temp	Uptime
15.4	1230233

Query B:

AQI	Errors
3.2	5

This flexible transformation simplifies the process of consolidating and summarizing data from multiple time series into a more manageable and organized format.

Rename by regex

Use this transformation to rename parts of the query results using a regular expression and replacement pattern.

You can specify a regular expression, which is only applied to matches, along with a replacement pattern that support back references. For example, let's imagine you're visualizing CPU usage per host and you want to remove the domain name. You could set the regex to '/^([^.]+).*/' and the replacement pattern to '$1', 'web-01.example.com' would become 'web-01'.

Note: The Rename by regex transformation was improved in Grafana v9.0.0 to allow global patterns of the form '//g'. Depending on the regex match used, this may cause some transformations to behave slightly differently. You can guarantee the same behavior as before by wrapping the match string in forward slashes '(/)', e.g. '*(.)' would become '/(.)/*'.

In the following example, we are stripping the 'A-' prefix from field names. In the before image, you can see everything is prefixed with 'A-':

With the transformation applied, you can see we are left with just the remainder of the string.

This transformation lets you to tailor your data to meet your visualization needs, making your dashboards more informative and user-friendly.

Rows to fields

Use this transformation to convert rows into separate fields. This can be useful because fields can be styled and configured individually. It can also use additional fields as sources for dynamic field configuration or map them to field labels. The additional labels can then be used to define better display names for the resulting fields.

This transformation includes a field table which lists all fields in the data returned by the configuration query. This table gives you control over what field should be mapped to each

configuration property (the **Use as** option). You can also choose which value to select if there are multiple rows in the returned data.

This transformation requires:

- One field to use as the source of field names.

  By default, the transform uses the first string field as the source. You can override this default setting by selecting **Field name** in the **Use as** column for the field you want to use instead.

- One field to use as the source of values.

  By default, the transform uses the first number field as the source. But you can override this default setting by selecting **Field value** in the **Use as** column for the field you want to use instead.

Useful when visualizing data in:

- Gauge
- Stat
- Pie chart

Map extra fields to labels

If a field does not map to config property Grafana will automatically use it as source for a label on the output field-

Example:

Name	DataCenter	Value
ServerA	US	100
ServerB	EU	200

Output:

ServerA (labels: DataCenter: US)	ServerB (labels: DataCenter: EU)
10	20

The extra labels can now be used in the field display name provide more complete field names.

If you want to extract config from one query and apply it to another you should use the config from query results transformation.

Example

Input:

Name	Value	Max
ServerA	10	100
ServerB	20	200
ServerC	30	300

Output:

ServerA (config: max=100)	ServerB (config: max=200)	ServerC (config: max=300)
10	20	30

As you can see each row in the source data becomes a separate field. Each field now also has a max config option set. Options like **Min**, **Max**, **Unit** and **Thresholds** are all part of field configuration and if set like this will be used by the visualization instead of any options manually configured in the panel editor options pane.

This transformation enables the conversion of rows into individual fields, facilitates dynamic field configuration, and maps additional fields to labels.

Series to rows

Use this transformation to combine the result from multiple time series data queries into one single result. This is helpful when using the table panel visualization.

The result from this transformation will contain three columns: Time, Metric, and Value. The Metric column is added so you easily can see from which query the metric originates from. Customize this value by defining Label on the source query.

In the example below, we have two queries returning time series data. It is visualized as two separate tables before applying the transformation.

Query A:

Time	Temperature
2020-07-07 11:34:20	25
2020-07-07 10:31:22	22

Time	Temperature
2020-07-07 09:30:05	19

Query B:

Time	Humidity
2020-07-07 11:34:20	24
2020-07-07 10:32:20	29
2020-07-07 09:30:57	33

Here is the result after applying the Series to rows transformation.

Time	Metric	Value
2020-07-07 11:34:20	Temperature	25
2020-07-07 11:34:20	Humidity	22
2020-07-07 10:32:20	Humidity	29
2020-07-07 10:31:22	Temperature	22
2020-07-07 09:30:57	Humidity	33
2020-07-07 09:30:05	Temperature	19

This transformation facilitates the consolidation of results from multiple time series queries, providing a streamlined and unified dataset for efficient analysis and visualization in a tabular format.

Note: This transformation is available in Grafana 7.1+.

Sort by

Use this transformation to sort each frame within a query result based on a specified field, making your data easier to understand and analyze. By configuring the desired field for sorting, you can control the order in which the data is presented in the table or visualization.

Use the **Reverse** switch to inversely order the values within the specified field. This functionality is particularly useful when you want to quickly toggle between ascending and descending order to suit your analytical needs.

For example, in a scenario where time-series data is retrieved from a data source, the **Sort by** transformation can be applied to arrange the data frames based on the timestamp, either in ascending or descending order, depending on the analytical requirements. This capability ensures that you can easily navigate and interpret time-series data, gaining valuable insights from the organized and visually coherent presentation.

Spatial

Use this transformation to apply spatial operations to query results.

- **Action** - Select an action:
  - **Prepare spatial field** - Set a geometry field based on the results of other fields.
    - **Location mode** - Select a location mode (these options are shared by the **Calculate value** and **Transform** modes):
      - **Auto** - Automatically identify location data based on default field names.
      - **Coords** - Specify latitude and longitude fields.
      - **Geohash** - Specify a geohash field.
      - **Lookup** - Specify Gazetteer location fields.
  - **Calculate value** - Use the geometry to define a new field (heading/distance/area).
    - **Function** - Choose a mathematical operation to apply to the geometry:
      - **Heading** - Calculate the heading (direction) between two points.
      - **Area** - Calculate the area enclosed by a polygon defined by the geometry.
      - **Distance** - Calculate the distance between two points.
  - **Transform** - Apply spatial operations to the geometry.
    - **Operation** - Choose an operation to apply to the geometry:
      - **As line** - Create a single line feature with a vertex at each row.
      - **Line builder** - Create a line between two points.

This transformation allows you to manipulate and analyze geospatial data, enabling operations such as creating lines between points, calculating spatial properties, and more.

Time series to table transform

Use this transformation to convert time series results into a table, transforming a time series data frame into a **Trend** field. The **Trend** field can then be rendered using the sparkline cell type, generating an inline sparkline for each table row. If there are multiple time series queries, each will result in a separate table data frame. These can be joined using join or merge transforms to produce a single table with multiple sparklines per row.

For each generated **Trend** field value, a calculation function can be selected. The default is **Last non-null value**. This value is displayed next to the sparkline and used for sorting table rows.

Note: This transformation is available in Grafana 9.5+ as an opt-in beta feature. Modify the Grafana configuration file to use it.

Regression analysis

Use this transformation to create a new data frame containing values predicted by a statistical model. This is useful for finding a trend in chaotic data. It works by fitting a mathematical function to the data, using either linear or polynomial regression. The data frame can then be used in a visualization to display a trendline.

There are two different models:

- **Linear regression** - Fits a linear function to the data.

- **Polynomial regression** - Fits a polynomial function to the data.

Note: This transformation is currently in public preview. Grafana Labs offers limited support, and breaking changes might occur prior to the feature being made generally available. Enable

the `regressionTransformation` feature toggle in Grafana to use this feature. Contact Grafana Support to enable this feature in Grafana Cloud.

Troubleshoot queries

This page provides information to solve common dashboard problems.

I get different results when I rearrange my functions

Function order is very important. Just like in math, the order that you place your functions can affect the result.

Inspect your query request and response

The most common problems are related to the query and response from your data source. Even if it looks like a bug or visualization issue in Grafana, it is almost always a problem with the data source query or the data source response. Start by inspecting your panel query and response.

For more information, refer to Inspect request and response data.

My query is slow

How many data points is your query returning? A query that returns lots of data points will be slow. Try this:

- In **Query options**, limit the **Max data points** returned.
- In **Query options**, increase the **Min interval** time.
- In your query, use a `group by` function.

Calculation types

The following table contains a list of calculations you can perform in Grafana. You can find these calculations in the **Transform** tab and in the bar gauge, gauge, and stat visualizations.

Calculation	Description
All nulls	True when all values are null
All values	Array with all values
All unique values	Array with all unique values
All zeros	True when all values are 0
Change count	Number of times the field's value changes
Count	Number of values in a field
Delta	Cumulative change in value, only counts increments
Difference	Difference between first and last value of a field
Difference percent	Percentage change between first and last value of a field
Distinct count	Number of unique values in a field
First	First value in a field
First* (not null)	First, not null value in a field (also excludes NaNs)
Last	Last value in a field
Last* (not null)	Last, not null value in a field (also excludes NaNs)
Max	Maximum value of a field
Mean	Mean value of all values in a field
Variance	Variance of all values in a field

Calculation	Description
StdDev	Standard deviation of all values in a field
Min	Minimum value of a field
Min (above zero)	Minimum, positive value of a field
Range	Difference between maximum and minimum values of a field
Step	Minimal interval between values of a field
Total	Sum of all values in a field

Configure panel options

There are settings common to all visualizations, which you set in the **Panel options** section of the panel editor pane. The following sections describe these options as well as how to set them.

Panel options

Set the following options to provide basic information about a panel and define basic display elements:

Option	Description
Title	Text entered in this field appears at the top of your panel in the panel editor and in the dashboard. You can use variables you have defined in the **Title** field, but not global variables.
Description	Text entered in this field appears in a tooltip in the upper-left corner of the panel. Add a description to a panel to share with users any important information about it, such as its purpose. You can use variables you have defined in the **Description** field, but not global variables.
Transparent background	Toggle this switch on and off to control whether or not the panel has the same background color as the dashboard.
Panel links	Add links to the panel to create shortcuts to other dashboards, panels, and external websites. Access panel links by clicking the icon next to the panel title.
Repeat options	Set whether to repeat the panel for each value in the selected variable. For more information, refer to Configure repeating panels.

You can use generative AI to populate the **Title** and **Description** fields with the Grafana LLM plugin, which is currently in public preview. To enable this, refer to Set up generative AI features for dashboards.

Configure repeating panels

You can configure Grafana to dynamically add panels or rows to a dashboard. A dynamic panel is a panel that the system creates based on the value of a variable. Variables dynamically change your queries across all panels in a dashboard. For more information about repeating rows, refer to Configure repeating rows.

To see an example of repeating panels, refer to this dashboard with repeating panels.

Before you begin:

- Ensure that the query includes a multi-value variable.

To configure repeating panels, follow these steps:

1. Navigate to the panel you want to update.
2. Hover over any part of the panel to display the menu on the top right corner.
3. Click the menu and select **Edit**.
4. Open the **Panel options** section of the panel editor pane.
5. Under **Repeat options**, select a variable in the **Repeat by variable** drop-down list.
6. Under **Repeat direction**, choose one of the following:
   - **Horizontal** - Arrange panels side-by-side. Grafana adjusts the width of a repeated panel. You can't mix other panels on a row with a repeated panel.
   - **Vertical** - Arrange panels in a column. The width of repeated panels is the same as the original, repeated panel.
7. If you selected **Horizontal** in the previous step, select a value in the **Max per row** drop-down list to control the maximum number of panels that can be in a row.
8. Click **Save**.
9. To propagate changes to all panels, reload the dashboard.

You can stop a panel from repeating by selecting **Disable repeating** in the **Repeat by variable** drop-down list.

Configure standard options

Standard options in the panel editor pane let you change how field data is displayed in your visualizations. Options that you apply don't change the data, they just change how Grafana *displays* the data.

When you set a standard option, the change is applied to all fields or series. For example, if you set the **Unit** option to **Percentage**, all fields with numeric values are displayed as percentages.

For more granular control over the display of fields, refer to Configure overrides.

Supported visualizations

You can configure standard options for the following visualizations:

Bar chart	Geomap	Status history
Bar gauge	Histogram	Table
Candlestick	Pie chart	Time series
Canvas	Stat	Trend
Gauge	State timeline	

Standard options

This section explains all available standard options.

To set these options, expand the **Standard options** section in the panel editor pane. Most field options won't affect the visualization until you click outside of the field option box you're editing or press Enter.

NOTE

Not all of the options listed apply to all visualizations with standard options.

Unit

This option lets you choose which unit a field should use. Click in the **Unit** field, then drill down until you find the unit you want. The unit you select is applied to all fields except time.

Custom units

You can also use the **Unit** drop-down to specify custom units, custom prefixes or suffixes, and date time formats.

To set a custom unit, enter the unit you want to use and then select it in the drop-down. It'll be the last option listed. For example, if you enter a unit called "Hearts", the drop-down will then include the option **Custom unit: Hearts**.

You can further define a custom unit with specific syntax. For example, to set a custom currency unit called "Gems", enter `currency:Gems` in the field. The drop-down will include the option **Custom unit: currency:Gems**:

The following table lists the special syntax options for custom units:

Custom unit	Description
`suffix:<suffix>`	Custom unit that should go after value.
`prefix:<prefix>`	Custom unit that should go before value.
`time:<format>`	Custom date time formats type, such as `time:YYYY-MM-DD`. Refer to formats for the format syntax and options.
`si:<base scale> <unit characters>`	Custom SI units, such as `si: mF`. You can specify both a unit and the source data scale. For example, if your source data is represented as milli-something, prefix the unit with the `m` SI scale character.
`count:<unit>`	Custom count unit.
`currency:<unit>`	Custom currency unit.

You can also paste a native emoji in the **Unit** drop-down and select it as a custom unit:

Control unit scaling

By default, Grafana automatically scales the unit based on the magnitude of the value. For example, if you have values of 0.14kW and 3000kW, Grafana displays them as 140W and 3MW, respectively. You can use custom units to control this behavior by setting a prefix, suffix, or custom SI unit.

String units

Sometimes Grafana is too aggressive in interpreting strings and displaying them as numbers. To configure Grafana to show the original string value, select **Misc > String** in the **Unit** drop-down.

Min

Set the minimum value used in percentage threshold calculations. Leave this field empty to automatically calculate the minimum.

Max

Set the maximum value used in percentage threshold calculations. Leave this field empty to automatically calculate the maximum.

Field min/max

By default, the calculated **Min** and **Max** are based on the minimum and maximum of all series and fields. When you enable **Field min/max**, Grafana calculates the min or max of each field individually, based on the minimum or maximum value of the field.

Decimals

Specify the number of decimals Grafana includes in the rendered value. If you leave this field empty, Grafana automatically truncates the number of decimals based on the value. For example 1.1234 displays as 1.12 and 100.456 displays as 100.

To display all decimals, set the unit to **String**.

Display name

Set the display title of all fields. You can use variables in the field title.

When multiple stats, fields, or series are displayed, this field controls the title in each stat. You can use expressions like `${__field.name}` to use only the series name or the field name in the title.

The following table shows examples of the different field names generated using various expressions. In this example, there's a field with a name of "Temp" and labels of {"Loc"="PBI", "Sensor"="3"}:

Expression syntax	Example	Renders to	Explanation
`${__field.displayName}`	Same as syntax	`Temp {Loc="PBI", Sensor="3"}`	Displays the field name, and labels in `{}` if they are present. If there is only one label key in the response, then for the label portion, Grafana displays the value of the label without the enclosing braces.
`${__field.name}`	Same as syntax	`Temp`	Displays the name of the field (without labels).
`${__field.labels}`	Same as syntax	`Loc="PBI", Sensor="3"`	Displays the labels without the name.
`${__field.labels.X}`	`${__field.labels.Loc}`	`PBI`	Displays the value of the specified label key.
`${__field.labels.__values}`	Same as Syntax	`PBI, 3`	Displays the values of the labels separated by a comma (without label keys).

If the value is an empty string after rendering the expression for a particular field, then the default display method is applied.

Color scheme

The **Color scheme** options let you set single or multiple colors for your entire visualization.

The color options and their effect on a visualization depend on the visualization you're working with and some visualizations have different color options.

Select one of the following schemes:

Color scheme	Description
Single color	Specifies a single color.
Shades of a color	Grafana selects shades of a single color.
From thresholds (by value)	The color is taken from the matching threshold. For some visualizations, you also need to choose if the color is set by the **Last**, **Min**, or **Max** value of the field or series.
Classic palette	Grafana automatically assigns a color for each field or series based on its order. If the order of a field changes in your query, the color also changes. Useful for graphs, pie charts, and other categorical data visualizations.
Classic palette (by series name)	Grafana automatically assigns colors based on the name of the series. Useful when the series names to be visualized can change based on the available data.
Multiple continuous colors (by value)	Grafana automatically assigns colors based on the percentage of a value relative to the min and the max of the field or series. For some visualizations, you also need to choose if the color is set by the **Last**, **Min**, or **Max** value of the field or series. Select from: **Green-Yellow-Red**, **Red-Yellow-Green**, **Blue-Yellow-Red**, **Yellow-Red**, **Blue-Purple**, and **Yellow-Blue**.
Single continuous color (by value)	Grafana automatically assigns shades of one color based on the percentage of a value relative to the min and the max of the field or series. For some visualizations, you also need to choose if the color is set by the **Last**, **Min**, or **Max** value of the field or series. Select from: **Blues**, **Reds**, **Greens**, and **Purples**.

You can also use the legend to open the color picker by clicking the legend series color icon. Setting color this way automatically creates an override rule that set's a specific color for a specific series.

No value

Enter what Grafana should display if the field value is empty or null. The default value is a hyphen (-).

Configure a legend

A panel includes a legend that you can use to identify and interpret data displayed in a visualization. Each legend option adds context and clarity to the data illustrated in a visualization.

Supported visualizations

Legends are supported for the following visualizations:

- Bar chart
- Candlestick
- Histogram
- Pie chart
- State timeline
- Status history
- Time series
- Trend

Geomaps and heatmaps also have legends, but they only provide the choice to display or not display a legend and don't support other legend options.

Legend options

You can find the following options under the **Legend** section in the panel edit pane.

> **NOTE**
>
> Not all of the options listed apply to all visualizations with legends.

Visibility

Set whether the legend is displayed or not. Use the switch to toggle a legend on or off.

Mode

Set the format in which the legend is displayed. Choose from:

- **List**
- **Table**

When you format a legend as a table, other information about the legend, such as associated values or where it's located in the visualization, might be displayed as well.

Placement

Set where on the visualization a legend is displayed. Choose from:

- **Bottom**
- **Right**

Width

If you set the legend placement to **Right**, the **Width** option becomes available. Leave the field empty to allow Grafana to automatically set the legend width or enter a value in the field.

Values

You can add more context to a visualization by adding series data values or calculations to a legend. You can add as many values as you'd like. After you apply your changes, you can scroll the legend to see all values.

Change a series color

By default, Grafana sets the colors of your series data, but you can change them through the panel legend. To change the series data color, follow these steps:

1. Navigate to the panel you want to update.
2. In the legend, click the color bar associated with the series.
3. Select a pre-set color in the **Colors** tab or set a custom color in the **Custom** tab, using the picker or RGB values.
4. Save the dashboard.

Isolate series data in a visualization

Visualizations can often be visually complex, and include many data series. You can simplify the view by removing series data from the visualization through the legend, which isolates the data you want to see. When you do this, Grafana automatically creates a new override in the **Override** section.

To isolate a series, follow these steps:

1. Navigate to the panel you want to update.
2. In the legend, click the label of the series you want to isolate.

   The system removes all other series data from view.
3. To incrementally add series data back to an isolated series, press the **Ctrl** or **Command** key and click the label of the series you want to add.
4. To save your changes so that they appear to all viewers of the panel, save the dashboard.

To revert back to the default view that includes all data, click any series label twice.

Sort series

When you format a legend as a table and add values to it, you can sort series in the table by those values. To do so, follow these steps:

1. Navigate to the panel you want to update.
2. Hover over any part of the panel you want to work on to display the menu on the top right corner.
3. Click the menu and select **Edit**.
4. Scroll to the **Legend** section of the panel edit pane.
5. Under **Mode**, select **Table**.

6 Under **Values**, select the value or calculation that you want to show.

The legend table now displays values.

7 Click the calculation name header in the legend table to sort the values in the table in ascending or descending order.

NOTE

This feature is only supported for the following visualizations: bar chart, histogram, time series.

Configure tooltips

NOTE

The new tooltip experience is currently in public preview. Grafana Labs offers limited support, and breaking changes might occur prior to the feature being made generally available. Enable the `newVizTooltips` feature toggle in Grafana to use this feature. Contact Grafana Support to enable this feature in Grafana Cloud.

When you hover your cursor over a visualization, Grafana can display tooltips that contain more information about a data point, like the exact time of a result. You can customize tooltips to control how many series they include and the order of those values. You can also copy the content from tooltips to use elsewhere. Learn more about configuring tooltips in Tooltip options.

Supported visualizations

You can configure tooltips for the following visualizations:

Bar chart	State timeline
Candlestick	Status history
Heatmap	Time series
Pie chart	Trend

Some visualizations, for example candlestick and flame graph, have tooltips, but they aren't configurable. These visualizations don't have a **Tooltip** section in the panel editor pane. Geomaps provide you the option to have tooltips triggered upon click or hover under the **Map controls** options in the panel editor pane.

Tooltip options

You can find the following options under the **Tooltip** section in the panel edit pane.

> **NOTE**
>
> Not all of the options listed apply to all visualizations with tooltips.

Tooltip mode

Choose how tooltips behave with the following options:

- **Single** - The tooltip only the single series that you're hovering over in the visualization.
- **All** - The tooltip shows all series in the visualization. Grafana highlights the series that you are hovering over in bold in the series list in the tooltip.
- **Hidden** - Tooltips aren't displayed when you interact with the visualization.

You can use a field override to hide individual series from the tooltip.

Values sort order

When you set the **Tooltip mode** to **All**, the **Values sort order** option is displayed. This option controls the order in which values are listed in a tooltip. Choose from the following:

- **None** - Grafana automatically sorts the values displayed in a tooltip.
- **Ascending** - Values in the tooltip are listed from smallest to largest.
- **Descending** - Values in the tooltip are listed from largest to smallest.

Hover proximity

Set the hover proximity (in pixels) to control how close the cursor must be to a data point to trigger the tooltip to display.

Max height

Set the maximum height of the tooltip box. The default is 600 pixels.

Show histogram (Y axis)

For the heatmap visualization only, when you set the **Tooltip mode** to **Single**, the **Show histogram (Y axis)** option is displayed. This option controls whether or not the tooltip includes a histogram representing the y-axis.

Show color scale

For the heatmap visualization only, when you set the **Tooltip mode** to **Single**, the **Show color scale** option is displayed. This option controls whether or not the tooltip includes the color scale that's also represented in the legend. When the color scale is included in the tooltip, it shows the hovered value on the scale:

Configure data links

Data links allow you to link to other panels, dashboards, and external resources while maintaining the context of the source panel. You can create links that include the series name or even the value under the cursor. For example, if your visualization shows four servers, you can add a data link to one or two of them.

The link itself is accessible in different ways depending on the visualization. For the time series visualization you need to click a data point or line:

For visualizations like stat, gauge, or bar gauge you can click anywhere on the visualization to open the context menu:

If there's only one data link in the visualization, clicking anywhere on the visualization opens the link rather than the context menu.

Supported visualizations

You can configure data links for the following visualizations:

Bar chart	Geomap	State timeline
Bar gauge	Heatmap	Status history
Candlestick	Histogram	Table
Canvas	Pie chart	Time series
Gauge	Stat	Trend

Data link variables

Variables in data links let you send people to a detailed dashboard with preserved data filters. For example, you could use variables to specify a label, time range, series, or variable selection.

To see a list of available variables, enter `$` in the data link **URL** field.

> **NOTE**
>
> These variables changed in 6.4 so if you have an older version of Grafana, then use the version picker to select docs for an older version of Grafana.

Azure Monitor, CloudWatch, and Google Cloud Monitoring have pre-configured data links called *deep links*.

You can also use template variables in your data links URLs. For more information, refer to Templates and variables.

Time range panel variables

These variables allow you to include the current time range in the data link URL:

Variable	Description
`__url_time_range`	Current dashboard's time range (for example, `?from=now-6h&to=now`)
`__from`	For more information, refer to Global variables.

Variable	Description
`__to`	For more information, refer to Global variables.

When you create data links using time range variables like `__url_time_range` in the URL, you have to form the query parameter syntax yourself; that is, you must format the URL by appending query parameters using the question mark (`?`) and ampersand (`&`) syntax. These characters aren't automatically generated.

Series variables

Series-specific variables are available under `__series` namespace:

Variable	Description
`__series.name`	Series name to the URL

Field variables

Field-specific variables are available under `__field` namespace:

Variable	Description
`__field.name`	The name of the field
`__field.labels.<LABEL>`	Label's value to the URL. If your label contains dots, then use `__field.labels["<LABEL>"]` syntax.

Value variables

Value-specific variables are available under `__value` namespace:

Variable	Description
`__value.time`	Value's timestamp (Unix ms epoch) to the URL (for example, `?time=1560268814105`)
`__value.raw`	Raw value
`__value.numeric`	Numeric representation of a value
`__value.text`	Text representation of a value
`__value.calc`	Calculation name if the value is result of calculation

Using value-specific variables in data links can show different results depending on the set option of Tooltip mode.

When you create data links using time range variables like `__value.time` in the URL, you have to form the query parameter syntax yourself; that is, you must add the question mark (`?`) and ampersand (`&`). These characters aren't automatically generated.

Data variables

To access values and labels from other fields use:

Variable	Description
`__data.fields[i]`	Value of field `i` (on the same row)
`__data.fields["NameOfField"]`	Value of field using name instead of index
`__data.fields[1].labels.cluster`	Access labels of another field

Template variables

When linking to another dashboard that uses template variables, select variable values for whoever clicks the link.

`${var-myvar:queryparam}` - where `var-myvar` is the name of the template variable that matches one in the current dashboard that you want to use.

Variable state	Result in the created URL
selected one value	`var-myvar=value1`
selected multiple values	`var-myvar=value1&var-myvar=value2`
selected `All`	`var-myvar=All`

If you want to add all of the current dashboard's variables to the URL, then use `${__all_variables}`.

Add a data link

1. Navigate to the panel to which you want to add the data link.
2. Hover over any part of the panel to display the menu icon in the upper-right corner.
3. Click the menu icon and select **Edit** to open the panel editor.
4. In the panel edit pane, scroll down to the **Data links** section and expand it.

5. Click **Add link**.
6. In the dialog box that opens, enter a **Title**. This is a human-readable label for the link, which will be displayed in the UI.
7. Enter the **URL** or variable to which you want to link.

   To add a data link variable, click in the **URL** field and enter `$` or press Ctrl+Space or Cmd+Space to see a list of available variables.
8. If you want the link to open in a new tab, then toggle the **Open in a new tab** switch.
9. Click **Save** to save changes and close the dialog box.
10. Click **Apply** to see your changes in the dashboard.
11. Click the **Save dashboard** icon to save your changes to the dashboard.

Configure value mappings

In addition to field overrides, value mapping is a technique you can use to change how data appears in a visualization.

For example, the mapping applied in the following image causes the visualization to display the text `Cold`, `Good`, and `Hot` in blue, green, and red for ranges of temperatures rather than actual temperature values. Using value mappings this way can make data faster and easier to understand and interpret.

Value mappings bypass unit formatting set in the **Standard options** section of panel editor, like color or number of decimal places displayed. When value mappings are present in a panel, Grafana displays a summary of them in the **Value mappings** section of the editor panel.

Supported visualizations

You can configure value mappings for the following visualizations:

Bar chart	Geomap	Status history
Bar gauge	Histogram	Table
Candlestick	Pie chart	Time series
Canvas	Stat	Trend
Gauge	State timeline	

Types of value mappings

Grafana supports the following value mapping types:

Value

A **Value** mapping maps specific values to text and a color. For example, you can configure a mapping so that all instances of the value `10` appear as **Perfection!** rather than the number. Use **Value** mapping when you want to format a single value.

Range

A **Range** mapping maps numerical ranges to text and a color. For example, if a value is within a certain range, you can configure a range value mapping to display **Low** or **High** rather than the number. Use **Range** mapping when you want to format multiple, continuous values.

Regex

A **Regex** mapping maps regular expressions to text and a color. For example, if a value is `www.example.com`, you can configure a regular expression value mapping so that Grafana displays **www** and truncates the domain. Use the **Regex** mapping when you want to format the text and color of a regular expression value.

Special

A **Special** mapping maps special values like `Null`, `NaN` (not a number), and boolean values like `true` and `false` to text and color. For example, you can configure a special value mapping so that `null` values appear as **N/A**. Use the **Special** mapping when you want to format uncommon, boolean, or empty values.

Examples

Refer to the following examples to learn more about value mapping.

Time series example

The following image shows a time series visualization with value mappings. Value mapping colors aren't applied to this visualization, but the display text is shown on the axis.

Stat example

The following image shows a stat visualization with value mappings and text colors applied. You can hide the sparkline so it doesn't interfere with the values.

Bar gauge example

The following image shows a bar gauge visualization with value mappings. Note that the value mapping colors are applied to the text, but not to the gauges.

Table example

The following image shows a table visualization with value mappings. If you want value mapping colors displayed on the table, then set the cell display mode to **Color text** or **Color background**.

Add a value mapping

1. Navigate to the panel you want to update.

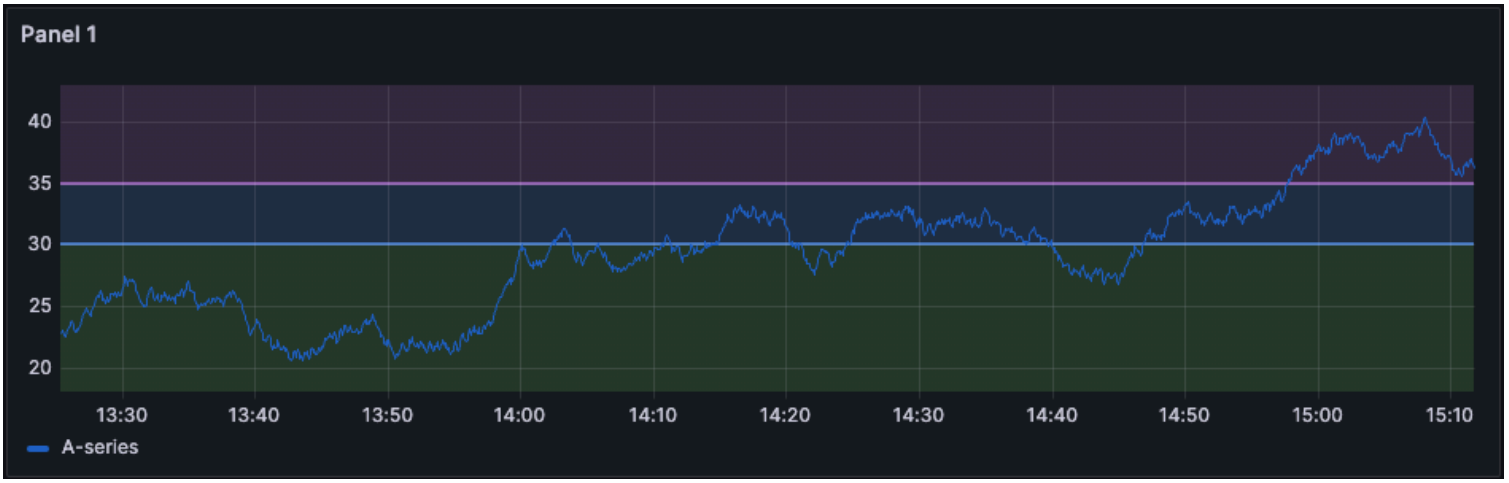
2. Hover over any part of the panel you want to work on to display the menu on the top right corner.
3. Click the menu and select **Edit**.
4. Scroll to the **Value mappings** section and expand it.
5. Click **Add value mappings**.
6. Click **Add a new mapping** and then select one of the following:
    - **Value** - Enter a single value to match.
    - **Range** - Enter the beginning and ending values of a range to match.
    - **Regex** - Enter a regular expression pattern to match.
    - **Special** - Select a special value to match.
7. (Optional) Enter display text.
8. (Optional) Set the color.
9. (Optional) Set an icon (canvas visualizations only).
10. Click **Update** to save the value mapping.

After you've added a mapping, the **Edit value mappings** button replaces the **Add value mappings** button. Click the edit button to add or update mappings.

Configure thresholds

In dashboards, a threshold is a value or limit you set for a metric that's reflected visually when it's met or exceeded. Thresholds are one way you can conditionally style and color your visualizations based on query results.

Using thresholds, you can color grid lines and regions in a time series visualization:



You can color the background or value text in a stat visualization:

You can define regions and region colors in a state timeline:

You can also use thresholds to:

- Color lines in a time series visualization
- Color the gauge and threshold markers in a gauge
- Color markers in a geomap
- Color cell text or background in a table

Give it a try using Grafana Play

With Grafana Play, you can explore and see how it works, learning from practical examples to accelerate your development. This feature can be seen on Threshold example.

Supported visualizations

You can set thresholds in the following visualizations:

Bar chart	Geomap	Status history
Bar gauge	Histogram	Table
Candlestick	Stat	Time series
Canvas	State timeline	Trend
Gauge		

Default thresholds

On visualizations that support thresholds, Grafana has the following default threshold settings:

- 80 = red

- Base = green
- Mode = Absolute
- Show thresholds = Off (for some visualizations); for more information, see the Show thresholds option.

Thresholds options

You can set the following options to further define how thresholds look.

Threshold value

This number is the value that triggers the threshold. You can also set the color associated with the threshold in this field.

The **Base** value represents minus infinity. By default, it's set to the color green, which is generally the "good" color.

Thresholds mode

There are two threshold modes:

- **Absolute** thresholds are defined by a number. For example, 80 on a scale of 1 to 150.
- **Percentage** thresholds are defined relative to minimum or maximum. For example, 80 percent.

Show thresholds

> **NOTE**
> This option is supported for the bar chart, candlestick, time series, and trend visualizations.

Set if and how thresholds are shown with the following options.

Option	Example
Off	
As lines	

Option	Example
As lines (dashed)	
As filled regions	
As filled regions and lines	
As filled regions and lines (dashed)	

Add a threshold

You can add as many thresholds to a visualization as you want. Grafana automatically sorts thresholds values from highest to lowest.

1. Navigate to the panel you want to update.
2. Hover over any part of the panel you want to work on to display the menu on the top right corner.
3. Click the menu and select **Edit**.
4. Scroll to the **Thresholds** section or enter `thresholds` in the search bar at the top of the panel edit pane.
5. Click **+ Add threshold**.

6. Enter a new threshold value or use the up and down arrows at the right side of the field to increase or decrease the value incrementally.
7. Click the colored circle to the left of the threshold value to open the color picker, where you can update the threshold color.
8. Under **Thresholds mode**, select either **Absolute** or **Percentage**.
9. Under **Show thresholds**, set how the threshold is displayed or turn it off.

To delete a threshold, navigate to the panel that contains the threshold and click the trash icon next to the threshold you want to remove.

Add a threshold to a legacy graph panel

> **CAUTION**
>
> Starting with Grafana v11, the legacy graph panel will be deprecated along with all other Angular panel plugins. For more information, refer to Angular support deprecation.

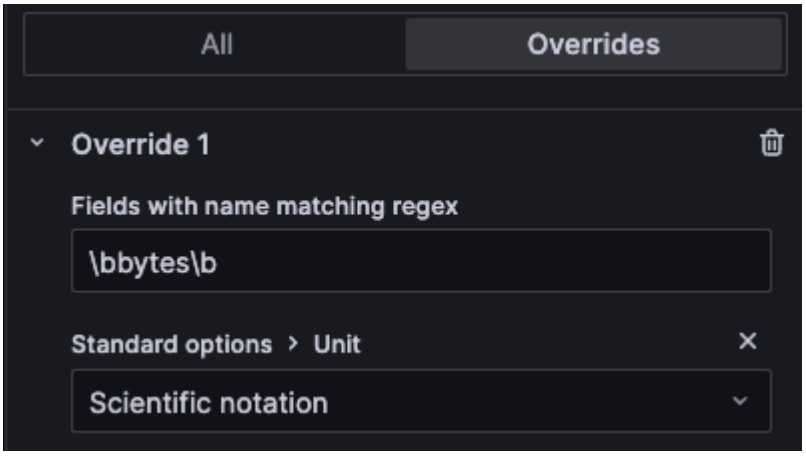
In the Graph panel visualization, thresholds enable you to add lines or sections to a graph to make it easier to recognize when the graph crosses a threshold.

1. Navigate to the graph panel to which you want to add a threshold.
2. On the **Panel** tab, click **Thresholds**.
3. Click **Add threshold**.
4. Complete the following fields:
   - **T1 -** Both values are required to display a threshold.
     - **lt** or **gt** - Select **lt** for less than or **gt** for greater than to indicate what the threshold applies to.
     - **Value -** Enter a threshold value. Grafana draws a threshold line along the Y-axis at that value.
   - **Color -** Choose a condition that corresponds to a color, or define your own color.
     - **custom -** You define the fill color and line color.
     - **critical -** Fill and line color are red.
     - **warning -** Fill and line color are yellow.
     - **ok -** Fill and line color are green.
   - **Fill -** Toggle the display of the threshold fill.
   - **Line -** Toggle the display of the threshold line.
   - **Y-Axis -** Choose to display the y-axis on either the **left** or **right** of the panel.
5. Click **Save** to save the changes in the dashboard.

Configure field overrides

Overrides allow you to customize visualization settings for specific fields or series. When you add an override rule, it targets a particular set of fields and lets you define multiple options for how that field is displayed.

For example, you can override the default unit measurement for all fields that include the text “bytes” by adding an override using the **Fields with name matching regex** matcher and then the **Standard options > Unit** setting to the override rule:



After you’ve set them, your overrides appear in both the **All** and **Overrides** tabs of the panel editor pane:

Supported visualizations

You can configure field overrides for the following visualizations:

Bar chart	Geomap	State timeline
Bar gauge	Heatmap	Status history
Candlestick	Histogram	Table
Canvas	Pie chart	Time series
Gauge	Stat	Trend

Override rules

You can choose from five types of override rules, which are described in the following sections.

Fields with name

Select a field from the list of all available fields. Properties you add to this type of rule are only applied to this single field.

Fields with name matching regex

Specify fields to override with a regular expression. Properties you add to this type of rule are applied to all fields where the field name matches the regular expression. This override doesn't rename the field; to do this, use the Rename by regex transformation.

Fields with type

Select fields by type, such as string, numeric, or time. Properties you add to this type of rule are applied to all fields that match the selected type.

Fields returned by query

Select all fields returned by a specific query, such as A, B, or C. Properties you add to this type of rule are applied to all fields returned by the selected query.

Fields with values

Select all fields returned by your defined reducer condition, such as **Min**, **Max**, **Count**, **Total**. Properties you add to this type of rule are applied to all fields returned by the selected condition.

Examples

The following examples demonstrate how you can use override rules to alter the display of fields in visualizations.

Example 1: Format temperature

The following result set is a data frame that consists of two fields: time and temperature.

time	temperature
2020-01-02 03:04:00	45.0
2020-01-02 03:05:00	47.0
2020-01-02 03:06:00	48.0

You can apply field options to each field (column) of this structure to alter the way its values are displayed. For example, you can set the following override rule:

- Rule: **Fields with type**
- Field: temperature
- Override property: **Standard options > Unit**
  - Selection: **Temperature > Celsius**

This results in the following table:

time	temperature
2020-01-02 03:04:00	45.0 °C
2020-01-02 03:05:00	47.0 °C
2020-01-02 03:06:00	48.0 °C

In addition, the decimal place isn't required, so you can remove it by adding another override property that changes the **Standard options > Decimals** setting from **auto** to `0`. That results in the following table:

time	temperature
2020-01-02 03:04:00	45 °C
2020-01-02 03:05:00	47 °C
2020-01-02 03:06:00	48 °C

Example 2: Format temperature and humidity

The following result set is a data frame that consists of four fields: time, high temp, low temp, and humidity.

time	high temp	low temp	humidity
2020-01-02 03:04:00	45.0	30.0	67
2020-01-02 03:05:00	47.0	34.0	68
2020-01-02 03:06:00	48.0	31.0	68

Use the following override rule and properties to add the **Celsius** unit option and remove the decimal place:

- Rule: **Fields with type**
- Field: temperature
- Override property: **Standard options > Unit**
  - Selection: **Temperature > Celsius**
- Override property: **Standard options > Decimals** -Change setting from **auto** to `0`

This results in the following table:

time	high temp	low temp	humidity
2020-01-02 03:04:00	45 °C	30 °C	67 °C
2020-01-02 03:05:00	47 °C	34 °C	68 °C
2020-01-02 03:06:00	48 °C	31 °C	68 °C

The temperature fields are displaying correctly, but the humidity has incorrect units. You can fix this by applying a **Misc > Percent (0-100)** override to the humidity field. This results in the following table:

time	high temp	low temp	humidity
2020-01-02 03:04:00	45 °C	30 °C	67%
2020-01-02 03:05:00	47 °C	34 °C	68%
2020-01-02 03:06:00	48 °C	31 °C	68%

Add a field override

To add a field override, follow these steps:

1. Navigate to the panel to which you want to add the data link.
2. Hover over any part of the panel to display the menu icon in the upper-right corner.
3. Click the menu icon and select **Edit** to open the panel editor.
4. At the bottom of the panel editor pane, click **Add field override**.
5. Select the fields to which the override will be applied:
    - **Fields with name**
    - **Fields with name matching regex**
    - **Fields with type**
    - **Fields returned by query**
    - **Fields with values**
6. Click **Add override property**.
7. Select the field option that you want to apply.
8. Continue to add overrides to this field by clicking **Add override property**.
9. Add as many overrides as you need.
10. When you're finished, click **Save** to save all panel edits to the dashboard.

Edit a field override

To edit a field override, follow these steps:

1. Navigate to the panel to which you want to add the data link.
2. Hover over any part of the panel to display the menu icon in the upper-right corner.
3. Click the menu icon and select **Edit** to open the panel editor.
4. In the panel editor pane, click the **Overrides** tab.
5. Locate the override you want to change.
6. Perform any of the following tasks:
    - Edit settings on existing overrides or field selection parameters.
    - Delete existing override properties by clicking the **X** next to the property.
    - Delete an override entirely by clicking the trash icon at the top-right corner.

The changes you make take effect immediately.

Search dashboards

You can search for dashboards by dashboard name and by panel title. When you search for dashboards, the system returns all dashboards available within the Grafana instance, even if you do not have permission to view the contents of the dashboard.

Search dashboards using dashboard name

Begin typing any part of the dashboard name in the search bar. The search returns results for any partial string match in real-time, as you type.

Dashboard search is:

- Real-time
- *Not* case sensitive
- Functional across stored *and* file based dashboards.

NOTE

You can use your keyboard arrow keys to navigate the results and press `Enter` to open the selected dashboard.

The following image shows the search results when you search using dashboard name.

Search dashboards using panel title

You can search for a dashboard by the title of a panel that appears in a dashboard. If a panel's title matches your search query, the dashboard appears in the search results.

This feature is available by default in Grafana Cloud and in Grafana OSS v9.1 and higher, you access this feature by enabling the `panelTitleSearch` feature toggle. For more information about enabling panel title search, refer to Enable the panelTitleSearch feature toggle.

The following image shows the search results when you search using panel title.

Enable the panelTitleSearch feature toggle

Complete the following steps to enable the `panelTitleSearch` feature toggle.

Before you begin:

- If you are running Grafana Enterprise with RBAC, enable service accounts.

To enable the panelTitleSearch feature toggle:

1. Open the Grafana configuration file.
2. Locate the feature_toggles section.
3. Add the following parameter to the `feature_toggles` section:

```
[feature_toggles]
# enable features, separated by spaces
enable = panelTitleSearch
```

Copy

4. Save your changes and restart the Grafana server.

Filter dashboard search results by tag(s)

Tags are a great way to organize your dashboards, especially as the number of dashboards grow. You can add and manage tags in dashboard `Settings`.

When you select multiple tags, Grafana shows dashboards that include all selected tags.

To filter dashboard search result by a tag, complete one of the following steps:

- To filter dashboard search results by tag, click a tag that appears in the right column of the search results.

  You can continue filtering by clicking additional tags.

- To see a list of all available tags, click the **Filter by tags** dropdown menu and select a tag.

  All tags will be shown, and when you select a tag, the dashboard search will be instantly filtered.

> **NOTE**
>
> When using only a keyboard, press the `tab` key and navigate to the **Filter by tag** drop-down menu, press the down arrow key `▼` to activate the menu and locate a tag, and press `Enter` to select the tag.

Command palette

The command palette enables you to:

- Search for and open dashboards and folders
- Create dashboards and alert rules
- Locate pages within Grafana
- Change the theme to dark or light

To open the command palette, press `cmd+K` in macOS or `ctrl+k` in Linux/Windows. You can also click on the input located in the navigation bar.

Note: To go to the previous step, press `backspace` with the command palette input empty.

Explore

Grafana's dashboard UI is all about building dashboards for visualization. Explore strips away the dashboard and panel options so that you can focus on the query. It helps you iterate until you have a working query and then think about building a dashboard.

Refer to Role-based access control in Grafana Enterprise to understand how you can control access with role-based permissions.

If you just want to explore your data and do not want to create a dashboard, then Explore makes this much easier. If your data source supports graph and table data, then Explore shows the results both as a graph and a table. This allows you to see trends in the data and more details at the same time. See also:

- Query management in Explore
- Logs integration in Explore
- Trace integration in Explore
- Explore metrics
- Correlations Editor in Explore
- Inspector in Explore

Start exploring

Refer to Role-based access Control in Grafana Enterprise to understand how you can manage Explore with role-based permissions.

In order to access Explore, you must have an editor or an administrator role, unless the viewers_can_edit option is enabled. Refer to About users and permissions for more information on what each role has access to.

> **NOTE**
>
> If you are using Grafana Cloud, open a support ticket in the Cloud Portal to enable the `viewers_can_edit` option

To access Explore:

1. Click on the Explore icon on the menu bar.

   An empty Explore tab opens.

   Alternately to start with an existing query in a panel, choose the Explore option from the Panel menu. This opens an Explore tab with the query from the panel and allows you to tweak or iterate in the query outside of your dashboard.

Screenshot of the panel menu including the Explore option

2. Choose your data source from the drop-down in the top left.

   You can also click **Open advanced data source picker** to see more options, including adding a data source (Admins only).

3. Write the query using a query editor provided by the selected data source. Please check data sources documentation to see how to use various query editors.

4. For general documentation on querying data sources in Grafana, see Query and transform data.

5. Run the query using the button in the top right corner.

Split and compare

The split view provides an easy way to compare visualizations side-by-side or to look at related data together on one page.

To open the split view:

1. Click the split button to duplicate the current query and split the page into two side-by-side queries.

It is possible to select another data source for the new query which for example, allows you to compare the same query for two different servers or to compare the staging environment to the production environment.

Screenshot of Explore screen split

In split view, timepickers for both panels can be linked (if you change one, the other gets changed as well) by clicking on one of the time-sync buttons attached to the timepickers. Linking of timepickers helps with keeping the start and the end times of the split view queries in sync. It ensures that you're looking at the same time interval in both split panels.

To close the newly created query, click on the Close Split button.

Content outline

The content outline is a side navigation bar that keeps track of the queries and visualization panels you created in Explore. It allows you to navigate between them quickly.

The content outline also works in a split view. When you are in split view, the content outline is generated for each pane.

To open the content outline:

1. Click the Outline button in the top left corner of the Explore screen.

You can then click on any panel icon in the content outline to navigate to that panel.

Share Explore URLs

When using Explore, the URL in the browser address bar updates as you make changes to the queries. You can share or bookmark this URL.

> **NOTE**
>
> Explore may generate relatively long URLs, some tools, like messaging or videoconferencing apps, may truncate messages to a fixed length. In such cases Explore will display a warning

message and load a default state. If you encounter issues when sharing Explore links in such apps, you can generate shortened links. See Share shortened link for more information.

Generating Explore URLs from external tools

Because Explore URLs have a defined structure, you can build a URL from external tools and open it in Grafana. The URL structure is:

```
http://<grafana_url>/explore?panes=<panes>&schemaVersion=<schema_version>&orgId=<org
```

Copy

where:

- `org_id` is the organization ID
- `schema_version` is the schema version (should be set to the latest version which is `1`)
- `panes` is a url-encoded JSON object of panes, where each key is the pane ID and each value is an object matching the following schema:

```
{
  datasource: string; // the pane's root datasource UID, or `-- Mixed --` for mixed
  queries: {
    refId: string; // an alphanumeric identifier for this query, must be unique with
    datasource: {
      uid: string; // the query's datasource UID ie: "AD7864H6422"
      type: string; // the query's datasource type-id, i.e: "loki"
    }
    // ... any other datasource-specific query parameters
  }[]; // array of queries for this pane
  range: {
    from: string; // the start time, in milliseconds since epoch
    to: string; // the end time, in milliseconds since epoch
  }
}
```

Copy

NOTE

The `from` and `to` also accept relative ranges defined in Time units and relative ranges.

Share shortened link

NOTE

Available in Grafana 7.3 and later versions.

The Share shortened link capability allows you to create smaller and simpler URLs of the format /goto/:uid instead of using longer URLs with query parameters. To create a shortened link to the executed query, click the **Share** option in the Explore toolbar.

A shortened link will automatically get deleted after seven (7) days from its creation if it's never used. If a link is used at least once, it won't ever get deleted.

Sharing shortened links with absolute time

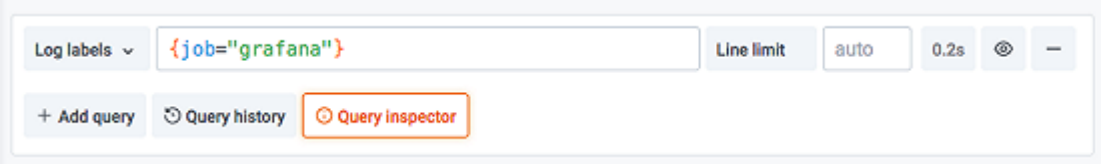
NOTE

Available in Grafana 10.3 and later versions.

Short links have two options - keeping relative time (for example, from two hours ago to now) or absolute time (for example, from 8am to 10am). Sharing a shortened link by default will copy the time range selected, relative or absolute. Clicking the dropdown button next to the share shortened link button and selecting one of the options under "Time-Sync URL Links" will allow you to create a short link with the absolute time - meaning anyone receiving the link will see the same data you are seeing, even if they open the link at another time. This will not affect your selected time range.

Query management in Explore

To help with debugging queries, Explore allows you to investigate query requests and responses, as well as query statistics, via the Query inspector. This functionality is similar to the panel inspector tasks Inspect query performance and Inspect query request and response data.



Screenshot of the query inspector button in Explore

Query history

Query history is a list of queries that you used in Explore. The history is stored in the Grafana database and it is not shared with other users. The retention period for queries in history is two weeks. Queries older than two weeks are automatically deleted. To open and interact with your history, click the **Query history** button in Explore.

> **NOTE**
>
> Starred queries are not subject to the two weeks retention period and they are not deleted.

View query history

Query history lets you view the history of your querying. For each individual query, you can:

- Run a query.
- Create and/or edit a comment.
- Copy a query to the clipboard.
- Copy a shortened link with the query to the clipboard.

- Star a query.

Manage favorite queries

All queries that have been starred in the Query history tab are displayed in the Starred tab. This allows you to access your favorite queries faster and to reuse these queries without typing them from scratch.

Sort query history

By default, query history shows you the most recent queries. You can sort your history by date or by data source name in ascending or descending order.

1. Click the **Sort queries by** field.
2. Select one of the following options:
   - Newest first
   - Oldest first

Filter query history

Filter query history in Query history and Starred tab by data source name:

1. Click the **Filter queries for specific data source(s)** field.
2. Select the data source for which you would like to filter your history. You can select multiple data sources.

Note: Queries ran using the Mixed data source will appear only when filtering for Mixed and not when filtering by their individual data sources.

In **Query history** tab it is also possible to filter queries by date using the slider:

- Use vertical slider to filter queries by date.
- By dragging bottom handle, adjust start date.
- By dragging top handle, adjust end date.

Search in query history

You can search in your history across queries and your comments. Search is possible for queries in the Query history tab and Starred tab.

1. Click the **Search queries** field.
2. Type the term you are searching for into search field.

Query history settings

You can customize the query history in the Settings tab. Options are described in the table below.

Setting	Default value
Change the default active tab	Query history tab

Note: Query history settings are global, and applied to both panels in split mode.

Logs in Explore

Explore is a powerful tool for logging and log analysis. It allows you to investigate logs from different data sources including:

- Loki
- Elasticsearch
- Cloudwatch
- InfluxDB
- Azure Monitor

With Explore, you can efficiently monitor, troubleshoot, and respond to incidents by analyzing your logs and identifying the root causes. It also helps you to correlate logs with other telemetry signals such as metrics, traces or profiles, by viewing them side-by-side.

The results of log queries are displayed as individual log lines and as a graph showing the logs volume for the selected time period.

Logs volume

When working with data sources that support a full range logs volume, Explore automatically displays a graph showing the log distribution for all the entered log queries. This feature is currently supported by Elasticsearch and Loki data sources.

Note: In Loki, this full range log volume is rendered by a metric query which can be expensive depending on time range queried. This query can be particularly challenging to process for smaller Loki installations. To mitigate this, we recommend using a proxy like nginx in front of Loki to set a custom timeout (for example, 10 seconds) for these queries. Log volume histogram queries can be identified by looking for queries with the HTTP header `X-Query-Tags`

with value `Source=logvolhist`; these headers are added by Grafana to all log volume histogram queries.

If the data source does not support loading the full range logs volume, the logs model calculates a time series by counting log rows and organizing them into buckets based on an automatically calculated time interval. The timestamp of the first log row is used to anchor the start of the logs volume in the results. The end of the time series is anchored to the time picker's **To** range. This way, you can still analyze and visualize log data efficiently even when the data source doesn't offer full range support.

Logs

In the following sections, you will find detailed explanations of how to visualize and interact with individual logs in Explore.

Logs navigation

Logs navigation, at the right side of the log lines, can be used to easily request additional logs. You can do this by clicking the **Older logs** button at the bottom of the navigation. This is especially useful when you reach the line limit and you want to see more logs. Each request that is run from the navigation is then displayed in the navigation as separate page. Every page shows `from` and `to` timestamps of the incoming log lines. You can see previous results by clicking on each page. Explore caches the last five requests run from the logs navigation, so you're not re-running the same queries when clicking on the pages, saving time and resources.

Visualization options

You can customize how logs are displayed and select which columns are shown.

Option	Description
Time	Shows or hides the time column. This is the timestamp associated with the log line as reported from the data source.
Unique labels	Shows or hides the unique labels column that includes only non-common labels. All common labels are displayed above.
Wrap lines	Set this to `true` if you want the display to use line wrapping. If set to `false`, it will result in horizontal scrolling.
Prettify JSON	Set this to `true` to pretty print all JSON logs. This setting does not affect logs in any format other than JSON.
Deduplication	Log data can be very repetitive and Explore can help by hiding duplicate log lines. There are a few different deduplication algorithms that you can use **Exact** matches are done on the whole line except for date fields. **Numbers** matches are done on the line after stripping out numbers such as durations, IP addresses, and so on. **Signature** is the most aggressive deduplication as it strips all letters and numbers and matches on the remaining whitespace and punctuation.
Display results order	You can change the order of received logs from the default descending order (newest first) to ascending order (oldest first).

Download log lines

To download log results in either `txt` or `json` format, simply use the **Download** button. This feature allows you to save the log data for further analysis or to share it with others in a convenient and accessible format.

Log result meta information

Above the received log lines you can find essential meta information, including:

- **Number of received logs**: Indicates the total count of logs received for the current query or time range.
- **Error**: Displays possible error in your log results
- **Common labels**: Shows common labels.
- **Total bytes processed**: Represents the cumulative size of the log data processed in bytes.

Note: The availability of certain meta information may depend on the data source, and as a result, you may only see some of these details for specific data sources.

Escaping newlines

Explore automatically detects some incorrectly escaped sequences in log lines, such as newlines (`\n`, `\r`) or tabs (`\t`). When it detects such sequences, Explore provides an “Escape newlines” option.

To automatically fix incorrectly escaped sequences that Explore has detected:

1. Click “Escape newlines” to replace the sequences.
2. Manually review the replacements to confirm their correctness.

Explore replaces these sequences. When it does so, the option will change from “Escape newlines” to “Remove escaping”. Evaluate the changes as the parsing may not be accurate based on the input received. You can revert the replacements by clicking “Remove escaping”.

Log level

For the logs where a `level` label is specified, we use the value of this label to determine the log level and update color of each log line accordingly. If the log doesn’t have specified level label, we try to find out if its content matches any of the supported expressions (see below for more information). The log level is always determined by the first match. In the case where Grafana is not able to infer a log level field, it will be visualized with an unknown log level.

> **Tip:** If you use a Loki data source and the “level” is part of your log line, you can use parsers (JSON, logfmt, regex,..) to extract the level information into a level label that is used to determine the level value. This will allow the histogram to show the various log levels as separate bars.

Supported log levels and mapping of log level abbreviation and expressions:

Log level	Color	Supported expressions
critical	purple	emerg, fatal, alert, crit, critical
error	red	err, eror, error
warning	yellow	warn, warning
info	green	info, information, informational, notice
debug	blue	dbug, debug
trace	light blue	trace
unknown	grey	*

Highlight searched words

When your query includes specific words or expressions to search for, Explore will highlight these in the log lines for better visibility. This highlighting feature makes it easier to identify and focus on the relevant content in your logs.

Note: The ability to highlight search words may vary depending on the data source. For some data sources, the highlighting of search words may not be available.

Log details view

In Explore, each log line has an expandable section called **Log details** that can be opened by clicking on the log line. The Log details view provides additional information and exploration options in the form of **Fields** and **Links** attached to the log lines, enabling a more robust interaction and analysis.

Fields

Within the Log details view, you have the ability to filter displayed fields in two ways: a positive filter (to focus on an specific field) and a negative filter (to exclude certain fields). These filters will update the corresponding query that produced the log line, adding equality and inequality expressions accordingly. If the data source has support, as it's the case for Loki and Elasticsearch, log details will check if the field is already present in the current query showing and active state (for positive filters only), allowing you to toggle it off the query, or changing the filter expression from positive to negative.

Additionally, you can select a subset of fields to visualize in the logs list instead of the complete log line by clicking on the eye icon. Finally, each field has a stats icon to display ad-hoc statistics in relation to all displayed logs.

Links

Grafana offers the functionality of data links or correlations, enabling you to convert any part of a log message into an internal or external link. These links can be used to navigate to related data or external resources, providing a seamless and convenient way to explore further information.

Data link in Explore

Log context

Log context is a feature that allows you to display additional lines of context surrounding a log entry that matches a particular search query. This can be helpful in understanding the log entry's context, and is similar to the `-C` parameter in the `grep` command.

You may encounter long lines of text that make it difficult to read and analyze the context around each log entry. This is where the **Wrap lines** toggle can come in handy. By enabling this toggle, Grafana will automatically wrap long lines of text so that they fit within the visible width of the viewer. This can make it easier to read and understand the log entries.

The **Open in split view** button allows you to execute the context query for a log entry in a split screen in the Explore view. Clicking this button will open a new Explore pane with the context query displayed alongside the log entry, making it easier to analyze and understand the surrounding context.

The log context query can also be opened in a new browser tab by pressing the Cmd/Ctrl button while clicking on the button to open the context modal. When opened in a new tab, the previously selected filters are applied as well.

Copy log line

You can easily copy the content of a selected log line to your clipboard by clicking on the `Copy log line` button.

Copy link to log line

Linking of log lines in Grafana allows you to quickly navigate to specific log entries for precise analysis. By clicking the **Copy shortlink** button for a log line, you can generate and copy a short URL that provides direct access to the exact log entry within an absolute time range. When you open the link, Grafana will automatically scroll to the corresponding log line and highlight it with a blue background, making it easy to identify and focus on the relevant information.

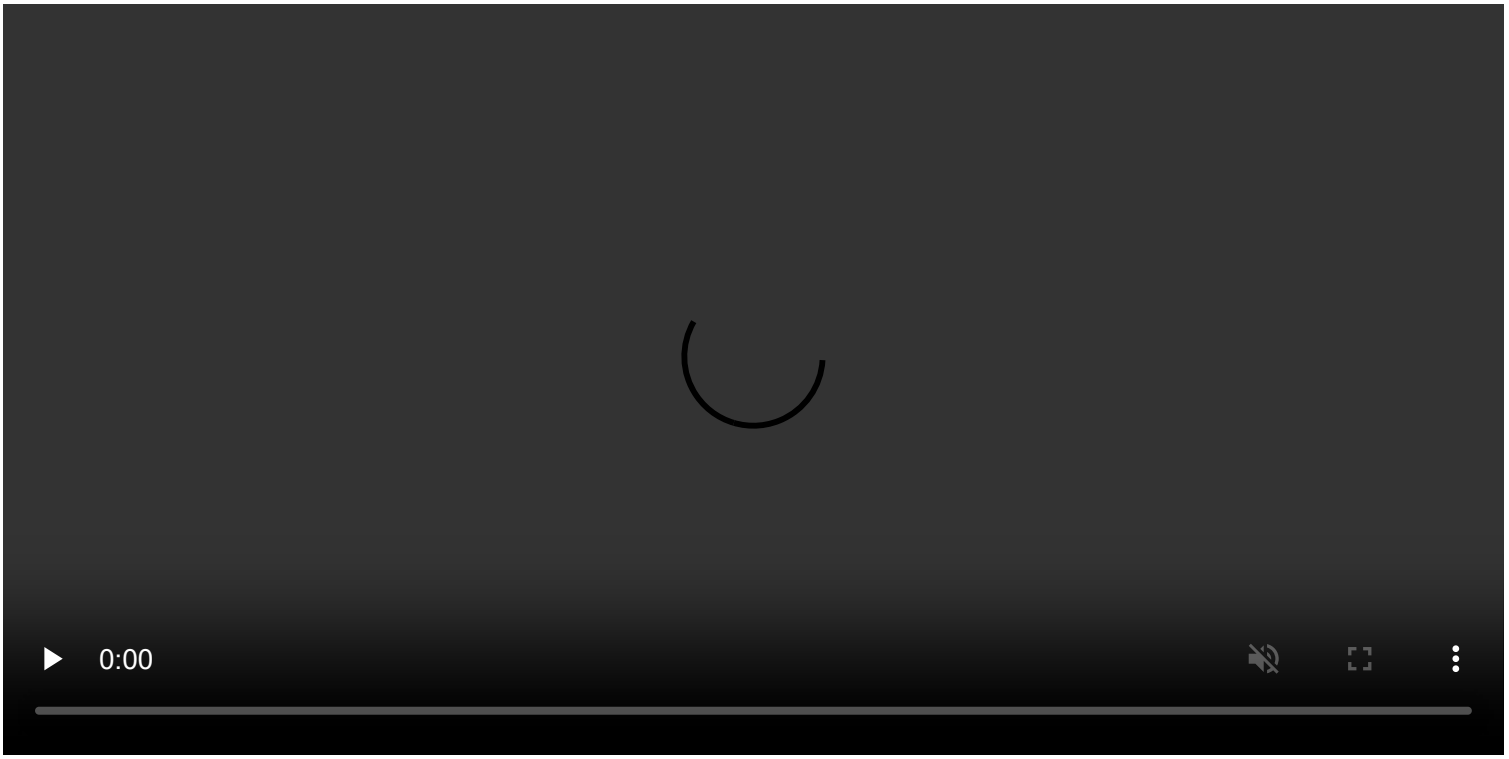
Note: This is currently only supported in Loki and other data sources that provide an `id` field.

Live tailing

To view real-time logs from supported data sources, you can leverage the Live tailing feature in Explore.

1. Click the **Live** button in the Explore toolbar to switch to Live tail view.
2. While in Live tail view, new logs will appear from the bottom of the screen, and they will have a fading contrasting background, allowing you to easily track what's new.
3. If you wish to pause the Live tailing and explore previous logs without any interruptions, you can do so by clicking the **Pause** button or simply scrolling through the logs view.
4. To clear the view and remove all logs from the display, click on the **Clear logs** button. This action will reset the log view and provide you with a clean slate to continue your log analysis.
5. To resume Live tailing and continue viewing real-time logs, click the **Resume** button.
6. If you want to exit Live tailing and return to the standard Explore view, click the **Stop** button.

Using the Live tailing feature, you can keep a close eye on the latest logs as they come in, making it easier to monitor real-time events and detect issues promptly.



Logs sample

If the selected data source implements logs sample, and supports both log and metric queries, then for metric queries you will be able to automatically see samples of log lines that contributed to visualized metrics. This feature is currently supported by Loki data sources.

Switch from metrics to logs

If you are coming from a metrics data source that implements `DataSourceWithQueryExportSupport` (such as Prometheus) to a logging data source that supports `DataSourceWithQueryImportSupport` (such as Loki), then it will keep the labels from your query that exist in the logs and use those to query the log streams.

For example, the following Prometheus query `grafana_alerting_active_alerts{job="grafana"}` after switching to the Loki data source, will change to `{job="grafana"}`. This will return a chunk of logs in the selected time range that can be grepped/text searched.

Tracing in Explore

You can use Explore to query and visualize traces from tracing data sources.

Supported data sources are:

- Tempo (supported ingestion formats: OpenTelemetry, Jaeger, and Zipkin)
- Jaeger
- Zipkin
- X-Ray
- Azure Monitor Application Insights

For information on how to configure queries for the data sources listed above, refer to the documentation for specific data source.

Query editor

You can query and search tracing data using a data source's query editor.

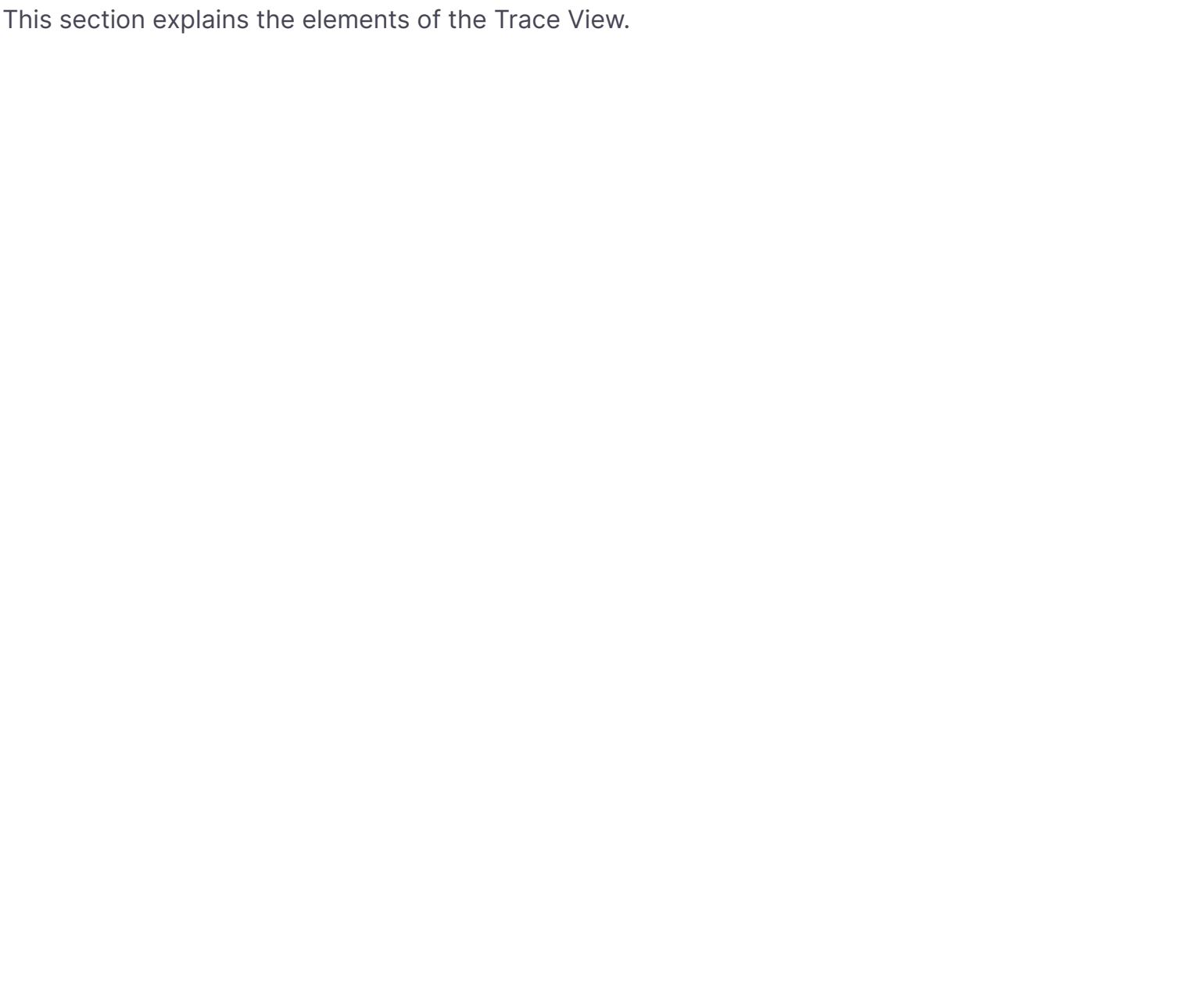
Each data source can have it's own query editor. The query editor for the Tempo data source is slightly different than the query editor for the Jaeger data source.

For information on querying each data source, refer to their documentation:

- Tempo query editor
- Jaeger query editor
- Zipkin query editor
- Azure Monitor Application Insights query editor

Trace view

This section explains the elements of the Trace View.



Screenshot of the trace view

Header

Screenshot of the trace view header

- Header title: Shows the name of the root span and trace ID.
- Search: Highlights spans containing the searched text.
- Metadata: Various metadata about the trace.

Minimap

Screenshot of the trace view minimap

Shows condensed view or the trace timeline. Drag your mouse over the minimap to zoom into smaller time range. Zooming will also update the main timeline, so it is easy to see shorter spans. Hovering over the minimap, when zoomed, will show Reset Selection button which resets the zoom.

Span filters

Using span filters, you can filter your spans in the trace timeline viewer. The more filters you add, the more specific are the filtered spans.

You can add one or more of the following filters:

- Resource service name
- Span name
- Duration
- Tags (which include tags, process tags, and log fields)

To only show the spans you have matched, you can press the `Show matches only` toggle.

Timeline

Screenshot of the trace view timeline

Shows list of spans within the trace. Each span row consists of these components:

- Expand children button: Expands or collapses all the children spans of selected span.
- Service name: Name of the service logged the span.
- Operation name: Name of the operation that this span represents.
- Span duration bar: Visual representation of the operation duration within the trace.

Clicking anywhere on the span row shows span details.

Span details

Screenshot of the trace view span details

- Operation name.
- Span metadata.
- Tags: Any tags associated with this span.
- Process metadata: Metadata about the process that logged this span.
- Logs: List of logs logged by this span and associated key values. In case of Zipkin logs section shows Zipkin annotations.

Trace to logs

You can navigate from a span in a trace view directly to logs relevant for that span. This feature is available for Tempo, Jaeger, and Zipkin data sources. Refer to their relevant documentation for configuration instructions.

Screenshot of the trace view in Explore with icon next to the spans

Click the document icon to open a split view in Explore with the configured data source and query relevant logs for the span.

Trace to metrics

> **NOTE**
>
> This feature is currently in beta and behind the `traceToMetrics` feature toggle.

You can navigate from a span in a trace view directly to metrics relevant for that span. This feature is available for Tempo, Jaeger, and Zipkin data sources. Refer to their relevant documentation for configuration instructions.

Trace to profiles

Using Trace to profiles, you can use Grafana's ability to correlate different signals by adding the functionality to link between traces and profiles. Refer to the relevant documentation for configuration instructions.

Node graph

You can optionally expand the node graph for the displayed trace. Depending on the data source, this can show spans of the trace as nodes in the graph, or as some additional context like service graph based on the current trace.

Screenshot of the node graph

Service Graph

The Service Graph visualizes the span metrics (traces data for rates, error rates, and durations (RED)) and service graphs. Once the requirements are set up, this pre-configured view is immediately available.

For more information, refer to the Service Graph view section of the Tempo data source page and the service graph view page in the Tempo documentation.

Screenshot of the Service Graph view

Data API

This visualization needs a specific shape of the data to be returned from the data source in order to correctly display it.

Data source needs to return data frame and set `frame.meta.preferredVisualisationType = 'trace'`.

Data frame structure

Required fields:

Field name	Type	Description
traceID	string	Identifier for the entire trace. There should be only one trace in the data frame.
spanID	string	Identifier for the current span. SpanIDs should be unique per trace.
parentSpanID	string	SpanID of the parent span to create child parent relationship in the trace view. Can be `undefined` for root span without a parent.

Field name	Type	Description
serviceName	string	Name of the service this span is part of.
serviceTags	TraceKeyValuePair[]	List of tags relevant for the service.
startTime	number	Start time of the span in millisecond epoch time.
duration	number	Duration of the span in milliseconds.

Optional fields:

Field name	Type	Description
logs	TraceLog[]	List of logs associated with the current span.
tags	TraceKeyValuePair[]	List of tags associated with the current span.
warnings	string[]	List of warnings associated with the current span.
stackTraces	string[]	List of stack traces associated with the current span.
errorIconColor	string	Color of the error icon in case span is tagged with `error: true`.

For details about the types see TraceSpanRow, TraceKeyValuePair and TraceLog.

Grafana Explore Metrics

Grafana Explore Metrics is a query-less experience for browsing **Prometheus-compatible** metrics. Quickly find related metrics with just a few simple clicks, without needing to write PromQL queries to retrieve metrics.

CAUTION

Explore Metrics is currently in public preview. Grafana Labs offers limited support, and breaking changes might occur prior to the feature being made generally available.

With Explore Metrics, you can:

- easily slice and dice metrics based on their labels, so you can immediately see anomalies and identify issues
- see the right visualization for your metric based on its type (gauge vs. counter, for example) without building it yourself
- surface other metrics relevant to the current metric
- “explore in a drawer” - expand a drawer over a dashboard with more content so you don’t lose your place
- view a history of user steps when navigating through metrics and their filters

You can access Explore Metrics either as a standalone experience or as part of Grafana dashboards.

Standalone experience

To access Explore Metrics as a standalone experience:

1. Click the arrow next to **Explore** in the Grafana left-side menu and click **Metrics**. You are taken to an overview page that shows recent metrics, bookmarks, and the option to select a new

metric exploration.

2. To get started with a new exploration, click **+ New metric exploration**.
3. Select **Prometheus** or any Prometheus-compatible data source available in the drop-down menu under **Data source**.
4. Click **+ Add label** to select a label-value pair from the drop-down menu. You can add multiple label-value pairs. A label type will appear above the selected label with a drop-down list of options from which to choose. For example, if you select the label `container` a drop-down list of available containers appears.
5. You can also search for metrics using keywords under **Search metrics** in the search bar.
6. Use the time picker to select a date and time range from the drop-down menu or use an absolute time range.
7. Click the down arrow next to the **Refresh** icon to set a refresh rate from the drop-down menu. The default is `Off`.
8. Click the **Settings** icon and toggle **Always keep selected metric graph in-view** to keep your main graph always in view on the Breakdown drill-down tab.

The **History** button in the upper left corner tracks every step navigating through metric exploration.

Metrics exploration

To further explore a metric, click **Select** in the upper right corner of the metric visualization.

- The **Overview** tab provides a description for each metric, as well as the metric `type` and `unit` associated with the metric. It also provides a list of labels associated with the metric. Click on any label to view drill-down visualizations.
- The **Breakdown** tab depicts time series visualizations for each of the label-value pairs for the selected metric. You can further drill down on each label and click **Add to filter** to add the label/value pair into your filters. You can also change the **View** from grid to rows.
- The **Related metrics** tab depicts related metrics with relevant key words. You can repeat the drill down process for any related metric. Toggle **SHow previews** to preview visualizations.

Once you have gathered your metrics exploration data you can:

- Click the **Explore** icon on the right side to open the graph in Explore, where you can modify the query or add the graph to a dashboard or incident.
- Click the **Share** icon on the right side to copy the metric drill down URL to the clipboard so it can be shared.
- Click the **Star** icon on the right side to bookmark and save the metrics exploration.

Dashboard experience

To access Explore Metrics via a dashboard:

1. Navigate to your dashboard.
2. Select a time series panel.
3. Click the panel menu in the upper right and select **Explore Metrics**. If there are multiple metrics, click on the one you want to explore.
4. You will see a slide out drawer with the Metrics Experience, starting with the drill down. You can access the standalone experience by clicking **Open** in the upper right.

Correlations Editor in Explore

NOTE

The Explore editor is available in 10.1 and later versions. In the editor, transformations is available in Grafana 10.3 and later versions.

Correlations allow users to build a link between any two data sources. For more information about correlations in general, please see the correlations topic in the administration page.

Create a correlation

1. In Grafana, navigate to the Explore page.
2. Select a data source that you would like to be the source data source for a new correlation.
3. Run a query producing data in a supported visualization.
4. Click **+ Add** in the top toolbar and select **Add correlation** (you can also select **Correlations Editor** from the Command Palette).
5. Explore is now in Correlations Editor mode indicated by a blue border and top bar. You can exit Correlations Editor by clicking **Exit** in the top bar.
6. You can now create the following new correlations for the visualization with links that are attached to the data that you can use to build a new query:
   - Logs: links are displayed next to field values inside log details for each log row
   - Table: every table cell is a link
7. Click on a link to add a new correlation. Links are associated with a field that is used as a result field of a correlation.

8. In the split view that opens, use the right pane to set up the target query source of the correlation.
9. Build a target query using variables syntax with variables from the list provided at the top of the pane. The list contains sample values from the selected data row.
10. Provide a label and description (optional). A label will be used as the name of the link inside the visualization and can contain variables.
11. Provide transformations (optional; see below for details).
12. Click **Save** in the top toolbar to save the correlation and exit Correlations Editor mode. The link used to create the correlation is replaced with a data link in each row. When the link is clicked, the query you defined will run in another pane, with the variables replaced dynamically with the values from the selected row.

Transformations

Transformations allow you to extract values that exist in a field with other data. For example, using a transformation, you can extract one portion of a log line to use in a correlation. For more details on transformations in correlations, see Correlations.

After clicking one of the generated links in the editor mode, you can add transformations by clicking **Add transformation** in the Transformations dropdown menu.

You can use a transformation in your correlation with the following steps:

1. Select a field to apply the transformation to. Select the portion of the field that you want to use for the transformation. For example, a log line. Once selected, the value of this field will be used to assist you in building the transformation.
2. Select the type of the transformation. See correlations for the options and relevant settings.
3. Based on your selection, you might see one or more variables populate, or you might need to provide more specifications in options that are displayed.
4. Select **Add transformation to correlation** to add the specified variables to the list of available variables.

Notes for regular expressions

For regular expressions in this dialog box, the `mapValue` referred to in other documentation is called `Variable Name` here. Grafana highlights any text that matches the expression in the field value. Use regular expression capture groups to select what portion of the match should be extracted. When a valid regular expression is provided, the variable and the value of that variable appear below the `Variable Name` field.

Correlations examples

The following examples show how to create correlations using the Correlations Editor in Explore. If you'd like to follow these examples, make sure to set up a test data source.

Create a text to graph correlation

This example shows how to create a correlation using Correlations Editor in Explore.

Correlations allow you to use results of one query to run a new query in any data source. In this example, you will run a query that renders tabular data. The data will be used to run a different query that yields a graph result.

To follow this example, make sure you have set up a test data source.

1. In Grafana, navigate to **Explore**.
2. Select the **test data source** from the dropdown menu at the top left of the page.
3. Click **+ Add** in the dropdown menu to the right and select **Add correlation**.
4. Explore is now in Correlations Editor mode, indicated by a blue border.
5. Select the following scenario from the scenario dropdown menu: **CSV File**.
6. Select the file, **population_by_state.csv**. Each cell is a link that you can click on to begin creating a new correlation.

Selecting the source of a correlation

7. Click on any cell in the `State` column to create a new correlation that attaches a data link to that entry. For example, select "California".
8. In the split view, select the same data source you selected in the left pane. The helper above the query editor contains all available variables you can use the target query. Variables contain all data fields (table columns) from the selected row.
9. In the **Scenario** menu, select **CSV Metric Values**. The `String Input` field in the Query editor provides variables with population values for each year: `${1980},${2000},${2020}`. This will generate a graph using variable values.
10. In the Query Editor **Alias** field, enter "${State}".

Setting up the target of a correlation

Run a query to see that it produces a graph using sample values from the variables.

11 Click **Save** to save the correlation and exit the Correlations Editor.

After the correlation is saved, Explore will rerun the query in the left pane. By clicking a state name, the query on the right is rerun with values from the row being inserted into the CSV, thus changing the graph. The query is rerun with updated values every time you click on a state name.

Result of clicking on a data link

You can apply the same steps to any data source. Correlations allow you to create links in visualizations to run dynamic queries based on selected data. In this example we used data returned by a query to build a new query generating different visualization using the same data source. However, you can create correlations between any data sources to create custom exploration flows.

Create a logs to table correlation

In this example, you will create a correlation to demonstrate how to use transformations to extract values from the log line and another field.

To follow this example, make sure you have set up a test data source.

1. In Grafana, navigate to **Explore**.
2. Select the **test data source** from the dropdown menu at the top left of the page.
3. Click **+ Add** in the dropdown menu to the right and select **Add correlation**.
4. Explore is now in Correlations Editor mode, indicated by a blue border.
5. In the **Scenario** menu, select **Logs**.
6. Expand a log line to see the correlation links. Select `Correlate with hostname`.
7. Explore opens in split view. Select the same data source you selected in the left pane. The helper above the query editor contains all available variables you can use the target query.
8. Expand the transformations section, and click **Add transformation**.
9. In the **Field** dropdown menu, select **message**. The log line shows up as example data.
10. Under **Type**, select **Logfmt**. This populates the list of variables.
11. Click **Add transformation to correlation**.
12. Click **Add transformation** again and under **Field**, select **hostname**.
13. Under **Type**, select **Regular expression**.
14. Under **Expression**, enter the following: `-([0-9]\*)` This selects any numbers to the right of the dash.
15. Under **Variable Name**, enter the following: hostNumber This populates the list of variables.
16. Click **Add transformation to correlation** to add it to the other variables.
17. In the data source editor, open the **Scenario** dropdown menu and select **CSV Content**.
18. In the text box below, provide the following and save the correlation:

```csv
time,msg,hostNumber,status
${time},${msg},${hostNumber},${status}
```

This closes the split view and reruns the left query. Expand any log line to see the correlation button. Clicking the correlation button opens the split view with the `time` (a field), `msg` (extracted with `logfmt` from the log line), host number (extracted with `regex` from the `hostname`) and the `status` (extracted with `logfmt` from the log line).

Inspector in Explore

The inspector helps you understand and troubleshoot your queries. You can inspect the raw data, export that data to a comma-separated values (CSV) file, export log results in TXT format, and view query requests.

Inspector UI

The inspector has following tabs:

- **Stats tab -** Shows how long your query takes and how much it returns.
- **Query tab -** Shows you the requests to the server sent when Grafana queries the data source.
- **JSON tab -** Allows you to view and copy the data JSON and data frame structure JSON.
- **Data tab -** Shows the raw data returned by the query.
- **Error tab -** Shows the error. Only visible when query returns error.

Inspector tasks

You can perform a variety of tasks in the Explore inspector.

Open the Inspector

1. Run the query you would like to inspect.
2. Click the **Inspector** button.

The inspector pane opens on the bottom of the screen.

Inspect raw query results

You can view raw query results, that is the data returned by the query in a table.

In the **Inspector** tab, click the **Data** tab.

For multiple queries or for queries multiple nodes, there are additional options.

- **Show data frame:** Select the result set data you want to view.
- **Series joined by time:** View the raw data from all of your queries at once, one result set per column. You can click a column heading to sort the data.

Download raw query results as CSV

Grafana generates a CSV file in your default browser download location. You can open it in the viewer of your choice.

1. In the **Inspector** tab, get raw query results by following the instructions described in Inspect raw query results.
2. Refine query settings until you can see the raw data that you want to export.
3. Click **Download CSV**.

In order to download a CSV file specifically formatted for Excel, expand **Data options** and then enable the **Download for Excel** toggle before you click the **Download CSV** option.

Download log results as TXT

Grafana generates a TXT file in your default browser download location. You can open it in the viewer of your choice.

1. Open the inspector.
2. Inspect the log query results as described above.
3. Click **Download logs**.

Download trace results

Based on the data source type, Grafana generates a JSON file for the trace results in one of the supported formats: Jaeger, Zipkin, or OTLP formats.

1. Open the inspector.
2. Inspect the trace query results as described above.
3. Click **Download traces**.

Inspect query performance

The Stats tab displays statistics that tell you how long your query takes, how many queries you send, and the number of rows returned. This information can help you troubleshoot your queries, especially if any of the numbers are unexpectedly high or low.

1. Open the inspector.
2. Navigate to the **Stats** tab.

Statistics are displayed in read-only format.

View JSON model

You can explore and export data as well as data frame JSON models.

1. In the Inspector panel, click the **JSON** tab.
2. From the Select source dropdown, choose one of the following options:
   - **Data -** Displays a JSON object representing the data that was returned to Explore.
   - **DataFrame structure -** Displays the raw result set.
3. You can expand or collapse portions of the JSON to view separate sections. You can also click the **Copy to clipboard** option to copy JSON body and paste it into another application.

View raw request and response to data source

1. Open the panel inspector and then click the **Query** tab.
2. Click **Refresh**.

Grafana sends the query to the server and displays the result. You can drill down on specific portions of the query, expand or collapse all of it, or copy the data to the clipboard to use in other applications.

Alerting

Grafana Alerting allows you to learn about problems in your systems moments after they occur.

Monitor your incoming metrics data or log entries and set up your Alerting system to watch for specific events or circumstances and then send notifications when those things are found.

In this way, you eliminate the need for manual monitoring and provide a first line of defense against system outages or changes that could turn into major incidents.

Using Grafana Alerting, you create queries and expressions from multiple data sources — no matter where your data is stored — giving you the flexibility to combine your data and alert on your metrics and logs in new and unique ways. You can then create, manage, and take action on your alerts from a single, consolidated view, and improve your team's ability to identify and resolve issues quickly.

Grafana Alerting is available for Grafana OSS, Grafana Enterprise, or Grafana Cloud. With Mimir and Loki alert rules you can run alert expressions closer to your data and at massive scale, all managed by the Grafana UI you are already familiar with.

Watch this video to learn more about Grafana Alerting:

Refer to Manage your alert rules for current instructions.

Key features and benefits

One page for all alerts

A single Grafana Alerting page consolidates both Grafana-managed alerts and alerts that reside in your Prometheus-compatible data source in one single place.

Multi-dimensional alerts

Alert rules can create multiple individual alert instances per alert rule, known as multi-dimensional alerts, giving you the power and flexibility to gain visibility into your entire system with just a single alert rule. You do this by adding labels to your query to specify which component is being monitored and generate multiple alert instances for a single alert rule. For example, if you want to monitor each server in a cluster, a multi-dimensional alert will alert on each CPU, whereas a standard alert will alert on the overall server.

Route alerts

Route each alert instance to a specific contact point based on labels you define. Notification policies are the set of rules for where, when, and how the alerts are routed to contact points.

Silence alerts

Silences stop notifications from getting created and last for only a specified window of time. Silences allow you to stop receiving persistent notifications from one or more alert rules. You can also partially pause an alert based on certain criteria. Silences have their own dedicated section for better organization and visibility, so that you can scan your paused alert rules without cluttering the main alerting view.

Mute timings

A mute timing is a recurring interval of time when no new notifications for a policy are generated or sent. Use them to prevent alerts from firing a specific and reoccurring period, for example, a regular maintenance period.

Similar to silences, mute timings do not prevent alert rules from being evaluated, nor do they stop alert instances from being shown in the user interface. They only prevent notifications from being created.

Design your Alerting system

Monitoring complex IT systems and understanding whether everything is up and running correctly is a difficult task. Setting up an effective alert management system is therefore essential to inform you when things are going wrong before they start to impact your business outcomes.

Designing and configuring an alert management set up that works takes time.

Here are some tips on how to create an effective alert management set up for your business:

Which are the key metrics for your business that you want to monitor and alert on?

- Find events that are important to know about and not so trivial or frequent that recipients ignore them.
- Alerts should only be created for big events that require immediate attention or intervention.
- Consider quality over quantity.

Which type of Alerting do you want to use?

- Choose between Grafana-managed Alerting or Grafana Mimir or Loki-managed Alerting; or both.

How do you want to organize your alerts and notifications?

- Be selective about who you set to receive alerts. Consider sending them to whoever is on call or a specific Slack channel.
- Automate as far as possible using the Alerting API or alerts as code (Terraform).

How can you reduce alert fatigue?

- Avoid noisy, unnecessary alerts by using silences, mute timings, or pausing alert rule evaluation.
- Continually tune your alert rules to review effectiveness. Remove alert rules to avoid duplication or ineffective alerts.
- Think carefully about priority and severity levels.
- Continually review your thresholds and evaluation rules.

Useful links

- Introduction to Alerting

Introduction to Alerting

Whether you're just starting out or you're a more experienced user of Grafana Alerting, learn more about the fundamentals and available features that help you create, manage, and respond to alerts; and improve your team's ability to resolve issues quickly. For a hands-on introduction, refer to our tutorial to get started with Grafana Alerting.

Principles

In Prometheus-based alerting systems, you have an alert generator that creates alerts and an alert receiver that receives alerts. For example, Prometheus is an alert generator and is responsible for evaluating alert rules, while Alertmanager is an alert receiver and is responsible for grouping, inhibiting, silencing, and sending notifications about firing and resolved alerts.

Grafana Alerting is built on the Prometheus model of designing alerting systems. It has an internal alert generator responsible for scheduling and evaluating alert rules, as well as an internal alert receiver responsible for grouping, inhibiting, silencing, and sending notifications. Grafana doesn't use Prometheus as its alert generator because Grafana Alerting needs to work with many other data sources in addition to Prometheus. However, it does use Alertmanager as its alert receiver.

Alerts are sent to the alert receiver where they are routed, grouped, inhibited, silenced and notified. In Grafana Alerting, the default alert receiver is the Alertmanager embedded inside Grafana, and is referred to as the Grafana Alertmanager. However, you can use other Alertmanagers too, and these are referred to as External Alertmanagers.

The following diagram gives you an overview of Grafana Alerting and introduces you to some of the fundamental features that are the principles of how Grafana Alerting works.

How Alerting works

Fundamentals

Alert rules

An alert rule is a set of criteria that determine when an alert should fire. It consists of one or more queries and expressions, a condition which needs to be met, an interval which determines how often the alert rule is evaluated, and a duration over which the condition must be met for an alert to fire.

Alert rules are evaluated over their interval, and each alert rule can have zero, one, or any number of alerts firing at a time. The state of the alert rule is determined by its most “severe” alert, which can be one of Normal, Pending, or Firing. For example, if at least one of an alert rule’s alerts are firing then the alert rule is also firing. The health of an alert rule is determined by the status of its most recent evaluation. These can be OK, Error, and NoData.

A very important feature of alert rules is that they support custom annotations and labels. These allow you to instrument alerts with additional metadata such as summaries and descriptions, and add additional labels to route alerts to specific notification policies.

Alerts

Alerts are uniquely identified by sets of key/value pairs called Labels. Each key is a label name and each value is a label value. For example, one alert might have the labels `foo=bar` and another alert might have the labels `foo=baz`. An alert can have many labels such as `foo=bar,bar=baz` but it cannot have the same label twice such as `foo=bar,foo=baz`. Two alerts cannot have the same labels either, and if two alerts have the same labels such as `foo=bar,bar=baz` and `foo=bar,bar=baz` then one of the alerts will be discarded. Alerts are resolved when the condition in the alert rule is no longer met, or the alert rule is deleted.

In Grafana Managed Alerts, alerts can be in Normal, Pending, Alerting, No Data or Error states. In Datasource Managed Alerts, such as Mimir and Loki, alerts can be in Normal, Pending and Alerting, but not NoData or Error.

Contact points

Contact points determine where notifications are sent. For example, you might have a contact point that sends notifications to an email address, to Slack, to an incident management system (IRM) such as Grafana OnCall or Pagerduty, or to a webhook.

The notifications that are sent from contact points can be customized using notification templates. You can use notification templates to change the title, message, and structure of the notification. Notification templates are not specific to individual integrations or contact points.

Notification policies

Notification policies group alerts and then route them to contact points. They determine when notifications are sent, and how often notifications should be repeated.

Alerts are matched to notification policies using label matchers. These are human-readable expressions that assert if the alert's labels exactly match, do not exactly match, contain, or do not contain some expected text. For example, the matcher `foo=bar` matches alerts with the label `foo=bar` while the matcher `foo=~[a-zA-Z]+` matches alerts with any label called foo with a value that matches the regular expression `[a-zA-Z]+`.

By default, an alert can only match one notification policy. However, with the `continue` feature alerts can be made to match any number of notification policies at the same time. For more information on notification policies, see fundamentals of Notification Policies.

Silences and mute timings

Silences and mute timings allow you to pause notifications for specific alerts or even entire notification policies. Use a silence to pause notifications on an ad-hoc basis, such as during a maintenance window; and use mute timings to pause notifications at regular intervals, such as evenings and weekends.

Provisioning

You can create your alerting resources (alert rules, notification policies, and so on) in the Grafana UI; configmaps, files and configuration management systems using file-based provisioning; and in Terraform using API-based provisioning.

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Grafana Cloud | Enterprise | Open source

Data sources and Grafana Alerting

There are a number of data sources that are compatible with Grafana Alerting. Each data source is supported by a plugin. You can use one of the built-in data sources listed below, use external data source plugins, or create your own data source plugin.

If you are creating your own data source plugin, make sure it is a backend plugin as Grafana Alerting requires this in order to be able to evaluate rules using the data source. Frontend data sources are not supported, because the evaluation engine runs on the backend.

Specifying `{ "alerting": true, “backend”: true }` in the plugin.json file indicates that the data source plugin is compatible with Grafana Alerting and includes the backend data-fetching code. For more information, refer to Build a data source backend plugin.

These are the data sources that are compatible with and supported by Grafana Alerting.

- AWS CloudWatch
- Azure Monitor
- Elasticsearch
- Google Cloud Monitoring
- Graphite
- InfluxDB
- Loki
- Microsoft SQL Server (MSSQL)
- MySQL
- Open TSDB
- PostgreSQL
- Prometheus
- Jaeger
- Zipkin

- Tempo
- Testdata

Useful links

- Grafana data sources

Alerting on numeric data

This topic describes how Grafana managed alerts are evaluated by the backend engine as well as how Grafana handles alerting on numeric rather than time series data.

Alert evaluation

Grafana managed alerts query the following backend data sources that have alerting enabled:

- built-in data sources or those developed and maintained by Grafana: `Graphite`, `Prometheus`, `Loki`, `InfluxDB`, `Elasticsearch`, `Google Cloud Monitoring`, `Cloudwatch`, `Azure Monitor`, `MySQL`, `PostgreSQL`, `MSSQL`, `OpenTSDB`, `Oracle`, and `Azure Monitor`
- community developed backend data sources with alerting enabled (`backend` and `alerting` properties are set in the plugin.json

Metrics from the alerting engine

The alerting engine publishes some internal metrics about itself. You can read more about how Grafana publishes internal metrics.

Metric Name	Type	Description
`grafana_alerting_alerts`	gauge	How many alerts by state
`grafana_alerting_request_duration`	histogram	Histogram of requests to the Alerting API
`grafana_alerting_active_configurations`	gauge	The number of active, non default Alertmanager configurations for grafana managed alerts
`grafana_alerting_rule_evaluations_total`	counter	The total number of rule evaluations
`grafana_alerting_rule_evaluation_failures_total`	counter	The total number of rule evaluation failures
`grafana_alerting_rule_evaluation_duration`	summary	The duration for a rule to execute
`grafana_alerting_rule_group_rules`	gauge	The number of rules

Alerting on numeric data

Among certain data sources numeric data that is not time series can be directly alerted on, or passed into Server Side Expressions (SSE). This allows for more processing and resulting efficiency within the data source, and it can also simplify alert rules. When alerting on numeric data instead of time series data, there is no need to reduce each labeled time series into a single number. Instead labeled numbers are returned to Grafana instead.

Tabular Data

This feature is supported with backend data sources that query tabular data:

- SQL data sources such as MySQL, Postgres, MSSQL, and Oracle.
- The Azure Kusto based services: Azure Monitor (Logs), Azure Monitor (Azure Resource Graph), and Azure Data Explorer.

A query with Grafana managed alerts or SSE is considered numeric with these data sources, if:

- The “Format AS” option is set to “Table” in the data source query.
- The table response returned to Grafana from the query includes only one numeric (e.g. int, double, float) column, and optionally additional string columns.

If there are string columns then those columns become labels. The name of column becomes the label name, and the value for each row becomes the value of the corresponding label. If multiple rows are returned, then each row should be uniquely identified their labels.

Example

For a MySQL table called “DiskSpace”:

Time	Host	Disk	PercentFree
2021-June-7	web1	/etc	3
2021-June-7	web2	/var	4
2021-June-7	web3	/var	8
...	...	...	...

You can query the data filtering on time, but without returning the time series to Grafana. For example, an alert that would trigger per Host, Disk when there is less than 5% free space:

sql Copy

```sql
SELECT Host, Disk, CASE WHEN PercentFree < 5.0 THEN PercentFree ELSE 0 END FROM (
  SELECT
      Host,
      Disk,
      Avg(PercentFree)
  FROM DiskSpace
  Group By
    Host,
    Disk
  Where __timeFilter(Time)
```

This query returns the following Table response to Grafana:

Host	Disk	PercentFree
web1	/etc	3
web2	/var	4
web3	/var	0

When this query is used as the **condition** in an alert rule, then the non-zero will be alerting. As a result, three alert instances are produced:

Labels	Status
{Host=web1,disk=/etc}	Alerting
{Host=web2,disk=/var}	Alerting
{Host=web3,disk=/var}	Normal

Labels and annotations

Labels and annotations contain information about an alert. Both labels and annotations have the same structure: a set of named values; however their intended uses are different. An example of label, or the equivalent annotation, might be `alertname="test"`.

The main difference between a label and an annotation is that labels are used to differentiate an alert from all other alerts, while annotations are used to add additional information to an existing alert.

For example, consider two high CPU alerts: one for `server1` and another for `server2`. In such an example we might have a label called `server` where the first alert has the label `server="server1"` and the second alert has the label `server="server2"`. However, we might also want to add a description to each alert such as `"The CPU usage for server1 is above 75%."`, where `server1` and `75%` are replaced with the name and CPU usage of the server (please refer to the documentation on templating labels and annotations for how to do this). This kind of description would be more suitable as an annotation.

Labels

Labels contain information that identifies an alert. An example of a label might be `server=server1`. Each alert can have more than one label, and the complete set of labels for an alert is called its label set. It is this label set that identifies the alert.

For example, an alert might have the label set `{alertname="High CPU usage",server="server1"}` while another alert might have the label set `{alertname="High CPU usage",server="server2"}`. These are two separate alerts because although their `alertname` labels are the same, their `server` labels are different.

The label set for an alert is a combination of the labels from the datasource, custom labels from the alert rule, and a number of reserved labels such as `alertname`.

Custom Labels

Custom labels are additional labels from the alert rule. Like annotations, custom labels must have a name, and their value can contain a combination of text and template code that is evaluated when an alert is fired. Documentation on how to template custom labels can be found here.

When using custom labels with templates it is important to make sure that the label value does not change between consecutive evaluations of the alert rule as this will end up creating large numbers of distinct alerts. However, it is OK for the template to produce different label values for different alerts. For example, do not put the value of the query in a custom label as this will end up creating a new set of alerts each time the value changes. Instead use annotations.

It is also important to make sure that the label set for an alert does not have two or more labels with the same name. If a custom label has the same name as a label from the datasource then it will replace that label. However, should a custom label have the same name as a reserved label then the custom label will be omitted from the alert.

Annotations

Annotations are named pairs that add additional information to existing alerts. There are a number of suggested annotations in Grafana such as `description`, `summary`, `runbook_url`, `dashboardUId` and `panelId`. Like custom labels, annotations must have a name, and their value can contain a combination of text and template code that is evaluated when an alert is fired. If an annotation contains template code, the template is evaluated once when the alert is fired. It is not re-evaluated, even when the alert is resolved. Documentation on how to template annotations can be found here.

How label matching works

Use labels and label matchers to link alert rules to notification policies and silences. This allows for a very flexible way to manage your alert instances, specify which policy should handle them, and which alerts to silence.

A label matchers consists of 3 distinct parts, the **label**, the **value** and the **operator**.

- The **Label** field is the name of the label to match. It must exactly match the label name.
- The **Value** field matches against the corresponding value for the specified **Label** name. How it matches depends on the **Operator** value.
- The **Operator** field is the operator to match against the label value. The available operators are:

Operator	Description
`=`	Select labels that are exactly equal to the value.
`!=`	Select labels that are not equal to the value.
`=~`	Select labels that regex-match the value.
`!~`	Select labels that do not regex-match the value.

If you are using multiple label matchers, they are combined using the AND logical operator. This means that all matchers must match in order to link a rule to a policy.

Example scenario

If you define the following set of labels for your alert:

```
{ foo=bar, baz=qux, id=12 }
```

then:

- A label matcher defined as `foo=bar` matches this alert rule.
- A label matcher defined as `foo!=bar` does *not* match this alert rule.
- A label matcher defined as `id=~[0-9]+` matches this alert rule.
- A label matcher defined as `baz!~[0-9]+` matches this alert rule.
- Two label matchers defined as `foo=bar` and `id=~[0-9]+` match this alert rule.

Exclude labels

You can also write label matchers to exclude labels.

Here is an example that shows how to exclude the label `Team`. You can choose between any of the values below to exclude labels.

Label	Operator	Value
`team`	`=`	`""`
`team`	`!~`	`.+`
`team`	`=~`	`^$`

Labels in Grafana Alerting

This topic explains why labels are a fundamental component of alerting.

- The complete set of labels for an alert is what uniquely identifies an alert within Grafana alerts.
- The Alertmanager uses labels to match alerts for silences and alert groups in notification policies.
- The alerting UI shows labels for every alert instance generated during evaluation of that rule.
- Contact points can access labels to dynamically generate notifications that contain information specific to the alert that is resulting in a notification.
- You can add labels to an alerting rule. Labels are manually configurable, use template functions, and can reference other labels. Labels added to an alerting rule take precedence in the event of a collision between labels (except in the case of Grafana reserved labels).

External Alertmanager Compatibility

Grafana's built-in Alertmanager supports both Unicode label keys and values. If you are using an external Prometheus Alertmanager, label keys must be compatible with their data model. This means that label keys must only contain **ASCII letters**, **numbers**, as well as **underscores** and match the regex `[a-zA-Z_][a-zA-Z0-9_]*`. Any invalid characters will be removed or replaced by the Grafana alerting engine before being sent to the external Alertmanager according to the following rules:

- `Whitespace` will be removed.
- `ASCII characters` will be replaced with `_`.
- `All other characters` will be replaced with their lower-case hex representation. If this is the first character it will be prefixed with `_`.

Example: A label key/value pair `Alert! 🔔="🔥"` will become `Alert_0x1f514="🔥"`.

Note If multiple label keys are sanitized to the same value, the duplicates will have a short hash of the original label appended as a suffix.

Grafana reserved labels

> **NOTE**
>
> Labels prefixed with `grafana_` are reserved by Grafana for special use. If a manually configured label is added beginning with `grafana_` it may be overwritten in case of collision. To stop the Grafana Alerting engine from adding a reserved label, you can disable it via the `disabled_labels` option in unified_alerting.reserved_labels configuration.

Grafana reserved labels can be used in the same way as manually configured labels. The current list of available reserved labels are:

Label	Description
grafana_folder	Title of the folder containing the alert.

Templating labels and annotations

You can use templates to include data from queries and expressions in labels and annotations. For example, you might want to set the severity label for an alert based on the value of the query, or use the instance label from the query in a summary annotation so you know which server is experiencing high CPU usage.

All templates should be written in text/template. Regardless of whether you are templating a label or an annotation, you should write each template inline inside the label or annotation that you are templating. This means you cannot share templates between labels and annotations, and instead you will need to copy templates wherever you want to use them.

Each template is evaluated whenever the alert rule is evaluated, and is evaluated for every alert separately. For example, if your alert rule has a templated summary annotation, and the alert rule has 10 firing alerts, then the template will be executed 10 times, once for each alert. You should try to avoid doing expensive computations in your templates as much as possible.

Examples

Rather than write a complete tutorial on text/template, the following examples attempt to show the most common use-cases we have seen for templates. You can use these examples verbatim, or adapt them as necessary for your use case. For more information on how to write text/template refer to the text/template documentation.

Print all labels, comma separated

To print all labels, comma separated, print the `$labels` variable:

```
{{ $labels }}
```

Copy

For example, given an alert with the labels `alertname=High CPU usage`, `grafana_folder=CPU alerts` and `instance=server1`, this would print:

```
alertname=High CPU usage, grafana_folder=CPU alerts, instance=server1
```
Copy

If you are using classic conditions then `$labels` will not contain any labels from the query. Refer to the $labels variable for more information.

Print all labels, one per line

To print all labels, one per line, use a `range` to iterate over each key/value pair and print them individually. Here `$k` refers to the name and `$v` refers to the value of the current label:

```
{{ range $k, $v := $labels -}}
{{ $k }}={{ $v }}
{{ end }}
```
Copy

For example, given an alert with the labels `alertname=High CPU usage`, `grafana_folder=CPU alerts` and `instance=server1`, this would print:

```
alertname=High CPU usage
grafana_folder=CPU alerts
instance=server1
```
Copy

If you are using classic conditions then `$labels` will not contain any labels from the query. Refer to the $labels variable for more information.

Print an individual label

To print an individual label use the `index` function with the `$labels` variable:

```
The host {{ index $labels "instance" }} has exceeded 80% CPU usage for the last 5 mi
```
Copy

For example, given an alert with the labels `instance=server1`, this would print:

```
The host server1 has exceeded 80% CPU usage for the last 5 minutes
```
Copy

If you are using classic conditions then `$labels` will not contain any labels from the query. Refer to the $labels variable for more information.

Print the value of a query

To print the value of an instant query you can print its Ref ID using the `index` function and the `$values` variable:

```
{{ index $values "A" }}
```

Copy

For example, given an instant query that returns the value 81.2345, this will print:

```
81.2345
```

Copy

To print the value of a range query you must first reduce it from a time series to an instant vector with a reduce expression. You can then print the result of the reduce expression by using its Ref ID instead. For example, if the reduce expression takes the average of A and has the Ref ID B you would write:

```
{{ index $values "B" }}
```

Copy

Print the humanized value of a query

To print the humanized value of an instant query use the `humanize` function:

```
{{ humanize (index $values "A").Value }}
```

Copy

For example, given an instant query that returns the value 81.2345, this will print:

```
81.234
```

Copy

To print the humanized value of a range query you must first reduce it from a time series to an instant vector with a reduce expression. You can then print the result of the reduce expression by using its Ref ID instead. For example, if the reduce expression takes the average of A and has the Ref ID B you would write:

```
{{ humanize (index $values "B").Value }}
```

Copy

Print the value of a query as a percentage

To print the value of an instant query as a percentage use the `humanizePercentage` function:

```
{{ humanizePercentage (index $values "A").Value }}
```

Copy

This function expects the value to be a decimal number between 0 and 1. If the value is instead a decimal number between 0 and 100 you can either divide it by 100 in your query or using a math

expression. If the query is a range query you must first reduce it from a time series to an instant vector with a reduce expression.

Set a severity from the value of a query

To set a severity label from the value of a query use an if statement and the greater than comparison function. Make sure to use decimals (`80.0`, `50.0`, `0.0`, etc) when doing comparisons against `$values` as text/template does not support type coercion. You can find a list of all the supported comparison functions here.

```
{{ if (gt $values.A.Value 80.0) -}}
high
{{ else if (gt $values.A.Value 50.0) -}}
medium
{{ else -}}
low
{{- end }}
```

Copy

Print all labels from a classic condition

You cannot use `$labels` to print labels from the query if you are using classic conditions, and must use `$values` instead. The reason for this is classic conditions discard these labels to enforce uni-dimensional behavior (at most one alert per alert rule). If classic conditions didn't discard these labels, then queries that returned many time series would cause alerts to flap between firing and resolved constantly as the labels would change every time the alert rule was evaluated.

Instead, the `$values` variable contains the reduced values of all time series for all conditions that are firing. For example, if you have an alert rule with a query A that returns two time series, and a classic condition B with two conditions, then `$values` would contain `B0`, `B1`, `B2` and `B3`. If the classic condition B had just one condition, then `$values` would contain just `B0` and `B1`.

To print all labels of all firing time series use the following template (make sure to replace `B` in the regular expression with the Ref ID of the classic condition if it's different):

```
{{ range $k, $v := $values -}}
{{ if (match "B[0-9]+" $k) -}}
{{ $k }}: {{ $v.Labels }}{{ end }}
{{ end }}
```

Copy

For example, a classic condition for two time series exceeding a single condition would print:

```
B0: instance=server1
B1: instance=server2
```

Copy

If the classic condition has two or more conditions, and a time series exceeds multiple conditions at the same time, then its labels will be duplicated for each condition that is exceeded:

```
B0: instance=server1
B1: instance=server2
B2: instance=server1
B3: instance=server2
```

Copy

If you need to print unique labels you should consider changing your alert rules from uni-dimensional to multi-dimensional instead. You can do this by replacing your classic condition with reduce and math expressions.

Print all values from a classic condition

To print all values from a classic condition take the previous example and replace `$v.Labels` with `$v.Value`:

```
{{ range $k, $v := $values -}}
{{ if (match "B[0-9]+" $k) -}}
{{ $k }}: {{ $v.Value }}{{ end }}
{{ end }}
```

Copy

For example, a classic condition for two time series exceeding a single condition would print:

```
B0: 81.2345
B1: 84.5678
```

Copy

If the classic condition has two or more conditions, and a time series exceeds multiple conditions at the same time, then `$values` will contain the values of all conditions:

```
B0: 81.2345
B1: 92.3456
B2: 84.5678
B3: 95.6789
```

Copy

Variables

The following variables are available to you when templating labels and annotations:

The labels variable

The `$labels` variable contains all labels from the query. For example, suppose you have a query that returns CPU usage for all of your servers, and you have an alert rule that fires when any of your servers have exceeded 80% CPU usage for the last 5 minutes. You want to add a summary annotation to the alert that tells you which server is experiencing high CPU usage. With the `$labels` variable you can write a template that prints a human-readable sentence such as:

```
CPU usage for {{ index $labels "instance" }} has exceeded 80% for the last 5 minutes
```
Copy

If you are using a classic condition then `$labels` will not contain any labels from the query. Classic conditions discard these labels in order to enforce uni-dimensional behavior (at most one alert per alert rule). If you want to use labels from the query in your template then use the example here.

The value variable

The `$value` variable is a string containing the labels and values of all instant queries; threshold, reduce and math expressions, and classic conditions in the alert rule. It does not contain the results of range queries, as these can return anywhere from 10s to 10,000s of rows or metrics. If it did, for especially large queries a single alert could use 10s of MBs of memory and Grafana would run out of memory very quickly.

To print the `$value` variable in the summary you would write something like this:

```
CPU usage for {{ index $labels "instance" }} has exceeded 80% for the last 5 minutes
```
Copy

And would look something like this:

```
CPU usage for instance1 has exceeded 80% for the last 5 minutes: [ var='A' labels={i
```
Copy

Here `var='A'` refers to the instant query with Ref ID A, `labels={instance=instance1}` refers to the labels, and `value=81.234` refers to the average CPU usage over the last 5 minutes.

If you want to print just some of the string instead of the full string then use the `$values` variable. It contains the same information as `$value`, but in a structured table, and is much easier to use then writing a regular expression to match just the text you want.

The values variable

The `$values` variable is a table containing the labels and floating point values of all instant queries and expressions, indexed by their Ref IDs.

To print the value of the instant query with Ref ID A:

```
CPU usage for {{ index $labels "instance" }} has exceeded 80% for the last 5 minutes
```
Copy

For example, given an alert with the labels `instance=server1` and an instant query with the value `81.2345`, this would print:

```
CPU usage for instance1 has exceeded 80% for the last 5 minutes: 81.2345
```
Copy

If the query in Ref ID A is a range query rather than an instant query then add a reduce expression with Ref ID B and replace `(index $values "A")` with `(index $values "B")`:

```
CPU usage for {{ index $labels "instance" }} has exceeded 80% for the last 5 minutes
```

Functions

The following functions are available to you when templating labels and annotations:

args

The `args` function translates a list of objects to a map with keys arg0, arg1 etc. This is intended to allow multiple arguments to be passed to templates:

```
{{define "x"}}{{.arg0}} {{.arg1}}{{end}}{{template "x" (args 1 "2")}}
```

```
1 2
```

externalURL

The `externalURL` function returns the external URL of the Grafana server as configured in the ini file(s):

```
{{ externalURL }}
```

```
https://example.com/grafana
```

graphLink

The `graphLink` function returns the path to the graphical view in Explore for the given expression and data source:

```
{{ graphLink "{\"expr\": \"up\", \"datasource\": \"gdev-prometheus\"}" }}
```

```
/explore?left=["now-1h","now","gdev-prometheus",{"datasource":"gdev-prometheus","exp
```

humanize

The `humanize` function humanizes decimal numbers:

```
{{ humanize 1000.0 }}
```

Copy

```
1k
```

Copy

humanize1024

The `humanize1024` works similar to `humanize` but but uses 1024 as the base rather than 1000:

```
{{ humanize1024 1024.0 }}
```

Copy

```
1ki
```

Copy

humanizeDuration

The `humanizeDuration` function humanizes a duration in seconds:

```
{{ humanizeDuration 60.0 }}
```

Copy

```
1m 0s
```

Copy

humanizePercentage

The `humanizePercentage` function humanizes a ratio value to a percentage:

```
{{ humanizePercentage 0.2 }}
```

Copy

```
20%
```

Copy

humanizeTimestamp

The `humanizeTimestamp` function humanizes a Unix timestamp:

```
{{ humanizeTimestamp 1577836800.0 }}
```

Copy

```
2020-01-01 00:00:00 +0000 UTC
```

Copy

match

The `match` function matches the text against a regular expression pattern:

```
{{ match "a.*" "abc" }}
```

Copy

```
true
```

Copy

pathPrefix

The `pathPrefix` function returns the path of the Grafana server as configured in the ini file(s):

```
{{ pathPrefix }}
```

Copy

```
/grafana
```

Copy

tableLink

The `tableLink` function returns the path to the tabular view in Explore for the given expression and data source:

```
{{ tableLink "{\"expr\": \"up\", \"datasource\": \"gdev-prometheus\"}" }}
```

Copy

```
/explore?left=["now-1h","now","gdev-prometheus",{"datasource":"gdev-prometheus","exp
```

Copy

title

The `title` function capitalizes the first character of each word:

```
{{ title "hello, world!" }}
```

Copy

```
Hello, World!
```

Copy

toLower

The `toLower` function returns all text in lowercase:

```
{{ toLower "Hello, world!" }}
```

Copy

```
hello, world!
```

Copy

toUpper

The `toUpper` function returns all text in uppercase:

```
{{ toUpper "Hello, world!" }}
```

Copy

```
HELLO, WORLD!
```

Copy

reReplaceAll

The `reReplaceAll` function replaces text matching the regular expression:

```
{{ reReplaceAll "localhost:(.*)" "example.com:$1" "localhost:8080" }}
```

Copy

```
example.com:8080
```

Copy

Alert rules

An alert rule is a set of evaluation criteria for when an alert rule should fire. An alert rule consists of one or more queries and expressions, a condition, and the duration over which the condition needs to be met to start firing.

While queries and expressions select the data set to evaluate, a condition sets the threshold that an alert must meet or exceed to create an alert.

An interval specifies how frequently an alert rule is evaluated. Duration, when configured, indicates how long a condition must be met. The alert rules can also define alerting behavior in the absence of data.

Alert rule types

Grafana supports two different alert rule types. Learn more about each of the alert rule types, how they work, and decide which one is best for your use case.

Grafana-managed alert rules

Grafana-managed alert rules are the most flexible alert rule type. They allow you to create alerts that can act on data from any of our supported data sources.

In addition to supporting multiple data sources, you can also add expressions to transform your data and set alert conditions. Using images in alert notifications is also supported. This is the only type of rule that allows alerting from multiple data sources in a single rule definition.

The following diagram shows how Grafana-managed alerting works.

Grafana-managed alerting

1. Alert rules are created within Grafana based on one or more data sources.
2. Alert rules are evaluated by the Alert Rule Evaluation Engine from within Grafana.
3. Alerts are delivered using the internal Grafana Alertmanager.

Note:

You can also configure alerts to be delivered using an external Alertmanager; or use both internal and external alertmanagers. For more information, see Add an external Alertmanager.

Data source-managed alert rules

To create data source-managed alert rules, you must have a compatible Prometheus or Loki data source.

You can check if your data source supports rule creation via Grafana by testing the data source and observing if the Ruler API is supported.

For more information on the Ruler API, refer to Ruler API.

The following diagram shows how data source-managed alerting works.

Grafana Mimir/Loki-managed alerting

1. Alert rules are created and stored within the data source itself.
2. Alert rules can only be created based on Prometheus data.
3. Alert rule evaluation and delivery is distributed across multiple nodes for high availability and fault tolerance.

Choose an alert rule type

When choosing which alert rule type to use, consider the following comparison between Grafana-managed alert rules and Grafana Mimir or Loki alert rules.

Feature	Grafana-managed alert rule	Loki/Mimir-managed alert rule
Create alert rules based on data from any of our supported data sources	Yes	No: You can only create alert rules that are based on Prometheus data. The data source must have the Ruler API enabled.
Mix and match data sources	Yes	No
Includes support for recording rules	No	Yes
Add expressions to transform your data and set alert conditions	Yes	No
Use images in alert notifications	Yes	No
Scaling	More resource intensive, depend on the database, and are likely to suffer from transient errors. They only scale vertically.	Store alert rules within the data source itself and allow for “infinite” scaling. Generate and send alert notifications from the location of your data.
Alert rule evaluation and delivery	Alert rule evaluation and delivery is done from within Grafana, using an external Alertmanager; or both.	Alert rule evaluation and alert delivery is distributed, meaning there is no single point of failure.

Note:

If you are using non-Prometheus data, we recommend choosing Grafana-managed alert rules. Otherwise, choose Grafana Mimir or Grafana Loki alert rules where possible.

Recording rules

Recording rules are only available for compatible Prometheus or Loki data sources.

A recording rule allows you to pre-compute frequently needed or computationally expensive expressions and save their result as a new set of time series. This is useful if you want to run alerts on aggregated data or if you have dashboards that query computationally expensive expressions repeatedly.

Querying this new time series is faster, especially for dashboards since they query the same expression every time the dashboards refresh.

Grafana Enterprise offers an alternative to recorded rules in the form of recorded queries that can be executed against any data source.

For more information on recording rules in Prometheus, refer to recording rules.

Queries and conditions

In Grafana, queries play a vital role in fetching and transforming data from supported data sources, which include databases like MySQL and PostgreSQL, time series databases like Prometheus, InfluxDB and Graphite, and services like Elasticsearch, AWS CloudWatch, Azure Monitor and Google Cloud Monitoring.

For more information on supported data sources, see Data sources.

The process of executing a query involves defining the data source, specifying the desired data to retrieve, and applying relevant filters or transformations. Query languages or syntaxes specific to the chosen data source are utilized for constructing these queries.

In Alerting, you define a query to get the data you want to measure and a condition that needs to be met before an alert rule fires.

An alert rule consists of one or more queries and expressions that select the data you want to measure.

For more information on queries and expressions, see Query and transform data.

Data source queries

Queries in Grafana can be applied in various ways, depending on the data source and query language being used. Each data source's query editor provides a customized user interface that helps you write queries that take advantage of its unique capabilities.

Because of the differences between query languages, each data source query editor looks and functions differently. Depending on your data source, the query editor might provide auto-completion features, metric names, variable suggestions, or a visual query-building interface.

Some common types of query components include:

Metrics or data fields: Specify the specific metrics or data fields you want to retrieve, such as CPU usage, network traffic, or sensor readings.

Time range: Define the time range for which you want to fetch data, such as the last hour, a specific day, or a custom time range.

Filters: Apply filters to narrow down the data based on specific criteria, such as filtering data by a specific tag, host, or application.

Aggregations: Perform aggregations on the data to calculate metrics like averages, sums, or counts over a given time period.

Grouping: Group the data by specific dimensions or tags to create aggregated views or breakdowns.

Note:

Grafana does not support alert queries with template variables. More information is available here.

Expression queries

In Grafana, an expression is used to perform calculations, transformations, or aggregations on the data source queried data. It allows you to create custom metrics or modify existing metrics based on mathematical operations, functions, or logical expressions.

By leveraging expression queries, users can perform tasks such as calculating the percentage change between two values, applying functions like logarithmic or trigonometric functions, aggregating data over specific time ranges or dimensions, and implementing conditional logic to handle different scenarios.

In Alerting, you can only use expressions for Grafana-managed alert rules. For each expression, you can choose from the math, reduce, and resample expressions. These are called multi-dimensional rules, because they generate a separate alert for each series.

You can also use classic condition, which creates an alert rule that triggers a single alert when its condition is met. As a result, Grafana sends only a single alert even when alert conditions are met for multiple series.

Note:

Classic conditions exist mainly for compatibility reasons and should be avoided if possible.

Reduce

Aggregates time series values in the selected time range into a single value.

Math

Performs free-form math functions/operations on time series and number data. Can be used to preprocess time series data or to define an alert condition for number data.

Resample

Realigns a time range to a new set of timestamps, this is useful when comparing time series data from different data sources where the timestamps would otherwise not align.

Threshold

Checks if any time series data matches the threshold condition.

The threshold expression allows you to compare two single values. It returns `0` when the condition is false and `1` if the condition is true. The following threshold functions are available:

- Is above (x > y)
- Is below (x < y)
- Is within range (x > y1 AND x < y2)
- Is outside range (x < y1 AND x > y2)

Classic condition

Checks if any time series data matches the alert condition.

Note:

Classic condition expression queries always produce one alert instance only, no matter how many time series meet the condition. Classic conditions exist mainly for compatibility reasons and should be avoided if possible.

Aggregations

Grafana Alerting provides the following aggregation functions to enable you to further refine your query.

These functions are available for **Reduce** and **Classic condition** expressions only.

Function	Expression	What it does
avg	Reduce / Classic	Displays the average of the values
min	Reduce / Classic	Displays the lowest value
max	Reduce / Classic	Displays the highest value
sum	Reduce / Classic	Displays the sum of all values
count	Reduce / Classic	Counts the number of values in the result
last	Reduce / Classic	Displays the last value
median	Reduce / Classic	Displays the median value
diff	Classic	Displays the difference between the newest and oldest value
diff_abs	Classic	Displays the absolute value of diff

Function	Expression	What it does
percent_diff	Classic	Displays the percentage value of the difference between newest and oldest value
percent_diff_abs	Classic	Displays the absolute value of percent_diff
count_non_null	Classic	Displays a count of values in the result set that aren't `null`

Alert condition

An alert condition is the query or expression that determines whether the alert will fire or not depending on the value it yields. There can be only one condition which will determine the triggering of the alert.

After you have defined your queries and/or expressions, choose one of them as the alert rule condition.

When the queried data satisfies the defined condition, Grafana triggers the associated alert, which can be configured to send notifications through various channels like email, Slack, or PagerDuty. The notifications inform you about the condition being met, allowing you to take appropriate actions or investigate the underlying issue.

By default, the last expression added is used as the alert condition.

Recovery threshold

> **NOTE**
>
> The recovery threshold feature is currently only available in OSS.

To reduce the noise of flapping alerts, you can set a recovery threshold different to the alert threshold.

Flapping alerts occur when a metric hovers around the alert threshold condition and may lead to frequent state changes, resulting in too many notifications being generated.

Grafana-managed alert rules are evaluated for a specific interval of time. During each evaluation, the result of the query is checked against the threshold set in the alert rule. If the value of a metric is above the threshold, an alert rule fires and a notification is sent. When the value goes below the threshold and there is an active alert for this metric, the alert is resolved, and another notification is sent.

It can be tricky to create an alert rule for a noisy metric. That is, when the value of a metric continually goes above and below a threshold. This is called flapping and results in a series of firing - resolved - firing notifications and a noisy alert state history.

For example, if you have an alert for latency with a threshold of 1000ms and the number fluctuates around 1000 (say 980 →1010 → 990 → 1020, and so on) then each of those will trigger a

notification.

To solve this problem, you can set a (custom) recovery threshold, which basically means having two thresholds instead of one. An alert is triggered when the first threshold is crossed and is resolved only when the second threshold is crossed.

For example, you could set a threshold of 1000ms and a recovery threshold of 900ms. This way, an alert rule will only stop firing when it goes under 900ms and flapping is reduced.

Alert instances

Grafana managed alerts support multi-dimensional alerting. Each alert rule can create multiple alert instances. This is exceptionally powerful if you are observing multiple series in a single expression.

Consider the following PromQL expression:

promql Copy

```
sum by(cpu) (
  rate(node_cpu_seconds_total{mode!="idle"}[1m])
)
```

A rule using this expression will create as many alert instances as the amount of CPUs we are observing after the first evaluation, allowing a single rule to report the status of each CPU.

A multi-dimensional Grafana managed alert rule

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Namespaces, folders, and groups

Alerts can be organized using folders for Grafana-managed rules and namespaces for Mimir or Loki rules and group names.

Namespaces and folders

When creating Grafana-managed rules, the folder can be used to perform access control and grant or deny access to all rules within a specific folder.

A namespace contains one or more groups. The rules within a group are run sequentially at a regular interval. The default interval is one (1) minute. You can rename Grafana Mimir or Loki rule namespaces and groups, and edit group evaluation intervals.

Groups

The rules within a group are run sequentially at a regular interval, meaning no rules will be evaluated at the same time and in order of appearance. The default interval is one (1) minute. You can rename Grafana Mimir or Loki rule namespaces and groups, and edit group evaluation intervals.

Note If you want rules to be evaluated concurrently and with different intervals, consider storing them in different groups.

Note Grafana managed alert rules are evaluated concurrently instead of sequentially.

Alert rule evaluation

Use alert rule evaluation to determine how frequently an alert rule should be evaluated and how quickly it should change its state.

To do this, you need to make sure that your alert rule is in the right evaluation group and set a pending period time that works best for your use case.

Evaluation group

Every alert rule is part of an evaluation group. Each evaluation group contains an evaluation interval that determines how frequently the alert rule is checked.

Data-source managed alert rules within the same group are evaluated one after the other, while alert rules in different groups can be evaluated simultaneously. This feature is especially useful when you want to ensure that recording rules are evaluated before any alert rules.

Grafana-managed alert rules are evaluated at the same time, regardless of alert rule group. The default evaluation interval is set at 10 seconds, which means that Grafana-managed alert rules are evaluated every 10 seconds to the closest 10-second window on the clock, for example, 10:00:00, 10:00:10, 10:00:20, and so on. You can also configure your own evaluation interval, if required.

Note:

Evaluation groups and alerts grouping in notification policies are two separate things. Grouping in notification policies allows multiple alerts sharing the same labels to be sent in the same time message.

Pending period

By setting a pending period, you can avoid unnecessary alerts for temporary problems.

In the pending period, you select the period in which an alert rule can be in breach of the condition until it fires.

Example

Imagine you have an alert rule evaluation interval set at every 30 seconds and the pending period to 90 seconds.

Evaluation will occur as follows:

[00:30][] First evaluation - condition not met.

[01:00][] Second evaluation - condition breached. Pending counter starts. **Alert starts pending.**

[01:30][] Third evaluation - condition breached. Pending counter = 30s. **Pending state.**

[02:00][] Fourth evaluation - condition breached. Pending counter = 60s **Pending state.**

[02:30][] Fifth evaluation - condition breached. Pending counter = 90s. **Alert starts firing**

If the alert rule has a condition that needs to be in breach for a certain amount of time before it takes action, then its state changes as follows:

- When the condition is first breached, the rule goes into a “pending” state.
- The rule stays in the “pending” state until the condition has been broken for the required amount of time - pending period.
- Once the required time has passed, the rule goes into a “firing” state.
- If the condition is no longer broken during the pending period, the rule goes back to its normal state.

Note:

If you want to skip the pending state, you can simply set the pending period to 0. This effectively skips the pending period and your alert rule will start firing as soon as the condition is breached.

When an alert rule fires, alert instances are produced, which are then sent to the Alertmanager.

State and health of alert rules

The state and health of alert rules help you understand several key status indicators about your alerts.

There are three key components: alert rule state, alert instance state, and alert rule health. Although related, each component conveys subtly different information.

Alert rule state

An alert rule can be in either of the following states:

State	Description
Normal	None of the time series returned by the evaluation engine is in a `Pending` or `Firing` state.
Pending	At least one time series returned by the evaluation engine is `Pending`.
Firing	At least one time series returned by the evaluation engine is `Firing`.

NOTE

Alerts will transition first to `pending` and then `firing`, thus it will take at least two evaluation cycles before an alert is fired.

Alert instance state

An alert instance can be in either of the following states:

State	Description
Normal	The state of an alert that is neither firing nor pending, everything is working correctly.
Pending	The state of an alert that has been active for less than the configured threshold duration.
Alerting	The state of an alert that has been active for longer than the configured threshold duration.
NoData	No data has been received for the configured time window.
Error	The error that occurred when attempting to evaluate an alert rule.

Keep last state

An alert rule can be configured to keep the last state when a `NoData` and/or `Error` state is encountered. This will both prevent alerts from firing, and from resolving and re-firing. Just like normal evaluation, the alert rule will transition from `Pending` to `Firing` after the pending period has elapsed.

Alert rule health

An alert rule can have one the following health statuses:

State	Description
Ok	No error when evaluating an alerting rule.
Error	An error occurred when evaluating an alerting rule.
NoData	The absence of data in at least one time series returned during a rule evaluation.
{status}, KeepLast	The rule would have received another status but was configured to keep the last state of the alert rule.

Special alerts for `NoData` and `Error`

When evaluation of an alert rule produces state `NoData` or `Error`, Grafana Alerting will generate alert instances that have the following additional labels:

Label	Description
alertname	Either `DatasourceNoData` or `DatasourceError` depending on the state.
datasource_uid	The UID of the data source that caused the state.

You can handle these alerts the same way as regular alerts by adding a silence, route to a contact point, and so on.

Notification templating

Notifications sent via contact points are built using notification templates. Grafana's default templates are based on the Go templating system where some fields are evaluated as text, while others are evaluated as HTML (which can affect escaping).

The default template default_template.go is a useful reference for custom templates.

Since most of the contact point fields can be templated, you can create reusable custom templates and use them in multiple contact points.

Using templates

The following example shows how to use default templates to render an alert message in Slack. The message title contains a count of alerts that are firing or were resolved. The message body lists the alerts and their status.

Default template

The following example shows the use of a custom template within one of the contact point fields.

Default template

Nested templates

You can embed templates within other templates.

For example, you can define a template fragment using the `define` keyword:

```
{{ define "mytemplate" }}
  {{ len .Alerts.Firing }} firing. {{ len .Alerts.Resolved }} resolved.
{{ end }}
```

Copy

You can then embed custom templates within this fragment using the `template` keyword. For example:

```
Alert summary:
{{ template "mytemplate" . }}
```

Copy

You can use any of the following built-in template options to embed custom templates.

Name	Notes
`default.title`	Displays high-level status information.
`default.message`	Provides a formatted summary of firing and resolved alerts.
`teams.default.message`	Similar to `default.message`, formatted for Microsoft Teams.

HTML in notification templates

HTML in alerting notification templates is escaped. We do not support rendering of HTML in the resulting notification.

Some notifiers support alternative methods of changing the look and feel of the resulting notification. For example, Grafana installs the base template for alerting emails to `<grafana-install-dir>/public/emails/ng_alert_notification.html`. You can edit this file to change the appearance of all alerting emails.

Alertmanager

Alertmanager enables you to quickly and efficiently manage and respond to alerts. It receives alerts, handles silencing, inhibition, grouping, and routing by sending notifications out via your channel of choice, for example, email or Slack.

In Grafana, you can use the Cloud Alertmanager, Grafana Alertmanager, or an external Alertmanager. You can also run multiple Alertmanagers; your decision depends on your set up and where your alerts are being generated.

Cloud Alertmanager

Cloud Alertmanager runs in Grafana Cloud and it can receive alerts from Grafana, Mimir, and Loki.

Grafana Alertmanager

Grafana Alertmanager is an internal Alertmanager that is pre-configured and available for selection by default if you run Grafana on-premises or open-source.

The Grafana Alertmanager can receive alerts from Grafana, but it cannot receive alerts from outside Grafana, for example, from Mimir or Loki.

Note that inhibition rules are not supported in the Grafana Alertmanager.

External Alertmanager

If you want to use a single Alertmanager to receive all your Grafana, Loki, Mimir, and Prometheus alerts, you can set up Grafana to use an external Alertmanager. This external Alertmanager can be configured and administered from within Grafana itself.

Here are two examples of when you may want to configure your own external alertmanager and send your alerts there instead of the Grafana Alertmanager:

1. You may already have Alertmanagers on-premises in your own Cloud infrastructure that you have set up and still want to use, because you have other alert generators, such as Prometheus.

2. You want to use both Prometheus on-premises and hosted Grafana to send alerts to the same Alertmanager that runs in your Cloud infrastructure.

Alertmanagers are visible from the drop-down menu on the Alerting Contact Points, Notification Policies, and Silences pages.

If you are provisioning your data source, set the flag `handleGrafanaManagedAlerts` in the `jsonData` field to `true` to send Grafana-managed alerts to this Alertmanager.

Useful links

Prometheus Alertmanager documentation

Add an external Alertmanager

Contact points

Contact points contain the configuration for sending notifications. A contact point is a list of integrations, each of which sends a notification to a particular email address, service or URL. Contact points can have multiple integrations of the same kind, or a combination of integrations of different kinds. For example, a contact point could contain a Pagerduty integration; an email and Slack integration; or a Pagerduty integration, a Slack integration, and two email integrations. You can also configure a contact point with no integrations; in which case no notifications are sent.

A contact point cannot send notifications until it has been added to a notification policy. A notification policy can only send alerts to one contact point, but a contact point can be added to a number of notification policies at the same time. When an alert matches a notification policy, the alert is sent to the contact point in that notification policy, which then sends a notification to each integration in its configuration.

Contact points can be configured for the Grafana Alertmanager as well as external alertmanagers.

You can also use notification templating to customize notification messages for contact point integrations.

Note:

If you've created an OnCall contact point in the Grafana OnCall application, you can view it in the Alerting application.

Supported contact point integrations

The following table lists the contact point integrations supported by Grafana.

Name	Type	Grafana Alertmanager	Other Alertmanagers
DingDing	`dingding`	Supported	N/A

Name	Type	Grafana Alertmanager	Other Alertmanagers
Discord	`discord`	Supported	N/A
Email	`email`	Supported	Supported
Google Chat	`googlechat`	Supported	N/A
Kafka	`kafka`	Supported	N/A
Line	`line`	Supported	N/A
Microsoft Teams	`teams`	Supported	Supported
Opsgenie	`opsgenie`	Supported	Supported
Pagerduty	`pagerduty`	Supported	Supported
Prometheus Alertmanager	`prometheus-alertmanager`	Supported	N/A
Pushover	`pushover`	Supported	Supported
Sensu Go	`sensugo`	Supported	N/A
Slack	`slack`	Supported	Supported
Telegram	`telegram`	Supported	N/A
Threema	`threema`	Supported	N/A
VictorOps	`victorops`	Supported	Supported
Webhook	`webhook`	Supported	Supported (different format)
Cisco Webex Teams	`webex`	Supported	Supported
WeCom	`wecom`	Supported	N/A
Zenduty	`webhook`	Supported	N/A

Notifications

Choosing how, when, and where to send your alert notifications is an important part of setting up your alerting system. These decisions will have a direct impact on your ability to resolve issues quickly and not miss anything important.

As a first step, define your contact points; where to send your alert notifications to. A contact point is a set of one or more integrations that are used to deliver notifications. Add notification templates to contact points for reuse and consistent messaging in your notifications.

Next, create a notification policy which is a set of rules for where, when and how your alerts are routed to contact points. In a notification policy, you define where to send your alert notifications by choosing one of the contact points you created.

Alertmanagers

Grafana uses Alertmanagers to send notifications for firing and resolved alerts. Grafana has its own Alertmanager, referred to as “Grafana” in the user interface, but also supports sending notifications from other Alertmanagers too, such as the Prometheus Alertmanager. The Grafana Alertmanager uses notification policies and contact points to configure how and where a notification is sent; how often a notification should be sent; and whether alerts should all be sent in the same notification, sent in grouped notifications based on a set of labels, or as separate notifications.

Notification policies

Notification policies control when and where notifications are sent. A notification policy can choose to send all alerts together in the same notification, send alerts in grouped notifications based on a set of labels, or send alerts as separate notifications. You can configure each notification policy to control how often notifications should be sent as well as having one or more mute timings to inhibit notifications at certain times of the day and on certain days of the week.

Notification policies are organized in a tree structure where at the root of the tree there is a notification policy called the default policy. There can be only one default policy and the default

policy cannot be deleted.

Specific routing policies are children of the default policy and can be used to match either all alerts or a subset of alerts based on a set of matching labels. A notification policy matches an alert when its matching labels match the labels in the alert.

A nested policy can have its own nested policies, which allow for additional matching of alerts. An example of a nested policy could be sending infrastructure alerts to the Ops team; while a nested policy might send high priority alerts to Pagerduty and low priority alerts as emails.

All alerts, irrespective of their labels, match the default policy. However, when the default policy receives an alert it looks at each nested policy and sends the alert to the first nested policy that matches the alert. If the nested policy has further nested policies, then it can attempt to the match the alert against one of its nested policies. If no nested policies match the alert then the policy itself is the matching policy. If there are no nested policies, or no nested policies match the alert, then the default policy is the matching policy.

Notification templates

You can customize notifications with templates. For example, templates can be used to change the subject and message of an email, or the title and message of notifications sent to Slack.

Templates are not limited to an individual integration or contact point, but instead can be used in a number of integrations in the same contact point and even integrations across different contact points. For example, a Grafana user can create a template called `custom_subject_or_title` and use it for both templating subjects in emails and titles of Slack messages without having to create two separate templates.

All notifications templates are written in Go's templating language, and are in the Contact points tab on the Alerting page.

Silences

You can use silences to mute notifications from one or more firing rules. Silences do not stop alerts from firing or being resolved, or hide firing alerts in the user interface. A silence lasts as long as its duration which can be configured in minutes, hours, days, months or years.

Notification policies

Notification policies provide you with a flexible way of routing alerts to various different receivers. Using label matchers, you can modify alert notification delivery without having to update every individual alert rule.

Learn more about how notification policies work and are structured, so that you can make the most out of setting up your notification policies.

Policy tree

Notification policies are *not* a list, but rather are structured according to a tree structure. This means that each policy can have child policies, and so on. The root of the notification policy tree is called the **Default notification policy**.

Each policy consists of a set of label matchers (0 or more) that specify which labels they are or aren't interested in handling.

For more information on label matching, see how label matching works.

> **NOTE**
>
> If you haven't configured any label matchers for your notification policy, your notification policy will match *all* alert instances. This may prevent child policies from being evaluated unless you have enabled **Continue matching siblings** on the notification policy.

Routing

To determine which notification policy will handle which alert instances, you have to start by looking at the existing set of notification policies, starting with the default notification policy.

If no policies other than the default policy are configured, the default policy will handle the alert instance.

If policies other than the default policy are defined, it will evaluate those notification policies in the order they are displayed.

If a notification policy has label matchers that match the labels of the alert instance, it will descend in to its child policies and, if there are any, will continue to look for any child policies that might have label matchers that further narrow down the set of labels, and so forth until no more child policies have been found.

If no child policies are defined in a notification policy or if none of the child policies have any label matchers that match the alert instance's labels, the default notification policy is used.

As soon as a matching policy is found, the system does not continue to look for other matching policies. If you want to continue to look for other policies that may match, enable **Continue matching siblings** on that particular policy.

Lastly, if none of the notification policies are selected the default notification policy is used.

Routing example

Here is an example of a relatively simple notification policy tree and some alert instances.

Notification policy routing

Here's a breakdown of how these policies are selected:

Pod stuck in CrashLoop does not have a `severity` label, so none of its child policies are matched. It does have a `team=operations` label, so the first policy is matched.

The `team=security` policy is not evaluated since we already found a match and **Continue matching siblings** was not configured for that policy.

Disk Usage – 80% has both a `team` and `severity` label, and matches a child policy of the operations team.

Unauthorized log entry has a `team` label but does not match the first policy (`team=operations`) since the values are not the same, so it will continue searching and match the `team=security` policy. It does not have any child policies, so the additional `severity=high` label is ignored.

Inheritance

In addition to child policies being a useful concept for routing alert instances, they also inherit properties from their parent policy. This also applies to any policies that are child policies of the default notification policy.

The following properties are inherited by child policies:

- Contact point
- Grouping options
- Timing options
- Mute timings

Each of these properties can be overwritten by an individual policy should you wish to override the inherited properties.

To inherit a contact point from the parent policy, leave it blank. To override the inherited grouping options, enable **Override grouping**. To override the inherited timing options, enable **Override general timings**.

Inheritance example

The example below shows how the notification policy tree from our previous example allows the child policies of the `team=operations` to inherit its contact point.

In this way, we can avoid having to specify the same contact point multiple times for each child policy.

Notification policy inheritance

Additional configuration options

Grouping

Grouping is an important feature of Grafana Alerting as it allows you to batch relevant alerts together into a smaller number of notifications. This is particularly important if notifications are delivered to first-responders, such as engineers on-call, where receiving lots of notifications in a short period of time can be overwhelming and in some cases can negatively impact a first-responders ability to respond to an incident. For example, consider a large outage where many of your systems are down. In this case, grouping can be the difference between receiving 1 phone call and 100 phone calls.

You choose how alerts are grouped together using the Group by option in a notification policy. By default, notification policies in Grafana group alerts together by alert rule using the `alertname` and `grafana_folder` labels (since alert names are not unique across multiple folders). Should you wish to group alerts by something other than the alert rule, change the grouping to any other combination of labels.

Disable grouping

Should you wish to receive every alert as a separate notification, you can do so by grouping by a special label called `...`. This is useful when your alerts are being delivered to an automated system instead of a first-responder.

A single group for all alerts

Should you wish to receive all alerts together in a single notification, you can do so by leaving Group by empty.

Timing options

The timing options decide how often notifications are sent for each group of alerts. There are three timers that you need to know about: Group wait, Group interval, and Repeat interval.

Group wait

Group wait is the amount of time Grafana waits before sending the first notification for a new group of alerts. The longer Group wait is the more time you have for other alerts to arrive. The shorter Group wait is the earlier the first notification will be sent, but at the risk of sending incomplete notifications. You should always choose a Group wait that makes the most sense for your use case.

Default 30 seconds

Group interval

Once the first notification has been sent for a new group of alerts, Grafana starts the Group interval timer. This is the amount of time Grafana waits before sending notifications about changes to the group. For example, another firing alert might have just been added to the group while an existing alert might have resolved. If an alert was too late to be included in the first notification due to Group wait, it will be included in subsequent notifications after Group interval. Once Group interval has elapsed, Grafana resets the Group interval timer. This repeats until there are no more alerts in the group after which the group is deleted.

Default 5 minutes

Repeat interval

Repeat interval decides how often notifications are repeated if the group has not changed since the last notification. You can think of these as reminders that some alerts are still firing. Repeat interval is closely related to Group interval, which means your Repeat interval must not only be greater than or equal to Group interval, but also must be a multiple of Group interval. If Repeat interval is not a multiple of Group interval it will be coerced into one. For example, if your Group interval is 5 minutes, and your Repeat interval is 9 minutes, the Repeat interval will be rounded up to the nearest multiple of 5 which is 10 minutes.

Default 4 hours

Alerting high availability

Grafana Alerting uses the Prometheus model of separating the evaluation of alert rules from the delivering of notifications. In this model the evaluation of alert rules is done in the alert generator and the delivering of notifications is done in the alert receiver. In Grafana Alerting, the alert generator is the Scheduler and the receiver is the Alertmanager.

High availability

When running multiple instances of Grafana, all alert rules are evaluated on all instances. You can think of the evaluation of alert rules as being duplicated. This is how Grafana Alerting makes sure that as long as at least one Grafana instance is working, alert rules will still be evaluated and notifications for alerts will still be sent. You will see this duplication in state history, and is a good way to tell if you are using high availability.

While the alert generator evaluates all alert rules on all instances, the alert receiver makes a best-effort attempt to avoid sending duplicate notifications. Alertmanager chooses availability over consistency, which may result in occasional duplicated or out-of-order notifications. It takes the opinion that duplicate or out-of-order notifications are better than no notifications.

The Alertmanager uses a gossip protocol to share information about notifications between Grafana instances. It also gossips silences, which means a silence created on one Grafana instance is replicated to all other Grafana instances. Both notifications and silences are persisted to the database periodically, and during graceful shut down.

It is important to make sure that gossiping is configured and tested. You can find the documentation on how to do that here.

Useful links

Configure alerting high availability

Set up Alerting

Set up or upgrade your implementation of Grafana Alerting.

Note:

These are set-up instructions for Grafana Alerting Open Source.

Before you begin

- Configure your data sources
- Check which data sources are compatible with and supported by Grafana Alerting

Watch this short video to get started.

Set up Alerting

To set up Alerting, you need to:

1. Configure alert rules
   - Create Grafana-managed or Mimir/Loki-managed alert rules and recording rules
2. Configure contact points
   - Check the default contact point and update the email address
   - Optional: Add new contact points and integrations
3. Configure notification policies
   - Check the default notification policy
   - Optional: Add additional nested policies
   - Optional: Add labels and label matchers to control alert routing
4. Optional: Integrate with Grafana OnCall

Advanced set up options

Grafana Alerting supports many additional configuration options, from configuring external Alertmanagers to routing Grafana-managed alerts outside of Grafana, to defining your alerting setup as code.

The following topics provide you with advanced configuration options for Grafana Alerting.

- Provision alert rules using file provisioning
- Provision alert rules using Terraform
- Add an external Alertmanager
- Configure high availability

Upgrade Alerting

WARNING

Legacy alerting will be removed in Grafana v11.0.0. Grafana v10.4 is the last version that offers legacy alerting and the last version of Grafana where automatic alert upgrades will be available.

Installing Grafana 11 before upgrading your legacy alerts will result in your existing alerts becoming inaccessible or lost.

For more information, refer to Legacy alerting removal: What you need to know about upgrading to Grafana Alerting.

What happens if I don't upgrade from legacy alerting to Grafana Alerting before installing Grafana 11?

Attempting to upgrade to Grafana 11 while still having legacy alerting explicitly enabled will result in Grafana not starting and an error message informing you to either:

1. Downgrade to a version prior to Grafana 11 and then upgrade from legacy alerting to Grafana Alerting using one of the below upgrade methods.
2. Confirm that you don't intend to upgrade your legacy alerts by disabling legacy alerting (optionally enable Grafana Alerting) in your configuration file.

NOTE

Confirming that you don't intend to upgrade your legacy alerts and continuing the Grafana 11 installation may result in your legacy alerts becoming inaccessible or lost.

Grafana will start with a blank alerting slate and you will need to recreate your alerts from scratch. The below automatic upgrade methods will not be available in Grafana 11.

As of v11.0, a rolling back to v10.4 will likely restore your legacy alerts. However, this is not guaranteed to always remain the case in future versions.

Upgrade methods

Grafana provides two methods for a seamless automatic upgrade of legacy alert rules and notification channels to Grafana Alerting:

1. **Upgrade with Preview** (Recommended): Offers a safe and controlled preview environment where you can review and adjust your upgraded alerts before fully enabling Grafana Alerting.
2. **Simple Upgrade**: One-step upgrade method for specific needs where a preview environment is not essential.

Key Considerations

Feature	Upgrade with Preview	Simple Upgrade
Safety and Control	☑ Preview environment for review and adjustment	✖ No preview, potential for unexpected issues
User Experience	☑ Seamless transition by handling issues early	✖ Possible disruption during upgrade
Granular Control	☑ Re-upgrade specific resources after resolving errors	✖ All or nothing upgrade, manual error correction
Stakeholder Involvement	☑ Collaboration and review of adjusted alerts	✖ Review only available after upgrade
Provisioning Support	☑ Configure new as-code before upgrading, simultaneous provisioning	✖ No built-in provisioning support
Simplicity	✖ May take longer to complete	☑ Fast, one-step process
Suited for:	☑ Complex setups, risk-averse environments, collaborative teams, heavy as-code use	☑ Simple setups, testing environments, large fleets
Version	Grafana v10.3.0 → v10.4.x	Grafana v9.0.0 → v10.4.x

NOTE

When upgrading with either method, your legacy dashboard alerts and notification channels are copied to a new format. This is non-destructive and can be rolled back easily.

Upgrade with Preview (Recommended)

Prerequisites

- Grafana `v10.3.0 or later`.
- Grafana administrator access.
- Enable `alertingPreviewUpgrade` feature toggle (enabled by default in v10.4.0 or later).

Suited for

- **Complex setups**: Large deployments with intricate alert rules and notification channels.
- **Risk-averse environments**: Situations where minimizing disruption and ensuring a smooth transition are critical.
- **Collaborative teams**: Projects where feedback and review from stakeholders are valuable.
- **Heavy as-code use**: Deployments with large or complex as-code configurations.

Overview

In **Alerts & IRM**, the **Alerting** section provides a preview of Grafana Alerting where you can review and modify your upgraded alerts before finalizing the upgrade.

In the **Alerting (legacy) → Alerting upgrade** section, you can upgrade your existing alert rules and notification channels, and view a summary of the upgrade to Grafana Alerting.

Finalize your upgrade by restarting Grafana with the `[unified_alerting]` section enabled in your configuration.

> **NOTE**
>
> Alerts generated by the new alerting system are visible in the **Alerting** section of the navigation panel but are not active until the upgrade is finalized.

To upgrade with preview, complete the following steps.

1. **Preview the Upgrade**:
   - **Initiate the process**: Access the upgrade functionality within Grafana by visiting the **Alerting upgrade** page in the **Alerting (legacy)** section of the navigation panel. From this page you can upgrade your existing alert rules and notification channels to the new Grafana Alerting system.

- **Review the summary table:** Review the detailed table outlining how your existing alert rules and notification channels were upgraded to resources in the new Grafana Alerting system.

2. **Investigate and Resolve Errors**:
   - **Identify errors**: Carefully examine the previewed upgrade:
     - Any alert rules or notification channels that couldn't be automatically upgraded will be highlighted with error indicators.
     - New or removed alert rules and notification channels will be highlighted with warning indicators.
   - **Address errors**: You have two options to resolve these issues:
     - **Fix legacy issues**: If possible, address the problems within your legacy alerting setup and attempt to upgrade the specific resource again.
     - **Create new resources**: If fixing legacy issues isn't viable, create new alert rules, notification policies, or contact points manually within the new Grafana Alerting system to replace the problematic ones.
3. **Update As-Code Setup** (Optional):
   - **Export upgraded resources**: If you use provisioning methods to manage alert rules and notification channels, you can export the upgraded versions to generate provisioning files compatible with Grafana Alerting.
   - **Test new provisioning definitions**: Ensure your as-code setup aligns with the new system before completing the upgrade process. Both legacy and Grafana Alerting alerts can be provisioned simultaneously to facilitate a smooth transition.
4. **Finalize the Upgrade**:
   - **Contact your Grafana server administrator**: Once you're confident in the state of your previewed upgrade, request to enable Grafana Alerting.
   - **Continued use for upgraded organizations**: Organizations that have already completed the preview upgrade will seamlessly continue using their configured setup.
   - **Automatic upgrade for others**: Organizations that haven't initiated the upgrade with preview process will undergo the traditional automatic upgrade during this restart.
   - **Address issues before restart**: Exercise caution, as Grafana will not start if any traditional automatic upgrades encounter errors. Ensure all potential issues are resolved before initiating this step.

Simple Upgrade

Prerequisites

- Grafana `v9.0.0 or later` (more recent versions are recommended).

Suited for

- **Simple setups**: Limited number of alerts and channels with minimal complexity.
- **Testing environments**: Where a quick upgrade without a preview is sufficient.
- **Large fleets**: Where manually reviewing each instance is not feasible.

Overview

While we recommend the **Upgrade with Preview** method for its enhanced safety and control, the **Simple Upgrade Method** exists for specific situations where a preview environment is not essential. For example, if you have a large fleet of Grafana instances and want to upgrade them all without the need to review and adjust each one individually.

Configure your Grafana instance to enable Grafana Alerting and disable legacy alerting. Then restart Grafana to automatically upgrade your existing alert rules and notification channels to the new Grafana Alerting system.

Once Grafana Alerting is enabled, you can review and adjust your upgraded alerts in the **Alerting** section of the navigation panel as well as export them for as-code setup.

To perform the simple upgrade, complete the following steps.

> **NOTE**
>
> Any errors encountered during the upgrade process will fail the upgrade and prevent Grafana from starting. If this occurs, you can roll back to legacy alerting.

1. **Upgrade to Grafana Alerting**:
   - **Enable Grafana Alerting**: Modify custom configuration file.
   - **Restart Grafana**: Restart Grafana for the configuration changes to take effect. Grafana will automatically upgrade your existing alert rules and notification channels to the new Grafana Alerting system.
2. **Review and Adjust Upgraded Alerts**:
   - **Review the upgraded alerts**: Go to the `Alerting` section of the navigation panel to review the upgraded alerts.
   - **Export upgraded resources**: If you use provisioning methods to manage alert rules and notification channels, you can export the upgraded versions to generate provisioning files compatible with Grafana Alerting.

Additional Information

Enable Grafana Alerting

Go to your custom configuration file ($WORKING_DIR/conf/custom.ini) and enter the following in your configuration:

```toml
[alerting]
enabled = false

[unified_alerting]
enabled = true
```

> **NOTE**
>
> If you have existing legacy alerts we advise using the Upgrade with Preview method first to ensure a smooth transition. Any organizations that have not completed the preview upgrade will automatically undergo the simple upgrade during the next restart.

Rolling back to legacy alerting

> **NOTE**
>
> For Grafana Cloud, contact customer support to enable or disable Grafana Alerting for your stack.

If you have upgraded to Grafana Alerting and want to roll back to legacy alerting, you can do so by disabling Grafana Alerting and re-enabling legacy alerting.

Go to your custom configuration file ($WORKING_DIR/conf/custom.ini) and enter the following in your configuration:

```toml
[alerting]
enabled = true

[unified_alerting]
enabled = false
```

This action is non-destructive. You can seamlessly switch between legacy alerting and Grafana Alerting at any time without losing any data. However, the upgrade process will only be performed

once. If you have opted out of Grafana Alerting and then opt in again, Grafana will not perform the upgrade again.

If, after rolling back, you wish to delete any existing Grafana Alerting configuration and upgrade your legacy alerting configuration again from scratch, you can enable the `clean_upgrade` option:

toml Copy

```
[unified_alerting.upgrade]
clean_upgrade = true
```

Differences and limitations

There are some differences between Grafana Alerting and legacy dashboard alerts, and a number of features that are no longer supported.

Differences

1. Read and write access to legacy dashboard alerts are governed by the dashboard permissions (including the inherited permissions from the folder) while Grafana alerts are governed by the permissions of the folder only. During the upgrade, an alert rule might be moved to a different folder to match the permissions of the dashboard. The following rules apply:
   - If the inherited dashboard permissions are different from the permissions of the folder, then the rule is moved to a new folder named after the original: `<Original folder name> - <Permission Hash>`.
   - If the inherited dashboard permissions are the same as the permissions of the folder, then the rule is moved to the original folder.
   - If the dashboard is in the `General` or `Dashboards` folder (i.e. no folder), then the rule is moved to a new `General Alerting - <Permission Hash>` folder.

> **NOTE**
>
> When updating your as-code provisioning setup for Grafana Alerting, newly generated folders will have a different UID from their original.

1. `NoData` and `Error` settings are upgraded as is to the corresponding settings in Grafana Alerting, except in two situations:
   - As there is no `Keep Last State` option in Grafana Alerting, this option becomes either `NoData` or `Error`. If using the `Simple Upgrade Method` Grafana automatically creates a 1 year silence for each alert rule with this configuration. If the alert evaluation returns no data or fails (error or timeout), then it creates a special alert, which will be silenced by the silence created during the upgrade.

- Due to lack of validation, legacy alert rules imported via JSON or provisioned along with dashboards can contain arbitrary values for `NoData` or `Error`. In this situation, Grafana will use the default setting: `NoData` for No data, and `Error` for Error.

2. Notification channels are upgraded to an Alertmanager configuration with the appropriate routes and receivers.

3. Unlike legacy dashboard alerts where images in notifications are enabled per contact point, images in notifications for Grafana Alerting must be enabled in the Grafana configuration, either in the configuration file or environment variables, and are enabled for either all or no contact points.

4. The JSON format for webhook notifications has changed in Grafana Alerting and uses the format from Prometheus Alertmanager.

5. Alerting on Prometheus `Both` type queries is not supported in Grafana Alerting. Existing legacy alerts with `Both` type queries are upgraded to Grafana Alerting as alerts with `Range` type queries.

Limitations

1. Since `Hipchat` and `Sensu` notification channels are no longer supported, legacy alerts associated with these channels are not automatically upgraded to Grafana Alerting. Assign the legacy alerts to a supported notification channel so that you continue to receive notifications for those alerts.

Legacy alerting deprecation

Starting with Grafana v9.0.0, legacy alerting is deprecated, meaning that it is no longer actively maintained or supported by Grafana. As of Grafana v10.0.0, we do not contribute or accept external contributions to the codebase apart from CVE fixes.

Legacy alerting refers to the old alerting system that was used prior to the introduction of Grafana Alerting; the new alerting system in Grafana.

The decision to deprecate legacy alerting was made to encourage users to migrate to the new alerting system, which offers a more powerful and flexible alerting experience based on Prometheus Alertmanager.

Users who are still using legacy alerting are encouraged to migrate their alerts to the new system as soon as possible to ensure that they continue to receive new features, bug fixes, and support.

However, we will still patch CVEs until legacy alerting is completely removed in Grafana 11; honoring our commitment to building and distributing secure software.

We have provided instructions on how to migrate to the new alerting system, making the process as easy as possible for users.

Why are we deprecating legacy alerting?

The new Grafana alerting system is more powerful and flexible than the legacy alerting feature.

The new system is based on Prometheus Alertmanager, which offers a more comprehensive set of features for defining and managing alerts. With the new alerting system, users can create alerts based on complex queries, configure alert notifications via various integrations, and set up sophisticated alerting rules with support for conditional expressions, aggregation, and grouping.

Overall, the new alerting system in Grafana is a major improvement over the legacy alerting feature, providing users with a more powerful and flexible alerting experience.

Additionally, legacy alerting still requires Angular to function and we are planning to remove support for it in Grafana 11.

When will we remove legacy alerting completely?

Legacy alerting will be removed from the code-base in Grafana 11, following the same timeline as the Angular deprecation.

How do I migrate to the new Grafana alerting?

Refer to our upgrade instructions.

Useful links

- Upgrade Alerting
- Angular support deprecation

Add an external Alertmanager

Set up Grafana to use an external Alertmanager as a single Alertmanager to receive all of your alerts. This external Alertmanager can then be configured and administered from within Grafana itself.

NOTE

Grafana Alerting does not support sending alerts to the AWS Managed Service for Prometheus due to the lack of sigv4 support in Prometheus.

Once you have added the Alertmanager, you can use the Grafana Alerting UI to manage silences, contact points, and notification policies. A drop-down option in these pages allows you to switch between alertmanagers.

NOTE

Starting with Grafana 9.2, the URL configuration of external alertmanagers from the Admin tab on the Alerting page is deprecated. It will be removed in a future release.

External alertmanagers should now be configured as data sources using Grafana Configuration from the main Grafana navigation menu. This enables you to manage the contact points and notification policies of external alertmanagers from within Grafana and also encrypts HTTP basic authentication credentials that were previously visible when configuring external alertmanagers by URL.

To add an external Alertmanager, complete the following steps.

1. Click **Connections** in the left-side menu.
2. On the Connections page, search for `Alertmanager`.
3. Click the **Create a new data source** button.

If you don’t see this button, you may need to install the plugin, relaunch your Cloud instance, and then repeat steps 1 and 2.

4. Fill out the fields on the page, as required.

   If you are provisioning your data source, set the flag `handleGrafanaManagedAlerts` in the `jsonData` field to `true` to send Grafana-managed alerts to this Alertmanager.

   Note: Prometheus, Grafana Mimir, and Cortex implementations of Alertmanager are supported. For Prometheus, contact points and notification policies are read-only in the Grafana Alerting UI.

5. Click **Save & test**.

Provision Alerting resources

Alerting infrastructure is often complex, with many pieces of the pipeline that often live in different places. Scaling this across multiple teams and organizations is an especially challenging task. Importing and exporting (or provisioning) your alerting resources in Grafana Alerting makes this process easier by enabling you to create, manage, and maintain your alerting data in a way that best suits your organization.

You can import alert rules, contact points, notification policies, mute timings, and templates.

You cannot edit imported alerting resources in the Grafana UI in the same way as alerting resources that were not imported. You can only edit imported contact points, notification policies, templates, and mute timings in the source where they were created. For example, if you manage your alerting resources using files from disk, you cannot edit the data in Terraform or from within Grafana.

Import alerting resources

Choose from the options below to import (or provision) your Grafana Alerting resources.

1. Use configuration files to provision your alerting resources, such as alert rules and contact points, through files on disk.

   NOTE

   - You cannot edit provisioned resources from files in the Grafana UI.
   - Provisioning with configuration files is not available in Grafana Cloud.

2. Use Terraform to provision alerting resources.
3. Use the Alerting provisioning HTTP API to manage alerting resources.

NOTE

The JSON output from the majority of Alerting HTTP endpoints isn't compatible for provisioning via configuration files.

If you need the alerting resources for file provisioning, use Export Alerting endpoints to return or download them in provisioning format.

Export alerting resources

You can export both manually created and provisioned alerting resources. You can also edit and export an alert rule without applying the changes.

For detailed instructions on the various export options, refer to Export alerting resources.

View provisioned alerting resources

To view your provisioned resources in Grafana, complete the following steps.

1. Open your Grafana instance.
2. Navigate to Alerting.
3. Click an alerting resource folder, for example, Alert rules.

Provisioned resources are labeled **Provisioned**, so that it is clear that they were not created manually.

Use configuration files to provision alerting resources

Manage your alerting resources using configuration files that can be version controlled. When Grafana starts, it provisions the resources defined in your configuration files. Provisioning can create, update, or delete existing resources in your Grafana instance.

This guide outlines the steps and references to provision alerting resources using YAML files. For a practical demo, you can clone and try this example using Grafana OSS and Docker Compose.

NOTE

- Provisioning Grafana with configuration files is not available in Grafana Cloud.
- You cannot edit provisioned resources from files in Grafana. You can only change the resource properties by changing the provisioning file and restarting Grafana or carrying out a hot reload. This prevents changes being made to the resource that would be overwritten if a file is provisioned again or a hot reload is carried out.
- Provisioning using configuration files takes place during the initial set up of your Grafana system, but you can re-run it at any time using the Grafana Admin API.
- Importing an existing alerting resource results in a conflict. First, when present, remove the resources you plan to import.

Details on how to set up the files and which fields are required for each object are listed below depending on which resource you are provisioning.

Import alert rules

Create or delete alert rules using provisioning files in your Grafana instance(s).

1. Find the alert rule group in Grafana.

2. Export and download a provisioning file for your alert rules.
3. Copy the contents into a YAML or JSON configuration file and add it to the `provisioning/alerting` directory of the Grafana instance you want to import the alerting resources to.

   Example configuration files can be found below.
4. Restart your Grafana instance (or reload the provisioned files using the Admin API).

Here is an example of a configuration file for creating alert rules.

yaml Copy

```yaml
# config file version
apiVersion: 1

# List of rule groups to import or update
groups:
  # <int> organization ID, default = 1
  - orgId: 1
    # <string, required> name of the rule group
    name: my_rule_group
    # <string, required> name of the folder the rule group will be stored in
    folder: my_first_folder
    # <duration, required> interval that the rule group should evaluated at
    interval: 60s
    # <list, required> list of rules that are part of the rule group
    rules:
      # <string, required> unique identifier for the rule. Should not exceed 40 symbols. Only lett
      - uid: my_id_1
        # <string, required> title of the rule that will be displayed in th
        title: my_first_rule
```

Expand code

Here is an example of a configuration file for deleting alert rules.

yaml Copy

```yaml
# config file version
apiVersion: 1

# List of alert rule UIDs that should be deleted
deleteRules:
  # <int> organization ID, default = 1
  - orgId: 1
```

```
    # <string, required> unique identifier for the rule
    uid: my_id_1
```

Import contact points

Create or delete contact points using provisioning files in your Grafana instance(s).

1. Find the contact point in Grafana.
2. Export and download a provisioning file for your contact point.
3. Copy the contents into a YAML or JSON configuration file and add it to the `provisioning/alerting` directory of the Grafana instance you want to import the alerting resources to.

   Example configuration files can be found below.
4. Restart your Grafana instance (or reload the provisioned files using the Admin API).

Here is an example of a configuration file for creating contact points.

yaml Copy

```yaml
# config file version
apiVersion: 1

# List of contact points to import or update
contactPoints:
  # <int> organization ID, default = 1
  - orgId: 1
    # <string, required> name of the contact point
    name: cp_1
    receivers:
      # <string, required> unique identifier for the receiver. Should not exceed 40 symbols. Only
      - uid: first_uid
        # <string, required> type of the receiver
        type: prometheus-alertmanager
        # <bool, optional> Disable the additional [Incident Resolved] follow-up alert, default = f
        disableResolveMessage: false
        # <object, required> settings for the specific receiver type
        settings:
          url: http://test:9000
```

Expand code

Here is an example of a configuration file for deleting contact points.

yaml Copy

```
# config file version
apiVersion: 1

# List of receivers that should be deleted
deleteContactPoints:
  # <int> organization ID, default = 1
  - orgId: 1
    # <string, required> unique identifier for the receiver
    uid: first_uid
```

Settings

Here are some examples of settings you can use for the different contact point integrations.

- Alertmanager
- DingDing
- Discord
- E-Mail
- Google Chat
- Kafka
- LINE
- Microsoft Teams
- OpsGenie
- PagerDuty
- Pushover
- Slack

Sensu Go

Telegram

Threema Gateway

VictorOps

Webhook

WeCom

Import templates

Create or delete templates using provisioning files in your Grafana instance(s).

1. Find the notification template in Grafana.
2. Export a template by copying the template content and title.
3. Copy the contents into a YAML or JSON configuration file and add it to the `provisioning/alerting` directory of the Grafana instance you want to import the alerting resources to.

   Example configuration files can be found below.
4. Restart your Grafana instance (or reload the provisioned files using the Admin API).

Here is an example of a configuration file for creating templates.

yaml Copy

```yaml
# config file version
apiVersion: 1

# List of templates to import or update
templates:
  # <int> organization ID, default = 1
  - orgId: 1
    # <string, required> name of the template, must be unique
    name: my_first_template
    # <string, required> content of the the template
    template: |
```

```
{{ define "my_first_template" }}
  Custom notification message
{{ end }}
```

Here is an example of a configuration file for deleting templates.

yaml Copy

```yaml
# config file version
apiVersion: 1

# List of alert rule UIDs that should be deleted
deleteTemplates:
  # <int> organization ID, default = 1
  - orgId: 1
    # <string, required> name of the template, must be unique
    name: my_first_template
```

Import notification policies

Create or reset the notification policy tree using provisioning files in your Grafana instance(s).

In Grafana, the entire notification policy tree is considered a single, large resource. Add new specific policies as sub-policies under the root policy. Since specific policies may depend on each other, you cannot provision subsets of the policy tree; the entire tree must be defined in a single place.

WARNING

Since the policy tree is a single resource, provisioning it will overwrite a policy tree created through any other means.

1. Find the notification policy tree in Grafana.
2. Export and download a provisioning file for your notification policy tree.
3. Copy the contents into a YAML or JSON configuration file and add it to the `provisioning/alerting` directory of the Grafana instance you want to import the alerting resources to.

   Example configuration files can be found below.
4. Restart your Grafana instance (or reload the provisioned files using the Admin API).

Here is an example of a configuration file for creating notification policies.

yaml Copy

```yaml
# config file version
apiVersion: 1

# List of notification policies
policies:
  # <int> organization ID, default = 1
  - orgId: 1
    # <string> name of the contact point that should be used for this route
    receiver: grafana-default-email
    # <list> The labels by which incoming alerts are grouped together. For example,
    #        multiple alerts coming in for cluster=A and alertname=LatencyHigh would
    #        be batched into a single group.
    #
    #        To aggregate by all possible labels use the special value '...' as
    #        the sole label name, for example:
    #        group_by: ['...']
    #        This effectively disables aggregation entirely, passing through all
    #        alerts as-is. This is unlikely to be what you want, unless you
    #        a very low alert volume or your upstream notification system performs
```

Expand code

Here is an example of a configuration file for resetting the policy tree back to its default value:

yaml Copy

```yaml
# config file version
apiVersion: 1

# List of orgIds that should be reset to the default policy
resetPolicies:
  - 1
```

Import templates

Create or delete templates in your Grafana instance(s).

1. Create a YAML or JSON configuration file.

   Example configuration files can be found below.

2. Add the file(s) to your GitOps workflow, so that they deploy alongside your Grafana instance(s).

Here is an example of a configuration file for creating templates.

yaml Copy

```yaml
# config file version
apiVersion: 1

# List of templates to import or update
templates:
  # <int> organization ID, default = 1
  - orgId: 1
    # <string, required> name of the template, must be unique
    name: my_first_template
    # <string, required> content of the template
    template: Alerting with a custom text template
```

Here is an example of a configuration file for deleting templates.

yaml Copy

```yaml
# config file version
apiVersion: 1

# List of alert rule UIDs that should be deleted
deleteTemplates:
  # <int> organization ID, default = 1
  - orgId: 1
    # <string, required> name of the template, must be unique
    name: my_first_template
```

Import mute timings

Create or delete mute timings via provisioning files using provisioning files in your Grafana instance(s).

1. Find the mute timing in Grafana.
2. Export and download a provisioning file for your mute timing.
3. Copy the contents into a YAML or JSON configuration file and add it to the `provisioning/alerting` directory of the Grafana instance you want to import the alerting resources to.

   Example configuration files can be found below.
4. Restart your Grafana instance (or reload the provisioned files using the Admin API).

Here is an example of a configuration file for creating mute timings.

yaml Copy

```yaml
# config file version
apiVersion: 1

# List of mute time intervals to import or update
muteTimes:
  # <int> organization ID, default = 1
  - orgId: 1
    # <string, required> name of the mute time interval, must be unique
    name: mti_1
    # <list> time intervals that should trigger the muting
    #        refer to https://prometheus.io/docs/alerting/latest/configuration/#time_interval-0
    time_intervals:
      - times:
          - start_time: '06:00'
            end_time: '23:59'
        location: 'UTC'
        weekdays: ['monday:wednesday', 'saturday', 'sunday']
        months: ['1:3', 'may:august', 'december']
        years: ['2020:2022', '2030']
```

Expand code

Here is an example of a configuration file for deleting mute timings.

```yaml
# config file version
apiVersion: 1

# List of mute time intervals that should be deleted
deleteMuteTimes:
  # <int> organization ID, default = 1
  - orgId: 1
    # <string, required> name of the mute time interval, must be unique
    name: mti_1
```

Template variable interpolation

Provisioning interpolates environment variables using the `$variable` syntax.

```yaml
contactPoints:
  - orgId: 1
    name: My Contact Email Point
```

```
    receivers:
      - uid: 1
        type: email
        settings:
          addresses: $EMAIL
```

In this example, provisioning replaces `$EMAIL` with the value of the `EMAIL` environment variable or an empty string if it is not present. For more information, refer to Using environment variables in the Provision documentation.

In alerting resources, most properties support template variable interpolation, with a few exceptions:

- Alert rule annotations: `groups[].rules[].annotations`
- Alert rule time range: `groups[].rules[].relativeTimeRange`
- Alert rule query model: `groups[].rules[].data.model`
- Mute timings name: `muteTimes[].name`
- Mute timings time intervals: `muteTimes[].time_intervals[]`
- Notification template name: `templates[].name`
- Notification template content: `templates[].template`

Note for properties that support interpolation, you may unexpectedly substitute template variables when not intended. To avoid this, you can escape the `$variable` with `$$variable`.

For example, when provisioning a `subject` property in a `contactPoints.receivers.settings` object that is meant to use the `$labels` variable.

1. `subject: '{{ $labels }}'` will interpolate, incorrectly defining the subject as `subject: '{{ }}'`.
2. `subject: '{{ $$labels }}'` will not interpolate, correctly defining the subject as `subject: '{{ $labels }}'`.

More examples

For more examples on the concept of this guide:

- Try provisioning alerting resources in Grafana OSS with YAML files through a demo project using Docker Compose or Kubernetes deployments.
- Review the distinct options about how Grafana provisions resources in the Provision Grafana documentation.
- For Helm support, review the examples provisioning alerting resources in the Grafana Helm Chart documentation.

Use Terraform to provision alerting resources

Use Terraform's Grafana Provider to manage your alerting resources and provision them into your Grafana system. Terraform provider support for Grafana Alerting makes it easy to create, manage, and maintain your entire Grafana Alerting stack as code.

This guide outlines the steps and references to provision alerting resources with Terraform. For a practical demo, you can clone and try this example using Grafana OSS and Docker Compose.

To create and manage your alerting resources using Terraform, you have to complete the following tasks.

1. Create an API key to configure the Terraform provider.
2. Create your alerting resources in Terraform format by
   - exporting configured alerting resources
   - or writing the Terraform Alerting schemas.

   > By default, you cannot edit provisioned resources. Enable `disable_provenance` in the Terraform resource to allow changes in the Grafana UI.
3. Run `terraform apply` to provision your alerting resources.

Before you begin, you should have available a Grafana instance and Terraform installed on your machine.

Create an API key and configure the Terraform provider

You can create a service account token to authenticate Terraform with Grafana. To create an API key for provisioning alerting resources, complete the following steps.

1. Create a new service account.

2. Assign the role or permission to access the Alerting provisioning API.
3. Create a new service account token.
4. Name and save the token for use in Terraform.

You can now move to the working directory for your Terraform configurations, and create a file named `main.tf` like:

main.tf

Copy

```
terraform {
    required_providers {
        grafana = {
            source = "grafana/grafana"
            version = ">= 2.9.0"
        }
    }
}

provider "grafana" {
    url = <grafana-url>
    auth = <api-key>
}
```

Replace the following values:

- `<grafana-url>` with the URL of the Grafana instance.
- `<api-key>` with the API token previously created.

This Terraform configuration installs the Grafana Terraform provider and authenticates against your Grafana instance using an API token. For other authentication alternatives including basic authentication, refer to the `auth` option documentation.

For Grafana Cloud, refer to the instructions to manage a Grafana Cloud stack with Terraform. For role-based access control, refer to Provisioning RBAC with Terraform and the alerting provisioning roles (`fixed:alerting.provisioning.*`).

Create Terraform configurations for alerting resources

Grafana Terraform provider enables you to manage the following alerting resources.

Alerting resource	Terraform resource
Alert rules	grafana_rule_group

Alerting resource	Terraform resource
Contact points	grafana_contact_point
Notification templates	grafana_message_template
Notification policy tree	grafana_notification_policy
Mute timings	grafana_mute_timing

In this section, we'll create Terraform configurations for each alerting resource and demonstrate how to link them together.

Add alert rules

Alert rules enable you to receive alerts by querying any backend Grafana data sources.

1. First, create a data source to query and a folder to store your rules in.

   In this example, the TestData data source is used.

   terraform Copy

   ```
   resource "grafana_data_source" "<terraform_data_source_name>" {
       name = "TestData"
       type = "testdata"
   }

   resource "grafana_folder" "<terraform_folder_name>" {
       title = "My Rule Folder"
   }
   ```

   Replace the following field values:

   - `<terraform_data_source_name>` with the terraform name of the data source.
   - `<terraform_folder_name>` with the terraform name of the folder.

2. Create or find an alert rule you want to import in Grafana.

3. Export the alert rule group in Terraform format. This exports the alert rule group as `grafana_rule_group` Terraform resource.

   You can edit the exported resource, or alternatively, consider creating the resource from scratch.

   terraform Copy

   ```
   resource "grafana_rule_group" "<terraform_rule_group_name>" {
       name = "My Alert Rules"
       folder_uid = grafana_folder.<terraform_folder_name>.uid
   ```

```
    interval_seconds = 60
    org_id = 1

    rule {
        name = "My Random Walk Alert"
        condition = "C"
        for = "0s"

        // Query the datasource.
        data {
            ref_id = "A"
            relative_time_range {
                from = 600
                to = 0
            }
```

Expand code

Replace the following field values:

- `<terraform_rule_group_name>` with the name of the alert rule group.

Note that the distinct Grafana resources are connected through `uid` values in their Terraform configurations. The `uid` value will be randomly generated when provisioning.

To link the alert rule group with its respective data source and folder in this example, replace the following field values:

- `<terraform_data_source_name>` with the terraform name of the previously defined data source.
- `<terraform_folder_name>` with the terraform name of the the previously defined folder.

4. Continue to add more Grafana resources or use the Terraform CLI for provisioning.

Add contact points

Contact points are the receivers of alert notifications.

1. Create or find the contact points you want to import in Grafana. Alternatively, consider writing the resource in code as demonstrated in the example below.
2. Export the contact point in Terraform format. This exports the contact point as `grafana_contact_point` Terraform resource—edit it if necessary.
3. In this example, notifications are muted on weekends.

terraform

Copy

```
resource "grafana_contact_point" "<terraform_contact_point_name>" {
    name = "My contact point email"

    email {
```

```
        addresses               = ["<email_address>"]
    }
}
```

Replace the following field values:

- `<terraform_contact_point_name>` with the terraform name of the contact point. It will be used to reference the contact point in other Terraform resources.
- `<email_address>` with the email to receive alert notifications.

4. Continue to add more Grafana resources or use the Terraform CLI for provisioning.

Add and enable templates

Notification templates allow customization of alert notifications across multiple contact points.

1. Create or find the notification template you want to import in Grafana. Alternatively, consider writing the resource in code as demonstrated in the example below.
2. Export the template as `grafana_message_template` Terraform resource.

   This example is a simple demo template defined as `custom_email.message`.

terraform — Copy

```
resource "grafana_message_template" "<terraform_message_template_name>" {
    name = "custom_email.message"

    template = <<EOT
{{ define "custom_email.message" }}
Lorem ipsum - Custom alert!
{{ end }}
EOT
}
```

3. In the previous contact point, enable the template by setting the `email.message` property as follows.

terraform — Copy

```
resource "grafana_contact_point" "<terraform_contact_point_name>" {
    name = "My contact point email"

    email {
        addresses               = ["<email_address>"]
        message                 = "{{ template \"custom_email.message\" .}}"
```

```
    }
}
```

4. Continue to add more Grafana resources or use the Terraform CLI for provisioning.

Add mute timings

Mute timings pause alert notifications during predetermined intervals.

1. Create or find the mute timings you want to import in Grafana. Alternatively, consider writing the resource in code as demonstrated in the example below.
2. Export the mute timing in Terraform format. This exports the mute timing as `grafana_mute_timing` Terraform resource—edit it if necessary.
3. This example turns off notifications on weekends.

```terraform
resource "grafana_mute_timing" "<terraform_mute_timing_name>" {
    name = "No weekends"

    intervals {
        weekdays = ["saturday", "sunday"]
    }
}
```

   Replace the following field values:

   - `<terraform_mute_timing_name>` with the name of the Terraform resource. It will be used to reference the mute timing in the Terraform notification policy tree.

4. Continue to add more Grafana resources or use the Terraform CLI for provisioning.

Add the notification policy tree

Notification policies defines how to route alert instances to your contact points.

> **WARNING**
>
> Since the policy tree is a single resource, provisioning the `grafana_notification_policy` resource will overwrite a policy tree created through any other means.

1. Find the default notification policy tree. Alternatively, consider writing the resource in code as demonstrated in the example below.
2. Export the notification policy tree in Terraform format. This exports it as `grafana_notification_policy` Terraform resource—edit it if necessary.

terraform

Copy

```
resource "grafana_notification_policy" "my_policy_tree" {
contact_point = grafana_contact_point.<terraform_contact_point_name>.name
...

policy {
    contact_point = grafana_contact_point.<terraform_contact_point_name>.name

    matcher {...}

    mute_timings = [grafana_mute_timing.<terraform_mute_timing_name>.name]
}
}
```

To configure the mute timing and contact point previously created in the notification policy tree, replace the following field values:

- `<terraform_data_source_name>` with the terraform name of the previously defined contact point.
- `<terraform_folder_name>` with the terraform name of the the previously defined mute timing.

3 Continue to add more Grafana resources or use the Terraform CLI for provisioning.

Enable editing resources in the Grafana UI

By default, you cannot edit resources provisioned via Terraform in Grafana. This ensures that your alerting stack always stays in sync with your Terraform code.

To make provisioned resources editable in the Grafana UI, enable the `disable_provenance` attribute on alerting resources.

terraform

Copy

```
resource "grafana_contact_point" "my_contact_point" {
  name = "My Contact Point"

  disable_provenance = true
}

resource "grafana_message_template" "my_template" {
  name     = "My Reusable Template"
  template = "{{define \"My Reusable Template\" }}\n template content\n{{ end }}"

  disable_provenance = true
```

```
}
...
```

Provision Grafana resources with Terraform

To create the previous alerting resources in Grafana with the Terraform CLI, complete the following steps.

1. Initialize the working directory containing the Terraform configuration files.

   shell Copy

   ```
   terraform init
   ```

   This command initializes the Terraform directory, installing the Grafana Terraform provider configured in the `main.tf` file.

2. Apply the Terraform configuration files to provision the resources.

   shell Copy

   ```
   terraform apply
   ```

   Before applying any changes to Grafana, Terraform displays the execution plan and requests your approval.

   shell Copy

   ```
   Plan: 4 to add, 0 to change, 0 to destroy.

   Do you want to perform these actions?
   Terraform will perform the actions described above.
   Only 'yes' will be accepted to approve.

   Enter a value:
   ```

   Once you have confirmed to proceed with the changes, Terraform will create the provisioned resources in Grafana!

   shell Copy

   ```
   Apply complete! Resources: 4 added, 0 changed, 0 destroyed.
   ```

You can now access Grafana to verify the creation of the distinct resources.

More examples

For more examples on the concept of this guide:

- Try the demo provisioning alerting resources in Grafana OSS using Terraform and Docker Compose.
- Review all the available options and examples of the Terraform Alerting schemas in the Grafana Terraform Provider documentation.
- Review the tutorial to manage a Grafana Cloud stack using Terraform.

Export alerting resources

Export your alerting resources, such as alert rules, contact points, and notification policies for provisioning, automatically importing single folders and single groups.

There are distinct methods to export your alerting resources:

- Grafana UI exports in Terraform format and YAML or JSON formats for file provisioning.
- HTTP Alerting API exports in JSON API format used by the HTTP Alerting API.
- HTTP Alerting API - Export endpoints exports in YAML or JSON formats for file provisioning.

NOTE

Alerting resources imported through file provisioning cannot be edited in the Grafana UI. This prevents changes made in the UI from being overridden by file provisioning during Grafana restarts.

If you need to modify provisioned alerting resources in Grafana, refer to edit HTTP API alerting resources in the Grafana UI or to edit Terraform alerting resources in the Grafana UI.

Export from the Grafana UI

The export options listed below enable you to download resources in YAML, JSON, or Terraform format, facilitating their provisioning through configuration files or Terraform.

Export alert rules

To export alert rules from the Grafana UI, complete the following steps.

1. Click **Alerts & IRM** → **Alert rules**.

2. To export all Grafana-managed rules, click **Export rules**.
3. To export a folder, change the **View as** to **List**.
4. Select the folder you want to export and click the **Export rules folder** icon.
5. To export a group, change the **View as** to **Grouped**.
6. Find the group you want to export and click the **Export rule group** icon.
7. Choose the format to export in.

   The exported alert rule data appears in different formats - YAML, JSON, Terraform.
8. Click **Copy Code** or **Download**.

Modify alert rule and export rule group without saving changes

> **NOTE**
>
> This feature is for Grafana-managed alert rules only. It is available to Admin, Viewer, and Editor roles.

Use the **Modify export** mode to edit and export an alert rule without updating it. The exported data includes all alert rules within the same alert group.

To export a modified alert rule without saving the modifications, complete the following steps from the Grafana UI.

1. Click **Alerts & IRM** → **Alert rules**.
2. Locate the alert rule you want to edit and click **More** → **Modify Export** to open the Alert Rule form.
3. From the Alert Rule form, edit the fields you want to change. Changes made are not applied to the alert rule.
4. Click **Export**.
5. Choose the format to export in.

   The exported alert rule group appears in different formats - YAML, JSON, Terraform.
6. Click **Copy Code** or **Download**.

Export contact points

To export contact points from the Grafana UI, complete the following steps.

1. Click **Alerts & IRM** → **Contact points**.
2. Find the contact point you want to export and click **More** → **Export**.

3. Choose the format to export in.

   The exported contact point appears in different formats - YAML, JSON, Terraform.

4. Click **Copy Code** or **Download**.

Export templates

Grafana currently doesn't offer an Export UI or Export endpoint for notification templates, unlike other Alerting resources presented in this documentation.

However, you can export it by manually copying the content template and title directly from the Grafana UI.

1. Click **Alerts & IRM** → **Contact points** → **Notification templates** tab.
2. Find the template you want to export.
3. Copy the content and title.
4. Adjust it for the file provisioning format or Terraform resource.

Export the notification policy tree

All notification policies are provisioned through a single resource: the root of the notification policy tree.

> **WARNING**
>
> Since the policy tree is a single resource, provisioning it will overwrite a policy tree created through any other means.

To export the notification policy tree from the Grafana UI, complete the following steps.

1. Click **Alerts & IRM** → **Notification policies**.
2. In the **Default notification policy** section, click **...** → **Export**.
3. Choose the format to export in.

   The exported contact point appears in different formats - YAML, JSON, Terraform.

4. Click **Copy Code** or **Download**.

Export mute timings

To export mute timings from the Grafana UI, complete the following steps.

1. Click **Alerts & IRM** → **Notification policies**, and then the **Mute timings** tab.
2. Find the mute timing you want to export and click **Export**.

3. Choose the format to export in.

   The exported contact point appears in different formats - YAML, JSON, Terraform.

4. Click **Copy Code** or **Download**.

HTTP Alerting API

You can use the Alerting HTTP API to return existing alerting resources in JSON and import them to another Grafana instance using the same endpoint.

Resource	URI
Alert rules	/api/v1/provisioning/alert-rules
Contact points	/api/v1/provisioning/contact-points
Notification policy tree	/api/v1/provisioning/policies
Mute timings	/api/v1/provisioning/mute-timings
Templates	/api/v1/provisioning/templates

However, note the standard endpoints return a JSON format that is not compatible for provisioning through configuration files or Terraform, except the `/export` endpoints listed below.

Export API endpoints

The **Alerting HTTP API** provides specific endpoints for exporting alerting resources in YAML or JSON formats, facilitating [provisioning via configuration files][alerting_file_provisioning]. Currently, Terraform format is not supported.

Resource	Method / URI	Summary
Alert rules	GET /api/v1/provisioning/alert-rules/export	Export all alert rules in provisioning file format.
Alert rules	GET /api/v1/provisioning/folder/:folderUid/rule-groups/:group/export	Export an alert rule group in provisioning file format.
Alert rules	GET /api/v1/provisioning/alert-rules/:uid/export	Export an alert rule in provisioning file format.
Contact points	GET /api/v1/provisioning/contact-points/export	Export all contact points in provisioning file format.
Notification policy tree	GET /api/v1/provisioning/policies/export	Export the notification policy tree in provisioning file format.

Resource	Method / URI	Summary
Mute timings	GET /api/v1/provisioning/mute-timings/export	Export all mute timings in provisioning file format.
Mute timings	GET /api/v1/provisioning/mute-timings/:name/export	Export a mute timing in provisioning file format.

These endpoints accept a `download` parameter to download a file containing the exported resources.

Use the HTTP API to manage alerting resources

The Alerting Provisioning HTTP API can be used to create, modify, and delete resources relevant to Grafana-managed alerts. This API is the one used by our Grafana Terraform provider.

For more information on the differences between Grafana-managed and data source-managed alerts, refer to Introduction to alert rules.

If you are running Grafana Enterprise, you need to add specific permissions for some endpoints. For more information, refer to Role-based access control permissions.

Grafana-managed endpoints

Note that the JSON format from most of the following endpoints is not fully compatible with provisioning via configuration JSON files.

Alert rules

Method	URI	Name	Summary
DELETE	/api/v1/provisioning/alert-rules/:uid	route delete alert rule	Delete a specific alert rule by UID.
GET	/api/v1/provisioning/alert-rules/:uid	route get alert rule	Get a specific alert rule by UID.
POST	/api/v1/provisioning/alert-rules	route post alert rule	Create a new alert rule.

Method	URI	Name	Summary
PUT	/api/v1/provisioning/alert-rules/:uid	route put alert rule	Update an existing alert rule.
GET	/api/v1/provisioning/alert-rules/:uid/export	route get alert rule export	Export an alert rule in provisioning file format.
GET	/api/v1/provisioning/folder/:folderUid/rule-groups/:group	route get alert rule group	Get a rule group.
PUT	/api/v1/provisioning/folder/:folderUid/rule-groups/:group	route put alert rule group	Update the interval of a rule group or modify the rules of the group.
GET	/api/v1/provisioning/folder/:folderUid/rule-groups/:group/export	route get alert rule group export	Export an alert rule group in provisioning file format.
GET	/api/v1/provisioning/alert-rules	route get alert rules	Get all the alert rules.
GET	/api/v1/provisioning/alert-rules/export	route get alert rules export	Export all alert rules in provisioning file format.

Example request for new alert rule:

http

Copy

```
POST /api/v1/provisioning/alert-rules
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "title": "TEST-API_1",
  "ruleGroup": "API",
  "folderUID": "SET_FOLDER_UID",
  "noDataState": "OK",
  "execErrState": "OK",
  "for": "5m",
  "orgId": 1,
  "uid": "",
  "condition": "B",
  "annotations": {
    "summary": "test_api_1"
  },
  "labels": {
```

Expand code

Example Response:

```http
HTTP/1.1 201 Created
Content-Type: application/json

{
  "id": 1,
  "uid": "XXXXXXXXX",
  "orgID": 1,
  "folderUID": "SET_FOLDER_UID",
  "ruleGroup": "API3",
  "title": "TEST-API_1",
  "condition": "B",
  "data": [
    {
      "refId": "A",
      "queryType": "",
      "relativeTimeRange": {
        "from": 600,
        "to": 0
      },
```

Copy

Expand code

Contact points

Method	URI	Name	Summary
DELETE	/api/v1/provisioning/contact-points/:uid	route delete contactpoints	Delete a contact point.
GET	/api/v1/provisioning/contact-points	route get contactpoints	Get all the contact points.
POST	/api/v1/provisioning/contact-points	route post contactpoints	Create a contact point.
PUT	/api/v1/provisioning/contact-points/:uid	route put contactpoint	Update an existing contact point.
GET	/api/v1/provisioning/contact-points/export	route get contactpoints export	Export all contact points in provisioning file format.

Example Request for all the contact points:

```http
GET /api/v1/provisioning/contact-points
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```http
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "uid": "",
    "name": "email receiver",
    "type": "email",
    "settings": {
      "addresses": "<example@email.com>"
    },
    "disableResolveMessage": false
  }
]
```

Notification policies

Method	URI	Name	Summary
DELETE	/api/v1/provisioning/policies	route reset policy tree	Clears the notification policy tree.
GET	/api/v1/provisioning/policies	route get policy tree	Get the notification policy tree.
PUT	/api/v1/provisioning/policies	route put policy tree	Sets the notification policy tree.
GET	/api/v1/provisioning/policies/export	route get policy tree export	Export the notification policy tree in provisioning file format.

Example Request for exporting the notification policy tree in YAML format:

```http
GET /api/v1/provisioning/policies/export?format=yaml
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```http
HTTP/1.1 200 OK
Content-Type: text/yaml

apiVersion: 1
policies:
    - orgId: 1
      receiver: My Contact Email Point
      group_by:
        - grafana_folder
        - alertname
      routes:
        - receiver: My Contact Email Point
          object_matchers:
            - - monitor
              - =
              - testdata
          mute_time_intervals:
            - weekends
```

Mute timings

Method	URI	Name	Summary
DELETE	/api/v1/provisioning/mute-timings/:name	route delete mute timing	Delete a mute timing.
GET	/api/v1/provisioning/mute-timings/:name	route get mute timing	Get a mute timing.
GET	/api/v1/provisioning/mute-timings	route get mute timings	Get all the mute timings.

Method	URI	Name	Summary
POST	/api/v1/provisioning/mute-timings	route post mute timing	Create a new mute timing.
PUT	/api/v1/provisioning/mute-timings/:name	route put mute timing	Replace an existing mute timing.
GET	/api/v1/provisioning/mute-timings/export	route get mute timings export	Export all mute timings in provisioning file format.
GET	/api/v1/provisioning/mute-timings/:name/export	route get mute timing export	Export a mute timing in provisioning file format.

Example Request for all mute timings:

http Copy

```
GET /api/v1/provisioning/mute-timings
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

http Copy

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "name": "weekends",
    "time_intervals": [
      {
        "weekdays": [
          "saturday",
          "sunday"
        ]
      }
    ],
    "version": "",
    "provenance": "file"
  }
]
```

Templates

Method	URI	Name	Summary
DELETE	/api/v1/provisioning/templates/:name	route delete template	Delete a template.
GET	/api/v1/provisioning/templates/:name	route get template	Get a notification template.
GET	/api/v1/provisioning/templates	route get templates	Get all notification templates.
PUT	/api/v1/provisioning/templates/:name	route put template	Create or update a notification template.

Example Request for all notification templates:

http Copy

```
GET /api/v1/provisioning/templates
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

http Copy

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "name": "custom_email.message",
    "template": "{{ define \"custom_email.message\" }}\n  Custom alert!\n{{ end }}",
    "provenance": "file"
  },
  {
    "name": "custom_email.subject",
    "template": "{{ define \"custom_email.subject\" }}\n{{ len .Alerts.Firing }} firing alert(s),
    "provenance": "file"
  }
]
```

Edit resources in the Grafana UI

By default, you cannot edit API-provisioned alerting resources in Grafana. To enable editing these resources in the Grafana UI, add the `X-Disable-Provenance` header to the following requests in the API:

- `POST /api/v1/provisioning/alert-rules`
- `PUT /api/v1/provisioning/folder/{FolderUID}/rule-groups/{Group}` (calling this endpoint will change provenance for all alert rules within the alert group)
- `POST /api/v1/provisioning/contact-points`
- `POST /api/v1/provisioning/mute-timings`
- `PUT /api/v1/provisioning/policies`
- `PUT /api/v1/provisioning/templates/{name}`

To reset the notification policy tree to the default and unlock it for editing in the Grafana UI, use the `DELETE /api/v1/provisioning/policies` endpoint.

Data source-managed resources

The Alerting Provisioning HTTP API can only be used to manage Grafana-managed alert resources. To manage resources related to data source-managed alerts, consider the following tools:

- mimirtool: to interact with the Mimir alertmanager and ruler configuration.
- cortex-tools: to interact with the Cortex alertmanager and ruler configuration.
- lokitool: to configure the Loki Ruler.

Alternatively, the Grafana Alerting API can be used to access data from data source-managed alerts. This API is primarily intended for internal usage, with the exception of the `/api/v1/provisioning/` endpoints. It's important to note that internal APIs may undergo changes without prior notice and are not officially supported for user consumption.

For Prometheus, `amtool` can also be used to interact with the AlertManager API.

Paths

Delete a specific alert rule by UID. (*RouteDeleteAlertRule*)

```
DELETE /api/v1/provisioning/alert-rules/:uid
```

Copy

Parameters

Name	Source	Type	Go type	Separator	Required	Default	Descriptic
UID	`path`	string	`string`		✓		Alert rule UID
X-Disable-Provenance: true	`header`	string	`string`				Allows editing of provisione resources in the Grafana U

All responses

Code	Status	Description	Has headers	Schema
204	No Content	The alert rule was deleted successfully.		schema

Responses

204 - The alert rule was deleted successfully.

Status: No Content

Schema

Delete a contact point. (*RouteDeleteContactpoints*)

```
DELETE /api/v1/provisioning/contact-points/:uid
```

Copy

Parameters

Name	Source	Type	Go type	Separator	Required	Default	Description
UID	`path`	string	`string`		✓		UID is the contact point unique identifier

All responses

Code	Status	Description	Has headers	Schema
204	No Content	The contact point was deleted successfully.		schema

Responses

204 - The contact point was deleted successfully.

Status: No Content

Schema

Delete a mute timing. (*RouteDeleteMuteTiming*)

```
DELETE /api/v1/provisioning/mute-timings/:name
```

Copy

Parameters

Name	Source	Type	Go type	Separator	Required	Default	Description
name	`path`	string	`string`		✓		Mute timing name
version	`query`	string	`string`				Current version of the resource. Used for optimistic concurrency validation. Keep empty to bypass validation.

All responses

Code	Status	Description	Has headers	Schema
204	No Content	The mute timing was deleted successfully.		schema
409	Conflict	GenericPublicError		schema

Responses

204 - The mute timing was deleted successfully.

Status: No Content

Schema

409 - Conflict

Status: Conflict

Schema

GenericPublicError

Delete a template. (*RouteDeleteTemplate*)

```
DELETE /api/v1/provisioning/templates/:name
```

Copy

Parameters

Name	Source	Type	Go type	Separator	Required	Default	Description
name	`path`	string	`string`		✓		Template Name
version	`query`	string	`string`				Current version of the resource. Used for optimistic concurrency validation. Keep empty to bypass validation.

All responses

Code	Status	Description	Has headers	Schema
204	No Content	The template was deleted successfully.		schema
409	Conflict	GenericPublicError		schema

Responses

204 - The template was deleted successfully.

Status: No Content

Schema

409 - Conflict

Status: Conflict

Schema

GenericPublicError

Get a specific alert rule by UID. (*RouteGetAlertRule*)

```
GET /api/v1/provisioning/alert-rules/:uid
```

Copy

Parameters

Name	Source	Type	Go type	Separator	Required	Default	Description
UID	`path`	string	`string`		✓		Alert rule UID

All responses

Code	Status	Description	Has headers	Schema
200	OK	ProvisionedAlertRule		schema
404	Not Found	Not found.		schema

Responses

200 - ProvisionedAlertRule

Status: OK

Schema

ProvisionedAlertRule

404 - Not found.

Status: Not Found

Schema

Export an alert rule in provisioning file format. (*RouteGetAlertRuleExport*)

```
GET /api/v1/provisioning/alert-rules/:uid/export
```

Copy

Produces

- application/json
- application/yaml
- application/terraform+hcl
- text/yaml
- text/hcl

Parameters

Name	Source	Type	Go type	Separator	Required	Default	Descriptio
UID	`path`	string	`string`		✓		Alert rule UID
download	`query`	boolean	`bool`				Whether to initiate a download the file or not.
format	`query`	string	`string`			`"yaml"`	Format of the downloade file, either yaml, json or hcl. Accept header car also be used, but the query parameter will take precedenc

All responses

Code	Status	Description	Has headers	Schema
200	OK	AlertingFileExport		schema
404	Not Found	Not found.		schema

Responses

200 - AlertingFileExport

Status: OK

Schema

AlertingFileExport

404 - Not found.

Status: Not Found

Schema

Get a rule group. (*RouteGetAlertRuleGroup*)

```
GET /api/v1/provisioning/folder/:folderUid/rule-groups/:group
```

Copy

Parameters

Name	Source	Type	Go type	Separator	Required	Default	Description
FolderUID	`path`	string	`string`		✓		
Group	`path`	string	`string`		✓		

All responses

Code	Status	Description	Has headers	Schema
200	OK	AlertRuleGroup		schema

Code	Status	Description	Has headers	Schema
404	Not Found	Not found.		schema

Responses

200 - AlertRuleGroup

Status: OK
Schema

AlertRuleGroup

404 - Not found.

Status: Not Found
Schema

Export an alert rule group in provisioning file format. (*RouteGetAlertRuleGroupExport*)

```
GET /api/v1/provisioning/folder/:folderUid/rule-groups/:group/export
```

Copy

Produces

- application/json
- application/yaml
- application/terraform+hcl
- text/yaml
- text/hcl

Parameters

Name	Source	Type	Go type	Separator	Required	Default	Descriptio
FolderUID	`path`	string	`string`		✓		
Group	`path`	string	`string`		✓		
download	`query`	boolean	`bool`				Whether t initiate a download

Name	Source	Type	Go type	Separator	Required	Default	Description
							the file or not.
format	`query`	string	`string`			`"yaml"`	Format of the downloade file, either yaml, json or hcl. Accept header ca also be used, but the query parameter will take precedenc

All responses

Code	Status	Description	Has headers	Schema
200	OK	AlertingFileExport		schema
404	Not Found	Not found.		schema

Responses

200 - AlertingFileExport

Status: OK

Schema

AlertingFileExport

404 - Not found.

Status: Not Found

Schema

Get all the alert rules. (*RouteGetAlertRules*)

```
GET /api/v1/provisioning/alert-rules
```

Copy

All responses

Code	Status	Description	Has headers	Schema
200	OK	ProvisionedAlertRules		schema

Responses

200 - ProvisionedAlertRules

Status: OK

Schema

ProvisionedAlertRules

Export all alert rules in provisioning file format. (*RouteGetAlertRulesExport*)

```
GET /api/v1/provisioning/alert-rules/export
```

Copy

Produces

- application/json
- application/yaml
- application/terraform+hcl
- text/yaml
- text/hcl

Parameters

Name	Source	Type	Go type	Separator	Required	Default	Descriptio
download	`query`	boolean	`bool`				Whether to initiate a download the file or not.
format	`query`	string	`string`			`"yaml"`	Format of the downloade file, either yaml, json or hcl. Accept

Name	Source	Type	Go type	Separator	Required	Default	Descriptio
							header car also be used, but the query parameter will take precedenc

All responses

Code	Status	Description	Has headers	Schema
200	OK	AlertingFileExport		schema
404	Not Found	Not found.		schema

Responses

200 - AlertingFileExport

Status: OK

Schema

AlertingFileExport

404 - Not found.

Status: Not Found

Schema

Get all the contact points. (*RouteGetContactpoints*)

```
GET /api/v1/provisioning/contact-points
```

Copy

Parameters

Name	Source	Type	Go type	Separator	Required	Default	Description
name	`query`	string	`string`				Filter by name

All responses

Code	Status	Description	Has headers	Schema
200	OK	ContactPoints		schema

Responses

200 - ContactPoints

Status: OK

Schema

ContactPoints

Export all contact points in provisioning file format. (*RouteGetContactpointsExport*)

```
GET /api/v1/provisioning/contact-points/export
```

Copy

Produces

- application/json
- application/yaml
- application/terraform+hcl
- text/yaml
- text/hcl

Parameters

Name	Source	Type	Go type	Separator	Required	Default	Descriptio
decrypt	query	boolean	bool				Whether a contained secure settings should be decrypted left redact

Name	Source	Type	Go type	Separator	Required	Default	Descriptio
							Redacted settings wi contain RedactedV instead. Currently, org admin view decrypted secure settings.
download	`query`	boolean	`bool`				Whether to initiate a download the file or r
format	`query`	string	`string`			`"yaml"`	Format of t downloade file, either yaml, json hcl. Accep header car also be use but the qu parameter take precedenc
name	`query`	string	`string`				Filter by na

All responses

Code	Status	Description	Has headers	Schema
200	OK	AlertingFileExport		schema
403	Forbidden	PermissionDenied		schema

Responses

200 - AlertingFileExport

Status: OK

Schema

AlertingFileExport

403 - PermissionDenied

Status: Forbidden

Schema

PermissionDenied

Get a mute timing. (*RouteGetMuteTiming*)

```
GET /api/v1/provisioning/mute-timings/:name
```

Copy

Parameters

Name	Source	Type	Go type	Separator	Required	Default	Description
name	`path`	string	`string`		✓		Mute timing name

All responses

Code	Status	Description	Has headers	Schema
200	OK	MuteTimeInterval		schema
404	Not Found	Not found.		schema

Responses

200 - MuteTimeInterval

Status: OK

Schema

MuteTimeInterval

404 - Not found.

Status: Not Found

Schema

Get all the mute timings. (*RouteGetMuteTimings*)

```
GET /api/v1/provisioning/mute-timings
```

Copy

All responses

Code	Status	Description	Has headers	Schema
200	OK	MuteTimings		schema

Responses

200 - MuteTimings

Status: OK

Schema

MuteTimings

Export all mute timings in provisioning file format. (*RouteGetMuteTimingsExport*)

```
GET /api/v1/provisioning/mute-timings/export
```

Copy

Produces

- application/json
- application/yaml
- application/terraform+hcl
- text/yaml
- text/hcl

Parameters

Name	Source	Type	Go type	Separator	Required	Default	Descriptio
download	`query`	boolean	`bool`				Whether to initiate a download the file or not.
format	`query`	string	`string`			`"yaml"`	Format of the

Name	Source	Type	Go type	Separator	Required	Default	Descriptio
							downloade file, either yaml, json or hcl. Accept header car also be used, but the query parameter will take precedenc

All responses

Code	Status	Description	Has headers	Schema
200	OK	MuteTimingsExport		schema
403	Forbidden	PermissionDenied		schema

Responses

200 - MuteTimingsExport

Status: OK

Schema

AlertingFileExport

403 - PermissionDenied

Status: Forbidden

Schema

PermissionDenied

Export a mute timing in provisioning file format. (*RouteGetMuteTimingExport*)

```
GET /api/v1/provisioning/mute-timings/:name/export
```

Copy

Produces

- application/json

- application/yaml
- application/terraform+hcl
- text/yaml
- text/hcl

Parameters

Name	Source	Type	Go type	Separator	Required	Default	Descriptio
name	`path`	string	`string`		✓		Mute timin name.
download	`query`	boolean	`bool`				Whether to initiate a download the file or not.
format	`query`	string	`string`			`"yaml"`	Format of the downloade file, either yaml, json or hcl. Accept header car also be used, but the query parameter will take precedenc

All responses

Code	Status	Description	Has headers	Schema
200	OK	MuteTimingExport		schema
403	Forbidden	PermissionDenied		schema

Responses

200 - MuteTimingExport

Status: OK

Schema

AlertingFileExport

403 - PermissionDenied

Status: Forbidden

Schema

PermissionDenied

Get the notification policy tree. (*RouteGetPolicyTree*)

```
GET /api/v1/provisioning/policies
```

Copy

All responses

Code	Status	Description	Has headers	Schema
200	OK	Route		schema

Responses

200 - Route

Status: OK

Schema

Route

Export the notification policy tree in provisioning file format. (*RouteGetPolicyTreeExport*)

```
GET /api/v1/provisioning/policies/export
```

Copy

Produces

- application/json
- application/yaml
- application/terraform+hcl
- text/yaml
- text/hcl

Parameters

Name	Source	Type	Go type	Separator	Required	Default	Descriptio
download	`query`	boolean	`bool`				Whether to initiate a download the file or not.
format	`query`	string	`string`			`"yaml"`	Format of the downloade file, either yaml, json or hcl. Accept header can also be used, but the query parameter will take precedenc

All responses

Code	Status	Description	Has headers	Schema
200	OK	AlertingFileExport		schema
404	Not Found	NotFound		schema

Responses

200 - AlertingFileExport

Status: OK

Schema

AlertingFileExport

404 - NotFound

Status: Not Found

Schema

NotFound

Get a notification template. (*RouteGetTemplate*)

```
GET /api/v1/provisioning/templates/:name
```

Copy

Parameters

Name	Source	Type	Go type	Separator	Required	Default	Description
name	`path`	string	`string`		✓		Template Name

All responses

Code	Status	Description	Has headers	Schema
200	OK	NotificationTemplate		schema
404	Not Found	Not found.		schema

Responses

200 - NotificationTemplate

Status: OK

Schema

NotificationTemplate

404 - Not found.

Status: Not Found

Schema

Get all notification templates. (*RouteGetTemplates*)

```
GET /api/v1/provisioning/templates
```

Copy

All responses

Code	Status	Description	Has headers	Schema
200	OK	NotificationTemplates		schema
404	Not Found	Not found.		schema

Responses

200 - NotificationTemplates

Status: OK

Schema

NotificationTemplates

404 - Not found.

Status: Not Found

Schema

Create a new alert rule. (*RoutePostAlertRule*)

```
POST /api/v1/provisioning/alert-rules
```

Copy

Parameters

Name	Source	Type	Go type	Separator	Requ
X-Disable-Provenance: true	`header`	string	`string`		
Body	`body`	ProvisionedAlertRule	`models.ProvisionedAlertRule`		

All responses

Code	Status	Description	Has headers	Schema
201	Created	ProvisionedAlertRule		schema

Code	Status	Description	Has headers	Schema
400	Bad Request	ValidationError		schema

Responses

201 - ProvisionedAlertRule

Status: Created

Schema

ProvisionedAlertRule

400 - ValidationError

Status: Bad Request

Schema

ValidationError

Create a contact point. (*RoutePostContactpoints*)

```
POST /api/v1/provisioning/contact-points
```

Copy

Parameters

Name	Source	Type	Go type	Separator	R
X-Disable-Provenance: true	`header`	string	`string`		
Body	`body`	EmbeddedContactPoint	`models.EmbeddedContactPoint`		

All responses

Code	Status	Description	Has headers	Schema
202	Accepted	EmbeddedContactPoint		schema
400	Bad Request	ValidationError		schema

Responses

202 - EmbeddedContactPoint

Status: Accepted

Schema

EmbeddedContactPoint

400 - ValidationError

Status: Bad Request

Schema

ValidationError

Create a new mute timing. (*RoutePostMuteTiming*)

```
POST /api/v1/provisioning/mute-timings
```

Copy

Parameters

Name	Source	Type	Go type	Separator	Required
X-Disable-Provenance: true	`header`	string	`string`		
Body	`body`	MuteTimeInterval	`models.MuteTimeInterval`		

All responses

Code	Status	Description	Has headers	Schema
201	Created	MuteTimeInterval		schema
400	Bad Request	ValidationError		schema

Responses

201 - MuteTimeInterval

Status: Created

Schema

MuteTimeInterval

400 - ValidationError

Status: Bad Request

Schema

ValidationError

Update an existing alert rule. (*RoutePutAlertRule*)

```
PUT /api/v1/provisioning/alert-rules/:uid
```

Copy

Parameters

Name	Source	Type	Go type	Separator	Requ
UID	`path`	string	`string`		✓
X-Disable-Provenance: true	`header`	string	`string`		
Body	`body`	ProvisionedAlertRule	`models.ProvisionedAlertRule`		

All responses

Code	Status	Description	Has headers	Schema
200	OK	ProvisionedAlertRule		schema
400	Bad Request	ValidationError		schema

Responses

200 - ProvisionedAlertRule

Status: OK

Schema

ProvisionedAlertRule

400 - ValidationError

Status: Bad Request

Schema

ValidationError

Update the interval or alert rules of a rule group. (*RoutePutAlertRuleGroup*)

```
PUT /api/v1/provisioning/folder/:folderUid/rule-groups/:group
```

Copy

Parameters

Name	Source	Type	Go type	Separator	Required
FolderUID	`path`	string	`string`		✓
Group	`path`	string	`string`		✓
X-Disable-Provenance: true	`header`	string	`string`		
Body	`body`	AlertRuleGroup	`models.AlertRuleGroup`		

All responses

Code	Status	Description	Has headers	Schema
200	OK	AlertRuleGroup		schema
400	Bad Request	ValidationError		schema

Responses

200 - AlertRuleGroup

Status: OK

Schema

AlertRuleGroup

400 - ValidationError

Status: Bad Request

Schema

ValidationError

Update an existing contact point. (*RoutePutContactpoint*)

```
PUT /api/v1/provisioning/contact-points/:uid
```

Copy

Parameters

Name	Source	Type	Go type	Separator	R
UID	`path`	string	`string`		✓
X-Disable-Provenance: true	`header`	string	`string`		
Body	`body`	EmbeddedContactPoint	`models.EmbeddedContactPoint`		

All responses

Code	Status	Description	Has headers	Schema
202	Accepted	Ack		schema
400	Bad Request	ValidationError		schema

Responses

202 - Ack

Status: Accepted

Schema

Ack

400 - ValidationError

Status: Bad Request

Schema

ValidationError

Replace an existing mute timing. (*RoutePutMuteTiming*)

```
PUT /api/v1/provisioning/mute-timings/:name
```

Copy

Parameters

Name	Source	Type	Go type	Separator	Required
name	`path`	string	`string`		✓
X-Disable-Provenance: true	`header`	string	`string`		
Body	`body`	MuteTimeInterval	`models.MuteTimeInterval`		

All responses

Code	Status	Description	Has headers	Schema
200	OK	MuteTimeInterval		schema
400	Bad Request	ValidationError		schema
409	Conflict	GenericPublicError		schema

Responses

200 - MuteTimeInterval

Status: OK

Schema

MuteTimeInterval

400 - ValidationError

Status: Bad Request

Schema

ValidationError

409 - Conflict

Status: Conflict

Schema

GenericPublicError

Sets the notification policy tree. (*RoutePutPolicyTree*)

```
PUT /api/v1/provisioning/policies
```

Copy

Parameters

Name	Source	Type	Go type	Separator	Required	Default	Desc
X-Disable-Provenance: true	`header`	string	`string`				Allov editi prov reso in th Graf
Body	`body`	Route	`models.Route`				The notif routi to us

All responses

Code	Status	Description	Has headers	Schema
202	Accepted	Ack		schema
400	Bad Request	ValidationError		schema

Responses

202 - Ack

Status: Accepted

Schema

Ack

400 - ValidationError

Status: Bad Request

Schema

ValidationError

Create or update a notification template. (*RoutePutTemplate*)

```
PUT /api/v1/provisioning/templates/:name
```

Copy

Parameters

Name	Source	Type	Go type	Separa
name	`path`	string	`string`	
X-Disable-Provenance: true	`header`	string	`string`	
Body	`body`	NotificationTemplateContent	`models.NotificationTemplateContent`	

All responses

Code	Status	Description	Has headers	Schema
202	Accepted	NotificationTemplate		schema
400	Bad Request	ValidationError		schema
409	Conflict	GenericPublicError		schema

Responses

202 - NotificationTemplate

Status: Accepted

Schema

NotificationTemplate

400 - ValidationError

Status: Bad Request

Schema

ValidationError

409 - Conflict

Status: Conflict

Schema

GenericPublicError

Clears the notification policy tree. (*RouteResetPolicyTree*)

```
DELETE /api/v1/provisioning/policies
```

Copy

All responses

Code	Status	Description	Has headers	Schema
202	Accepted	Ack		schema

Responses

202 - Ack

Status: Accepted

Schema

Ack

Models

Ack

interface{}

AlertQuery

Properties

Name	Type	Go type	Required	Default	Descriptio
datasourceUid	string	`string`			Grafana data sourc unique identifier; i should be '**expr**' for a Server Sid Expressior operation.
model	interface{}	`interface{}`			JSON is th raw JSON query and includes the above properties as well as custom properties
queryType	string	`string`			QueryType is an optional identifier for the typ of query.
It can be used to distinguish different types of queries.					
refId	string	`string`			RefID is th unique identifier c the query, set by the frontend call.
relativeTimeRange	RelativeTimeRange	`RelativeTimeRange`			

AlertQueryExport

Properties

Name	Type	Go type	Required	Default	Descriptic
datasourceUid	string	`string`			
model	interface{}	`interface{}`			
queryType	string	`string`			
refId	string	`string`			
relativeTimeRange	RelativeTimeRange	`RelativeTimeRange`			

AlertRuleExport

Properties

Name	Type	Go type	Required	Default	Description
annotations	map of string	`map[string]string`			
condition	string	`string`			
dashboardUid	string	`string`			
data	[]AlertQueryExport	`[]*AlertQueryExport`			
execErrState	string	`string`			
for	Duration	`Duration`			
isPaused	boolean	`bool`			
labels	map of string	`map[string]string`			
noDataState	string	`string`			
panelId	int64 (formatted integer)	`int64`			
title	string	`string`			
uid	string	`string`			

AlertRuleGroup

Properties

Name	Type	Go type	Required	Default	Descriptio
folderUid	string	`string`			
interval	int64 (formatted integer)	`int64`			
rules	[]ProvisionedAlertRule	`[]*ProvisionedAlertRule`			
title	string	`string`			

AlertRuleGroupExport

Properties

Name	Type	Go type	Required	Default	Description	Exa
folder	string	`string`				
interval	Duration	`Duration`				
name	string	`string`				
orgId	int64 (formatted integer)	`int64`				
rules	[]AlertRuleExport	`[]*AlertRuleExport`				

AlertingFileExport

Properties

Name	Type	Go type	Required	Default
apiVersion	int64 (formatted integer)	`int64`		
contactPoints	[]ContactPointExport	`[]*ContactPointExport`		
groups	[]AlertRuleGroupExport	`[]*AlertRuleGroupExport`		
policies	[]NotificationPolicyExport	`[]*NotificationPolicyExport`		

ContactPointExport

Properties

Name	Type	Go type	Required	Default	Description	Exa
name	string	`string`				
orgId	int64 (formatted integer)	`int64`				
receivers	[]ReceiverExport	`[]*ReceiverExport`				

ContactPoints

[]EmbeddedContactPoint

Duration

Name	Type	Go type	Default	Description	Example
Duration	string	int64			

EmbeddedContactPoint

EmbeddedContactPoint is the contact point type that is used by grafanas embedded alertmanager implementation.

Properties

Name	Type	Go type	Required	Default	Description
disableResolveMessage	boolean	`bool`			
name	string	`string`			Name is used as grouping key in the UI. Contact points with the same name will be grouped in the UI.
webhook_1					

Name	Type	Go type	Required	Default	Description
provenance	string	`string`			
settings	JSON	`JSON`	✓		
type	string	`string`	✓		
uid	string	`string`			UID is the unique identifier of the contact point. The UID can be
set by the user.	`my_external_reference`				

Json

interface{}

MatchRegexps

MatchRegexps

MatchType

Name	Type	Go type	Default	Description	Example
MatchType	int64 (formatted integer)	int64			

Matcher

Properties

Name	Type	Go type	Required	Default	Description	Example
Name	string	`string`				
Type	MatchType	`MatchType`				
Value	string	`string`				

Matchers

Matchers is a slice of Matchers that is sortable, implements Stringer, and provides a Matches method to match a LabelSet against all Matchers in the slice. Note that some users of Matchers might require it to be sorted.

[]Matcher

MuteTimeInterval

Properties

Name	Type	Go type	Required	Default	Description	Exa
name	string	`string`				
time_intervals	[]TimeInterval	`[]*TimeInterval`				
version	string	`string`			Version of resource	

MuteTimingExport

Properties

MuteTimingsExport

Properties

MuteTimings

[]MuteTimeInterval

NotFound

interface{}

NotificationPolicyExport

Properties

Name	Type	Go type	Required	Default	Description	Example
Policy	RouteExport	`RouteExport`			inline	

Name	Type	Go type	Required	Default	Description	Example
orgId	int64 (formatted integer)	`int64`				

NotificationTemplate

Properties

Name	Type	Go type	Required	Default	Description	Example
name	string	`string`				
provenance	Provenance	`Provenance`				
template	string	`string`				
version	string	`string`			Version of resource	

NotificationTemplateContent

Properties

Name	Type	Go type	Required	Default	Description	Example
template	string	`string`				
version	string	`string`			Version of resource. Should be empty for new templates.	

NotificationTemplates

[]NotificationTemplate

ObjectMatchers

Matchers

Inlined models

PermissionDenied

interface{}

Provenance

Name	Type	Go type	Default	Description	Example
Provenance	string	string			

ProvisionedAlertRule

Properties

Name	Type	Go type	Required	Default	Description	Exar
annotations	map of string	`map[string]string`				{"ru
condition	string	`string`	✓			A
data	[]AlertQuery	`[]*AlertQuery`	✓			[{"d {"par {"typ 1","h {"fro
execErrState	string	`string`	✓			
folderUID	string	`string`	✓			proj
for	Duration	`Duration`	✓			
id	int64 (formatted integer)	`int64`				
isPaused	boolean	`bool`				fals
labels	map of string	`map[string]string`				{"te
noDataState	string	`string`	✓			
orgID	int64 (formatted integer)	`int64`	✓			
provenance	Provenance	`Provenance`				

Name	Type	Go type	Required	Default	Description	Exar
ruleGroup	string	`string`	✓			`eval`
title	string	`string`	✓			`Alwa`
uid	string	`string`				
updated	date-time (formatted string)	`strfmt.DateTime`				

ProvisionedAlertRules

[]ProvisionedAlertRule

RawMessage

interface{}

ReceiverExport

Properties

Name	Type	Go type	Required	Default	Description
disableResolveMessage	boolean	`bool`			
settings	RawMessage	`RawMessage`			
type	string	`string`			
uid	string	`string`			

Regexp

A Regexp is safe for concurrent use by multiple goroutines, except for configuration methods, such as Longest.

interface{}

RelativeTimeRange

RelativeTimeRange is the per query start and end time for requests.

Properties

Name	Type	Go type	Required	Default	Description	Example
from	Duration	`Duration`				
to	Duration	`Duration`				

Route

A Route is a node that contains definitions of how to handle alerts. This is modified from the upstream alertmanager in that it adds the ObjectMatchers property.

Properties

Name	Type	Go type	Required	Default	Description
continue	boolean	`bool`			
group_by	[]string	`[]string`			
group_interval	string	`string`			
group_wait	string	`string`			
match	map of string	`map[string]string`			Deprecated Remove before v1.0 release.
match_re	MatchRegexps	`MatchRegexps`			
matchers	Matchers	`Matchers`			
mute_time_intervals	[]string	`[]string`			
object_matchers	ObjectMatchers	`ObjectMatchers`			
provenance	Provenance	`Provenance`			
receiver	string	`string`			
repeat_interval	string	`string`			

Name	Type	Go type	Required	Default	Description
routes	[]Route	`[]*Route`			

RouteExport

RouteExport is the provisioned file export of definitions.Route. This is needed to hide fields that aren't usable in provisioning file format. An alternative would be to define a custom MarshalJSON and MarshalYAML that excludes them.

Properties

Name	Type	Go type	Required	Default	Description
continue	boolean	`bool`			
group_by	[]string	`[]string`			
group_interval	string	`string`			
group_wait	string	`string`			
match	map of string	`map[string]string`			Deprecated Remove before v1.0 release.
match_re	MatchRegexps	`MatchRegexps`			
matchers	Matchers	`Matchers`			
mute_time_intervals	[]string	`[]string`			
object_matchers	ObjectMatchers	`ObjectMatchers`			
receiver	string	`string`			
repeat_interval	string	`string`			
routes	[]RouteExport	`[]*RouteExport`			

TimeInterval

TimeInterval describes intervals of time. ContainsTime will tell you if a golang time is contained within the interval.

Properties

Name	Type	Go type	Required	Default	Description	Examp
days_of_month	[]string	`[]string`				
location	string	`string`				
months	[]string	`[]string`				
times	[]TimeRange	`[]*TimeRange`				
weekdays	[]string	`[]string`				
years	[]string	`[]string`				

TimeRange

For example, 4:00PM to End of the day would Begin at 1020 and End at 1440.

Properties

Name	Type	Go type	Required	Default	Description	Example
end_time	string	`string`				`"end_time": "24:00"`
start_time	string	`string`				`"start_time": "18:00"`

ValidationError

Properties

Name	Type	Go type	Required	Default	Description	Example
msg	string	`string`				`error message`

GenericPublicError

Properties

Name	Type	Go type	Required	Default	Description	Example
statusCode	string	`string`	✓		HTTP Status Code	
messageId	string	`string`	✓		Unique code of the error	
message	string	`string`			Error message	
extra	map of any	`map[string]any`			Extra information about the error. Format is specific to the error code.	

Enable alerting high availability

You can enable alerting high availability support by updating the Grafana configuration file. If you run Grafana in a Kubernetes cluster, additional steps are required. Both options are described below. Please note that the deduplication is done for the notification, but the alert will still be evaluated on every Grafana instance. This means that events in alerting state history will be duplicated by the number of Grafana instances running.

NOTE

If using a mix of `execute_alerts=false` and `execute_alerts=true` on the HA nodes, since the alert state is not shared amongst the Grafana instances, the instances with `execute_alerts=false` will not show any alert status. This is because the HA settings (`ha_peers`, etc), only apply to the alert notification delivery (i.e. de-duplication of alert notifications, and silences, as mentioned above).

Enable alerting high availability in Grafana using Memberlist

Before you begin

Since gossiping of notifications and silences uses both TCP and UDP port `9094`, ensure that each Grafana instance is able to accept incoming connections on these ports.

To enable high availability support:

1. In your custom configuration file ($WORKING_DIR/conf/custom.ini), go to the `[unified_alerting]` section.
2. Set `[ha_peers]` to the number of hosts for each Grafana instance in the cluster (using a format of host:port), for example, `ha_peers=10.0.0.5:9094,10.0.0.6:9094,10.0.0.7:9094`. You must have at least one (1) Grafana instance added to the `ha_peers` section.

3. Set `[ha_listen_address]` to the instance IP address using a format of `host:port` (or the Pod's IP in the case of using Kubernetes). By default, it is set to listen to all interfaces (`0.0.0.0`).
4. Set `[ha_peer_timeout]` in the `[unified_alerting]` section of the custom.ini to specify the time to wait for an instance to send a notification via the Alertmanager. The default value is 15s, but it may increase if Grafana servers are located in different geographic regions or if the network latency between them is high.

Enable alerting high availability in Grafana using Redis

As an alternative to Memberlist, you can use Redis for high availability. This is useful if you want to have a central database for HA and cannot support the meshing of all Grafana servers.

1. Make sure you have a redis server that supports pub/sub. If you use a proxy in front of your redis cluster, make sure the proxy supports pub/sub.
2. In your custom configuration file ($WORKING_DIR/conf/custom.ini), go to the `[unified_alerting]` section.
3. Set `ha_redis_address` to the redis server address Grafana should connect to.
4. Optional: Set the username and password if authentication is enabled on the redis server using `ha_redis_username` and `ha_redis_password`.
5. Optional: Set `ha_redis_prefix` to something unique if you plan to share the redis server with multiple Grafana instances.

The following metrics can be used for meta monitoring, exposed by Grafana's `/metrics` endpoint:

Metric	Description
alertmanager_cluster_messages_received_total	Total number of cluster messages received.
alertmanager_cluster_messages_received_size_total	Total size of cluster messages received.
alertmanager_cluster_messages_sent_total	Total number of cluster messages sent.
alertmanager_cluster_messages_sent_size_total	Total number of cluster messages received.
alertmanager_cluster_messages_publish_failures_total	Total number of messages that failed to be published.
alertmanager_cluster_members	Number indicating current number of members in cluster.
alertmanager_peer_position	Position the Alertmanager instance believes it's in. The position determines a peer's behavior in the cluster.
alertmanager_cluster_pings_seconds	Histogram of latencies for ping messages.
alertmanager_cluster_pings_failures_total	Total number of failed pings.

Enable alerting high availability using Kubernetes

If you are using Kubernetes, you can expose the pod IP through an environment variable via the container definition.

```yaml
env:
  - name: POD_IP
    valueFrom:
      fieldRef:
        fieldPath: status.podIP
```

1. Add the port 9094 to the Grafana deployment:

```yaml
ports:
  - containerPort: 3000
    name: http-grafana
    protocol: TCP
  - containerPort: 9094
    name: grafana-alert
    protocol: TCP
```

1. Add the environment variables to the Grafana deployment:

```yaml
env:
  - name: POD_IP
    valueFrom:
      fieldRef:
        fieldPath: status.podIP
```

1. Create a headless service that returns the pod IP instead of the service IP, which is what the `ha_peers` need:

```yaml
apiVersion: v1
kind: Service
```

```yaml
metadata:
  name: grafana-alerting
  namespace: grafana
  labels:
    app.kubernetes.io/name: grafana-alerting
    app.kubernetes.io/part-of: grafana
spec:
  type: ClusterIP
  clusterIP: 'None'
  ports:
    - port: 9094
  selector:
    app: grafana
```

1. Make sure your grafana deployment has the label matching the selector, e.g. `app:grafana`:
2. Add in the grafana.ini:

bash Copy

```
[unified_alerting]
enabled = true
ha_listen_address = "${POD_IP}:9094"
ha_peers = "grafana-alerting.grafana:9094"
ha_advertise_address = "${POD_IP}:9094"
ha_peer_timeout = 15s
```

Configure Alert State History

Starting with Grafana 10, Alerting can record all alert rule state changes for your Grafana managed alert rules in a Loki instance.

This allows you to explore the behavior of your alert rules in the Grafana explore view and levels up the existing state history modal with a powerful new visualisation.

Configuring Loki

To set up alert state history, make sure to have a Loki instance Grafana can write data to. The default settings might need some tweaking as the state history modal might query up to 30 days of data.

The following change to the default configuration should work for most instances, but we recommend looking at the full Loki configuration settings and adjust according to your needs.

As this might impact the performances of an existing Loki instance, we recommend using a separate Loki instance for the alert state history.

yaml Copy

```yaml
limits_config:
  split_queries_by_interval: '24h'
  max_query_parallelism: 32
```

Configuring Grafana

We need some additional configuration in the Grafana configuration file to have it working with the alert state history.

The example below instructs Grafana to write alert state history to a local Loki instance:

toml

Copy

```toml
[unified_alerting.state_history]
enabled = true
backend = "loki"
loki_remote_url = "http://localhost:3100"

[feature_toggles]
enable = alertStateHistoryLokiSecondary, alertStateHistoryLokiPrimary, alertStateHistoryLokiOnly
```

Adding the Loki data source

See our instructions on adding a data source.

Querying the history

If everything is set up correctly you can use the Grafana Explore view to start querying the Loki data source.

A simple litmus test to see if data is being written correctly into the Loki instance is the following query:

logQL

Copy

```logql
{ from="state-history" } | json
```

Performance considerations and limitations

Grafana Alerting supports multi-dimensional alerting, where one alert rule can generate many alerts. For example, you can configure an alert rule to fire an alert every time the CPU of individual VMs max out. This topic discusses performance considerations resulting from multi-dimensional alerting.

Evaluating alerting rules consumes RAM and CPU to compute the output of an alerting query, and network resources to send alert notifications and write the results to the Grafana SQL database. The configuration of individual alert rules affects the resource consumption and, therefore, the maximum number of rules a given configuration can support.

The following section provides a list of alerting performance considerations.

- Frequency of rule evaluation consideration. The “Evaluate Every” property of an alert rule controls the frequency of rule evaluation. We recommend using the lowest acceptable evaluation frequency to support more concurrent rules.
- Cardinality of the rule’s result set. For example, suppose you are monitoring API response errors for every API path, on every VM in your fleet. This set has a cardinality of *n* number of paths multiplied by *v* number of VMs. You can reduce the cardinality of a result set - perhaps by monitoring errors-per-VM instead of for each path per VM.
- Complexity of the alerting query consideration. Queries that data sources can process and respond to quickly consume fewer resources. Although this consideration is less important than the other considerations listed above, if you have reduced those as much as possible, looking at individual query performance could make a difference.

Each evaluation of an alert rule generates a set of alert instances; one for each member of the result set. The state of all the instances is written to the `alert_instance` table in Grafana’s SQL database. This number of write-heavy operations can cause issues when using SQLite.

Grafana Alerting exposes a metric, `grafana_alerting_rule_evaluations_total` that counts the number of alert rule evaluations. To get a feel for the influence of rule evaluations on your Grafana instance, you can observe the rate of evaluations and compare it with resource consumption. In a Prometheus-compatible database, you can use the query

`rate(grafana_alerting_rule_evaluations_total[5m])` to compute the rate over 5 minute windows of time. It's important to remember that this isn't the full picture of rule evaluation. For example, the load will be unevenly distributed if you have some rules that evaluate every 10 seconds, and others every 30 minutes.

These factors all affect the load on the Grafana instance, but you should also be aware of the performance impact that evaluating these rules has on your data sources. Alerting queries are often the vast majority of queries handled by monitoring databases, so the same load factors that affect the Grafana instance affect them as well.

Limited rule sources support

Grafana Alerting can retrieve alerting and recording rules **stored** in most available Prometheus, Loki, Mimir, and Alertmanager compatible data sources.

It does not support reading or writing alerting rules from any other data sources but the ones previously mentioned at this time.

Prometheus version support

We support the latest two minor versions of both Prometheus and Alertmanager. We cannot guarantee that older versions will work.

As an example, if the current Prometheus version is `2.31.1`, we support >= `2.29.0`.

The Grafana Alertmanager can only receive Grafana managed alerts

Grafana cannot be used to receive external alerts. You can only send alerts to the Grafana Alertmanager using Grafana managed alerts.

You have the option to send Grafana managed alerts to an external Alertmanager, you can find this option in the admin tab on the Alerting page.

For more information, refer to this GitHub discussion.

High load on database caused by a high number of alert instances

If you have a high number of alert instances, it can happen that the load on the database gets very high, as each state transition of an alert instance will be saved in the database.

This can be prevented by writing to the database periodically. For this the feature flag `alertingSaveStatePeriodic` needs to be enabled. By default it will save the states every 5 minutes to the database and on each shutdown. The periodic interval can also be configured using the `state_periodic_save_interval` configuration flag.

The time it takes to write to the database periodically can be monitored using the `state_full_sync_duration_seconds` metric that is exposed by Grafana.

If Grafana crashes or is force killed, then the database can be up to `state_periodic_save_interval` seconds out of date. When Grafana restarts, the UI might show incorrect state for some alerts until

the alerts are re-evaluated. In some cases, alerts that were firing before the crash might fire again. If this happens, Grafana might send duplicate notifications for firing alerts.

Configure Alerting

Configure the features and integrations that you need to create and manage your alerts.

Configure alert rules

Configure Grafana-managed alert rules.

Configure data source-managed alert rules

Configure recording rules

Recording rules are only available for compatible Prometheus or Loki data sources.

For more information, see Configure recording rules.

Configure contact points

For information on how to configure contact points, see Configure contact points.

Configure notification policies

For information on how to configure notification policies, see Configure notification policies.

Configure Grafana-managed alert rules

Grafana-managed rules are the most flexible alert rule type. They allow you to create alerts that can act on data from any of our supported data sources. In addition to supporting multiple data sources, you can also add expressions to transform your data and set alert conditions. Using images in alert notifications is also supported. This is the only type of rule that allows alerting from multiple data sources in a single rule definition.

Multiple alert instances can be created as a result of one alert rule (also known as a multi-dimensional alerting).

NOTE

For Grafana Cloud, there are limits on how many Grafana-managed alert rules you can create. These are as follows:

- Free: 100 alert rules
- Paid: 2000 alert rules

Grafana managed alert rules can only be edited or deleted by users with Edit permissions for the folder storing the rules.

If you delete an alerting resource created in the UI, you can no longer retrieve it. To make a backup of your configuration and to be able to restore deleted alerting resources, create your alerting resources using file provisioning, Terraform, or the Alerting API.

Watch this video to learn more about creating alert rules:

In the following sections, we'll guide you through the process of creating your Grafana-managed alert rules.

To create a Grafana-managed alert rule, use the in-product alert creation flow and follow these steps to help you.

Set alert rule name

1. Click **Alerts & IRM → Alert rules → + New alert rule**.
2. Enter a name to identify your alert rule.

   This name is displayed in the alert rule list. It is also the `alertname` label for every alert instance that is created from this rule.

Define query and condition

Define a query to get the data you want to measure and a condition that needs to be met before an alert rule fires.

1. Select a data source.
2. From the **Options** dropdown, specify a time range.

   Note:

   Grafana Alerting only supports fixed relative time ranges, for example, `now-24hr: now`.

   It does not support absolute time ranges: `2021-12-02 00:00:00 to 2021-12-05 23:59:592` or semi-relative time ranges: `now/d to: now`.
3. Add a query.

   To add multiple queries, click **Add query**.

   All alert rules are managed by Grafana by default. If you want to switch to a data source-managed alert rule, click **Switch to data source-managed alert rule**.
4. Add one or more expressions.

   a. For each expression, select either **Classic condition** to create a single alert rule, or choose from the **Math**, **Reduce**, and **Resample** options to generate separate alert for each series.

   > **NOTE**
   >
   > When using Prometheus, you can use an instant vector and built-in functions, so you don't need to add additional expressions.

   b. Click **Preview** to verify that the expression is successful.

> **NOTE**
>
> The recovery threshold feature is currently only available in OSS.

1. To add a recovery threshold, turn the **Custom recovery threshold** toggle on and fill in a value for when your alert rule should stop firing.

   You can only add one recovery threshold in a query and it must be the alert condition.

2. Click **Set as alert condition** on the query or expression you want to set as your alert condition.

Set alert evaluation behavior

Use alert rule evaluation to determine how frequently an alert rule should be evaluated and how quickly it should change its state.

To do this, you need to make sure that your alert rule is in the right evaluation group and set a pending period time that works best for your use case.

1. Select a folder or click **+ New folder**.
2. Select an evaluation group or click **+ New evaluation group**.

   If you are creating a new evaluation group, specify the interval for the group.

   All rules within the same group are evaluated concurrently over the same time interval.

3. Enter a pending period.

   The pending period is the period in which an alert rule can be in breach of the condition until it fires.

   Once a condition is met, the alert goes into the **Pending** state. If the condition remains active for the duration specified, the alert transitions to the **Firing** state, else it reverts to the **Normal** state.

4. Turn on pause alert notifications, if required.

   Note:

   Pause alert rule evaluation to prevent noisy alerting while tuning your alerts. Pausing stops alert rule evaluation and does not create any alert instances. This is different to mute timings, which stop notifications from being delivered, but still allow for alert rule evaluation and the creation of alert instances.

   You can pause alert rule evaluation to prevent noisy alerting while tuning your alerts. Pausing stops alert rule evaluation and does not create any alert instances. This is different to mute timings, which stop notifications from being delivered, but still allow for alert rule evaluation and the creation of alert instances.

5. In **Configure no data and error handling**, configure alerting behavior in the absence of data.

   Use the guidelines in No data and error handling.

Configure notifications

> **NOTE**
>
> To try out a simplified version of routing your alerts, enable the alertingSimplifiedRouting feature toggle and refer to the following section Configure notifications (simplified).

1. Add labels to your alert rules to set which notification policy should handle your firing alert instances.

   All alert rules and instances, irrespective of their labels, match the default notification policy. If there are no nested policies, or no nested policies match the labels in the alert rule or alert instance, then the default notification policy is the matching policy.

   Add labels if you want to change the way your notifications are routed.

   Add custom labels by selecting existing key-value pairs from the drop down, or add new labels by entering the new key or value.

2. Preview your alert instance routing set up.

   Based on the labels added, alert instances are routed to the following notification policies displayed.

   Expand each notification policy below to view more details.

3. Click See details to view alert routing details and an email preview.

4. Click **Save rule**.

Configure notifications (simplified)

> **NOTE**
>
> To try this out, enable the alertingSimplifiedRouting feature toggle.
>
> This feature is currently not available for Grafana Cloud.

In the **Labels** section, you can optionally choose whether to add labels to organize your alert rules, make searching easier, as well as set which notification policy should handle your firing alert instance.

In the **Configure notifications** section, you can choose to select a contact point directly from the alert rule form or choose to use notification policy routing as well as set up mute timings and groupings.

Complete the following steps to set up labels and notifications.

1. Add labels, if required.

   Add custom labels by selecting existing key-value pairs from the drop down, or add new labels by entering the new key or value.

2. Configure who receives a notification when an alert rule fires by either choosing **Select contact point** or **Use notification policy**.

   Select contact point

   a. Choose this option to select an existing contact point.

      All notifications for this alert rule are sent to this contact point automatically and notification policies are not used.

   b. You can also optionally select a mute timing as well as groupings and timings to define when not to send notifications.

   > **NOTE**
   >
   > ```
   > An auto-generated notification policy is generated. Only admins can view these au
   > ```

   Use notification policy

   a. Choose this option to use the notification policy tree to direct your notifications.

   > **NOTE**
   >
   > ```
   > All alert rules and instances, irrespective of their labels, match the default no
   > ```

   b. Preview your alert instance routing set up.

      Based on the labels added, alert instances are routed to the following notification policies displayed.

   c. Expand each notification policy below to view more details.

   d. Click **See details** to view alert routing details and an email preview.

Add annotations

Add annotations. to provide more context on the alert in your alert notification message.

Annotations add metadata to provide more information on the alert in your alert notification message. For example, add a **Summary** annotation to tell you which value caused the alert to fire or which server it happened on.

1. Optional: Add a summary.

   Short summary of what happened and why.

2. Optional: Add a description.

   Description of what the alert rule does.

3. Optional: Add a Runbook URL.

   Webpage where you keep your runbook for the alert

4. Optional: Add a custom annotation
5. Optional: Add a dashboard and panel link.

   Links alerts to panels in a dashboard.

6. Click **Save rule**.

Single and multi-dimensional rule

For Grafana managed alerts, you can create a rule with a classic condition or you can create a multi-dimensional rule.

Rule with classic condition

Use the classic condition expression to create a rule that triggers a single alert when its condition is met. For a query that returns multiple series, Grafana does not track the alert state of each series. As a result, Grafana sends only a single alert even when alert conditions are met for multiple series.

For more information, see expressions documentation.

Multi-dimensional rule

To generate a separate alert for each series, create a multi-dimensional rule. Use `Math`, `Reduce`, or `Resample` expressions to create a multi-dimensional rule. For example:

- Add a `Reduce` expression for each query to aggregate values in the selected time range into a single value. (Not needed for rules using numeric data.
- Add a `Math` expression with the condition for the rule. Not needed in case a query or a reduce expression already returns 0 if rule should not fire, or a positive number if it should fire. Some examples: `$B > 70` if it should fire in case value of B query/expression is more than 70. `$B < $C * 100` in case it should fire if value of B is less than value of C multiplied by 100. If queries being compared have multiple series in their results, series from different queries are matched if they have the same labels or one is a subset of the other.

Note: Grafana does not support alert queries with template variables. More information is available at https://community.grafana.com/t/template-variables-are-not-supported-in-alert-queries-while-setting-up-alert/2514.

Configure no data and error handling

Configure alerting behavior when your alert rule evaluation returns no data or an error.

Note: Alert rules that are configured to fire when an evaluation returns no data or error only fire when the entire duration of the evaluation period has finished. This means that rather than immediately firing when the alert rule condition is breached, the alert rule waits until the time set as the **For** field has finished and then fires, reducing alert noise and allowing for temporary data availability issues.

If your alert rule evaluation returns no data, you can set the state on your alert rule to appear as follows:

No Data	Description
No Data	Creates a new alert `DatasourceNoData` with the name and UID of the alert rule, and UID of the datasource that returned no data as labels.
Alerting	Sets alert rule state to `Alerting`. The alert rule waits until the time set in the **For** field has finished before firing.
Ok	Sets alert rule state to `Normal`.

If your evaluation returns an error, you can set the state on your alert rule to appear as follows:

Error	Description
Error	Creates an alert instance `DatasourceError` with the name and UID of the alert rule, and UID of the datasource that returned no data as labels.
Alerting	Sets alert rule state to `Alerting`. The alert rule waits until the time set in the **For** field has finished before firing.
Ok	Sets alert rule state to `Normal`.

Resolve stale alert instances

An alert instance is considered stale if its dimension or series has disappeared from the query results entirely for two evaluation intervals.

Stale alert instances that are in the **Alerting**/**NoData**/**Error** states are automatically marked as **Resolved** and the grafana_state_reason annotation is added to the alert instance with the reason **MissingSeries**.

Create alerts from panels

Create alerts from any panel type. This means you can reuse the queries in the panel and create alerts based on them.

1. Navigate to a dashboard in the **Dashboards** section.
2. In the top right corner of the panel, click on the three dots (ellipses).
3. From the dropdown menu, select **More...** and then choose **New alert rule**.

This will open the alert rule form, allowing you to configure and create your alert based on the current panel's query.

Configure data source-managed alert rules

Create alert rules for an external Grafana Mimir or Loki instance that has ruler API enabled; these are called data source-managed alert rules.

Note:

Alert rules for an external Grafana Mimir or Loki instance can be edited or deleted by users with Editor or Admin roles.

If you delete an alerting resource created in the UI, you can no longer retrieve it. To make a backup of your configuration and to be able to restore deleted alerting resources, create your alerting resources using file provisioning, Terraform, or the Alerting API.

Before you begin

- Verify that you have write permission to the Prometheus or Loki data source. Otherwise, you will not be able to create or update Grafana Mimir managed alert rules.
- For Grafana Mimir and Loki data sources, enable the Ruler API by configuring their respective services.
  - **Loki** - The `local` rule storage type, default for the Loki data source, supports only viewing of rules. To edit rules, configure one of the other rule storage types.
  - **Grafana Mimir** - use the `/prometheus` prefix. The Prometheus data source supports both Grafana Mimir and Prometheus, and Grafana expects that both the Query API and Ruler API are under the same URL. You cannot provide a separate URL for the Ruler API.

Watch this video to learn more about how to create a Mimir managed alert rule:

NOTE

If you do not want to manage alert rules for a particular Loki or Prometheus data source, go to its settings and clear the **Manage alerts via Alerting UI** checkbox.

In the following sections, we'll guide you through the process of creating your data source-managed alert rules.

To create a data source-managed alert rule, use the in-product alert creation flow and follow these steps to help you.

Set alert rule name

1. Click **Alerts & IRM** → **Alert rules** → **+ New alert rule**.
2. Enter a name to identify your alert rule.

   This name is displayed in the alert rule list. It is also the `alertname` label for every alert instance that is created from this rule.

Define query and condition

Define a query to get the data you want to measure and a condition that needs to be met before an alert rule fires.

Note:

All alert rules are managed by Grafana by default. To switch to a data source-managed alert rule, click **Switch to data source-managed alert rule**.

1. Select a data source from the drop-down list.

   You can also click **Open advanced data source picker** to see more options, including adding a data source (Admins only).
2. Enter a PromQL or LogQL query.
3. Click **Preview alerts**.

Set alert evaluation behavior

Use alert rule evaluation to determine how frequently an alert rule should be evaluated and how quickly it should change its state.

1. Select a namespace or click **+ New namespace**.
2. Select an evaluation group or click **+ New evaluation group**.

   If you are creating a new evaluation group, specify the interval for the group.

   All rules within the same group are evaluated sequentially over the same time interval.
3. Enter a pending period.

The pending period is the period in which an alert rule can be in breach of the condition until it fires.

Once a condition is met, the alert goes into the **Pending** state. If the condition remains active for the duration specified, the alert transitions to the **Firing** state, else it reverts to the **Normal** state.

Configure notifications

Add labels to your alert rules to set which notification policy should handle your firing alert instances.

All alert rules and instances, irrespective of their labels, match the default notification policy. If there are no nested policies, or no nested policies match the labels in the alert rule or alert instance, then the default notification policy is the matching policy.

1. Add labels if you want to change the way your notifications are routed.

   Add custom labels by selecting existing key-value pairs from the drop down, or add new labels by entering the new key or value.

Add annotations

Add annotations. to provide more context on the alert in your alert notifications.

Annotations add metadata to provide more information on the alert in your alert notifications. For example, add a **Summary** annotation to tell you which value caused the alert to fire or which server it happened on.

1. Optional: Add a summary.

   Short summary of what happened and why.

2. Optional: Add a description.

   Description of what the alert rule does.

3. Optional: Add a Runbook URL.

   Webpage where you keep your runbook for the alert

4. Optional: Add a custom annotation
5. Optional: Add a dashboard and panel link.

   Links alerts to panels in a dashboard.

6. Click **Save rule**.

Configure recording rules

You can create and manage recording rules for an external Grafana Mimir or Loki instance. Recording rules calculate frequently needed expressions or computationally expensive expressions in advance and save the result as a new set of time series. Querying this new time series is faster, especially for dashboards since they query the same expression every time the dashboards refresh.

Note:

Recording rules are run as instant rules, which means that they run every 10s. To overwrite this configuration, update the min_interval in your custom configuration file.

min_interval sets the minimum interval to enforce between rule evaluations. The default value is 10s which equals the scheduler interval. Rules will be adjusted if they are less than this value or if they are not multiple of the scheduler interval (10s). Higher values can help with resource management as fewer evaluations are scheduled over time.

This setting has precedence over each individual rule frequency. If a rule frequency is lower than this value, then this value is enforced.

Before you begin

- Verify that you have write permission to the Prometheus or Loki data source. Otherwise, you will not be able to create or update Grafana Mimir managed alerting rules.
- For Grafana Mimir and Loki data sources, enable the ruler API by configuring their respective services.
  - **Loki** - The `local` rule storage type, default for the Loki data source, supports only viewing of rules. To edit rules, configure one of the other rule storage types.
  - **Grafana Mimir** - use the `/prometheus` prefix. The Prometheus data source supports both Grafana Mimir and Prometheus, and Grafana expects that both the Query API and Ruler API are under the same URL. You cannot provide a separate URL for the Ruler API.

Create recording rules

To create recording rules, follow these steps.

1. Click **Alerts & IRM** → **Alerting** → **Alert rules**.
2. Click **New recording rule**.
3. Set rule name.

   The recording rule name must be a Prometheus metric name and contain no whitespace.
4. Define query.
   - Select your Loki or Prometheus data source.
   - Enter a query.
5. Add namespace and group.
   - From the **Namespace** dropdown, select an existing rule namespace or add a new one. Namespaces can contain one or more rule groups and only have an organizational purpose.
   - From the **Group** dropdown, select an existing group within the selected namespace or add a new one. Newly created rules are appended to the end of the group. Rules within a group are run sequentially at a regular interval, with the same evaluation time.
6. Add labels.
   - Add custom labels selecting existing key-value pairs from the drop down, or add new labels by entering the new key or value .
7. Click **Save rule** to save the rule or **Save rule and exit** to save the rule and go back to the Alerting page.

Configure contact points

Use contact points to define how your contacts are notified when an alert rule fires. You can add, edit, delete, and test a contact point.

Add a contact point

Complete the following steps to add a contact point.

1. In the left-side menu, click **Alerts & IRM** and then **Alerting**.
2. Click **Contact points**.
3. From the **Choose Alertmanager** dropdown, select an Alertmanager. By default, **Grafana Alertmanager** is selected.
4. On the **Contact Points** tab, click **+ Add contact point**.
5. Enter a descriptive name for the contact point.
6. From **Integration**, select a type and fill out mandatory fields. For example, if you choose email, enter the email addresses. Or if you choose Slack, enter the Slack channel(s) and users who should be contacted.
7. Some contact point integrations, like email or webhook, have optional settings. In **Optional settings**, specify additional settings for the selected contact point integration.
8. In Notification settings, optionally select **Disable resolved message** if you do not want to be notified when an alert resolves.
9. To add another contact point integration, click **Add contact point integration** and repeat steps 6 through 8.
10. Save your changes.

Edit a contact point

Complete the following steps to edit a contact point.

1. In the left-side menu, click **Alerts & IRM** and then **Alerting**.
2. Click **Contact points** to view a list of existing contact points.
3. On the **Contact Points** tab, find the contact point you want to edit, and then click **Edit**.
4. Update the contact point and save your changes.

Delete a contact point

Complete the following steps to delete a contact point.

1. In the left-side menu, click **Alerts & IRM** and then **Alerting**.
2. Click **Contact points** to view a list of existing contact points.
3. On the **Contact Points** tab, find the contact point you want to delete, and then click **More** → **Delete**.
4. In the confirmation dialog, click **Yes, delete**.

> **NOTE**
>
> You cannot delete contact points that are in use by a notification policy. Either delete the notification policy or update it to use another contact point.

Test a contact point

Complete the following steps to test a contact point.

1. In the left-side menu, click **Alerts & IRM** and then **Alerting**.
2. Click **Contact points** to view a list of existing contact points.
3. On the **Contact Points** tab, find the contact point you want to test, then click **Edit**. You can also create a new contact point if needed.
4. Click **Test** to open the contact point testing modal.
5. Choose whether to send a predefined test notification or choose custom to add your own custom annotations and labels to include in the notification.
6. Click **Send test notification** to fire the alert.

Configure contact point integrations

Configure contact point integrations in Grafana to select your preferred communication channel for receiving notifications when your alert rules are firing. Each integration has its own configuration options and setup process. In most cases, this involves providing an API key or a Webhook URL.

Once configured, you can use integrations as part of your contact points to receive notifications whenever your alert changes its state. In this section, we'll cover the basic steps to configure your integrations, so you can start receiving real-time alerts and stay on top of your monitoring data.

List of supported integrations

Name	Type
DingDing	`dingding`
Discord	`discord`
Email	`email`
Google Chat	`googlechat`
Hipchat	`hipchat`
Kafka	`kafka`
Line	`line`
Microsoft Teams	`teams`
Opsgenie	`opsgenie`
Pagerduty	`pagerduty`
Prometheus Alertmanager	`prometheus-alertmanager`

Name	Type
Pushover	`pushover`
Sensu	`sensu`
Sensu Go	`sensugo`
Slack	`slack`
Telegram	`telegram`
Threema	`threema`
VictorOps	`victorops`
Webhook	`webhook`

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Configure the webhook notifier for Alerting

Example JSON body:

json Copy

```
{
  "receiver": "My Super Webhook",
  "status": "firing",
  "orgId": 1,
  "alerts": [
    {
      "status": "firing",
      "labels": {
        "alertname": "High memory usage",
        "team": "blue",
        "zone": "us-1"
      },
      "annotations": {
        "description": "The system has high memory usage",
        "runbook_url": "https://myrunbook.com/runbook/1234",
        "summary": "This alert was triggered for zone us-1"
      },
      "startsAt": "2021-10-12T09:51:03.157076+02:00",
      "endsAt": "0001-01-01T00:00:00Z",
```

Expand code

Webhook fields

Body

Key	Type	Description
receiver	string	Name of the webhook
status	string	Current status of the alert, `firing` or `resolved`
orgId	number	ID of the organization related to the payload
alerts	array of alerts	Alerts that are triggering
groupLabels	object	Labels that are used for grouping, map of string keys to string values
commonLabels	object	Labels that all alarms have in common, map of string keys to string values
commonAnnotations	object	Annotations that all alarms have in common, map of string keys to string values
externalURL	string	External URL to the Grafana instance sending this webhook
version	string	Version of the payload
groupKey	string	Key that is used for grouping
truncatedAlerts	number	Number of alerts that were truncated
title	string	**Will be deprecated soon**
state	string	**Will be deprecated soon**
message	string	**Will be deprecated soon**

Alert

Key	Type	Description
status	string	Current status of the alert, `firing` or `resolved`
labels	object	Labels that are part of this alert, map of string keys to string values
annotations	object	Annotations that are part of this alert, map of string keys to string values
startsAt	string	Start time of the alert
endsAt	string	End time of the alert, default value when not resolved is `0001-01-01T00:00:00Z`
values	object	Values that triggered the current status
generatorURL	string	URL of the alert rule in the Grafana UI

Key	Type	Description
fingerprint	string	The labels fingerprint, alarms with the same labels will have the same fingerprint
silenceURL	string	URL to silence the alert rule in the Grafana UI
dashboardURL	string	**Will be deprecated soon**
panelURL	string	**Will be deprecated soon**
imageURL	string	URL of a screenshot of a panel assigned to the rule that created this notification

Removed fields related to dashboards

Alerts are not coupled to dashboards anymore therefore the fields related to dashboards `dashboardId` and `panelId` have been removed.

WeCom

WeCom contact points need a Webhook URL. These are obtained by setting up a WeCom robot on the corresponding group chat. To obtain a Webhook URL using the WeCom desktop Client please follow these steps:

1. Click the "..." in the top right corner of a group chat that you want your alerts to be delivered to
2. Click "Add Group Robot", select "New Robot" and give your robot a name. Click "Add Robot"
3. There should be a Webhook URL in the panel.

Setting	Description
Url	WeCom webhook URL.

Configure Grafana OnCall for Alerting

Use the Grafana Alerting - Grafana OnCall integration to effortlessly connect alerts generated by Grafana Alerting with Grafana OnCall, where you can then route them according to defined escalation chains and schedules.

You can set up the integration using the Grafana Alerting application or the Grafana OnCall application. For more information on setting it up from the Grafana OnCall application, see Grafana OnCall documentation.

Before you begin

- Ensure you have Installed and enabled the Grafana OnCall plugin
- Ensure your version of Grafana is up-to-date and supports the new features

Procedure

To set up the Grafana OnCall integration using the Grafana Alerting application, complete the following steps.

1. Navigate to **Alerts&IRM** → **Alerting** → **Contact points**.
2. Click **+ Add contact point**.
3. Enter a contact point name.
4. From the Integration list, select Grafana OnCall.

   Note:

   The Grafana OnCall integration is only available for Grafana Alertmanager.
5. Choose whether to add a new OnCall integration or add an existing one.
   - If you add a new one, enter an Integration name.

- If you add an existing one, choose from the list of available integrations

6. Click **Save contact point**.

7. On the Contact points list view page, you can see the contact point with the Grafana OnCall icon.

   If the integration is not yet being used anywhere in the notification policies tree, it will have **Unused** as its status in the **Health** column. It won't receive any notifications, because there are no notifications using that integration.

8. Connect your contact point to a notification policy.

   For more information on connecting your contact point to a notification policy, see Create notification policy.

9. To view your integration in the Grafana OnCall application and set up routes and escalation chains, click the Link next to the integration on the Contact points list view page in the **Type** column.

   This redirects you to the Grafana OnCall integration page in the Grafana OnCall application. From there, you can add routes and escalation chains.

Configure Slack for Alerting

Use the Grafana Alerting - Slack integration to send Slack notifications when your alerts are firing.

There are two ways of integrating Slack into Grafana Alerting.

1. Use a Slack API token

   Enable your app to access the Slack API. If, for example, you are interested in more granular control over permissions, or your project is expected to regularly scale, resulting in new channels being created, this is the best option.

2. Use a Webhook URL

   Webhooks is the simpler way to post messages into Slack. Slack automatically creates a bot user with all the necessary permissions to post messages to one particular channel of your choice.

> **NOTE**
>
> Grafana Alerting only allows one Slack channel per contact point.

Before you begin

Slack API Token

If you are using a Slack API Token, complete the following steps.

1. Follow steps 1 and 2 of the Slack API Quickstart.
2. Add the chat:write.public scope to give your app the ability to post in all public channels without joining.
3. In OAuth Tokens for Your Workspace, copy the Bot User OAuth Token.

4. Open your Slack workplace.
5. Right click the channel you want to receive notifications in.
6. Click View channel details.
7. Scroll down and copy the Channel ID.

> **NOTE**
>
> While going through these steps, Slack may prompt you to Reinstall your app in order for the changes to take effect.

Webhook URL

If you are using a Webhook URL, follow steps 1 and 5 in the Slack API Quickstart.

> **NOTE**
>
> Make sure you copy the Slack app Webhook URL. You will need this when setting up your contact point integration in Grafana Alerting.

Procedure

To create your Slack integration in Grafana Alerting, complete the following steps.

1. Navigate to Alerts&IRM → Alerting → Contact points.
2. Click **+ Add contact point**.
3. Enter a contact point name.
4. From the Integration list, select Slack.
5. If you are using a Slack API token:
   - In the **Recipient** field, copy in the channel ID.
   - In the **Token** field, copy in the Bot User OAuth Token that starts with “xoxb-”.
6. If you are using a Webhook URL, in the **Webhook** field, copy in your Slack app Webhook URL.
7. Click **Test** to check that your integration works.
8. Click **Save contact point**.

Next steps

To add the contact point and integration you created to your default notification policy, complete the following steps.

1. Navigate to **Alerts&IRM** → **Alerting** → **Notification policies**.
2. In the **Default policy**, click the ellipsis icon (...) and then **Edit**,
3. Change the default policy to the contact point you created.
4. Click **Update default policy**.

Note: If you have more than one contact point, add a new notification policy rather than edit the default one, so you can route specific alerts to Slack. For more information, refer to Notification policies.

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Configure PagerDuty for Alerting

To set up PagerDuty, provide an integration key.

Setting	Description
Integration Key	Integration key for PagerDuty
Severity	Level for dynamic notifications, default is `critical`
Custom Details	Additional details about the event

The `CustomDetails` field is an object containing arbitrary key-value pairs. The user-defined details are merged with the ones we use by default.

Our default values for `CustomDetails` are:

Go Copy

```
{
    "firing":       `{{ template "__text_alert_list" .Alerts.Firing }}`,
    "resolved":     `{{ template "__text_alert_list" .Alerts.Resolved }}`,
    "num_firing":   `{{ .Alerts.Firing | len }}`,
    "num_resolved": `{{ .Alerts.Resolved | len }}`,
}
```

In case of duplicate keys, the user-defined details overwrite the default ones.

Configure notification policies

Notification policies determine how alerts are routed to contact points.

Policies have a tree structure, where each policy can have one or more nested policies. Each policy, except for the default policy, can also match specific alert labels.

Each alert is evaluated by the default policy and subsequently by each nested policy.

If the **Continue matching subsequent sibling nodes** option is enabled for a nested policy, then evaluation continues even after one or more matches. A parent policy's configuration settings and contact point information govern the behavior of an alert that does not match any of the nested policies. A default policy governs any alert that does not match a nested policy.

You can configure Grafana-managed notification policies as well as notification policies for an external Alertmanager data source.

For more information on notification policies, see fundamentals of Notification Policies.

Edit default notification policy

> **NOTE**
>
> Before Grafana v8.2, the configuration of the embedded Alertmanager was shared across organizations. Users of Grafana 8.0 and 8.1 are advised to use the new Grafana 8 Alerts only if they have one organization. Otherwise, silences for the Grafana managed alerts will be visible by all organizations.

1. In the left-side menu, click **Alerts & IRM** and then **Alerting**.
2. Click **Notification policies**.
3. From the **Choose Alertmanager** dropdown, select an external Alertmanager. By default, the **Grafana Alertmanager** is selected.

4. In the Default policy section, click **...** → **Edit**.
5. In **Default contact point**, update the contact point to whom notifications should be sent for rules when alert rules do not match any specific policy.
6. In **Group by**, choose labels to group alerts by. If multiple alerts are matched for this policy, then they are grouped by these labels. A notification is sent per group. If the field is empty (default), then all notifications are sent in a single group. Use a special label `...` to group alerts by all labels (which effectively disables grouping).
7. In **Timing options**, select from the following options:
   - **Group wait** Time to wait to buffer alerts of the same group before sending an initial notification. Default is 30 seconds.
   - **Group interval** Minimum time interval between two notifications for a group. Default is 5 minutes.
   - **Repeat interval** Minimum time interval for re-sending a notification if no new alerts were added to the group. Default is 4 hours.
8. Click **Save** to save your changes.

Add new nested policy

To create a new notification policy, you need to follow its tree structure. New policies created on the trunk of the tree (default policy), are the tree branches. And, subsequently, each branch can bear their own child policies. This is why you will always be adding a new **nested** policy under either the default policy, or under a already nested policy.

1. In the left-side menu, click **Alerts & IRM** and then **Alerting**.
2. Click **Notification policies**.
3. From the **Choose Alertmanager** dropdown, select an Alertmanager. By default, the **Grafana Alertmanager** is selected.
4. To add a top level specific policy, go to the Specific routing section (either to the default policy, or to another existing policy in which you would like to add a new nested policy) and click **+New nested policy**.
5. In the Matching labels section, add one or more rules for matching alert labels.
6. In the **Contact point** dropdown, select the contact point to send notification to if alert matches only this specific policy and not any of the nested policies.
7. Optionally, enable **Continue matching subsequent sibling nodes** to continue matching sibling policies even after the alert matched the current policy. When this option is enabled, you can get more than one notification for one alert.
8. Optionally, enable **Override grouping** to specify the same grouping as the default policy. If this option is not enabled, the default policy grouping is used.
9. Optionally, enable **Override general timings** to override the timing options configured in the group notification policy.

10. Click **Save policy** to save your changes.

Add nested policy

1. In the left-side menu, click **Alerts & IRM** and then **Alerting**.
2. Click **Notification policies**.
3. Expand the specific policy you want to update.
4. Click **+ Add nested policy**, then add the details using information in Add new specific policy.
5. Click **Save policy** to save your changes.

Edit notification policies

1. In the left-side menu, click **Alerts & IRM**, and then **Alerting**.
2. Click **Notification policies**.
3. Find the policy you want to edit, then click **...** → **Edit**.
4. Make any changes using instructions in Add new specific policy.
5. Save your changes.

Searching for policies

Grafana allows you to search within the tree of policies by the following:

- **Label matchers**
- **Contact Points**

To search by contact point simply select a contact point from the **Search by contact point** dropdown. The policies that use that contact point will be highlighted in the user interface.

To search by label matchers simply enter a valid matcher in the **Search by matchers** input field. Multiple matchers can be combined with a comma (`,`).

An example of a valid matchers search input is:

```
severity=high, region=~EMEA|NASA
```

> All matched policies will be **exact** matches, we currently do not support regex-style or partial matching.

Example

An example of an alert configuration.

- Create a “default” contact point for slack notifications, and set it on the default policy.
- Edit the default policy grouping to group alerts by `cluster`, `namespace` and `severity` so that you get a notification per alert rule and specific kubernetes cluster and namespace.
- Create specific route for alerts coming from the development cluster with an appropriate contact point.
- Create a specific route for alerts with “critical” severity with a more invasive contact point integration, like pager duty notification.
- Create specific routes for particular teams that handle their own on-call rotations.

Manage your alerts

Once you have set up your alert rules, contact points, and notification policies, you can use Grafana Alerting to:

Create silences

Create mute timings

Declare incidents from firing alerts

View the state and health of alert rules

View and filter alert rules

Customize notifications

Customize your notifications with notifications templates.

You can use notification templates to change the title, message, and format of the message in your notifications.

Notification templates are not tied to specific contact point integrations, such as email or Slack. However, you can choose to create separate notification templates for different contact point integrations.

You can use notification templates to:

- Customize the subject of an email or the title of a message.
- Add, change or remove text in notifications. For example, to select or omit certain labels, annotations and links.
- Format text in bold and italic, and add or remove line breaks.

You cannot use notification templates to:

- Add HTML and CSS to email notifications to change their visual appearance.
- Change the design of notifications in instant messaging services such as Slack and Microsoft Teams. For example, to add or remove custom blocks with Slack Block Kit or adaptive cards with Microsoft Teams.
- Choose the number and size of images, or where in the notification images are shown.
- Customize the data in webhooks, including the fields or structure of the JSON data or send the data in other formats such as XML.
- Add or remove HTTP headers in webhooks other than those in the contact point configuration.

Using Go's templating language

Learn how to write the content of your notification templates in Go's templating language.

Create reusable notification templates for your contact points.

Use notification templates

Use notification templates to send notifications to your contact points.

Reference

Data that is available when writing templates.

Using Go's templating language

You write notification templates in Go's templating language, text/template.

Before you start creating your own notification templates, we recommend that you read through this topic, which provides you with an overview of Go's templating language and writing templates in text/template.

Dot

In text/template there is a special cursor called dot, and is written as `.`. You can think of this cursor as a variable whose value changes depending where in the template it is used. For example, at the start of a notification template `.` refers to something called `ExtendedData` which contains a number of fields including `Alerts`, `Status`, `GroupLabels`, `CommonLabels`, `CommonAnnotations` and `ExternalURL`. However, dot might refer to something else when used in a range over a list, when used inside a `with`, or when writing feature templates to be used in other templates. You can see examples of this in Create notification templates, and all data and functions in the Reference.

Opening and closing tags

In text/template, templates start with `{{` and end with `}}` irrespective of whether the template prints a variable or executes control structures such as if statements. This is different from other templating languages such as Jinja where printing a variable uses `{{` and `}}` and control structures use `{%` and `%}`.

Print

To print the value of something use `{{` and `}}`. You can print the value of dot, a field of dot, the result of a function, and the value of a variable. For example, to print the `Alerts` field where dot refers to `ExtendedData` you would write the following:

```
{{ .Alerts }}
```

Copy

Iterate over alerts

To print just the labels of each alert, rather than all information about the alert, you can use a `range` to iterate the alerts in `ExtendedData`:

```
{{ range .Alerts }}
{{ .Labels }}
{{ end }}
```

Copy

Inside the range dot no longer refers to `ExtendedData`, but to an `Alert`. You can use `{{ .Labels }}` to print the labels of each alert. This works because `{{ range .Alerts }}` changes dot to refer to the current alert in the list of alerts. When the range is finished dot is reset to the value it had before the start of the range, which in this example is `ExtendedData`:

```
{{ range .Alerts }}
{{ .Labels }}
{{ end }}
{{/* does not work, .Labels does not exist here */}}
{{ .Labels }}
{{/* works, cursor was reset */}}
{{ .Status }}
```

Copy

Iterate over annotations and labels

Let's write a template to print the labels of each alert in the format `The name of the label is $name, and the value is $value`, where `$name` and `$value` contain the name and value of each label.

Like in the previous example, use a range to iterate over the alerts in `.Alerts` such that dot refers to the current alert in the list of alerts, and then use a second range on the sorted labels so dot is updated a second time to refer to the current label. Inside the second range use `.Name` and `.Value` to print the name and value of each label:

```
{{ range .Alerts }}
{{ range .Labels.SortedPairs }}
The name of the label is {{ .Name }}, and the value is {{ .Value }}
{{ end }}
{{ range .Annotations.SortedPairs }}
The name of the annotation is {{ .Name }}, and the value is {{ .Value }}
{{ end }}
{{ end }}
```

Copy

The index function

To print a specific annotation or label use the `index` function.

```
{{ range .Alerts }}
The name of the alert is {{ index .Labels "alertname" }}
{{ end }}
```

Copy

If statements

You can use if statements in templates. For example, to print `There are no alerts` if there are no alerts in `.Alerts` you would write the following:

```
{{ if .Alerts }}
There are alerts
{{ else }}
There are no alerts
{{ end }}
```

Copy

With

With is similar to if statements, however unlike if statements, `with` updates dot to refer to the value of the with:

```
{{ with .Alerts }}
There are {{ len . }} alert(s)
{{ else }}
There are no alerts
{{ end }}
```

Copy

Variables

Variables in text/template must be created within the template. For example, to create a variable called `$variable` with the current value of dot you would write the following:

```
{{ $variable := . }}
```

Copy

You can use `$variable` inside a range or `with` and it will refer to the value of dot at the time the variable was defined, not the current value of dot.

For example, you cannot write a template that use `{{ .Labels }}` in the second range because here dot refers to the current label, not the current alert:

```
{{ range .Alerts }}
{{ range .Labels.SortedPairs }}
{{ .Name }} = {{ .Value }}
{{/* does not work because in the second range . is a label not an alert */}}
There are {{ len .Labels }}
{{ end }}
{{ end }}
```

Copy

You can fix this by defining a variable called `$alert` in the first range and before the second range:

```
{{ range .Alerts }}
{{ $alert := . }}
{{ range .Labels.SortedPairs }}
{{ .Name }} = {{ .Value }}
{{/* works because $alert refers to the value of dot inside the first range */}}
There are {{ len $alert.Labels }}
{{ end }}
{{ end }}
```

Copy

Range with index

You can get the index of each alert within a range by defining index and value variables at the start of the range:

```
{{ $num_alerts := len .Alerts }}
{{ range $index, $alert := .Alerts }}
This is alert {{ $index }} out of {{ $num_alerts }}
{{ end }}
```

Copy

Define templates

You can define templates using `define` and the name of the template in double quotes. You should not define templates with the same name as other templates, including default templates such as `__subject`, `__text_values_list`, `__text_alert_list`, `default.title` and `default.message`. Where a template has been created with the same name as a default template, or a template in another notification template, Grafana might use either template. Grafana does not prevent, or show an error message, when there are two or more templates with the same name.

```
{{ define "print_labels" }}
```

Copy

```
{{ end }}
```

Execute templates

You can execute defined templates using `template`, the name of the template in double quotes, and the cursor that should be passed to the template:

```
{{ template "print_labels" . }}
```
Copy

Pass data to templates

Within a template dot refers to the value that is passed to the template.

For example, if a template is passed a list of firing alerts then dot refers to that list of firing alerts:

```
{{ template "print_alerts" .Alerts }}
```
Copy

If the template is passed the sorted labels for an alert then dot refers to the list of sorted labels:

```
{{ template "print_labels" .SortedLabels }}
```
Copy

This is useful when writing reusable templates. For example, to print all alerts you might write the following:

```
{{ template "print_alerts" .Alerts }}
```
Copy

Then to print just the firing alerts you could write this:

```
{{ template "print_alerts" .Alerts.Firing }}
```
Copy

This works because both `.Alerts` and `.Alerts.Firing` is a list of alerts.

```
{{ define "print_alerts" }}
{{ range . }}
{{ template "print_labels" .SortedLabels }}
{{ end }}
{{ end }}
```
Copy

Comments

You can add comments with `{{/*` and `*/}}`:

```
{{/* This is a comment */}}
```

Copy

To prevent comments from adding line breaks use:

```
{{- /* This is a comment with no leading or trailing line breaks */ -}}
```

Copy

Indentation

You can use indentation, both tabs and spaces, and line breaks, to make templates more readable:

```
{{ range .Alerts }}
  {{ range .Labels.SortedPairs }}
    {{ .Name }} = {{ .Value }}
  {{ end }}
{{ end }}
```

Copy

However, indentation in the template will also be present in the text. Next we will see how to remove it.

Remove spaces and line breaks

In text/template use `{{-` and `-}}` to remove leading and trailing spaces and line breaks.

For example, when using indentation and line breaks to make a template more readable:

```
{{ range .Alerts }}
  {{ range .Labels.SortedPairs }}
    {{ .Name }} = {{ .Value }}
  {{ end }}
{{ end }}
```

Copy

The indentation and line breaks will also be present in the text:

```
    alertname = "Test"

    grafana_folder = "Test alerts"
```

Copy

You can remove the indentation and line breaks from the text changing `}}` to `-}}` at the start of each range:

```
{{ range .Alerts -}}
  {{ range .Labels.SortedPairs -}}
    {{ .Name }} = {{ .Value }}
```

Copy

```
  {{ end }}
{{ end }}
```

The indentation and line breaks in the template are now absent from the text:

```
alertname = "Test"
grafana_folder = "Test alerts"
```

Copy

Create notification templates

Create reusable notification templates to send to your contact points.

You can add one or more templates to your notification template.

Your notification template name must be unique. You cannot have two templates with the same name in the same notification template or in different notification templates. Avoid defining templates with the same name as default templates, such as: `__subject`, `__text_values_list`, `__text_alert_list`, `default.title` and `default.message`.

To create a notification template, complete the following steps.

1. Click **Alerts & IRM** → **Contact points**.
2. Click the **Notification Templates** tab and then **+ Add notification template**.
3. Enter a name for the notification template.
4. Write the content of the template in the content field.
5. Save your changes.

`{{ define "email.subject" }}` and `{{ end }}` is automatically added to the start and end of the content:

To create a notification template that contains more than one template:

1. Click **+ Add notification template**.
2. Enter a name for the notification template.
3. Write each template in the Content field, including `{{ define "name-of-template" }}` and `{{ end }}` at the start and end of each template.
4. Save your changes.

Preview notification templates

Preview how your notification templates will look before using them in your contact points, helping you understand the result of the template you are creating as well as enabling you to fix any errors before saving it.

Note: This feature is only for Grafana Alertmanager.

To preview your notification templates:

1. Navigate to **Alerts&IRM** → **Alerting** → **Contact points** → **Notification Templates**.
2. Click **+ Add notification template** or edit an existing template.
3. Add or update your template content.

   Default data is provided and you can add or edit alert data to it as well as alert instances. You can add alert data directly in the Payload data window itself or click **Select alert instances** or **Add custom alerts**.
4. Optional: To add alert data from existing alert instances:

   a. Click **Select alert instances**.

   b. Hover over the alert instances to view more information on each alert instance.

   c. Click **Confirm** to add the alert instance(s) to the payload.
5. Optional: To add alert data using the Alert data editor, click **Add custom data:**

   a. Add annotations, custom labels and/or set a dashboard or a panel.

   b. Toggle Firing/resolved depending on whether you want to add firing or resolved alerts to your notification.

   c. Click **Add alert data**.

   d. Click **Refresh preview** to see what your template content will look like and the corresponding payload data.

   If there are any errors in your template, they are displayed in the Preview and you can correct them before saving.
6. Save your changes.

Template the subject of an email

Template the subject of an email to contain the number of firing and resolved alerts:

```
1 firing alert(s), 0 resolved alerts(s)
```

Copy

1. Create a template called `email.subject` with the following content:

```
{{ define "email.subject" }}
{{ len .Alerts.Firing }} firing alert(s), {{ len .Alerts.Resolved }} resolved
{{ end }}
```

Copy

2 Execute the template from the subject field in your contact point integration:

```
{{ template "email.subject" . }}
```

Copy

Template the message of an email

Template the message of an email to contain a summary of all firing and resolved alerts:

```
There are 2 firing alert(s), and 1 resolved alert(s)

Firing alerts:

- alertname=Test 1 grafana_folder=GrafanaCloud has value(s) B=1
- alertname=Test 2 grafana_folder=GrafanaCloud has value(s) B=2

Resolved alerts:

- alertname=Test 3 grafana_folder=GrafanaCloud has value(s) B=0
```

Copy

1 Create a notification template called `email` with two templates in the content: `email.message_alert` and `email.message`.

The `email.message_alert` template is used to print the labels and values for each firing and resolved alert while the `email.message` template contains the structure of the email.

```
{{- define "email.message_alert" -}}
{{- range .Labels.SortedPairs }}{{ .Name }}={{ .Value }} {{ end }} has value(s
{{- range $k, $v := .Values }} {{ $k }}={{ $v }}{{ end }}
{{- end -}}

{{ define "email.message" }}
There are {{ len .Alerts.Firing }} firing alert(s), and {{ len .Alerts.Resolve

{{ if .Alerts.Firing -}}
Firing alerts:
{{- range .Alerts.Firing }}
- {{ template "email.message_alert" . }}
{{- end }}
{{- end }}
```

Copy

```
{{ if .Alerts.Resolved -}}
Resolved alerts:
{{- range .Alerts.Resolved }}
- {{ template "email.message_alert" . }}
```

Expand code

2 Execute the template from the message field in your contact point integration:

```
{{ template "email.message" . }}
```

Copy

Template the title of a Slack message

Template the title of a Slack message to contain the number of firing and resolved alerts:

```
1 firing alert(s), 0 resolved alerts(s)
```

Copy

1 Create a template called `slack.title` with the following content:

```
{{ define "slack.title" }}
{{ len .Alerts.Firing }} firing alert(s), {{ len .Alerts.Resolved }} resolved
{{ end }}
```

Copy

2 Execute the template from the title field in your contact point integration:

```
{{ template "slack.title" . }}
```

Copy

Template the content of a Slack message

Template the content of a Slack message to contain a description of all firing and resolved alerts, including their labels, annotations, Silence URL and Dashboard URL.

Note:

This template is for Grafana-managed alerts only. To use the template for Grafana Mimir/Loki-managed alerts, delete the references to DashboardURL and SilenceURL. For more information, see the Prometheus documentation on notifications.

```
1 firing alert(s):

[firing] Test1
Labels:
- alertname: Test1
- grafana_folder: GrafanaCloud
Annotations:
```

Copy

```
- description: This is a test alert
Silence: https://example.com/alerting/silence/new?alertmanager=grafana&matcher=alert
Go to dashboard: https://example.com/d/dlhdLqF4z?orgId=1

1 resolved alert(s):

[firing] Test2
Labels:
- alertname: Test2
- grafana_folder: GrafanaCloud
Annotations:
- description: This is another test alert
Silence: https://example.com/alerting/silence/new?alertmanager=grafana&matcher=alert
```

Expand code

1. Create a template called `slack` with two templates in the content: `slack.print_alert` and `slack.message`.

   The `slack.print_alert` template is used to print the labels, annotations, SilenceURL and DashboardURL while the `slack.message` template contains the structure of the notification.

```
{{ define "slack.print_alert" -}}
[{{.Status}}] {{ .Labels.alertname }}
Labels:
{{ range .Labels.SortedPairs -}}
- {{ .Name }}: {{ .Value }}
{{ end -}}
{{ if .Annotations -}}
Annotations:
{{ range .Annotations.SortedPairs -}}
- {{ .Name }}: {{ .Value }}
{{ end -}}
{{ end -}}
{{ if .SilenceURL -}}
  Silence: {{ .SilenceURL }}
{{ end -}}
{{ if .DashboardURL -}}
  Go to dashboard: {{ .DashboardURL }}
{{- end }}
{{- end }}
```

Copy

Expand code

2. Execute the template from the text body field in your contact point integration:

```
{{ template "slack.message" . }}
```

Copy

Template both email and Slack with shared templates

Instead of creating separate notification templates for email and Slack, you can share the same template.

For example, if you want to send an email with this subject and Slack message with this title:

```
1 firing alert(s), 0 resolved alerts(s)
```

Copy

1. Create a template called `common.subject_title` with the following content:

```
{{ define "common.subject_title" }}
{{ len .Alerts.Firing }} firing alert(s), {{ len .Alerts.Resolved }} resolved
{{ end }}
```

Copy

2. For email, execute the template from the subject field in your email contact point integration:

```
{{ template "common.subject_title" . }}
```

Copy

3. For Slack, execute the template from the title field in your Slack contact point integration:

```
{{ template "common.subject_title" . }}
```

Copy

Use notification templates

Use templates in contact points to customize your notifications.

In the Contact points tab, you can see a list of your contact points.

1. To create a new contact point, click New.

   Note: You can edit an existing contact by clicking the Edit icon.

2. Execute a template from one or more fields such as Message and Subject:

   Use notification template

   For more information on how to write and execute templates, refer to Using Go's templating language and Create notification templates.

3. Click **Save contact point**.

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Reference

Data

Alert

Name	Kind	Description	Example
Status	`string`	Firing or resolved	`{{ .Status }}`
Labels	`KV`	The labels for this alert	`{{ .Labels }}`
Annotations	`KV`	The annotations for this alert	`{{ .Annotations }}`
Values	`KV`	The values of all expressions, including Classic Conditions	`{{ .Values }}`
StartsAt	`Time`	The time the alert fired	`{{ .StartsAt }}`
EndsAt	`Time`		`{{ .EndsAt }}`
GeneratorURL	`string`	A link to Grafana, or the Alertmanager if using an external Alertmanager	`{{ .GeneratorURL }}`
SilenceURL	`string`	A link to silence the alert	`{{ .SilenceURL }}`
DashboardURL	`string`	A link to the Grafana Dashboard if the alert has a Dashboard UID annotation	`{{ .DashboardURL }}`
PanelURL	`string`	A link to the panel if the alert has a Panel ID annotation	`{{ .PanelURL }}`
Fingerprint	`string`	A unique string that identifies the alert	`{{ .Fingerprint }}`

Name	Kind	Description	Example
ValueString	`string`	A string that contains the labels and value of each reduced expression in the alert.	`{{ .ValueString }}`

ExtendedData

Name	Kind	Description	Example
Receiver	`string`	The name of the contact point sending the notification	`{{ .Receiver }}`
Status	`string`	The status is `firing` if at least one alert is firing, otherwise `resolved`	`{{ .Status }}`
Alerts	`[]Alert`	List of all firing and resolved alerts in this notification	`There are {{ len .Alerts }} alerts`
Firing alerts	`[]Alert`	List of all firing alerts in this notification	`There are {{ len .Alerts.Firing }} firing alerts`
Resolved alerts	`[]Alert`	List of all resolved alerts in this notification	`There are {{ len .Alerts.Resolved }} resolved alerts`
GroupLabels	`KV`	The labels that group these alerts in this	`{{ .GroupLabels }}`
CommonLabels	`KV`	The labels common to all alerts in this notification	`{{ .CommonLabels }}`
CommonAnnotations	`KV`	The annotations common to all alerts in this notification	`{{ .CommonAnnotations }}`
ExternalURL	`string`	A link to Grafana, or the Alertmanager that sent this notification if using an external Alertmanager	`{{ .ExternalURL }}`

KV

`KV` is a set of key value pairs, where each key and value is a string. If a KV happens to contain numbers or bools then these are string representations of the numeric or boolean value.

Here is an example of a KV, the annotations of an alert:

yaml Copy

```yaml
summary: 'A summary of the alert'
description: 'A description of the alert'
```

In addition to iterating over each key value pair, you can sort the pairs, remove keys, and iterate over just the keys or the values.

Name	Description	Arguments	Returns	Example
SortedPairs	Sorts			`{{ .Annotations.SortedPairs }}`
Remove	Returns a copy of the KV with the keys removed	[]string		`{{ .Annotations.Remove "summary" }}`
Names	A list of the names			`{{ .Names }}`
Values	A list of the values			`{{ .Values }}`

Time

Time is from the Go `time` package. You can print a time in a number of different formats. For example, to print the time that an alert fired in the format `Monday, 1st January 2022 at 10:00AM` you would write the following template:

```
{{ .StartsAt.Format "Monday, 2 January 2006 at 3:04PM" }}
```

Copy

You can find a reference for Go's time format here.

Use images in notifications

Images in notifications helps recipients of alert notifications better understand why an alert has fired or resolved by including a screenshot of the panel associated with the alert.

NOTE

This feature is not supported in Mimir or Loki, or when Grafana is configured to send alerts to other Alertmanagers such as the Prometheus Alertmanager

When an alert is fired or resolved Grafana takes a screenshot of the panel associated with the alert. This is determined via the Dashboard UID and Panel ID annotations of the rule. Grafana cannot take a screenshot for alerts that are not associated with a panel.

Grafana takes at most two screenshots for each alert: once when the alert fires and again when the alert is resolved. Screenshots are not re-taken over the lifetime of the alert, instead you should open the panel in Grafana to follow the data in real time. In addition, depending on how alerts are grouped in your notification policies, Grafana might send a notification with many screenshots of the same panel. This happens because Grafana does not know how your alerts are grouped at the time a screenshot is taken, and so acts conservatively by taking a screenshot for every alert.

Once a screenshot has been taken Grafana can either upload it to a cloud storage service such as Amazon S3, Azure Blob Storage or Google Cloud Storage; upload the screenshot to it's internal web server; or upload it to the service that is receiving the notification, such as Slack. Which option you should choose depends on how your Grafana is managed and which integrations you use. More information on this can be found in Requirements.

Refer to the table at the end of this page for a list of contact points and their support for images in notifications.

Requirements

1. To use images in notifications, Grafana must be set up to use image rendering. You can either install the image rendering plugin or run it as a remote rendering service.
2. When a screenshot is taken it is saved to the data folder, even if Grafana is configured to upload screenshots to a cloud storage service. Grafana must have write-access to this folder otherwise screenshots cannot be saved to disk and an error will be logged for each failed screenshot attempt.
3. You should use a cloud storage service unless sending alerts to Discord, Email, Pushover, Slack or Telegram. These integrations support either embedding screenshots in the email or attaching screenshots to the notification, while other integrations must link screenshots uploaded to a cloud storage bucket. If a cloud storage service has been configured then integrations that support both will link screenshots from the cloud storage bucket instead of embedding or attaching screenshots to the notification.
4. If uploading screenshots to a cloud storage service such as Amazon S3, Azure Blob Storage or Google Cloud Storage; and accessing screenshots in the bucket requires authentication, logging into a VPN or corporate network; then image previews might not work in all instant messaging and communication platforms as some services rewrite URLs to use their CDN. If this happens we recommend using integrations which support uploading images or disabling images in notifications altogether.
5. When uploading screenshots to a cloud storage service Grafana uses a random 20 character (30 characters for Azure Blob Storage) filename for each image. This makes URLs hard to guess but not impossible.
6. Grafana does not delete screenshots from cloud storage. We recommend configuring a retention policy with your cloud storage service to delete screenshots older than 1 month.
7. If Grafana is configured to upload screenshots to its internal web server, and accessing Grafana requires logging into a VPN or corporate network; image previews might not work in all instant messaging and communication platforms as some services rewrite URLs to use their CDN. If this happens we recommend using integrations which support uploading images or disabling images in notifications altogether.
8. Grafana does not delete screenshots uploaded to its internal web server. To delete screenshots from `static_root_path/images/attachments` after a certain amount of time we recommend setting up a CRON job.

Configuration

> **NOTE**
>
> Grafana Cloud users can request this feature by opening a support ticket in the Cloud Portal.

Having installed either the image rendering plugin, or set up Grafana to use a remote rendering service, set `capture` in `[unified_alerting.screenshots]` to `true`:

```
# Enable screenshots in notifications. You must have either installed the Grafana image rendering
# plugin, or set up Grafana to use a remote rendering service.
# For more information on configuration options, refer to [rendering].
capture = false
```

If screenshots should be uploaded to cloud storage then `upload_external_image_storage` should also be set to `true`:

```
# Uploads screenshots to the local Grafana server or remote storage such as Azure, S3 and GCS. Ple
# see [external_image_storage] for further configuration options. If this option is false, screens
# will be persisted to disk for up to temp_data_lifetime.
upload_external_image_storage = false
```

For more information on image rendering, refer to image rendering.

Restart Grafana for the changes to take effect.

Advanced configuration

We recommended that `max_concurrent_screenshots` is less than or equal to `concurrent_render_request_limit`. The default value for both `max_concurrent_screenshots` and `concurrent_render_request_limit` is `5`:

```
# The maximum number of screenshots that can be taken at the same time. This option is different f
# concurrent_render_request_limit as max_concurrent_screenshots sets the number of concurrent scre
# that can be taken at the same time for all firing alerts where as concurrent_render_request_limi
# the total number of concurrent screenshots across all Grafana services.
max_concurrent_screenshots = 5
```

Supported contact points

Grafana supports a wide range of contact points with varied support for images in notifications. The table below shows the list of all contact points supported in Grafana and their support for uploading screenshots to the receiving service and referencing screenshots that have been uploaded to a cloud storage service.

Name	Upload from disk	Reference from cloud storage
DingDing	No	No
Discord	Yes (Maximum of 10 per notification)	Yes (Maximum of 10 per notification)
Email	Yes (Embedded in the email)	Yes

Name	Upload from disk	Reference from cloud storage
Google Chat	No	Yes
Kafka	No	No
Line	No	No
Microsoft Teams	No	Yes
Opsgenie	No	Yes
Pagerduty	No	Yes
Prometheus Alertmanager	No	No
Pushover	Yes (Maximum of 1 per notification)	No
Sensu Go	No	No
Slack	Yes (when using Bot tokens, maximum of 5 per notification)	Yes (when using webhooks, maximum of 1 per notification)
Telegram	Yes	No
Threema	No	No
VictorOps	No	No
Webhook	No	Yes

Limitations

- This feature is not supported in Mimir or Loki, or when Grafana is configured to send alerts to other Alertmanagers such as the Prometheus Alertmanager.
- A number of contact points support at most one image per notification. In this case, just the first image is either uploaded to the receiving service or referenced from cloud storage per notification.
- When multiple alerts are sent in a single notification a screenshot might be included for each alert. The order the images are shown is random.
- If uploading screenshots to a cloud storage service such as Amazon S3, Azure Blob Storage or Google Cloud Storage; and accessing screenshots in the bucket requires authentication, logging into a VPN or corporate network; image previews might not work in all instant messaging and communication platforms as some services rewrite URLs to use their CDN.

Troubleshooting

If Grafana has been set up to send images in notifications, however notifications are still being received without them, follow the troubleshooting steps below:

1. Check that images in notifications has been set up as per the instructions.
2. Enable debug logging in Grafana and look for logs with the logger `ngalert.image`.
3. If the alert is not associated with a dashboard there will be logs for `Cannot take screenshot for alert rule as it is not associated with a dashboard`.
4. If the alert is associated with a dashboard, but no panel in the dashboard, there will be logs for `Cannot take screenshot for alert rule as it is not associated with a panel`.
5. If images cannot be taken because of mis-configuration or an issue with image rendering there will be logs for `Failed to take an image` including the Dashboard UID, Panel ID, and the error message.
6. Check that the contact point supports images in notifications and whether it supports uploading images to the receiving service or referencing images that have been uploaded to a cloud storage service.

Metrics

Grafana provides the following metrics to observe the performance and failure rate of images in notifications. For example, if a screenshot could not be taken within the expected time (10 seconds) then the counter `grafana_screenshot_failures_total` is updated.

- `grafana_alerting_image_cache_hits_total`
- `grafana_alerting_image_cache_misses_total`
- `grafana_screenshot_duration_seconds`
- `grafana_screenshot_failures_total`
- `grafana_screenshot_successes_total`
- `grafana_screenshot_upload_failures_total`
- `grafana_screenshot_upload_successes_total`

Manage contact points

The Contact points list view lists all existing contact points and notification templates.

On the **Contact Points** tab, you can:

- Search for name and type of contact points and integrations
- View all existing contact points and integrations
- View how many notification policies each contact point is being used for and navigate directly to the linked notification policies
- View the status of notification deliveries
- Export individual contact points or all contact points in JSON, YAML, or Terraform format
- Delete contact points that are not in use by a notification policy

On the **Notification templates** tab, you can:

- View, edit, copy or delete existing notification templates

Manage silences

Silences stop notifications from getting created and last for only a specified window of time.

NOTE

- Inhibition rules are not supported in the Grafana Alertmanager.
- The preview of silenced alerts only applies to alerts in firing state.

Add silences

To add a silence, complete the following steps.

1. In the left-side menu, click **Alerts & IRM** and then **Alerting**.
2. Click **Silences**.
3. From the **Alertmanager** dropdown, select an external Alertmanager to create and manage silences for the external data source. Otherwise, keep the default option of Grafana.
4. Click **Create silence** to open the Create silence page.
5. In **Silence start and end**, select the start and end date to indicate when the silence should go into effect and expire.
6. Optionally, in **Duration**, specify how long the silence is enforced. This automatically updates the end time in the **Silence start and end** field.
7. In the **Label** and **Value** fields, enter one or more *Matching Labels*. Matchers determine which rules the silence will apply to. Any matching alerts (in firing state) will show in the **Affected alert instances** field
8. In **Comment**, add details about the silence.

9. Click **Submit**.

Edit silences

To edit a silence, complete the following steps.

1. In the left-side menu, click **Alerts & IRM** and then **Alerting**.
2. Click **Silences** to view the list of existing silences.
3. Find the silence you want to edit, then click **Edit** (pen icon).
4. Make the desired changes, then click **Submit** to save your changes.

Create a URL to link to a silence form

When linking to a silence form, provide the default matching labels and comment via `matcher` and `comment` query parameters. The `matcher` parameter should be in the following format `[label][operator][value]` where the `operator` parameter can be one of the following: `=` (equals, not regex), `!=` (not equals, not regex), `=~` (equals, regex), `!~` (not equals, regex). The URL can contain many query parameters with the key `matcher`. For example, to link to silence form with matching labels `severity=critical` & `cluster!~europe-.*` and comment `Silence critical EU alerts`, create a URL `https://mygrafana/alerting/silence/new?matcher=severity%3Dcritical&matcher=cluster!~europe-*&comment=Silence%20critical%20EU%20alert`.

To link to a new silence page for an external Alertmanager, add a `alertmanager` query parameter with the Alertmanager data source name.

Remove silences

To remove a silence, complete the following steps.

1. In the left-side menu, click **Alerts & IRM** and then **Alerting**.
2. Click **Silences** to view the list of existing silences.
3. Select the silence you want to end, then click **Unsilence**.

Note: You cannot remove a silence manually. Silences that have ended are retained and listed for five days.

Useful links

Aggregation operators

View and filter alert rules

The Alerting page lists all existing alert rules. By default, rules are grouped by types of data sources. The Grafana section lists all Grafana managed rules. Alert rules for Prometheus compatible data sources are also listed here. You can view alert rules for Prometheus compatible data sources but you cannot edit them.

The Mimir/Cortex/Loki rules section lists all rules for Mimir, Cortex, or Loki data sources. Cloud alert rules are also listed in this section.

When managing large volumes of alerts, you can use extended alert rule search capabilities to filter on folders, evaluation groups, and rules. Additionally, you can filter alert rules by their properties like labels, state, type, and health.

View alert rules

To view alerting details:

1. Click **Alerts & IRM** → **Alert rules**. By default, the List view displays.
2. In **View as**, toggle between Grouped, List, or State views by clicking the relevant option. See Grouped view and State view for more information.

3 Expand the rule row to view the rule labels, annotations, data sources the rule queries, and a list of alert instances resulting from this rule.

Alerting rule details

From the Alert list page, you can also make copies of alert rules to help you reuse existing alert rules.

Export alert rules

Click the **Export rule group** icon next to each alert rule group to export to YAML, JSON, or Terraform.

Click **Export rules** to export all Grafana-managed alert rules to YAML, JSON, or Terraform.

Click **More** → **Modify export** next to each individual alert rule within a group to edit provisioned alert rules and export a modified version.

View query definitions for provisioned alerts

View read-only query definitions for provisioned alerts. Check quickly if your alert rule queries are correct, without diving into your “as-code” repository for rule definitions.

Grouped view

Grouped view shows Grafana alert rules grouped by folder and Loki or Prometheus alert rules grouped by `namespace` + `group`. This is the default rule list view, intended for managing rules. You can expand each group to view a list of rules in this group. Expand a rule further to view its details. You can also expand action buttons and alerts resulting from the rule to view their details.

Alerting grouped view

State view

State view shows alert rules grouped by state. Use this view to get an overview of which rules are in what state. Each rule can be expanded to view its details. Action buttons and any alerts generated by this rule, and each alert can be further expanded to view its details.

Alerting state view

Filter alert rules

To filter alert rules:

- From **Select data sources**, select a data source. You can see alert rules that query the selected data source.
- In the **Search by label**, enter search criteria using label selectors. For example, `environment=production,region=~US|EU,severity!=warning`.
- From **Filter alerts by state**, select an alerting state you want to see. You can see alerting rules that match the state. Rules matching other states are hidden.

Create mute timings

A mute timing is a recurring interval of time when no new notifications for a policy are generated or sent. Use them to prevent alerts from firing a specific and reoccurring period, for example, a regular maintenance period.

Similar to silences, mute timings do not prevent alert rules from being evaluated, nor do they stop alert instances from being shown in the user interface. They only prevent notifications from being created.

You can configure Grafana managed mute timings as well as mute timings for an external Alertmanager data source. For more information, refer to Alertmanager documentation.

Mute timings vs silences

The following table highlights the key differences between mute timings and silences.

Mute timing	Silence
Uses time interval definitions that can reoccur	Has a fixed start and end time
Is created and then added to notification policies	Uses labels to match against an alert to determine whether to silence or not

Add mute timings

1. In the left-side menu, click **Alerts & IRM**, and then **Alerting**.
2. Click **Notification policies** and then the **Mute Timings** tab.

3. From the **Alertmanager** dropdown, select an external Alertmanager. By default, the **Grafana Alertmanager** is selected.
4. Click **+ Add mute timing**.
5. Fill out the form to create a time interval to match against for your mute timing.
6. Save your mute timing.

Add mute timing to a notification policy

1. In the left-side menu, click **Alerts & IRM**, and then **Alerting**.
2. Click **Notification policies** and make sure you are on the **Notification Policies** tab.
3. Find the notification policy you would like to add the mute timing to and click **...** → **Edit**.
4. From the **Mute timings** dropdown, choose the mute timings you would like to add to the policy.
5. Save your changes.

Time intervals

A time interval is a specific duration during which alerts are suppressed. The duration typically consists of a specific time range and the days of the week, month, or year.

Supported time interval options are:

- Time range: The time inclusive of the start and exclusive of the end time (in UTC if no location has been selected, otherwise local time).
- Location: Depending on the location you select, the time range is displayed in local time.
- Days of the week: The day or range of days of the week. Example: `monday:thursday`.
- Days of the month: The date 1-31 of a month. Negative values can also be used to represent days that begin at the end of the month. For example: `-1` for the last day of the month.
- Months: The months of the year in either numerical or the full calendar month. For example: `1, may:august`.
- Years: The year or years for the interval. For example: `2021:2024`.

All fields are lists; to match the field, at least one list element must be satisfied. Fields also support ranges using `:` (e.g., `monday:thursday`).

If a field is left blank, any moment of time will match the field. For an instant of time to match a complete time interval, all fields must match. A mute timing can contain multiple time intervals.

If you want to specify an exact duration, specify all the options. For example, if you wanted to create a time interval for the first Monday of the month, for March, June, September, and December, between the hours of 12:00 and 24:00 UTC your time interval specification would be:

- Time range:
  - Start time: `12:00`
  - End time: `24:00`
- Days of the week: `monday`
- Months: `3, 6, 9, 12`
- Days of the month: `1:7`

View the state and health of alert rules

The state and health of alert rules helps you understand several key status indicators about your alerts.

There are three key components: alert rule state, alert instance state, and alert rule health. Although related, each component conveys subtly different information.

To view the state and health of your alert rules:

1. In the left-side menu, click **Alerts & IRM** and then **Alerting**.
2. Click **Alert rules** to view the list of existing alerts.
3. Click an alert rule to view its state, health, and state history.

Alert rule state

An alert rule can be in either of the following states:

State	Description
Normal	None of the time series returned by the evaluation engine is in a `Pending` or `Firing` state.
Pending	At least one time series returned by the evaluation engine is `Pending`.
Firing	At least one time series returned by the evaluation engine is `Firing`.

NOTE

Alerts will transition first to `pending` and then `firing`, thus it will take at least two evaluation cycles before an alert is fired.

Alert instance state

An alert instance can be in either of the following states:

State	Description
Normal	The state of an alert that is neither firing nor pending, everything is working correctly.
Pending	The state of an alert that has been active for less than the configured threshold duration.
Alerting	The state of an alert that has been active for longer than the configured threshold duration.
NoData	No data has been received for the configured time window.
Error	The error that occurred when attempting to evaluate an alerting rule.

Keep last state

An alert rule can be configured to keep the last state when a `NoData` and/or `Error` state is encountered. This will both prevent alerts from firing, and from resolving and re-firing. Just like normal evaluation, the alert rule will transition from `Pending` to `Firing` after the pending period has elapsed.

Alert rule health

An alert rule can have one the following health statuses:

State	Description
Ok	No error when evaluating an alerting rule.
Error	An error occurred when evaluating an alerting rule.
NoData	The absence of data in at least one time series returned during a rule evaluation.
{status}, KeepLast	The rule would have received another status but was configured to keep the last state of the alert rule.

Special alerts for `NoData` and `Error`

When evaluation of an alerting rule produces state `NoData` or `Error`, Grafana Alerting will generate alert instances that have the following additional labels:

Label	Description
alertname	Either `DatasourceNoData` or `DatasourceError` depending on the state.
datasource_uid	The UID of the data source that caused the state.

> **NOTE**
>
> You will need to set the No Data and Error Handling to `No Data` or `Error` in the alert rule as per this doc: https://grafana.com/docs/grafana/latest/alerting/alerting-rules/create-grafana-managed-rule/#configure-no-data-and-error-handling in order to generate the additional labels.

You can handle these alerts the same way as regular alerts by adding a silence, route to a contact point, and so on.

State history view

Use the State history view to get insight into how your alert instances behave over time. View information on when a state change occurred, what the previous state was, the current state, any other alert instances that changed their state at the same time as well as what the query value was that triggered the change.

> **NOTE**
>
> Open source users must configure alert state history in order to be able to access the view.

View state history

To use the State history view, complete the following steps.

1. Navigate to **Alerts & IRM** → **Alerting** → **Alert rules**.
2. Click an alert rule.
3. Select **Show state history**.

   The State history view opens.

   The timeline view at the top displays a timeline of changes for the past hour, so you can track how your alert instances are behaving over time.

   The bottom part shows the alert instances, their previous and current state, the value of each part of the expression and a unique set of labels.

   Common labels are displayed at the top to make it easier to identify different alert instances.
4. From the timeline view, hover over a time to get an automatic display of all the changes that happened at that particular moment.

These changes are displayed in real time in the timestamp view at the bottom of the page. The timestamp view is a list of all the alert instances that changed state at that point in time. The visualization only displays 12 instances by default.

The value shown for each instance is for each part of the expression that was evaluated.

5. Click the labels to filter and narrow down the results.

View and filter by alert groups

Alert groups show grouped alerts from an Alertmanager instance. By default, alert rules are grouped by the label keys for the default policy in notification policies. Grouping common alert rules into a single alert group prevents duplicate alert rules from being fired.

You can view alert groups and also filter for alert rules that match specific criteria.

View alert groups

To view alert groups, complete the following steps.

1. In the left-side menu, click **Alerts & IRM** and then **Alerting**.
2. Click **Groups** to view the list of existing groups.
3. From the **Alertmanager** dropdown, select an external Alertmanager as your data source. By default, the `Grafana` Alertmanager is selected.
4. From **Custom group by** dropdown, select a combination of labels to view a grouping other than the default. This is useful for debugging and verifying your grouping of notification policies.

If an alert does not contain labels specified either in the grouping of the default policy or the custom grouping, then the alert is added to a catch all group with a header of `No grouping`.

Filter alerts

You can filter by label or state.

Search by label

In **Search**, enter an existing label to view alerts matching the label.

For example, `environment=production,region=~US|EU,severity!=warning`.

Filter by state

In **States**, select from Active, Suppressed, or Unprocessed states to view alerts matching your selected state. All other alerts are hidden.

View notification errors

View notification errors and understand why they failed to be sent or were not received.

Note: This feature only works if you are using Grafana Alertmanager.

To view notification errors, complete the following steps.

1. Navigate to Alerting → Contact points.

   If any contact points are failing, a message at the right-hand corner of the screen alerts the user to the fact that there are errors and how many.

2. Click on the contact point to view the details of errors for each contact point.

   Error details are displayed if you hover over the Error icon.

   If a contact point has more than one integration, you see all errors for each of the integrations listed.

3. In the Health column, check the status of the notification.

   This can be either OK, No attempts, or Error.

Useful links

Receivers API

Declare incidents from firing alerts

Declare an incident from a firing alert to streamline your alert to incident workflow.

Before you begin

- Ensure you have Grafana Incident installed
- You must have a firing alert

Procedure

To declare an incident from a firing alert, complete the following steps.

1. Navigate to Alerts & Incidents → Alerting → Alert rules.
2. From the Alert rules list view, click the firing alert that you want to declare an incident for.

   Note:

   You can also access **Declare Incident** from the alert details page.
3. Click **Declare Incident**. The **Declare Incident** pop-up opens in the Grafana Incident application.
4. In the **Declare Incident** pop-up, enter what's going on.

   Note: This field is pre-filled with the name of the alert rule, but you can edit it as required.

   The alert rule is also linked to the incident.
5. Select a severity.
6. Add labels, as required.
7. Click **More options** to include a channel prefix and status.

8 Click **Declare Incident**.

Next steps

View and track the incident in the Grafana Incident application.

For more information, refer to Grafana Incident documentation.

Meta monitoring

Monitor your alerting metrics to ensure you identify potential issues before they become critical.

Meta monitoring is the process of monitoring your monitoring system and alerting when your monitoring is not working as it should.

In order to enable you to meta monitor, Grafana provides predefined metrics.

Identify which metrics are critical to your monitoring system (i.e. Grafana) and then set up how you want to monitor them.

You can use meta-monitoring metrics to understand the health of your alerting system in the following ways:

1. Optional: Create a dashboard in Grafana that uses this metric in a panel (just like you would for any other kind of metric).
2. Optional: Create an alert rule in Grafana that checks this metric regularly (just like you would do for any other kind of alert rule).
3. Optional: Use the Explore module in Grafana.

Metrics for Grafana-managed alerts

To meta monitor Grafana-managed alerts, you need a Prometheus server, or other metrics database to collect and store metrics exported by Grafana.

For example, if you are using Prometheus, add a `scrape_config` to Prometheus to scrape metrics from Grafana, Alertmanager, or your data sources.

Example

yaml Copy

```
- job_name: grafana
  honor_timestamps: true
  scrape_interval: 15s
  scrape_timeout: 10s
  metrics_path: /metrics
  scheme: http
  follow_redirects: true
  static_configs:
    - targets:
        - grafana:3000
```

List of available metrics

The Grafana ruler, which is responsible for evaluating alert rules, and the Grafana Alertmanager, which is responsible for sending notifications of firing and resolved alerts, provide a number of metrics that let you observe them.

grafana_alerting_alerts

This metric is a counter that shows you the number of `normal`, `pending`, `alerting`, `nodata` and `error` alerts. For example, you might want to create an alert that fires when `grafana_alerting_alerts{state="error"}` is greater than 0.

grafana_alerting_schedule_alert_rules

This metric is a gauge that shows you the number of alert rules scheduled. An alert rule is scheduled unless it is paused, and the value of this metric should match the total number of non-paused alert rules in Grafana.

grafana_alerting_schedule_periodic_duration_seconds_bucket

This metric is a histogram that shows you the time it takes to process an individual tick in the scheduler that evaluates alert rules. If the scheduler takes longer than 10 seconds to process a tick then pending evaluations will start to accumulate such that alert rules might later than expected.

grafana_alerting_schedule_query_alert_rules_duration_seconds_bucket

This metric is a histogram that shows you how long it takes the scheduler to fetch the latest rules from the database. If this metric is elevated then so will `schedule_periodic_duration_seconds`.

grafana_alerting_scheduler_behind_seconds

This metric is a gauge that shows you the number of seconds that the scheduler is behind where it should be. This number will increase if `schedule_periodic_duration_seconds` is longer than 10 seconds, and decrease when it is less than 10 seconds. The smallest possible value of this metric is 0.

grafana_alerting_notification_latency_seconds_bucket

This metric is a histogram that shows you the number of seconds taken to send notifications for firing and resolved alerts. This metric will let you observe slow or over-utilized integrations, such as an SMTP server that is being given emails faster than it can send them.

Metrics for Mimir-managed alerts

To meta monitor Grafana Mimir-managed alerts, open source and on-premise users need a Prometheus/Mimir server, or another metrics database to collect and store metrics exported by the Mimir ruler.

rule_evaluation_failures_total

This metric is a counter that shows you the total number of rule evaluation failures.

Metrics for Alertmanager

To meta monitor the Alertmanager, you need a Prometheus/Mimir server, or another metrics database to collect and store metrics exported by Alertmanager.

For example, if you are using Prometheus you should add a `scrape_config` to Prometheus to scrape metrics from your Alertmanager.

Example

yaml Copy

```yaml
- job_name: alertmanager
  honor_timestamps: true
  scrape_interval: 15s
  scrape_timeout: 10s
  metrics_path: /metrics
  scheme: http
  follow_redirects: true
  static_configs:
    - targets:
        - alertmanager:9093
```

List of available metrics

The following is a list of available metrics for Alertmanager.

alertmanager_alerts

This metric is a counter that shows you the number of active, suppressed, and unprocessed alerts in Alertmanager. Suppressed alerts are silenced alerts, and unprocessed alerts are alerts that have been sent to the Alertmanager but have not been processed.

alertmanager_alerts_invalid_total

This metric is a counter that shows you the number of invalid alerts that were sent to Alertmanager. This counter should not exceed 0, and so in most cases you will want to create an alert that fires if whenever this metric increases.

alertmanager_notifications_total

This metric is a counter that shows you how many notifications have been sent by Alertmanager. The metric uses a label “integration” to show the number of notifications sent by integration, such as email.

alertmanager_notifications_failed_total

This metric is a counter that shows you how many notifications have failed in total. This metric also uses a label “integration” to show the number of failed notifications by integration, such as failed emails. In most cases you will want to use the `rate` function to understand how often notifications are failing to be sent.

alertmanager_notification_latency_seconds_bucket

This metric is a histogram that shows you the amount of time it takes Alertmanager to send notifications and for those notifications to be accepted by the receiving service. This metric uses a label “integration” to show the amount of time by integration. For example, you can use this metric to show the 95th percentile latency of sending emails.

Metrics for Alertmanager in high availability mode

If you are using Alertmanager in high availability mode there are a number of additional metrics that you might want to create alerts for.

alertmanager_cluster_members

This metric is a gauge that shows you the current number of members in the cluster. The value of this gauge should be the same across all Alertmanagers. If different Alertmanagers are showing different numbers of members then this is indicative of an issue with your Alertmanager cluster. You should look at the metrics and logs from your Alertmanagers to better understand what might be going wrong.

alertmanager_cluster_failed_peers

This metric is a gauge that shows you the current number of failed peers.

alertmanager_cluster_health_score

This metric is a gauge showing the health score of the Alertmanager. Lower values are better, and zero means the Alertmanager is healthy.

alertmanager_cluster_peer_info

This metric is a gauge. It has a constant value `1`, and contains a label called “peer” containing the Peer ID of each known peer.

alertmanager_cluster_reconnections_failed_total

This metric is a counter that shows you the number of failed peer connection attempts. In most cases you will want to use the `rate` function to understand how often reconnections fail as this may be indicative of an issue or instability in your network.

Troubleshooting

This page lists some tools and advice to help troubleshoot common Grafana issues.

Troubleshoot with logs

If you encounter an error or problem, then you can check the Grafana server log. Usually located at `/var/log/grafana/grafana.log` on Unix systems or in `<grafana_install_dir>/data/log` on other platforms and manual installations.

You can enable more logging by changing log level in the Grafana configuration file.

For more information, refer to Enable debug logging in Grafana CLI and the log section in Configuration.

Troubleshoot with Dashboards Panels

If you have an issue with your Dashboard panel, you can send us debug information. For more information, refer to Send a panel to Grafana Labs support.

Troubleshoot with support bundles

If you have an issue with your Grafana instance, you can generate an archive containing information concerning the state and the configuration of the instance.

To send us a bundle for advanced support, refer to Send a support bundle to Grafana Labs support.

Troubleshoot transformations

Order of transformations matters. If the final data output from multiple transformations looks wrong, try changing the transformation order. Each transformation transforms data returned by the previous transformation, not the original raw data.

For more information, refer to Debug a transformation.

Text missing with server-side image rendering (RPM-based Linux)

Server-side image (png) rendering is a feature that is optional but very useful when sharing visualizations, for example in alert notifications.

If the image is missing text, then make sure you have font packages installed.

bash Copy

```
sudo yum install fontconfig
sudo yum install freetype*
sudo yum install urw-fonts
```

More help

Check out the Grafana Community for more troubleshooting help (you must be logged in to post or comment).

Send a panel to Grafana Labs support

When you encounter problems with any of your visualizations, you can send the panel JSON model to Grafana Labs Technical Support and request help with troubleshooting your issue.

The panel that you send includes all query response data and all visualizations settings. Upon receiving your panel, Grafana Labs Technical Support imports your data into a local version of Grafana and begins researching your problem.

1. Open the dashboard that contains the panel you want to send to Grafana Labs.
2. Hover over any part of the panel to display the actions menu on the top right corner.
3. Click the menu and select **More > Get help**.

   Grafana opens a standalone support dashboard that contains the data you are sending to Grafana Labs Technical Support.

   If you enable iframes, the support dashboard includes the visualization that looks similar to the following image.

4. To send the panel data to Grafana Labs via Github:

   a. Click **Copy to clipboard**.

   b. In the Grafana/Grafana repository, create an issue, and paste the contents of the support dashboard.

5. To send the panel data to Grafana Labs via a support ticket:

   a. Click **Dashboard**.

   Grafana downloads the support dashboard to a TXT file.

   b. Attach the TXT file to a support ticket that you send to Grafana Labs Technical Support.

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Send a support bundle to Grafana Labs support

When you encounter problems with your Grafana instance, you can send us a support bundle that contains information about your Grafana instance, including:

- Grafana version
- Installed plugins
- Grafana configuration
- Deployed database information and migrations

Note: The Support Bundle is available on Grafana instances running 9.5 and above.

Available support bundle components

A support bundle can include any of the following components:

- **Usage statistics**: Usage statistic for the Grafana instance
- **User information**: A list of users of the Grafana instance
- **Database and Migration information**: Database information and migration log
- **Plugin information**: Plugin information for the Grafana instance
- **Basic information**: Basic information about the Grafana instance (version, memory usage, and so on)
- **Settings**: Settings for the Grafana instance
- **SAML**: Healthcheck connection and metadata for SAML (only displayed if SAML is enabled)
- **LDAP**: Healthcheck connection and metadata for LDAP (only displayed if LDAP is enabled)

- **OAuth2**: Healthcheck connection and metadata for each OAuth2 Provider supporter (only displayed if OAuth provider is enabled)

Before you begin

To follow these instructions, you need the following permissions:

- In Grafana Cloud, you need the organization administrator role.
- In Grafana on-premises, you need the Grafana server administrator role.

  Note that you can set `server_admin_only` configuration option to `false` to allow organization administrators to access support bundles in Grafana on-premises.

Steps

To generate a support bundle and send the support bundle to Grafana Labs via a support ticket:

1. Click the Help icon.
2. Click **Support Bundles**.
3. Click **New Support Bundle**.
4. Select the components that you want to include in the support bundle.
5. Click **Create**.
6. After the support bundle is ready, click **Download**.

   Grafana downloads the support bundle to an archive (tar.gz) file.
7. Attach the archive (tar.gz) file to a support ticket that you send to Grafana Labs Technical Support.

Support bundle configuration

You can configure the following settings for support bundles in the Grafana configuration file:

ini

Copy

```ini
[support_bundles]
# Enable support bundle creation (default: true)
enabled = true
# Only server admins can generate and view support bundles. When set to false, organization admins
server_admin_only = true
# If set, bundles will be encrypted with the provided public keys separated by whitespace
public_keys = ""
```

Encrypting a support bundle

Support bundles can be encrypted with age before they are sent to recipients. This is useful when you want to send a support bundle to Grafana through a channel that is not private.

Generate a key pair

Ensure age is installed on your system.

bash

Copy

```bash
$ age-keygen -o key.txt
Public key: age1ql3z7hjy54pw3hyww5ayyfg7zqgvc7w3j2elw8zmrj2kg5sfn9aqmcac8p
```

Support bundle encryption

Ensure age is installed on your system.

Add the public key to the `public_keys` setting in the `support_bundle` section of the Grafana configuration file.

ini

Copy

```ini
[support_bundles]
public_keys = "age1ql3z7hjy54pw3hyww5ayyfg7zqgvc7w3j2elw8zmrj2kg5sfn9aqmcac8p"
```

Multiple public keys can be defined by separating them with whitespace. All included public keys will be able to decrypt the support bundle.

Example:

```ini
[support_bundles]
public_keys = "age1ql3z7hjy54pw3hyww5ayyfg7zqgvc7w3j2elw8zmrj2kg5sfn9aqmcac8p age1yu8vzu554pv3klw4
```

When you restart Grafana, new support bundles will be encrypted with the provided public keys. The support bundle file extension is `tar.gz.age`.

Decrypt a support bundle

Ensure age is installed on your system.

Execute the following command to decrypt the support bundle:

```bash
age --decrypt -i keyfile -o output.tar.gz downloaded.tar.gz.age
```

Example:

```bash
age --decrypt -i key.txt -o data.tar.gz af6684b4-d613-4b31-9fc3-7cb579199bea.tar.gz.age
```

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Developers

Go to the Grafana developer portal to access the following documentation:

- Grafana plugin development
- Grafana design system
- Grafana Scenes
- Grafana data plane

This section of our documentation contains additional resources:

- HTTP API
- Contribute to Grafana
- Contributor License Agreement (CLA)
- Angular support deprecation

You might also find the following resources to be helpful:

- Grafana Tutorials: Step-by-step guides that help you make the most of Grafana.
- Grafana Community Forums: Get technical support for open source Grafana, Loki, and Tempo.

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

HTTP API reference

The Grafana backend exposes an HTTP API, which is the same API that is used by the frontend to do everything from saving dashboards, creating users, and updating data sources.

Since version 8.4, HTTP API details are specified using OpenAPI v2.

Starting from version 9.1, there is also a OpenAPI v3 specification (generated by the v2 one).

Users can browser and try out both via the Swagger UI editor (served by the grafana server) by navigating to `/swagger-ui`.

Authenticating API requests

You can authenticate requests using basic auth, a service account token or a session cookie (acquired using regular login or OAuth).

Basic Auth

If basic auth is enabled (it is enabled by default), then you can authenticate your HTTP request via standard basic auth. Basic auth will also authenticate LDAP users.

curl example:

bash Copy

```
curl http://admin:admin@localhost:3000/api/org
{"id":1,"name":"Main Org."}
```

Service Account Token

To create a service account token, click on **Administration** in the left-side menu, click **Users and access**, then **Service Accounts**. For more information on how to use service account tokens, refer

to the Service Accounts documentation.

You use the token in all requests in the `Authorization` header, like this:

Example:

```http
GET http://your.grafana.com/api/dashboards/db/mydash HTTP/1.1
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

The `Authorization` header value should be `Bearer <your service account token>`.

X-Grafana-Org-Id Header

X-Grafana-Org-Id is an optional property that specifies the organization to which the action is applied. If it is not set, the created key belongs to the current context org. Use this header in all requests except those regarding admin.

Example Request:

```http
GET /api/org/ HTTP/1.1
Accept: application/json
Content-Type: application/json
X-Grafana-Org-Id: 2
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

HTTP APIs

- Admin API
- Alerting API (unstable)
- Alerting Provisioning API
- Annotations API
- Correlations API
- Dashboard API
- Dashboard Permissions API
- Dashboard Versions API
- Data source API
- Folder API

- Folder Permissions API
- Folder/Dashboard Search API
- Library Element API
- Organization API
- Other API
- Playlists API
- Preferences API
- Short URL API
- Query history API
- Snapshot API
- Team API
- User API

Deprecated HTTP APIs

- Alerting Notification Channels API
- Alerting API
- Authentication API

Grafana Enterprise HTTP APIs

Grafana Enterprise includes all of the Grafana OSS APIs as well as those that follow:

- Role-based access control API
- Data source permissions API
- Team sync API
- License API
- Reporting API
- Query and resource caching API

Create Service Account tokens and dashboards for an organization

Use the Grafana API to set up new Grafana organizations or to add dynamically generated dashboards to an existing organization.

Authentication

There are two authentication methods to access the API:

- Basic authentication: A Grafana Admin user can access some parts of the Grafana API through basic authentication.
- Service Account tokens: All organization actions can be accessed through a Service Account token. A Service Account token is associated with an organization. It can be used to create dashboards and other components specific for that organization.

How to create a new organization and a Service Account Token

The task is to create a new organization and then add a Token that can be used by other users. In the examples below which use basic auth, the user is `admin` and the password is `admin`.

1. Create the org. Here is an example using curl:

```bash
curl -X POST -H "Content-Type: application/json" -d '{"name":"apiorg"}' http://admin:admin@l
```

Copy

This should return a response: `{"message":"Organization created","orgId":6}`. Use the orgId for the next steps.

2 Optional step. If the org was created previously and/or step 3 fails then first add your Admin user to the org:

```bash
curl -X POST -H "Content-Type: application/json" -d '{"loginOrEmail":"admin", "role": "Admin'
```

3 Switch the org context for the Admin user to the new org:

```bash
curl -X POST http://admin:admin@localhost:3000/api/user/using/<id of new org>
```

4 Create a Service Account:

```bash
curl -X POST -H "Content-Type: application/json" -d '{"name":"test", "role": "Admin"}' http:/
```

5 Create a Service Account token for the service account created in the previous step:

```bash
curl -X POST -H "Content-Type: application/json" -d '{"name":"test-token"}' http://admin:admi
```

This should return a response:

```http
HTTP/1.1 200
Content-Type: application/json

{
  "id": 7,
  "name": "test-token",
  "key": "eyJrIjoiVjFxTHZ6dGdPSjg5Um92MjN1RlhjMkNqYkZUbm9jYkwiLCJuIjoiZ3JhZmFuYSIsImlkIjoxfQ=
}
```

Save the key returned here in your password manager as it is not possible to fetch again it in the future.

How to add a dashboard

Using the Token that was created in the previous step, you can create a dashboard or carry out other actions without having to switch organizations.

1 Add a dashboard using the key (or bearer token as it is also called):

bash Copy

```
curl -X POST --insecure -H "Authorization: Bearer eyJrIjoiR0ZXZmt1UFc0OEpIOGN5RWdUalBJTllUTk83Vlht
  "dashboard": {
    "id": null,
    "title": "Production Overview",
    "tags": [ "templated" ],
    "timezone": "browser",
    "rows": [
      {
      }
    ],
    "schemaVersion": 6,
    "version": 0
  },
  "overwrite": false
}' http://localhost:3000/api/dashboards/db
```

Note: If you export a dashboard for sharing externally using the Share > Export menu in the Grafana UI, you cannot import that dashboard. Instead, click **View JSON** and save it to a file or fetch the JSON output through the API.

Admin API

The Admin HTTP API does not currently work with an API Token. API Tokens are currently only linked to an organization and an organization role. They cannot be given the permission of server admin, only users can be given that permission. So in order to use these API calls you will have to use Basic Auth and the Grafana user must have the Grafana Admin permission. (The default admin user is called `admin` and has permission to use this API.)

If you are running Grafana Enterprise, for some endpoints you'll need to have specific permissions. Refer to Role-based access control permissions for more information.

Fetch settings

`GET /api/admin/settings`

Only works with Basic Authentication (username and password). See introduction for an explanation.

Required permissions

See note in the introduction for an explanation.

Action	Scope
settings:read	settings:* *settings:auth.saml:* settings:auth.saml:enabled (property level)

Example Request:

```http
GET /api/admin/settings
Accept: application/json
Content-Type: application/json
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{
  "DEFAULT": {
    "app_mode":"production"
  },
  "analytics": {
    "google_analytics_ua_id":"",
    "reporting_enabled":"false"
  },
  "auth.anonymous":{
    "enabled":"true",
    "org_name":"Main Org.",
    "org_role":"Viewer"
  },
  "auth.basic":{
    "enabled":"false"
  },
```

Expand code

Update settings

`PUT /api/admin/settings`

> **NOTE**
> Available in Grafana Enterprise v8.0+.

Updates / removes and reloads database settings. You must provide either `updates`, `removals` or both.

This endpoint only supports changes to `auth.saml` configuration.

Required permissions

See note in the introduction for an explanation.

Action	Scope
settings:write	settings:* *settings:auth.saml:* settings:auth.saml:enabled (property level)

Example request:

http Copy

```
PUT /api/admin/settings
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "updates": {
    "auth.saml": {
      "enabled": "true"
    }
  },
  "removals": {
    "auth.saml": ["single_logout"]
  },
}
```

Example response:

http Copy

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 32

{
  "message":"Settings updated"
}
```

Status codes:

- **200** - OK

- **400** - Bad Request
- **401** - Unauthorized
- **403** - Forbidden
- **500** - Internal Server Error

Grafana Stats

`GET /api/admin/stats`

Only works with Basic Authentication (username and password). See introduction for an explanation.

Required permissions

See note in the introduction for an explanation.

Action	Scope
server.stats:read	n/a

Example Request:

http Copy

```
GET /api/admin/stats
Accept: application/json
Content-Type: application/json
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{
  "users":2,
  "orgs":1,
  "dashboards":4,
  "snapshots":2,
  "tags":6,
  "datasources":1,
  "playlists":1,
```

```
  "stars":2,
  "alerts":2,
  "activeUsers":1
}
```


Grafana Usage Report preview

`GET /api/admin/usage-report-preview`

Preview usage report to be sent to vendor.

Only works with Basic Authentication (username and password). See introduction for an explanation.

Example Request:

http Copy

```
GET /api/admin/usage-report-preview
Accept: application/json
Content-Type: application/json
```

Example Response:

http Copy


```
HTTP/1.1 200
Content-Type: application/json

{
    "version": "8_4_0",
    "metrics": {
        "stats.active_admins.count": 1,
        "stats.active_editors.count": 1,
        "stats.active_sessions.count": 0,
        "stats.active_users.count": 2,
        "stats.active_viewers.count": 0,
        "stats.admins.count": 1,
        "stats.alert_rules.count": 0,
        "stats.alerting.ds.other.count": 0,
        "stats.alerts.count": 5,
        "stats.annotations.count": 6,
        "stats.api_keys.count": 1
  }
}
```


Expand code

Global Users

`POST /api/admin/users`

Create new user. Only works with Basic Authentication (username and password). See introduction for an explanation.

Required permissions

See note in the introduction for an explanation.

Action	Scope
users:create	n/a

Example Request:

http Copy

```
POST /api/admin/users HTTP/1.1
Accept: application/json
Content-Type: application/json

{
  "name":"User",
  "email":"user@graf.com",
  "login":"user",
  "password":"userpassword",
  "OrgId": 1
}
```

Note that `OrgId` is an optional parameter that can be used to assign a new user to a different organization when auto_assign_org is set to `true`.

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{"id":5,"message":"User created"}
```

Password for User

`PUT /api/admin/users/:id/password`

Only works with Basic Authentication (username and password). See introduction for an explanation. Change password for a specific user.

Required permissions

See note in the introduction for an explanation.

Action	Scope
users.password:write	global.users:*

Example Request:

http Copy

```
PUT /api/admin/users/2/password HTTP/1.1
Accept: application/json
Content-Type: application/json

{"password":"userpassword"}
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{"message": "User password updated"}
```

Permissions

`PUT /api/admin/users/:id/permissions`

Only works with Basic Authentication (username and password). See introduction for an explanation.

Required permissions

See note in the introduction for an explanation.

Action	Scope
users.permissions:write	global.users:*

Example Request:

```http
PUT /api/admin/users/2/permissions HTTP/1.1
Accept: application/json
Content-Type: application/json

{"isGrafanaAdmin": true}
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{"message": "User permissions updated"}
```

Delete global User

`DELETE /api/admin/users/:id`

Only works with Basic Authentication (username and password). See introduction for an explanation.

Required permissions

See note in the introduction for an explanation.

Action	Scope
users:delete	global.users:*

Example Request:

```http
```

```
DELETE /api/admin/users/2 HTTP/1.1
Accept: application/json
Content-Type: application/json
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{"message": "User deleted"}
```

Pause all alerts

`POST /api/admin/pause-all-alerts`

> **NOTE**
>
> This API is relevant for the legacy dashboard alerts only. For default alerting, use silences to stop alerts from being delivered.

Only works with Basic Authentication (username and password). See introduction for an explanation.

Example Request:

http Copy

```
POST /api/admin/pause-all-alerts HTTP/1.1
Accept: application/json
Content-Type: application/json

{
  "paused": true
}
```

JSON Body schema:

- **paused** – If true then all alerts are to be paused, false unpauses all alerts.

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{
  "state":   "Paused",
  "message": "alert paused",
  "alertsAffected": 1
}
```

Auth tokens for User

`GET /api/admin/users/:id/auth-tokens`

Return a list of all auth tokens (devices) that the user currently have logged in from.

Only works with Basic Authentication (username and password). See introduction for an explanation.

Required permissions

See note in the introduction for an explanation.

Action	Scope
users.authtoken:read	global.users:*

Example Request:

http Copy

```
GET /api/admin/users/1/auth-tokens HTTP/1.1
Accept: application/json
Content-Type: application/json
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

[
  {
    "id": 361,
```

```
    "isActive": false,
    "clientIp": "127.0.0.1",
    "browser": "Chrome",
    "browserVersion": "72.0",
    "os": "Linux",
    "osVersion": "",
    "device": "Other",
    "createdAt": "2019-03-05T21:22:54+01:00",
    "seenAt": "2019-03-06T19:41:06+01:00"
  },
  {
    "id": 364,
```


Expand code

Revoke auth token for User

`POST /api/admin/users/:id/revoke-auth-token`

Revokes the given auth token (device) for the user. User of issued auth token (device) will no longer be logged in and will be required to authenticate again upon next activity.

Only works with Basic Authentication (username and password). See introduction for an explanation.

Required permissions

See note in the introduction for an explanation.

Action	Scope
users.authtoken:write	global.users:*

Example Request:

http

Copy

```
POST /api/admin/users/1/revoke-auth-token HTTP/1.1
Accept: application/json
Content-Type: application/json

{
  "authTokenId": 364
}
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{
  "message": "User auth token revoked"
}
```

Logout User

`POST /api/admin/users/:id/logout`

Logout user revokes all auth tokens (devices) for the user. User of issued auth tokens (devices) will no longer be logged in and will be required to authenticate again upon next activity.

Only works with Basic Authentication (username and password). See introduction for an explanation.

Required permissions

See note in the introduction for an explanation.

Action	Scope
users.logout	global.users:*

Example Request:

```http
POST /api/admin/users/1/logout HTTP/1.1
Accept: application/json
Content-Type: application/json
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{
```

```
  "message": "User auth token revoked"
}
```

Reload provisioning configurations

`POST /api/admin/provisioning/dashboards/reload`

`POST /api/admin/provisioning/datasources/reload`

`POST /api/admin/provisioning/plugins/reload`

`POST /api/admin/provisioning/notifications/reload`

`POST /api/admin/provisioning/access-control/reload`

`POST /api/admin/provisioning/alerting/reload`

Reloads the provisioning config files for specified type and provision entities again. It won't return until the new provisioned entities are already stored in the database. In case of dashboards, it will stop polling for changes in dashboard files and then restart it with new configurations after returning.

Only works with Basic Authentication (username and password). See introduction for an explanation.

Required permissions

See note in the introduction for an explanation.

Action	Scope	Provision entity
provisioning:reload	provisioners:accesscontrol	accesscontrol
provisioning:reload	provisioners:dashboards	dashboards
provisioning:reload	provisioners:datasources	datasources
provisioning:reload	provisioners:plugins	plugins
provisioning:reload	provisioners:notifications	notifications
provisioning:reload	provisioners:alerting	alerting

Example Request:

```http
POST /api/admin/provisioning/dashboards/reload HTTP/1.1
Accept: application/json
Content-Type: application/json
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{
  "message": "Dashboards config reloaded"
}
```

Reload LDAP configuration

`POST /api/admin/ldap/reload`

Reloads the LDAP configuration.

Only works with Basic Authentication (username and password). See introduction for an explanation.

Example Request:

```http
POST /api/admin/ldap/reload HTTP/1.1
Accept: application/json
Content-Type: application/json
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{
  "message": "LDAP config reloaded"
}
```

Rotate data encryption keys

`POST /api/admin/encryption/rotate-data-keys`

Rotates data encryption keys.

Example Request:

```http
POST /api/admin/encryption/rotate-data-keys HTTP/1.1
Accept: application/json
Content-Type: application/json
```

Example Response:

```http
HTTP/1.1 204
Content-Type: application/json
```

Re-encrypt data encryption keys

`POST /api/admin/encryption/reencrypt-data-keys`

Re-encrypts data encryption keys.

Example Request:

```http
POST /api/admin/encryption/reencrypt-data-keys HTTP/1.1
Accept: application/json
Content-Type: application/json
```

Example Response:

```http
HTTP/1.1 204
Content-Type: application/json
```

Re-encrypt secrets

`POST /api/admin/encryption/reencrypt-secrets`

Re-encrypts secrets.

Example Request:

```http
```

```
POST /api/admin/encryption/reencrypt-secrets HTTP/1.1
Accept: application/json
Content-Type: application/json
```

Example Response:

```
http
HTTP/1.1 204
Content-Type: application/json
```

Roll back secrets

`POST /api/admin/encryption/rollback-secrets`

Rolls back secrets.

Example Request:

```
http
POST /api/admin/encryption/rollback-secrets HTTP/1.1
Accept: application/json
Content-Type: application/json
```

Example Response:

```
http
HTTP/1.1 204
Content-Type: application/json
```

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Alerting provisioning HTTP API

The Alerting Provisioning HTTP API can be used to create, modify, and delete resources relevant to Grafana-managed alerts. This API is the one used by our Grafana Terraform provider.

For more information on the differences between Grafana-managed and data source-managed alerts, refer to Introduction to alert rules.

If you are running Grafana Enterprise, you need to add specific permissions for some endpoints. For more information, refer to Role-based access control permissions.

Grafana-managed endpoints

Note that the JSON format from most of the following endpoints is not fully compatible with provisioning via configuration JSON files.

Alert rules

Method	URI	Name	Summary
DELETE	/api/v1/provisioning/alert-rules/:uid	route delete alert rule	Delete a specific alert rule by UID.
GET	/api/v1/provisioning/alert-rules/:uid	route get alert rule	Get a specific alert rule by UID.
POST	/api/v1/provisioning/alert-rules	route post alert rule	Create a new alert rule.

Method	URI	Name	Summary
PUT	/api/v1/provisioning/alert-rules/:uid	route put alert rule	Update an existing alert rule.
GET	/api/v1/provisioning/alert-rules/:uid/export	route get alert rule export	Export an alert rule in provisioning file format.
GET	/api/v1/provisioning/folder/:folderUid/rule-groups/:group	route get alert rule group	Get a rule group.
PUT	/api/v1/provisioning/folder/:folderUid/rule-groups/:group	route put alert rule group	Update the interval of a rule group or modify the rules of the group.
GET	/api/v1/provisioning/folder/:folderUid/rule-groups/:group/export	route get alert rule group export	Export an alert rule group in provisioning file format.
GET	/api/v1/provisioning/alert-rules	route get alert rules	Get all the alert rules.
GET	/api/v1/provisioning/alert-rules/export	route get alert rules export	Export all alert rules in provisioning file format.

Example request for new alert rule:

http

Copy

```
POST /api/v1/provisioning/alert-rules
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "title": "TEST-API_1",
  "ruleGroup": "API",
  "folderUID": "SET_FOLDER_UID",
  "noDataState": "OK",
  "execErrState": "OK",
  "for": "5m",
  "orgId": 1,
  "uid": "",
  "condition": "B",
  "annotations": {
    "summary": "test_api_1"
  },
  "labels": {
```

Expand code

Example Response:

```http
HTTP/1.1 201 Created
Content-Type: application/json

{
  "id": 1,
  "uid": "XXXXXXXXX",
  "orgID": 1,
  "folderUID": "SET_FOLDER_UID",
  "ruleGroup": "API3",
  "title": "TEST-API_1",
  "condition": "B",
  "data": [
    {
      "refId": "A",
      "queryType": "",
      "relativeTimeRange": {
        "from": 600,
        "to": 0
      },
```

Copy

Expand code

Contact points

Method	URI	Name	Summary
DELETE	/api/v1/provisioning/contact-points/:uid	route delete contactpoints	Delete a contact point.
GET	/api/v1/provisioning/contact-points	route get contactpoints	Get all the contact points.
POST	/api/v1/provisioning/contact-points	route post contactpoints	Create a contact point.
PUT	/api/v1/provisioning/contact-points/:uid	route put contactpoint	Update an existing contact point.
GET	/api/v1/provisioning/contact-points/export	route get contactpoints export	Export all contact points in provisioning file format.

Example Request for all the contact points:

```http
GET /api/v1/provisioning/contact-points
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```http
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "uid": "",
    "name": "email receiver",
    "type": "email",
    "settings": {
      "addresses": "<example@email.com>"
    },
    "disableResolveMessage": false
  }
]
```

Notification policies

Method	URI	Name	Summary
DELETE	/api/v1/provisioning/policies	route reset policy tree	Clears the notification policy tree.
GET	/api/v1/provisioning/policies	route get policy tree	Get the notification policy tree.
PUT	/api/v1/provisioning/policies	route put policy tree	Sets the notification policy tree.
GET	/api/v1/provisioning/policies/export	route get policy tree export	Export the notification policy tree in provisioning file format.

Example Request for exporting the notification policy tree in YAML format:

```http
GET /api/v1/provisioning/policies/export?format=yaml
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```http
HTTP/1.1 200 OK
Content-Type: text/yaml

apiVersion: 1
policies:
    - orgId: 1
      receiver: My Contact Email Point
      group_by:
        - grafana_folder
        - alertname
      routes:
        - receiver: My Contact Email Point
          object_matchers:
            - - monitor
              - =
              - testdata
          mute_time_intervals:
            - weekends
```

Mute timings

Method	URI	Name	Summary
DELETE	/api/v1/provisioning/mute-timings/:name	route delete mute timing	Delete a mute timing.
GET	/api/v1/provisioning/mute-timings/:name	route get mute timing	Get a mute timing.
GET	/api/v1/provisioning/mute-timings	route get mute timings	Get all the mute timings.

Method	URI	Name	Summary
POST	/api/v1/provisioning/mute-timings	route post mute timing	Create a new mute timing.
PUT	/api/v1/provisioning/mute-timings/:name	route put mute timing	Replace an existing mute timing.
GET	/api/v1/provisioning/mute-timings/export	route get mute timings export	Export all mute timings in provisioning file format.
GET	/api/v1/provisioning/mute-timings/:name/export	route get mute timing export	Export a mute timing in provisioning file format.

Example Request for all mute timings:

http Copy

```
GET /api/v1/provisioning/mute-timings
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

http Copy

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "name": "weekends",
    "time_intervals": [
      {
        "weekdays": [
          "saturday",
          "sunday"
        ]
      }
    ],
    "version": "",
    "provenance": "file"
  }
]
```

Templates

Method	URI	Name	Summary
DELETE	/api/v1/provisioning/templates/:name	route delete template	Delete a template.
GET	/api/v1/provisioning/templates/:name	route get template	Get a notification template.
GET	/api/v1/provisioning/templates	route get templates	Get all notification templates.
PUT	/api/v1/provisioning/templates/:name	route put template	Create or update a notification template.

Example Request for all notification templates:

http Copy

```
GET /api/v1/provisioning/templates
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

http Copy

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "name": "custom_email.message",
    "template": "{{ define \"custom_email.message\" }}\n  Custom alert!\n{{ end }}",
    "provenance": "file"
  },
  {
    "name": "custom_email.subject",
    "template": "{{ define \"custom_email.subject\" }}\n{{ len .Alerts.Firing }} firing alert(s),
    "provenance": "file"
  }
]
```

Edit resources in the Grafana UI

By default, you cannot edit API-provisioned alerting resources in Grafana. To enable editing these resources in the Grafana UI, add the `X-Disable-Provenance` header to the following requests in the API:

- `POST /api/v1/provisioning/alert-rules`
- `PUT /api/v1/provisioning/folder/{FolderUID}/rule-groups/{Group}` (calling this endpoint will change provenance for all alert rules within the alert group)
- `POST /api/v1/provisioning/contact-points`
- `POST /api/v1/provisioning/mute-timings`
- `PUT /api/v1/provisioning/policies`
- `PUT /api/v1/provisioning/templates/{name}`

To reset the notification policy tree to the default and unlock it for editing in the Grafana UI, use the `DELETE /api/v1/provisioning/policies` endpoint.

Data source-managed resources

The Alerting Provisioning HTTP API can only be used to manage Grafana-managed alert resources. To manage resources related to data source-managed alerts, consider the following tools:

- mimirtool: to interact with the Mimir alertmanager and ruler configuration.
- cortex-tools: to interact with the Cortex alertmanager and ruler configuration.
- lokitool: to configure the Loki Ruler.

Alternatively, the Grafana Alerting API can be used to access data from data source-managed alerts. This API is primarily intended for internal usage, with the exception of the `/api/v1/provisioning/` endpoints. It's important to note that internal APIs may undergo changes without prior notice and are not officially supported for user consumption.

For Prometheus, `amtool` can also be used to interact with the AlertManager API.

Paths

Delete a specific alert rule by UID. (*RouteDeleteAlertRule*)

```
DELETE /api/v1/provisioning/alert-rules/:uid
```

Copy

Parameters

Name	Source	Type	Go type	Separator	Required	Default	Descriptic
UID	`path`	string	`string`		✓		Alert rule UID
X-Disable-Provenance: true	`header`	string	`string`				Allows editing of provisione resources in the Grafana U

All responses

Code	Status	Description	Has headers	Schema
204	No Content	The alert rule was deleted successfully.		schema

Responses

204 - The alert rule was deleted successfully.

Status: No Content

Schema

Delete a contact point. (*RouteDeleteContactpoints*)

```
DELETE /api/v1/provisioning/contact-points/:uid
```

Copy

Parameters

Name	Source	Type	Go type	Separator	Required	Default	Description
UID	`path`	string	`string`		✓		UID is the contact point unique identifier

All responses

Code	Status	Description	Has headers	Schema
204	No Content	The contact point was deleted successfully.		schema

Responses

204 - The contact point was deleted successfully.

Status: No Content

Schema

Delete a mute timing. (*RouteDeleteMuteTiming*)

```
DELETE /api/v1/provisioning/mute-timings/:name
```

Copy

Parameters

Name	Source	Type	Go type	Separator	Required	Default	Description
name	`path`	string	`string`		✓		Mute timing name
version	`query`	string	`string`				Current version of the resource. Used for optimistic concurrency validation. Keep empty to bypass validation.

All responses

Code	Status	Description	Has headers	Schema
204	No Content	The mute timing was deleted successfully.		schema
409	Conflict	GenericPublicError		schema

Responses

204 - The mute timing was deleted successfully.

Status: No Content

Schema

409 - Conflict

Status: Conflict

Schema

GenericPublicError

Delete a template. (*RouteDeleteTemplate*)

```
DELETE /api/v1/provisioning/templates/:name
```

Copy

Parameters

Name	Source	Type	Go type	Separator	Required	Default	Description
name	`path`	string	`string`		✓		Template Name
version	`query`	string	`string`				Current version of the resource. Used for optimistic concurrency validation. Keep empty to bypass validation.

All responses

Code	Status	Description	Has headers	Schema
204	No Content	The template was deleted successfully.		schema
409	Conflict	GenericPublicError		schema

Responses

204 - The template was deleted successfully.

Status: No Content

Schema

409 - Conflict

Status: Conflict

Schema

GenericPublicError

Get a specific alert rule by UID. (*RouteGetAlertRule*)

```
GET /api/v1/provisioning/alert-rules/:uid
```

Copy

Parameters

Name	Source	Type	Go type	Separator	Required	Default	Description
UID	`path`	string	`string`		✓		Alert rule UID

All responses

Code	Status	Description	Has headers	Schema
200	OK	ProvisionedAlertRule		schema
404	Not Found	Not found.		schema

Responses

200 - ProvisionedAlertRule

Status: OK

Schema

ProvisionedAlertRule

404 - Not found.

Status: Not Found

Schema

Export an alert rule in provisioning file format. (*RouteGetAlertRuleExport*)

```
GET /api/v1/provisioning/alert-rules/:uid/export
```

Copy

Produces

- application/json
- application/yaml
- application/terraform+hcl
- text/yaml
- text/hcl

Parameters

Name	Source	Type	Go type	Separator	Required	Default	Descriptio
UID	`path`	string	`string`		✓		Alert rule UID
download	`query`	boolean	`bool`				Whether to initiate a download the file or not.
format	`query`	string	`string`			`"yaml"`	Format of the downloade file, either yaml, json or hcl. Accept header car also be used, but the query parameter will take precedenc

All responses

Code	Status	Description	Has headers	Schema
200	OK	AlertingFileExport		schema
404	Not Found	Not found.		schema

Responses

200 - AlertingFileExport

Status: OK

Schema

AlertingFileExport

404 - Not found.

Status: Not Found

Schema

Get a rule group. (*RouteGetAlertRuleGroup*)

```
GET /api/v1/provisioning/folder/:folderUid/rule-groups/:group
```

Copy

Parameters

Name	Source	Type	Go type	Separator	Required	Default	Description
FolderUID	`path`	string	`string`		✓		
Group	`path`	string	`string`		✓		

All responses

Code	Status	Description	Has headers	Schema
200	OK	AlertRuleGroup		schema

Code	Status	Description	Has headers	Schema
404	Not Found	Not found.		schema

Responses

200 - AlertRuleGroup

Status: OK
Schema

AlertRuleGroup

404 - Not found.

Status: Not Found
Schema

Export an alert rule group in provisioning file format. (*RouteGetAlertRuleGroupExport*)

```
GET /api/v1/provisioning/folder/:folderUid/rule-groups/:group/export
```

Copy

Produces

- application/json
- application/yaml
- application/terraform+hcl
- text/yaml
- text/hcl

Parameters

Name	Source	Type	Go type	Separator	Required	Default	Descriptio
FolderUID	`path`	string	`string`		✓		
Group	`path`	string	`string`		✓		
download	`query`	boolean	`bool`				Whether t initiate a download

Name	Source	Type	Go type	Separator	Required	Default	Description
							the file or not.
format	`query`	string	`string`			`"yaml"`	Format of the downloade file, either yaml, json or hcl. Accept header ca also be used, but the query parameter will take precedenc

All responses

Code	Status	Description	Has headers	Schema
200	OK	AlertingFileExport		schema
404	Not Found	Not found.		schema

Responses

200 - AlertingFileExport

Status: OK

Schema

AlertingFileExport

404 - Not found.

Status: Not Found

Schema

Get all the alert rules. (*RouteGetAlertRules*)

```
GET /api/v1/provisioning/alert-rules
```

Copy

All responses

Code	Status	Description	Has headers	Schema
200	OK	ProvisionedAlertRules		schema

Responses

200 - ProvisionedAlertRules

Status: OK

Schema

ProvisionedAlertRules

Export all alert rules in provisioning file format. (*RouteGetAlertRulesExport*)

```
GET /api/v1/provisioning/alert-rules/export
```

Copy

Produces

- application/json
- application/yaml
- application/terraform+hcl
- text/yaml
- text/hcl

Parameters

Name	Source	Type	Go type	Separator	Required	Default	Descriptio
download	`query`	boolean	`bool`				Whether to initiate a download the file or not.
format	`query`	string	`string`			`"yaml"`	Format of the downloade file, either yaml, json or hcl. Accept

Name	Source	Type	Go type	Separator	Required	Default	Descriptio
							header car also be used, but the query parameter will take precedenc

All responses

Code	Status	Description	Has headers	Schema
200	OK	AlertingFileExport		schema
404	Not Found	Not found.		schema

Responses

200 - AlertingFileExport

Status: OK

Schema

AlertingFileExport

404 - Not found.

Status: Not Found

Schema

Get all the contact points. (*RouteGetContactpoints*)

```
GET /api/v1/provisioning/contact-points
```

Copy

Parameters

Name	Source	Type	Go type	Separator	Required	Default	Description
name	`query`	string	`string`				Filter by name

All responses

Code	Status	Description	Has headers	Schema
200	OK	ContactPoints		schema

Responses

200 - ContactPoints

Status: OK

Schema

ContactPoints

Export all contact points in provisioning file format. (*RouteGetContactpointsExport*)

```
GET /api/v1/provisioning/contact-points/export
```

Copy

Produces

- application/json
- application/yaml
- application/terraform+hcl
- text/yaml
- text/hcl

Parameters

Name	Source	Type	Go type	Separator	Required	Default	Descriptio
decrypt	query	boolean	bool				Whether a contained secure settings should be decrypted left redact

Name	Source	Type	Go type	Separator	Required	Default	Descriptio
							Redacted settings wi contain RedactedV instead. Currently, org admin view decrypted secure settings.
download	`query`	boolean	`bool`				Whether to initiate a download the file or r
format	`query`	string	`string`			`"yaml"`	Format of t downloade file, either yaml, json hcl. Accep header car also be use but the qu parameter take precedenc
name	`query`	string	`string`				Filter by na

All responses

Code	Status	Description	Has headers	Schema
200	OK	AlertingFileExport		schema
403	Forbidden	PermissionDenied		schema

Responses

200 - AlertingFileExport

Status: OK

Schema

AlertingFileExport

403 - PermissionDenied

Status: Forbidden

Schema

PermissionDenied

Get a mute timing. (*RouteGetMuteTiming*)

```
GET /api/v1/provisioning/mute-timings/:name
```

Copy

Parameters

Name	Source	Type	Go type	Separator	Required	Default	Description
name	`path`	string	`string`		✓		Mute timing name

All responses

Code	Status	Description	Has headers	Schema
200	OK	MuteTimeInterval		schema
404	Not Found	Not found.		schema

Responses

200 - MuteTimeInterval

Status: OK

Schema

MuteTimeInterval

404 - Not found.

Status: Not Found

Schema

Get all the mute timings. (*RouteGetMuteTimings*)

```
GET /api/v1/provisioning/mute-timings
```

Copy

All responses

Code	Status	Description	Has headers	Schema
200	OK	MuteTimings		schema

Responses

200 - MuteTimings

Status: OK

Schema

MuteTimings

Export all mute timings in provisioning file format. (*RouteGetMuteTimingsExport*)

```
GET /api/v1/provisioning/mute-timings/export
```

Copy

Produces

- application/json
- application/yaml
- application/terraform+hcl
- text/yaml
- text/hcl

Parameters

Name	Source	Type	Go type	Separator	Required	Default	Descriptio
download	`query`	boolean	`bool`				Whether to initiate a download the file or not.
format	`query`	string	`string`			`"yaml"`	Format of the

Name	Source	Type	Go type	Separator	Required	Default	Descriptio
							downloade file, either yaml, json or hcl. Accept header car also be used, but the query parameter will take precedenc

All responses

Code	Status	Description	Has headers	Schema
200	OK	MuteTimingsExport		schema
403	Forbidden	PermissionDenied		schema

Responses

200 - MuteTimingsExport

Status: OK

Schema

AlertingFileExport

403 - PermissionDenied

Status: Forbidden

Schema

PermissionDenied

Export a mute timing in provisioning file format. (*RouteGetMuteTimingExport*)

```
GET /api/v1/provisioning/mute-timings/:name/export
```

Copy

Produces

- application/json

- application/yaml
- application/terraform+hcl
- text/yaml
- text/hcl

Parameters

Name	Source	Type	Go type	Separator	Required	Default	Descriptio
name	`path`	string	`string`		✓		Mute timin name.
download	`query`	boolean	`bool`				Whether to initiate a download the file or not.
format	`query`	string	`string`			`"yaml"`	Format of the downloade file, either yaml, json or hcl. Accept header car also be used, but the query parameter will take precedenc

All responses

Code	Status	Description	Has headers	Schema
200	OK	MuteTimingExport		schema
403	Forbidden	PermissionDenied		schema

Responses

200 - MuteTimingExport

Status: OK

Schema

AlertingFileExport

403 - PermissionDenied

Status: Forbidden

Schema

PermissionDenied

Get the notification policy tree. (*RouteGetPolicyTree*)

```
GET /api/v1/provisioning/policies
```

Copy

All responses

Code	Status	Description	Has headers	Schema
200	OK	Route		schema

Responses

200 - Route

Status: OK

Schema

Route

Export the notification policy tree in provisioning file format. (*RouteGetPolicyTreeExport*)

```
GET /api/v1/provisioning/policies/export
```

Copy

Produces

- application/json
- application/yaml
- application/terraform+hcl
- text/yaml
- text/hcl

Parameters

Name	Source	Type	Go type	Separator	Required	Default	Descriptio
download	`query`	boolean	`bool`				Whether to initiate a download the file or not.
format	`query`	string	`string`			`"yaml"`	Format of the downloade file, either yaml, json or hcl. Accept header can also be used, but the query parameter will take precedenc

All responses

Code	Status	Description	Has headers	Schema
200	OK	AlertingFileExport		schema
404	Not Found	NotFound		schema

Responses

200 - AlertingFileExport

Status: OK

Schema

AlertingFileExport

404 - NotFound

Status: Not Found

Schema

NotFound

Get a notification template. (*RouteGetTemplate*)

```
GET /api/v1/provisioning/templates/:name
```

Copy

Parameters

Name	Source	Type	Go type	Separator	Required	Default	Description
name	`path`	string	`string`		✓		Template Name

All responses

Code	Status	Description	Has headers	Schema
200	OK	NotificationTemplate		schema
404	Not Found	Not found.		schema

Responses

200 - NotificationTemplate

Status: OK

Schema

NotificationTemplate

404 - Not found.

Status: Not Found

Schema

Get all notification templates. (*RouteGetTemplates*)

```
GET /api/v1/provisioning/templates
```

Copy

All responses

Code	Status	Description	Has headers	Schema
200	OK	NotificationTemplates		schema
404	Not Found	Not found.		schema

Responses

200 - NotificationTemplates

Status: OK

Schema

NotificationTemplates

404 - Not found.

Status: Not Found

Schema

Create a new alert rule. (*RoutePostAlertRule*)

```
POST /api/v1/provisioning/alert-rules
```

Copy

Parameters

Name	Source	Type	Go type	Separator	Requ
X-Disable-Provenance: true	`header`	string	`string`		
Body	`body`	ProvisionedAlertRule	`models.ProvisionedAlertRule`		

All responses

Code	Status	Description	Has headers	Schema
201	Created	ProvisionedAlertRule		schema

Code	Status	Description	Has headers	Schema
400	Bad Request	ValidationError		schema

Responses

201 - ProvisionedAlertRule

Status: Created

Schema

ProvisionedAlertRule

400 - ValidationError

Status: Bad Request

Schema

ValidationError

Create a contact point. (*RoutePostContactpoints*)

```
POST /api/v1/provisioning/contact-points
```

Copy

Parameters

Name	Source	Type	Go type	Separator	R
X-Disable-Provenance: true	`header`	string	`string`		
Body	`body`	EmbeddedContactPoint	`models.EmbeddedContactPoint`		

All responses

Code	Status	Description	Has headers	Schema
202	Accepted	EmbeddedContactPoint		schema
400	Bad Request	ValidationError		schema

Responses

202 - EmbeddedContactPoint

Status: Accepted

Schema

EmbeddedContactPoint

400 - ValidationError

Status: Bad Request

Schema

ValidationError

Create a new mute timing. (*RoutePostMuteTiming*)

```
POST /api/v1/provisioning/mute-timings
```

Copy

Parameters

Name	Source	Type	Go type	Separator	Required
X-Disable-Provenance: true	`header`	string	`string`		
Body	`body`	MuteTimeInterval	`models.MuteTimeInterval`		

All responses

Code	Status	Description	Has headers	Schema
201	Created	MuteTimeInterval		schema
400	Bad Request	ValidationError		schema

Responses

201 - MuteTimeInterval

Status: Created

Schema

MuteTimeInterval

400 - ValidationError

Status: Bad Request

Schema

ValidationError

Update an existing alert rule. (*RoutePutAlertRule*)

```
PUT /api/v1/provisioning/alert-rules/:uid
```

Copy

Parameters

Name	Source	Type	Go type	Separator	Requ
UID	`path`	string	`string`		✓
X-Disable-Provenance: true	`header`	string	`string`		
Body	`body`	ProvisionedAlertRule	`models.ProvisionedAlertRule`		

All responses

Code	Status	Description	Has headers	Schema
200	OK	ProvisionedAlertRule		schema
400	Bad Request	ValidationError		schema

Responses

200 - ProvisionedAlertRule

Status: OK

Schema

ProvisionedAlertRule

400 - ValidationError

Status: Bad Request

Schema

ValidationError

Update the interval or alert rules of a rule group. (*RoutePutAlertRuleGroup*)

```
PUT /api/v1/provisioning/folder/:folderUid/rule-groups/:group
```

Copy

Parameters

Name	Source	Type	Go type	Separator	Required
FolderUID	`path`	string	`string`		✓
Group	`path`	string	`string`		✓
X-Disable-Provenance: true	`header`	string	`string`		
Body	`body`	AlertRuleGroup	`models.AlertRuleGroup`		

All responses

Code	Status	Description	Has headers	Schema
200	OK	AlertRuleGroup		schema
400	Bad Request	ValidationError		schema

Responses

200 - AlertRuleGroup

Status: OK

Schema

AlertRuleGroup

400 - ValidationError

Status: Bad Request

Schema

ValidationError

Update an existing contact point. (*RoutePutContactpoint*)

```
PUT /api/v1/provisioning/contact-points/:uid
```

Copy

Parameters

Name	Source	Type	Go type	Separator	R
UID	`path`	string	`string`		✓
X-Disable-Provenance: true	`header`	string	`string`		
Body	`body`	EmbeddedContactPoint	`models.EmbeddedContactPoint`		

All responses

Code	Status	Description	Has headers	Schema
202	Accepted	Ack		schema
400	Bad Request	ValidationError		schema

Responses

202 - Ack

Status: Accepted

Schema

Ack

400 - ValidationError

Status: Bad Request

Schema

ValidationError

Replace an existing mute timing. (*RoutePutMuteTiming*)

```
PUT /api/v1/provisioning/mute-timings/:name
```

Copy

Parameters

Name	Source	Type	Go type	Separator	Required
name	`path`	string	`string`		✓
X-Disable-Provenance: true	`header`	string	`string`		
Body	`body`	MuteTimeInterval	`models.MuteTimeInterval`		

All responses

Code	Status	Description	Has headers	Schema
200	OK	MuteTimeInterval		schema
400	Bad Request	ValidationError		schema
409	Conflict	GenericPublicError		schema

Responses

200 - MuteTimeInterval

Status: OK

Schema

MuteTimeInterval

400 - ValidationError

Status: Bad Request

Schema

ValidationError

409 - Conflict

Status: Conflict

Schema

GenericPublicError

Sets the notification policy tree. (*RoutePutPolicyTree*)

```
PUT /api/v1/provisioning/policies
```

Copy

Parameters

Name	Source	Type	Go type	Separator	Required	Default	Desc
X-Disable-Provenance: true	`header`	string	`string`				Allov editi prov reso in th Graf
Body	`body`	Route	`models.Route`				The notif routi to us

All responses

Code	Status	Description	Has headers	Schema
202	Accepted	Ack		schema
400	Bad Request	ValidationError		schema

Responses

202 - Ack

Status: Accepted

Schema

Ack

400 - ValidationError

Status: Bad Request

Schema

ValidationError

Create or update a notification template. (*RoutePutTemplate*)

```
PUT /api/v1/provisioning/templates/:name
```

Copy

Parameters

Name	Source	Type	Go type	Separa
name	`path`	string	`string`	
X-Disable-Provenance: true	`header`	string	`string`	
Body	`body`	NotificationTemplateContent	`models.NotificationTemplateContent`	

All responses

Code	Status	Description	Has headers	Schema
202	Accepted	NotificationTemplate		schema
400	Bad Request	ValidationError		schema
409	Conflict	GenericPublicError		schema

Responses

202 - NotificationTemplate

Status: Accepted

Schema

NotificationTemplate

400 - ValidationError

Status: Bad Request

Schema

ValidationError

409 - Conflict

Status: Conflict

Schema

GenericPublicError

Clears the notification policy tree. (*RouteResetPolicyTree*)

```
DELETE /api/v1/provisioning/policies
```

Copy

All responses

Code	Status	Description	Has headers	Schema
202	Accepted	Ack		schema

Responses

202 - Ack

Status: Accepted

Schema

Ack

Models

Ack

interface{}

AlertQuery

Properties

Name	Type	Go type	Required	Default	Descriptio
datasourceUid	string	`string`			Grafana data sourc unique identifier; i should be '**expr**' for a Server Sid Expressior operation.
model	interface{}	`interface{}`			JSON is th raw JSON query and includes the above properties as well as custom properties
queryType	string	`string`			QueryType is an optional identifier for the typ of query.
It can be used to distinguish different types of queries.					
refId	string	`string`			RefID is th unique identifier c the query, set by the frontend call.
relativeTimeRange	RelativeTimeRange	`RelativeTimeRange`			

AlertQueryExport

Properties

Name	Type	Go type	Required	Default	Descriptic
datasourceUid	string	`string`			
model	interface{}	`interface{}`			
queryType	string	`string`			
refId	string	`string`			
relativeTimeRange	RelativeTimeRange	`RelativeTimeRange`			

AlertRuleExport

Properties

Name	Type	Go type	Required	Default	Description
annotations	map of string	`map[string]string`			
condition	string	`string`			
dashboardUid	string	`string`			
data	[]AlertQueryExport	`[]*AlertQueryExport`			
execErrState	string	`string`			
for	Duration	`Duration`			
isPaused	boolean	`bool`			
labels	map of string	`map[string]string`			
noDataState	string	`string`			
panelId	int64 (formatted integer)	`int64`			
title	string	`string`			
uid	string	`string`			

AlertRuleGroup

Properties

Name	Type	Go type	Required	Default	Descriptio
folderUid	string	`string`			
interval	int64 (formatted integer)	`int64`			
rules	[]ProvisionedAlertRule	`[]*ProvisionedAlertRule`			
title	string	`string`			

AlertRuleGroupExport

Properties

Name	Type	Go type	Required	Default	Description	Exa
folder	string	`string`				
interval	Duration	`Duration`				
name	string	`string`				
orgId	int64 (formatted integer)	`int64`				
rules	[]AlertRuleExport	`[]*AlertRuleExport`				

AlertingFileExport

Properties

Name	Type	Go type	Required	Default
apiVersion	int64 (formatted integer)	`int64`		
contactPoints	[]ContactPointExport	`[]*ContactPointExport`		
groups	[]AlertRuleGroupExport	`[]*AlertRuleGroupExport`		
policies	[]NotificationPolicyExport	`[]*NotificationPolicyExport`		

ContactPointExport

Properties

Name	Type	Go type	Required	Default	Description	Exa
name	string	`string`				
orgId	int64 (formatted integer)	`int64`				
receivers	[]ReceiverExport	`[]*ReceiverExport`				

ContactPoints

[]EmbeddedContactPoint

Duration

Name	Type	Go type	Default	Description	Example
Duration	string	int64			

EmbeddedContactPoint

EmbeddedContactPoint is the contact point type that is used by grafanas embedded alertmanager implementation.

Properties

Name	Type	Go type	Required	Default	Description
disableResolveMessage	boolean	`bool`			
name	string	`string`			Name is used as grouping key in the UI. Contact points with the same name will be grouped in the UI.
webhook_1					

Name	Type	Go type	Required	Default	Description
provenance	string	`string`			
settings	JSON	`JSON`	✓		
type	string	`string`	✓		
uid	string	`string`			UID is the unique identifier of the contact point. The UID can be
set by the user.	`my_external_reference`				

Json

interface{}

MatchRegexps

MatchRegexps

MatchType

Name	Type	Go type	Default	Description	Example
MatchType	int64 (formatted integer)	int64			

Matcher

Properties

Name	Type	Go type	Required	Default	Description	Example
Name	string	`string`				
Type	MatchType	`MatchType`				
Value	string	`string`				

Matchers

Matchers is a slice of Matchers that is sortable, implements Stringer, and provides a Matches method to match a LabelSet against all Matchers in the slice. Note that some users of Matchers might require it to be sorted.

[]Matcher

MuteTimeInterval

Properties

Name	Type	Go type	Required	Default	Description	Exa
name	string	`string`				
time_intervals	[]TimeInterval	`[]*TimeInterval`				
version	string	`string`			Version of resource	

MuteTimingExport

Properties

MuteTimingsExport

Properties

MuteTimings

[]MuteTimeInterval

NotFound

interface{}

NotificationPolicyExport

Properties

Name	Type	Go type	Required	Default	Description	Example
Policy	RouteExport	`RouteExport`			inline	

Name	Type	Go type	Required	Default	Description	Example
orgId	int64 (formatted integer)	`int64`				

NotificationTemplate

Properties

Name	Type	Go type	Required	Default	Description	Example
name	string	`string`				
provenance	Provenance	`Provenance`				
template	string	`string`				
version	string	`string`			Version of resource	

NotificationTemplateContent

Properties

Name	Type	Go type	Required	Default	Description	Example
template	string	`string`				
version	string	`string`			Version of resource. Should be empty for new templates.	

NotificationTemplates

[]NotificationTemplate

ObjectMatchers

Matchers

Inlined models

PermissionDenied

interface{}

Provenance

Name	Type	Go type	Default	Description	Example
Provenance	string	string			

ProvisionedAlertRule

Properties

Name	Type	Go type	Required	Default	Description	Exar
annotations	map of string	`map[string]string`				{"ru
condition	string	`string`	✓			A
data	[]AlertQuery	`[]*AlertQuery`	✓			[{"d {"par {"typ 1","h {"fro
execErrState	string	`string`	✓			
folderUID	string	`string`	✓			proj
for	Duration	`Duration`	✓			
id	int64 (formatted integer)	`int64`				
isPaused	boolean	`bool`				fals
labels	map of string	`map[string]string`				{"te
noDataState	string	`string`	✓			
orgID	int64 (formatted integer)	`int64`	✓			
provenance	Provenance	`Provenance`				

Name	Type	Go type	Required	Default	Description	Exar
ruleGroup	string	`string`	✓			`eval`
title	string	`string`	✓			`Alwa`
uid	string	`string`				
updated	date-time (formatted string)	`strfmt.DateTime`				

ProvisionedAlertRules

[]ProvisionedAlertRule

RawMessage

interface{}

ReceiverExport

Properties

Name	Type	Go type	Required	Default	Description
disableResolveMessage	boolean	`bool`			
settings	RawMessage	`RawMessage`			
type	string	`string`			
uid	string	`string`			

Regexp

A Regexp is safe for concurrent use by multiple goroutines, except for configuration methods, such as Longest.

interface{}

RelativeTimeRange

RelativeTimeRange is the per query start and end time for requests.

Properties

Name	Type	Go type	Required	Default	Description	Example
from	Duration	`Duration`				
to	Duration	`Duration`				

Route

A Route is a node that contains definitions of how to handle alerts. This is modified from the upstream alertmanager in that it adds the ObjectMatchers property.

Properties

Name	Type	Go type	Required	Default	Description
continue	boolean	`bool`			
group_by	[]string	`[]string`			
group_interval	string	`string`			
group_wait	string	`string`			
match	map of string	`map[string]string`			Deprecated Remove before v1.0 release.
match_re	MatchRegexps	`MatchRegexps`			
matchers	Matchers	`Matchers`			
mute_time_intervals	[]string	`[]string`			
object_matchers	ObjectMatchers	`ObjectMatchers`			
provenance	Provenance	`Provenance`			
receiver	string	`string`			
repeat_interval	string	`string`			

Name	Type	Go type	Required	Default	Description
routes	[]Route	`[]*Route`			

RouteExport

RouteExport is the provisioned file export of definitions.Route. This is needed to hide fields that aren't usable in provisioning file format. An alternative would be to define a custom MarshalJSON and MarshalYAML that excludes them.

Properties

Name	Type	Go type	Required	Default	Description
continue	boolean	`bool`			
group_by	[]string	`[]string`			
group_interval	string	`string`			
group_wait	string	`string`			
match	map of string	`map[string]string`			Deprecated Remove before v1.0 release.
match_re	MatchRegexps	`MatchRegexps`			
matchers	Matchers	`Matchers`			
mute_time_intervals	[]string	`[]string`			
object_matchers	ObjectMatchers	`ObjectMatchers`			
receiver	string	`string`			
repeat_interval	string	`string`			
routes	[]RouteExport	`[]*RouteExport`			

TimeInterval

TimeInterval describes intervals of time. ContainsTime will tell you if a golang time is contained within the interval.

Properties

Name	Type	Go type	Required	Default	Description	Examp
days_of_month	[]string	`[]string`				
location	string	`string`				
months	[]string	`[]string`				
times	[]TimeRange	`[]*TimeRange`				
weekdays	[]string	`[]string`				
years	[]string	`[]string`				

TimeRange

For example, 4:00PM to End of the day would Begin at 1020 and End at 1440.

Properties

Name	Type	Go type	Required	Default	Description	Example
end_time	string	`string`				`"end_time": "24:00"`
start_time	string	`string`				`"start_time": "18:00"`

ValidationError

Properties

Name	Type	Go type	Required	Default	Description	Example
msg	string	`string`				`error message`

GenericPublicError

Properties

Name	Type	Go type	Required	Default	Description	Example
statusCode	string	`string`	✓		HTTP Status Code	
messageId	string	`string`	✓		Unique code of the error	
message	string	`string`			Error message	
extra	map of any	`map[string]any`			Extra information about the error. Format is specific to the error code.	

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Annotations API

Annotations are saved in the Grafana database (sqlite, mysql or postgres). Annotations can be organization annotations that can be shown on any dashboard by configuring an annotation data source - they are filtered by tags. Or they can be tied to a panel on a dashboard and are then only shown on that panel.

If you are running Grafana Enterprise, for some endpoints you'll need to have specific permissions. Refer to Role-based access control permissions for more information.

Find Annotations

`GET /api/annotations?from=1506676478816&to=1507281278816&tags=tag1&tags=tag2&limit=100`

Required permissions

See note in the introduction for an explanation.

Action	Scope
annotations:read	annotations:type:

Example Request:

http Copy

```
GET /api/annotations?from=1506676478816&to=1507281278816&tags=tag1&tags=tag2&limit=100 HTTP/1.1
Accept: application/json
```

```
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=
```

Query Parameters:

- `from`: epoch datetime in milliseconds. Optional.
- `to`: epoch datetime in milliseconds. Optional.
- `limit`: number. Optional - default is 100. Max limit for results returned.
- `alertId`: number. Optional. Find annotations for a specified alert.
- `dashboardId`: number. Optional. Find annotations that are scoped to a specific dashboard
- `dashboardUID`: string. Optional. Find annotations that are scoped to a specific dashboard, when dashboardUID presents, dashboardId would be ignored.
- `panelId`: number. Optional. Find annotations that are scoped to a specific panel
- `userId`: number. Optional. Find annotations created by a specific user
- `type`: string. Optional. `alert` | `annotation` Return alerts or user created annotations
- `tags`: string. Optional. Use this to filter organization annotations. Organization annotations are annotations from an annotation data source that are not connected specifically to a dashboard or panel. To do an "AND" filtering with multiple tags, specify the tags parameter multiple times e.g. `tags=tag1&tags=tag2`.

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json
[
    {
        "id": 1124,
        "alertId": 0,
        "dashboardId": 468,
        "dashboardUID": "uGlb_lG7z",
        "panelId": 2,
        "userId": 1,
        "userName": "",
        "newState": "",
        "prevState": "",
        "time": 1507266395000,
        "timeEnd": 1507266395000,
        "text": "test",
        "metric": "",
        "tags": [
            "tag1",
```

Expand code

Starting in Grafana v6.4 regions annotations are now returned in one entity that now includes the timeEnd property.

Create Annotation

Creates an annotation in the Grafana database. The `dashboardId` and `panelId` fields are optional. If they are not specified then an organization annotation is created and can be queried in any dashboard that adds the Grafana annotations data source. When creating a region annotation include the timeEnd property.

The format for `time` and `timeEnd` should be epoch numbers in millisecond resolution.

`POST /api/annotations`

Required permissions

See note in the introduction for an explanation.

Action	Scope
annotations:create	annotations:type:

Required JSON Body Fields

- `text`: description of the annotation.

Example Request:

http

Copy

```
POST /api/annotations HTTP/1.1
Accept: application/json
Content-Type: application/json

{
  "dashboardUID":"jcIIG-07z",
  "panelId":1,
  "time":1507037197339,
  "timeEnd":1507180805056,
  "tags":["tag1","tag2"],
  "text":"Annotation Description"
}
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{
    "message":"Annotation added",
    "id": 1,
}
```

The response for this HTTP request is slightly different in versions prior to v6.4. In prior versions you would also get an endId if you where creating a region. But in 6.4 regions are represented using a single event with time and timeEnd properties.

Create Annotation in Graphite format

Creates an annotation by using Graphite-compatible event format. The `when` and `data` fields are optional. If `when` is not specified then the current time will be used as annotation's timestamp. The `tags` field can also be in prior to Graphite `0.10.0` format (string with multiple tags being separated by a space).

`POST /api/annotations/graphite`

Required permissions

See note in the introduction for an explanation.

Action	Scope
annotations:create	annotations:type:organization

Example Request:

```http
POST /api/annotations/graphite HTTP/1.1
Accept: application/json
Content-Type: application/json

{
```

```
  "what": "Event - deploy",
  "tags": ["deploy", "production"],
  "when": 1467844481,
  "data": "deploy of master branch happened at Wed Jul 6 22:34:41 UTC 2016"
}
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{
    "message":"Graphite annotation added",
    "id": 1
}
```

Update Annotation

`PUT /api/annotations/:id`

Updates all properties of an annotation that matches the specified id. To only update certain property, consider using the Patch Annotation operation.

Required permissions

See note in the introduction for an explanation.

Action	Scope
annotations:write	annotations:type:

Example Request:

http Copy

```
PUT /api/annotations/1141 HTTP/1.1
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
Content-Type: application/json

{
  "time":1507037197339,
```

```
  "timeEnd":1507180805056,
  "text":"Annotation Description",
  "tags":["tag3","tag4","tag5"]
}
```

Example Response:

```
http
```

Copy

```
HTTP/1.1 200
Content-Type: application/json

{
    "message":"Annotation updated"
}
```

Patch Annotation

This is available in Grafana 6.0.0-beta2 and above.

`PATCH /api/annotations/:id`

Updates one or more properties of an annotation that matches the specified id.

This operation currently supports updating of the `text`, `tags`, `time` and `timeEnd` properties.

Required permissions

See note in the introduction for an explanation.

Action	Scope
annotations:write	annotations:type:

Example Request:

```
http
```

Copy

```
PATCH /api/annotations/1145 HTTP/1.1
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
Content-Type: application/json
```

```
{
  "text":"New Annotation Description",
  "tags":["tag6","tag7","tag8"]
}
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{
    "message":"Annotation patched"
}
```

Delete Annotation By Id

`DELETE /api/annotations/:id`

Deletes the annotation that matches the specified id.

Required permissions

See note in the introduction for an explanation.

Action	Scope
annotations:delete	annotations:type:

Example Request:

```http
DELETE /api/annotations/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```http
```

```
HTTP/1.1 200
Content-Type: application/json

{
    "message":"Annotation deleted"
}
```

Find Annotations Tags

`GET /api/annotations/tags`

Find all the event tags created in the annotations.

Required permissions

See note in the introduction for an explanation.

Action	Scope
annotations:read	N/A

Example Request:

http Copy

```
GET /api/annotations/tags?tag=out HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=
```

Query Parameters:

- `tag`: Optional. A string that you can use to filter tags.
- `limit`: Optional. A number, where the default is 100. Max limit for results returned.

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{
    "result": {
```

```
        "tags": [
            {
                "tag": "outage",
                "count": 1
            }
        ]
    }
}
```

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Enterprise Open source

Authentication API

The Authentication HTTP API is used to manage API keys.

NOTE

If you use Grafana v9.1 or newer, use service accounts instead of API keys. For more information, refer to Grafana service account API reference.

If you are running Grafana Enterprise, for some endpoints you would need to have relevant permissions. Refer to Role-based access control permissions for more information.

List API keys

`GET /api/auth/keys`

Required permissions

See note in the introduction for an explanation.

Action	Scope
`apikeys:read`	`apikeys:*`

Example Request:

```http
```

Copy

```
GET /api/auth/keys HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Query Parameters:

- `includeExpired`: boolean. enable listing of expired keys. Optional.

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

[
  {
    "id": 3,
    "name": "API",
    "role": "Admin"
  },
  {
    "id": 1,
    "name": "TestAdmin",
    "role": "Admin",
    "expiration": "2019-06-26T10:52:03+03:00"
  }
]
```

Create API Key

`POST /api/auth/keys`

Required permissions

See note in the introduction for an explanation.

Action	Scope
`apikeys:create`	n/a

Example Request:

http Copy

```
POST /api/auth/keys HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "name": "mykey",
  "role": "Admin",
  "secondsToLive": 86400
}
```

JSON Body schema:

- **name** – The key name
- **role** – Sets the access level/Grafana Role for the key. Can be one of the following values: `Viewer`, `Editor` or `Admin`.
- **secondsToLive** – Sets the key expiration in seconds. It is optional. If it is a positive number an expiration date for the key is set. If it is null, zero or is omitted completely (unless `api_key_max_seconds_to_live` configuration option is set) the key will never expire.

Error statuses:

- **400** – `api_key_max_seconds_to_live` is set but no `secondsToLive` is specified or `secondsToLive` is greater than this value.
- **500** – The key was unable to be stored in the database.

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{"name":"mykey","key":"eyJrIjoiWHZiSWd3NzdCYUZnNUtibE9obUpESmE3bzJYNDRIc0UiLCJuIjoibXlrZXkiLCJpZCI
```

Delete API Key

`DELETE /api/auth/keys/:id`

Required permissions

See note in the introduction for an explanation.

Action	Scope
`apikeys:delete`	apikeys:*

Example Request:

http

```http
DELETE /api/auth/keys/3 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

http

```http
HTTP/1.1 200
Content-Type: application/json

{"message":"API key deleted"}
```

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Correlations API

This API can be used to define correlations between data sources.

Create correlations

`POST /api/datasources/uid/:sourceUID/correlations`

Creates a correlation between two data sources - the source data source identified by `sourceUID` in the path, and the target data source which is specified in the body.

Example request:

http Copy

```
POST /api/datasources/uid/uyBf2637k/correlations HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
{
    "targetUID": "PDDA8E780A17E7EF1",
    "label": "My Label",
    "description": "Logs to Traces",
  "config": {
    "type": "query",
    "field": "message",
    "target": {},
  }
}
```

JSON body schema:

- **targetUID** – Target data source uid.
- **label** – A label for the correlation.
- **description** – A description for the correlation.

Example response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json
{
  "message": "Correlation created",
  "result": {
    "description": "Logs to Traces",
    "label": "My Label",
    "sourceUID": "uyBf2637k",
    "targetUID": "PDDA8E780A17E7EF1",
    "uid": "50xhMlg9k",
    "config": {
      "type": "query",
      "field": "message",
      "target": {},
    }
  }
}
```

Status codes:

- **200** – OK
- **400** - Errors (invalid JSON, missing or invalid fields)
- **401** – Unauthorized
- **403** – Forbidden, source data source is read-only
- **404** – Not found, either source or target data source could not be found
- **500** – Internal error

Delete correlations

`DELETE /api/datasources/uid/:sourceUID/correlations/:correlationUID`

Deletes a correlation.

Example request:

```http
DELETE /api/datasources/uid/uyBf2637k/correlations/J6gn7d31L HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response:

```http
HTTP/1.1 200
Content-Type: application/json
{
  "message": "Correlation deleted"
}
```

Status codes:

- **200** – OK
- **401** – Unauthorized
- **403** – Forbidden, data source is read-only
- **404** – Correlation not found
- **500** – Internal error

Update correlations

`PATCH /api/datasources/uid/:sourceUID/correlations/:correlationUID`

Updates a correlation.

Example request:

```http
POST /api/datasources/uid/uyBf2637k/correlations/J6gn7d31L HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
{
    "label": "My Label",
    "description": "Logs to Traces",
}
```

JSON body schema:

- **label** – A label for the correlation.
- **description** – A description for the correlation.

Example response:

```http
HTTP/1.1 200
Content-Type: application/json
{
  "message": "Correlation updated",
  "result": {
    "description": "Logs to Traces",
    "label": "My Label",
    "sourceUID": "uyBf2637k",
    "targetUID": "PDDA8E780A17E7EF1",
    "uid": "J6gn7d31L",
    "config": {
            "type": "query",
            "field": "message",
            "target": {}
        }
  }
}
```

Status codes:

- **200** – OK
- **400** – Bad request
- **401** – Unauthorized
- **403** – Forbidden, source data source is read-only
- **404** – Not found, either source or target data source could not be found
- **500** – Internal error

Get single correlation

`GET /api/datasources/uid/:sourceUID/correlations/:correlationUID`

Gets a single correlation.

Example request:

http Copy

```
GET /api/datasources/uid/uyBf2637k/correlations/J6gn7d31L HTTP/1.1
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json
{
  "description": "Logs to Traces",
  "label": "My Label",
  "sourceUID": "uyBf2637k",
  "targetUID": "PDDA8E780A17E7EF1",
  "uid": "J6gn7d31L",
  "provisioned": false,
  "config": {
    "type": "query",
    "field": "message",
    "target": {},
  }
}
```

Status codes:

- **200** – OK
- **401** – Unauthorized
- **404** – Not found, either source data source or correlation were not found
- **500** – Internal error

Get all correlations originating from a given data source

`GET /api/datasources/uid/:sourceUID/correlations`

Get all correlations originating from the data source identified by the given `sourceUID` in the path.

Example request:

http Copy

```
```

```
GET /api/datasources/uid/uyBf2637k/correlations HTTP/1.1
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json
[
  {
    "description": "Logs to Traces",
    "label": "My Label",
    "sourceUID": "uyBf2637k",
    "targetUID": "PDDA8E780A17E7EF1",
    "uid": "J6gn7d31L",
    "provisioned": false,
    "config": {
      "type": "query",
      "field": "message",
      "target": {},
    }
  },
  {
    "description": "Logs to Metrics",
    "label": "Another Label",
```

Expand code

Status codes:

- **200** – OK
- **401** – Unauthorized
- **404** – Not found, either source data source is not found or no correlation exists originating from the given data source
- **500** – Internal error

Get all correlations

`GET /api/datasources/correlations`

Get all correlations.

Query parameters:

- **page** - Optional. Specify which page number to return. Use the limit parameter to specify the number of correlations per page. The default is page 1.
- **limit** - Optional. Limits the number of returned correlations per page. The default is 100 correlations per page. The maximum limit is 1000 correlations in a page.
- **sourceUID** - Optional. Specify a source datasource UID to filter by. This can be repeated to filter by multiple datasources.

Example request:

```http
GET /api/datasources/correlations HTTP/1.1
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response:

```http
HTTP/1.1 200
Content-Type: application/json
[
  {
    "description": "Prometheus to Loki",
    "label": "My Label",
    "sourceUID": "uyBf2637k",
    "targetUID": "PDDA8E780A17E7EF1",
    "uid": "J6gn7d31L",
    "provisioned": false,
    "config": {
      "type": "query",
      "field": "message",
      "target": {},
    }
  },
  {
    "description": "Loki to Tempo",
    "label": "Another Label",
```

Status codes:

- **200** – OK
- **401** – Unauthorized
- **404** – Not found, no correlation is found
- **500** – Internal error

cURL examples

This page provides examples of calls to the Grafana API using cURL.

The most basic example for a dashboard for which there is no authentication. You can test the following on your local machine, assuming a default installation and anonymous access enabled, required:

```
curl http://localhost:3000/api/search
```

Copy

Here's a cURL command that works for getting the home dashboard when you are running Grafana locally with basic authentication enabled using the default admin credentials:

```
curl http://admin:admin@localhost:3000/api/search
```

Copy

To pass a username and password with HTTP basic authorization, encode them as base64. You can't use authorization tokens in the request.

For example, to list permissions associated with roles given a username of `user` and password of `password`, use:

```
curl --location '<grafana_url>/api/access-control/builtin-roles' --user 'user:passwc
```

Copy

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Enterprise Open source

Dashboard API

If you are running Grafana Enterprise, for some endpoints you'll need to have specific permissions. Refer to Role-based access control permissions for more information.

Identifier (id) vs unique identifier (uid)

The identifier (id) of a dashboard is an auto-incrementing numeric value and is only unique per Grafana install.

The unique identifier (uid) of a dashboard can be used for uniquely identify a dashboard between multiple Grafana installs. It's automatically generated if not provided when creating a dashboard. The uid allows having consistent URLs for accessing dashboards and when syncing dashboards between multiple Grafana installs, see dashboard provisioning for more information. This means that changing the title of a dashboard will not break any bookmarked links to that dashboard.

The uid can have a maximum length of 40 characters.

Create / Update dashboard

`POST /api/dashboards/db`

Creates a new dashboard or updates an existing dashboard. When updating existing dashboards, if you do not define the `folderId` or the `folderUid` property, then the dashboard(s) are moved to the root level. (You need to define only one property, not both).

Note: This endpoint is not intended for creating folders, use `POST /api/folders` for that.

Required permissions

See note in the introduction for an explanation.

Action	Scope
`dashboards:create`	`folders:*`

Example Request for new dashboard:

http

Copy

```
POST /api/dashboards/db HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "dashboard": {
    "id": null,
    "uid": null,
    "title": "Production Overview",
    "tags": [ "templated" ],
    "timezone": "browser",
    "schemaVersion": 16,
    "refresh": "25s"
  },
  "folderUid": "l3KqBxCMz",
  "message": "Made changes to xyz",
  "overwrite": false
}
```

Expand code

JSON Body schema:

- **dashboard** – The complete dashboard model.
- **dashboard.id** – id = null to create a new dashboard.
- **dashboard.uid** – Optional unique identifier when creating a dashboard. uid = null will generate a new uid.
- **dashboard.refresh** - Set the dashboard refresh interval. If this is lower than the minimum refresh interval, then Grafana will ignore it and will enforce the minimum refresh interval.
- **folderId** – The id of the folder to save the dashboard in.
- **folderUid** – The UID of the folder to save the dashboard in. Overrides the `folderId`.

- **overwrite** – Set to true if you want to overwrite existing dashboard with newer version, same dashboard title in folder or same dashboard uid.
- **message** - Set a commit message for the version history.

Example Request for updating a dashboard:

http Copy

```
POST /api/dashboards/db HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "dashboard": {
    "id": 1,
    "title": "Production Overview Updated",
    "tags": [ "templated" ],
    "timezone": "browser",
    "schemaVersion": 16,
    "version": 1,
    "refresh": "25s"
  },
  "folderUid": "l3KqBxCMz",
  "message": "Made changes to xyz",
  "overwrite": false
}
```

Expand code

Example Response:

http Copy

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 78

{
    "id": 1,
    "uid": "e883f11b-77c0-4ee3-9a70-3ba223d66e56",
    "url": "/d/e883f11b-77c0-4ee3-9a70-3ba223d66e56/production-overview-updated",
    "status": "success",
    "version": 2
    "slug": "production-overview-updated",
}
```

Status Codes:

- **200** – Created
- **400** – Errors (invalid json, missing or invalid fields, etc)
- **401** – Unauthorized
- **403** – Access denied
- **412** – Precondition failed

The **412** status code is used for explaining that you cannot create the dashboard and why. There can be different reasons for this:

- The dashboard has been changed by someone else, `status=version-mismatch`
- A dashboard with the same name in the folder already exists, `status=name-exists`
- A dashboard with the same uid already exists, `status=name-exists`
- The dashboard belongs to plugin `<plugin title>`, `status=plugin-dashboard`

The response body will have the following properties:

http Copy

```
HTTP/1.1 412 Precondition Failed
Content-Type: application/json; charset=UTF-8
Content-Length: 97

{
  "message": "The dashboard has been changed by someone else",
  "status": "version-mismatch"
}
```

In case of title already exists the `status` property will be `name-exists`.

Get dashboard by uid

`GET /api/dashboards/uid/:uid`

Will return the dashboard given the dashboard unique identifier (uid). Information about the unique identifier of a folder containing the requested dashboard might be found in the metadata.

Required permissions

See note in the introduction for an explanation.

Action	Scope
`dashboards:read`	`dashboards:*`

Example Request:

```http
GET /api/dashboards/uid/cIBgcSjkk HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{
  "dashboard": {
    "id": 1,
    "uid": "cIBgcSjkk",
    "title": "Production Overview",
    "tags": [ "templated" ],
    "timezone": "browser",
    "schemaVersion": 16,
    "version": 0
  },
  "meta": {
    "isStarred": false,
    "url": "/d/cIBgcSjkk/production-overview",
    "folderId": 2,
    "folderUid": "l3KqBxCMz",
    "slug": "production-overview" //deprecated in Grafana v5.0
```

Status Codes:

- **200** – Found
- **401** – Unauthorized
- **403** – Access denied
- **404** – Not found

Delete dashboard by uid

`DELETE /api/dashboards/uid/:uid`

Will delete the dashboard given the specified unique identifier (uid).

Required permissions

See note in the introduction for an explanation.

Action	Scope
`dashboards:delete`	`dashboards:*` `folders:*`

Example Request:

http Copy

```
DELETE /api/dashboards/uid/cIBgcSjkk HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{
  "title": "Production Overview",
  "message": "Dashboard Production Overview deleted",
  "id": 2
}
```

Status Codes:

- **200** – Deleted
- **401** – Unauthorized
- **403** – Access denied
- **404** – Not found

Gets the home dashboard

`GET /api/dashboards/home`

Will return the home dashboard.

Example Request:

```http
GET /api/dashboards/home HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{
  "dashboard": {
    "editable":false,
    "nav":[
      {
        "enable":false,
        "type":"timepicker"
      }
    ],
    "style":"dark",
    "tags":[],
    "templating":{
      "list":[
      ]
    },
    "time":{
```

Tags for Dashboard

`GET /api/dashboards/tags`

Get all tags of dashboards

Example Request:

```http
GET /api/dashboards/tags HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

[
  {
    "term":"tag1",
    "count":1
  },
  {
    "term":"tag2",
    "count":4
  }
]
```

Dashboard Search

See Folder/Dashboard Search API.

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Dashboard Permissions API

This API can be used to update/get the permissions for a dashboard.

Permissions with `dashboardId=-1` are the default permissions for users with the Viewer and Editor roles. Permissions can be set for a user, a team or a role (Viewer or Editor). Permissions cannot be set for Admins - they always have access to everything.

The permission levels for the permission field:

- 1 = View
- 2 = Edit
- 4 = Admin

If you are running Grafana Enterprise, for some endpoints you'll need to have specific permissions. Refer to Role-based access control permissions for more information.

Get permissions for a dashboard

`GET /api/dashboards/uid/:uid/permissions`

Gets all existing permissions for the dashboard with the given `uid`.

Required permissions

See note in the introduction for an explanation.

Action	Scope
`dashboards.permissions:read`	`dashboards:uid:*` `folders:uid:*`

Example request:

```http
GET /api/dashboards/uid/dHEquNzGz/permissions HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response

```http
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 551

[
  {
    "id": 1,
    "dashboardId": -1,
    "created": "2017-06-20T02:00:00+02:00",
    "updated": "2017-06-20T02:00:00+02:00",
    "userId": 0,
    "userLogin": "",
    "userEmail": "",
    "teamId": 0,
    "team": "",
    "role": "Viewer",
    "permission": 1,
    "permissionName": "View",
    "uid": "dHEquNzGz",
```

Status Codes:

- **200** - Ok
- **401** - Unauthorized
- **403** - Access denied
- **404** - Dashboard not found

Update permissions for a dashboard

`POST /api/dashboards/uid/:uid/permissions`

Updates permissions for a dashboard. This operation will remove existing permissions if they’re not included in the request.

Required permissions

See note in the introduction for an explanation.

Action	Scope
`dashboards.permissions:write`	`dashboards:uid:*` `folders:uid:*`

Example request:

http Copy

```
POST /api/dashboards/uid/dHEquNzGz/permissions
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "items": [
    {
      "role": "Viewer",
      "permission": 1
    },
    {
      "role": "Editor",
      "permission": 2
    },
    {
      "teamId": 1,
      "permission": 1
    },
```

Expand code

JSON body schema:

- **items** - The permission items to add/update. Items that are omitted from the list will be removed.

Example response:

```http
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 35

{"message":"Dashboard permissions updated"}
```

Status Codes:

- **200** - Ok
- **401** - Unauthorized
- **403** - Access denied
- **404** - Dashboard not found

Get permissions for a dashboard by id

> **WARNING**
>
> This API is deprecated since Grafana v9.0.0 and will be removed in a future release. Refer to the new dashboard permissions API.

`GET /api/dashboards/id/:dashboardId/permissions`

Gets all existing permissions for the dashboard with the given `dashboardId`.

Required permissions

See note in the introduction for an explanation.

Action	Scope
`dashboards.permissions:read`	`dashboards:*` `folders:*`

Example request:

```http
GET /api/dashboards/id/1/permissions HTTP/1.1
Accept: application/json
```

```
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response

http

Copy

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 551

[
  {
    "id": 1,
    "dashboardId": -1,
    "created": "2017-06-20T02:00:00+02:00",
    "updated": "2017-06-20T02:00:00+02:00",
    "userId": 0,
    "userLogin": "",
    "userEmail": "",
    "teamId": 0,
    "team": "",
    "role": "Viewer",
    "permission": 1,
    "permissionName": "View",
    "uid": "",
```

Expand code

Status Codes:

- **200** - Ok
- **401** - Unauthorized
- **403** - Access denied
- **404** - Dashboard not found

Update permissions for a dashboard by id

WARNING

This API is deprecated since Grafana v9.0.0 and will be removed in a future release. Refer to the new dashboard permissions API.

`POST /api/dashboards/id/:dashboardId/permissions`

Updates permissions for a dashboard. This operation will remove existing permissions if they're not included in the request.

Required permissions

See note in the introduction for an explanation.

Action	Scope
`dashboards.permissions:write`	`dashboards:*` `folders:*`

Example request:

http Copy

```
POST /api/dashboards/id/1/permissions
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "items": [
    {
      "role": "Viewer",
      "permission": 1
    },
    {
      "role": "Editor",
      "permission": 2
    },
    {
      "teamId": 1,
      "permission": 1
    },
```

Expand code

JSON body schema:

- **items** - The permission items to add/update. Items that are omitted from the list will be removed.

Example response:

http Copy

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 35

{"message":"Dashboard permissions updated"}
```

Status Codes:

- **200** - Ok
- **401** - Unauthorized
- **403** - Access denied
- **404** - Dashboard not found

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Dashboard Versions

Get all dashboard versions

WARNING

This API is deprecated since Grafana v9.0.0 and will be removed in a future release. Refer to the new dashboard versions API.

Query parameters:

- **limit** - Maximum number of results to return
- **start** - Version to start from when returning queries

`GET /api/dashboards/id/:dashboardId/versions`

Gets all existing dashboard versions for the dashboard with the given `dashboardId`.

Example request for getting all dashboard versions:

http Copy

```
GET /api/dashboards/id/1/versions?limit=2?start=0 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response

http Copy

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 428

[
  {
    "id": 2,
    "dashboardId": 1,
    "parentVersion": 1,
    "restoredFrom": 0,
    "version": 2,
    "created": "2017-06-08T17:24:33-04:00",
    "createdBy": "admin",
    "message": "Updated panel title"
  },
  {
    "id": 1,
    "dashboardId": 1,
    "parentVersion": 0,
```

Expand code

Status Codes:

- **200** - Ok
- **400** - Errors
- **401** - Unauthorized
- **404** - Dashboard version not found

Get all dashboard versions by dashboard UID

Query parameters:

- **limit** - Maximum number of results to return
- **start** - Version to start from when returning queries

`GET /api/dashboards/uid/:uid/versions`

Gets all existing dashboard versions for the dashboard with the given `uid`.

Example request for getting all dashboard versions:

http Copy

```
GET /api/dashboards/uid/QA7wKklGz/versions?limit=2?start=0 HTTP/1.1
Accept: application/json
```

```
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response

http

Copy

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 428

[
  {
    "id": 2,
    "dashboardId": 1,
    "uid": "QA7wKklGz",
    "parentVersion": 1,
    "restoredFrom": 0,
    "version": 2,
    "created": "2017-06-08T17:24:33-04:00",
    "createdBy": "admin",
    "message": "Updated panel title"
  },
  {
    "id": 1,
    "dashboardId": 1,
```

Expand code

Status Codes:

- **200** - Ok
- **400** - Errors
- **401** - Unauthorized
- **404** - Dashboard version not found

Get dashboard version

> **WARNING**
>
> This API is deprecated since Grafana v9.0.0 and will be removed in a future release. Refer to the new get dashboard version API.

`GET /api/dashboards/id/:dashboardId/versions/:version`

Get the dashboard version with the given version, for the dashboard with the given id.

Example request for getting a dashboard version:

```http
GET /api/dashboards/id/1/versions/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response:

```http
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 1300

{
  "id": 1,
  "dashboardId": 1,
  "parentVersion": 0,
  "restoredFrom": 0,
  "version": 1,
  "created": "2017-04-26T17:18:38-04:00",
  "message": "Initial save",
  "data": {
    "annotations": {
      "list": [

      ]
    },
    "editable": true,
```

Expand code

Status Codes:

- **200** - Ok
- **401** - Unauthorized
- **404** - Dashboard version not found

Get dashboard version by dashboard UID

`GET /api/dashboards/uid/:uid/versions/:version`

Get the dashboard version with the given version, for the dashboard with the given UID.

Example request for getting a dashboard version:

```http
GET /api/dashboards/id/1/versions/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response:

```http
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 1300

{
  "id": 1,
  "dashboardId": 1,
  "uid": "QA7wKklGz",
  "parentVersion": 0,
  "restoredFrom": 0,
  "version": 1,
  "created": "2017-04-26T17:18:38-04:00",
  "message": "Initial save",
  "data": {
    "annotations": {
      "list": [

      ]
    },
```

Status Codes:

- **200** - Ok
- **401** - Unauthorized
- **404** - Dashboard version not found

Restore dashboard

WARNING

This API is deprecated since Grafana v9.0.0 and will be removed in a future release. Refer to the new restore dashboard API.

`POST /api/dashboards/id/:dashboardId/restore`

Restores a dashboard to a given dashboard version.

Example request for restoring a dashboard version:

```http
POST /api/dashboards/id/1/restore
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "version": 1
}
```

JSON body schema:

- **version** - The dashboard version to restore to

Example response:

```http
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 67

{
  "slug": "my-dashboard",
  "status": "success",
  "version": 3
}
```

JSON response body schema:

- **slug** - the URL friendly slug of the dashboard's title
- **status** - whether the restoration was successful or not
- **version** - the new dashboard version, following the restoration

Status codes:

- **200** - OK

- **401** - Unauthorized
- **404** - Not found (dashboard not found or dashboard version not found)
- **500** - Internal server error (indicates issue retrieving dashboard tags from database)

Example error response

```http
HTTP/1.1 404 Not Found
Content-Type: application/json; charset=UTF-8
Content-Length: 46

{
  "message": "Dashboard version not found"
}
```

JSON response body schema:

- **message** - Message explaining the reason for the request failure.

Restore dashboard by dashboard UID

`POST /api/dashboards/uid/:uid/restore`

Restores a dashboard to a given dashboard version using `uid`.

Example request for restoring a dashboard version:

```http
POST /api/dashboards/uid/QA7wKklGz/restore
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "version": 1
}
```

JSON body schema:

- **version** - The dashboard version to restore to

Example response:

```http
```

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 67

{
  "id": 70,
  "slug": "my-dashboard",
  "status": "success",
  "uid": "QA7wKklGz",
  "url": "/d/QA7wKklGz/my-dashboard",
  "version": 3
}
```

JSON response body schema:

- **slug** - the URL friendly slug of the dashboard's title
- **status** - whether the restoration was successful or not
- **version** - the new dashboard version, following the restoration

Status codes:

- **200** - OK
- **401** - Unauthorized
- **404** - Not found (dashboard not found or dashboard version not found)
- **500** - Internal server error (indicates issue retrieving dashboard tags from database)

Example error response

http Copy

```
HTTP/1.1 404 Not Found
Content-Type: application/json; charset=UTF-8
Content-Length: 46

{
  "message": "Dashboard version not found"
}
```

JSON response body schema:

- **message** - Message explaining the reason for the request failure.

Compare dashboard versions

`POST /api/dashboards/calculate-diff`

Compares two dashboard versions by calculating the JSON diff of them.

Example request:

```http
POST /api/dashboards/calculate-diff HTTP/1.1
Accept: text/html
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "base": {
    "dashboardId": 1,
    "version": 1
  },
  "new": {
    "dashboardId": 1,
    "version": 2
  },
  "diffType": "json"
}
```

JSON body schema:

- **base** - an object representing the base dashboard version
- **new** - an object representing the new dashboard version
- **diffType** - the type of diff to return. Can be “json” or “basic”.

Example response (JSON diff):

```http
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8

<p id="l1" class="diff-line diff-json-same">
  <!-- Diff omitted -->
</p>
```

The response is a textual representation of the diff, with the dashboard values being in JSON, similar to the diffs seen on sites like GitHub or GitLab.

Status Codes:

- **200** - Ok
- **400** - Bad request (invalid JSON sent)
- **401** - Unauthorized
- **404** - Not found

Example response (basic diff):

```http
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8

<div class="diff-group">
  <!-- Diff omitted -->
</div>
```

The response here is a summary of the changes, derived from the diff between the two JSON objects.

Status Codes:

- **200** - OK
- **400** - Bad request (invalid JSON sent)
- **401** - Unauthorized
- **404** - Not found

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Enterprise Open source

Data source API

If you are running Grafana Enterprise, for some endpoints you'll need to have specific permissions. Refer to Role-based access control permissions for more information.

Get all data sources

`GET /api/datasources`

WARNING

This API currently doesn't handle pagination. The default maximum number of data sources returned is 5000. You can change this value in the default.ini file.

Required permissions

See note in the introduction for an explanation.

Action	Scope
datasources:read	datasources:*

Examples

Example Request:

http Copy

```
GET /api/datasources HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

[
  {
    "id": 1,
    "orgId": 1,
    "uid": "H8joYFVGz"
    "name": "datasource_elastic",
    "type": "elasticsearch",
    "typeLogoUrl": "public/app/plugins/datasource/elasticsearch/img/elasticsearch.svg",
    "access": "proxy",
    "url": "http://mydatasource.com",
    "password": "",
    "user": "",
    "database": "grafana-dash",
    "basicAuth": false,
    "isDefault": false,
    "jsonData": {
```

Expand code

Get a single data source by id

`GET /api/datasources/:datasourceId`

WARNING

This API is deprecated since Grafana v9.0.0 and will be removed in a future release. Refer to the API for getting a single data source by UID or to the API for getting a single data source by its name.

Required permissions

See note in the introduction for an explanation.

Action	Scope
datasources:read	datasources:* datasources:id:* datasources:id:1 (single data source)

Examples

Example Request:

```http
GET /api/datasources/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{
  "id": 1,
  "uid": "kLtEtcRGk",
  "orgId": 1,
  "name": "test_datasource",
  "type": "graphite",
  "typeLogoUrl": "",
  "access": "proxy",
  "url": "http://mydatasource.com",
  "password": "",
  "user": "",
  "database": "",
  "basicAuth": false,
  "basicAuthUser": "",
  "basicAuthPassword": "",
  "withCredentials": false,
```

Get a single data source by uid

`GET /api/datasources/uid/:uid`

Required permissions

See note in the introduction for an explanation.

Action	Scope
datasources:read	datasources:* datasources:uid:* datasources:uid:kLtEtcRGk (single data source)

Examples

Example request:

http Copy

```
GET /api/datasources/uid/kLtEtcRGk HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{
  "id": 1,
  "uid": "kLtEtcRGk",
  "orgId": 1,
  "name": "test_datasource",
  "type": "graphite",
  "typeLogoUrl": "",
  "access": "proxy",
  "url": "http://mydatasource.com",
  "password": "",
  "user": "",
  "database": "",
  "basicAuth": false,
  "basicAuthUser": "",
  "basicAuthPassword": "",
```

Expand code

Get a single data source by name

`GET /api/datasources/name/:name`

Required permissions

See note in the introduction for an explanation.

Action	Scope
datasources:read	datasources:* datasources:name:* datasources:name:test_datasource (single data source)

Examples

Example Request:

http Copy

```
GET /api/datasources/name/test_datasource HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{
  "id": 1,
  "uid": "kLtEtcRGk",
  "orgId": 1,
  "name": "test_datasource",
  "type": "graphite",
  "typeLogoUrl": "",
  "access": "proxy",
  "url": "http://mydatasource.com",
  "password": "",
  "user": "",
```

```
  "database": "",
  "basicAuth": false,
  "basicAuthUser": "",
  "basicAuthPassword": "",
  "withCredentials": false
```

Expand code

Get data source Id by name

`GET /api/datasources/id/:name`

Required permissions

See note in the introduction for an explanation.

Action	Scope
datasources.id:read	datasources:* datasources:name:* datasources:name:test_datasource (single data source)

Examples

Example Request:

http Copy

```
GET /api/datasources/id/test_datasource HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{
  "id":1
}
```

Create a data source

`POST /api/datasources`

Required permissions

See note in the introduction for an explanation.

Action	Scope
datasources:create	n/a

Examples

Example Graphite Request:

http Copy

```
POST /api/datasources HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "name":"test_datasource",
  "type":"graphite",
  "url":"http://mydatasource.com",
  "access":"proxy",
  "basicAuth":false
}
```

Example Graphite Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{
  "datasource": {
    "id": 1,
    "orgId": 1,
```

```
    "name": "test_datasource",
    "type": "graphite",
    "typeLogoUrl": "",
    "access": "proxy",
    "url": "http://mydatasource.com",
    "password": "",
    "user": "",
    "database": "",
    "basicAuth": false,
    "basicAuthUser": "",
    "basicAuthPassword": "",
```


Expand code

NOTE

By defining `password` and `basicAuthPassword` under `secureJsonData` Grafana encrypts them securely as an encrypted blob in the database. The response then lists the encrypted fields under `secureJsonFields`.

Example Graphite Request with basic auth enabled:

http

Copy

```
POST /api/datasources HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "name": "test_datasource",
  "type": "graphite",
  "url": "http://mydatasource.com",
  "access": "proxy",
  "basicAuth": true,
  "basicAuthUser": "basicuser",
  "secureJsonData": {
    "basicAuthPassword": "basicpassword"
  }
}
```

Example Response with basic auth enabled:

http

Copy

```
HTTP/1.1 200
Content-Type: application/json

{
  "datasource": {
    "id": 1,
    "orgId": 1,
    "name": "test_datasource",
    "type": "graphite",
    "typeLogoUrl": "",
    "access": "proxy",
    "url": "http://mydatasource.com",
    "password": "",
    "user": "",
    "database": "",
    "basicAuth": true,
    "basicAuthUser": "basicuser",
    "basicAuthPassword": "",
    "withCredentials": false,
```

Expand code

Example CloudWatch Request:

http Copy

```
POST /api/datasources HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "name": "test_datasource",
  "type": "cloudwatch",
  "url": "http://monitoring.us-west-1.amazonaws.com",
  "access": "proxy",
  "jsonData": {
    "authType": "keys",
    "defaultRegion": "us-west-1"
  },
  "secureJsonData": {
    "accessKey": "Ol4pIDpeKSA6XikgOl4p",
    "secretKey": "dGVzdCBrZXkgYmxlYXNlIGRvbid0IHN0ZWFs"
  }
}
```

Expand code

Update an existing data source by id

`PUT /api/datasources/:datasourceId`

> **WARNING**
>
> This API is deprecated since Grafana v9.0.0 and will be removed in a future release. Refer to the new data source update API.

Required permissions

See note in the introduction for an explanation.

Action	Scope
datasources:write	datasources:* datasources:id:* datasources:id:1 (single data source)

Examples

Example Request:

http

Copy

```
PUT /api/datasources/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "id":1,
  "orgId":1,
  "name":"test_datasource",
  "type":"graphite",
  "access":"proxy",
  "url":"http://mydatasource.com",
  "password":"",
  "user":"",
  "database":"",
  "basicAuth":true,
  "basicAuthUser":"basicuser",
  "secureJsonData": {
```

Expand code

"basicAuthPassword": "basicpassword"

Example Response:

http

Copy

```
HTTP/1.1 200
Content-Type: application/json

{
  "datasource": {
    "id": 1,
    "uid": "kLtEtcRGk",
    "orgId": 1,
    "name": "test_datasource",
    "type": "graphite",
    "typeLogoUrl": "",
    "access": "proxy",
    "url": "http://mydatasource.com",
    "password": "",
    "user": "",
    "database": "",
    "basicAuth": true,
    "basicAuthUser": "basicuser",
    "basicAuthPassword": "",
```

Expand code

NOTE

Similar to creating a data source, `password` and `basicAuthPassword` should be defined under `secureJsonData` in order to be stored securely as an encrypted blob in the database. Then, the encrypted fields are listed under `secureJsonFields` section in the response.

Update an existing data source

`PUT /api/datasources/uid/:uid`

Required permissions

See note in the introduction for an explanation.

Action	Scope
datasources:write	datasources:* datasources:uid:* datasources:uid:kLtEtcRGk (single data source)

Examples

Example Request:

http

Copy

```
PUT /api/datasources/uid/kLtEtcRGk HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "id":1,
  "uid": "updated UID",
  "orgId":1,
  "name":"test_datasource",
  "type":"graphite",
  "access":"proxy",
  "url":"http://mydatasource.com",
  "password":"",
  "user":"",
  "database":"",
  "basicAuth":true,
  "basicAuthUser":"basicuser",
  "secureJsonData": {
```

Expand code

Example Response:

http

Copy

```
HTTP/1.1 200
Content-Type: application/json

{
  "datasource": {
    "id": 1,
    "uid": "updated UID",
    "orgId": 1,
    "name": "test_datasource",
    "type": "graphite",
    "typeLogoUrl": "",
    "access": "proxy",
    "url": "http://mydatasource.com",
    "password": "",
```

```
"user": "",
"database": "",
"basicAuth": true,
"basicAuthUser": "basicuser",
```

Expand code

> **NOTE**
>
> Similar to creating a data source, `password` and `basicAuthPassword` should be defined under `secureJsonData` in order to be stored securely as an encrypted blob in the database. Then, the encrypted fields are listed under `secureJsonFields` section in the response.## Update an existing data source by id

Delete an existing data source by id

`DELETE /api/datasources/:datasourceId`

> **WARNING**
>
> This API is deprecated since Grafana v9.0.0 and will be removed in a future release. Refer to the API for deleting an existing data source by UID or to the API for deleting an existing data source by its name

Required permissions

See note in the introduction for an explanation.

Action	Scope
datasources:delete	datasources:* datasources:id:* datasources:id:1 (single data source)

Examples

Example Request:

http

Copy

```
DELETE /api/datasources/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{"message":"Data source deleted"}
```

Delete an existing data source by uid

`DELETE /api/datasources/uid/:uid`

Required permissions

See note in the introduction for an explanation.

Action	Scope
datasources:delete	datasources:* datasources:uid:* datasources:uid:kLtEtcRGk (single data source)

Examples

Example request:

```http
DELETE /api/datasources/uid/kLtEtcRGk HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response:

```http
HTTP/1.1 200
Content-Type: application/json

{
    "message": "Data source deleted",
```

```
    "id": 1
}
```

Delete an existing data source by name

`DELETE /api/datasources/name/:datasourceName`

Required permissions

See note in the introduction for an explanation.

Action	Scope
datasources:delete	datasources:* datasources:name:* datasources:name:test_datasource (single data source)

Examples

Example Request:

http Copy

```
DELETE /api/datasources/name/test_datasource HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{
  "message":"Data source deleted",
  "id": 1
}
```

Data source proxy calls by id

WARNING

This API is deprecated since Grafana v9.0.0 and will be removed in a future release. Refer to the new data source API for proxying requests.

`GET /api/datasources/proxy/:datasourceId/*`

Proxies all calls to the actual data source identified by the `datasourceId`.

Data source proxy calls

`GET /api/datasources/proxy/uid/:uid/*`

Proxies all calls to the actual data source identified by the `uid`.

Check data source health by id

Warning: This API is deprecated since Grafana v9.0.0 and will be removed in a future release. Refer to the new data source health check API.

`GET /api/datasources/:datasourceId/health`

Makes a call to the health endpoint of data source identified by the given `datasourceId`. This is not mandatory - every plugin author has to implement support for health checks in their plugin themselves.

Examples

Example Request:

http Copy

```
GET api/datasources/112/health HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json
```

```
{
  "message": "1. Successfully queried the CloudWatch metrics API.\n2. Successfully queried the Clo
  "status": "OK"
}
```


Check data source health

`GET /api/datasources/uid/:uid/health`

Makes a call to the health endpoint of data source identified by the given `uid`. This is not mandatory - every plugin author has to implement support for health checks in their plugin themselves.

Examples

Example Request:

http Copy

```
GET api/datasources/uid/P8045C56BDA891CB2/health HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

http Copy


```
HTTP/1.1 200
Content-Type: application/json

{
  "message": "1. Successfully queried the CloudWatch metrics API.\n2. Successfully queried the Clo
  "status": "OK"
}
```


Fetch data source resources by id

WARNING

This API is deprecated since Grafana v9.0.0 and will be removed in a future release. Refer to the new data source resources API.

`GET /api/datasources/:datasourceId/resources/*`

Makes a call to the resources endpoint of data source identified by the given `dashboardId`.

Examples

Example Request:

```http
GET api/datasources/112/resources/dimension-keys?region=us-east-2&namespace=AWS%2FEC2&dimensionFil
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

[
    {
        "text": "AutoScalingGroupName",
        "value": "AutoScalingGroupName",
        "label": "AutoScalingGroupName"
    },
    {
        "text": "ImageId",
        "value": "ImageId",
        "label": "ImageId"
    },
    {
        "text": "InstanceId",
        "value": "InstanceId",
        "label": "InstanceId"
    },
```

Fetch data source resources

`GET /api/datasources/uid/:uid/resources/*`

Makes a call to the resources endpoint of data source identified by the given `uid`.

Examples

Example Request:

```http
GET api/datasources/uid/P8045C56BDA891CB2/resources/dimension-keys?region=us-east-2&namespace=AWS%
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

[
    {
        "text": "AutoScalingGroupName",
        "value": "AutoScalingGroupName",
        "label": "AutoScalingGroupName"
    },
    {
        "text": "ImageId",
        "value": "ImageId",
        "label": "ImageId"
    },
    {
        "text": "InstanceId",
        "value": "InstanceId",
        "label": "InstanceId"
    },
```

Query a data source

Queries a data source having a backend implementation.

`POST /api/ds/query`

> **NOTE**

Grafana's built-in data sources usually have a backend implementation.

Example request for the Test data source:

http

Copy

```
POST /api/ds/query HTTP/1.1
Accept: application/json
Content-Type: application/json

{
   "queries":[
      {
         "refId":"A",
         "scenarioId":"csv_metric_values",
         "datasource":{
            "uid":"PD8C576611E62080A"
         },
         "format": "table"
         "maxDataPoints":1848,
         "intervalMs":200,
         "stringInput":"1,20,90,30,5,0",
      }
   ],
   "from":"now-5m",
```

Expand code

JSON Body schema:

- **from/to** – Specifies the time range for the queries. The time can be either epoch timestamps in milliseconds or relative using Grafana time units. For example, `now-5m`.
- **queries** – Specifies one or more queries. Must contain at least 1.
- **queries.datasource.uid** – Specifies the UID of data source to be queried. Each query in the request must have a unique `datasource`.
- **queries.refId** – Specifies an identifier of the query. Defaults to "A".
- **queries.format** – Specifies the format the data should be returned in. Valid options are `time_series` or `table` depending on the data source.
- **queries.maxDataPoints** - Species the maximum amount of data points that a dashboard panel can render. Defaults to 100.
- **queries.intervalMs** - Specifies the time series time interval in milliseconds. Defaults to 1000.

In addition, specific properties of each data source should be added in a request (for example **queries.stringInput** as shown in the request above). To better understand how to form a query for a certain data source, use the Developer Tools in your browser of choice and inspect the HTTP requests being made to `/api/ds/query`.

Example Test data source time series query response:

json

Copy


```
{
  "results": {
    "A": {
      "frames": [
        {
          "schema": {
            "refId": "A",
            "fields": [
              {
                "name": "time",
                "type": "time",
                "typeInfo": {
                  "frame": "time.Time"
                }
              },
              {
                "name": "A-series",
                "type": "number",
                "typeInfo": {
```


Expand code

Status codes

Code	Description
200	All data source queries returned a successful response.
400	Bad request due to invalid JSON, missing content type, missing or invalid fields, etc. Or one or more data source queries were unsuccessful. Refer to the body for more details.
403	Access denied.
404	Either the data source or plugin required to fulfil the request could not be found.
500	Unexpected error. Refer to the body and/or server logs for more details.

Data Source Permissions API

The Data Source Permissions is only available in Grafana Enterprise. Read more about Grafana Enterprise.

If you are running Grafana Enterprise, for some endpoints you'll need to have specific permissions. Refer to Role-based access control permissions for more information.

This API can be used to list, add and remove permissions for a data source.

Permissions can be set for a user, team, service account or a basic role (Admin, Editor, Viewer).

Get permissions for a data source

`GET /api/access-control/datasources/:uid`

Gets all existing permissions for the data source with the given `uid`.

Required permissions

See note in the introduction for an explanation.

Action	Scope
datasources.permissions:read	datasources:* datasources:uid:* datasources:uid:my_datasource (single data source)

Examples

Example request:

```http
GET /api/access-control/datasources/my_datasource HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response:

```http
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 551

[
    {
        "id": 1,
        "roleName": "fixed:datasources:reader",
        "isManaged": false,
        "isInherited": false,
        "isServiceAccount": false,
        "userId": 1,
        "userLogin": "admin_user",
        "userAvatarUrl": "/avatar/admin_user",
        "actions": [
            "datasources:read",
            "datasources:query",
            "datasources:read",
            "datasources:query",
```

Status codes:

- **200** - Ok
- **401** - Unauthorized
- **403** - Access denied
- **500** - Internal error

Add or revoke access to a data source for a user

`POST /api/access-control/datasources/:uid/users/:id`

Sets user permission for the data source with the given `uid`.

To add a permission, set the `permission` field to either `Query`, `Edit`, or `Admin`. To remove a permission, set the `permission` field to an empty string.

Required permissions

See note in the introduction for an explanation.

Action	Scope
datasources.permissions:write	datasources:* datasources:uid:* datasources:uid:my_datasource (single data source)

Examples

Example request:

http Copy

```
POST /api/access-control/datasources/my_datasource/users/1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "permission": "Query",
}
```

Example response:

http Copy

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 35

{"message": "Permission updated"}
```

Example request:

```http
POST /api/access-control/datasources/my_datasource/users/1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "permission": "",
}
```

Example response:

```http
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 35

{"message": "Permission removed"}
```

Status codes:

- **200** - Ok
- **400** - Permission cannot be added, see response body for details
- **401** - Unauthorized
- **403** - Access denied

Add or revoke access to a data source for a team

`POST /api/access-control/datasources/:uid/teams/:id`

Sets team permission for the data source with the given `uid`.

To add a permission, set the `permission` field to either `Query`, `Edit`, or `Admin`. To remove a permission, set the `permission` field to an empty string.

Required permissions

See note in the introduction for an explanation.

Action	Scope
datasources.permissions:write	datasources:* datasources:uid:* datasources:uid:my_datasource (single data source)

Examples

Example request:

http

Copy

```
POST /api/access-control/datasources/my_datasource/teams/1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "permission": "Edit",
}
```

Example response:

http

Copy

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 35

{"message": "Permission updated"}
```

Example request:

http

Copy

```
POST /api/access-control/datasources/my_datasource/teams/1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "permission": "",
}
```

Example response:

```http
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 35

{"message": "Permission removed"}
```

Status codes:

- **200** - Ok
- **400** - Permission cannot be added, see response body for details
- **401** - Unauthorized
- **403** - Access denied

Add or revoke access to a data source for a basic role

`POST /api/access-control/datasources/:uid/builtInRoles/:builtinRoleName`

Sets permission for the data source with the given `uid` to all users who have the specified basic role.

You can set permissions for the following basic roles: `Admin`, `Editor`, `Viewer`.

To add a permission, set the `permission` field to either `Query`, `Edit`, or `Admin`. To remove a permission, set the `permission` field to an empty string.

Required permissions

See note in the introduction for an explanation.

Action	Scope
datasources.permissions:write	datasources:* datasources:uid:* datasources:uid:my_datasource (single data source)

Examples

Example request:

```http
```

```
POST /api/access-control/datasources/my_datasource/builtInRoles/Admin
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "permission": "Edit",
}
```

Example response:

http Copy

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 35

{"message": "Permission updated"}
```

Example request:

http Copy

```
POST /api/access-control/datasources/my_datasource/builtInRoles/Viewer
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "permission": "",
}
```

Example response:

http Copy

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 35

{"message": "Permission removed"}
```

Status codes:

- **200** - Ok
- **400** - Permission cannot be added, see response body for details
- **401** - Unauthorized
- **403** - Access denied

Folder API

If you are running Grafana Enterprise, for some endpoints you'll need to have specific permissions. Refer to Role-based access control permissions for more information.

Identifier (id) vs unique identifier (uid)

The identifier (id) of a folder is an auto-incrementing numeric value and is only unique per Grafana install.

The unique identifier (uid) of a folder can be used for uniquely identify folders between multiple Grafana installs. It's automatically generated if not provided when creating a folder. The uid allows having consistent URLs for accessing folders and when syncing folders between multiple Grafana installs. This means that changing the title of a folder will not break any bookmarked links to that folder.

The uid can have a maximum length of 40 characters.

Get all folders

`GET /api/folders`

Returns all folders that the authenticated user has permission to view. You can control the maximum number of folders returned through the `limit` query parameter, the default is 1000. You can also pass the `page` query parameter for fetching folders from a page other than the first one.

If nested folders are enabled, the operation expects an additional optional query parameter `parentUid` with the parent folder UID, and returns the immediate subfolders that the authenticated user has permission to view. If the parameter is not supplied, then the operation returns immediate subfolders under the root that the authenticated user has permission to view.

Required permissions

See note in the introduction for an explanation.

Action	Scope
`folders:read`	`folders:*`

Example Request:

```http
GET /api/folders?limit=10 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

[
  {
    "id":1,
    "uid": "nErXDvCkzz",
    "title": "Department ABC"
  },
  {
    "id":2,
    "uid": "k3S1cklGk",
    "title": "Department RND"
  }
]
```

Get folder by uid

`GET /api/folders/:uid`

Will return the folder given the folder uid.

Required permissions

See note in the introduction for an explanation.

Action	Scope
`folders:read`	`folders:*`

Example Request:

http Copy

```
GET /api/folders/nErXDvCkzzh HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{
  "id":1,
  "uid": "nErXDvCkzz",
  "title": "Department ABC",
  "url": "/dashboards/f/nErXDvCkzz/department-abc",
  "hasAcl": false,
  "canSave": true,
  "canEdit": true,
  "canAdmin": true,
  "createdBy": "admin",
  "created": "2018-01-31T17:43:12+01:00",
  "updatedBy": "admin",
  "updated": "2018-01-31T17:43:12+01:00",
  "version": 1
}
```

If nested folders are enabled, and the folder is nested (lives under another folder), then the response additionally contains:

- **parentUid** - The parent folder UID.

- **parents** - An array with the whole tree hierarchy, starting from the root going down up to the parent folder.

Status Codes:

- **200** – Found
- **401** – Unauthorized
- **403** – Access Denied
- **404** – Folder not found

Create folder

`POST /api/folders`

Creates a new folder.

Required permissions

See note in the introduction for an explanation.

`folders:create` allows creating folders in the root level. To create a subfolder, `folders:write` scoped to the parent folder is required in addition to `folders:create`.

Action	Scope
`folders:create`	n/a
`folders:write`	`folders:*`

Example Request:

http Copy

```
POST /api/folders HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "uid": "nErXDvCkzz",
  "title": "Department ABC",
  "parentUid": "fgnj5e52gel76g"
}
```

JSON Body schema:

- **uid** – Optional unique identifier.
- **title** – The title of the folder.
- **parentUid** - Optional field, the unique identifier of the parent folder under which the folder should be created. Requires nested folders to be enabled.

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{
  "id":1,
  "uid": "nErXDvCkzz",
  "title": "Department ABC",
  "url": "/dashboards/f/nErXDvCkzz/department-abc",
  "hasAcl": false,
  "canSave": true,
  "canEdit": true,
  "canAdmin": true,
  "createdBy": "admin",
  "created": "2018-01-31T17:43:12+01:00",
  "updatedBy": "admin",
  "updated": "2018-01-31T17:43:12+01:00",
  "version": 1
}
```

If nested folders are enabled, and the folder is nested (lives under another folder) then the response additionally contains:

- **parentUid** - the parent folder UID.
- **parents** - an array with the whole tree hierarchy starting from the root going down up to the parent folder.

Status Codes:

- **200** – Created
- **400** – Errors (invalid json, missing or invalid fields, etc)
- **401** – Unauthorized
- **403** – Access Denied
- **409** - Folder already exists

Update folder

`PUT /api/folders/:uid`

Updates an existing folder identified by uid.

Required permissions

See note in the introduction for an explanation.

Action	Scope
`folders:write`	`folders:*`

Example Request:

http

```
PUT /api/folders/nErXDvCkzz HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "title":"Department DEF",
  "version": 1
}
```

JSON Body schema:

- **title** – The title of the folder.
- **version** – Provide the current version to be able to update the folder. Not needed if `overwrite=true`.
- **overwrite** – Set to true if you want to overwrite existing folder with newer version.

Example Response:

http

```
HTTP/1.1 200
Content-Type: application/json

{
  "id":1,
```

```
  "uid": "nErXDvCkzz",
  "title": "Department DEF",
  "url": "/dashboards/f/nErXDvCkzz/department-def",
  "hasAcl": false,
  "canSave": true,
  "canEdit": true,
  "canAdmin": true,
  "createdBy": "admin",
  "created": "2018-01-31T17:43:12+01:00",
  "updatedBy": "admin",
  "updated": "2018-01-31T17:43:12+01:00",
  "version": 1
}
```


If nested folders are enabled, and the folder is nested (lives under another folder) then the response additionally contains:

- **parentUid** - the parent folder UID.
- **parents** - an array with the whole tree hierarchy starting from the root going down up to the parent folder.

Status Codes:

- **200** – Updated
- **400** – Errors (invalid json, missing or invalid fields, etc)
- **401** – Unauthorized
- **403** – Access Denied
- **404** – Folder not found
- **412** – Precondition failed

The **412** status code is used for explaining that you cannot update the folder and why. There can be different reasons for this:

- The folder has been changed by someone else, `status=version-mismatch`

The response body will have the following properties:

http Copy


```
HTTP/1.1 412 Precondition Failed
Content-Type: application/json; charset=UTF-8
Content-Length: 97

{
  "message": "The folder has been changed by someone else",
```

```
  "status": "version-mismatch"
}
```

Delete folder

`DELETE /api/folders/:uid`

Deletes an existing folder identified by UID along with all dashboards (and their alerts) stored in the folder. This operation cannot be reverted.

If Grafana Alerting is enabled, you can set an optional query parameter `forceDeleteRules=false` so that requests will fail with 400 (Bad Request) error if the folder contains any Grafana alerts. However, if this parameter is set to `true` then it will delete any Grafana alerts under this folder.

Required permissions

See note in the introduction for an explanation.

Action	Scope
`folders:delete`	`folders:*`

Example Request:

http Copy

```
DELETE /api/folders/nErXDvCkzz HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{
  "message":"Folder deleted",
  "id": 2
}
```

Status Codes:

- **200** – Deleted
- **401** – Unauthorized
- **400** – Bad Request
- **403** – Access Denied
- **404** – Folder not found

Get folder by id

`GET /api/folders/id/:id`

Will return the folder identified by id.

This is deprecated. Use get folder by UID instead.

Required permissions

See note in the introduction for an explanation.

Action	Scope
`folders:read`	`folders:*`

Example Request:

http Copy

```
GET /api/folders/id/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{
  "id":1,
  "uid": "nErXDvCkzz",
  "title": "Department ABC",
  "url": "/dashboards/f/nErXDvCkzz/department-abc",
  "hasAcl": false,
```

```
  "canSave": true,
  "canEdit": true,
  "canAdmin": true,
  "createdBy": "admin",
  "created": "2018-01-31T17:43:12+01:00",
  "updatedBy": "admin",
  "updated": "2018-01-31T17:43:12+01:00",
  "version": 1
}
```


Status Codes:

- **200** – Found
- **401** – Unauthorized
- **403** – Access Denied
- **404** – Folder not found

Move folder

`POST /api/folders/:uid/move`

Moves the folder.

This is relevant only if nested folders are enabled.

Required permissions

See note in the introduction for an explanation.

If moving the folder under another folder:

Action	Scope
`folders:write`	`folders:uid:<destination folder UID>`

If moving the folder under root:

Action	Scope
`folders:create`	`folders:*`

JSON body schema:

- **parentUid** – Optional unique identifier of the new parent folder. If this is empty, then the folder is moved under the root.

Example Request:

```http
POST /api/folders/a5393ec3-5568-4e88-8809-b866968ae8a6/move HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "parentUid": "d80b18c0-266a-4aa4-ad5d-5537a00cb8e8",
}
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{
    "id": 4,
    "uid": "a5393ec3-5568-4e88-8809-b866968ae8a6",
    "title": "just-testing",
    "url": "/dashboards/f/a5393ec3-5568-4e88-8809-b866968ae8a6/just-testing",
    "hasAcl": false,
    "canSave": true,
    "canEdit": true,
    "canAdmin": true,
    "canDelete": true,
    "createdBy": "Anonymous",
    "created": "2023-04-27T21:55:01.593741+03:00",
    "updatedBy": "Anonymous",
    "updated": "2023-04-27T21:55:15.747444+03:00",
    "parentUid": "d80b18c0-266a-4aa4-ad5d-5537a00cb8e8",
    "parents": [
```

Status Codes:

- **200** – Moved
- **400** – Errors (invalid JSON, missing or invalid fields, and so on)
- **401** – Unauthorized

- **403** – Access denied
- **404** – Not found

Folder Permissions API

This API can be used to update/get the permissions for a folder.

Permissions with `folderId=-1` are the default permissions for users with the Viewer and Editor roles. Permissions can be set for a user, a team or a role (Viewer or Editor). Permissions cannot be set for Admins - they always have access to everything.

The permission levels for the permission field:

- 1 = View
- 2 = Edit
- 4 = Admin

If you are running Grafana Enterprise, for some endpoints you'll need to have specific permissions. Refer to Role-based access control permissions for more information.

Get permissions for a folder

`GET /api/folders/:uid/permissions`

Gets all existing permissions for the folder with the given `uid`.

Required permissions

See note in the introduction for an explanation.

Action	Scope
`folders.permissions:read`	`folders:*`

Example request:

http

```
GET /api/folders/nErXDvCkzz/permissions HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response

http

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 551

[
  {
    "id": 1,
    "folderId": -1,
    "created": "2017-06-20T02:00:00+02:00",
    "updated": "2017-06-20T02:00:00+02:00",
    "userId": 0,
    "userLogin": "",
    "userEmail": "",
    "teamId": 0,
    "team": "",
    "role": "Viewer",
    "permission": 1,
    "permissionName": "View",
    "uid": "nErXDvCkzz",
```

Expand code

Status Codes:

- **200** - Ok
- **401** - Unauthorized
- **403** - Access denied
- **404** - Folder not found

Update permissions for a folder

`POST /api/folders/:uid/permissions`

Updates permissions for a folder. This operation will remove existing permissions if they're not included in the request.

Required permissions

See note in the introduction for an explanation.

Action	Scope
`folders.permissions:write`	`folders:*`

Example request:

http Copy

```
POST /api/folders/nErXDvCkzz/permissions
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
{
  "items": [
    {
      "role": "Viewer",
      "permission": 1
    },
    {
      "role": "Editor",
      "permission": 2
    },
    {
      "teamId": 1,
      "permission": 1
    },
    {
```

Expand code

JSON body schema:

- **items** - The permission items to add/update. Items that are omitted from the list will be removed.

Example response:

http

Copy

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 35

{"message":"Folder permissions updated","id":1,"title":"Department ABC"}
```

Status Codes:

- **200** - Ok
- **401** - Unauthorized
- **403** - Access denied
- **404** - Dashboard not found

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Enterprise Open source

Folder/Dashboard Search API

Search folders and dashboards

`GET /api/search/`

Note: When using Role-based access control, search results will contain only dashboards and folders which you have access to.

Query parameters:

- **query** – Search Query
- **tag** – List of tags to search for
- **type** – Type to search for, `dash-folder` or `dash-db`
- **dashboardIds** – List of dashboard id's to search for
- **dashboardUID** - List of dashboard uid's to search for, It is deprecated since Grafana v9.1, please use dashboardUIDs instead
- **dashboardUIDs** – List of dashboard uid's to search for
- **folderUIDs** – List of folder UIDs to search in
- **starred** – Flag indicating if only starred Dashboards should be returned
- **limit** – Limit the number of returned results (max is 5000; default is 1000)
- **page** – Use this parameter to access hits beyond limit. Numbering starts at 1. limit param acts as page size. Only available in Grafana v6.2+.

Example request for retrieving folders and dashboards at the root level:

```
http
```

Copy

```
GET /api/search?query=&starred=false HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response for retrieving folders and dashboards at the root level:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

[
  {
    "id": 163,
    "uid": "000000163",
    "title": "Folder",
    "url": "/dashboards/f/000000163/folder",
    "type": "dash-folder",
    "tags": [],
    "isStarred": false,
    "uri":"db/folder" // deprecated in Grafana v5.0
  },
  {
    "id":1,
    "uid": "cIBgcSjkk",
    "title":"Production Overview",
    "url": "/d/cIBgcSjkk/production-overview",
```

Expand code

Example request searching for dashboards:

http Copy

```
GET /api/search?query=Production%20Overview&starred=true&tag=prod HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response searching for dashboards:

http Copy

```
HTTP/1.1 200
Content-Type: application/json
```

```
[
  {
    "id":1,
    "uid": "cIBgcSjkk",
    "title":"Production Overview",
    "url": "/d/cIBgcSjkk/production-overview",
    "type":"dash-db",
    "tags":[prod],
    "isStarred":true,
    "folderId": 2,
    "folderUid": "000000163",
    "folderTitle": "Folder",
    "folderUrl": "/dashboards/f/000000163/folder",
    "uri":"db/production-overview" // deprecated in Grafana v5.0
  }
]
```


Expand code

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

User and Org Preferences API

Keys:

- **theme** - One of: `light`, `dark`, or an empty string for the default theme
- **homeDashboardId** - The numerical `:id` of a favorited dashboard, default: `0`
- **timezone** - One of: `utc`, `browser`, or an empty string for the default

Omitting a key will cause the current value to be replaced with the system default value.

Get Current User Prefs

`GET /api/user/preferences`

Example Request:

http Copy

```
GET /api/user/preferences HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{
    "theme": "",
```

```
    "homeDashboardId": 217,
    "homeDashboardUID": "jcIIG-07z",
    "timezone": "utc",
    "weekStart": "",
    "navbar": {
        "savedItems": null
    },
    "queryHistory": {
        "homeTab": ""
    }
}
```


Update Current User Prefs

`PUT /api/user/preferences`

Example Request:

http Copy

```
PUT /api/user/preferences HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "theme": "",
  "homeDashboardUID":"home",
  "timezone":"utc"
}
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: text/plain; charset=utf-8

{"message":"Preferences updated"}
```

Patch Current User Prefs

Update one or more preferences without modifying the others.

`PATCH /api/user/preferences`

Example Request:

```http
PATCH /api/user/preferences HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "theme": "dark"
}
```

Example Response:

```http
HTTP/1.1 200
Content-Type: text/plain; charset=utf-8

{"message":"Preferences updated"}
```

Get Current Org Prefs

`GET /api/org/preferences`

Example Request:

```http
GET /api/org/preferences HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{
```

```
    "theme": "",
    "homeDashboardId": 0,
    "timezone": "",
    "weekStart": "",
    "navbar": {
        "savedItems": null
    },
    "queryHistory": {
        "homeTab": ""
    }
}
```

Update Current Org Prefs

`PUT /api/org/preferences`

Example Request:

http Copy

```
PUT /api/org/preferences HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "theme": "",
  "homeDashboardUID":"home",
  "timezone":"utc"
}
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: text/plain; charset=utf-8

{"message":"Preferences updated"}
```

Patch Current Org Prefs

Update one or more preferences without modifying the others.

`PATCH /api/org/preferences`

Example Request:

http

```http
PATCH /api/org/preferences HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "theme": "dark"
}
```

Example Response:

http

```http
HTTP/1.1 200
Content-Type: text/plain; charset=utf-8

{"message":"Preferences updated"}
```

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Snapshot API

Create new snapshot

`POST /api/snapshots`

Example Request:

http Copy

```
POST /api/snapshots HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "dashboard": {
    "editable":false,
    "nav":[
    {
      "enable":false,
    "type":"timepicker"
    }
    ],
    "rows": [
      {

      }
    ],
```

Expand code

JSON Body schema:

- **dashboard** – Required. The complete dashboard model.
- **name** – Optional. snapshot name
- **expires** - Optional. When the snapshot should expire in seconds. 3600 is 1 hour, 86400 is 1 day. Default is never to expire.
- **external** - Optional. Save the snapshot on an external server rather than locally. Default is `false`.
- **key** - Optional. Define the unique key. Required if **external** is `true`.
- **deleteKey** - Optional. Unique key used to delete the snapshot. It is different from the **key** so that only the creator can delete the snapshot. Required if **external** is `true`.

> **NOTE**
>
> When creating a snapshot using the API, you have to provide the full dashboard payload including the snapshot data. This endpoint is designed for the Grafana UI.

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json
{
  "deleteKey":"XXXXXXX",
  "deleteUrl":"myurl/api/snapshots-delete/XXXXXXX",
  "key":"YYYYYYY",
  "url":"myurl/dashboard/snapshot/YYYYYYY",
  "id": 1
}
```

Keys:

- **deleteKey** – Key generated to delete the snapshot
- **key** – Key generated to share the dashboard

Get list of Snapshots

`GET /api/dashboard/snapshots`

Query parameters:

- **query** – Search Query
- **limit** – Limit the number of returned results

Example Request:

http Copy

```
GET /api/dashboard/snapshots HTTP/1.1
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

[
  {
    "id":8,
    "name":"Home",
    "key":"YYYYYYY",
    "orgId":1,
    "userId":1,
    "external":false,
    "externalUrl":"",
    "expires":"2200-13-32T25:23:23+02:00",
    "created":"2200-13-32T28:24:23+02:00",
    "updated":"2200-13-32T28:24:23+02:00"
  }
]
```

Get Snapshot by Key

`GET /api/snapshots/:key`

Example Request:

http Copy

```
GET /api/snapshots/YYYYYYY HTTP/1.1
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{
  "meta":{
    "isSnapshot":true,
    "type":"snapshot",
    "canSave":false,
    "canEdit":false,
    "canStar":false,
    "slug":"",
    "expires":"2200-13-32T25:23:23+02:00",
    "created":"2200-13-32T28:24:23+02:00"
  },
  "dashboard": {
    "editable":false,
    "nav": [
      {
        "enable":false,
```

Expand code

Delete Snapshot by Key

`DELETE /api/snapshots/:key`

Example Request:

http

Copy

```
DELETE /api/snapshots/YYYYYYY HTTP/1.1
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

http

Copy

```
HTTP/1.1 200
Content-Type: application/json

{"message":"Snapshot deleted. It might take an hour before it's cleared from any CDN caches.", "id
```

Delete Snapshot by deleteKey

This API call can be used without authentication by using the secret delete key for the snapshot.

`GET /api/snapshots-delete/:deleteKey`

Example Request:

```http
GET /api/snapshots-delete/XXXXXXX HTTP/1.1
Accept: application/json
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{"message":"Snapshot deleted. It might take an hour before it's cleared from any CDN caches.", "id
```

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Legacy Alerting API

NOTE

Starting with v9.0, the Legacy Alerting HTTP API is deprecated. It will be removed in a future release.

This topic is relevant for the legacy dashboard alerts only.

If you are using Grafana Alerting, refer to Alerting provisioning API

You can find Grafana Alerting API specification details here. Also, refer to Grafana Alerting alerts documentation for details on how to create and manage new alerts.

You can use the Alerting API to get information about legacy dashboard alerts and their states but this API cannot be used to modify the alert. To create new alerts or modify them you need to update the dashboard JSON that contains the alerts.

Get alerts

`GET /api/alerts/`

Example Request:

http Copy

```
GET /api/alerts HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Querystring Parameters:

These parameters are used as querystring parameters. For example:

`/api/alerts?dashboardId=1`

- **dashboardId** – Limit response to alerts in specified dashboard(s). You can specify multiple dashboards, e.g. dashboardId=23&dashboardId=35.
- **panelId** – Limit response to alert for a specified panel on a dashboard.
- **query** - Limit response to alerts having a name like this value.
- **state** - Return alerts with one or more of the following alert states: `ALL`, `no_data`, `paused`, `alerting`, `ok`, `pending`. To specify multiple states use the following format: `?state=paused&state=alerting`
- **limit** - Limit response to *X* number of alerts.
- **folderId** – Limit response to alerts of dashboards in specified folder(s). You can specify multiple folders, e.g. folderId=23&folderId=35.
- **dashboardQuery** - Limit response to alerts having a dashboard name like this value.
- **dashboardTag** - Limit response to alerts of dashboards with specified tags. To do an "AND" filtering with multiple tags, specify the tags parameter multiple times e.g. dashboardTag=tag1&dashboardTag=tag2.

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

[
  {
    "id": 1,
    "dashboardId": 1,
    "dashboardUId": "ABcdEFghij"
    "dashboardSlug": "sensors",
    "panelId": 1,
    "name": "fire place sensor",
    "state": "alerting",
    "newStateDate": "2018-05-14T05:55:20+02:00",
    "evalDate": "0001-01-01T00:00:00Z",
    "evalData": null,
    "executionError": "",
    "url": "http://grafana.com/dashboard/db/sensors"
  }
]
```

Expand code

Get alert by id

`GET /api/alerts/:id`

Example Request:

```http
GET /api/alerts/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{
  "id": 1,
  "dashboardId": 1,
  "dashboardUId": "ABcdEFghij"
  "dashboardSlug": "sensors",
  "panelId": 1,
  "name": "fire place sensor",
  "state": "alerting",
  "message": "Someone is trying to break in through the fire place",
  "newStateDate": "2018-05-14T05:55:20+02:00",
  "evalDate": "0001-01-01T00:00:00Z",
  "evalData": "evalMatches": [
    {
      "metric": "movement",
      "tags": {
        "name": "fireplace_chimney"
```

Important Note: “evalMatches” data is cached in the db when and only when the state of the alert changes (e.g. transitioning from “ok” to “alerting” state).

If data from one server triggers the alert first and, before that server is seen leaving alerting state, a second server also enters a state that would trigger the alert, the second server will not be visible in “evalMatches” data.

Pause alert by id

`POST /api/alerts/:id/pause`

Example Request:

```http
POST /api/alerts/1/pause HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "paused": true
}
```

The :id query parameter is the id of the alert to be paused or unpaused.

JSON Body Schema:

- **paused** – Can be `true` or `false`. True to pause an alert. False to unpause an alert.

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{
  "alertId": 1,
  "state":   "Paused",
  "message": "alert paused"
}
```

Pause all alerts

See Admin API.

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Legacy Alerting Notification Channels API

NOTE

Starting with v9.0, the Legacy Alerting Notification Channels API is deprecated. It will be removed in a future release.

This page documents the Alerting Notification Channels API.

Identifier (id) vs unique identifier (uid)

The identifier (id) of a notification channel is an auto-incrementing numeric value and is only unique per Grafana install.

The unique identifier (uid) of a notification channel can be used for uniquely identify a notification channel between multiple Grafana installs. It's automatically generated if not provided when creating a notification channel. The uid allows having consistent URLs for accessing notification channels and when syncing notification channels between multiple Grafana installations, refer to alert notification channel provisioning.

The uid can have a maximum length of 40 characters.

Get all notification channels

Returns all notification channels that the authenticated user has permission to view.

`GET /api/alert-notifications`

Example request:

http Copy

```
GET /api/alert-notifications HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

[
  {
    "id": 1,
    "uid": "team-a-email-notifier",
    "name": "Team A",
    "type": "email",
    "isDefault": false,
    "sendReminder": false,
    "disableResolveMessage": false,
    "settings": {
      "addresses": "dev@grafana.com"
    },
    "created": "2018-04-23T14:44:09+02:00",
    "updated": "2018-08-20T15:47:49+02:00"
  }
]
```

Expand code

Get all notification channels (lookup)

Returns all notification channels, but with less detailed information. Accessible by any authenticated user and is mainly used by providing alert notification channels in Grafana UI when configuring alert rule.

`GET /api/alert-notifications/lookup`

Example request:

http Copy

```
GET /api/alert-notifications/lookup HTTP/1.1
Accept: application/json
```

```
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

[
  {
    "id": 1,
    "uid": "000000001",
    "name": "Test",
    "type": "email",
    "isDefault": false
  },
  {
    "id": 2,
    "uid": "000000002",
    "name": "Slack",
    "type": "slack",
    "isDefault": false
  }
]
```

Expand code

Get notification channel by uid

`GET /api/alert-notifications/uid/:uid`

Returns the notification channel given the notification channel uid.

Example request:

http Copy

```
GET /api/alert-notifications/uid/team-a-email-notifier HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{
  "id": 1,
  "uid": "team-a-email-notifier",
  "name": "Team A",
  "type": "email",
  "isDefault": false,
  "sendReminder": false,
  "disableResolveMessage": false,
  "settings": {
    "addresses": "dev@grafana.com"
  },
  "created": "2018-04-23T14:44:09+02:00",
  "updated": "2018-08-20T15:47:49+02:00"
}
```

Get notification channel by id

`GET /api/alert-notifications/:id`

Returns the notification channel given the notification channel id.

Example request:

http Copy

```
GET /api/alert-notifications/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{
  "id": 1,
  "uid": "team-a-email-notifier",
  "name": "Team A",
```

```
  "type": "email",
  "isDefault": false,
  "sendReminder": false,
  "disableResolveMessage": false,
  "settings": {
    "addresses": "dev@grafana.com"
  },
  "created": "2018-04-23T14:44:09+02:00",
  "updated": "2018-08-20T15:47:49+02:00"
}
```

Create notification channel

You can find the full list of supported notifiers on the alert notifiers page.

`POST /api/alert-notifications`

Example request:

http Copy

```
POST /api/alert-notifications HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "uid": "new-alert-notification", // optional
  "name": "new alert notification",  //Required
  "type":  "email", //Required
  "isDefault": false,
  "sendReminder": false,
  "settings": {
    "addresses": "dev@grafana.com"
  }
}
```

Example response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json
```

```
{
  "id": 1,
  "uid": "new-alert-notification",
  "name": "new alert notification",
  "type": "email",
  "isDefault": false,
  "sendReminder": false,
  "settings": {
    "addresses": "dev@grafana.com"
  },
  "created": "2018-04-23T14:44:09+02:00",
  "updated": "2018-08-20T15:47:49+02:00"
}
```

Update notification channel by uid

`PUT /api/alert-notifications/uid/:uid`

Updates an existing notification channel identified by uid.

Example request:

http Copy

```
PUT /api/alert-notifications/uid/cIBgcSjkk HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "uid": "new-alert-notification", // optional
  "name": "new alert notification",  //Required
  "type":  "email", //Required
  "isDefault": false,
  "sendReminder": true,
  "frequency": "15m",
  "settings": {
    "addresses": "dev@grafana.com"
  }
}
```

Example response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{
  "id": 1,
  "uid": "new-alert-notification",
  "name": "new alert notification",
  "type": "email",
  "isDefault": false,
  "sendReminder": true,
  "frequency": "15m",
  "settings": {
    "addresses": "dev@grafana.com"
  },
  "created": "2017-01-01 12:34",
  "updated": "2017-01-01 12:34"
}
```

Update notification channel by id

`PUT /api/alert-notifications/:id`

Updates an existing notification channel identified by id.

Example request:

http Copy

```
PUT /api/alert-notifications/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "id": 1,
  "uid": "new-alert-notification", // optional
  "name": "new alert notification",  //Required
  "type":  "email", //Required
  "isDefault": false,
  "sendReminder": true,
  "frequency": "15m",
  "settings": {
    "addresses": "dev@grafana.com"
```

```
  }
}
```

Example response:

```http
HTTP/1.1 200
Content-Type: application/json

{
  "id": 1,
  "uid": "new-alert-notification",
  "name": "new alert notification",
  "type": "email",
  "isDefault": false,
  "sendReminder": true,
  "frequency": "15m",
  "settings": {
    "addresses": "dev@grafana.com"
  },
  "created": "2017-01-01 12:34",
  "updated": "2017-01-01 12:34"
}
```

Delete alert notification by uid

`DELETE /api/alert-notifications/uid/:uid`

Deletes an existing notification channel identified by uid.

Example request:

```http
DELETE /api/alert-notifications/uid/team-a-email-notifier HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response:

```http
```

```
HTTP/1.1 200
Content-Type: application/json

{
  "message": "Notification deleted"
}
```

Delete alert notification by id

`DELETE /api/alert-notifications/:id`

Deletes an existing notification channel identified by id.

Example request:

http Copy

```
DELETE /api/alert-notifications/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{
  "message": "Notification deleted"
}
```

Test notification channel

Sends a test notification message for the given notification channel type and settings. You can find the full list of supported notifiers at the alert notifiers page.

`POST /api/alert-notifications/test`

Example request:

http Copy

```
POST /api/alert-notifications/test HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "type":  "email",
  "settings": {
    "addresses": "dev@grafana.com"
  }
}
```

Example response:

http

Copy

```
HTTP/1.1 200
Content-Type: application/json

{
  "message": "Test notification sent"
}
```

Library Element API

Identifier (id) vs unique identifier (uid)

The identifier (ID) of a library element is an auto-incrementing numeric value that is unique per Grafana install.

The unique identifier (UID) of a library element uniquely identifies library elements between multiple Grafana installs. It's automatically generated unless you specify it during library element creation. The UID provides consistent URLs for accessing library elements and when syncing library elements between multiple Grafana installs.

The maximum length of a UID is 40 characters.

Get all library elements

`GET /api/library-elements`

Returns a list of all library elements the authenticated user has permission to view. Use the `perPage` query parameter to control the maximum number of library elements returned; the default limit is 100. You can also use the `page` query parameter to fetch library elements from any page other than the first one.

Query parameters:

- `searchString`: Part of the name or description searched for.
- `kind`: Kind of element to search for. Use `1` for library panels or `2` for library variables.
- `sortDirection`: Sort order of elements. Use `alpha-asc` for ascending and `alpha-desc` for descending sort order.
- `typeFilter`: A comma separated list of types to filter the elements by.
- `excludeUid`: Element UID to exclude from search results.
- `folderFilter`: A comma separated list of folder IDs to filter the elements by.

- `perPage`: The number of results per page; default is 100.
- `page`: The page for a set of records, given that only `perPage` records are returned at a time. Numbering starts at `1`.

Example Request:

```http
GET /api/library-elements?perPage=10 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{
  "result": {
    "totalCount": 15,
    "page": 1,
    "perPage": 10
    "elements": [
      {
          "id": 25,
          "orgId": 1,
          "folderId": 0,
          "uid": "V--OrYHnz",
          "name": "API docs Example",
          "kind": 1,
          "type": "text",
          "description": "",
          "model": {...},
```

Status Codes:

- `200`: Found
- `401`: Unauthorized

Get library element by uid

`GET /api/library-elements/:uid`

Returns a library element with the given UID.

Example Request:

```http
GET /api/library-elements/V--OrYHnz HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{
    "result": {
      "id": 25,
      "orgId": 1,
      "folderId": 0,
      "uid": "V--OrYHnz",
      "name": "API docs Example",
      "kind": 1,
      "type": "text",
      "description": "",
      "model": {...},
      "version": 1,
      "meta": {
          "folderName": "General",
          "folderUid": "",
          "connectedDashboards": 1,
```

Expand code

Status Codes:

- `200`: Found
- `401`: Unauthorized
- `404`: Library element not found

Get library element by name

`GET /api/library-elements/name/:name`

Returns a library element with the given name

Example Request:

```http
GET /api/library-elements/name/API docs Example HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{
    "result": [
        {
            "id": 25,
            "orgId": 1,
            "folderId": 0,
            "uid": "V--OrYHnz",
            "name": "API docs Example",
            "kind": 1,
            "type": "text",
            "description": "",
            "model": {...},
            "version": 1,
            "meta": {
                "folderName": "General",
                "folderUid": "",
```

Expand code

Status Codes:

- `200`: Found
- `401`: Unauthorized
- `404`: Library element not found

Get library element connections

`GET /api/library-elements/:uid/connections`

Returns a list of connections for a library element based on the UID specified.

Example Request:

```http
GET /api/library-elements/V--OrYHnz/connections HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{
    "result": [
        {
            "id": 148,
            "kind": 1,
            "elementId": 25,
            "connectionId": 527,
            "connectionUid": "dHEquNzGz",
            "created": "2021-09-27T10:00:07+02:00",
            "createdBy": {
                "id": 1,
                "name": "admin",
                "avatarUrl": "/avatar/46d229b033af06a191ff2267bca9ae56"
            }
        }
    ]
```

Expand code

Status Codes:

- `200`: Found
- `401`: Unauthorized
- `404`: Library element not found

Create library element

`POST /api/library-elements`

Creates a new library element.

JSON Body schema:

- `folderId`: ID of the folder where the library element is stored. It is deprecated since Grafana v9
- `folderUid`: Optional, the UID of the folder where the library element is stored, empty string when it is at the root level.
- `name`: Optional, the name of the library element.
- `model`: The JSON model for the library element.
- `kind`: Kind of element to create, Use `1` for library panels or `2` for library variables.
- `uid`: Optional, the unique identifier.

Example Request:

http Copy

```
POST /api/library-elements HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "uid": "nErXDvCkzz",
  "folderUid": "",
  "name": "Example library panel",
  "model": {...},
  "kind": 1
}
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{
    "result": {
        "id": 28,
        "orgId": 1,
        "folderId": 0,
        "folderUid": "",
        "uid": "nErXDvCkzz",
        "name": "Example library panel",
```

```
        "kind": 1,
        "type": "",
        "description": "",
        "model": {...},
        "version": 1,
        "meta": {
            "folderName": "General",
```

Expand code

Status Codes:

- `200`: Created
- `400`: Errors (for example, name or UID already exists, invalid JSON, missing or invalid fields, and so on).
- `401`: Unauthorized
- `403`: Access denied

Update library element

`PATCH /api/library-elements/:uid`

Updates an existing library element identified by uid.

JSON Body schema:

- `folderId`: ID of the folder where the library element is stored. It is deprecated since Grafana v9
- `folderUid`: UID of the folder where the library element is stored, empty string when it is at the root level.
- `name`: Name of the library element.
- `model`: The JSON model for the library element.
- `kind`: Kind of element to create. Use `1` for library panels or `2` for library variables.
- `version`: Version of the library element you are updating.
- `uid`: Optional, the unique identifier.

Example Request:

http

Copy

```
PATCH /api/library-elements/nErXDvCkzz HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "name": "Renamed library panel",
```

```
  "kind": 1,
  "version": 1
}
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{
    "result": {
        "id": 28,
        "orgId": 1,
        "folderId": 0,
        "folderUid": "",
        "uid": "nErXDvCkzz",
        "name": "Renamed library panel",
        "kind": 1,
        "type": "",
        "description": "",
        "model": {
            "description": "",
            "type": ""
        },
        "version": 2,
```

Expand code

Status Codes:

- `200`: Updated
- `400`: Errors (for example, name or UID already exists, invalid JSON, missing or invalid fields, and so on).
- `401`: Unauthorized
- `403`: Access denied
- `404`: Library element not found
- `412`: Version mismatch

Delete library element

`DELETE /api/library-elements/:uid`

Deletes an existing library element as specified by the UID. This operation cannot be reverted.

> **NOTE**
>
> You cannot delete a library element that is connected. This operation cannot be reverted.

Example Request:

```http
DELETE /api/library-elements/nErXDvCkzz HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{
    "message": "Library element deleted",
    "id": 28
}
```

Status Codes:

- `200`: Deleted
- `401`: Unauthorized
- `400`: Bad request
- `403`: Access denied
- `404`: Library element not found

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Enterprise Open source

Enterprise License API

Licensing is only available in Grafana Enterprise. Read more about Grafana Enterprise.

If you are running Grafana Enterprise, for some endpoints you'll need to have specific permissions. Refer to Role-based access control permissions for more information.

Check license availability

Note: Available in Grafana Enterprise v7.4+.

`GET /api/licensing/check`

Checks if a valid license is available.

Required permissions

See note in the introduction for an explanation.

Action	Scope
licensing:read	n/a

Examples

Example request:

```http
GET /api/licensing/check
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response:

```http
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 4

true
```

Status codes:

- **200** - OK

Add license

> **NOTE**
> Available in Grafana Enterprise v7.4+.

`POST /api/licensing/token`

Applies a license to a Grafana instance.

Required permissions

See note in the introduction for an explanation.

Action	Scope
licensing:write	n/a

Examples

Example request:

```http
POST /licensing/token
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{"token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0aGlzIjoiaXMiLCJub3QiOiJhIiwidmFsaWQiOiJsaWNlbnN
```

Example response:

```http
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 357

{
  "jti":"2",
  "iss":"https://grafana.com",
  "sub":"https://play.grafana.org/"
  "lid":"1",
  "included_users":15,
  "lic_exp_warn_days":30,
  "tok_exp_warn_days":2,
  "update_days":1,
  "prod":["grafana-enterprise"],
  "company":"Grafana Labs"
}
```

The response is a JSON blob available for debugging purposes. The available fields may change at any time without any prior notice.

Status Codes:

- **200** - OK
- **400** - Bad request
- **500** - Internal server error (refer to server logs for more details)

Manually force license refresh

> **NOTE**

Available in Grafana Enterprise v7.4+.

`POST /api/licensing/token/renew`

Manually ask license issuer for a new token.

Required permissions

See note in the introduction for an explanation.

Action	Scope
licensing:write	n/a

Examples

Example request:

http

Copy

```
POST /api/licensing/token/renew
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{}
```

Example response:

http

Copy

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 357

{
  "jti":"2",
  "iss":"https://grafana.com",
  "sub":"https://play.grafana.org/"
  "lid":"1",
  "included_users":15,
  "lic_exp_warn_days":30,
  "tok_exp_warn_days":2,
  "update_days":1,
```

```
  "prod":["grafana-enterprise"],
  "company":"Grafana Labs"
}
```

The response is a JSON blob available for debugging purposes. The available fields may change at any time without any prior notice.

Status Codes:

- **200** - OK
- **401** - Unauthorized
- **403** - Access denied

Remove license from database

> **NOTE**
> Available in Grafana Enterprise v7.4+.

`DELETE /api/licensing/token`

Removes the license stored in the Grafana database.

Required permissions

See note in the introduction for an explanation.

Action	Scope
licensing:delete	n/a

Examples

Example request:

http Copy

```
DELETE /api/licensing/token
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

```
{"instance": "http://play.grafana.org/"}
```

JSON Body schema:

- **instance** – Root URL for the instance for which the license should be deleted. Required.

Example response:

```http
HTTP/1.1 202 Accepted
Content-Type: application/json
Content-Length: 2

{}
```

Status codes:

- **202** - Accepted, license removed or did not exist.
- **401** - Unauthorized
- **403** - Access denied
- **422** - Unprocessable entity, incorrect instance name provided.

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Organization API

The Organization HTTP API is divided in two resources, `/api/org` (current organization) and `/api/orgs` (admin organizations). One big difference between these are that the admin of all organizations API only works with basic authentication, see Admin Organizations API for more information.

If you are running Grafana Enterprise, for some endpoints you'll need to have specific permissions. Refer to Role-based access control permissions for more information.

Current Organization API

Get current Organization

`GET /api/org/`

Required permissions

See note in the introduction for an explanation.

Action	Scope
orgs:read	N/A

Example Request:

```
http
```

Copy

```
GET /api/org/ HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{
  "id":1,
  "name":"Main Org."
}
```

Get all users within the current organization

`GET /api/org/users`

Returns all org users within the current organization. Accessible to users with org admin role.

Required permissions

See note in the introduction for an explanation.

Action	Scope
org.users:read	users:*

Example Request:

http Copy

```
GET /api/org/users HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

[
  {
    "orgId": 1,
    "userId": 1,
    "email": "admin@localhost",
    "avatarUrl": "/avatar/46d229b033af06a191ff2267bca9ae56",
    "login": "admin",
    "role": "Admin",
    "lastSeenAt": "2019-08-09T11:02:49+02:00",
    "lastSeenAtAge": "< 1m"
  }
]
```

Get all users within the current organization (lookup)

`GET /api/org/users/lookup`

Returns all org users within the current organization, but with less detailed information. Accessible to users with org admin role, admin in any folder or admin of any team. Mainly used by Grafana UI for providing list of users when adding team members and when editing folder/dashboard permissions.

Required permissions

See note in the introduction for an explanation.

Action	Scope
org.users:read	users:*

Example Request:

http Copy

```
GET /api/org/users/lookup HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

[
  {
    "userId": 1,
    "login": "admin",
    "avatarUrl": "/avatar/46d229b033af06a191ff2267bca9ae56"
  }
]
```

Updates the given user

`PATCH /api/org/users/:userId`

Required permissions

See note in the introduction for an explanation.

Action	Scope
org.users:write	users:*

Example Request:

http Copy

```
PATCH /api/org/users/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "role": "Viewer",
}
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{"message":"Organization user updated"}
```

Delete user in current organization

`DELETE /api/org/users/:userId`

Required permissions

See note in the introduction for an explanation.

Action	Scope
org.users:remove	users:*

Example Request:

```http
DELETE /api/org/users/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{"message":"User removed from organization"}
```

Update current Organization

`PUT /api/org`

Required permissions

See note in the introduction for an explanation.

Action	Scope
orgs:write	N/A

Example Request:

```http
PUT /api/org HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "name":"Main Org."
}
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{"message":"Organization updated"}
```

Add a new user to the current organization

`POST /api/org/users`

Adds a global user to the current organization.

Required permissions

See note in the introduction for an explanation.

Action	Scope
org.users:add	users:*

Example Request:

```http
```

```
POST /api/org/users HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "role": "Admin",
  "loginOrEmail": "admin"
}
```

Example Response:

```
http
HTTP/1.1 200
Content-Type: application/json

{"message":"User added to organization","userId":1}
```

Admin Organizations API

The Admin Organizations HTTP API does not currently work with an API Token. API Tokens are currently only linked to an organization and an organization role. They cannot be given the permission of server admin, only users can be given that permission. So in order to use these API calls you will have to use Basic Auth and the Grafana user must have the Grafana Admin permission (The default admin user is called `admin` and has permission to use this API).

Get Organization by Id

`GET /api/orgs/:orgId`

Only works with Basic Authentication (username and password), see introduction.

Required permissions

See note in the introduction for an explanation.

Action	Scope
orgs:read	N/A

Example Request:

```
http
```

```
GET /api/orgs/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{
  "id":1,
  "name":"Main Org.",
  "address":{
    "address1":"",
    "address2":"",
    "city":"",
    "zipCode":"",
    "state":"",
    "country":""
  }
}
```

Get Organization by Name

`GET /api/orgs/name/:orgName`

Only works with Basic Authentication (username and password), see introduction.

Required permissions

See note in the introduction for an explanation.

Action	Scope	Note
orgs:read	N/A	Needs to be assigned globally.

Example Request:

http Copy

```
```

```
GET /api/orgs/name/Main%20Org%2E HTTP/1.1
Accept: application/json
Content-Type: application/json
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{
  "id":1,
  "name":"Main Org.",
  "address":{
    "address1":"",
    "address2":"",
    "city":"",
    "zipCode":"",
    "state":"",
    "country":""
  }
}
```

Create Organization

`POST /api/orgs`

Only works with Basic Authentication (username and password), see introduction.

Required permissions

See note in the introduction for an explanation.

Action	Scope	Note
orgs:create	N/A	Needs to be assigned globally.

Example Request:

http Copy

```
POST /api/orgs HTTP/1.1
Accept: application/json
Content-Type: application/json

{
  "name":"New Org."
}
```

Note: The api will work in the following two ways

1. Need to set GF_USERS_ALLOW_ORG_CREATE=true
2. Set the config value users.allow_org_create to true in ini file

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{
  "orgId":"1",
  "message":"Organization created"
}
```

Search all Organizations

`GET /api/orgs?perpage=10&page=1`

Only works with Basic Authentication (username and password), see introduction.

Required permissions

See note in the introduction for an explanation.

Action	Scope	Note
orgs:read	N/A	Needs to be assigned globally.

Example Request:

http Copy

```
GET /api/orgs HTTP/1.1
Accept: application/json
Content-Type: application/json
```

Note: The api will only work when you pass the admin name and password to the request HTTP URL, like http://admin:admin@localhost:3000/api/orgs

Default value for the `perpage` parameter is `1000` and for the `page` parameter is `0`.

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

[
  {
    "id":1,
    "name":"Main Org."
  }
]
```

Update Organization

`PUT /api/orgs/:orgId`

Update Organization, fields *Address 1*, *Address 2*, *City* are not implemented yet. Only works with Basic Authentication (username and password), see introduction.

Required permissions

See note in the introduction for an explanation.

Action	Scope
orgs:write	N/A

Example Request:

http Copy

```
PUT /api/orgs/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
```

```
{
  "name":"Main Org 2."
}
```

Example Response:

```
http
HTTP/1.1 200
Content-Type: application/json

{"message":"Organization updated"}
```

Delete Organization

`DELETE /api/orgs/:orgId`

Only works with Basic Authentication (username and password), see introduction.

Required permissions

See note in the introduction for an explanation.

Action	Scope
orgs:delete	N/A

Example Request:

```
http
DELETE /api/orgs/1 HTTP/1.1
Accept: application/json
```

Example Response:

```
http
HTTP/1.1 200
Content-Type: application/json

{"message":"Organization deleted"}
```

Get Users in Organization

`GET /api/orgs/:orgId/users`

Only works with Basic Authentication (username and password), see introduction.

Required permissions

See note in the introduction for an explanation.

Action	Scope
org.users:read	users:*

Example Request:

```http
GET /api/orgs/1/users HTTP/1.1
Accept: application/json
Content-Type: application/json
```

Note: The api will only work when you pass the admin name and password to the request HTTP URL, like http://admin:admin@localhost:3000/api/orgs/1/users

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json
[
  {
    "orgId":1,
    "userId":1,
    "email":"admin@mygraf.com",
    "login":"admin",
    "role":"Admin"
  }
]
```

Add User in Organization

`POST /api/orgs/:orgId/users`

Only works with Basic Authentication (username and password), see introduction.

Required permissions

See note in the introduction for an explanation.

Action	Scope
org.users:add	users:*

Example Request:

```http
POST /api/orgs/1/users HTTP/1.1
Accept: application/json
Content-Type: application/json

{
  "loginOrEmail":"user",
  "role":"Viewer"
}
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{"message":"User added to organization", "userId": 1}
```

Update Users in Organization

`PATCH /api/orgs/:orgId/users/:userId`

Only works with Basic Authentication (username and password), see introduction.

Required permissions

See note in the introduction for an explanation.

Action	Scope
org.users:write	users:*

Example Request:

```http
PATCH /api/orgs/1/users/2 HTTP/1.1
Accept: application/json
Content-Type: application/json

{
  "role":"Admin"
}
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{"message":"Organization user updated"}
```

Delete User in Organization

`DELETE /api/orgs/:orgId/users/:userId`

Only works with Basic Authentication (username and password), see introduction.

Required permissions

See note in the introduction for an explanation.

Action	Scope
org.users:remove	users:*

Example Request:

```http
DELETE /api/orgs/1/users/2 HTTP/1.1
Accept: application/json
```

```
Content-Type: application/json
```

Example Response:

http

Copy

```
HTTP/1.1 200
Content-Type: application/json

{"message":"User removed from organization"}
```

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Frontend Settings API

Get Settings

`GET /api/frontend/settings`

Example Request:

http Copy

```
GET /api/frontend/settings HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{
  "allowOrgCreate":true,
  "appSubUrl":"",
  "buildInfo":{
    "buildstamp":xxxxxx,
    "commit":"vyyyy",
    "version":"zzzzz"
  },
  "datasources":{
```

```
    "datasourcename":{
      "index":"grafana-dash",
      "meta":{
        "annotations":true,
        "module":"plugins/datasource/grafana/datasource",
        "name":"Grafana",
```


Expand code

Login API

Renew session based on remember cookie

`GET /api/login/ping`

Example Request:

http

Copy

```
GET /api/login/ping HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

http

Copy

```
HTTP/1.1 200
Content-Type: application/json

{"message": "Logged in"}
```

Health API

Returns health information about Grafana

`GET /api/health`

Example Request

http

Copy

```
GET /api/health
Accept: application/json
```

Example Response:

http

Copy

```
HTTP/1.1 200 OK

{
  "commit": "087143285",
  "database": "ok",
  "version": "5.1.3"
}
```

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Playlist API

Search Playlist

`GET /api/playlists`

Get all existing playlist for the current organization using pagination

Example Request:

http Copy

```
GET /api/playlists HTTP/1.1
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Querystring Parameters:

These parameters are used as querystring parameters.

- **query** - Limit response to playlist having a name like this value.
- **limit** - Limit response to *X* number of playlist.

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json
[
  {
    "uid": "1",
    "name": "my playlist",
```

```
    "interval": "5m"
  }
]
```

Get one playlist

`GET /api/playlists/:uid`

Example Request:

http Copy

```
GET /api/playlists/1 HTTP/1.1
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json
{
  "uid" : "1",
  "name": "my playlist",
  "interval": "5m",
  "items": [
    {
      "id": 1,
      "playlistUid": "1",
      "type": "dashboard_by_id",
      "value": "3",
      "order": 1,
      "title":"my third dashboard"
    },
    {
      "id": 2,
      "playlistUid": "1",
      "type": "dashboard_by_tag",
```

Expand code

Get Playlist items

`GET /api/playlists/:uid/items`

Example Request:

```http
GET /api/playlists/1/items HTTP/1.1
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json
[
  {
    "id": 1,
    "playlistUid": "1",
    "type": "dashboard_by_id",
    "value": "3",
    "order": 1,
    "title":"my third dashboard"
  },
  {
    "id": 2,
    "playlistUid": "1",
    "type": "dashboard_by_tag",
    "value": "myTag",
    "order": 2,
    "title":"my other dashboard"
  }
```

Expand code

Create a playlist

`POST /api/playlists/`

Example Request:

```http
PUT /api/playlists/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

```
  {
    "name": "my playlist",
    "interval": "5m",
    "items": [
      {
        "type": "dashboard_by_id",
        "value": "3",
        "order": 1,
        "title":"my third dashboard"
      },
      {
        "type": "dashboard_by_tag",
        "value": "myTag",
        "order": 2,
```

Expand code

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json
  {
    "uid": "1",
    "name": "my playlist",
    "interval": "5m"
  }
```

Update a playlist

`PUT /api/playlists/:uid`

Example Request:

http Copy

```
PUT /api/playlists/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
  {
    "name": "my playlist",
    "interval": "5m",
    "items": [
      {
```

```
    "playlistUid": "1",
    "type": "dashboard_by_id",
    "value": "3",
    "order": 1,
    "title":"my third dashboard"
  },
  {
    "playlistUid": "1",
    "type": "dashboard_by_tag",
```


Expand code

Example Response:

http

Copy


```
HTTP/1.1 200
Content-Type: application/json
{
  "uid" : "1",
  "name": "my playlist",
  "interval": "5m",
  "items": [
    {
      "id": 1,
      "playlistUid": "1",
      "type": "dashboard_by_id",
      "value": "3",
      "order": 1,
      "title":"my third dashboard"
    },
    {
      "id": 2,
      "playlistUid": "1",
      "type": "dashboard_by_tag",
```


Expand code

Delete a playlist

`DELETE /api/playlists/:uid`

Example Request:

http

Copy


```
DELETE /api/playlists/1 HTTP/1.1
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

http

Copy

```
HTTP/1.1 200
Content-Type: application/json
{}
```

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Enterprise | Open source

Public Dashboard API

NOTE

If you're running Grafana Enterprise, you'll need to have specific permissions for some endpoints. Refer to Role-based access control permissions for more information.

Create a public dashboard

`POST /api/dashboards/uid/:uid/public-dashboards/`

Creates a new public dashboard.

Required permissions

See note in the introduction for an explanation.

Action	Scope
`dashboards.public:write`	`dashboards:uid:<dashboard UID>`

Example Request for new public dashboard:

http

Copy

```
POST /api/dashboards/uid/xCpsVuc4z/public-dashboards/ HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

```
{
    "uid": "cd56d9fd-f3d4-486d-afba-a21760e2acbe",
    "accessToken": "5c948bf96e6a4b13bd91975f9a2028b7",
    "timeSelectionEnabled": false,
    "isEnabled": true,
    "annotationsEnabled": false,
    "share": "public"
}
```

JSON Body schema:

- **uid** – Optional. Unique identifier when creating a public dashboard. If it's null, it will generate a new uid.
- **accessToken** – Optional. Unique access token. If it's null, it will generate a new access token.
- **timeSelectionEnabled** – Optional. Set to `true` to enable the time picker in the public dashboard. The default value is `false`.
- **isEnabled** – Optional. Set to `true` to enable the public dashboard. The default value is `false`.
- **annotationsEnabled** – Optional. Set to `true` to show annotations. The default value is `false`.
- **share** – Optional. Set the share mode. The default value is `public`.

Example Response:

http Copy

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 78

{
    "uid": "cd56d9fd-f3d4-486d-afba-a21760e2acbe",
    "dashboardUid": "xCpsVuc4z",
    "accessToken": "5c948bf96e6a4b13bd91975f9a2028b7",
    "createdBy": 1,
    "updatedBy": 1,
    "createdAt": "2023-09-05T15:48:21-03:00",
    "updatedAt": "2023-09-05T15:48:21-03:00",
    "timeSelectionEnabled": false,
    "isEnabled": false,
    "annotationsEnabled": false,
    "share": "public"
}
```

Status Codes:

- **200** – Created
- **400** – Errors (such as invalid json, missing or invalid fields, or dashboard is public)
- **401** – Unauthorized
- **403** – Access denied
- **404** – Dashboard not found

The error response body will have the following properties:

```http
HTTP/1.1 400 Bad request
Content-Type: application/json; charset=UTF-8
Content-Length: 107

{
    "statusCode": 400,
    "messageId": "publicdashboards.dashboardIsPublic",
    "message": "Dashboard is already public"
}
```

Update a public dashboard

`PATCH /api/dashboards/uid/:uid/public-dashboards/:publicDashboardUid`

Will update the public dashboard given the specified unique identifier (uid).

Required permissions

See note in the introduction for an explanation.

Action	Scope
`dashboards.public:write`	`dashboards:uid:<dashboard UID>`

Example Request for updating a public dashboard:

```http
PATCH /api/dashboards/uid/xCpsVuc4z/public-dashboards/cd56d9fd-f3d4-486d-afba-a21760e2acbe HTTP/1.
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

```
{
    "timeSelectionEnabled": false,
    "isEnabled": true,
    "annotationsEnabled": false,
    "share": "public"
}
```

JSON Body schema:

- **timeSelectionEnabled** – Optional. Set to `true` to enable the time picker in the public dashboard. The default value is `false`.
- **isEnabled** – Optional. Set to `true` to enable the public dashboard. The default value is `false`.
- **annotationsEnabled** – Optional. Set to `true` to show annotations. The default value is `false`.
- **share** – Optional. Set the share mode. The default value is `public`.

Example Response:

http Copy

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 78

{
    "uid": "cd56d9fd-f3d4-486d-afba-a21760e2acbe",
    "dashboardUid": "xCpsVuc4z",
    "accessToken": "5c948bf96e6a4b13bd91975f9a2028b7",
    "createdBy": 1,
    "updatedBy": 1,
    "createdAt": "2023-09-05T15:48:21-03:00",
    "updatedAt": "2023-09-05T15:48:21-03:00",
    "timeSelectionEnabled": false,
    "isEnabled": false,
    "annotationsEnabled": false,
    "share": "public"
}
```

Status Codes:

- **200** – Updated
- **400** – Errors (such as invalid json, missing or invalid fields)
- **401** – Unauthorized
- **403** – Access denied

- **404** – Public dashboard not found

The error response body will have the following properties:

```http
HTTP/1.1 400 Bad request
Content-Type: application/json; charset=UTF-8
Content-Length: 107

{
    "statusCode": 400,
    "messageId": "publicdashboards.dashboardIsPublic",
    "message": "Dashboard is already public"
}
```

Get public dashboard by dashboard uid

`GET /api/dashboards/uid/:uid/public-dashboards/`

Will return the public dashboard given the dashboard unique identifier (uid).

Required permissions

See note in the introduction for an explanation.

Action	Scope
`dashboards:read`	`dashboards:uid:<dashboard UID>`

Example Request:

```http
GET /api/dashboards/uid/xCpsVuc4z/public-dashboards/ HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```http
```

```
HTTP/1.1 200
Content-Type: application/json

{
    "uid": "e71950f3-e7dd-4d1e-aa8a-a857bc5e7d64",
    "dashboardUid": "xCpsVuc4z",
    "accessToken": "dab10f3a4fbb4342a602b03079c7ed64",
    "createdBy": 1,
    "updatedBy": 1,
    "createdAt": "2023-09-05T15:48:21-03:00",
    "updatedAt": "2023-09-05T15:48:21-03:00",
    "timeSelectionEnabled": false,
    "isEnabled": false,
    "annotationsEnabled": false,
    "share": "public"
}
```

Status Codes:

- **200** – Found
- **401** – Unauthorized
- **403** – Access denied
- **404** – Not found

Delete public dashboard by dashboard uid and public dashboard uid

`DELETE /api/dashboards/uid/:uid/public-dashboards/:publicDashboardUid`

Will delete the public dashboard given the specified unique identifier (uid).

Required permissions

See note in the introduction for an explanation.

Action	Scope
`dashboards.public:write`	`dashboards:uid:<dashboard UID>`

Example Request:

```
http
```

Copy

```
DELETE /api/dashboards/uid/xCpsVuc4z/public-dashboards/cd56d9fd-f3d4-486d-afba-a21760e2acbe HTTP/1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Status Codes:

- **200** – Deleted
- **401** – Unauthorized
- **403** – Access denied

Get a list of all public dashboards with pagination

`GET /api/dashboards/public-dashboards`

Required permissions

See note in the introduction for an explanation.

Action	Scope
`dashboards:read`	`dashboards:uid:<dashboard UID>`

Example Request:

http Copy

```
GET /api/dashboards/public-dashboards?perpage=2&page=3 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{
    "publicDashboards": [
        {
            "uid": "e9f29a3c-fcc3-4fc5-a690-ae39c97d24ba",
```

```
        "accessToken": "6c13ec1997ba48c5af8c9c5079049692",
        "title": "Datasource Shared Queries",
        "dashboardUid": "d2f21d0a-76c7-47ec-b5f3-9dda16e5a996",
        "isEnabled": true
    },
    {
        "uid": "a174f604-6fe7-47de-97b4-48b7e401b540",
        "accessToken": "d1fcff345c0f45e8a78c096c9696034a",
        "title": "Datasource with template variables",
        "dashboardUid": "51DiOw0Vz",
        "isEnabled": true
```


Expand code

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Enterprise Open source

Query and resource caching API

NOTE

If you are running Grafana Enterprise, for some endpoints you'll need to have specific permissions. Refer to Role-based access control permissions for more information.

Enable caching for a data source

`POST /api/datasources/:dataSourceUID/cache/enable`

Required permissions

See note in the introduction for an explanation.

Action	Scope
datasources.caching:write	datasources:*

Examples

Example Request:

http Copy

```
POST /api/datasources/jZrmlLCGka/cache/enable HTTP/1.1
Accept: application/json
```

```
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

http

Copy

```
HTTP/1.1 200
Content-Type: application/json

{
   "message": "Data source cache enabled",
   "dataSourceID": 1,
   "dataSourceUID": "jZrmlLCGka",
   "enabled": true,
   "ttlQueriesMs": 300000,
   "ttlResourcesMs": 300000,
   "useDefaultTTL": true,
   "defaultTTLMs": 300000,
   "created": "2023-04-21T11:49:22-04:00",
   "updated": "2023-04-24T16:30:29-04:00"
}
```

Status codes

Code	Description
200	Cache was successfully enabled for the data source
500	Unexpected error. Refer to the body and/or server logs for more details.

Disable caching for a data source

`POST /api/datasources/:dataSourceUID/cache/disable`

Required permissions

See note in the introduction for an explanation.

Action	Scope
datasources.caching:write	datasources:*

Examples

Example Request:

http

Copy

```
POST /api/datasources/jZrmlLCGka/cache/disable HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

http

Copy

```
HTTP/1.1 200
Content-Type: application/json

{
   "message": "Data source cache disabled",
   "dataSourceID": 1,
   "dataSourceUID": "jZrmlLCGka",
   "enabled": false,
   "ttlQueriesMs": 300000,
   "ttlResourcesMs": 300000,
   "useDefaultTTL": true,
   "defaultTTLMs": 0,
   "created": "2023-04-21T11:49:22-04:00",
   "updated": "2023-04-24T16:30:31-04:00"
}
```

Status codes

Code	Description
200	Cache was successfully enabled for the data source
500	Unexpected error. Refer to the body and/or server logs for more details.

Clean cache for all data sources

`POST /api/datasources/:dataSourceUID/cache/clean`

Will clean cached data for *all* data sources with caching enabled. The `dataSourceUID` specified will only be used to return the configuration for that data source.

Required permissions

See note in the introduction for an explanation.

Action	Scope
datasources.caching:write	datasources:*

Examples

Example Request:

http

Copy

```
POST /api/datasources/jZrmlLCGka/cache/clean HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

http

Copy

```
HTTP/1.1 200
Content-Type: application/json

{
   "message": "Data source cache cleaned",
   "dataSourceID": 1,
   "dataSourceUID": "jZrmlLCGka",
   "enabled": false,
   "ttlQueriesMs": 300000,
   "ttlResourcesMs": 300000,
   "useDefaultTTL": true,
   "defaultTTLMs": 0,
   "created": "2023-04-21T11:49:22-04:00",
```

```
    "updated": "2023-04-24T16:30:31-04:00"
}
```

Status codes

Code	Description
200	Cache was successfully enabled for the data source
500	Unexpected error. Refer to the body and/or server logs for more details.

Update cache configuration for a data source

`POST /api/datasources/:dataSourceUID/cache`

Required permissions

See note in the introduction for an explanation.

Action	Scope
datasources.caching:write	datasources:*

Examples

Example Request:

http Copy

```
POST /api/datasources/jZrmlLCGka/cache HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
   "dataSourceID": 1,
   "dataSourceUID": "jZrmlLCGka",
   "enabled": true,
   "useDefaultTTL": false,
   "ttlQueriesMs": 60000,
```

```
  "ttlResourcesMs": 300000,
}
```

JSON Body Schema

Field name	Data type	Description
dataSourceID	number	The ID of the data source to configure.
dataSourceUID	string	The UID of the data source to configure.
enabled	boolean	Whether or not to enable caching for this data source.
useDefaultTTL	boolean	Whether the configured default TTL (Time-To-Live) should be used for both query and resource caching, instead of the user-specified values.
ttlQueriesMs	number	The TTL to use for query caching, in milliseconds.
ttlResourcesMs	number	The TTL to use for resource caching, in milliseconds.

Example Response:

http

Copy

```
HTTP/1.1 200
Content-Type: application/json

{
  "message": "Data source cache settings updated",
  "dataSourceID": 1,
  "dataSourceUID": "jZrmlLCGka",
  "enabled": true,
  "useDefaultTTL": false,
  "ttlQueriesMs": 60000,
  "ttlResourcesMs": 300000,
  "defaultTTLMs": 300000,
  "created": "2023-04-21T11:49:22-04:00",
  "updated": "2023-04-24T17:03:40-04:00"
}
```

Status codes

Code	Description
200	Cache was successfully enabled for the data source
400	Request errors (invalid json, missing or invalid fields, etc)
500	Unexpected error. Refer to the body and/or server logs for more details.

Get cache configuration for a data source

`GET /api/datasources/:dataSourceUID/cache`

Required permissions

See note in the introduction for an explanation.

Action	Scope
datasources.caching:read	datasources:*

Examples

Example Request:

http Copy

```
GET /api/datasources/jZrmlLCGka/cache HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{
   "message": "Data source cache settings loaded",
   "dataSourceID": 1,
   "dataSourceUID": "jZrmlLCGka",
   "enabled": true,
   "useDefaultTTL": false,
```

```
    "ttlQueriesMs": 60000,
    "ttlResourcesMs": 300000,
    "defaultTTLMs": 300000,
    "created": "2023-04-21T11:49:22-04:00",
    "updated": "2023-04-24T17:03:40-04:00"
}
```


Status codes

Code	Description
200	Cache was successfully enabled for the data source
500	Unexpected error. Refer to the body and/or server logs for more details.

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Query history API

This API can be used to add queries to Query history. It requires that the user is logged in and that Query history feature is enabled in config file.

Add query to Query history

`POST /api/query-history`

Adds query to query history.

Example request:

http Copy

```
POST /api/query-history HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
{
  "datasourceUid": "PE1C5CBDA0504A6A3",
  "queries": [
    {
        "refId": "A",
        "key": "Q-87fed8e3-62ba-4eb2-8d2a-4129979bb4de-0",
        "scenarioId": "csv_content",
        "datasource": {
            "type": "testdata",
            "uid": "PD8C576611E62080A"
        }
    }
```

```
]
}
```

JSON body schema:

- **datasourceUid** – Data source uid.
- **queries** – JSON of query or queries.

Example response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json
{
  "result": {
    "uid": "Ahg678z",
    "datasourceUid": "PE1C5CBDA0504A6A3",
    "createdBy": 1,
    "createdAt": 1643630762,
    "starred": false,
    "comment": "",
    "queries": [
      {
        "refId": "A",
        "key": "Q-87fed8e3-62ba-4eb2-8d2a-4129979bb4de-0",
        "scenarioId": "csv_content",
        "datasource": {
            "type": "testdata",
            "uid": "PD8C576611E62080A"
        }
```

Expand code

Status codes:

- **200** – OK
- **400** - Errors (invalid JSON, missing or invalid fields)
- **401** – Unauthorized
- **500** – Internal error

Query history search

`GET /api/query-history`

Returns a list of queries in the query history that matches the search criteria. Query history search supports pagination. Use the `limit` parameter to control the maximum number of queries returned;

the default limit is 100. You can also use the `page` query parameter to fetch queries from any page other than the first one.

Query parameters:

- **datasourceUid** - Filter the query history for the selected data source. To perform an "AND" filtering with multiple data sources, specify the data source parameter using the following format: `datasourceUid=uid1&datasourceUid=uid2`.
- **searchString** – Filter the query history based on the content.
- **sort** - Specify the sorting order. Sorting can be `time-asc` or `time-desc`. The default is `time-desc`.
- **onlyStarred** - Search for queries that are starred. Defaults to `false`.
- **page** - Search supports pagination. Specify which page number to return. Use the limit parameter to specify the number of queries per page.
- **limit** - Limits the number of returned query history items per page. The default is 100 queries per page.
- **from/to** - Specifies time range for the query history search. The time can be either epoch timestamps in milliseconds or relative using Grafana time units. For example, now-5m.

Example request for query history search:

http Copy

```
GET /api/query-history?datasourceUid="PE1C5CBDA0504A6A3"&datasourceUid="FG1C1CBDA0504A6EL"&searchS
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response for query history search:

http Copy

```
HTTP/1.1 200
Content-Type: application/json
{
  "result": {
    "totalCount": 150,
    "page": 1,
    "perPage": 100
    "queryHistory":[{
    "uid": "Ahg678z",
    "datasourceUid": "PE1C5CBDA0504A6A3",
    "createdBy": 1,
    "createdAt": 1643630762,
    "starred": false,
```

```
    "comment": "",
    "queries": [
      {
        "refId": "A",
        "key": "Q-87fed8e3-62ba-4eb2-8d2a-4129979bb4de-0",
```


Expand code

Status codes:

- **200** – OK
- **401** – Unauthorized
- **500** – Internal error

Delete query from Query history by UID

`DELETE /api/query-history/:uid`

Deletes the query in query history that matches the specified uid. It requires that the user is logged in and that Query history feature is enabled in config file.

Example Request:

http

Copy

```
DELETE /api/query-history/P8zM2I1nz HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

http

Copy

```
HTTP/1.1 200
Content-Type: application/json

{
    "message": "Query deleted",
    "id": 28
}
```

Status codes:

- **200** – OK
- **401** – Unauthorized
- **500** – Internal error

Update comment of query in Query history by UID

`PATCH /api/query-history/:uid`

Updates comment of a query with a specific uid that is stored in the query history.

Query parameters:

- **comment** – New comment that will be added to the specified query.

Example Request:

http Copy

```
PATCH /api/query-history/P8zM2I1nz HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
{
  "comment": "Debugging query",
}
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json
{
  "result": {
    "uid": "P8zM2I1nz",
    "datasourceUid": "PE1C5CBDA0504A6A3",
    "createdBy": 1,
    "createdAt": 1643630762,
    "starred": false,
    "comment": "Debugging query",
    "queries": [
      {
        "refId": "A",
        "key": "Q-87fed8e3-62ba-4eb2-8d2a-4129979bb4de-0",
        "scenarioId": "csv_content",
        "datasource": {
            "type": "testdata",
            "uid": "PD8C576611E62080A"
        }
```

Expand code

Status codes:

- **200** – OK
- **400** - Errors (invalid JSON, missing or invalid fields)
- **401** – Unauthorized
- **500** – Internal error

Star query in Query history

`POST /api/query-history/star/:uid`

Stars query in query history.

Example request:

http Copy

```
POST /api/query-history/star/P8zM2I1nz HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json
{
  "result": {
    "uid": "P8zM2I1nz",
    "datasourceUid": "PE1C5CBDA0504A6A3",
    "createdBy": 1,
    "createdAt": 1643630762,
    "starred": false,
    "comment": "Debugging query",
    "queries": [
      {
        "refId": "A",
        "key": "Q-87fed8e3-62ba-4eb2-8d2a-4129979bb4de-0",
        "scenarioId": "csv_content",
        "datasource": {
            "type": "testdata",
            "uid": "PD8C576611E62080A"
```

Expand code

Status codes:

- **200** – OK
- **401** – Unauthorized
- **500** – Internal error

Unstar query in Query history

`DELETE /api/query-history/star/:uid`

Removes stars from query in query history.

Example request:

http Copy

```
DELETE /api/query-history/star/P8zM2I1nz  HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json
{
  "result": {
    "uid": "P8zM2I1nz",
    "datasourceUid": "PE1C5CBDA0504A6A3",
    "createdBy": 1,
    "createdAt": 1643630762,
    "starred": false,
    "comment": "Debugging query",
    "queries": [
      {
        "refId": "A",
        "key": "Q-87fed8e3-62ba-4eb2-8d2a-4129979bb4de-0",
        "scenarioId": "csv_content",
        "datasource": {
            "type": "testdata",
            "uid": "PD8C576611E62080A"
        }
```

Expand code

Status codes:

- **200** – OK
- **401** – Unauthorized
- **500** – Internal error

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Enterprise | Open source

RBAC API

Role-based access control API is only available in Grafana Cloud or Grafana Enterprise. Read more about Grafana Enterprise.

The API can be used to create, update, delete, get, and list roles.

To check which basic or fixed roles have the required permissions, refer to RBAC role definitions.

Get status

`GET /api/access-control/status`

Returns an indicator to check if role-based access control is enabled or not.

Required permissions

Action	Scope
status:accesscontrol	services:accesscontrol

Example request

http

Copy

```
GET /api/access-control/status
Accept: application/json
```

```
Content-Type: application/json
```

Example response

```http
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8

{
  "enabled": true
}
```

Status codes

Code	Description
200	Returned a flag indicating if the role-based access control is enabled or no.
403	Access denied
404	Not found, an indication that role-based access control is not available at all.
500	Unexpected error. Refer to body and/or server logs for more details.

Create and manage custom roles

Get all roles

`GET /api/access-control/roles`

Gets all existing roles. The response contains all global and organization local roles, for the organization which user is signed in.

Query Parameters:

- `includeHidden`: Optional. Set to `true` to include roles that are `hidden`.

Required permissions

Action	Scope
roles:read	roles:*

Example request

http

Copy

```
GET /api/access-control/roles
Accept: application/json
Content-Type: application/json
```

Example response

http

Copy

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8

[
    {
        "version": 3,
        "uid": "XvHQJq57z",
        "name": "fixed:reports:reader",
        "displayName": "Report reader",
        "description": "Read all reports and shared report settings.",
        "group": "Reports",
        "updated": "2021-11-19T10:48:00+01:00",
        "created": "2021-11-19T10:48:00+01:00",
        "global": false
    },
    {
        "version": 5,
        "uid": "vi9mlLjGz",
        "name": "fixed:datasources.permissions:writer",
```

Expand code

Status codes

Code	Description
200	Global and organization local roles are returned.

Code	Description
403	Access denied
500	Unexpected error. Refer to body and/or server logs for more details.

Get a role

`GET /api/access-control/roles/:uid`

Get a role for the given UID.

Required permissions

Action	Scope
roles:read	roles:*

Example request

http Copy

```
GET /api/access-control/roles/PYnDO3rMk
Accept: application/json
Content-Type: application/json
```

Example response

http Copy

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8

{
    "version": 4,
    "uid": "6dNwJq57z",
    "name": "fixed:reports:writer",
    "displayName": "Report writer",
    "description": "Create, read, update, or delete all reports and shared report settings.",
    "group": "Reports",
    "permissions": [
        {
```

```
        "action": "reports:delete",
        "scope": "reports:*",
        "updated": "2021-11-19T10:48:00+01:00",
        "created": "2021-11-19T10:48:00+01:00"
    },
    {
```


Expand code

Status codes

Code	Description
200	Role is returned.
403	Access denied.
500	Unexpected error. Refer to body and/or server logs for more details.

Create a new custom role

`POST /api/access-control/roles`

Creates a new custom role and maps given permissions to that role. Note that roles with the same prefix as Fixed roles can't be created.

Required permissions

`permissions:type:delegate` scope ensures that users can only create custom roles with the same, or a subset of permissions which the user has. For example, if a user does not have required permissions for creating users, they won't be able to create a custom role which allows to do that. This is done to prevent escalation of privileges.

Action	Scope
roles:write	permissions:type:delegate

Example request

http

Copy

```
POST /api/access-control/roles
Accept: application/json
Content-Type: application/json
```

```
{
    "version": 1,
    "uid": "jZrmlLCGka",
    "name": "custom:delete:roles",
    "displayName": "custom delete roles",
    "description": "My custom role which gives users permissions to delete roles",
    "group":"My Group",
    "displayName": "My Custom Role",
    "global": false,
    "permissions": [
        {
            "action": "roles:delete",
            "scope": "permissions:type:delegate"
        }
    ]
```


Expand code

JSON body schema

Field Name	Date Type	Required	Description
uid	string	No	UID of the role. If not present, the UID will be automatically created for you and returned in response. Refer to the Custom roles for more information.
global	boolean	No	A flag indicating if the role is global or not. If set to `false`, the default org ID of the authenticated user will be used from the request.
version	number	No	Version of the role. If not present, version 0 will be assigned to the role and returned in the response. Refer to the Custom roles for more information.
name	string	Yes	Name of the role. Refer to Custom roles for more information.
description	string	No	Description of the role.
displayName	string	No	Display name of the role, visible in the UI.
group	string	No	The group name the role belongs to.
hidden	boolean	No	Specify whether the role is hidden or not. If set to `true`, then the role does not show in the role picker. It will not be listed by API endpoints unless explicitly specified.
permissions	Permission	No	If not present, the role will be created without any permissions.

Permission

Field Name	Data Type	Required	Description
action	string	Yes	Refer to Custom role actions and scopes for full list of available actions.
scope	string	No	If not present, no scope will be mapped to the permission. Refer to Custom role actions and scopes for full list of available scopes.

Example response

http

Copy

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8

{
    "version": 2,
    "uid": "jZrmlLCGka",
    "name": "custom:delete:create:roles",
    "displayName": "custom delete create roles",
    "description": "My custom role which gives users permissions to delete and create roles",
    "group":"My Group",
    "displayName": "My Custom Role",
    "global": false,
    "permissions": [
        {
            "action": "roles:delete",
            "scope": "permissions:type:delegate",
            "updated": "2021-05-13T23:19:46+02:00",
            "created": "2021-05-13T23:19:46+02:00"
        }
```

Expand code

Create role validation errors

Permission validation only occurs when permission validation is enabled (`rbac.permission_validation_enabled = true`).

It has been enabled by default since Grafana 10.2.

Invalid action

The following example shows a request with an invalid action. The action `serviceaccounts.permissions:reader` is not a valid action. The valid action should be

`serviceaccounts.permissions:read`.

http Copy

```
POST /api/access-control/roles HTTP/1.1
Content-Type: application/json
{
    "Name": "Read Service Account with id 6",
    "Permissions": [
            {
            "action": "serviceaccounts.permissions:reader",
            "scope": "serviceaccounts:uid:6"
        }
    ]
}
```

http Copy

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
{
    "extra": {
        "validationError": "the provided action was not found in the list of valid actions: servic
    },
    "message": "Permission contains an invalid action",
    "messageId": "accesscontrol.permission-invalid-action",
    "statusCode": 400,
    "traceID": ""
}
```

Invalid scope

The following example shows a request with an invalid scope. The scope `serviceaccounts:serviceaccount6` is not a valid scope for the action `serviceaccounts.permissions:read`. The valid scopes for this action are `*`, `serviceaccounts:*` and `serviceaccounts:id:*`.

http Copy

```
POST /api/access-control/roles HTTP/1.1
Content-Type: application/json
{
    "Name": "Read Service Account with id 6",
    "Permissions": [
            {
```

```
            "action": "serviceaccounts.permissions:read",
            "scope": "serviceaccounts:serviceaccount6"
        }
    ]
}
```

http

Copy

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
{
    "extra": {
        "validationError": "unknown scope: serviceaccounts:serviceaccount6 for action: serviceacc
    },
    "message": "Invalid scope",
    "messageId": "accesscontrol.permission-invalid-scope",
    "statusCode": 400,
    "traceID": ""
}
```

Status codes

Code	Description
200	Role is updated.
400	Bad request (invalid json, missing content-type, missing or invalid fields, etc.).
403	Access denied (one of the specified permissions is not assigned to the requester)
500	Unexpected error. Refer to body and/or server logs for more details.

Update a role

`PUT /api/access-control/roles/:uid`

Update the role with the given UID, and its permissions. The operation is idempotent and all permissions of the role will be replaced based on the request content. You need to increment the version of the role with each update, otherwise the request will fail.

You can update `custom` roles and `basic` roles permissions. However `fixed` roles cannot be updated.

Required permissions

`permissions:type:delegate` scope ensures that users can only update custom roles with the same, or a subset of permissions which the user has. For example, if a user does not have required permissions for creating users, they won't be able to update a custom role which allows to do that. This is done to prevent escalation of privileges.

Action	Scope
roles:write	permissions:type:delegate

Example request

http

Copy

```
PUT /api/access-control/roles/jZrmlLCGka
Accept: application/json
Content-Type: application/json

{
    "version": 3,
    "name": "custom:delete:write:roles",
    "displayName": "custom delete write roles",
    "description": "My custom role which gives users permissions to delete and write roles",
    "group":"My Group",
    "displayName": "My Custom Role",
    "global": false,
    "permissions": [
        {
            "action": "roles:delete",
            "scope": "permissions:type:delegate"
        },
        {
            "action": "roles:write",
```

Expand code

JSON body schema

Field Name	Data Type	Required	Description
version	number	Yes	Version of the role. Must be incremented for update to work.
name	string	Yes	Name of the role.

Field Name	Data Type	Required	Description
description	string	No	Description of the role.
displayName	string	No	Display name of the role, visible in the UI.
group	string	No	The group name the role belongs to.
hidden	boolean	No	Specify whether the role is hidden or not. If set to `true`, then the role does not show in the role picker. It will not be listed by API endpoints unless explicitly specified.
permissions	List of Permissions	No	The full list of permissions for the role after the update.

Permission

Field Name	Data Type	Required	Description
action	string	Yes	Refer to Custom role actions and scopes for full list of available actions.
scope	string	No	If not present, no scope will be mapped to the permission. Refer to Custom role actions and scopes for full list of available scopes.

Example response

http Copy

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8

{
    "version":3,
    "uid":"jZrmlLCGka",
    "name":"custom:delete:write:roles",
    "displayName":"custom delete write roles",
    "description":"My custom role which gives users permissions to delete and write roles",
    "group":"My Group",
    "displayName": "My Custom Role",
    "permissions":[
        {
            "action":"roles:delete",
            "scope":"permissions:type:delegate",
            "updated":"2021-08-06T18:27:40+02:00",
```

```
        "created":"2021-08-06T18:27:40+02:00"
    },
    {
```


Expand code

Update role validation errors

Permission validation only occurs when permission validation is enabled (`rbac.permission_validation_enabled = true`).

> It has been enabled by default since Grafana 10.2.

For more information, refer to Create role validation errors.

Status codes

Code	Description
200	Role is updated.
400	Bad request (invalid json, missing content-type, missing or invalid fields, etc.).
403	Access denied (one of the specified permissions is not assigned to the requester)
404	Role was not found to update.
500	Unexpected error. Refer to body and/or server logs for more details.

Delete a custom role

`DELETE /api/access-control/roles/:uid?force=false`

Delete a role with the given UID, and it's permissions. If the role is assigned, the deletion operation will fail, unless the `force` query param is set to `true`, and in that case all assignments will also be deleted.

Required permissions

`permissions:type:delegate` scope ensures that users can only delete a custom role with the same, or a subset of permissions which the user has. For example, if a user does not have required permissions for creating users, they won't be able to delete a custom role which allows to do that.

Action	Scope
roles:delete	permissions:type:delegate

Example request

```http
DELETE /api/access-control/roles/jZrmlLCGka?force=true&global=false
Accept: application/json
```

Query parameters

Param	Type	Required	Description
force	boolean	No	When set to `true`, the role will be deleted with all its assignments.
global	boolean	No	A flag indicating if the role is global or not. If set to false, the default org ID of the authenticated user will be used from the request. Refer to the About RBAC for more information.

Example response

```http
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8

{
    "message": "Role deleted"
}
```

Status codes

Code	Description
200	Role is deleted.
400	Bad request (invalid json, missing content-type, missing or invalid fields, etc.).
403	Access denied
500	Unexpected error. Refer to body and/or server logs for more details.

Create and remove user role assignments

List roles assigned to a user

`GET /api/access-control/users/:userId/roles`

Lists the roles that have been directly assigned to a given user. The list does not include basic roles (Viewer, Editor, Admin or Grafana Admin), and it does not include roles that have been inherited from a team.

Query Parameters:

- `includeHidden`: Optional. Set to `true` to include roles that are `hidden`.

Required permissions

Action	Scope
users.roles:read	users:id:`<user ID>`

Example request

http Copy

```
GET /api/access-control/users/1/roles
Accept: application/json
```

Example response

http Copy

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8

[
    {
        "version": 4,
        "uid": "6dNwJq57z",
        "name": "fixed:reports:writer",
        "displayName": "Report writer",
        "description": "Create, read, update, or delete all reports and shared report settings.",
```

```
        "group": "Reports",
        "updated": "2021-11-19T10:48:00+01:00",
        "created": "2021-11-19T10:48:00+01:00",
        "global": false
    }
]
```


Status codes

Code	Description
200	Set of assigned roles is returned.
403	Access denied.
500	Unexpected error. Refer to body and/or server logs for more details.

List your permissions

`GET /api/access-control/user/permissions`

Lists the permissions granted to the signed in user.

Required permissions

No permission is required.

Query parameters

Param	Type	Required	Description
reloadcache	boolean	No	A flag to reload the permission cache.

Example request

http

Copy

```
GET /api/access-control/user/permissions
Accept: application/json
```

Example response

http

Copy

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8

{
  "dashboards:read": ["dashboards:uid:70KrY6IVz"],
  "dashboards:write": ["dashboards:uid:70KrY6IVz"],
  "datasources.id:read": ["datasources:*"],
  "datasources:read": ["datasources:*"],
  "datasources:explore": [""],
  "datasources:query": ["datasources:uid:grafana"],
  "datasources:read": ["datasources:uid:grafana"],
  "orgs:read": [""]
}
```

Status codes

Code	Description
200	Set of assigned permissions is returned.
403	Access denied.
500	Unexpected error. Refer to body and/or server logs for more details.

List permissions assigned to a user

`GET /api/access-control/users/:userId/permissions`

Lists the permissions granted to a given user.

Required permissions

Action	Scope
users.permissions:read	users:id:`<user ID>`

Example request

```http
GET /api/access-control/users/1/permissions
Accept: application/json
```

Example response

```http
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8

[
    {
        "action": "ldap.status:read",
        "scope": ""
    },
    {
        "action": "ldap.user:read",
        "scope": ""
    }
]
```

Status codes

Code	Description
200	Set of assigned permissions is returned.
403	Access denied.
500	Unexpected error. Refer to body and/or server logs for more details.

Add a user role assignment

`POST /api/access-control/users/:userId/roles`

Assign a role to a specific user.

For bulk updates consider Set user role assignments.

Required permissions

`permissions:type:delegate` scope ensures that users can only assign roles which have same, or a subset of permissions which the user has. For example, if a user does not have required permissions for creating users, they won't be able to assign a role which will allow to do that. This is done to prevent escalation of privileges.

Action	Scope
users.roles:add	permissions:type:delegate

Example request

http Copy

```
POST /api/access-control/users/1/roles
Accept: application/json
Content-Type: application/json

{
    "global": false,
    "roleUid": "XvHQJq57z"
}
```

JSON body schema

Field Name	Data Type	Required	Description
roleUid	string	Yes	UID of the role.
global	boolean	No	A flag indicating if the assignment is global or not. If set to `false`, the default org ID of the authenticated user will be used from the request to create organization local assignment.

Example response

http Copy

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
```

```
{
    "message": "Role added to the user."
}
```

Status codes

Code	Description
200	Role is assigned to a user.
403	Access denied.
404	Role not found.
500	Unexpected error. Refer to body and/or server logs for more details.

Remove a user role assignment

`DELETE /api/access-control/users/:userId/roles/:roleUID`

Revoke a role from a user.

For bulk updates consider Set user role assignments.

Required permissions

`permissions:type:delegate` scope ensures that users can only unassign roles which have same, or a subset of permissions which the user has. For example, if a user does not have required permissions for creating users, they won't be able to unassign a role which will allow to do that. This is done to prevent escalation of privileges.

Action	Scope
users.roles:remove	permissions:type:delegate

Query parameters

Param	Type	Required	Description
global	boolean	No	A flag indicating if the assignment is global or not. If set to `false`, the default org ID of the authenticated user will be used from the request to remove assignment.

Example request

```http
DELETE /api/access-control/users/1/roles/AFUXBHKnk
Accept: application/json
```

Example response

```http
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8

{
    "message": "Role removed from user."
}
```

Status codes

Code	Description
200	Role is unassigned.
403	Access denied.
500	Unexpected error. Refer to body and/or server logs for more details.

Set user role assignments

`PUT /api/access-control/users/:userId/roles`

Update the user's role assignments to match the provided set of UIDs. This will remove any assigned roles that aren't in the request and add roles that are in the set but are not already assigned to the user.

If you want to add or remove a single role, consider using Add a user role assignment or Remove a user role assignment instead.

Required permissions

`permissions:type:delegate` scope ensures that users can only assign or unassign roles which have same, or a subset of permissions which the user has. For example, if a user does not have required

permissions for creating users, they won't be able to assign or unassign a role which will allow to do that. This is done to prevent escalation of privileges.

Action	Scope
users.roles:add	permissions:type:delegate
users.roles:remove	permissions:type:delegate

Example request

http Copy

```
PUT /api/access-control/users/1/roles
Accept: application/json
Content-Type: application/json

{
    "global": false,
    "roleUids": [
        "ZiHQJq5nk",
        "GzNQ1357k"
    ]
}
```

JSON body schema

Field Name	Date Type	Required	Description
global	boolean	No	A flag indicating if the assignment is global or not. If set to `false`, the default org ID of the authenticated user will be used from the request.
roleUids	list	Yes	List of role UIDs.
includeHidden	boolean	No	Specify whether the hidden role assignments should be updated.

Example response

http Copy

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8

{
    "message": "User roles have been updated."
}
```

Status codes

Code	Description
200	Roles have been assigned.
403	Access denied.
404	Role not found.
500	Unexpected error. Refer to body and/or server logs for more details.

Create and remove service account role assignments

List roles assigned to a service account

`GET /api/access-control/users/:serviceAccountId/roles`

Lists the roles that have been directly assigned to a given service account. The list does not include basic roles (Viewer, Editor, Admin or Grafana Admin).

Query Parameters:

- `includeHidden`: Optional. Set to `true` to include roles that are `hidden`.

Required permissions

Action	Scope
users.roles:read	users:id:`<service account ID>`

Example request

http Copy

```
GET /api/access-control/users/1/roles
Accept: application/json
```

Example response

http

Copy

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8

[
    {
        "version": 4,
        "uid": "6dNwJq57z",
        "name": "fixed:reports:writer",
        "displayName": "Report writer",
        "description": "Create, read, update, or delete all reports and shared report settings.",
        "group": "Reports",
        "updated": "2021-11-19T10:48:00+01:00",
        "created": "2021-11-19T10:48:00+01:00",
        "global": false
    }
]
```

Status codes

Code	Description
200	Set of assigned roles is returned.
403	Access denied.
500	Unexpected error. Refer to body and/or server logs for more details.

List permissions assigned to a service account

`GET /api/access-control/users/:serviceAccountId/permissions`

Lists the permissions that a given service account has.

Required permissions

Action	Scope
users.permissions:read	users:id:`<service account ID>`

Example request

http

```
GET /api/access-control/users/1/permissions
Accept: application/json
```

Example response

http

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8

[
    {
        "action": "ldap.status:read",
        "scope": ""
    },
    {
        "action": "ldap.user:read",
        "scope": ""
    }
]
```

Status codes

Code	Description
200	Set of assigned permissions is returned.
403	Access denied.
500	Unexpected error. Refer to body and/or server logs for more details.

Add a service account role assignment

`POST /api/access-control/users/:serviceAccountId/roles`

Assign a role to a specific service account.

For bulk updates consider Set service account role assignments.

Required permissions

`permissions:type:delegate` scope ensures that users can only assign roles which have same, or a subset of permissions which the user has. For example, if a user does not have required permissions for creating users, they won't be able to assign a role which will allow to do that. This is done to prevent escalation of privileges.

Action	Scope
users.roles:add	permissions:type:delegate

Example request

http

Copy

```
POST /api/access-control/users/1/roles
Accept: application/json
Content-Type: application/json

{
    "global": false,
    "roleUid": "XvHQJq57z"
}
```

JSON body schema

Field Name	Data Type	Required	Description
roleUid	string	Yes	UID of the role.
global	boolean	No	A flag indicating if the assignment is global or not. If set to `false`, the default org ID of the authenticated user will be used from the request to create organization local assignment.

Example response

```http
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8

{
    "message": "Role added to the user."
}
```

Status codes

Code	Description
200	Role is assigned to a user.
403	Access denied.
404	Role not found.
500	Unexpected error. Refer to body and/or server logs for more details.

Remove a service account role assignment

`DELETE /api/access-control/users/:serviceAccountId/roles/:roleUID`

Revoke a role from a service account.

For bulk updates consider Set service account role assignments.

Required permissions

`permissions:type:delegate` scope ensures that users can only unassign roles which have same, or a subset of permissions which the user has. For example, if a user does not have required permissions for creating users, they won't be able to unassign a role which will allow to do that. This is done to prevent escalation of privileges.

Action	Scope
users.roles:remove	permissions:type:delegate

Query parameters

Param	Type	Required	Description
global	boolean	No	A flag indicating if the assignment is global or not. If set to `false`, the default org ID of the authenticated user will be used from the request to remove assignment.

Example request

http

Copy

```
DELETE /api/access-control/users/1/roles/AFUXBHKnk
Accept: application/json
```

Example response

http

Copy

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8

{
    "message": "Role removed from user."
}
```

Status codes

Code	Description
200	Role is unassigned.
403	Access denied.
500	Unexpected error. Refer to body and/or server logs for more details.

Set service account role assignments

`PUT /api/access-control/users/:serviceAccountId/roles`

Update the service accounts's role assignments to match the provided set of UIDs. This will remove any assigned roles that aren't in the request and add roles that are in the set but are not already assigned to the service account.

If you want to add or remove a single role, consider using Add a service account role assignment or Remove a service account role assignment instead.

Required permissions

`permissions:type:delegate` scope ensures that users can only assign or unassign roles which have same, or a subset of permissions which the user has. For example, if a user does not have required permissions for creating users, they won't be able to assign or unassign a role which will allow to do that. This is done to prevent escalation of privileges.

Action	Scope
users.roles:add	permissions:type:delegate
users.roles:remove	permissions:type:delegate

Example request

http Copy

```
PUT /api/access-control/users/1/roles
Accept: application/json
Content-Type: application/json

{
    "global": false,
    "roleUids": [
        "ZiHQJq5nk",
        "GzNQ1357k"
    ]
}
```

JSON body schema

Field Name	Date Type	Required	Description
global	boolean	No	A flag indicating if the assignment is global or not. If set to `false`, the default org ID of the authenticated user

Field Name	Date Type	Required	Description
			will be used from the request.
roleUids	list	Yes	List of role UIDs.
includeHidden	boolean	No	Specify whether the hidden role assignments should be updated.

Example response

http Copy

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8

{
    "message": "User roles have been updated."
}
```

Status codes

Code	Description
200	Roles have been assigned.
403	Access denied.
404	Role not found.
500	Unexpected error. Refer to body and/or server logs for more details.

Create and remove team role assignments

List roles assigned to a team

`GET /api/access-control/teams/:teamId/roles`

Lists the roles that have been directly assigned to a given team.

Query Parameters:

- `includeHidden`: Optional. Set to `true` to include roles that are `hidden`.

Required permissions

Action	Scope
teams.roles:read	teams:id:`<team ID>`

Example request

http

Copy

```
GET /api/access-control/teams/1/roles
Accept: application/json
```

Example response

http

Copy

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8

[
    {
        "version": 4,
        "uid": "j08ZBi-nk",
        "name": "fixed:licensing:reader",
        "displayName": "Licensing reader",
        "description": "Read licensing information and licensing reports.",
        "group": "Licenses",
        "updated": "2022-02-03T14:19:50+01:00",
        "created": "0001-01-01T00:00:00Z",
        "global": false
    }
]
```

Status codes

Code	Description
200	Set of assigned roles is returned.
403	Access denied.
500	Unexpected error. Refer to body and/or server logs for more details.

Add a team role assignment

`POST /api/access-control/teams/:teamId/roles`

Assign a role to a specific team.

For bulk updates consider Set team role assignments.

Required permissions

`permissions:type:delegate` scope ensures that users can only assign roles which have same, or a subset of permissions which the user has. For example, if a user does not have the permissions required to create users, they won't be able to assign a role that contains these permissions. This is done to prevent escalation of privileges.

Action	Scope
teams.roles:add	permissions:type:delegate

Example request

http

Copy

```
POST /api/access-control/teams/1/roles
Accept: application/json
Content-Type: application/json

{
    "roleUid": "XvHQJq57z"
}
```

JSON body schema

Field Name	Data Type	Required	Description
roleUid	string	Yes	UID of the role.

Example response

http

Copy

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8

{
    "message": "Role added to the team."
}
```

Status codes

Code	Description
200	Role is assigned to a team.
403	Access denied.
404	Role not found.
500	Unexpected error. Refer to body and/or server logs for more details.

Remove a team role assignment

`DELETE /api/access-control/teams/:teams/roles/:roleUID`

Revoke a role from a team.

For bulk updates consider Set team role assignments.

Required permissions

`permissions:type:delegate` scope ensures that users can only unassign roles which have same, or a subset of permissions which the user has. For example, if a user does not have the permissions required to create users, they won't be able to assign a role that contains these permissions. This is done to prevent escalation of privileges.```

Action	Scope
teams.roles:remove	permissions:type:delegate

Example request

```http
DELETE /api/access-control/teams/1/roles/AFUXBHKnk
Accept: application/json
```

Example response

```http
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8

{
    "message": "Role removed from team."
}
```

Status codes

Code	Description
200	Role is unassigned.
403	Access denied.
500	Unexpected error. Refer to body and/or server logs for more details.

Set team role assignments

`PUT /api/access-control/teams/:teamId/roles`

Update the team's role assignments to match the provided set of UIDs. This will remove any assigned roles that aren't in the request and add roles that are in the set but are not already assigned to the user.

If you want to add or remove a single role, consider using Add a team role assignment or Remove a team role assignment instead.

Required permissions

`permissions:type:delegate` scope ensures that users can only assign or unassign roles which have same, or a subset of permissions which the user has. For example, if a user does not have required permissions for creating users, they won't be able to assign or unassign a role to a team which will allow to do that. This is done to prevent escalation of privileges.

Action	Scope
teams.roles:add	permissions:type:delegate
teams.roles:remove	permissions:type:delegate

Example request

http Copy

```
PUT /api/access-control/teams/1/roles
Accept: application/json
Content-Type: application/json

{
    "roleUids": [
        "ZiHQJq5nk",
        "GzNQ1357k"
    ]
}
```

JSON body schema

Field Name	Date Type	Required	Description
roleUids	list	Yes	List of role UIDs.
includeHidden	boolean	No	Specify whether the hidden role assignments should be updated.

Example response

http Copy

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8

{
    "message": "Team roles have been updated."
}
```

Status codes

Code	Description
200	Roles have been assigned.
403	Access denied.
404	Role not found.
500	Unexpected error. Refer to body and/or server logs for more details.

Reset basic roles to their default

`POST /api/access-control/roles/hard-reset`

`permissions:type:escalate` scope enables users to reset basic roles permissions. This could result in basic roles having permissions exceedind those of callers.

Reset basic roles permissions to their default.

Required permissions

Action	Scope
roles:write	permissions:type:escalate

Example request

http Copy

```
POST /api/access-control/roles/hard-reset
Accept: application/json
Content-Type: application/json
```

```
{
    "BasicRoles": true
}
```

JSON body schema

Field Name	Data Type	Required	Description
BasicRoles	boolean	No	Option to reset basic roles permissions.

Example response

http

Copy

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8

{
    "message": "Reset performed"
}
```

Status codes

Code	Description
200	Reset performed
500	Failed to reset basic roles

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Enterprise Open source

Reporting API

This API allows you to interact programmatically with the Reporting feature.

The Reporting API is not stabilized yet, it is still in active development and may change without prior notice.

Reporting is only available in Grafana Enterprise. Read more about Grafana Enterprise.

If you are running Grafana Enterprise, for some endpoints you'll need to have specific permissions. Refer to Role-based access control permissions for more information.

List all reports

`GET /api/reports`

Required permissions

See note in the introduction for an explanation.

Action	Scope
reports:read	reports:* reports:id:*

Example request

```http
GET /api/reports HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```http
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 1840

[
    {
        "id": 2,
        "userId": 1,
        "orgId": 1,
        "name": "Report 2",
        "recipients": "example-report@grafana.com",
        "replyTo": "",
        "message": "Hi, \nPlease find attached a PDF status report. If you have any questions, fee
        "schedule": {
            "startDate": "2022-10-02T00:00:00+02:00",
            "endDate": null,
            "frequency": "once",
            "intervalFrequency": "",
            "intervalAmount": 0,
```

Status Codes

- **200** – OK
- **401** - Authentication failed, refer to Authentication API.
- **500** – Unexpected error or server misconfiguration. Refer to server logs for more details.

Get a report

`GET /api/reports/:id`

Required permissions

See note in the introduction for an explanation.

Action	Scope
reports:read	reports:* reports:id:* reports:id:1(single report)

Example request

http

```
GET /api/reports/2 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

http

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 940

{
    "id": 2,
    "userId": 1,
    "orgId": 1,
    "name": "Report 2",
    "recipients": "example-report@grafana.com",
    "replyTo": "",
    "message": "Hi, \nPlease find attached a PDF status report. If you have any questions, feel fr
    "schedule": {
```

```
"startDate": "2022-10-02T00:00:00+02:00",
"endDate": null,
"frequency": "once",
"intervalFrequency": "",
"intervalAmount": 0,
```

Expand code

Status Codes

- **200** – OK
- **400** – Bad request (invalid report ID).
- **401** - Authentication failed, refer to Authentication API.
- **403** – Forbidden (access denied to a report or a dashboard used in the report).
- **404** – Not found (such report does not exist).
- **500** – Unexpected error or server misconfiguration. Refer to server logs for more details.

Create a report

`POST /api/reports`

Required permissions

See note in the introduction for an explanation.

Action	Scope
reports:create	n/a

Example request

http

Copy

```
POST /api/reports HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
    "name": "Report 4",
    "recipients": "texample-report@grafana.com",
    "replyTo": "",
```

```
    "message": "Hello, please, find the report attached",
    "schedule": {
        "startDate": "2022-10-02T10:00:00+02:00",
        "endDate": "2022-11-02T20:00:00+02:00",
        "frequency": "daily",
        "intervalFrequency": "",
        "intervalAmount": 0,
        "workdaysOnly": true,
        "timeZone": "Europe/Warsaw"
```


Expand code

Config JSON Body Schema

Field name	Data type	Description
name	string	Name of the report that is used as an email subject.
recipients	string	Comma-separated list of emails to which to send the report to.
replyTo	string	Comma-separated list of emails used in a reply-to field of the report email.
message	string	Text message used for the body of the report email.
startDate	string	Report distribution starts from this date.
endDate	string	Report distribution ends on this date.
frequency	string	Specifies how often the report should be sent. Can be `once`, `hourly`, `daily`, `weekly`, `monthly`, `last` or `custom`. `last` - schedules the report for the last day of month. `custom` - schedules the report to be sent on a custom interval. It requires `intervalFrequency` and `intervalAmount` to be specified: for example, every 2 weeks, where 2 is an `intervalAmount` and `weeks` is an `intervalFrequency`.
intervalFrequency	string	The type of the `custom` interval: `hours`, `days`, `weeks`, `months`.
intervalAmount	number	`custom` interval amount.
workdaysOnly	string	Send the report only on Monday-Friday. Applicable to `hourly` and `daily` types of schedule.
timeZone	string	Time zone used to schedule report execution.
orientation	string	Can be `portrait` or `landscape`.
layout	string	Can be `grid` or `simple`.
enableDashboardUrl	bool	Adds a dashboard url to the bottom of the report email.
formats	[]string	Specified what kind of attachment to generate for the report - `csv`, `pdf`, `image`.

Field name	Data type	Description
		`pdf` is the default one. `csv` attaches a CSV file for each table panel. `image` embeds an image of a dashboard into the email's body.
dashboards	[]object	Dashboards to generate a report for. See "Report Dashboard Schema" section below.

Report Dashboard Schema

Field name	Data type	Description
dashboard.uid	string	Dashboard UID.
timeRange.from	string	Dashboard time range from.
timeRange.to	string	Dashboard time range to.
reportVariables.	string	Key-value pairs containing the template variables for this report, in JSON format. If empty, the template variables from the report's dashboard will be used.

Example response

http

Copy

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 35

{
    "id": 4,
    "message": "Report created"
}
```

Status Codes

- **200** – OK
- **400** – Bad request (invalid json, missing or invalid fields values, etc.).
- **403** - Forbidden (access denied to a report or a dashboard used in the report).
- **500** - Unexpected error or server misconfiguration. Refer to server logs for more details

Update a report

`PUT /api/reports/:id`

Required permissions

See note in the introduction for an explanation.

Action	Scope
reports:write	reports:* reports:id:* reports:1(single report)

Example request

See JSON body schema for fields description.

http

Copy

```
GET /api/reports HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
    "name": "Updated Report",
    "recipients": "example-report@grafana.com",
    "replyTo": "",
    "message": "Hello, please, find the report attached",
    "schedule": {
        "frequency": "hourly",
        "timeZone": "Africa/Cairo",
        "workdaysOnly": true,
        "startDate": "2022-10-10T10:00:00+02:00",
        "endDate": "2022-11-20T19:00:00+02:00"
    },
    "options": {
        "orientation": "landscape",
```

Expand code

Example response

```http
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 28

{
    "message": "Report updated"
}
```

Status Codes

- **200** – OK
- **400** – Bad request (invalid json, missing or invalid fields values, etc.).
- **401** - Authentication failed, refer to Authentication API.
- **403** – Forbidden (access denied to a report or a dashboard used in the report).
- **404** – Not found (such report does not exist).
- **500** – Unexpected error or server misconfiguration. Refer to server logs for more details.

Delete a report

`DELETE /api/reports/:id`

Required permissions

See note in the introduction for an explanation.

Action	Scope
reports:delete	reports:* reports:id:* reports:1(single report)

Example request

```http
GET /api/reports/6 HTTP/1.1
Accept: application/json
```

```
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

http Copy

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 39

{
    "message": "Report config was removed"
}
```

Status Codes

- **200** – OK
- **400** – Bad request (invalid report ID).
- **401** - Authentication failed, refer to Authentication API.
- **404** - Not found (report with this ID does not exist).
- **500** - Unexpected error or server misconfiguration. Refer to server logs for more details

Send a report

`POST /api/reports/email`

Generate and send a report. This API waits for the report to be generated before returning. We recommend that you set the client's timeout to at least 60 seconds.

Required permissions

See note in the introduction for an explanation.

Action	Scope
reports:send	n/a

Example request

```http
POST /api/reports/email HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "id":"3",
  "useEmailsFromReport": true
}
```

JSON Body Schema

Field name	Data type	Description
id	string	ID of the report to send. It is the same as in the URL when editing a report, not to be confused with the ID of the dashboard. Required.
emails	string	Comma-separated list of emails to which to send the report to. Overrides the emails from the report. Required if `useEmailsFromReport` is not present.
useEmailsFromReport	boolean	Send the report to the emails specified in the report. Required if `emails` is not present.

Example response

```http
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 29

{"message":"Report was sent"}
```

Status Codes

- **200** – Report was sent.
- **400** – Bad request (invalid json, missing content-type, missing or invalid fields, etc.).

- **401** - Authentication failed, refer to Authentication API.
- **403** - Forbidden (access denied to a report or a dashboard used in the report).
- **404** - Report not found.
- **500** - Unexpected error or server misconfiguration. Refer to server logs for more details.

Get reports branding settings

`GET /api/reports/settings`

Returns reports branding settings that are global and used across all the reports.

Required permissions

See note in the introduction for an explanation.

Action	Scope
reports.settings:read	n/a

Example request

http Copy

```
GET /api/reports/settings HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

http Copy

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 181

{
    "id": 1,
    "userId": 1,
    "orgId": 1,
```

```
    "branding": {
        "reportLogoUrl": "",
        "emailLogoUrl": "",
        "emailFooterMode": "sent-by",
        "emailFooterText": "Grafana Labs",
        "emailFooterLink": "https://grafana.com/"
    }
}
```


Status Codes

- **200** – OK
- **401** - Authentication failed, refer to Authentication API.
- **500** - Unexpected error or server misconfiguration. Refer to server logs for more detail

Save reports branding settings

`POST /api/reports/settings`

Creates settings if they don't exist, otherwise updates them. These settings are global and used across all the reports.

Required permissions

See note in the introduction for an explanation.

Action	Scope
reports.settings:write	n/a

Example request

http Copy


```
POST /api/reports/settings HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
    "branding": {
```

```
        "reportLogoUrl": "https://grafana.com/reportLogo.jpg",
        "emailLogoUrl": "https://grafana.com/emailLogo.jpg",
        "emailFooterMode": "sent-by",
        "emailFooterText": "Grafana Labs",
        "emailFooterLink": "https://grafana.com/"
    }
}
```

JSON Body Schema

Field name	Data type	Description
branding.reportLogoUrl	string	URL of an image used as a logo on every page of the report.
branding.emailLogoUrl	string	URL of an image used as a logo in the email.
branding.emailFooterMode	string	Can be `sent-by` or `none`. `sent-by` adds a “Sent by `branding.emailFooterText`” footer link to the email. Requires specifying values in the `branding.emailFooterText` and `branding.emailFooterLink` fields. `none` suppresses adding a “Sent by” footer link to the email.
branding.emailFooterText	string	Text of a URL added to the email “Sent by” footer.
branding.emailFooterLink	string	URL address value added to the email “Sent by” footer.

Example response

http Copy

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 35

{
    "message": "Report settings saved"
}
```

Status Codes

- **200** – OK
- **400** – Bad request (invalid json, missing or invalid fields values, etc.).

- **401** - Authentication failed, refer to Authentication API.
- **500** - Unexpected error or server misconfiguration. Refer to server logs for more detail

Send a test email

`POST /api/reports/test-email`

Sends a test email with a report without persisting it in the database.

Required permissions

See note in the introduction for an explanation.

Action	Scope
reports:send	n/a

Example request

See JSON body schema for fields description.

http Copy

```
POST /api/reports/test-email HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{{
    "name": "Report 4",
    "recipients": "example-report@grafana.com",
    "replyTo": "",
    "message": "Hello, please, find the report attached",
    "schedule": {
        "startDate": "2022-10-02T10:00:00+02:00",
        "endDate": "2022-11-02T20:00:00+02:00",
        "frequency": "daily",
        "intervalFrequency": "",
        "intervalAmount": 0,
        "workdaysOnly": true,
        "timeZone": "Europe/Warsaw"
    },
```

Expand code

Example response

http

Copy

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 29

{
    "message": "Test email sent"
}
```

Status Codes

- **200** – OK
- **400** – Bad request (invalid json, missing or invalid fields values, etc.).
- **401** - Authentication failed, refer to Authentication API.
- **403** - Forbidden (access denied to a report or a dashboard used in the report).
- **500** - Unexpected error or server misconfiguration. Refer to server logs for more details

Service account API

If you are running Grafana Enterprise, for some endpoints you'll need to have specific permissions. Refer to Role-based access control permissions for more information. For Grafana Cloud instances, please use a Bearer token to authenticate. The examples within this section reference Basic authentication which is for On-Prem Grafana instances.

Search service accounts with Paging

`GET /api/serviceaccounts/search?perpage=10&page=1&query=myserviceaccount`

Required permissions

See note in the introduction for an explanation.

Action	Scope
serviceaccounts:read	n/a

Example Request:

http Copy

```
GET /api/serviceaccounts/search?perpage=10&page=1&query=mygraf HTTP/1.1
Accept: application/json
```

```
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=
```

Default value for the `perpage` parameter is `1000` and for the `page` parameter is `1`. The `totalCount` field in the response can be used for pagination of the user list E.g. if `totalCount` is equal to 100 users and the `perpage` parameter is set to 10 then there are 10 pages of users. The `query` parameter is optional and it will return results where the query value is contained in one of the `name`. Query values with spaces need to be URL encoded e.g. `query=Jane%20Doe`.

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json
{
    "totalCount": 2,
    "serviceAccounts": [
        {
            "id": 1,
            "name": "grafana",
            "login": "sa-grafana",
            "orgId": 1,
            "isDisabled": false,
            "role": "Viewer",
            "tokens": 0,
            "avatarUrl": "/avatar/85ec38023d90823d3e5b43ef35646af9",
            "accessControl": {
                "serviceaccounts:delete": true,
                "serviceaccounts:read": true,
                "serviceaccounts:write": true
            }
```

Expand code

Create service account

`POST /api/serviceaccounts`

Required permissions

See note in the introduction for an explanation.

Action	Scope
serviceaccounts:create	n/a

Example Request:

```http
POST /api/serviceaccounts HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=

{
  "name": "grafana",
  "role": "Viewer",
  "isDisabled": false
}
```

Example Response:

```http
HTTP/1.1 201
Content-Type: application/json

{
    "id": 1,
    "name": "test",
    "login": "sa-test",
    "orgId": 1,
    "isDisabled": false,
    "createdAt": "2022-03-21T14:35:33Z",
    "updatedAt": "2022-03-21T14:35:33Z",
    "avatarUrl": "/avatar/8ea890a677d6a223c591a1beea6ea9d2",
    "role": "Viewer",
    "teams": []
}
```

Get a service account by ID

`GET /api/serviceaccounts/:id`

Required permissions

See note in the introduction for an explanation.

Action	Scope
serviceaccounts:read	serviceaccounts:id:*

Example Request:

```http
GET /api/serviceaccounts/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{
    "id": 1,
    "name": "test",
    "login": "sa-test",
    "orgId": 1,
    "isDisabled": false,
    "createdAt": "2022-03-21T14:35:33Z",
    "updatedAt": "2022-03-21T14:35:33Z",
    "avatarUrl": "/avatar/8ea890a677d6a223c591a1beea6ea9d2",
    "role": "Viewer",
    "teams": []
}
```

Update service account

`PATCH /api/serviceaccounts/:id`

Required permissions

See note in the introduction for an explanation.

Action	Scope
serviceaccounts:write	serviceaccounts:id:*

Example Request:

http Copy

```
PATCH /api/serviceaccounts/2 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=

{
  "name": "test",
    "role": "Editor"
}
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{
    "id": 2,
    "name": "test",
    "login": "sa-grafana",
    "orgId": 1,
    "isDisabled": false,
    "createdAt": "2022-03-21T14:35:44Z",
    "updatedAt": "2022-03-21T14:35:44Z",
    "avatarUrl": "/avatar/8ea890a677d6a223c591a1beea6ea9d2",
    "role": "Editor",
    "teams": []
}
```

Delete service account

`DELETE /api/serviceaccounts/:id`

Required permissions

See note in the introduction for an explanation.

Action	Scope
serviceaccounts:delete	serviceaccounts:id:*

Example Request:

```http
DELETE /api/serviceaccounts/2 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{
    "message": "Service account deleted"
}
```

Migrate API keys to service accounts

`POST /api/serviceaccounts/migrate`

Required permissions

See note in the introduction for an explanation.

Action	Scope
serviceaccounts:write	serviceaccounts:*

Example Request:

```http
```

```
POST /api/serviceaccounts/migrate HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{
    "message": "API keys migrated to service accounts"
}
```

Migrate API key to service account

`POST /api/serviceaccounts/migrate/:keyId`

Required permissions

See note in the introduction for an explanation.

Action	Scope
serviceaccounts:write	serviceaccounts:*

Example Request:

http Copy

```
POST /api/serviceaccounts/migrate/4 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json
```

```
{
    "message": "Service accounts migrated"
}
```

Get API key to service account migration status

`GET /api/serviceaccounts/migrationstatus`

Required permissions

See note in the introduction for an explanation.

Action	Scope
serviceaccounts:read	serviceaccounts:*

Example Request:

http Copy

```
POST /api/serviceaccounts/migrationstatus HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{
    "migrated": true
}
```

Hide the API keys tab

`GET /api/serviceaccounts/hideApiKeys`

Required permissions

See note in the introduction for an explanation.

Action	Scope
serviceaccounts:write	serviceaccounts:*

Example Request:

```http
POST /api/serviceaccounts/hideApiKeys HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{
    "message": "API keys hidden"
}
```

Get service account tokens

`GET /api/serviceaccounts/:id/tokens`

Required permissions

See note in the introduction for an explanation.

Action	Scope
serviceaccounts:read	serviceaccounts:id:*

Example Request:

```http
GET /api/serviceaccounts/2/tokens HTTP/1.1
Accept: application/json
```

```
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=
```

Example Response:

http

Copy

```
HTTP/1.1 200
Content-Type: application/json

[
    {
        "id": 1,
        "name": "grafana",
        "role": "Viewer",
        "created": "2022-03-23T10:31:02Z",
        "expiration": null,
        "secondsUntilExpiration": 0,
        "hasExpired": false
    }
]
```

Create service account tokens

`POST /api/serviceaccounts/:id/tokens`

Required permissions

See note in the introduction for an explanation.

Action	Scope
serviceaccounts:write	serviceaccounts:id:*

Example Request:

http

Copy

```
POST /api/serviceaccounts/2/tokens HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=
```

```
{
    "name": "grafana"
}
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{
    "id": 7,
    "name": "grafana",
    "key": "eyJrIjoiVjFxTHZ6dGdPSjg5Um92MjN1RlhjMkNqYkZUbm9jYkwiLCJuIjoiZ3JhZmFuYSIsImlkIjoxfQ=="
}
```

Delete service account tokens

`DELETE /api/serviceaccounts/:id/tokens/:tokenId`

Required permissions

See note in the introduction for an explanation.

Action	Scope
serviceaccounts:write	serviceaccounts:id:*

Example Request:

```http
DELETE /api/serviceaccounts/2/tokens/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json
```

```
{
    "message": "API key deleted"
}
```

Revert service account token to API key

`DELETE /api/serviceaccounts/:serviceAccountId/revert/:keyId`

This operation will delete the service account and create a legacy API Key for the given `keyId`.

Required permissions

See note in the introduction for an explanation.

Action	Scope
serviceaccounts:delete	serviceaccounts:id:*

Example Request:

http Copy

```
DELETE /api/serviceaccounts/1/revert/glsa_VVQjot0nijQ59lun6pMZRtsdBXxnFQ9M_77c34a79 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{
    "message": "Reverted service account to API key"
}
```

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Short URL API

Use this API to create shortened URLs. A short URL represents a longer URL containing complex query parameters in a smaller and simpler format.

Create short URL

`POST /api/short-urls`

Creates a short URL.

Example request:

http Copy

```
POST /api/short-urls HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "path": "d/TxKARsmGz/new-dashboard?orgId=1&from=1599389322894&to=1599410922894"
}
```

JSON body schema:

- **path** – The path to shorten, relative to the Grafana root_url.

Example response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{
  "uid": AT76wBvGk,
  "url": http://localhost:3000/goto/AT76wBvGk?orgId=1
}
```

Status codes:

- **200** – Created
- **400** – Errors (invalid JSON, missing or invalid fields)

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Enterprise Open source

SSO Settings API

If you are running Grafana Enterprise, for some endpoints you'll need to have specific permissions. Refer to Role-based access control permissions for more information.

The API can be used to create, update, delete, get, and list SSO Settings.

List SSO Settings

`GET /api/v1/sso-settings`

Lists the SSO Settings for all providers.

Required permissions

See note in the introduction for an explanation.

Action	Scope
`settings:read`	`settings:auth.{provider}:*`

Example Request:

http

Copy

```
GET /api/v1/sso-settings HTTP/1.1
Accept: application/json
```

```
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json
[
  {
    "id":        "1",
    "provider":  "github",
    "settings": {
      "apiUrl": "https://api.github.com/user",
      "clientId": "my_github_client",
      "clientSecret": "*********",
      "enabled": true,
      "scopes": "user:email,read:org"
      // rest of the settings
    },
    "source":    "system",
  },
  {
    "id":        "2",
    "provider":  "azuread",
```

Expand code

Status Codes:

- **200** – SSO Settings found
- **400** – Bad Request
- **401** – Unauthorized
- **403** – Access Denied

Get SSO Settings

`GET /api/v1/sso-settings/:provider`

Gets the SSO Settings for a provider.

Required permissions

See note in the introduction for an explanation.

Action	Scope
`settings:read`	`settings:auth.{provider}:*`

Example Request:

```http
GET /api/v1/sso-settings/github HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json
ETag: db87f729761898ee
{
  "id":        "1",
  "provider":  "github",
  "settings": {
    "apiUrl": "https://api.github.com/user",
    "clientId": "my_github_client",
    "clientSecret": "*********",
    "enabled": true,
    "scopes": "user:email,read:org"
    // rest of the settings
  },
  "source":    "system",
}
```

Status Codes:

- **200** – SSO Settings found
- **400** – Bad Request
- **401** – Unauthorized
- **403** – Access Denied
- **404** – SSO Settings not found

Update SSO Settings

`PUT /api/v1/sso-settings/:provider`

Updates the SSO Settings for a provider.

Required permissions

See note in the introduction for an explanation.

Action	Scope
`settings:write`	`settings:auth.{provider}:*`

Example Request:

http

```
PUT /api/v1/sso-settings/github HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "settings": {
    "apiUrl": "https://api.github.com/user",
    "clientId": "my_github_client",
    "clientSecret": "my_github_secret",
    "enabled": true,
    "scopes": "user:email,read:org"
  }
}
```

Example Response:

http

```
HTTP/1.1 204
Content-Type: application/json
```

Status Codes:

- **204** – SSO Settings updated
- **400** – Bad Request

- **401** – Unauthorized
- **403** – Access Denied

Delete SSO Settings

`DELETE /api/v1/sso-settings/:provider`

Deletes an existing SSO Settings entry for a provider.

Required permissions

See note in the introduction for an explanation.

Action	Scope
`settings:write`	`settings:auth.{provider}:*`

Example Request:

```http
DELETE /api/v1/sso-settings/azuread HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```http
HTTP/1.1 204
Content-Type: application/json
```

Status Codes:

- **204** – SSO Settings deleted
- **400** – Bad Request
- **401** – Unauthorized
- **403** – Access Denied
- **404** – SSO Settings not found

Team API

This API can be used to manage Teams and Team Memberships.

Access to these API endpoints is restricted as follows:

- All authenticated users are able to view details of teams they are a member of.
- Organization Admins are able to manage all teams and team members.
- If you enable `editors_can_admin` configuration flag, then Organization Editors can create teams and manage teams where they are Admin.
  - If you enable `editors_can_admin` configuration flag, Editors can find out whether a team that they are not members of exists by trying to create a team with the same name.

If you are running Grafana Enterprise, for some endpoints you'll need to have specific permissions. Refer to Role-based access control permissions for more information.

Team Search With Paging

`GET /api/teams/search?perpage=50&page=1&query=myteam&sort=memberCount-desc`

or

`GET /api/teams/search?name=myteam`

Required permissions

See note in the introduction for an explanation.

Action	Scope
teams:read	teams:*

Example Request:

```http
GET /api/teams/search?perpage=10&page=1&query=mytestteam HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{
  "totalCount": 1,
  "teams": [
    {
      "id": 1,
      "orgId": 1,
      "name": "MyTestTeam",
      "email": "",
      "avatarUrl": "\/avatar\/3f49c15916554246daa714b9bd0ee398",
      "memberCount": 1
    }
  ],
  "page": 1,
  "perPage": 1000
}
```

Using the query parameter

Default value for the `perpage` parameter is `1000` and for the `page` parameter is `1`.

The `totalCount` field in the response can be used for pagination of the teams list E.g. if `totalCount` is equal to 100 teams and the `perpage` parameter is set to 10 then there are 10 pages of teams.

The `query` parameter is optional and it will return results where the query value is contained in the `name` field. Query values with spaces need to be URL encoded e.g. `query=my%20team`.

The `sort` param is an optional comma separated list of options to order the search result. Accepted values for the sort filter are: `name-asc`, `name-desc`, `email-asc`, `email-desc`, `memberCount-asc`, `memberCount-desc`. By default, if `sort` is not specified, the teams list will be ordered by `name` in ascending order.

Using the name parameter

The `name` parameter returns a single team if the parameter matches the `name` field.

Status Codes:

- **200** - Ok
- **400** - Bad Request
- **401** - Unauthorized
- **403** - Permission denied
- **404** - Team not found (if searching by name)

Get Team By Id

`GET /api/teams/:id`

Required permissions

See note in the introduction for an explanation.

Action	Scope
teams:read	teams:*

Example Request:

http Copy

```
GET /api/teams/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{
  "id": 1,
  "orgId": 1,
  "name": "MyTestTeam",
  "email": "",
  "created": "2017-12-15T10:40:45+01:00",
  "updated": "2017-12-15T10:40:45+01:00"
}
```

Status Codes:

- **200** - Ok
- **401** - Unauthorized
- **403** - Permission denied
- **404** - Team not found

Add Team

The Team `name` needs to be unique. `name` is required and `email`,`orgId` is optional.

`POST /api/teams`

Required permissions

See note in the introduction for an explanation.

Action	Scope
teams:create	N/A

Example Request:

http Copy

```
POST /api/teams HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=

{
```

```
  "name": "MyTestTeam",
  "email": "email@test.com",
  "orgId": 2
}
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{"message":"Team created","teamId":2}
```

Status Codes:

- **200** - Ok
- **401** - Unauthorized
- **403** - Permission denied
- **409** - Team name is taken

Update Team

There are two fields that can be updated for a team: `name` and `email`.

`PUT /api/teams/:id`

Required permissions

See note in the introduction for an explanation.

Action	Scope
teams:write	teams:*

Example Request:

```http
PUT /api/teams/2 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=
```

```
{
  "name": "MyTestTeam",
  "email": "email@test.com"
}
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{"message":"Team updated"}
```

Status Codes:

- **200** - Ok
- **401** - Unauthorized
- **403** - Permission denied
- **404** - Team not found
- **409** - Team name is taken

Delete Team By Id

`DELETE /api/teams/:id`

Required permissions

See note in the introduction for an explanation.

Action	Scope
teams:delete	teams:*

Example Request:

```http
DELETE /api/teams/2 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{"message":"Team deleted"}
```

Status Codes:

- **200** - Ok
- **401** - Unauthorized
- **403** - Permission denied
- **404** - Failed to delete Team. ID not found

Get Team Members

`GET /api/teams/:teamId/members`

Required permissions

See note in the introduction for an explanation.

Action	Scope
teams.permissions:read	teams:*

Example Request:

```http
GET /api/teams/1/members HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json
```

```
[
  {
    "orgId": 1,
    "teamId": 1,
    "userId": 3,
    "email": "user1@email.com",
    "login": "user1",
    "avatarUrl": "\/avatar\/1b3c32f6386b0185c40d359cdc733a79"
  },
  {
    "orgId": 1,
    "teamId": 1,
    "userId": 2,
    "email": "user2@email.com",
    "login": "user2",
```


Expand code

Status Codes:

- **200** - Ok
- **401** - Unauthorized
- **403** - Permission denied

Add Team Member

`POST /api/teams/:teamId/members`

Required permissions

See note in the introduction for an explanation.

Action	Scope
teams.permissions:write	teams:*

Example Request:

http

Copy

```
POST /api/teams/1/members HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=
```

```
{
  "userId": 2
}
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{"message":"Member added to Team"}
```

Status Codes:

- **200** - Ok
- **400** - User is already added to this team
- **401** - Unauthorized
- **403** - Permission denied
- **404** - Team not found

Remove Member From Team

`DELETE /api/teams/:teamId/members/:userId`

Required permissions

See note in the introduction for an explanation.

Action	Scope
teams.permissions:write	teams:*

Example Request:

```http
DELETE /api/teams/2/members/3 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{"message":"Team Member removed"}
```

Status Codes:

- **200** - Ok
- **401** - Unauthorized
- **403** - Permission denied
- **404** - Team not found/Team member not found

Get Team Preferences

`GET /api/teams/:teamId/preferences`

Required permissions

See note in the introduction for an explanation.

Action	Scope
teams:read	teams:*

Example Request:

```http
GET /api/teams/2/preferences HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{
```

```
  "theme": "",
  "homeDashboardId": 0,
  "timezone": ""
}
```

Update Team Preferences

`PUT /api/teams/:teamId/preferences`

Required permissions

See note in the introduction for an explanation.

Action	Scope
teams:write	teams:*

Example Request:

http Copy

```
PUT /api/teams/2/preferences HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "theme": "dark",
  "homeDashboardId": 39,
  "timezone": "utc"
}
```

JSON Body Schema:

- **theme** - One of: `light`, `dark`, or an empty string for the default theme
- **homeDashboardId** - The numerical `:id` of a dashboard, default: `0`
- **timezone** - One of: `utc`, `browser`, or an empty string for the default

Omitting a key will cause the current value to be replaced with the system default value.

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: text/plain; charset=utf-8


{
  "message":"Preferences updated"
}
```

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Enterprise Open source

Team Sync API

Team Sync is only available in Grafana Enterprise. Read more about Grafana Enterprise.

If you are running Grafana Enterprise, for some endpoints you'll need to have specific permissions. Refer to Role-based access control permissions for more information.

Get External Groups

`GET /api/teams/:teamId/groups`

Required permissions

See note in the introduction for an explanation.

Action	Scope
teams.permissions:read	teams:*

Example Request:

http Copy

```
GET /api/teams/1/groups HTTP/1.1
Accept: application/json
```

```
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

[
  {
    "orgId": 1,
    "teamId": 1,
    "groupId": "cn=editors,ou=groups,dc=grafana,dc=org"
  }
]
```

Status Codes:

- **200** - Ok
- **401** - Unauthorized
- **403** - Permission denied

Add External Group

`POST /api/teams/:teamId/groups`

Required permissions

See note in the introduction for an explanation.

Action	Scope
teams.permissions:write	teams:*

Example Request:

```http
POST /api/teams/1/groups HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=
```

```
{
  "groupId": "cn=editors,ou=groups,dc=grafana,dc=org"
}
```

Example Response:

```
http
HTTP/1.1 200
Content-Type: application/json

{"message":"Group added to Team"}
```

Status Codes:

- **200** - Ok
- **400** - Group is already added to this team
- **401** - Unauthorized
- **403** - Permission denied
- **404** - Team not found

Remove External Group

`DELETE /api/teams/:teamId/groups?groupId=external-group-id`

Required permissions

See note in the introduction for an explanation.

Action	Scope
teams.permissions:write	teams:*

Example Request:

```
http
DELETE /api/teams/1/groups?groupId=cn%3Deditors%2Cou%3Dgroups%2Cdc%3Dgrafana%2Cdc%3Dorg HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{"message":"Team Group removed"}
```

Status Codes:

- **200** - Ok
- **401** - Unauthorized
- **403** - Permission denied
- **404** - Team not found/Group not found

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Enterprise Open source

User API

The Users HTTP API does not currently work with an API Token. API Tokens are linked to an organization and an organization role. They cannot be given the permission of server users access, only users can be given that permission. To use these API calls you can use Basic Auth and the Grafana user must have the Grafana Admin role.

API Tokens can be used with Organization HTTP API to get users of specific organization.

If you are running Grafana Enterprise, for some endpoints you'll need to have specific permissions. Refer to Role-based access control permissions for more information.

Search Users

`GET /api/users?perpage=10&page=1&sort=login-asc,email-asc`

Required permissions

See note in the introduction for an explanation.

Action	Scope
users:read	global.users:*

Example Request:

```http
```

```
GET /api/users HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=
```

Default value for the `perpage` parameter is `1000` and for the `page` parameter is `1`. Requires basic authentication and that the authenticated user is a Grafana Admin.

The `sort` param is an optional comma separated list of options to order the search result. Accepted values for the sort filter are: `login-asc`, `login-desc`, `email-asc`, `email-desc`, `name-asc`, `name-desc`, `lastSeenAtAge-asc`, `lastSeenAtAge-desc`. By default, if `sort` is not specified, the user list will be ordered by `login`, `email` in ascending order.

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

[
  {
    "id": 1,
    "name": "Admin",
    "login": "admin",
    "email": "admin@mygraf.com",
    "isAdmin": true,
    "isDisabled": false,
    "lastSeenAt": "2020-04-10T20:29:27+03:00",
    "lastSeenAtAge": "2m",
    "authLabels": ["OAuth"]
  },
  {
    "id": 2,
    "name": "User",
    "login": "user",
```

Expand code

Search Users with Paging

`GET /api/users/search?perpage=10&page=1&query=mygraf&sort=login-asc,email-asc`

Required permissions

See note in the introduction for an explanation.

Action	Scope
users:read	global.users:*

Example Request:

http Copy

```
GET /api/users/search?perpage=10&page=1&query=mygraf HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=
```

Default value for the `perpage` parameter is `1000` and for the `page` parameter is `1`. The `totalCount` field in the response can be used for pagination of the user list E.g. if `totalCount` is equal to 100 users and the `perpage` parameter is set to 10 then there are 10 pages of users. The `query` parameter is optional and it will return results where the query value is contained in one of the `name`, `login` or `email` fields. Query values with spaces need to be URL encoded e.g. `query=Jane%20Doe`.

The `sort` param is an optional comma separated list of options to order the search result. Accepted values for the sort filter are: `login-asc`, `login-desc`, `email-asc`, `email-desc`, `name-asc`, `name-desc`, `lastSeenAtAge-asc`, `lastSeenAtAge-desc`. By default, if `sort` is not specified, the user list will be ordered by `login`, `email` in ascending order.

Requires basic authentication and that the authenticated user is a Grafana Admin.

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json
{
  "totalCount": 2,
  "users": [
    {
      "id": 1,
      "name": "Admin",
      "login": "admin",
      "email": "admin@mygraf.com",
      "isAdmin": true,
      "isDisabled": false,
      "lastSeenAt": "2020-04-10T20:29:27+03:00",
      "lastSeenAtAge': "2m",
      "authLabels": ["OAuth"]
    },
```

```
{
  "id": 2,
```

Expand code

Get single user by Id

`GET /api/users/:id`

Required permissions

See note in the introduction for an explanation.

Action	Scope
users:read	global.users:*

Example Request:

```http
GET /api/users/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=
```

Copy

Requires basic authentication and that the authenticated user is a Grafana Admin.

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{
  "id": "1",
  "email": "user@mygraf.com",
  "name": "admin",
  "login": "admin",
  "theme": "light",
  "orgId": 1,
  "isGrafanaAdmin": true,
  "isDisabled": true,
  "isExternal": false,
  "authLabels": [],
```

Copy

```
  "updatedAt": "2019-09-09T11:31:26+01:00",
  "createdAt": "2019-09-09T11:31:26+01:00",
  "avatarUrl": ""
}
```

Get single user by Username(login) or Email

`GET /api/users/lookup?loginOrEmail=user@mygraf.com`

Required permissions

See note in the introduction for an explanation.

Action	Scope
users:read	global.users:*

Example Request using the email as option:

http Copy

```
GET /api/users/lookup?loginOrEmail=user@mygraf.com HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=
```

Example Request using the username as option:

http Copy

```
GET /api/users/lookup?loginOrEmail=admin HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=
```

Requires basic authentication and that the authenticated user is a Grafana Admin.

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{
```

```
  "id": 1,
  "email": "user@mygraf.com",
  "name": "admin",
  "login": "admin",
  "theme": "light",
  "orgId": 1,
  "isGrafanaAdmin": true,
  "isDisabled": false,
  "isExternal": false,
  "authLabels": null,
  "updatedAt": "2019-09-25T14:44:37+01:00",
  "createdAt": "2019-09-25T14:44:37+01:00",
  "avatarUrl":""
}
```


User Update

`PUT /api/users/:id`

Required permissions

See note in the introduction for an explanation.

Action	Scope
users:write	global.users:*

Example Request:

http Copy

```
PUT /api/users/2 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=

{
  "email":"user@mygraf.com",
  "name":"User2",
  "login":"user",
  "theme":"light"
}
```

Requires basic authentication and that the authenticated user is a Grafana Admin.

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{"message":"User updated"}
```

Get Organizations for user

`GET /api/users/:id/orgs`

Required permissions

See note in the introduction for an explanation.

Action	Scope
users:read	global.users:*

Example Request:

```http
GET /api/users/1/orgs HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=
```

Requires basic authentication and that the authenticated user is a Grafana Admin.

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

[
  {
    "orgId":1,
```

```
    "name":"Main Org.",
    "role":"Admin"
  }
]
```


Get Teams for user

`GET /api/users/:id/teams`

Required permissions

See note in the introduction for an explanation.

Action	Scope
users:read	global.users:*
teams:read	teams:*

Example Request:

http Copy

```
GET /api/users/1/teams HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=
```

Requires basic authentication and that the authenticated user is a Grafana Admin.

Example Response:

http Copy


```
HTTP/1.1 200
Content-Type: application/json

[
  {
    "id":1,
    "orgId":1,
    "name":"team1",
    "email":"",
    "avatarUrl":"/avatar/3fcfe295eae3bcb67a49349377428a66",
```

```
    "memberCount":1
  }
]
```

User

Actual User

`GET /api/user`

Example Request:

http Copy

```
GET /api/user HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=
```

Requires basic authentication.

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json

{
  "id":1,
  "email":"admin@mygraf.com",
  "name":"Admin",
  "login":"admin",
  "theme":"light",
  "orgId":1,
  "isGrafanaAdmin":true,
  "isDisabled":false
  "isExternal": false,
  "authLabels": [],
  "updatedAt": "2019-09-09T11:31:26+01:00",
  "createdAt": "2019-09-09T11:31:26+01:00",
  "avatarUrl": ""
}
```

Change Password

`PUT /api/user/password`

Changes the password for the user. Requires basic authentication.

Example Request:

```http
PUT /api/user/password HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=

{
  "oldPassword": "old_password",
  "newPassword": "new_password"
}
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{"message":"User password changed"}
```

Change Password with a Script

If you need to change a password with a script, here is an example of changing the Admin password using curl with basic auth:

```bash
curl -X PUT -H "Content-Type: application/json" -d '{
  "oldPassword": "oldpass",
  "newPassword": "newpass",
  "confirmNew": "newpass"
}' http://admin:oldpass@<your_grafana_host>:3000/api/user/password
```

Switch user context for a specified user

`POST /api/users/:userId/using/:organizationId`

Switch user context to the given organization. Requires basic authentication and that the authenticated user is a Grafana Admin.

Example Request:

```http
POST /api/users/7/using/2 HTTP/1.1
Authorization: Basic YWRtaW46YWRtaW4=
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{"message":"Active organization changed"}
```

Switch user context for signed in user

`POST /api/user/using/:organizationId`

Switch user context to the given organization.

Example Request:

```http
POST /api/user/using/2 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{"message":"Active organization changed"}
```

Organizations of the actual User

`GET /api/user/orgs`

Return a list of all organizations of the current user. Requires basic authentication.

Example Request:

```http
GET /api/user/orgs HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Basic YWRtaW46YWRtaW4=
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

[
  {
    "orgId":1,
    "name":"Main Org.",
    "role":"Admin"
  }
]
```

Teams that the actual User is member of

`GET /api/user/teams`

Return a list of all teams that the current user is member of.

Example Request:

```http
GET /api/user/teams HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

[
  {
    "id": 1,
    "orgId": 1,
    "name": "MyTestTeam",
    "email": "",
    "avatarUrl": "\/avatar\/3f49c15916554246daa714b9bd0ee398",
    "memberCount": 1
  }
]
```

Star a dashboard

`POST /api/user/stars/dashboard/:dashboardId`

Stars the given Dashboard for the actual user.

Example Request:

```http
POST /api/user/stars/dashboard/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{"message":"Dashboard starred!"}
```

Unstar a dashboard

`DELETE /api/user/stars/dashboard/:dashboardId`

Deletes the starring of the given Dashboard for the actual user.

Example Request:

```http
DELETE /api/user/stars/dashboard/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json

{"message":"Dashboard unstarred"}
```

Auth tokens of the actual User

`GET /api/user/auth-tokens`

Return a list of all auth tokens (devices) that the actual user currently have logged in from.

Example Request:

```http
GET /api/user/auth-tokens HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```http
HTTP/1.1 200
Content-Type: application/json
```

```
[
  {
    "id": 361,
    "isActive": true,
    "clientIp": "127.0.0.1",
    "browser": "Chrome",
    "browserVersion": "72.0",
    "os": "Linux",
    "osVersion": "",
    "device": "Other",
    "createdAt": "2019-03-05T21:22:54+01:00",
    "seenAt": "2019-03-06T19:41:06+01:00"
  },
  {
    "id": 364,
```


Expand code

Revoke an auth token of the actual User

`POST /api/user/revoke-auth-token`

Revokes the given auth token (device) for the actual user. User of issued auth token (device) will no longer be logged in and will be required to authenticate again upon next activity.

Example Request:

http Copy

```
POST /api/user/revoke-auth-token HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "authTokenId": 364
}
```

Example Response:

http Copy

```
HTTP/1.1 200
Content-Type: application/json
```

```
{
  "message": "User auth token revoked"
}
```

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Enterprise Open source

Contribute to Grafana

This page lists resources for developers who want to contribute to the Grafana software ecosystem or build plugins for Grafana.

General resources

These resources are useful for all developers.

- Contributing to Grafana: Start here to learn how you can contribute your skills to make Grafana even better.
- Developer guide: A guide to help you get started developing Grafana software, includes instructions for how to configure Grafana for development.
- Contributing to documentation: A guide to help you contribute to Grafana documentation, includes links to beginner-friendly issues.
- Architecture guides: These guides explain Grafana's background architecture.
- Create a pull request: A guide for new contributors about how to create your first Grafana pull request.
- REST APIs allow you to interact programmatically with the Grafana backend.

Best practices and style

Our style guides outline Grafana style for frontend, backend, documentation, and more, including best practices. Please read through them before you start editing or coding!

- Backend style guide explains how we want to write Go code in the future.
- Documentation style guide applies to all documentation created for Grafana products.
- End to end test framework provides guidance for Grafana e2e tests.

- Frontend style guide provides rules and guidance on developing in React for Grafana.
- Redux framework explains how Grafana handles Redux boilerplate code.
- Styling Grafana expands on styling React components with Emotion.
- Theming Grafana explains how to use themes and ThemeContext in Grafana code.

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Grafana Labs Software Grant and Contributor License Agreement (“Agreement”)

This agreement is based on the Apache Software Foundation Contributor License Agreement. (v r190612)

Thank you for your interest in software projects stewarded by Raintank, Inc. dba Grafana Labs (“Grafana Labs”). In order to clarify the intellectual property license granted with Contributions from any person or entity, Grafana Labs must have a Contributor License Agreement (CLA) on file that has been agreed to by each Contributor, indicating agreement to the license terms below. This license is for your protection as a Contributor as well as the protection of Grafana Labs and its users; it does not change your rights to use your own Contributions for any other purpose. This Agreement allows an individual to contribute to Grafana Labs on that individual’s own behalf, or an entity (the “Corporation”) to submit Contributions to Grafana Labs, to authorize Contributions submitted by its designated employees to Grafana Labs, and to grant copyright and patent licenses thereto.

You accept and agree to the following terms and conditions for Your present and future Contributions submitted to Grafana Labs. Except for the license granted herein to Grafana Labs and recipients of software distributed by Grafana Labs, You reserve all right, title, and interest in and to Your Contributions.

1. Definitions. “You” (or “Your”) shall mean the copyright owner or legal entity authorized by the copyright owner that is making this Agreement with Grafana Labs. For legal entities, the entity making a Contribution and all other entities that control, are controlled by, or are under common control with that entity are considered to be a single Contributor. For the purposes of this definition, “control” means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity. “Contribution” shall mean any work, as well as any modifications or additions to an existing work, that is intentionally submitted by You to Grafana Labs for inclusion in, or documentation of, any of the products owned or managed by Grafana Labs (the “Work”). For the purposes of this definition, “submitted” means any form of electronic, verbal, or written communication

sent to Grafana Labs or its representatives, including but not limited to communication on electronic mailing lists, source code control systems (such as GitHub), and issue tracking systems that are managed by, or on behalf of, Grafana Labs for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by You as “Not a Contribution.”

2. Grant of Copyright License. Subject to the terms and conditions of this Agreement, You hereby grant to Grafana Labs and to recipients of software distributed by Grafana Labs a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare derivative works of, publicly display, publicly perform, sublicense, and distribute Your Contributions and such derivative works.

3. Grant of Patent License. Subject to the terms and conditions of this Agreement, You hereby grant to Grafana Labs and to recipients of software distributed by Grafana Labs a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by You that are necessarily infringed by Your Contribution(s) alone or by combination of Your Contribution(s) with the Work to which such Contribution(s) were submitted. If any entity institutes patent litigation against You or any other entity (including a cross-claim or counterclaim in a lawsuit) alleging that your Contribution, or the Work to which you have contributed, constitutes direct or contributory patent infringement, then any patent licenses granted to that entity under this Agreement for that Contribution or Work shall terminate as of the date such litigation is filed.

4. You represent that You are legally entitled to grant the above license. If You are an individual, and if Your employer(s) has rights to intellectual property that you create that includes Your Contributions, you represent that You have received permission to make Contributions on behalf of that employer, or that Your employer has waived such rights for your Contributions to Grafana Labs. If You are a Corporation, any individual who makes a contribution from an account associated with You will be considered authorized to Contribute on Your behalf.

5. You represent that each of Your Contributions is Your original creation (see section 7 for submissions on behalf of others).

6. You are not expected to provide support for Your Contributions,except to the extent You desire to provide support. You may provide support for free, for a fee, or not at all. Unless required by applicable law or agreed to in writing, You provide Your Contributions on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE.

7. Should You wish to submit work that is not Your original creation, You may submit it to Grafana Labs separately from any Contribution, identifying the complete details of its source and of any license or other restriction (including, but not limited to, related patents, trademarks, and license agreements) of which you are personally aware, and conspicuously marking the work as “Submitted on behalf of a third-party: [named here]”.

Angular support deprecation

Angular plugin support is deprecated and will be removed in a future release. There are legacy core Grafana visualizations and external plugins that rely on Grafana's Angular plugin support to work. The same is likely true for private plugins that have been developed by Grafana users for use on their own instances over the years. From Grafana v9 and onwards, there is a server configuration option that's global to the entire instance and controls whether Angular plugin support is available or not. In Grafana 11, we will change the default value for the configuration to remove support.

Warning messages are displayed if a dashboard depends on an a panel visualization or data source which requires AngularJS as shown in the following video:

To avoid disruption:

- Ensure that you are running the latest version of plugins by following this guide on updating. Many panels and data sources have migrated from AngularJS.
- If you are using legacy Core Grafana visualizations such as Graph or Table-old, migrate to their replacements using the provided automatic migrations.
- Review the list of current Angular plugins to discover which Core and external plugins are impacted, and whether an update or alternative is required.

Why are we deprecating Angular support?

AngularJS is an old frontend framework whose active development stopped many years ago. Therefore, it poses a security risk. As Grafana itself had already started migrating to React in v5, this presented the most logical framework for our plugin platform. AngularJS also requires unsafe-eval in the CSP (Content Security Policy) settings, which also reduces the security of running JavaScript in the browser.

When will Angular plugins stop working?

In Grafana 11, which will be released in preview in April 2024 and generally available in May, we will change the default behavior of the angular_support_enabled configuration parameter to turn off support for AngularJS based plugins. In case you still rely on AngularJS-based plugins developed internally or by the community, you will need to enable this option to continue using them.

New Grafana Cloud users will be unable to request for support to be added to their instance.

When will we remove Angular support completely?

Our current plan is to completely remove any remaining support for Angular plugins in version 12. Including the removal of the angular_support_enabled configuration parameter.

A dashboard I use is displaying a warning, what do I need to do?

A dashboard displays warnings when one or more panel visualizations or data sources in the dashboard have a dependency on Angular. Contact your system administrator to advise them of the issue or follow the preceding guidance on avoiding disruption.

How do I migrate an Angular plugin to React?

Depending on if it's a data source plugin, panel plugin, or app plugin the process will differ.

For panels, the rendering logic could in some cases be easily preserved, but all options need to be redone to use the declarative options framework. For data source plugins the query editor and config options will likely need a total rewrite.

How do I encourage a community plugin to migrate?

We encourage you to locate the repository of the corresponding plugin and create or upvote an Issue within it if you are using a plugin that is still based on Angular.

Links

- Migrate Angular to React
- Build a panel plugin
- Build a data source plugin
- List of current Angular plugins

Plugins using AngularJS

The use of AngularJS in Grafana has been deprecated in favor of React. Support for AngularJS will be turned off by default in Grafana 11.

This page explains how Grafana users might be impacted by the removal of Angular support based on plugins dependent on this legacy framework. You will also see if there is a migration option available for a given plugin.

NOTE

We are greatly appreciative of the developers who have contributed plugins to the Grafana ecosystem. Guidance on migrating a plugin to React can be found in our migration guide.

What should I do with the list of AngularJS plugins?

Refer to the table below and take the appropriate action for you.

- Consider the advice on whether to update, migrate to a listed alternative, or explore the Grafana plugins catalog to find the most suitable option for your use case.
- Use our detect-angular-dashboards open source tooling to list dashboards which have a dependency on Angular plugins.
- Check your Grafana instances for usage of these plugins. Refer to the documentation on browsing installed plugins.
- Customers of Grafana Enterprise and users of Grafana Cloud can also leverage usage insights to prioritize any migration efforts.
- Review the plugin source repositories to add your support to any migration issues or consider forking the repo.

NOTE

If you want to add any specific migration guidance for your plugin here or update our assessment, please open a PR by clicking **Suggest an edit** at the bottom of this page.

Private plugins

Grafana OSS and Grafana Enterprise support the creation of private plugins for use on local instances. These plugins may have a dependency on AngularJS and require an update.

The `detect-angular-dashboards` tool listed above will include private plugins in its report **if the Grafana version is v10.1.0 or later**.

Additionally, warning icons and messages will be displayed when browsing the catalog via **Administration** > **Plugins and Data** > **Plugins** in your local instance.

Automatic migration of plugins

Certain legacy Grafana panel plugins automatically update to their React-based replacements when Angular support is disabled. This migration is usually available within the panel options, as shown in the screenshot below for World Map. Automatic migration can be triggered by setting the feature toggle `autoMigrateOldPanels` to `true`.

Automatic migration is supported for the plugins shown in the following table. Each of the target plugins are included in Grafana as Core plugins which don't require installation.

Plugin	Migration target
Graph (old)	Time Series
Singlestat	Stat
Stat (old)	Stat
Table (old)	Table
Worldmap	Geomap

A dashboard must still be saved with the new plugin ID to persist the change.

AngularJS-based plugins

This table lists plugins which we have detected as having a dependency on AngularJS. For alternatives, consider included Visualizations and Data sources, as well as external plugins from the catalog.

Plugin ID	Name	Action
grafana-worldmap-panel	Worldmap Panel	Migrate - Geomap (core) replaced Worldmap - Note this should migrate when Angular is disabled.
natel-discrete-panel	Discrete	Migrate - Browse included visualizations and plugins catalog for potential alternatives.
vonage-status-panel	Status Panel	Migrate - Consider Stat (core) or Polystat as potential alternatives.
grafana-simple-json-datasource	SimpleJson	Migrate - Consider Infinity as a potential alternative.
natel-plotly-panel	Plotly	Migrate - Consider alternative nline-plotlyjs-panel plugin.
agenty-flowcharting-panel	FlowCharting	Migrate - Consider Canvas (core) or Diagram as potential alternatives.
camptocamp-prometheus-alertmanager-datasource	Prometheus AlertManager	Update - Note the minimum version for React is 2.0.0.
briangann-gauge-panel	D3 Gauge	Update - Note the minimum version for React is 2.0.0. We recommend the latest.
yesoreyeram-boomtable-panel	Boom Table	Migrate - Consider Table (core) and transformations as appropriate.
briangann-datatable-panel	Datatable Panel	Wait - New version with React migration is planned.
flant-statusmap-panel	Statusmap	Migrate - Browse included visualizations and plugins catalog for potential alternatives.
michaeldmoore-multistat-panel	Multistat	Migrate - Browse included visualizations and plugins catalog for potential alternatives.
pr0ps-trackmap-panel	TrackMap	Migrate - Browse included visualizations and plugins catalog for potential alternatives.
snuids-trafficlights-panel	Traffic Lights	Migrate - Consider Traffic Light as a potential alternative.
vertamedia-clickhouse-datasource	Altinity plugin for ClickHouse	Update - Note the minimum version for React is 3.0.0.
petrslavotinek-carpetplot-panel	Carpet plot	Migrate - Browse included visualizations and plugins catalog for potential alternatives.
ryantxu-ajax-panel	AJAX	Migrate - Browse included visualizations and plugins catalog for potential alternatives.

Plugin ID	Name	Action
michaeldmoore-annunciator-panel	Annunciator	Migrate - Consider Stat (core).
marcuscalidus-svg-panel	SVG	Migrate - Consider alternatives such as Canvas (core), Colored SVG, or others.
neocat-cal-heatmap-panel	Cal-HeatMap	Migrate - Consider Heatmap (core) visualization.
blackmirror1-singlestat-math-panel	Singlestat Math	Migrate - Browse included visualizations and plugins catalog for potential alternatives.
scadavis-synoptic-panel	SCADAvis Synoptic Panel	Update - Note the minimum version for React is 2.0.
farski-blendstat-panel	Blendstat	Migrate - Consider Stat (core) and transformations as appropriate.
savantly-heatmap-panel	Heatmap	Migrate - Consider Heatmap (core) visualization.
mtanda-histogram-panel	Histogram	Migrate - Consider included Histogram visualization.
snuids-radar-panel	Radar Graph	Migrate - Browse included visualizations and plugins catalog for potential alternatives.
fatcloud-windrose-panel	WindRose	Migrate - Consider alternative Operator Windrose plugin.
bessler-pictureit-panel	PictureIt	Migrate - Consider alternative ePict plugin.
digrich-bubblechart-panel	Bubble Chart	Update - Note the minimum version for React is 2.0.1. We recommend the latest.
corpglory-progresslist-panel	Progress List	Migrate - Browse included visualizations and plugins catalog for potential alternatives.
aidanmountford-html-panel	HTML	Migrate - Consider alternatives such as Text (core), HTML, or others.
fifemon-graphql-datasource	GraphQL Data Source	Wait - Removal of AngularJS is planned. Consider Infinity plugin as alternative.
goshposh-metaqueries-datasource	MetaQueries	Migrate - Browse included data sources and plugins catalog for potential alternatives.
mxswat-separator-panel	Separator	Migrate - Consider alternative Text panel (core) which can be empty and used as a separator.
natel-influx-admin-panel	Influx Admin	Migrate - Browse included data sources and plugins catalog for potential alternatives.
doitintl-bigquery-datasource	Google BigQuery	Migrate - Consider Grafana Big Query plugin.

Plugin ID	Name	Action
satellogic-3d-globe-panel	3D Globe Panel	Migrate - Browse included visualizations and plugins catalog for potential alternatives.
foursquare-clouderamanager-datasource	Cloudera Manager	Migrate - Browse included data sources and plugins catalog for potential alternatives.
grafana-splunk-datasource	Splunk	Update - Note the minimum version for React is 4.1.0. We recommend the latest.
grafana-singlestat-panel	Singlestat	Migrate - Consider Stat panel (core).
blackmirror1-statusbygroup-panel	Status By Group Panel	Migrate - Browse included visualizations and plugins catalog for potential alternatives.
novalabs-annotations-panel	Annotation Panel	Migrate - Browse included visualizations and plugins catalog for potential alternatives.
jasonlashua-prtg-datasource	PRTG	Migrate - Browse included data sources and plugins catalog for potential alternatives.
ryantxu-annolist-panel	Annotation List	Migrate - Consider annotations list (core).
cloudflare-app	Cloudflare Grafana App	Migrate - Consider using the Cloudflare Dashboard or DNS Analytics API.
smartmakers-trafficlight-panel	TrafficLight	Migrate - Consider Traffic Light as a potential alternative.
zuburqan-parity-report-panel	Parity Report	Migrate - Browse included visualizations and plugins catalog for potential alternatives.
citilogics-geoloop-panel	GeoLoop	Migrate - Browse included visualizations and plugins catalog for potential alternatives.
xginn8-pagerduty-datasource	Pagerduty	Wait - We are developing an Enterprise plugin for Pagerduty targeted for availability in Q1 2024. Note that all roadmap items are subject to change.
gretamosa-topology-panel	Topology Panel	Migrate - Browse included visualizations and plugins catalog for potential alternatives.
fzakaria-simple-annotations-datasource	Simple Annotations	Migrate - Check for annotations support within your data sources to remove dependency on this plugin.
oci-metrics-datasource	Oracle Cloud Infrastructure Metrics	Update - Note the minimum version for React is 5.0.0.
abhisant-druid-datasource	Druid	Migrate - Migrate to replacement Druid plugin.

Plugin ID	Name	Action
devopsprodigy-kubegraf-app	DevOpsProdigy KubeGraf	Migrate - Consider Grafana Kubernetes Monitoring (Grafana Cloud only).
mtanda-heatmap-epoch-panel	HeatmapEpoch	Migrate - Consider Heatmap (core) visualization.
alexandra-trackmap-panel	Track Map	Migrate - Browse included visualizations and plugins catalog for potential alternatives.
gnocchixyz-gnocchi-datasource	Gnocchi	Migrate - Browse included data sources and plugins catalog for potential alternatives.
tencentcloud-monitor-app	Tencent Cloud Monitor	Migrate - Browse included data sources and plugins catalog for potential alternatives.
andig-darksky-datasource	DarkSky	Remove - Note that support for the DarkSky API was ended by Apple in March 2023.
mtanda-google-calendar-datasource	GoogleCalendar	Wait - Migration to React is planned.
ntop-ntopng-datasource	ntopng	Migrate - Consider InfluxDB (core), with additional guidance available here.
ayoungprogrammer-finance-datasource	Finance	Migrate - Browse included data sources and plugins catalog for potential alternatives.
grafana-kairosdb-datasource	KairosDB	Migrate - Browse included data sources and plugins catalog for potential alternatives.
fastweb-openfalcon-datasource	Open-Falcon	Migrate - Browse included data sources and plugins catalog for potential alternatives.
praj-ams-datasource	Ambari Metrics	Migrate - Browse included data sources and plugins catalog for potential alternatives.
monasca-datasource	Monasca	Migrate - Browse included data sources and plugins catalog for potential alternatives.
grafana-strava-datasource	Strava	Update - Note the minimum version for React is 1.6.0. We recommend the latest.
gridprotectionalliance-osisoftpi-datasource	OSIsoft-PI	Update - Note the minimum version for React is 4.0.0. We recommend the latest.
monitoringartist-monitoringart-datasource	Monitoring Art	Migrate - Browse included visualizations and plugins catalog for potential alternatives.

Plugin ID	Name	Action
hawkular-datasource	Hawkular	Migrate - Browse included data sources and plugins catalog for potential alternatives.
ovh-warp10-datasource	Warp 10	Migrate - Browse included data sources and plugins catalog for potential alternatives.
natel-usgs-datasource	USGS Water Services	Migrate - Browse included data sources and plugins catalog for potential alternatives.
radensolutions-netxms-datasource	NetXMS	Migrate - Browse included data sources and plugins catalog for potential alternatives.
ibm-apm-datasource	IBM APM	Migrate - Browse included data sources and plugins catalog for potential alternatives.
cognitedata-datasource	Cognite Data Fusion	Update - Note the minimum version for React is 4.0.0. We recommend the latest.
linksmart-sensorthings-datasource	LinkSmart SensorThings	Migrate - Browse included data sources and plugins catalog for potential alternatives.
pue-solr-datasource	Solr	Migrate - Consider this guidance on using solr-exporter for prometheus.
paytm-kapacitor-datasource	KapacitorSimpleJson	Migrate - Browse included data sources and plugins catalog for potential alternatives.
oci-logs-datasource	Oracle Cloud Infrastructure Logs	Update - Note the minimum version for React is 4.0.0.
gridprotectionalliance-openhistorian-datasource	openHistorian	Wait - Note that new version with React migration is planned.
devicehive-devicehive-datasource	DeviceHive	Migrate - Browse included data sources and plugins catalog for potential alternatives.
rackerlabs-blueflood-datasource	Blueflood	Migrate - Browse included data sources and plugins catalog for potential alternatives.
udoprog-heroic-datasource	Heroic	Migrate - Note that Heroic DB has been discontinued.
akumuli-datasource	Akumuli	Migrate - Browse included data sources and plugins catalog for potential alternatives.
bmchelix-ade-datasource	BMC Helix	Migrate - Browse included data sources and plugins catalog for potential alternatives.
sidewinder-datasource	Sidewinder	Migrate - Browse included data sources and plugins catalog for potential

Plugin ID	Name	Action
		alternatives.
linksmart-hds-datasource	LinkSmart HDS Datasource	Migrate - browse included data sources and plugins catalog for potential alternatives.
skydive-datasource	Skydive	Migrate - Browse included data sources and plugins catalog for potential alternatives.
quasardb-datasource	QuasarDB	Migrate - Browse included data sources and plugins catalog for potential alternatives.
spotify-heroic-datasource	Heroic	Migrate - Note that Heroic DB has been discontinued.
grafana-es-open-distro-datasource	Open Distro for Elasticsearch	Migrate - Note that OpenSearch replaced Open Distro for Elasticseach.
humio-datasource	Humio	Migrate - Browse included data sources and plugins catalog for potential alternatives.
yeya24-chaosmesh-datasource	Chaos Mesh	Migrate - Note that plugin was replaced by chaosmeshorg-datasource.
kentik-connect-app	Kentik Connect Pro	Update - Note the minimum version for React is 1.7.0.
chaosmeshorg-datasource	Chaos Mesh	Update - Note the minimum version for React is 3.0.0.
aquaqanalytics-kdbadaptor-datasource	kdb+	Migrate - Note that kdb+ backend replaced kdb.+.
alexanderzobnin-zabbix-app	Zabbix	Update - Note the minimum version for React is 4.3.0. We recommend the latest. Recently brought under Grafana signature.
bosun-app	Bosun	Migrate - Browse included data sources and plugins catalog for potential alternatives.
belugacdn-app	BelugaCDN	Migrate - Browse included data sources and plugins catalog for potential alternatives.
grafana-azure-data-explorer-datasource	Azure Data Explorer Datasource	Update - The minimum supported version is 3.5.1. We recommend the latest.
ddurieux-glpi-app	glpi	Migrate - Browse included data sources and plugins catalog for potential alternatives.
fetzerch-sunandmoon-datasource	Sun and Moon	Update - Note the minimum version for React is 0.3.0.
grafana-clock-panel	Clock	Update - Note the minimum version for React is 1.1.0. We recommend the latest.

Plugin ID	Name	Action
grafana-github-datasource	GitHub	Update - Note the minimum version for React is 1.3.3. We recommend the latest.
grafana-datadog-datasource	Datadog	Update - Note the minimum version for React is 3.0.0. We recommend the latest.
grafana-gitlab-datasource	Gitlab	Update - Note the minimum version for React is 1.1.0. We recommend the latest.
grafana-iot-twinmaker-app	AWS IoT TwinMaker App	Update - Note the minimum version for React is 1.6.3. We recommend the latest.
grafana-newrelic-datasource	New Relic	Update - Note the minimum version for React is 3.0.0. We recommend the latest.
grafana-opensearch-datasource	Opensearch	Update - Note the minimum version for React is 2.0.0. We recommend the latest.
grafana-oracle-datasource	Oracle	Update - Note the minimum version for React is 2.0.6. We recommend the latest.
grafana-piechart-panel	Pie Chart	Migrate - Note that Pie Chart (core) replaced Pie Chart.
grafana-polystat-panel	Polystat	Update - Note the minimum version for React is 2.0.0. We recommend the latest.
grafana-servicenow-datasource	ServiceNow	Update - Note the minimum version for React is 2.0.2. We recommend the latest.
grafana-synthetic-monitoring-app	Synthetic Monitoring	Update - Note the minimum version for React is 0.7.3. We recommend the latest.
grafana-wavefront-datasource	Wavefront	Update - Note the minimum version for React is 2.0.0. We recommend the latest.
hadesarchitect-cassandra-datasource	Apache Cassandra	Update - Note the minimum version for React is 2.1.1. We recommend the latest.
instana-datasource	Instana	Update - Note the minimum version for React is 3.0.0. We recommend the latest.
jdbranham-diagram-panel	Diagram	Update - Note the minimum version for React is 1.7.1. We recommend the latest.
larona-epict-panel	ePict	Update - Note the minimum version for React is 2.0.0. We recommend the latest.
moogsoft-aiops-app	Moogsoft AIOps	Update - Note the minimum version for React is 9.0.0.
opennms-helm-app	OpenNMS Helm	Migrate - Note that OpenNMS Plugin for Grafana replaced OpenNMS Helm.
percona-percona-app	Percona	Migrate - Consider use of Percona dashboards.
novatec-sdg-panel	Service Dependency Graph	Update - Note the minimum version for React is 4.0.3.

Plugin ID	Name	Action
pierosavi-imageit-panel	ImageIt	Migrate - Consider ePict or browse plugins catalog for potential alternatives.
redis-app	Redis Application	Update - Note the minimum version for React is 1.2.0. We recommend the latest.
sbueringer-consul-datasource	Consul	Migrate - Browse included data sources and plugins catalog for potential alternatives.
simpod-json-datasource	JSON	Update - Note the minimum version for React is 0.3.0. We recommend the latest.
singlestat	Singlestat	Migrate - Note that Stat (core) replaced Singlestat.
sni-pnp-datasource	PNP	Update - Note the minimum version for React is 2.0.0. We recommend the latest.
sni-thruk-datasource	Thruk	Update - Note the minimum version for React is 2.0.0. We recommend the latest.
stagemonitor-elasticsearch-app	stagemonitor Elasticsearch	Migrate - Browse included data sources and plugins catalog for potential alternatives.
tdengine-datasource	TDengine Datasource	Update - Note the minimum version for React is 3.3.0. We recommend the latest.
vertica-grafana-datasource	Vertica	Update - Note the minimum version for React is 2.0.0. We recommend the latest.
voxter-app	Voxter VoIP Platform Metrics	Migrate - Browse included data sources and plugins catalog for potential alternatives.
graph	Graph (old)	Migrate - Note that this is replaced by Time Series (core) - This plugin should migrate when Angular is disabled. Also consider Bar Chart or Histogram if appropriate.
table-old	Table (old)	Migrate - Note that this is replaced by Table (core) - This plugin should migrate when AngularJS is disabled.
shorelinesoftware-shoreline-datasource	Shoreline Data Source	Update - Note the minimum version for React is 1.2.1. We recommend the latest.

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Grafana CLI

Grafana CLI is a small executable that is bundled with Grafana server. It can be executed on the same machine Grafana server is running on. Grafana CLI has `plugins` and `admin` commands, as well as global options.

To list all commands and options:

```
grafana cli -h
```

Copy

Invoking Grafana CLI

To invoke Grafana CLI, add the path to the grafana binaries in your `PATH` environment variable. Alternately, if your current directory is the `bin` directory, use `./grafana cli`. Otherwise, you can specify full path to the CLI. For example, on Linux `/usr/share/grafana/bin/grafana` and on Windows `C:\Program Files\GrafanaLabs\grafana\bin\grafana.exe`, and invoke it with `grafana cli`.

NOTE

Some commands, such as installing or removing plugins, require `sudo` on Linux. If you are on Windows, run Windows PowerShell as Administrator.

Grafana CLI command syntax

The general syntax for commands in Grafana CLI is:

bash

Copy

```
grafana cli [global options] command [command options] [arguments...]
```

Global options

Grafana CLI allows you to temporarily override certain Grafana default settings. Except for `--help` and `--version`, most global options are only used by developers.

Each global option applies only to the command in which it is used. For example, `--pluginsDir value` does not permanently change where Grafana saves plugins. It only changes it for command in which you apply the option.

Display Grafana CLI help

`--help` or `-h` displays the help, including default paths and Docker configuration information.

Example:

```bash
grafana cli -h
```

Display Grafana CLI version

`--version` or `-v` prints the version of Grafana CLI currently running.

Example:

```bash
grafana cli -v
```

Override default plugin directory

`--pluginsDir value` overrides the path to where your local Grafana instance stores plugins. Use this option if you want to install, update, or remove a plugin somewhere other than the default directory ("/var/lib/grafana/plugins") [$GF_PLUGIN_DIR].

Example:

```bash
grafana cli --pluginsDir "/var/lib/grafana/devplugins" plugins install <plugin-id>
```

Override default plugin repo URL

`--repo value` allows you to download and install or update plugins from a repository other than the default Grafana repo.

Example:

```bash
grafana cli --repo "https://example.com/plugins" plugins install <plugin-id>
```

Override default plugin .zip URL

`--pluginUrl value` allows you to download a .zip file containing a plugin from a local URL instead of downloading it from the default Grafana source.

Example:

```bash
grafana cli --pluginUrl https://company.com/grafana/plugins/<plugin-id>-<plugin-version>.zip plugi
```

Override Transport Layer Security

Warning: Turning off TLS is a significant security risk. We do not recommend using this option.

`--insecure` allows you to turn off Transport Layer Security (TLS) verification (insecure). You might want to do this if you are downloading a plugin from a non-default source.

Example:

```bash
grafana cli --insecure --pluginUrl https://company.com/grafana/plugins/<plugin-id>-<plugin-versior
```

Enable debug logging

`--debug` or `-d` enables debug logging. Debug output is returned and shown in the terminal.

Example:

```bash
grafana cli --debug plugins install <plugin-id>
```

Override a configuration setting

`--configOverrides` is a command line argument that acts like an environmental variable override.

For example, you can use it to redirect logging to another file (maybe to log plugin installations in Grafana Cloud) or when resetting the admin password and you have non-default values for some important configuration value (like where the database is located).

Example:

```bash
grafana cli --configOverrides cfg:default.paths.log=/dev/null plugins install <plugin-id>
```

Override homepath value

Sets the path for the Grafana install/home path, defaults to working directory. You do not need to use this if you are in the Grafana installation directory when using the CLI.

Example:

```bash
grafana cli --homepath "/usr/share/grafana" admin reset-admin-password <new password>
```

Override config file

`--config value` overrides the default location where Grafana expects the configuration file. Refer to Configuration for more information about configuring Grafana and default configuration file locations.

Example:

```bash
grafana cli --config "/etc/configuration/" admin reset-admin-password mynewpassword
```

Plugins commands

Grafana CLI allows you to install, upgrade, and manage your Grafana plugins. For more information about installing plugins, refer to plugins page.

All listed commands apply to the Grafana default repositories and directories. You can override the defaults with Global Options.

List available plugins

```bash
```

```
grafana cli plugins list-remote
```

Install the latest version of a plugin

bash Copy

```
grafana cli plugins install <plugin-id>
```

Install a specific version of a plugin

bash Copy

```
grafana cli plugins install <plugin-id> <version>
```

List installed plugins

bash Copy

```
grafana cli plugins ls
```

Update all installed plugins

bash Copy

```
grafana cli plugins update-all
```

Update one plugin

bash Copy

```
grafana cli plugins update <plugin-id>
```

Remove one plugin

bash Copy

```
grafana cli plugins remove <plugin-id>
```

Admin commands

Admin commands are only available in Grafana 4.1 and later.

Show all admin commands

```bash
grafana cli admin
```

Reset admin password

`grafana cli admin reset-admin-password <new password>` resets the password for the admin user using the CLI. You might need to do this if you lose the admin password.

If there are two flags being used to set the homepath and the config file path, then running the command returns this error:

> Could not find config defaults, make sure homepath command line parameter is set or working directory is homepath

To correct this, use the `--homepath` global option to specify the Grafana default homepath for this command:

```bash
grafana cli --homepath "/usr/share/grafana" admin reset-admin-password <new password>
```

If you have not lost the admin password, we recommend that you change the user password either in the User Preferences or in the Server Admin > User tab.

If you need to set the password in a script, then you can use the Grafana User API.

Migrate data and encrypt passwords

`data-migration` runs a script that migrates or cleans up data in your database.

`encrypt-datasource-passwords` migrates passwords from unsecured fields to secure_json_data field. Returns `ok` unless there is an error. Safe to execute multiple times.

Example:

```bash
```

```
grafana cli admin data-migration encrypt-datasource-passwords
```

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Release notes

Here you can find detailed release notes that list everything included in past releases, as well as notices about deprecations, breaking changes, and changes related to plugin development.

NOTE

As of Grafana v9.2 we no longer publish release notes, which are redundant with other release lists that we publish:

- For details about new features, deprecations, and breaking changes in new Grafana releases, see What's New in Grafana.
- For lists of changes to Grafana, with links to pull requests and related issues when available, see the Changelog.

- Release notes for 9.0.6
- Release notes for 9.0.5
- Release notes for 9.0.4
- Release notes for 9.0.3
- Release notes for 9.0.2
- Release notes for 9.0.1
- Release notes for 9.0.0
- Release notes for 9.0.0-beta3
- Release notes for 9.0.0-beta2
- Release notes for 9.0.0-beta1
- Release notes for 8.5.13
- Release notes for 8.5.11
- Release notes for 8.5.10
- Release notes for 8.5.9
- Release notes for 8.5.6
- Release notes for 8.5.5
- Release notes for 8.5.4
- Release notes for 8.5.3
- Release notes for 8.5.2
- Release notes for 8.5.1
- Release notes for 8.5.0
- Release notes for 8.5.0-beta1
- Release notes for 8.4.11
- Release notes for 8.4.10
- Release notes for 8.4.7
- Release notes for 8.4.6
- Release notes for 8.4.5
- Release notes for 8.4.4
- Release notes for 8.4.3
- Release notes for 8.4.2
- Release notes for 8.4.1
- Release notes for 8.4.0-beta1
- Release notes for 8.3.11
- Release notes for 8.3.7
- Release notes for 8.3.6

- Release notes for 8.3.5
- Release notes for 8.3.4
- Release notes for 8.3.3
- Release notes for 8.3.2
- Release notes for 8.3.1
- Release notes for 8.3.0
- Release notes for 8.3.0-beta2
- Release notes for 8.3.0-beta1
- Release notes for 8.2.7
- Release notes for 8.2.6
- Release notes for 8.2.5
- Release notes for 8.2.4
- Release notes for 8.2.3
- Release notes for 8.2.2
- Release notes for 8.2.1
- Release notes for 8.2.0
- Release notes for 8.2.0-beta2
- Release notes for 8.2.0-beta1
- Release notes for 8.1.8
- Release notes for 8.1.7
- Release notes for 8.1.6
- Release notes for 8.1.5
- Release notes for 8.1.4
- Release notes for 8.1.3
- Release notes for 8.1.2
- Release notes for 8.1.1
- Release notes for 8.1.0
- Release notes for 8.1.0-beta3
- Release notes for 8.1.0-beta2
- Release notes for 8.1.0-beta1
- Release notes for 8.0.7
- Release notes for 8.0.6
- Release notes for 8.0.5
- Release notes for 8.0.4
- Release notes for 8.0.3

- Release notes for 8.0.2
- Release notes for 8.0.1
- Release notes for 8.0.0
- Release notes for 8.0.0-beta3
- Release notes for 8.0.0-beta2
- Release notes for 8.0.0-beta1
- Release notes for 7.5.15
- Release notes for 7.5.13
- Release notes for 7.5.12
- Release notes for 7.5.11
- Release notes for 7.5.10
- Release notes for 7.5.9
- Release notes for 7.5.8
- Release notes for 7.5.7
- Release notes for 7.5.6
- Release notes for 7.5.5
- Release notes for 7.5.4
- Release notes for 7.5.3
- Release notes for 7.5.2
- Release notes for 7.5.1
- Release notes for 7.5.0
- Release notes for 7.5.0-beta2
- Release notes for 7.5.0-beta1
- Release notes for 7.4.5
- Release notes for 7.4.3
- Release notes for 7.4.2
- Release notes for 7.4.1
- Release notes for 7.4.0
- Release notes for 7.3.10
- Release notes for 7.3.7
- Release notes for 7.3.6
- Release notes for 7.3.5
- Release notes for 7.3.4
- Release notes for 7.3.3
- Release notes for 7.3.2

- Release notes for 7.3.1
- Release notes for 7.3.0

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Install Grafana on Debian or Ubuntu

This topic explains how to install Grafana dependencies, install Grafana on Linux Debian or Ubuntu, and start the Grafana server on your Debian or Ubuntu system.

There are multiple ways to install Grafana: using the Grafana Labs APT repository, by downloading a `.deb` package, or by downloading a binary `.tar.gz` file. Choose only one of the methods below that best suits your needs.

NOTE

If you install via the `.deb` package or `.tar.gz` file, then you must manually update Grafana for each new version.

The following video demonstrates how to install Grafana on Debian and Ubuntu as outlined in this document:

Install from APT repository

If you install from the APT repository, Grafana automatically updates when you run `apt-get update`.

Grafana Version	Package	Repository
Grafana Enterprise	grafana-enterprise	`https://apt.grafana.com stable main`
Grafana Enterprise (Beta)	grafana-enterprise	`https://apt.grafana.com beta main`
Grafana OSS	grafana	`https://apt.grafana.com stable main`
Grafana OSS (Beta)	grafana	`https://apt.grafana.com beta main`

NOTE

Grafana Enterprise is the recommended and default edition. It is available for free and includes all the features of the OSS edition. You can also upgrade to the full Enterprise feature set, which has support for Enterprise plugins.

Complete the following steps to install Grafana from the APT repository:

1. Install the prerequisite packages:

```bash
```

Copy

```
sudo apt-get install -y apt-transport-https software-properties-common wget
```

2. Import the GPG key:

```bash
sudo mkdir -p /etc/apt/keyrings/
wget -q -O - https://apt.grafana.com/gpg.key | gpg --dearmor | sudo tee /etc/apt/keyrings/gra
```

3. To add a repository for stable releases, run the following command:

```bash
echo "deb [signed-by=/etc/apt/keyrings/grafana.gpg] https://apt.grafana.com stable main" | su
```

4. To add a repository for beta releases, run the following command:

```bash
echo "deb [signed-by=/etc/apt/keyrings/grafana.gpg] https://apt.grafana.com beta main" | sudo
```

5. Run the following command to update the list of available packages:

```bash
# Updates the list of available packages
sudo apt-get update
```

6. To install Grafana OSS, run the following command:

```bash
# Installs the latest OSS release:
sudo apt-get install grafana
```

7. To install Grafana Enterprise, run the following command:

```bash
# Installs the latest Enterprise release:
sudo apt-get install grafana-enterprise
```

Install Grafana using a deb package or as a standalone binary

If you choose not to install Grafana using APT, you can download and install Grafana using the deb package or as a standalone binary.

Complete the following steps to install Grafana using DEB or the standalone binaries:

1. Navigate to the Grafana download page.
2. Select the Grafana version you want to install.
   - The most recent Grafana version is selected by default.
   - The **Version** field displays only tagged releases. If you want to install a nightly build, click **Nightly Builds** and then select a version.
3. Select an **Edition**.
   - **Enterprise:** This is the recommended version. It is functionally identical to the open source version, but includes features you can unlock with a license, if you so choose.
   - **Open Source:** This version is functionally identical to the Enterprise version, but you will need to download the Enterprise version if you want Enterprise features.
4. Depending on which system you are running, click the **Linux** or **ARM** tab on the download page.
5. Copy and paste the code from the download page into your command line and run.

Uninstall on Debian or Ubuntu

Complete any of the following steps to uninstall Grafana.

To uninstall Grafana, run the following commands in a terminal window:

1. If you configured Grafana to run with systemd, stop the systemd servivce for Grafana server:

```shell
sudo systemctl stop grafana-server
```

2. If you configured Grafana to run with init.d, stop the init.d service for Grafana server:

```shell
sudo service grafana-server stop
```

3. To uninstall Grafana OSS:

```shell
```

```
sudo apt-get remove grafana
```

4 To uninstall Grafana Enterprise:

shell Copy

```
sudo apt-get remove grafana-enterprise
```

5 Optional: To remove the Grafana repository:

bash Copy

```
sudo rm -i /etc/apt/sources.list.d/grafana.list
```

Next steps

- Start the Grafana server

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Install Grafana on macOS

You can use Grafana Cloud to avoid installing, maintaining, and scaling your own instance of Grafana. Create a free account to get started, which includes free forever access to 10k metrics, 50GB logs, 50GB traces, 500VUh k6 testing & more.

This page explains how to install Grafana on macOS.

The following video demonstrates how to install Grafana on macOS as outlined in this document:

Install Grafana on macOS using Homebrew

To install Grafana on macOS using Homebrew, complete the following steps:

1. On the Homebrew homepage, search for Grafana.

   The last stable and released version is listed.

2. Open a terminal and run the following commands:

   ```
   brew update
   brew install grafana
   ```

   Copy

   The brew page downloads and untars the files into:

   - `/usr/local/Cellar/grafana/[version]` (Intel Silicon)
   - `/opt/homebrew/Cellar/grafana/[version]` (Apple Silicon)

3. To start Grafana, run the following command:

   bash

   ```
   brew services start grafana
   ```

   Copy

Install standalone macOS binaries

To install Grafana on macOS using the standalone binaries, complete the following steps:

1. Navigate to the Grafana download page.
2. Select the Grafana version you want to install.
   - The most recent Grafana version is selected by default.
   - The **Version** field displays only tagged releases. If you want to install a nightly build, click **Nightly Builds** and then select a version.
3. Select an **Edition**.
   - **Enterprise:** This is the recommended version. It is functionally identical to the open source version, but includes features you can unlock with a license, if you so choose.
   - **Open Source:** This version is functionally identical to the Enterprise version, but you will need to download the Enterprise version if you want Enterprise features.
4. Click **Mac**.
5. Copy and paste the code from the download page into your command line and run.
6. Untar the `gz` file and copy the files to the location of your preference.

7 To start Grafana service, go to the directory and run the command:

```bash
./bin/grafana server
```

Alternatively, watch the Grafana for Beginners video below:

Next steps

- Start the Grafana server

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Install Grafana on Windows

You can use Grafana Cloud to avoid installing, maintaining, and scaling your own instance of Grafana. Create a free account to get started, which includes free forever access to 10k metrics, 50GB logs, 50GB traces, 500VUh k6 testing & more.

The following video demonstrates how to install Grafana using the Windows standalone installer as outlined in this document:

You install Grafana using the Windows installer or using the standalone Windows binary file.

1. Navigate to the Grafana download page.
2. Select the Grafana version you want to install.
   - The most recent Grafana version is selected by default.
   - The **Version** field displays only tagged releases. If you want to install a nightly build, click **Nightly Builds** and then select a version.
3. Select an **Edition**.
   - **Enterprise:** This is the recommended version. It is functionally identical to the open source version, but includes features you can unlock with a license, if you so choose.
   - **Open Source:** This version is functionally identical to the Enterprise version, but you will need to download the Enterprise version if you want Enterprise features.
4. Click **Windows**.
5. To use the Windows installer, complete the following steps:

   a. Click **Download the installer**.

   b. Open and run the installer.
6. To install the standalone Windows binary, complete the following steps:

   a. Click **Download the zip file**.

   b. Right-click the downloaded file, select **Properties**, select the `unblock` checkbox, and click `OK`.

   c. Extract the ZIP file to any folder.

Start Grafana by executing `grafana-server.exe`, located in the `bin` directory, preferably from the command line. If you want to run Grafana as a Windows service, then download NSSM. It is very easy to add Grafana as a Windows service using that tool.

1. To run Grafana, open your browser and go to the Grafana port (http://localhost:3000/ is default) and then follow the instructions in Getting Started.

   Note: The default Grafana port is `3000`. This port might require extra permissions on Windows. If it does not appear in the default port, you can change the port number.

2. To change the port, perform the following steps:

   a. Open the `conf` directory and copy `sample.ini` to `custom.ini`.

   Note: You should edit `custom.ini`, never `defaults.ini`.

   b. Edit `custom.ini` and uncomment the `http_port` configuration option.

`;` is the comment character in ini files.

c. Change the port to `8080` or something similar.

Port `8080` should not require extra Windows privileges.

Next steps

- Start the Grafana server

Run Grafana Docker image

This topic guides you through installing Grafana via the official Docker images. Specifically, it covers running Grafana via the Docker command line interface (CLI) and docker-compose.

Grafana Docker images come in two editions:

- **Grafana Enterprise**: `grafana/grafana-enterprise`
- **Grafana Open Source**: `grafana/grafana-oss`

> **Note:** The recommended and default edition of Grafana is Grafana Enterprise. It is free and includes all the features of the OSS edition. Additionally, you have the option to upgrade to the full Enterprise feature set, which includes support for Enterprise plugins.

The default images for Grafana are created using the Alpine Linux project and can be found in the Alpine official image. For instructions on configuring a Docker image for Grafana, refer to Configure a Grafana Docker image.

Run Grafana via Docker CLI

This section shows you how to run Grafana using the Docker CLI.

> **Note:** If you are on a Linux system (for example, Debian or Ubuntu), you might need to add `sudo` before the command or add your user to the `docker` group. For more information, refer to Linux post-installation steps for Docker Engine.

To run the latest stable version of Grafana, run the following command:

```bash
docker run -d -p 3000:3000 --name=grafana grafana/grafana-enterprise
```

Where:

- `docker run` is a Docker CLI command that runs a new container from an image
- `-d` (`--detach`) runs the container in the background
- `-p <host-port>:<container-port>` (`--publish`) publish a container's port(s) to the host, allowing you to reach the container's port via a host port. In this case, we can reach the container's port `3000` via the host's port `3000`
- `--name` assign a logical name to the container (e.g. `grafana`). This allows you to refer to the container by name instead of by ID.
- `grafana/grafana-enterprise` is the image to run

Stop the Grafana container

To stop the Grafana container, run the following command:

```bash
```

```
# The `docker ps` command shows the processes running in Docker
docker ps

# This will display a list of containers that looks like the following:
CONTAINER ID   IMAGE   COMMAND   CREATED   STATUS   PORTS    NAMES
cd48d3994968   grafana/grafana-enterprise   "/run.sh"   8 seconds ago   Up 7 seconds   0.0.0.0:300

# To stop the grafana container run the command
# docker stop CONTAINER-ID or use
# docker stop NAME, which is `grafana` as previously defined
docker stop grafana
```

Save your Grafana data

By default, Grafana uses an embedded SQLite version 3 database to store configuration, users, dashboards, and other data. When you run Docker images as containers, changes to these Grafana data are written to the filesystem within the container, which will only persist for as long as the container exists. If you stop and remove the container, any filesystem changes (i.e. the Grafana data) will be discarded. To avoid losing your data, you can set up persistent storage using Docker volumes or bind mounts for your container.

Note: Though both methods are similar, there is a slight difference. If you want your storage to be fully managed by Docker and accessed only through Docker containers and the Docker CLI, you should choose to use persistent storage. However, if you need full control of the storage and want to allow other processes besides Docker to access or modify the storage layer, then bind mounts is the right choice for your environment.

Use Docker volumes (recommended)

Use Docker volumes when you want the Docker Engine to manage the storage volume.

To use Docker volumes for persistent storage, complete the following steps:

1. Create a Docker volume to be used by the Grafana container, giving it a descriptive name (e.g. `grafana-storage`). Run the following command:

bash Copy

```
# create a persistent volume for your data
docker volume create grafana-storage

# verify that the volume was created correctly
```

```
# you should see some JSON output
docker volume inspect grafana-storage
```

2. Start the Grafana container by running the following command:

```bash
# start grafana
docker run -d -p 3000:3000 --name=grafana \
  --volume grafana-storage:/var/lib/grafana \
  grafana/grafana-enterprise
```

Use bind mounts

If you plan to use directories on your host for the database or configuration when running Grafana in Docker, you must start the container with a user with permission to access and write to the directory you map.

To use bind mounts, run the following command:

```bash
# create a directory for your data
mkdir data

# start grafana with your user id and using the data directory
docker run -d -p 3000:3000 --name=grafana \
  --user "$(id -u)" \
  --volume "$PWD/data:/var/lib/grafana" \
  grafana/grafana-enterprise
```

Use environment variables to configure Grafana

Grafana supports specifying custom configuration settings using environment variables.

```bash
# enable debug logs

docker run -d -p 3000:3000 --name=grafana \
  -e "GF_LOG_LEVEL=debug" \
  grafana/grafana-enterprise
```

Install plugins in the Docker container

You can install plugins in Grafana from the official and community plugins page or by using a custom URL to install a private plugin. These plugins allow you to add new visualization types, data sources, and applications to help you better visualize your data.

Grafana currently supports three types of plugins: panel, data source, and app. For more information on managing plugins, refer to Plugin Management.

To install plugins in the Docker container, complete the following steps:

1. Pass the plugins you want to be installed to Docker with the `GF_INSTALL_PLUGINS` environment variable as a comma-separated list.

   This sends each plugin name to `grafana-cli plugins install ${plugin}` and installs them when Grafana starts.

   For example:

   bash Copy

   ```
   docker run -d -p 3000:3000 --name=grafana \
     -e "GF_INSTALL_PLUGINS=grafana-clock-panel, grafana-simple-json-datasource" \
     grafana/grafana-enterprise
   ```

2. To specify the version of a plugin, add the version number to the `GF_INSTALL_PLUGINS` environment variable.

   For example:

   bash Copy

   ```
   docker run -d -p 3000:3000 --name=grafana \
     -e "GF_INSTALL_PLUGINS=grafana-clock-panel 1.0.1" \
     grafana/grafana-enterprise
   ```

   Note: If you do not specify a version number, the latest version is used.

3. To install a plugin from a custom URL, use the following convention to specify the URL: `<url to plugin zip>;<plugin install directory name>`.

   For example:

   bash Copy

```
docker run -d -p 3000:3000 --name=grafana \
  -e "GF_INSTALL_PLUGINS=https://github.com/VolkovLabs/custom-plugin.zip;custom-plugin" \
  grafana/grafana-enterprise
```

Example

The following example runs the latest stable version of Grafana, listening on port 3000, with the container named `grafana`, persistent storage in the `grafana-storage` docker volume, the server root URL set, and the official clock panel plugin installed.

bash Copy

```
# create a persistent volume for your data
docker volume create grafana-storage

# start grafana by using the above persistent storage
# and defining environment variables

docker run -d -p 3000:3000 --name=grafana \
  --volume grafana-storage:/var/lib/grafana \
  -e "GF_SERVER_ROOT_URL=http://my.grafana.server/" \
  -e "GF_INSTALL_PLUGINS=grafana-clock-panel" \
  grafana/grafana-enterprise
```

Run Grafana via Docker Compose

Docker Compose is a software tool that makes it easy to define and share applications that consist of multiple containers. It works by using a YAML file, usually called `docker-compose.yaml`, which lists all the services that make up the application. You can start the containers in the correct order with a single command, and with another command, you can shut them down. For more information about the benefits of using Docker Compose and how to use it refer to Use Docker Compose.

Before you begin

To run Grafana via Docker Compose, install the compose tool on your machine. To determine if the compose tool is available, run the following command:

bash Copy

```
docker compose version
```

If the compose tool is unavailable, refer to Install Docker Compose.

Run the latest stable version of Grafana

This section shows you how to run Grafana using Docker Compose. The examples in this section use Compose version 3. For more information about compatibility, refer to Compose and Docker compatibility matrix.

Note: If you are on a Linux system (for example, Debian or Ubuntu), you might need to add `sudo` before the command or add your user to the `docker` group. For more information, refer to Linux post-installation steps for Docker Engine.

To run the latest stable version of Grafana using Docker Compose, complete the following steps:

1. Create a `docker-compose.yaml` file.

```bash
# first go into the directory where you have created this docker-compose.yaml file
cd /path/to/docker-compose-directory

# now create the docker-compose.yaml file
touch docker-compose.yaml
```

2. Now, add the following code into the `docker-compose.yaml` file.

   For example:

```bash
services:
  grafana:
    image: grafana/grafana-enterprise
    container_name: grafana
    restart: unless-stopped
    ports:
     - '3000:3000'
```

3. To run `docker-compose.yaml`, run the following command:

```bash
# start the grafana container
docker compose up -d
```

Where:

d = detached mode

up = to bring the container up and running

To determine that Grafana is running, open a browser window and type `IP_ADDRESS:3000`. The sign in screen should appear.

Stop the Grafana container

To stop the Grafana container, run the following command:

```bash
docker compose down
```

Note: For more information about using Docker Compose commands, refer to docker compose.

Save your Grafana data

By default, Grafana uses an embedded SQLite version 3 database to store configuration, users, dashboards, and other data. When you run Docker images as containers, changes to these Grafana data are written to the filesystem within the container, which will only persist for as long as the container exists. If you stop and remove the container, any filesystem changes (i.e. the Grafana data) will be discarded. To avoid losing your data, you can set up persistent storage using Docker volumes or bind mounts for your container.

Use Docker volumes (recommended)

Use Docker volumes when you want the Docker Engine to manage the storage volume.

To use Docker volumes for persistent storage, complete the following steps:

1. Create a `docker-compose.yaml` file

```bash
# first go into the directory where you have created this docker-compose.yaml file
cd /path/to/docker-compose-directory

# now create the docker-compose.yaml file
touch docker-compose.yaml
```

2. Add the following code into the `docker-compose.yaml` file.

```yaml
services:
  grafana:
    image: grafana/grafana-enterprise
    container_name: grafana
    restart: unless-stopped
    ports:
      - '3000:3000'
    volumes:
      - grafana-storage:/var/lib/grafana
volumes:
  grafana-storage: {}
```

3. Save the file and run the following command:

```bash
docker compose up -d
```

Use bind mounts

If you plan to use directories on your host for the database or configuration when running Grafana in Docker, you must start the container with a user that has the permission to access and write to the directory you map.

To use bind mounts, complete the following steps:

1. Create a `docker-compose.yaml` file

```bash
# first go into the directory where you have created this docker-compose.yaml file
cd /path/to/docker-compose-directory

# now create the docker-compose.yaml file
touch docker-compose.yaml
```

2. Create the directory where you will be mounting your data, in this case is `/data` e.g. in your current working directory:

```bash
mkdir $PWD/data
```

3 Now, add the following code into the `docker-compose.yaml` file.

```yaml
services:
  grafana:
    image: grafana/grafana-enterprise
    container_name: grafana
    restart: unless-stopped
    # if you are running as root then set it to 0
    # else find the right id with the id -u command
    user: '0'
    ports:
      - '3000:3000'
    # adding the mount volume point which we create earlier
    volumes:
      - '$PWD/data:/var/lib/grafana'
```

4 Save the file and run the following command:

```bash
docker compose up -d
```

Example

The following example runs the latest stable version of Grafana, listening on port 3000, with the container named `grafana`, persistent storage in the `grafana-storage` docker volume, the server root URL set, and the official clock panel plugin installed.

```bash
services:
  grafana:
    image: grafana/grafana-enterprise
    container_name: grafana
    restart: unless-stopped
    environment:
     - GF_SERVER_ROOT_URL=http://my.grafana.server/
     - GF_INSTALL_PLUGINS=grafana-clock-panel
    ports:
     - '3000:3000'
    volumes:
```

```
    - 'grafana_storage:/var/lib/grafana'
volumes:
  grafana_storage: {}
```

Note: If you want to specify the version of a plugin, add the version number to the `GF_INSTALL_PLUGINS` environment variable. For example: `-e "GF_INSTALL_PLUGINS=grafana-clock-panel 1.0.1,grafana-simple-json-datasource 1.3.5"`. If you do not specify a version number, the latest version is used.

Next steps

Refer to the Getting Started guide for information about logging in, setting up data sources, and so on.

Configure Docker image

Refer to Configure a Grafana Docker image page for details on options for customizing your environment, logging, database, and so on.

Configure Grafana

Refer to the Configuration page for details on options for customizing your environment, logging, database, and so on.

Deploy Grafana on Kubernetes

On this page, you will find instructions for installing and running Grafana on Kubernetes using Kubernetes manifests for the setup. If Helm is your preferred option, refer to Grafana Helm community charts.

Watch this video to learn more about installing Grafana on Kubernetes:

Before you begin

To follow this guide:

- You need the latest version of Kubernetes running either locally or remotely on a public or private cloud.
- If you plan to use it in a local environment, you can use various Kubernetes options such as minikube, kind, Docker Desktop, and others.
- If you plan to use Kubernetes in a production setting, it's recommended to utilize managed cloud services like Google Kubernetes Engine (GKE), Amazon Elastic Kubernetes Service (EKS), or Azure Kubernetes Service (AKS).

System requirements

This section provides minimum hardware and software requirements.

Minimum Hardware Requirements

- Disk space: 1 GB
- Memory: 750 MiB (approx 750 MB)
- CPU: 250m (approx 2.5 cores)

Supported databases

For a list of supported databases, refer to supported databases.

Supported web browsers

For a list of support web browsers, refer to supported web browsers.

> **NOTE**
>
> Enable port `3000` in your network environment, as this is the Grafana default port.

Deploy Grafana OSS on Kubernetes

This section explains how to install Grafana OSS using Kubernetes.

> **NOTE**

> If you want to install Grafana Enterprise on Kubernetes, refer to Deploy Grafana Enterprise on Kubernetes.

If you deploy an application in Kubernetes, it will use the default namespace which may already have other applications running. This can result in conflicts and other issues.

It is recommended to create a new namespace in Kubernetes to better manage, organize, allocate, and manage cluster resources. For more information about Namespaces, refer to the official Kubernetes documentation.

1. To create a namespace, run the following command:

   ```bash
   kubectl create namespace my-grafana
   ```

   In this example, the namespace is `my-grafana`

2. To verify and view the newly created namespace, run the following command:

   ```bash
   kubectl get namespace my-grafana
   ```

   The output of the command provides more information about the newly created namespace.

3. Create a YAML manifest file named `grafana.yaml`. This file will contain the necessary code for deployment.

   ```bash
   touch grafana.yaml
   ```

   In the next step you define the following three objects in the YAML file.

Object	Description
Persistent Volume Claim (PVC)	This object stores the data.
Service	This object provides network access to the Pod defined in the deployment.
Deployment	This object is responsible for creating the pods, ensuring they stay up to date, and managing Replicaset and Rolling updates.

4. Copy and paste the following contents and save it in the `grafana.yaml` file.

```yaml
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: grafana-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
---
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: grafana
  name: grafana
spec:
```

Copy

Expand code

5 Run the following command to send the manifest to the Kubernetes API server:

```bash
kubectl apply -f grafana.yaml --namespace=my-grafana
```

Copy

This command creates the PVC, Deployment, and Service objects.

6 Complete the following steps to verify the deployment status of each object.

a. For PVC, run the following command:

```bash
kubectl get pvc --namespace=my-grafana -o wide
```

Copy

b. For Deployment, run the following command:

```bash
kubectl get deployments --namespace=my-grafana -o wide
```

Copy

c. For Service, run the following command:

```bash
kubectl get svc --namespace=my-grafana -o wide
```

Access Grafana on Managed K8s Providers

In this task, you access Grafana deployed on a Managed Kubernetes provider using a web browser. Accessing Grafana via a web browser is straightforward if it is deployed on a Managed Kubernetes Provider as it uses the cloud provider's **LoadBalancer** to which the external load balancer routes are automatically created.

1. Run the following command to obtain the deployment information:

```bash
kubectl get all --namespace=my-grafana
```

The output returned should look similar to the following:

```bash
NAME                           READY   STATUS    RESTARTS   AGE
pod/grafana-69946c9bd6-kwjb6   1/1     Running   0          7m27s

NAME              TYPE           CLUSTER-IP     EXTERNAL-IP      PORT(S)          AGE
service/grafana   LoadBalancer   10.5.243.226   1.120.130.330   3000:31171/TCP   7m27s

NAME                      READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/grafana   1/1     1            1           7m29s

NAME                                 DESIRED   CURRENT   READY   AGE
replicaset.apps/grafana-69946c9bd6   1         1         1       7m30s
```

2. Identify the **EXTERNAL-IP** value in the output and type it into your browser.

   The Grafana sign-in page appears.

3. To sign in, enter `admin` for both the username and password.

4. If you do not see the EXTERNAL-IP, complete the following steps:

   a. Run the following command to do a port-forwarding of the Grafana service on port `3000`.

```bash
```

```
kubectl port-forward service/grafana 3000:3000 --namespace=my-grafana
```

For more information about port-forwarding, refer to Use Port Forwarding to Access Applications in a Cluster.

b. Navigate to `localhost:3000` in your browser.

The Grafana sign-in page appears.

c. To sign in, enter `admin` for both the username and password.

Access Grafana using minikube

There are multiple ways to access the Grafana UI on a web browser when using minikube. For more information about minikube, refer to How to access applications running within minikube.

This section lists the two most common options for accessing an application running in minikube.

Option 1: Expose the service

This option uses the `type: LoadBalancer` in the `grafana.yaml` service manifest, which makes the service accessible through the `minikube service` command. For more information, refer to minikube Service command usage.

1. Run the following command to obtain the Grafana service IP:

bash Copy

```
minikube service grafana --namespace=my-grafana
```

The output returns the Kubernetes URL for service in your local cluster.

bash Copy

```
|-----------|---------|-------------|-----------------------------|
| NAMESPACE |  NAME   | TARGET PORT |             URL             |
|-----------|---------|-------------|-----------------------------|
| my-grafana | grafana |        3000 | http://192.168.122.144:32182 |
|-----------|---------|-------------|-----------------------------|
Opening service my-grafana/grafana in default browser...
http://192.168.122.144:32182
```

2. Run a `curl` command to verify whether a given connection should work in a browser under ideal circumstances.

bash Copy

```
curl 192.168.122.144:32182
```

The following example output shows that an endpoint has been located:

```
<a href="/login">Found</a>.
```

3. Access the Grafana UI in the browser using the provided IP:Port from the command above. For example `192.168.122.144:32182`

   The Grafana sign-in page appears.

4. To sign in to Grafana, enter `admin` for both the username and password.

Option 2: Use port forwarding

If Option 1 does not work in your minikube environment (this mostly depends on the network), then as an alternative you can use the port forwarding option for the Grafana service on port `3000`.

For more information about port forwarding, refer to Use Port Forwarding to Access Applications in a Cluster.

1. To find the minikube IP address, run the following command:

   bash Copy

   ```
   minikube ip
   ```

   The output contains the IP address that you use to access the Grafana Pod during port forwarding.

   A Pod is the smallest deployment unit in Kubernetes and is the core building block for running applications in a Kubernetes cluster. For more information about Pods, refer to Pods.

2. To obtain the Grafana Pod information, run the following command:

   bash Copy

   ```
   kubectl get pods --namespace=my-grafana
   ```

   The output should look similar to the following:

   bash Copy

   ```
   NAME                       READY   STATUS    RESTARTS   AGE
   grafana-58445b6986-dxrrw   1/1     Running   0          9m54s
   ```

   The output shows the Grafana POD name in the `NAME` column, that you use for port forwarding.

3. Run the following command for enabling the port forwarding on the POD:

```bash
kubectl port-forward pod/grafana-58445b6986-dxrrw --namespace=my-grafana --address 0.0.0.0 3
```

4. To access the Grafana UI on the web browser, type the minikube IP along with the forwarded port. For example `192.168.122.144:3000`

   The Grafana sign-in page appears.

5. To sign in to Grafana, enter `admin` for both the username and password.

Update an existing deployment using a rolling update strategy

Rolling updates enable deployment updates to take place with no downtime by incrementally updating Pods instances with new ones. The new Pods will be scheduled on nodes with available resources. For more information about rolling updates, refer to Performing a Rolling Update.

The following steps use the `kubectl annotate` command to add the metadata and keep track of the deployment. For more information about `kubectl annotate`, refer to kubectl annotate documentation.

> **NOTE**
>
> Instead of using the `annotate` flag, you can still use the `--record` flag. However, it has been deprecated and will be removed in the future version of Kubernetes. See: https://github.com/kubernetes/kubernetes/issues/40422

1. To view the current status of the rollout, run the following command:

```bash
kubectl rollout history deployment/grafana --namespace=my-grafana
```

   The output will look similar to this:

```bash
deployment.apps/grafana
REVISION  CHANGE-CAUSE
1         NONE
```

   The output shows that nothing has been updated or changed after applying the `grafana.yaml` file.

2. To add metadata to keep record of the initial deployment, run the following command:

```bash
kubectl annotate deployment/grafana kubernetes.io/change-cause='deployed the default base yar
```

3. To review the rollout history and verify the changes, run the following command:

```bash
kubectl rollout history deployment/grafana --namespace=my-grafana
```

You should see the updated information that you added in the `CHANGE-CAUSE` earlier.

Change Grafana image version

1. To change the deployed Grafana version, run the following `kubectl edit` command:

```bash
kubectl edit deployment grafana --namespace=my-grafana
```

2. In the editor, change the container image under the `kind: Deployment` section.

   For example:

   - From
     - `yaml image: grafana/grafana-oss:10.0.1`
   - To
     - `yaml image: grafana/grafana-oss-dev:10.1.0-124419pre`

3. Save the changes.

   Once you save the file, you receive a message similar to the following:

```bash
deployment.apps/grafana edited
```

This means that the changes have been applied.

4. To verify that the rollout on the cluster is successful, run the following command:

```bash
kubectl rollout status deployment grafana --namespace=my-grafana
```

A successful deployment rollout means that the Grafana Dev cluster is now available.

5. To check the statuses of all deployed objects, run the following command and include the `-o wide` flag to get more detailed output:

```bash
kubectl get all --namespace=my-grafana -o wide
```

   You should see the newly deployed `grafana-oss-dev` image.

6. To verify it, access the Grafana UI in the browser using the provided IP:Port from the command above.

   The Grafana sign-in page appears.

7. To sign in to Grafana, enter `admin` for both the username and password.

8. In the top-right corner, click the help icon.

   The version information appears.

9. Add the `change cause` metadata to keep track of things using the commands:

```bash
kubectl annotate deployment grafana --namespace=my-grafana kubernetes.io/change-cause='using
```

10. To verify, run the `kubectl rollout history` command:

```bash
kubectl rollout history deployment grafana --namespace=my-grafana
```

    You will see an output similar to this:

```bash
deployment.apps/grafana
REVISION  CHANGE-CAUSE
1         deploying the default yaml
2         using grafana-oss-dev:10.1.0-124419pre for testing
```

This means that `REVISION#2` is the current version.

NOTE

The last line of the `kubectl rollout history deployment` command output is the one which is currently active and running on your Kubernetes environment.

Roll back a deployment

When the Grafana deployment becomes unstable due to crash looping, bugs, and so on, you can roll back a deployment to an earlier version (a `REVISION`).

By default, Kubernetes deployment rollout history remains in the system so that you can roll back at any time. For more information, refer to Rolling Back to a Previous Revision.

1. To list all possible `REVISION` values, run the following command:

```bash
kubectl rollout history deployment grafana --namespace=my-grafana
```

2. To roll back to a previous version, run the `kubectl rollout undo` command and provide a revision number.

   Example: To roll back to a previous version, specify the `REVISION` number, which appears after you run the `kubectl rollout history deployment` command, in the `--to-revision` parameter.

```bash
kubectl rollout undo deployment grafana --to-revision=1 --namespace=my-grafana
```

3. To verify that the rollback on the cluster is successful, run the following command:

```bash
kubectl rollout status deployment grafana --namespace=my-grafana
```

4. Access the Grafana UI in the browser using the provided IP:Port from the command above.

   The Grafana sign-in page appears.

5. To sign in to Grafana, enter `admin` for both the username and password.
6. In the top-right corner, click the help icon to display the version number.
7. To see the new rollout history, run the following command:

```bash
kubectl rollout history deployment grafana --namespace=my-grafana
```

If you need to go back to any other `REVISION`, just repeat the steps above and use the correct revision number in the `--to-revision` parameter.

Troubleshooting

This section includes troubleshooting tips you might find helpful when deploying Grafana on Kubernetes.

Collecting logs

It is important to view the Grafana server logs while troubleshooting any issues.

1. To check the Grafana logs, run the following command:

```bash
# dump Pod logs for a Deployment (single-container case)
kubectl logs --namespace=my-grafana deploy/grafana
```

2. If you have multiple containers running in the deployment, run the following command to obtain the logs only for the Grafana deployment:

```bash
# dump Pod logs for a Deployment (multi-container case)
kubectl logs --namespace=my-grafana deploy/grafana -c grafana
```

For more information about accessing Kubernetes application logs, refer to Pods and Deployments.

Increasing log levels to debug mode

By default, the Grafana log level is set to `info`, but you can increase it to `debug` mode to fetch information needed to diagnose and troubleshoot a problem. For more information about Grafana log levels, refer to Configuring logs.

The following example uses the Kubernetes ConfigMap which is an API object that stores non-confidential data in key-value pairs. For more information, refer to Kubernetes ConfigMap Concept.

1. Create a empty file and name it `grafana.ini` and add the following:

```bash
[log]
; # Either "debug", "info", "warn", "error", "critical", default is "info"
; # we change from info to debug level
level = debug
```

This example adds the portion of the log section from the configuration file. You can refer to the Configure Grafana documentation to view all the default configuration settings.

2. To add the configuration file into the Kubernetes cluster via the ConfigMap object, run the following command:

```bash
kubectl create configmap ge-config --from-file=/path/to/file/grafana.ini --namespace=my-grafa
```

3. To verify the ConfigMap object creation, run the following command:

```bash
kubectl get configmap --namespace=my-grafana
```

4. Open the `grafana.yaml` file and In the Deployment section, provide the mount path to the custom configuration (`/etc/grafana`) and reference the newly created ConfigMap for it.

```bash
---
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: grafana
  name: grafana
# the rest of the code remains the same.
...
....
...
            requests:
            cpu: 250m
            memory: 750Mi
        volumeMounts:
          - mountPath: /var/lib/grafana
            name: grafana-pv
           # This is to mount the volume for the custom configuration
          - mountPath: /etc/grafana
```

Expand code

5. Deploy the manifest using the following kubectl apply command:

```bash
```

```
kubectl apply -f grafana.yaml --namespace=my-grafana
```

6. To verify the status, run the following commands:

```bash
# first check the rollout status
kubectl rollout status deployment grafana --namespace=my-grafana

# then check the deployment and configMap information
kubectl get all --namespace=my-grafana
```

7. To verify it, access the Grafana UI in the browser using the provided IP:Port

   The Grafana sign-in page appears.

8. To sign in to Grafana, enter `admin` for both the username and password.

9. Navigate to **Server Admin > Settings** and then search for log.

   You should see the level to debug mode.

Using the –dry-run command

You can use the Kubernetes `--dry-run` command to send requests to modifying endpoints and determine if the request would have succeeded.

Performing a dry run can be useful for catching errors or unintended consequences before they occur. For more information, refer to Kubernetes Dry-run.

Example:

The following example shows how to perform a dry run when you make changes to the `grafana.yaml` such as using a new image version, or adding new labels and you want to determine if there are syntax errors or conflicts.

To perform a dry run, run the following command:

```bash
kubectl apply -f grafana.yaml --dry-run=server --namespace=grafana
```

If there are no errors, then the output will look similar to this:

```bash
persistentvolumeclaim/grafana-pvc unchanged (server dry run)
deployment.apps/grafana unchanged (server dry run)
```

```
service/grafana unchanged (server dry run)
```

If there are errors or warnings, you will see them in the terminal.

Remove Grafana

If you want to remove any of the Grafana deployment objects, use the `kubectl delete command`.

1. If you want to remove the complete Grafana deployment, run the following command:

   ```bash
   kubectl delete -f grafana.yaml --namespace=my-grafana
   ```

   This command deletes the deployment, persistentvolumeclaim, and service objects.

2. To delete the ConfigMap, run the following command:

   ```bash
   kubectl delete configmap ge-config --namespace=my-grafana
   ```

Deploy Grafana Enterprise on Kubernetes

The process for deploying Grafana Enterprise is almost identical to the preceding process, except for additional steps that are required for adding your license file.

Obtain Grafana Enterprise license

To run Grafana Enterprise, you need a valid license. To obtain a license, contact a Grafana Labs representative. This topic assumes that you have a valid license in a `license.jwt` file. Associate your license with a URL that you can use later in the topic.

Create license secret

Create a Kubernetes secret from your license file using the following command:

```bash
kubectl create secret generic ge-license --from-file=/path/to/your/license.jwt
```

Create Grafana Enterprise configuration

1. Create a Grafana configuration file with the name `grafana.ini`

2. Paste the following YAML contents into the file you created:

```yaml
[enterprise]
license_path = /etc/grafana/license/license.jwt
[server]
root_url =/your/license/root/url
```

3. Update the `root_url` field to the url associated with the license provided to you.

Create Configmap for Grafana Enterprise configuration

Create a Kubernetes Configmap from your `grafana.ini` file with the following command:

```bash
kubectl create configmap ge-config --from-file=/path/to/your/grafana.ini
```

Create Grafana Enterprise Kubernetes manifest

1. Create a `grafana.yaml` file, and copy-and-paste the following content into it.

   The following YAML is identical to the one for a Grafana installation, except for the additional references to the Configmap that contains your Grafana configuration file and the secret that has your license.

```yaml
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: grafana-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
---
apiVersion: apps/v1
kind: Deployment
```

```
metadata:
  labels:
    app: grafana
  name: grafana
```

Expand code

CAUTION

If you use `LoadBalancer` in the Service and depending on your cloud platform and network configuration, doing so might expose your Grafana instance to the Internet. To eliminate this risk, use `ClusterIP` to restrict access from within the cluster Grafana is deployed to.

2. To send the manifest to Kubernetes API Server, run the following command: `kubectl apply -f grafana.yaml`
3. To verify the manifest was sent, run the following command: `kubectl port-forward service/grafana 3000:3000`
4. Navigate to `localhost:3000` in your browser.

   You should see the Grafana login page.
5. Use `admin` for both the username and password to login.
6. To verify you are working with an enterprise license, scroll to the bottom of the page where you should see `Enterprise (Licensed)`.

Build your first dashboard

This topic helps you get started with Grafana and build your first dashboard using the built-in `Grafana` data source. To learn more about Grafana, refer to Introduction to Grafana.

NOTE

Grafana also offers a free account with Grafana Cloud to help getting started even easier and faster. You can install Grafana to self-host or get a free Grafana Cloud account.

Install Grafana

Grafana can be installed on many different operating systems. For a list of the minimum hardware and software requirements, as well as instructions on installing Grafana, refer to Install Grafana.

Sign in to Grafana

To sign in to Grafana for the first time:

1. Open your web browser and go to http://localhost:3000/.

   The default HTTP port that Grafana listens to is `3000` unless you have configured a different port.

2. On the sign-in page, enter `admin` for both the username and password.
3. Click **Sign in**.

   If successful, you'll see a prompt to change the password.

4. Click **OK** on the prompt and change your password.

NOTE

We strongly recommend that you change the default administrator password.

Create a dashboard

If you've already set up a data source that you know how to query, refer to Create a dashboard instead.

To create your first dashboard using the built-in `-- Grafana --` data source:

1. Click **Dashboards** in the left-side menu.
2. On the Dashboards page, click **New** and select **New Dashboard** from the drop-down menu.
3. On the dashboard, click **+ Add visualization**.

4. In the dialog box that opens, click `-- Grafana --`:

   This configures your query and generates the Random Walk dashboard.
5. Click the Refresh dashboard icon to query the data source.

6. When you've finished editing your panel, click **Save** to save the dashboard.

   Alternatively, click **Apply** if you want to see your changes applied to the dashboard first. Then click the save icon in the dashboard header.
7. Add a descriptive title for the dashboard, and then click **Save**.

   Alternatively, Grafana can generate a dashboard title and summary for you using the OpenAI integration. Learn more in the Set up generative AI features for dashboards documentation.

Congratulations, you have created your first dashboard and it's displaying results.

Next

steps

Continue to experiment with what you have built, try the explore workflow or another visualization feature. Refer to Data sources for a list of supported data sources and instructions on how to add a data source. The following topics will be of interest to you:

- Panels and visualizations
- Dashboards
- Keyboard shortcuts
- Plugins

Admins

The following topics are of interest to Grafana server admin users:

- Grafana configuration
- Authentication
- User permissions and roles
- Provisioning
- Grafana CLI

Introduction

This section provides basic information about observability topics in general and Grafana in particular. These topics will help people who are just starting out with observability and monitoring.

Grafana Open Source

Grafana open source is open source visualization and analytics software. It allows you to query, visualize, alert on, and explore your metrics, logs, and traces no matter where they are stored. It provides you with tools to turn your time-series database (TSDB) data into insightful graphs and visualizations.

Grafana Loki

Grafana Loki is an open source, set of components that can be composed into a fully featured logging stack. For more information, refer to Loki documentation.

Grafana Tempo

Grafana Tempo is an open source, easy-to-use and high-volume distributed tracing backend. For more information, refer to Tempo documentation.

Grafana Mimir

Grafana Mimir is an open source software project that provides a scalable long-term storage for Prometheus. For more information about Grafana Mimir, refer to Grafana Mimir documentation.

Grafana Oncall

Grafana OnCall is an open source incident response management tool built to help teams improve their collaboration and resolve incidents faster. For more information about Grafana OnCall, refer to Grafana OnCall documentation.

Grafana Cloud

Grafana Cloud is a highly available, fast, fully managed OpenSaaS logging and metrics platform. It is everything you love about Grafana, but Grafana Labs hosts it for you and handles all the headaches.

Grafana Enterprise

Grafana Enterprise is the commercial edition of Grafana that includes additional features not found in the open source version. Building on everything you already know and love about Grafana, Grafana Enterprise adds enterprise data sources, advanced authentication options, more permission controls, 24×7×365 support, and training from the core Grafana team.

Learn more about Grafana Enterprise and contact the Grafana Labs Sales Team to purchase an Enterprise license. You can also obtain a trial license before the purchase.

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Configure Grafana

You can use Grafana Cloud to avoid installing, maintaining, and scaling your own instance of Grafana. Create a free account to get started, which includes free forever access to 10k metrics, 50GB logs, 50GB traces, 500VUh k6 testing & more.

Grafana has default and custom configuration files. You can customize your Grafana instance by modifying the custom configuration file or by using environment variables. To see the list of settings for a Grafana instance, refer to View server settings.

NOTE

After you add custom options, uncomment the relevant sections of the configuration file. Restart Grafana for your changes to take effect.

Configuration file location

The default settings for a Grafana instance are stored in the `$WORKING_DIR/conf/defaults.ini` file. *Do not* change this file.

Depending on your OS, your custom configuration file is either the `$WORKING_DIR/conf/custom.ini` file or the `/usr/local/etc/grafana/grafana.ini` file. The custom configuration file path can be overridden using the `--config` parameter.

Linux

If you installed Grafana using the `deb` or `rpm` packages, then your configuration file is located at `/etc/grafana/grafana.ini` and a separate `custom.ini` is not used. This path is specified in the Grafana

init.d script using `--config` file parameter.

Docker

Refer to Configure a Grafana Docker image for information about environmental variables, persistent storage, and building custom Docker images.

Windows

On Windows, the `sample.ini` file is located in the same directory as `defaults.ini` file. It contains all the settings commented out. Copy `sample.ini` and name it `custom.ini`.

macOS

By default, the configuration file is located at `/usr/local/etc/grafana/grafana.ini`. For a Grafana instance installed using Homebrew, edit the `grafana.ini` file directly. Otherwise, add a configuration file named `custom.ini` to the `conf` folder to override the settings defined in `conf/defaults.ini`.

Remove comments in the .ini files

Grafana uses semicolons (the `;` char) to comment out lines in a `.ini` file. You must uncomment each line in the `custom.ini` or the `grafana.ini` file that you are modify by removing `;` from the beginning of that line. Otherwise your changes will be ignored.

For example:

```
# The HTTP port  to use
;http_port = 3000
```

Copy

Override configuration with environment variables

Do not use environment variables to *add* new configuration settings. Instead, use environmental variables to *override* existing options.

To override an option:

bash

Copy

```
GF_<SectionName>_<KeyName>
```

Where the section name is the text within the brackets. Everything should be uppercase, `.` and `-` should be replaced by `_`. For example, if you have these configuration settings:

bash

Copy

```
# default section
instance_name = ${HOSTNAME}

[security]
admin_user = admin

[auth.google]
client_secret = 0ldS3cretKey

[plugin.grafana-image-renderer]
rendering_ignore_https_errors = true

[feature_toggles]
enable = newNavigation
```

You can override variables on Linux machines with:

bash Copy

```
export GF_DEFAULT_INSTANCE_NAME=my-instance
export GF_SECURITY_ADMIN_USER=owner
export GF_AUTH_GOOGLE_CLIENT_SECRET=newS3cretKey
export GF_PLUGIN_GRAFANA_IMAGE_RENDERER_RENDERING_IGNORE_HTTPS_ERRORS=true
export GF_FEATURE_TOGGLES_ENABLE=newNavigation
```

Variable expansion

> **NOTE**
>
> Only available in Grafana 7.1+.

If any of your options contains the expression `$__<provider>{<argument>}` or `${<environment variable>}`, then they will be processed by Grafana's variable expander. The expander runs the provider with the provided argument to get the final value of the option.

There are three providers: `env`, `file`, and `vault`.

Env provider

The `env` provider can be used to expand an environment variable. If you set an option to `$__env{PORT}` the `PORT` environment variable will be used in its place. For environment variables you

can also use the short-hand syntax `${PORT}`. Grafana's log directory would be set to the `grafana` directory in the directory behind the `LOGDIR` environment variable in the following example.

```ini
[paths]
logs = $__env{LOGDIR}/grafana
```

File provider

`file` reads a file from the filesystem. It trims whitespace from the beginning and the end of files. The database password in the following example would be replaced by the content of the `/etc/secrets/gf_sql_password` file:

```ini
[database]
password = $__file{/etc/secrets/gf_sql_password}
```

Vault provider

The `vault` provider allows you to manage your secrets with Hashicorp Vault.

> Vault provider is only available in Grafana Enterprise v7.1+. For more information, refer to Vault integration in Grafana Enterprise.

---

app_mode

Options are `production` and `development`. Default is `production`. *Do not* change this option unless you are working on Grafana development.

instance_name

Set the name of the grafana-server instance. Used in logging, internal metrics, and clustering info. Defaults to: `${HOSTNAME}`, which will be replaced with environment variable `HOSTNAME`, if that is empty or does not exist Grafana will try to use system calls to get the machine name.

force_migration

NOTE

This option is deprecated - See `clean_upgrade` option instead.

When you restart Grafana to rollback from Grafana Alerting to legacy alerting, delete any existing Grafana Alerting data, such as alert rules, contact points, and notification policies. Default is `false`.

If `false` or unset, existing Grafana Alerting data is not changed or deleted when rolling back to legacy alerting.

NOTE

It should be kept false or unset when not needed, as it may cause unintended data loss if left enabled.

[paths]

data

Path to where Grafana stores the sqlite3 database (if used), file-based sessions (if used), and other data. This path is usually specified via command line in the init.d script or the systemd service file.

macOS: The default SQLite database is located at `/usr/local/var/lib/grafana`

temp_data_lifetime

How long temporary images in `data` directory should be kept. Defaults to: `24h`. Supported modifiers: `h` (hours), `m` (minutes), for example: `168h`, `30m`, `10h30m`. Use `0` to never clean up temporary files.

logs

Path to where Grafana stores logs. This path is usually specified via command line in the init.d script or the systemd service file. You can override it in the configuration file or in the default environment variable file. However, please note that by overriding this the default log path will be used temporarily until Grafana has fully initialized/started.

Override log path using the command line argument `cfg:default.paths.logs`:

bash Copy

```
./grafana-server --config /custom/config.ini --homepath /custom/homepath cfg:default.paths.logs=/c
```

macOS: By default, the log file should be located at `/usr/local/var/log/grafana/grafana.log`.

plugins

Directory where Grafana automatically scans and looks for plugins. For information about manually or automatically installing plugins, refer to Install Grafana plugins.

macOS: By default, the Mac plugin location is: `/usr/local/var/lib/grafana/plugins`.

provisioning

Folder that contains provisioning config files that Grafana will apply on startup. Dashboards will be reloaded when the json files changes.

[server]

protocol

`http`, `https`, `h2` or `socket`

min_tls_version

The TLS Handshake requires a minimum TLS version. The available options are TLS1.2 and TLS1.3. If you do not specify a version, the system uses TLS1.2.

http_addr

The host for the server to listen on. If your machine has more than one network interface, you can use this setting to expose the Grafana service on only one network interface and not have it available on others, such as the loopback interface. An empty value is equivalent to setting the value to `0.0.0.0`, which means the Grafana service binds to all interfaces.

In environments where network address translation (NAT) is used, ensure you use the network interface address and not a final public address; otherwise, you might see errors such as `bind: cannot assign requested address` in the logs.

http_port

The port to bind to, defaults to `3000`. To use port 80 you need to either give the Grafana binary permission for example:

bash Copy

```
$ sudo setcap 'cap_net_bind_service=+ep' /usr/sbin/grafana-server
```

Or redirect port 80 to the Grafana port using:

bash Copy

```
$ sudo iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-port 3000
```

Another way is to put a web server like Nginx or Apache in front of Grafana and have them proxy requests to Grafana.

domain

This setting is only used in as a part of the `root_url` setting (see below). Important if you use GitHub or Google OAuth.

enforce_domain

Redirect to correct domain if the host header does not match the domain. Prevents DNS rebinding attacks. Default is `false`.

root_url

This is the full URL used to access Grafana from a web browser. This is important if you use Google or GitHub OAuth authentication (for the callback URL to be correct).

> **NOTE**
>
> This setting is also important if you have a reverse proxy in front of Grafana that exposes it through a subpath. In that case add the subpath to the end of this URL setting.

serve_from_sub_path

Serve Grafana from subpath specified in `root_url` setting. By default it is set to `false` for compatibility reasons.

By enabling this setting and using a subpath in `root_url` above, e.g. `root_url = http://localhost:3000/grafana`, Grafana is accessible on `http://localhost:3000/grafana`. If accessed without subpath Grafana will redirect to an URL with the subpath.

router_logging

Set to `true` for Grafana to log all HTTP requests (not just errors). These are logged as Info level events to the Grafana log.

static_root_path

The path to the directory where the front end files (HTML, JS, and CSS files). Defaults to `public` which is why the Grafana binary needs to be executed with working directory set to the installation path.

enable_gzip

Set this option to `true` to enable HTTP compression, this can improve transfer speed and bandwidth utilization. It is recommended that most users set it to `true`. By default it is set to `false` for compatibility reasons.

cert_file

Path to the certificate file (if `protocol` is set to `https` or `h2`).

cert_key

Path to the certificate key file (if `protocol` is set to `https` or `h2`).

socket_gid

GID where the socket should be set when `protocol=socket`. Make sure that the target group is in the group of Grafana process and that Grafana process is the file owner before you change this setting. It is recommended to set the gid as http server user gid. Not set when the value is -1.

socket_mode

Mode where the socket should be set when `protocol=socket`. Make sure that Grafana process is the file owner before you change this setting.

socket

Path where the socket should be created when `protocol=socket`. Make sure Grafana has appropriate permissions for that path before you change this setting.

cdn_url

> **NOTE**
>
> Available in Grafana v7.4 and later versions.

Specify a full HTTP URL address to the root of your Grafana CDN assets. Grafana will add edition and version paths.

For example, given a cdn url like `https://cdn.myserver.com` grafana will try to load a javascript file from `http://cdn.myserver.com/grafana-oss/7.4.0/public/build/app.<hash>.js`.

read_timeout

Sets the maximum time using a duration format (5s/5m/5ms) before timing out read of an incoming request and closing idle connections. `0` means there is no timeout for reading the request.

[server.custom_response_headers]

This setting enables you to specify additional headers that the server adds to HTTP(S) responses.

```
exampleHeader1 = exampleValue1
exampleHeader2 = exampleValue2
```

Copy

[database]

Grafana needs a database to store users and dashboards (and other things). By default it is configured to use `sqlite3` which is an embedded database (included in the main Grafana binary).

type

Either `mysql`, `postgres` or `sqlite3`, it's your choice.

host

Only applicable to MySQL or Postgres. Includes IP or hostname and port or in case of Unix sockets the path to it. For example, for MySQL running on the same host as Grafana: `host = 127.0.0.1:3306` or with Unix sockets: `host = /var/run/mysqld/mysqld.sock`

name

The name of the Grafana database. Leave it set to `grafana` or some other name.

user

The database user (not applicable for `sqlite3`).

password

The database user's password (not applicable for `sqlite3`). If the password contains `#` or `;` you have to wrap it with triple quotes. For example `"""#password;"""`

url

Use either URL or the other fields below to configure the database Example: `mysql://user:secret@host:port/database`

max_idle_conn

The maximum number of connections in the idle connection pool.

max_open_conn

The maximum number of open connections to the database. For MYSQL, configure this setting on both Grafana and the database. For more information, refer to `sysvar max connections`.

conn_max_lifetime

Sets the maximum amount of time a connection may be reused. The default is 14400 (which means 14400 seconds or 4 hours). For MySQL, this setting should be shorter than the `wait timeout` variable.

locking_attempt_timeout_sec

For "mysql", if the `migrationLocking` feature toggle is set, specify the time (in seconds) to wait before failing to lock the database for the migrations. Default is 0.

log_queries

Set to `true` to log the sql calls and execution times.

ssl_mode

For Postgres, use use any valid libpq `sslmode`, e.g. `disable`, `require`, `verify-full`, etc. For MySQL, use either `true`, `false`, or `skip-verify`.

ssl_sni

For Postgres, set to `0` to disable Server Name Indication. This is enabled by default on SSL-enabled connections.

isolation_level

Only the MySQL driver supports isolation levels in Grafana. In case the value is empty, the driver's default isolation level is applied. Available options are "READ-UNCOMMITTED", "READ-COMMITTED", "REPEATABLE-READ" or "SERIALIZABLE".

ca_cert_path

The path to the CA certificate to use. On many Linux systems, certs can be found in `/etc/ssl/certs`.

client_key_path

The path to the client key. Only if server requires client authentication.

client_cert_path

The path to the client cert. Only if server requires client authentication.

server_cert_name

The common name field of the certificate used by the `mysql` or `postgres` server. Not necessary if `ssl_mode` is set to `skip-verify`.

path

Only applicable for `sqlite3` database. The file path where the database will be stored.

cache_mode

For “sqlite3” only. Shared cache setting used for connecting to the database. (private, shared) Defaults to `private`.

wal

For “sqlite3” only. Setting to enable/disable Write-Ahead Logging. The default value is `false` (disabled).

query_retries

This setting applies to `sqlite` only and controls the number of times the system retries a query when the database is locked. The default value is `0` (disabled).

transaction_retries

This setting applies to `sqlite` only and controls the number of times the system retries a transaction when the database is locked. The default value is `5`.

instrument_queries

Set to `true` to add metrics and tracing for database queries. The default value is `false`.

[remote_cache]

Caches authentication details and session information in the configured database, Redis or Memcached. This setting does not configure Query Caching in Grafana Enterprise.

type

Either `redis`, `memcached`, or `database`. Defaults to `database`

connstr

The remote cache connection string. The format depends on the `type` of the remote cache. Options are `database`, `redis`, and `memcache`.

database

Leave empty when using `database` since it will use the primary database.

redis

Example connstr: `addr=127.0.0.1:6379,pool_size=100,db=0,ssl=false`

- `addr` is the host `:` port of the redis server.
- `pool_size` (optional) is the number of underlying connections that can be made to redis.
- `db` (optional) is the number identifier of the redis database you want to use.
- `ssl` (optional) is if SSL should be used to connect to redis server. The value may be `true`, `false`, or `insecure`. Setting the value to `insecure` skips verification of the certificate chain and hostname when making the connection.

memcache

Example connstr: `127.0.0.1:11211`

[dataproxy]

logging

This enables data proxy logging, default is `false`.

timeout

How long the data proxy should wait before timing out. Default is 30 seconds.

This setting also applies to core backend HTTP data sources where query requests use an HTTP client with timeout set.

keep_alive_seconds

Interval between keep-alive probes. Default is `30` seconds. For more details check the Dialer.KeepAlive documentation.

tls_handshake_timeout_seconds

The length of time that Grafana will wait for a successful TLS handshake with the datasource. Default is `10` seconds. For more details check the Transport.TLSHandshakeTimeout documentation.

expect_continue_timeout_seconds

The length of time that Grafana will wait for a datasource's first response headers after fully writing the request headers, if the request has an "Expect: 100-continue" header. A value of `0` will result in the body being sent immediately. Default is `1` second. For more details check the Transport.ExpectContinueTimeout documentation.

max_conns_per_host

Optionally limits the total number of connections per host, including connections in the dialing, active, and idle states. On limit violation, dials are blocked. A value of `0` means that there are no limits. Default is `0`. For more details check the Transport.MaxConnsPerHost documentation.

max_idle_connections

The maximum number of idle connections that Grafana will maintain. Default is `100`. For more details check the Transport.MaxIdleConns documentation.

idle_conn_timeout_seconds

The length of time that Grafana maintains idle connections before closing them. Default is `90` seconds. For more details check the Transport.IdleConnTimeout documentation.

send_user_header

If enabled and user is not anonymous, data proxy will add X-Grafana-User header with username into the request. Default is `false`.

response_limit

Limits the amount of bytes that will be read/accepted from responses of outgoing HTTP requests. Default is `0` which means disabled.

row_limit

Limits the number of rows that Grafana will process from SQL (relational) data sources. Default is `1000000`.

user_agent

Sets a custom value for the `User-Agent` header for outgoing data proxy requests. If empty, the default value is `Grafana/<BuildVersion>` (for example `Grafana/9.0.0`).

[analytics]

enabled

This option is also known as *usage analytics*. When `false`, this option disables the writers that write to the Grafana database and the associated features, such as dashboard and data source insights, presence indicators, and advanced dashboard search. The default value is `true`.

reporting_enabled

When enabled Grafana will send anonymous usage statistics to `stats.grafana.org`. No IP addresses are being tracked, only simple counters to track running instances, versions, dashboard and error counts. It is very helpful to us, so please leave this enabled. Counters are sent every 24 hours. Default value is `true`.

check_for_updates

Set to false, disables checking for new versions of Grafana from Grafana's GitHub repository. When enabled, the check for a new version runs every 10 minutes. It will notify, via the UI, when a new version is available. The check itself will not prompt any auto-updates of the Grafana software, nor will it send any sensitive information.

check_for_plugin_updates

> **NOTE**
>
> Available in Grafana v8.5.0 and later versions.

Set to false disables checking for new versions of installed plugins from https://grafana.com. When enabled, the check for a new plugin runs every 10 minutes. It will notify, via the UI, when a new plugin update exists. The check itself will not prompt any auto-updates of the plugin, nor will it send any sensitive information.

google_analytics_ua_id

If you want to track Grafana usage via Google analytics specify *your* Universal Analytics ID here. By default this feature is disabled.

google_analytics_4_id

If you want to track Grafana usage via Google Analytics 4 specify *your* GA4 ID here. By default this feature is disabled.

google_tag_manager_id

Google Tag Manager ID, only enabled if you enter an ID here.

rudderstack_write_key

If you want to track Grafana usage via Rudderstack specify *your* Rudderstack Write Key here. The `rudderstack_data_plane_url` must also be provided for this feature to be enabled. By default this feature is disabled.

rudderstack_data_plane_url

Rudderstack data plane url that will receive Rudderstack events. The `rudderstack_write_key` must also be provided for this feature to be enabled.

rudderstack_sdk_url

Optional. If tracking with Rudderstack is enabled, you can provide a custom URL to load the Rudderstack SDK.

rudderstack_config_url

Optional. If tracking with Rudderstack is enabled, you can provide a custom URL to load the Rudderstack config.

rudderstack_integrations_url

Optional. If tracking with Rudderstack is enabled, you can provide a custom URL to load the SDK for destinations running in device mode. This setting is only valid for Rudderstack version 1.1 and higher.

application_insights_connection_string

If you want to track Grafana usage via Azure Application Insights, then specify *your* Application Insights connection string. Since the connection string contains semicolons, you need to wrap it in backticks (`). By default, tracking usage is disabled.

application_insights_endpoint_url

Optionally, use this option to override the default endpoint address for Application Insights data collecting. For details, refer to the Azure documentation.

---

feedback_links_enabled

Set to `false` to remove all feedback links from the UI. Default is `true`.

[security]

disable_initial_admin_creation

> Only available in Grafana v6.5+.

Disable creation of admin user on first start of Grafana. Default is `false`.

admin_user

The name of the default Grafana Admin user, who has full permissions. Default is `admin`.

admin_password

The password of the default Grafana Admin. Set once on first-run. Default is `admin`.

admin_email

The email of the default Grafana Admin, created on startup. Default is `admin@localhost`.

secret_key

Used for signing some data source settings like secrets and passwords, the encryption format used is AES-256 in CFB mode. Cannot be changed without requiring an update to data source settings to re-encode them.

disable_gravatar

Set to `true` to disable the use of Gravatar for user profile images. Default is `false`.

data_source_proxy_whitelist

Define a whitelist of allowed IP addresses or domains, with ports, to be used in data source URLs with the Grafana data source proxy. Format: `ip_or_domain:port` separated by spaces. PostgreSQL, MySQL, and MSSQL data sources do not use the proxy and are therefore unaffected by this setting.

disable_brute_force_login_protection

Set to `true` to disable brute force login protection. Default is `false`. An existing user's account will be locked after 5 attempts in 5 minutes.

cookie_secure

Set to `true` if you host Grafana behind HTTPS. Default is `false`.

cookie_samesite

Sets the `SameSite` cookie attribute and prevents the browser from sending this cookie along with cross-site requests. The main goal is to mitigate the risk of cross-origin information leakage. This setting also provides some protection against cross-site request forgery attacks (CSRF), read more about SameSite here. Valid values are `lax`, `strict`, `none`, and `disabled`. Default is `lax`. Using value `disabled` does not add any `SameSite` attribute to cookies.

allow_embedding

When `false`, the HTTP header `X-Frame-Options: deny` will be set in Grafana HTTP responses which will instruct browsers to not allow rendering Grafana in a `<frame>`, `<iframe>`, `<embed>` or `<object>`. The main goal is to mitigate the risk of Clickjacking. Default is `false`.

strict_transport_security

Set to `true` if you want to enable HTTP `Strict-Transport-Security` (HSTS) response header. Only use this when HTTPS is enabled in your configuration, or when there is another upstream system that ensures your application does HTTPS (like a frontend load balancer). HSTS tells browsers that the site should only be accessed using HTTPS.

strict_transport_security_max_age_seconds

Sets how long a browser should cache HSTS in seconds. Only applied if strict_transport_security is enabled. The default value is `86400`.

strict_transport_security_preload

Set to `true` to enable HSTS `preloading` option. Only applied if strict_transport_security is enabled. The default value is `false`.

strict_transport_security_subdomains

Set to `true` to enable the HSTS includeSubDomains option. Only applied if strict_transport_security is enabled. The default value is `false`.

x_content_type_options

Set to `false` to disable the X-Content-Type-Options response header. The X-Content-Type-Options response HTTP header is a marker used by the server to indicate that the MIME types advertised in the Content-Type headers should not be changed and be followed. The default value is `true`.

x_xss_protection

Set to `false` to disable the X-XSS-Protection header, which tells browsers to stop pages from loading when they detect reflected cross-site scripting (XSS) attacks. The default value is `true`.

content_security_policy

Set to `true` to add the Content-Security-Policy header to your requests. CSP allows to control resources that the user agent can load and helps prevent XSS attacks.

content_security_policy_template

Set the policy template that will be used when adding the `Content-Security-Policy` header to your requests. `$NONCE` in the template includes a random nonce.

content_security_policy_report_only

Set to `true` to add the `Content-Security-Policy-Report-Only` header to your requests. CSP in Report Only mode enables you to experiment with policies by monitoring their effects without enforcing them. You can enable both policies simultaneously.

content_security_policy_template

Set the policy template that will be used when adding the `Content-Security-Policy-Report-Only` header to your requests. `$NONCE` in the template includes a random nonce.

---

angular_support_enabled

This currently defaults to `true` but will default to `false` in a future release. When set to false the angular framework and support components will not be loaded. This means that all plugins and core features that depend on angular support will stop working.

The core features that depend on angular are:

- Old graph panel
- Old table panel
- Legacy alerting edit rule UI

These features each have supported alternatives, and we recommend using them.

csrf_trusted_origins

List of additional allowed URLs to pass by the CSRF check. Suggested when authentication comes from an IdP.

csrf_additional_headers

List of allowed headers to be set by the user. Suggested to use for if authentication lives behind reverse proxies.

csrf_always_check

Set to `true` to execute the CSRF check even if the login cookie is not in a request (default `false`).

disable_frontend_sandbox_for_plugins

Comma-separated list of plugins ids that won't be loaded inside the frontend sandbox. It is recommended to only use this option for plugins that are known to have problems running inside the frontend sandbox.

[snapshots]

enabled

Set to `false` to disable the snapshot feature (default `true`).

external_enabled

Set to `false` to disable external snapshot publish endpoint (default `true`).

external_snapshot_url

Set root URL to a Grafana instance where you want to publish external snapshots (defaults to https://snapshots.raintank.io).

external_snapshot_name

Set name for external snapshot button. Defaults to `Publish to snapshots.raintank.io`.

public_mode

Set to true to enable this Grafana instance to act as an external snapshot server and allow unauthenticated requests for creating and deleting snapshots. Default is `false`.

snapshot_remove_expired

Enable this to automatically remove expired snapshots. Default is `true`.

[dashboards]

versions_to_keep

Number dashboard versions to keep (per dashboard). Default: `20`, Minimum: `1`.

min_refresh_interval

> Only available in Grafana v6.7+.

This feature prevents users from setting the dashboard refresh interval to a lower value than a given interval value. The default interval value is 5 seconds. The interval string is a possibly signed sequence of decimal numbers, followed by a unit suffix (ms, s, m, h, d), e.g. `30s` or `1m`.

As of Grafana v7.3, this also limits the refresh interval options in Explore.

default_home_dashboard_path

Path to the default home dashboard. If this value is empty, then Grafana uses StaticRootPath + “dashboards/home.json”.

> **NOTE**
>
> On Linux, Grafana uses `/usr/share/grafana/public/dashboards/home.json` as the default home dashboard location.

[sql_datasources]

max_open_conns_default

For SQL data sources (MySql, Postgres, MSSQL) you can override the default maximum number of open connections (default: 100). The value configured in data source settings will be preferred over the default value.

max_idle_conns_default

For SQL data sources (MySql, Postgres, MSSQL) you can override the default allowed number of idle connections (default: 100). The value configured in data source settings will be preferred over the default value.

max_conn_lifetime_default

For SQL data sources (MySql, Postgres, MSSQL) you can override the default maximum connection lifetime specified in seconds (default: 14400). The value configured in data source settings will be preferred over the default value.

[users]

allow_sign_up

Set to `false` to prohibit users from being able to sign up / create user accounts. Default is `false`. The admin user can still create users. For more information about creating a user, refer to Add a user.

allow_org_create

Set to `false` to prohibit users from creating new organizations. Default is `false`.

auto_assign_org

Set to `true` to automatically add new users to the main organization (id 1). When set to `false`, new users automatically cause a new organization to be created for that new user. The organization will be created even if the `allow_org_create` setting is set to `false`. Default is `true`.

auto_assign_org_id

Set this value to automatically add new users to the provided org. This requires `auto_assign_org` to be set to `true`. Please make sure that this organization already exists. Default is 1.

auto_assign_org_role

The `auto_assign_org_role` setting determines the default role assigned to new users in the main organization (if `auto_assign_org` setting is set to true). The available options are `Viewer` (default), `Admin`, `Editor`, and `None`. For example:

```
auto_assign_org_role = Viewer
```

verify_email_enabled

Require email validation before sign up completes. Default is `false`.

login_hint

Text used as placeholder text on login page for login/username input.

password_hint

Text used as placeholder text on login page for password input.

default_theme

Sets the default UI theme: `dark`, `light`, or `system`. The default theme is `dark`.

`system` matches the user's system theme.

default_language

This option will set the default UI language if a supported IETF language tag like `en-US` is available. If set to `detect`, the default UI language will be determined by browser preference. The default is `en-US`.

home_page

Path to a custom home page. Users are only redirected to this if the default home dashboard is used. It should match a frontend route and contain a leading slash.

External user management

If you manage users externally you can replace the user invite button for organizations with a link to an external site together with a description.

viewers_can_edit

Viewers can access and use Explore and perform temporary edits on panels in dashboards they have access to. They cannot save their changes. Default is `false`.

editors_can_admin

Editors can administrate dashboards, folders and teams they create. Default is `false`.

user_invite_max_lifetime_duration

The duration in time a user invitation remains valid before expiring. This setting should be expressed as a duration. Examples: 6h (hours), 2d (days), 1w (week). Default is `24h` (24 hours). The minimum supported duration is `15m` (15 minutes).

hidden_users

This is a comma-separated list of usernames. Users specified here are hidden in the Grafana UI. They are still visible to Grafana administrators and to themselves.

[auth]

Grafana provides many ways to authenticate users. Refer to the Grafana Authentication overview and other authentication documentation for detailed instructions on how to set up and configure authentication.

login_cookie_name

The cookie name for storing the auth token. Default is `grafana_session`.

login_maximum_inactive_lifetime_duration

The maximum lifetime (duration) an authenticated user can be inactive before being required to login at next visit. Default is 7 days (7d). This setting should be expressed as a duration, e.g. 5m (minutes), 6h (hours), 10d (days), 2w (weeks), 1M (month). The lifetime resets at each successful token rotation (token_rotation_interval_minutes).

login_maximum_lifetime_duration

The maximum lifetime (duration) an authenticated user can be logged in since login time before being required to login. Default is 30 days (30d). This setting should be expressed as a duration, e.g. 5m (minutes), 6h (hours), 10d (days), 2w (weeks), 1M (month).

token_rotation_interval_minutes

How often auth tokens are rotated for authenticated users when the user is active. The default is each 10 minutes.

disable_login_form

Set to true to disable (hide) the login form, useful if you use OAuth. Default is false.

disable_signout_menu

Set to `true` to disable the signout link in the side menu. This is useful if you use auth.proxy. Default is `false`.

signout_redirect_url

The URL the user is redirected to upon signing out. To support OpenID Connect RP-Initiated Logout, the user must add `post_logout_redirect_uri` to the `signout_redirect_url`.

Example:

signout_redirect_url = http://localhost:8087/realms/grafana/protocol/openid-connect/logout?post_logout_redirect_uri=http%3A%2F%2Flocalhost%3A3000%2Flogin

oauth_auto_login

> **NOTE**
>
> This option is deprecated - use `auto_login` option for specific OAuth provider instead.

Set to `true` to attempt login with OAuth automatically, skipping the login screen. This setting is ignored if multiple OAuth providers are configured. Default is `false`.

oauth_state_cookie_max_age

How many seconds the OAuth state cookie lives before being deleted. Default is `600` (seconds) Administrators can increase this if they experience OAuth login state mismatch errors.

oauth_skip_org_role_update_sync

> **NOTE**
>
> This option is deprecated in favor of OAuth provider specific `skip_org_role_sync` settings. The following sections explain settings for each provider.

If you want to change the `oauth_skip_org_role_update_sync` setting to `false`, then for each provider you have set up, use the `skip_org_role_sync` setting to specify whether you want to skip the synchronization.

> **WARNING**
>
> Currently if no organization role mapping is found for a user, Grafana doesn't update the user's organization role. With Grafana 10, if `oauth_skip_org_role_update_sync` option is set to `false`, users with no mapping will be reset to the default organization role on every login. See `auto_assign_org_role` option.

skip_org_role_sync

`skip_org_role_sync` prevents the synchronization of organization roles for a specific OAuth integration, while the deprecated setting `oauth_skip_org_role_update_sync` affects all configured OAuth providers.

The default value for `skip_org_role_sync` is `false`.

With `skip_org_role_sync` set to `false`, the users' organization and role is reset on every new login, based on the external provider's role. See your provider in the tables below.

With `skip_org_role_sync` set to `true`, when a user logs in for the first time, Grafana sets the organization role based on the value specified in `auto_assign_org_role` and forces the organization to `auto_assign_org_id` when specified, otherwise it falls back to OrgID `1`.

> **Note**: Enabling `skip_org_role_sync` also disables the synchronization of Grafana Admins from the external provider, as such `allow_assign_grafana_admin` is ignored.

Use this setting when you want to manage the organization roles of your users from within Grafana and be able to manually assign them to multiple organizations, or to prevent synchronization conflicts when they can be synchronized from another provider.

The behavior of `oauth_skip_org_role_update_sync` and `skip_org_role_sync`, can be seen in the tables below:

[auth.grafana_com]

`oauth_skip_org_role_update_sync`	`skip_org_role_sync`	Resulting Org Role	Modifiable
false	false	Synchronize user organization role with Grafana.com role. If no role is provided, `auto_assign_org_role` is set.	false
true	false	Skips organization role synchronization for all OAuth providers' users. Role is set to `auto_assign_org_role`.	true
false	true	Skips organization role synchronization for Grafana.com users. Role is set to `auto_assign_org_role`.	true
true	true	Skips organization role synchronization for Grafana.com users and all other OAuth providers. Role is set to `auto_assign_org_role`.	true

[auth.azuread]

`oauth_skip_org_role_update_sync`	`skip_org_role_sync`	Resulting Org Role	Modifiable
false	false	Synchronize user organization role with AzureAD role. If no role is provided, `auto_assign_org_role` is set.	false
true	false	Skips organization role synchronization for all OAuth providers' users. Role is set to `auto_assign_org_role`.	true
false	true	Skips organization role synchronization for AzureAD users. Role is set to `auto_assign_org_role`.	true
true	true	Skips organization role synchronization for AzureAD users and all other OAuth providers. Role is set to `auto_assign_org_role`.	true

[auth.google]

oauth_skip_org_role_update_sync	skip_org_role_sync	Resulting Org Role	Modifiable
false	false	User organization role is set to `auto_assign_org_role` and cannot be changed.	false
true	false	User organization role is set to `auto_assign_org_role` and can be changed in Grafana.	true
false	true	User organization role is set to `auto_assign_org_role` and can be changed in Grafana.	true
true	true	User organization role is set to `auto_assign_org_role` and can be changed in Grafana.	true

NOTE

For GitLab, GitHub, Okta, Generic OAuth providers, Grafana synchronizes organization roles and sets Grafana Admins. The `allow_assign_grafana_admin` setting is also accounted for, to allow or not setting the Grafana Admin role from the external provider.

[auth.github]

oauth_skip_org_role_update_sync	skip_org_role_sync	Resulting Org Role	Modifiable
false	false	Synchronize user organization role with GitHub role. If no role is provided, `auto_assign_org_role` is set.	false
true	false	Skips organization role synchronization for all OAuth providers' users. Role is set to `auto_assign_org_role`.	true
false	true	Skips organization role and Grafana Admin synchronization for GitHub users. Role is set to `auto_assign_org_role`.	true
true	true	Skips organization role synchronization for all OAuth providers and skips Grafana Admin synchronization for GitHub users. Role is set to `auto_assign_org_role`.	true

[auth.gitlab]

`oauth_skip_org_role_update_sync`	`skip_org_role_sync`	Resulting Org Role	Modifiable
false	false	Synchronize user organization role with Gitlab role. If no role is provided, `auto_assign_org_role` is set.	false
true	false	Skips organization role synchronization for all OAuth providers' users. Role is set to `auto_assign_org_role`.	true
false	true	Skips organization role and Grafana Admin synchronization for Gitlab users. Role is set to `auto_assign_org_role`.	true
true	true	Skips organization role synchronization for all OAuth providers and skips Grafana Admin synchronization for Gitlab users. Role is set to `auto_assign_org_role`.	true

[auth.generic_oauth]

`oauth_skip_org_role_update_sync`	`skip_org_role_sync`	Resulting Org Role	Modifiable
false	false	Synchronize user organization role with the provider's role. If no role is provided, `auto_assign_org_role` is set.	false
true	false	Skips organization role synchronization for all OAuth providers' users. Role is set to `auto_assign_org_role`.	true
false	true	Skips organization role and Grafana Admin synchronization for the provider's users. Role is set to `auto_assign_org_role`.	true
true	true	Skips organization role synchronization for all OAuth providers and skips Grafana Admin synchronization for the provider's users. Role is set to `auto_assign_org_role`.	true

[auth.okta]

oauth_skip_org_role_update_sync	skip_org_role_sync	Resulting Org Role	Modifiable
false	false	Synchronize user organization role with Okta role. If no role is provided, `auto_assign_org_role` is set.	false
true	false	Skips organization role synchronization for all OAuth providers' users. Role is set to `auto_assign_org_role`.	true
false	true	Skips organization role and Grafana Admin synchronization for Okta users. Role is set to `auto_assign_org_role`.	true
true	true	Skips organization role synchronization for all OAuth providers and skips Grafana Admin synchronization for Okta users. Role is set to `auto_assign_org_role`.	true

Example skip_org_role_sync

[auth.google]

oauth_skip_org_role_update_sync	skip_org_role_sync	Resulting Org Role	Example Scenario
false	false	Synchronized with Google Auth organization roles	A user logs in to Grafana using their Google account and their organization role is automatically set based on their role in Google.
true	false	Skipped synchronization of organization roles from all OAuth providers	A user logs in to Grafana using their Google account and their organization role is **not** set based on their role. But Grafana Administrators can modify the role from the UI.
false	true	Skipped synchronization of	A user logs in to Grafana using their

`oauth_skip_org_role_update_sync`	`skip_org_role_sync`	**Resulting Org Role** organization roles Google	**Example Scenario** Google account and their organization role is **not** set based on their role in Google. But Grafana Administrators can modify the role from the UI.
true	true	Skipped synchronization of organization roles from all OAuth providers including Google	A user logs in to Grafana using their Google account and their organization role is **not** set based on their role in Google. But Grafana Administrators can modify the role from the UI.

api_key_max_seconds_to_live

Limit of API key seconds to live before expiration. Default is -1 (unlimited).

sigv4_auth_enabled

> Only available in Grafana 7.3+.

Set to `true` to enable the AWS Signature Version 4 Authentication option for HTTP-based datasources. Default is `false`.

sigv4_verbose_logging

> Only available in Grafana 8.4+.

Set to `true` to enable verbose request signature logging when AWS Signature Version 4 Authentication is enabled. Default is `false`.

[auth.anonymous]

Refer to Anonymous authentication for detailed instructions.

[auth.github]

Refer to GitHub OAuth2 authentication for detailed instructions.

[auth.gitlab]

Refer to Gitlab OAuth2 authentication for detailed instructions.

[auth.google]

Refer to Google OAuth2 authentication for detailed instructions.

[auth.grafananet]

Legacy key names, still in the config file so they work in env variables.

[auth.grafana_com]

Legacy key names, still in the config file so they work in env variables.

[auth.azuread]

Refer to Azure AD OAuth2 authentication for detailed instructions.

[auth.okta]

Refer to Okta OAuth2 authentication for detailed instructions.

[auth.generic_oauth]

Refer to Generic OAuth authentication for detailed instructions.

[auth.basic]

Refer to Basic authentication for detailed instructions.

[auth.proxy]

Refer to Auth proxy authentication for detailed instructions.

[auth.ldap]

Refer to LDAP authentication for detailed instructions.

[aws]

You can configure core and external AWS plugins.

allowed_auth_providers

Specify what authentication providers the AWS plugins allow. For a list of allowed providers, refer to the data-source configuration page for a given plugin. If you configure a plugin by provisioning, only providers that are specified in `allowed_auth_providers` are allowed.

Options: `default` (AWS SDK default), `keys` (Access and secret key), `credentials` (Credentials file), `ec2_iam_role` (EC2 IAM role)

assume_role_enabled

Set to `false` to disable AWS authentication from using an assumed role with temporary security credentials. For details about assume roles, refer to the AWS API reference documentation about the AssumeRole operation.

If this option is disabled, the **Assume Role** and the **External Id** field are removed from the AWS data source configuration page. If the plugin is configured using provisioning, it is possible to use an assumed role as long as `assume_role_enabled` is set to `true`.

list_metrics_page_limit

Use the List Metrics API option to load metrics for custom namespaces in the CloudWatch data source. By default, the page limit is 500.

[azure]

Grafana supports additional integration with Azure services when hosted in the Azure Cloud.

cloud

Azure cloud environment where Grafana is hosted:

Azure Cloud	Value
Microsoft Azure public cloud	AzureCloud (*default*)
Microsoft Chinese national cloud	AzureChinaCloud
US Government cloud	AzureUSGovernment
Microsoft German national cloud (“Black Forest”)	AzureGermanCloud

managed_identity_enabled

Specifies whether Grafana hosted in Azure service with Managed Identity configured (e.g. Azure Virtual Machines instance). Disabled by default, needs to be explicitly enabled.

managed_identity_client_id

The client ID to use for user-assigned managed identity.

Should be set for user-assigned identity and should be empty for system-assigned identity.

workload_identity_enabled

Specifies whether Azure AD Workload Identity authentication should be enabled in datasources that support it.

For more documentation on Azure AD Workload Identity, review Azure AD Workload Identity documentation.

Disabled by default, needs to be explicitly enabled.

workload_identity_tenant_id

Tenant ID of the Azure AD Workload Identity.

Allows to override default tenant ID of the Azure AD identity associated with the Kubernetes service account.

workload_identity_client_id

Client ID of the Azure AD Workload Identity.

Allows to override default client ID of the Azure AD identity associated with the Kubernetes service account.

workload_identity_token_file

Custom path to token file for the Azure AD Workload Identity.

Allows to set a custom path to the projected service account token file.

user_identity_enabled

Specifies whether user identity authentication (on behalf of currently signed-in user) should be enabled in datasources that support it (requires AAD authentication).

Disabled by default, needs to be explicitly enabled.

user_identity_token_url

Override token URL for Azure Active Directory.

By default is the same as token URL configured for AAD authentication settings.

user_identity_client_id

Override ADD application ID which would be used to exchange users token to an access token for the datasource.

By default is the same as used in AAD authentication or can be set to another application (for OBO flow).

user_identity_client_secret

Override the AAD application client secret.

By default is the same as used in AAD authentication or can be set to another application (for OBO flow).

forward_settings_to_plugins

Set plugins that will receive Azure settings via plugin context.

By default, this will include all Grafana Labs owned Azure plugins or those that use Azure settings (Azure Monitor, Azure Data Explorer, Prometheus, MSSQL).

[auth.jwt]

Refer to JWT authentication for more information.

[smtp]

Email server settings.

enabled

Enable this to allow Grafana to send email. Default is `false`.

host

Default is `localhost:25`. Use port 465 for implicit TLS.

user

In case of SMTP auth, default is `empty`.

password

In case of SMTP auth, default is `empty`. If the password contains `#` or `;`, then you have to wrap it with triple quotes. Example: """#password;"""

cert_file

File path to a cert file, default is `empty`.

key_file

File path to a key file, default is `empty`.

skip_verify

Verify SSL for SMTP server, default is `false`.

from_address

Address used when sending out emails, default is `admin@grafana.localhost`.

from_name

Name to be used when sending out emails, default is `Grafana`.

ehlo_identity

Name to be used as client identity for EHLO in SMTP dialog, default is `<instance_name>`.

startTLS_policy

Either “OpportunisticStartTLS”, “MandatoryStartTLS”, “NoStartTLS”. Default is `empty`.

enable_tracing

Enable trace propagation in e-mail headers, using the `traceparent`, `tracestate` and (optionally) `baggage` fields. Default is `false`. To enable, you must first configure tracing in one of the `tracing.oentelemetry.*` sections.

[smtp.static_headers]

Enter key-value pairs on their own lines to be included as headers on outgoing emails. All keys must be in canonical mail header format. Examples: `Foo=bar`, `Foo-Header=bar`.

[emails]

welcome_email_on_sign_up

Default is `false`.

templates_pattern

Enter a comma separated list of template patterns. Default is `emails/*.html, emails/*.txt`.

content_types

Enter a comma-separated list of content types that should be included in the emails that are sent. List the content types according descending preference, e.g. `text/html, text/plain` for HTML as the most preferred. The order of the parts is significant as the mail clients will use the content type that is supported and most preferred by the sender. Supported content types are `text/html` and `text/plain`. Default is `text/html`.

[log]

Grafana logging options.

mode

Options are “console”, “file”, and “syslog”. Default is “console” and “file”. Use spaces to separate multiple modes, e.g. `console file`.

level

Options are “debug”, “info”, “warn”, “error”, and “critical”. Default is `info`.

filters

Optional settings to set different levels for specific loggers. For example: `filters = sqlstore:debug`

user_facing_default_error

Use this configuration option to set the default error message shown to users. This message is displayed instead of sensitive backend errors, which should be obfuscated. The default message is `Please inspect the Grafana server log for details.`.

[log.console]

Only applicable when “console” is used in `[log]` mode.

level

Options are “debug”, “info”, “warn”, “error”, and “critical”. Default is inherited from `[log]` level.

format

Log line format, valid options are text, console and json. Default is `console`.

[log.file]

Only applicable when “file” used in `[log]` mode.

level

Options are “debug”, “info”, “warn”, “error”, and “critical”. Default is inherited from `[log]` level.

format

Log line format, valid options are text, console and json. Default is `text`.

log_rotate

Enable automated log rotation, valid options are `false` or `true`. Default is `true`. When enabled use the `max_lines`, `max_size_shift`, `daily_rotate` and `max_days` to configure the behavior of the log rotation.

max_lines

Maximum lines per file before rotating it. Default is `1000000`.

max_size_shift

Maximum size of file before rotating it. Default is `28`, which means `1 << 28`, `256MB`.

daily_rotate

Enable daily rotation of files, valid options are `false` or `true`. Default is `true`.

max_days

Maximum number of days to keep log files. Default is `7`.

[log.syslog]

Only applicable when “syslog” used in `[log]` mode.

level

Options are “debug”, “info”, “warn”, “error”, and “critical”. Default is inherited from `[log]` level.

format

Log line format, valid options are text, console, and json. Default is `text`.

network and address

Syslog network type and address. This can be UDP, TCP, or UNIX. If left blank, then the default UNIX endpoints are used.

facility

Syslog facility. Valid options are user, daemon or local0 through local7. Default is empty.

tag

Syslog tag. By default, the process’s `argv[0]` is used.

[log.frontend]

Note: This feature is available in Grafana 7.4+.

enabled

Faro javascript agent is initialized. Default is `false`.

custom_endpoint

Custom HTTP endpoint to send events captured by the Faro agent to. Default, `/log-grafana-javascript-agent`, will log the events to stdout.

log_endpoint_requests_per_second_limit

Requests per second limit enforced per an extended period, for Grafana backend log ingestion endpoint, `/log-grafana-javascript-agent`. Default is `3`.

log_endpoint_burst_limit

Maximum requests accepted per short interval of time for Grafana backend log ingestion endpoint, `/log-grafana-javascript-agent`. Default is `15`.

instrumentations_errors_enabled

Turn on error instrumentation. Only affects Grafana Javascript Agent.

instrumentations_console_enabled

Turn on console instrumentation. Only affects Grafana Javascript Agent

instrumentations_webvitals_enabled

Turn on webvitals instrumentation. Only affects Grafana Javascript Agent

api_key

If `custom_endpoint` required authentication, you can set the api key here. Only relevant for Grafana Javascript Agent provider.

[quota]

Set quotas to `-1` to make unlimited.

enabled

Enable usage quotas. Default is `false`.

org_user

Limit the number of users allowed per organization. Default is 10.

org_dashboard

Limit the number of dashboards allowed per organization. Default is 100.

org_data_source

Limit the number of data sources allowed per organization. Default is 10.

org_api_key

Limit the number of API keys that can be entered per organization. Default is 10.

org_alert_rule

Limit the number of alert rules that can be entered per organization. Default is 100.

user_org

Limit the number of organizations a user can create. Default is 10.

global_user

Sets a global limit of users. Default is -1 (unlimited).

global_org

Sets a global limit on the number of organizations that can be created. Default is -1 (unlimited).

global_dashboard

Sets a global limit on the number of dashboards that can be created. Default is -1 (unlimited).

global_api_key

Sets global limit of API keys that can be entered. Default is -1 (unlimited).

global_session

Sets a global limit on number of users that can be logged in at one time. Default is -1 (unlimited).

global_alert_rule

Sets a global limit on number of alert rules that can be created. Default is -1 (unlimited).

global_correlations

Sets a global limit on number of correlations that can be created. Default is -1 (unlimited).

[unified_alerting]

For more information about the Grafana alerts, refer to About Grafana Alerting.

enabled

Enable or disable Grafana Alerting. If disabled, all your legacy alerting data will be available again. The default value is `true`.

Alerting Rules migrated from dashboards and panels will include a link back via the `annotations`.

disabled_orgs

Comma-separated list of organization IDs for which to disable Grafana 8 Unified Alerting.

admin_config_poll_interval

Specify the frequency of polling for admin config changes. The default value is `60s`.

The interval string is a possibly signed sequence of decimal numbers, followed by a unit suffix (ms, s, m, h, d), e.g. 30s or 1m.

alertmanager_config_poll_interval

Specify the frequency of polling for Alertmanager config changes. The default value is `60s`.

The interval string is a possibly signed sequence of decimal numbers, followed by a unit suffix (ms, s, m, h, d), e.g. 30s or 1m.

ha_redis_address

The Redis server address that should be connected to.

ha_redis_username

The username that should be used to authenticate with the Redis server.

ha_redis_password

The password that should be used to authenticate with the Redis server.

ha_redis_db

The Redis database. The default value is `0`.

ha_redis_prefix

A prefix that is used for every key or channel that is created on the Redis server as part of HA for alerting.

ha_redis_peer_name

The name of the cluster peer that will be used as an identifier. If none is provided, a random one will be generated.

ha_redis_max_conns

The maximum number of simultaneous Redis connections.

ha_listen_address

Listen IP address and port to receive unified alerting messages for other Grafana instances. The port is used for both TCP and UDP. It is assumed other Grafana instances are also running on the same port. The default value is `0.0.0.0:9094`.

ha_advertise_address

Explicit IP address and port to advertise other Grafana instances. The port is used for both TCP and UDP.

ha_peers

Comma-separated list of initial instances (in a format of host:port) that will form the HA cluster. Configuring this setting will enable High Availability mode for alerting.

ha_peer_timeout

Time to wait for an instance to send a notification via the Alertmanager. In HA, each Grafana instance will be assigned a position (e.g. 0, 1). We then multiply this position with the timeout to indicate how long should each instance wait before sending the notification to take into account replication lag. The default value is `15s`.

The interval string is a possibly signed sequence of decimal numbers, followed by a unit suffix (ms, s, m, h, d), e.g. 30s or 1m.

ha_label

The label is an optional string to include on each packet and stream. It uniquely identifies the cluster and prevents cross-communication issues when sending gossip messages in an environment with multiple clusters.

ha_gossip_interval

The interval between sending gossip messages. By lowering this value (more frequent) gossip messages are propagated across cluster more quickly at the expense of increased bandwidth usage. The default value is `200ms`.

The interval string is a possibly signed sequence of decimal numbers, followed by a unit suffix (ms, s, m, h, d), e.g. 30s or 1m.

ha_push_pull_interval

The interval between gossip full state syncs. Setting this interval lower (more frequent) will increase convergence speeds across larger clusters at the expense of increased bandwidth usage. The default value is `60s`.

The interval string is a possibly signed sequence of decimal numbers, followed by a unit suffix (ms, s, m, h, d), e.g. 30s or 1m.

execute_alerts

Enable or disable alerting rule execution. The default value is `true`. The alerting UI remains visible. This option has a legacy version in the alerting section that takes precedence.

evaluation_timeout

Sets the alert evaluation timeout when fetching data from the data source. The default value is `30s`. This option has a legacy version in the alerting section that takes precedence.

The timeout string is a possibly signed sequence of decimal numbers, followed by a unit suffix (ms, s, m, h, d), e.g. 30s or 1m.

max_attempts

Sets a maximum number of times we'll attempt to evaluate an alert rule before giving up on that evaluation. The default value is `1`.

min_interval

Sets the minimum interval to enforce between rule evaluations. The default value is `10s` which equals the scheduler interval. Rules will be adjusted if they are less than this value or if they are not multiple of the scheduler interval (10s). Higher values can help with resource management as we'll schedule fewer evaluations over time. This option has a legacy version in the alerting section that takes precedence.

The interval string is a possibly signed sequence of decimal numbers, followed by a unit suffix (ms, s, m, h, d), e.g. 30s or 1m.

Note. This setting has precedence over each individual rule frequency. If a rule frequency is lower than this value, then this value is enforced.

[unified_alerting.screenshots]

For more information about screenshots, refer to Images in notifications.

capture

Enable screenshots in notifications. This option requires a remote HTTP image rendering service. Please see `[rendering]` for further configuration options.

max_concurrent_screenshots

The maximum number of screenshots that can be taken at the same time. This option is different from `concurrent_render_request_limit` as `max_concurrent_screenshots` sets the number of concurrent screenshots that can be taken at the same time for all firing alerts where as concurrent_render_request_limit sets the total number of concurrent screenshots across all Grafana services.

upload_external_image_storage

Uploads screenshots to the local Grafana server or remote storage such as Azure, S3 and GCS. Please see `[external_image_storage]` for further configuration options. If this option is false then screenshots will be persisted to disk for up to `temp_data_lifetime`.

[unified_alerting.reserved_labels]

For more information about Grafana Reserved Labels, refer to Labels in Grafana Alerting

disabled_labels

Comma-separated list of reserved labels added by the Grafana Alerting engine that should be disabled.

For example: `disabled_labels=grafana_folder`

[unified_alerting.upgrade]

For more information about upgrading to Grafana Alerting, refer to Upgrade Alerting.

clean_upgrade

When you restart Grafana to upgrade from legacy alerting to Grafana Alerting, delete any existing Grafana Alerting data from a previous upgrade, such as alert rules, contact points, and notification policies. Default is `false`.

If `false` or unset, existing Grafana Alerting data is not changed or deleted when you switch between legacy and Unified Alerting.

> **NOTE**
>
> It should be kept false when not needed, as it may cause unintended data loss if left enabled.

[alerting]

For more information about the legacy dashboard alerting feature in Grafana, refer to the legacy Grafana alerts.

enabled

Set to `true` to enable legacy dashboard alerting. The default value is `false`.

execute_alerts

Turns off alert rule execution, but alerting is still visible in the Grafana UI.

error_or_timeout

Default setting for new alert rules. Defaults to categorize error and timeouts as alerting. (alerting, keep_state)

nodata_or_nullvalues

Defines how Grafana handles nodata or null values in alerting. Options are `alerting`, `no_data`, `keep_state`, and `ok`. Default is `no_data`.

concurrent_render_limit

Alert notifications can include images, but rendering many images at the same time can overload the server. This limit protects the server from render overloading and ensures notifications are sent

out quickly. Default value is `5`.

evaluation_timeout_seconds

Sets the alert calculation timeout. Default value is `30`.

notification_timeout_seconds

Sets the alert notification timeout. Default value is `30`.

max_attempts

Sets a maximum limit on attempts to sending alert notifications. Default value is `3`.

min_interval_seconds

Sets the minimum interval between rule evaluations. Default value is `1`.

Note. This setting has precedence over each individual rule frequency. If a rule frequency is lower than this value, then this value is enforced.

max_annotation_age =

Configures for how long alert annotations are stored. Default is 0, which keeps them forever. This setting should be expressed as a duration. Examples: 6h (hours), 10d (days), 2w (weeks), 1M (month).

max_annotations_to_keep =

Configures max number of alert annotations that Grafana stores. Default value is 0, which keeps all alert annotations.

[annotations]

cleanupjob_batchsize

Configures the batch size for the annotation clean-up job. This setting is used for dashboard, API, and alert annotations.

tags_length

Enforces the maximum allowed length of the tags for any newly introduced annotations. It can be between 500 and 4096 (inclusive). Default value is 500. Setting it to a higher value would impact

performance therefore is not recommended.

[annotations.dashboard]

Dashboard annotations means that annotations are associated with the dashboard they are created on.

max_age

Configures how long dashboard annotations are stored. Default is 0, which keeps them forever. This setting should be expressed as a duration. Examples: 6h (hours), 10d (days), 2w (weeks), 1M (month).

max_annotations_to_keep

Configures max number of dashboard annotations that Grafana stores. Default value is 0, which keeps all dashboard annotations.

[annotations.api]

API annotations means that the annotations have been created using the API without any association with a dashboard.

max_age

Configures how long Grafana stores API annotations. Default is 0, which keeps them forever. This setting should be expressed as a duration. Examples: 6h (hours), 10d (days), 2w (weeks), 1M (month).

max_annotations_to_keep

Configures max number of API annotations that Grafana keeps. Default value is 0, which keeps all API annotations.

[explore]

For more information about this feature, refer to Explore.

enabled

Enable or disable the Explore section. Default is `enabled`.

[help]

Configures the help section.

enabled

Enable or disable the Help section. Default is `enabled`.

[profile]

Configures the Profile section.

enabled

Enable or disable the Profile section. Default is `enabled`.

[news]

news_feed_enabled

Enables the news feed section. Default is `true`

[query]

concurrent_query_limit

Set the number of queries that can be executed concurrently in a mixed data source panel. Default is the number of CPUs.

[query_history]

Configures Query history in Explore.

enabled

Enable or disable the Query history. Default is `enabled`.

[metrics]

For detailed instructions, refer to Internal Grafana metrics.

enabled

Enable metrics reporting. defaults true. Available via HTTP API `<URL>/metrics`.

interval_seconds

Flush/write interval when sending metrics to external TSDB. Defaults to `10`.

disable_total_stats

If set to `true`, then total stats generation (`stat_totals_*` metrics) is disabled. Default is `false`.

total_stats_collector_interval_seconds

Sets the total stats collector interval. The default is 1800 seconds (30 minutes).

basic_auth_username and basic_auth_password

If both are set, then basic authentication is required to access the metrics endpoint.

[metrics.environment_info]

Adds dimensions to the `grafana_environment_info` metric, which can expose more information about the Grafana instance.

```
; exampleLabel1 = exampleValue1
; exampleLabel2 = exampleValue2
```

Copy

[metrics.graphite]

Use these options if you want to send internal Grafana metrics to Graphite.

address

Enable by setting the address. Format is `<Hostname or ip>`:port.

prefix

Graphite metric prefix. Defaults to `prod.grafana.%(instance_name)s.`

[grafana_net]

url

Default is https://grafana.com.

[grafana_com]

url

Default is https://grafana.com.

[tracing.jaeger]

[Deprecated - use tracing.opentelemetry.jaeger or tracing.opentelemetry.otlp instead]

Configure Grafana's Jaeger client for distributed tracing.

You can also use the standard `JAEGER_*` environment variables to configure Jaeger. See the table at the end of https://www.jaegertracing.io/docs/1.16/client-features/ for the full list. Environment variables will override any settings provided here.

address

The host:port destination for reporting spans. (ex: `localhost:6831`)

Can be set with the environment variables `JAEGER_AGENT_HOST` and `JAEGER_AGENT_PORT`.

always_included_tag

Comma-separated list of tags to include in all new spans, such as `tag1:value1,tag2:value2`.

Can be set with the environment variable `JAEGER_TAGS` (use `=` instead of `:` with the environment variable).

sampler_type

Default value is `const`.

Specifies the type of sampler: `const`, `probabilistic`, `ratelimiting`, or `remote`.

Refer to https://www.jaegertracing.io/docs/1.16/sampling/#client-sampling-configuration for details on the different tracing types.

Can be set with the environment variable `JAEGER_SAMPLER_TYPE`.

To override this setting, enter `sampler_type` in the `tracing.opentelemetry` section.

sampler_param

Default value is `1`.

This is the sampler configuration parameter. Depending on the value of `sampler_type`, it can be `0`, `1`, or a decimal value in between.

- For `const` sampler, `0` or `1` for always `false`/`true` respectively
- For `probabilistic` sampler, a probability between `0` and `1.0`
- For `rateLimiting` sampler, the number of spans per second
- For `remote` sampler, param is the same as for `probabilistic` and indicates the initial sampling rate before the actual one is received from the mothership

May be set with the environment variable `JAEGER_SAMPLER_PARAM`.

Setting `sampler_param` in the `tracing.opentelemetry` section will override this setting.

sampling_server_url

sampling_server_url is the URL of a sampling manager providing a sampling strategy.

Setting `sampling_server_url` in the `tracing.opentelemetry` section will override this setting.

zipkin_propagation

Default value is `false`.

Controls whether or not to use Zipkin's span propagation format (with `x-b3-` HTTP headers). By default, Jaeger's format is used.

Can be set with the environment variable and value `JAEGER_PROPAGATION=b3`.

disable_shared_zipkin_spans

Default value is `false`.

Setting this to `true` turns off shared RPC spans. Leaving this available is the most common setting when using Zipkin elsewhere in your infrastructure.

[tracing.opentelemetry]

Configure general parameters shared between OpenTelemetry providers.

custom_attributes

Comma-separated list of attributes to include in all new spans, such as `key1:value1,key2:value2`.

Can be set with the environment variable `OTEL_RESOURCE_ATTRIBUTES` (use `=` instead of `:` with the environment variable).

sampler_type

Default value is `const`.

Specifies the type of sampler: `const`, `probabilistic`, `ratelimiting`, or `remote`.

sampler_param

Default value is `1`.

Depending on the value of `sampler_type`, the sampler configuration parameter can be `0`, `1`, or any decimal value between `0` and `1`.

- For the `const` sampler, use `0` to never sample or `1` to always sample
- For the `probabilistic` sampler, you can use a decimal value between `0.0` and `1.0`
- For the `rateLimiting` sampler, enter the number of spans per second
- For the `remote` sampler, use a decimal value between `0.0` and `1.0` to specify the initial sampling rate used before the first update is received from the sampling server

sampling_server_url

When `sampler_type` is `remote`, this specifies the URL of the sampling server. This can be used by all tracing providers.

Use a sampling server that supports the Jaeger remote sampling API, such as jaeger-agent, jaeger-collector, opentelemetry-collector-contrib, or Grafana Alloy.

[tracing.opentelemetry.jaeger]

Configure Grafana's Jaeger client for distributed tracing.

address

The host:port destination for reporting spans. (ex: `localhost:14268/api/traces`)

propagation

The propagation specifies the text map propagation format. The values `jaeger` and `w3c` are supported. Add a comma (`,`) between values to specify multiple formats (for example, `"jaeger,w3c"`). The default value is `w3c`.

[tracing.opentelemetry.otlp]

Configure Grafana's otlp client for distributed tracing.

address

The host:port destination for reporting spans. (ex: `localhost:4317`)

propagation

The propagation specifies the text map propagation format. The values `jaeger` and `w3c` are supported. Add a comma (`,`) between values to specify multiple formats (for example, `"jaeger,w3c"`). The default value is `w3c`.

[external_image_storage]

These options control how images should be made public so they can be shared on services like Slack or email message.

provider

Options are s3, webdav, gcs, azure_blob, local). If left empty, then Grafana ignores the upload action.

[external_image_storage.s3]

endpoint

Optional endpoint URL (hostname or fully qualified URI) to override the default generated S3 endpoint. If you want to keep the default, just leave this empty. You must still provide a `region` value if you specify an endpoint.

path_style_access

Set this to true to force path-style addressing in S3 requests, i.e., `http://s3.amazonaws.com/BUCKET/KEY`, instead of the default, which is virtual hosted bucket addressing when possible (`http://BUCKET.s3.amazonaws.com/KEY`).

> **NOTE**
>
> This option is specific to the Amazon S3 service.

bucket_url

(for backward compatibility, only works when no bucket or region are configured) Bucket URL for S3. AWS region can be specified within URL or defaults to 'us-east-1', e.g.

- http://grafana.s3.amazonaws.com/
- https://grafana.s3-ap-southeast-2.amazonaws.com/

bucket

Bucket name for S3. e.g. grafana.snapshot.

region

Region name for S3. e.g. ‘us-east-1’, ‘cn-north-1’, etc.

path

Optional extra path inside bucket, useful to apply expiration policies.

access_key

Access key, e.g. AAAAAAAAAAAAAAAAAAA.

Access key requires permissions to the S3 bucket for the ‘s3:PutObject’ and ‘s3:PutObjectAcl’ actions.

secret_key

Secret key, e.g. AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA.

[external_image_storage.webdav]

url

URL where Grafana sends PUT request with images.

username

Basic auth username.

password

Basic auth password.

public_url

Optional URL to send to users in notifications. If the string contains the sequence `{{file}}`, it is replaced with the uploaded filename. Otherwise, the file name is appended to the path part of the URL, leaving any query string unchanged.

[external_image_storage.gcs]

key_file

Optional path to JSON key file associated with a Google service account to authenticate and authorize. If no value is provided it tries to use the application default credentials. Service Account keys can be created and downloaded from https://console.developers.google.com/permissions/serviceaccounts.

Service Account should have “Storage Object Writer” role. The access control model of the bucket needs to be “Set object-level and bucket-level permissions”. Grafana itself will make the images public readable when signed urls are not enabled.

bucket

Bucket Name on Google Cloud Storage.

path

Optional extra path inside bucket.

enable_signed_urls

If set to true, Grafana creates a signed URL for the image uploaded to Google Cloud Storage.

signed_url_expiration

Sets the signed URL expiration, which defaults to seven days.

[external_image_storage.azure_blob]

account_name

Storage account name.

account_key

Storage account key

container_name

Container name where to store “Blob” images with random names. Creating the blob container beforehand is required. Only public containers are supported.

sas_token_expiration_days

Number of days for SAS token validity. If specified SAS token will be attached to image URL. Allow storing images in private containers.

[external_image_storage.local]

This option does not require any configuration.

[rendering]

Options to configure a remote HTTP image rendering service, e.g. using https://github.com/grafana/grafana-image-renderer.

renderer_token

> **NOTE**
>
> Available in Grafana v9.1.2 and Image Renderer v3.6.1 or later.

An auth token will be sent to and verified by the renderer. The renderer will deny any request without an auth token matching the one configured on the renderer.

server_url

URL to a remote HTTP image renderer service, e.g. http://localhost:8081/render, will enable Grafana to render panels and dashboards to PNG-images using HTTP requests to an external service.

callback_url

If the remote HTTP image renderer service runs on a different server than the Grafana server you may have to configure this to a URL where Grafana is reachable, e.g. http://grafana.domain/.

concurrent_render_request_limit

Concurrent render request limit affects when the /render HTTP endpoint is used. Rendering many images at the same time can overload the server, which this setting can help protect against by only allowing a certain number of concurrent requests. Default is `30`.

[panels]

enable_alpha

Set to `true` if you want to test alpha panels that are not yet ready for general usage. Default is `false`.

disable_sanitize_html

> **NOTE**
>
> This configuration is not available in Grafana Cloud instances.

If set to true Grafana will allow script tags in text panels. Not recommended as it enables XSS vulnerabilities. Default is false.

[plugins]

enable_alpha

Set to `true` if you want to test alpha plugins that are not yet ready for general usage. Default is `false`.

allow_loading_unsigned_plugins

Enter a comma-separated list of plugin identifiers to identify plugins to load even if they are unsigned. Plugins with modified signatures are never loaded.

We do *not* recommend using this option. For more information, refer to Plugin signatures.

plugin_admin_enabled

Available to Grafana administrators only, enables installing / uninstalling / updating plugins directly from the Grafana UI. Set to `true` by default. Setting it to `false` will hide the install / uninstall / update controls.

For more information, refer to Plugin catalog.

plugin_admin_external_manage_enabled

Set to `true` if you want to enable external management of plugins. Default is `false`. This is only applicable to Grafana Cloud users.

plugin_catalog_url

Custom install/learn more URL for enterprise plugins. Defaults to https://grafana.com/grafana/plugins/.

plugin_catalog_hidden_plugins

Enter a comma-separated list of plugin identifiers to hide in the plugin catalog.

public_key_retrieval_disabled

Disable download of the public key for verifying plugin signature. The default is `false`. If disabled, it will use the hardcoded public key.

public_key_retrieval_on_startup

Force download of the public key for verifying plugin signature on startup. The default is `false`. If disabled, the public key will be retrieved every 10 days. Requires `public_key_retrieval_disabled` to be false to have any effect.

disable_plugins

Enter a comma-separated list of plugin identifiers to avoid loading (including core plugins). These plugins will be hidden in the catalog.

[live]

max_connections

> **NOTE**
>
> Available in Grafana v8.0 and later versions.

The `max_connections` option specifies the maximum number of connections to the Grafana Live WebSocket endpoint per Grafana server instance. Default is `100`.

Refer to Grafana Live configuration documentation if you specify a number higher than default since this can require some operating system and infrastructure tuning.

0 disables Grafana Live, -1 means unlimited connections.

allowed_origins

> **NOTE**
>
> Available in Grafana v8.0.4 and later versions.

The `allowed_origins` option is a comma-separated list of additional origins (`Origin` header of HTTP Upgrade request during WebSocket connection establishment) that will be accepted by Grafana Live.

If not set (default), then the origin is matched over root_url which should be sufficient for most scenarios.

Origin patterns support wildcard symbol “*”.

For example:

```ini
[live]
allowed_origins = "https://*.example.com"
```

ha_engine

> **NOTE**
>
> Available in Grafana v8.1 and later versions.

Experimental

The high availability (HA) engine name for Grafana Live. By default, it’s not set. The only possible value is “redis”.

For more information, refer to the Configure Grafana Live HA setup.

ha_engine_address

> **NOTE**
>
> Available in Grafana v8.1 and later versions.

Experimental

Address string of selected the high availability (HA) Live engine. For Redis, it’s a `host:port` string. Example:

```ini
[live]
ha_engine = redis
ha_engine_address = 127.0.0.1:6379
```

[plugin.plugin_id]

This section can be used to configure plugin-specific settings. Replace the `plugin_id` attribute with the plugin ID present in `plugin.json`.

Properties described in this section are available for all plugins, but you must set them individually for each plugin.

tracing

> **NOTE**
>
> Available in Grafana v9.5.0 or later, and OpenTelemetry must be configured as well.

If `true`, propagate the tracing context to the plugin backend and enable tracing (if the backend supports it).

as_external

Load an external version of a core plugin if it has been installed.

Experimental. Requires the feature toggle `externalCorePlugins` to be enabled.

[plugin.grafana-image-renderer]

For more information, refer to Image rendering.

rendering_timezone

Instruct headless browser instance to use a default timezone when not provided by Grafana, e.g. when rendering panel image of alert. See ICUs metaZones.txt for a list of supported timezone IDs. Fallbacks to TZ environment variable if not set.

rendering_language

Instruct headless browser instance to use a default language when not provided by Grafana, e.g. when rendering panel image of alert. Refer to the HTTP header Accept-Language to understand how to format this value, e.g. 'fr-CH, fr;q=0.9, en;q=0.8, de;q=0.7, *;q=0.5'.

rendering_viewport_device_scale_factor

Instruct headless browser instance to use a default device scale factor when not provided by Grafana, e.g. when rendering panel image of alert. Default is `1`. Using a higher value will produce

more detailed images (higher DPI), but requires more disk space to store an image.

rendering_ignore_https_errors

Instruct headless browser instance whether to ignore HTTPS errors during navigation. Per default HTTPS errors are not ignored. Due to the security risk, we do not recommend that you ignore HTTPS errors.

rendering_verbose_logging

Instruct headless browser instance whether to capture and log verbose information when rendering an image. Default is `false` and will only capture and log error messages.

When enabled, debug messages are captured and logged as well.

For the verbose information to be included in the Grafana server log you have to adjust the rendering log level to debug, configure [log].filter = rendering:debug.

rendering_dumpio

Instruct headless browser instance whether to output its debug and error messages into running process of remote rendering service. Default is `false`.

It can be useful to set this to `true` when troubleshooting.

rendering_timing_metrics

Note: Available from grafana-image-renderer v3.9.0+

Instruct a headless browser instance on whether to record metrics for the duration of every rendering step. Default is `false`.

Setting this to `true` when optimizing the rendering mode settings to improve the plugin performance or when troubleshooting can be useful.

rendering_args

Additional arguments to pass to the headless browser instance. Defaults are `--no-sandbox,--disable-gpu`. The list of Chromium flags can be found at (https://peter.sh/experiments/chromium-command-line-switches/). Separate multiple arguments with commas.

rendering_chrome_bin

You can configure the plugin to use a different browser binary instead of the pre-packaged version of Chromium.

Please note that this is *not* recommended. You might encounter problems if the installed version of Chrome/Chromium is not compatible with the plugin.

rendering_mode

Instruct how headless browser instances are created. Default is `default` and will create a new browser instance on each request.

Mode `clustered` will make sure that only a maximum of browsers/incognito pages can execute concurrently.

Mode `reusable` will have one browser instance and will create a new incognito page on each request.

rendering_clustering_mode

When rendering_mode = clustered, you can instruct how many browsers or incognito pages can execute concurrently. Default is `browser` and will cluster using browser instances.

Mode `context` will cluster using incognito pages.

rendering_clustering_max_concurrency

When rendering_mode = clustered, you can define the maximum number of browser instances/incognito pages that can execute concurrently. Default is `5`.

rendering_clustering_timeout

> **NOTE**
>
> Available in grafana-image-renderer v3.3.0 and later versions.

When rendering_mode = clustered, you can specify the duration a rendering request can take before it will time out. Default is `30` seconds.

rendering_viewport_max_width

Limit the maximum viewport width that can be requested.

rendering_viewport_max_height

Limit the maximum viewport height that can be requested.

rendering_viewport_max_device_scale_factor

Limit the maximum viewport device scale factor that can be requested.

grpc_host

Change the listening host of the gRPC server. Default host is `127.0.0.1`.

grpc_port

Change the listening port of the gRPC server. Default port is `0` and will automatically assign a port not in use.

[enterprise]

For more information about Grafana Enterprise, refer to Grafana Enterprise.

[feature_toggles]

enable

Keys of features to enable, separated by space.

FEATURE_TOGGLE_NAME = false

Some feature toggles for stable features are on by default. Use this setting to disable an on-by-default feature toggle with the name FEATURE_TOGGLE_NAME, for example, `exploreMixedDatasource = false`.

[feature_management]

The options in this section configure the experimental Feature Toggle Admin Page feature, which is enabled using the `featureToggleAdminPage` feature toggle. Grafana Labs offers support on a best-effort basis, and breaking changes might occur prior to the feature being made generally available.

Please see Configure feature toggles for more information.

allow_editing

Lets you switch the feature toggle state in the feature management page. The default is `false`.

update_webhook

Set the URL of the controller that manages the feature toggle updates. If not set, feature toggles in the feature management page will be read-only.

> **NOTE**
>
> The API for feature toggle updates has not been defined yet.

hidden_toggles

Hide additional specific feature toggles from the feature management page. By default, feature toggles in the `unknown`, `experimental`, and `private preview` stages are hidden from the UI. Use this option to hide toggles in the `public preview`, `general availability`, and `deprecated` stages.

read_only_toggles

Use to disable updates for additional specific feature toggles in the feature management page. By default, feature toggles can only be updated if they are in the `general availability` and `deprecated` stages. Use this option to disable updates for toggles in those stages.

[date_formats]

> **NOTE**
>
> The date format options below are only available in Grafana v7.2+.

This section controls system-wide defaults for date formats used in time ranges, graphs, and date input boxes.

The format patterns use Moment.js formatting tokens.

full_date

Full date format used by time range picker and in other places where a full date is rendered.

intervals

These intervals formats are used in the graph to show only a partial date or time. For example, if there are only minutes between Y-axis tick labels then the `interval_minute` format is used.

Defaults

```
interval_second = HH:mm:ss
interval_minute = HH:mm
interval_hour = MM/DD HH:mm
interval_day = MM/DD
```

Copy

```
interval_month = YYYY-MM
interval_year = YYYY
```

use_browser_locale

Set this to `true` to have date formats automatically derived from your browser location. Defaults to `false`. This is an experimental feature.

default_timezone

Used as the default time zone for user preferences. Can be either `browser` for the browser local time zone or a time zone name from the IANA Time Zone database, such as `UTC` or `Europe/Amsterdam`.

default_week_start

Set the default start of the week, valid values are: `saturday`, `sunday`, `monday` or `browser` to use the browser locale to define the first day of the week. Default is `browser`.

[expressions]

> **NOTE**
>
> This feature is available in Grafana v7.4 and later versions.

enabled

Set this to `false` to disable expressions and hide them in the Grafana UI. Default is `true`.

[geomap]

This section controls the defaults settings for Geomap Plugin.

default_baselayer_config

The json config used to define the default base map. Four base map options to choose from are `carto`, `esriXYZTiles`, `xyzTiles`, `standard`. For example, to set cartoDB light as the default base layer:

ini Copy

```
default_baselayer_config = `{
  "type": "xyz",
  "config": {
    "attribution": "Open street map",
```

```
    "url": "https://tile.openstreetmap.org/{z}/{x}/{y}.png"
  }
}`
```

enable_custom_baselayers

Set this to `false` to disable loading other custom base maps and hide them in the Grafana UI. Default is `true`.

[rbac]

Refer to Role-based access control for more information.

[navigation.app_sections]

Move an app plugin (referenced by its id), including all its pages, to a specific navigation section. Format: `<pluginId> = <sectionId> <sortWeight>`

[navigation.app_standalone_pages]

Move an individual app plugin page (referenced by its `path` field) to a specific navigation section. Format: `<pageUrl> = <sectionId> <sortWeight>`

[public_dashboards]

This section configures the public dashboards feature.

enabled

Set this to `false` to disable the public dashboards feature. This prevents users from creating new public dashboards and disables existing ones.

Introduction to time series

Imagine you wanted to know how the temperature outside changes throughout the day. Once every hour, you'd check the thermometer and write down the time along with the current temperature. After a while, you'd have something like this:

Time	Value
09:00	24°C
10:00	26°C
11:00	27°C

Temperature data like this is one example of what we call a *time series* — a sequence of measurements, ordered in time. Every row in the table represents one individual measurement at a specific time.

Tables are useful when you want to identify individual measurements, but they make it difficult to see the big picture. A more common visualization for time series is the *graph*, which instead places each measurement along a time axis. Visual representations like the graph make it easier to discover patterns and features of the data that otherwise would be difficult to see.

Temperature data like the one in the example, is far from the only example of a time series. Other examples of time series are:

- CPU and memory usage
- Sensor data
- Stock market index

While each of these examples are sequences of chronologically ordered measurements, they also share other attributes:

- New data is appended at the end, at regular intervals — for example, hourly at 09:00, 10:00, 11:00, and so on.
- Measurements are seldom updated after they were added — for example, yesterday's temperature doesn't change.

Time series are powerful. They help you understand the past by letting you analyze the state of the system at any point in time. Time series could tell you that the server crashed moments after the free disk space went down to zero.

Time series can also help you predict the future, by uncovering trends in your data. If the number of registered users has been increasing monthly by 4% for the past few months, you can predict how big your user base is going to be at the end of the year.

Some time series have patterns that repeat themselves over a known period. For example, the temperature is typically higher during the day, before it dips down at night. By identifying these periodic, or *seasonal*, time series, you can make confident predictions about the next period. If you know that the system load peaks every day around 18:00, you can add more machines right before.

Aggregating time series

Depending on what you're measuring, the data can vary greatly. What if you wanted to compare periods longer than the interval between measurements? If you'd measure the temperature once every hour, you'd end up with 24 data points per day. To compare the temperature in August over the years, you'd have to combine the 31 times 24 data points into one.

Combining a collection of measurements is called *aggregation*. There are several ways to aggregate time series data. Here are some common ones:

- **Average** returns the sum of all values divided by the total number of values.

- **Min** and **Max** return the smallest and largest value in the collection.
- **Sum** returns the sum of all values in the collection.
- **Count** returns the number of values in the collection.

For example, by aggregating the data in a month, you can determine that August 2017 was, on average, warmer than the year before. Instead, to see which month had the highest temperature, you'd compare the maximum temperature for each month.

How you choose to aggregate your time series data is an important decision and depends on the story you want to tell with your data. It's common to use different aggregations to visualize the same time series data in different ways.

Time series and monitoring

In the IT industry, time series data is often collected to monitor things like infrastructure, hardware, or application events. Machine-generated time series data is typically collected with short intervals, which allows you to react to any unexpected changes, moments after they occur. As a consequence, data accumulates at a rapid pace, making it vital to have a way to store and query data efficiently. As a result, databases optimized for time series data have seen a rise in popularity in recent years.

Time series databases

A time series database (TSDB) is a database explicitly designed for time series data. While it's possible to use any regular database to store measurements, a TSDB comes with some useful optimizations.

Modern time series databases take advantage of the fact that measurements are only ever appended, and rarely updated or removed. For example, the timestamps for each measurement change very little over time, which results in redundant data being stored.

Look at this sequence of Unix timestamps:

```
1572524345, 1572524375, 1572524404, 1572524434, 1572524464
```
Copy

Looking at these timestamps, they all start with `1572524`, leading to poor use of disk space. Instead, we could store each subsequent timestamp as the difference, or *delta*, from the first one:

```
1572524345, +30, +29, +30, +30
```
Copy

We could even take it a step further, by calculating the deltas of these deltas:

```
1572524345, +30, -1, +1, +0
```
Copy

If measurements are taken at regular intervals, most of these delta-of-deltas will be 0. Because of optimizations like these, TSDBs use drastically less space than other databases.

Another feature of a TSDB is the ability to filter measurements using *tags*. Each data point is labeled with a tag that adds context information, such as where the measurement was taken. Here's an example of the InfluxDB data format that demonstrates how each measurement is stored.

```
weather,location=us-midwest temperature=82 1465839830100400200
  |    -------------------- --------------  |
  |             |             |             |
  |             |             |             |
+-----------+--------+-+---------+-+---------+
|measurement|,tag_set| |field_set| |timestamp|
+-----------+--------+-+---------+-+---------+
```

Copy

Here are some of the TSDBs supported by Grafana:

- Graphite
- InfluxDB
- Prometheus

Collecting time series data

Now that we have a place to store our time series, how do we actually gather the measurements? To collect time series data, you'd typically install a *collector* on the device, machine, or instance you want to monitor. Some collectors are made with a specific database in mind, and some support different output destinations.

Here are some examples of collectors:

- collectd
- statsd
- Prometheus exporters
- Telegraf

A collector either *pushes* data to a database or lets the database *pull* the data from it. Both methods come with their own set of pros and cons:

	Pros	Cons
Push	Easier to replicate data to multiple destinations.	The TSDB has no control over how much data gets sent.
Pull	Better control of how much data that gets ingested, and its authenticity.	Firewalls, VPNs or load balancers can make it hard to access the agents.

Since it would be inefficient to write every measurement to the database, collectors pre-aggregate the data and write to the time series database at regular intervals.

Provision Grafana

In previous versions of Grafana, you could only use the API for provisioning data sources and dashboards. But that required the service to be running before you started creating dashboards and you also needed to set up credentials for the HTTP API. In v5.0 we decided to improve this experience by adding a new active provisioning system that uses config files. This will make GitOps more natural as data sources and dashboards can be defined via files that can be version controlled. We hope to extend this system to later add support for users, orgs and alerts as well.

Config File

See Configuration for more information on what you can configure in `grafana.ini`.

Config File Locations

- Default configuration from `$WORKING_DIR/conf/defaults.ini`
- Custom configuration from `$WORKING_DIR/conf/custom.ini`
- The custom configuration file path can be overridden using the `--config` parameter

NOTE

If you have installed Grafana using the `deb` or `rpm` packages, then your configuration file is located at `/etc/grafana/grafana.ini`. This path is specified in the Grafana init.d script using `--config` file parameter.

Using Environment Variables

It is possible to use environment variable interpolation in all 3 provisioning configuration types. Allowed syntax is either `$ENV_VAR_NAME` or `${ENV_VAR_NAME}` and can be used only for values not for

keys or bigger parts of the configurations. It is not available in the dashboard's definition files just the dashboard provisioning configuration. Example:

```yaml
datasources:
  - name: Graphite
    url: http://localhost:$PORT
    user: $USER
    secureJsonData:
      password: $PASSWORD
```

If you have a literal `$` in your value and want to avoid interpolation, `$$` can be used.

Configuration Management Tools

Currently we do not provide any scripts/manifests for configuring Grafana. Rather than spending time learning and creating scripts/manifests for each tool, we think our time is better spent making Grafana easier to provision. Therefore, we heavily rely on the expertise of the community.

Tool	Project
Puppet	https://forge.puppet.com/puppet/grafana
Ansible	https://github.com/grafana/grafana-ansible-collection
Chef	https://github.com/sous-chefs/chef-grafana
Saltstack	https://github.com/salt-formulas/salt-formula-grafana
Jsonnet	https://github.com/grafana/grafonnet-lib/
NixOS	services.grafana.provision module

Data sources

> **NOTE**
>
> Available in Grafana v5.0 and higher.

You can manage data sources in Grafana by adding YAML configuration files in the `provisioning/datasources` directory. Each config file can contain a list of `datasources` to add or update

during startup. If the data source already exists, Grafana reconfigures it to match the provisioned configuration file.

The configuration file can also list data sources to automatically delete, called `deleteDatasources`. Grafana deletes the data sources listed in `deleteDatasources` *before* adding or updating those in the `datasources` list.

Running multiple Grafana instances

If you run multiple instances of Grafana, add a version number to each data source in the configuration and increase it when you update the configuration. Grafana updates only data sources with the same or lower version number than specified in the config. This prevents old configurations from overwriting newer ones if you have different versions of the `datasource.yaml` file that don't define version numbers, and then restart instances at the same time.

Example data source config file

This example provisions a Graphite data source:

yaml Copy

```yaml
# Configuration file version
apiVersion: 1

# List of data sources to delete from the database.
deleteDatasources:
  - name: Graphite
    orgId: 1

# List of data sources to insert/update depending on what's
# available in the database.
datasources:
  # <string, required> Sets the name you use to refer to
  # the data source in panels and queries.
  - name: Graphite
    # <string, required> Sets the data source type.
    type: graphite
    # <string, required> Sets the access mode, either
    # proxy or direct (Server or Browser in the UI).
    # Some data sources are incompatible with any setting
```

Expand code

For provisioning examples of specific data sources, refer to that data source's documentation.

JSON Data

Since not all data sources have the same configuration settings, we include only the most common ones as fields. To provision the rest of a data source's settings, include them as a JSON blob in the

`jsonData` field.

Common settings in the built-in core data sources include:

> **NOTE**
>
> Data sources tagged with *HTTP** communicate using the HTTP protocol, which includes all core data source plugins except MySQL, PostgreSQL, and MSSQL.

Name	Type	Data source	Description
tlsAuth	boolean	*HTTP**, MySQL	Enable TLS authentication using client cert configured in secure json data
tlsAuthWithCACert	boolean	*HTTP**, MySQL, PostgreSQL	Enable TLS authentication using CA cert
tlsSkipVerify	boolean	*HTTP**, MySQL, PostgreSQL, MSSQL	Controls whether a client verifies the server's certificate chain and host name.
serverName	string	*HTTP**, MSSQL	Optional. Controls the server name used for certificate common name/subject alternative name verification. Defaults to using the data source URL.
timeout	string	*HTTP**	Request timeout in seconds. Overrides dataproxy.timeout option
graphiteVersion	string	Graphite	Graphite version
timeInterval	string	Prometheus, Elasticsearch, InfluxDB, MySQL, PostgreSQL and MSSQL	Lowest interval/step value that should be used for this data source.
httpMode	string	Influxdb	HTTP Method. 'GET', 'POST', defaults to GET
maxSeries	number	Influxdb	Max number of series/tables that Grafana processes
httpMethod	string	Prometheus	HTTP Method. 'GET', 'POST', defaults to POST
customQueryParameters	string	Prometheus	Query parameters to add, as a URL-encoded string.

Name	Type	Data source	Description
manageAlerts	boolean	Prometheus and Loki	Manage alerts via Alerting UI
alertmanagerUid	string	Prometheus and Loki	UID of Alert Manager that manages Alert for this data source.
timeField	string	Elasticsearch	Which field that should be used as timestamp
interval	string	Elasticsearch	Index date time format. nil(No Pattern), 'Hourly', 'Daily', 'Weekly', 'Monthly' or 'Yearly'
logMessageField	string	Elasticsearch	Which field should be used as the log message
logLevelField	string	Elasticsearch	Which field should be used to indicate the priority of the log message
maxConcurrentShardRequests	number	Elasticsearch	Maximum number of concurrent shard requests that each sub-search request executes per node
sigV4Auth	boolean	Elasticsearch and Prometheus	Enable usage of SigV4
sigV4AuthType	string	Elasticsearch and Prometheus	SigV4 auth provider. default/credentials/keys
sigV4ExternalId	string	Elasticsearch and Prometheus	Optional SigV4 External ID
sigV4AssumeRoleArn	string	Elasticsearch and Prometheus	Optional SigV4 ARN role to assume
sigV4Region	string	Elasticsearch and Prometheus	SigV4 AWS region
sigV4Profile	string	Elasticsearch and Prometheus	Optional SigV4 credentials profile
authType	string	Cloudwatch	Auth provider. default/credentials/keys
externalId	string	Cloudwatch	Optional External ID
assumeRoleArn	string	Cloudwatch	Optional ARN role to assume
defaultRegion	string	Cloudwatch	Optional default AWS region
customMetricsNamespaces	string	Cloudwatch	Namespaces of Custom Metrics

Name	Type	Data source	Description
profile	string	Cloudwatch	Optional credentials profile
tsdbVersion	string	OpenTSDB	Version
tsdbResolution	string	OpenTSDB	Resolution
sslmode	string	PostgreSQL	SSLmode. 'disable', 'require', 'verify-ca' or 'verify-full'
tlsConfigurationMethod	string	PostgreSQL	SSL Certificate configuration, either by 'file-path' or 'file-content'
sslRootCertFile	string	PostgreSQL, MSSQL	SSL server root certificate file, must be readable by the Grafana user
sslCertFile	string	PostgreSQL	SSL client certificate file, must be readable by the Grafana user
sslKeyFile	string	PostgreSQL	SSL client key file, must be readable by *only* the Grafana user
encrypt	string	MSSQL	Determines SSL encryption handling. Options include: `disable` - data sent between client and server is not encrypted; `false` - data sent between client and server is not encrypted beyond the login packet; `true` - data sent between client and server is encrypted. Default is `false`.
postgresVersion	number	PostgreSQL	Postgres version as a number (903/904/905/906/1000) meaning v9.3, v9.4, ..., v10
timescaledb	boolean	PostgreSQL	Enable usage of TimescaleDB extension
maxOpenConns	number	MySQL, PostgreSQL and MSSQL	Maximum number of open connections to the database (Grafana v5.4+)
maxIdleConns	number	MySQL, PostgreSQL and MSSQL	Maximum number of connections in the idle connection pool (Grafana v5.4+)
connMaxLifetime	number	MySQL, PostgreSQL and MSSQL	Maximum amount of time in seconds a connection may be reused (Grafana v5.4+)
keepCookies	array	*HTTP**	Cookies that needs to be passed along while communicating with data sources
prometheusVersion	string	Prometheus	The version of the Prometheus data source, such as `2.37.0`,

Name	Type	Data source	Description
			`2.24.0`
prometheusType	string	Prometheus	Prometheus database type. Options are `Prometheus`, `Cortex`, `Mimir` or `Thanos`.
cacheLevel	string	Prometheus	Determines the duration of the browser cache. Valid values include: `Low`, `Medium`, `High`, and `None`. This field is configurable when you enable the `prometheusResourceBrowserCache` feature flag.
incrementalQuerying	string	Prometheus	Experimental: Turn on incremental querying to enhance dashboard reload performance with slow data sources
incrementalQueryOverlapWindow	string	Prometheus	Experimental: Configure incremental query overlap window. Requires a valid duration string, i.e. `180s` or `15m` Default value is `10m` (10 minutes).
disableRecordingRules	boolean	Prometheus	Experimental: Turn off Prometheus recording rules
implementation	string	AlertManager	The implementation of the AlertManager data source, such as `prometheus`, `cortex` or `mimir`
handleGrafanaManagedAlerts	boolean	AlertManager	When enabled, Grafana-managed alerts are sent to this Alertmanager

For examples of specific data sources' JSON data, refer to that data source's documentation.

Secure JSON Data

Secure JSON data is a map of settings that will be encrypted with secret key from the Grafana config. The purpose of this is only to hide content from the users of the application. This should be used for storing TLS Cert and password that Grafana will append to the request on the server side. All of these settings are optional.

> **NOTE**
>
> The *HTTP** tag denotes data sources that communicate using the HTTP protocol, including all core data source plugins except MySQL, PostgreSQL, and MSSQL.

Name	Type	Data source	Description
tlsCACert	string	*HTTP**, MySQL, PostgreSQL	CA cert for out going requests
tlsClientCert	string	*HTTP**, MySQL, PostgreSQL	TLS Client cert for outgoing requests
tlsClientKey	string	*HTTP**, MySQL, PostgreSQL	TLS Client key for outgoing requests
password	string	*HTTP**, MySQL, PostgreSQL, MSSQL	password
basicAuthPassword	string	*HTTP**	password for basic authentication
accessKey	string	Cloudwatch	Access key for connecting to Cloudwatch
secretKey	string	Cloudwatch	Secret key for connecting to Cloudwatch
sigV4AccessKey	string	Elasticsearch and Prometheus	SigV4 access key. Required when using keys auth provider
sigV4SecretKey	string	Elasticsearch and Prometheus	SigV4 secret key. Required when using keys auth provider

Custom HTTP headers for data sources

Data sources managed by Grafanas provisioning can be configured to add HTTP headers to all requests going to that data source. The header name is configured in the `jsonData` field and the header value should be configured in `secureJsonData`.

yaml Copy

```yaml
apiVersion: 1

datasources:
  - name: Graphite
    jsonData:
      httpHeaderName1: 'HeaderName'
      httpHeaderName2: 'Authorization'
    secureJsonData:
      httpHeaderValue1: 'HeaderValue'
      httpHeaderValue2: 'Bearer XXXXXXXXX'
```

Plugins

NOTE

Available in Grafana v7.1 and higher.

You can manage plugin applications in Grafana by adding one or more YAML config files in the `provisioning/plugins` directory. Each config file can contain a list of `apps` that will be updated during start up. Grafana updates each app to match the configuration file.

NOTE

This feature enables you to provision plugin configurations, not the plugins themselves. The plugins must already be installed on the Grafana instance.

Example plugin configuration file

yaml Copy

```yaml
apiVersion: 1

apps:
  # <string> the type of app, plugin identifier. Required
  - type: raintank-worldping-app
    # <int> Org ID. Default to 1, unless org_name is specified
    org_id: 1
    # <string> Org name. Overrides org_id unless org_id not specified
    org_name: Main Org.
    # <bool> disable the app. Default to false.
    disabled: false
    # <map> fields that will be converted to json and stored in jsonData. Custom per app.
    jsonData:
      # key/value pairs of string to object
      key: value
    # <map> fields that will be converted to json, encrypted and stored in secureJsonData. Custom
    secureJsonData:
      # key/value pairs of string to string
      key: value
```

Expand code

Dashboards

You can manage dashboards in Grafana by adding one or more YAML config files in the `provisioning/dashboards` directory. Each config file can contain a list of `dashboards providers` that load dashboards into Grafana from the local filesystem.

The dashboard provider config file looks somewhat like this:

yaml Copy

```yaml
apiVersion: 1

providers:
  # <string> an unique provider name. Required
  - name: 'a unique provider name'
    # <int> Org id. Default to 1
    orgId: 1
    # <string> name of the dashboard folder.
    folder: ''
    # <string> folder UID. will be automatically generated if not specified
    folderUid: ''
    # <string> provider type. Default to 'file'
    type: file
    # <bool> disable dashboard deletion
    disableDeletion: false
    # <int> how often Grafana will scan for changed dashboards
    updateIntervalSeconds: 10
    # <bool> allow updating provisioned dashboards from the UI
    allowUiUpdates: false
```

Expand code

When Grafana starts, it will update/insert all dashboards available in the configured path. Then later on poll that path every **updateIntervalSeconds** and look for updated json files and update/insert those into the database.

Note: Dashboards are provisioned to the root level if the `folder` option is missing or empty.

Making changes to a provisioned dashboard

It's possible to make changes to a provisioned dashboard in the Grafana UI. However, it is not possible to automatically save the changes back to the provisioning source. If `allowUiUpdates` is set to `true` and you make changes to a provisioned dashboard, you can `Save` the dashboard then changes will be persisted to the Grafana database.

Note: If a provisioned dashboard is saved from the UI and then later updated from the source, the dashboard stored in the database will always be overwritten. The `version` property in the JSON file will not affect this, even if it is lower than the existing dashboard.

If a provisioned dashboard is saved from the UI and the source is removed, the dashboard stored in the database will be deleted unless the configuration option `disableDeletion` is set to

true.

If `allowUiUpdates` is configured to `false`, you are not able to make changes to a provisioned dashboard. When you click `Save`, Grafana brings up a *Cannot save provisioned dashboard* dialog. The screenshot below illustrates this behavior.

Grafana offers options to export the JSON definition of a dashboard. Either `Copy JSON to Clipboard` or `Save JSON to file` can help you synchronize your dashboard changes back to the provisioning source.

Note: The JSON definition in the input field when using `Copy JSON to Clipboard` or `Save JSON to file` will have the `id` field automatically removed to aid the provisioning workflow.

Reusable Dashboard URLs

If the dashboard in the JSON file contains an UID, Grafana forces insert/update on that UID. This allows you to migrate dashboards between Grafana instances and provisioning Grafana from configuration without breaking the URLs given because the new dashboard URL uses the UID as identifier. When Grafana starts, it updates/inserts all dashboards available in the configured folders. If you modify the file, then the dashboard is also updated. By default, Grafana deletes dashboards in the database if the file is removed. You can disable this behavior using the `disableDeletion` setting.

> **NOTE**
>
> Provisioning allows you to overwrite existing dashboards which leads to problems if you reuse settings that are supposed to be unique. Be careful not to reuse the same `title` multiple times within a folder or `uid` within the same installation as this will cause weird behaviors.

Provision folders structure from filesystem to Grafana

If you already store your dashboards using folders in a git repo or on a filesystem, and also you want to have the same folder names in the Grafana menu, you can use `foldersFromFilesStructure` option.

For example, to replicate these dashboards structure from the filesystem to Grafana,

```
/etc/dashboards
├── /server
│   ├── /common_dashboard.json
│   └── /network_dashboard.json
└── /application
    ├── /requests_dashboard.json
    └── /resources_dashboard.json
```

Copy

you need to specify just this short provision configuration file.

yaml

Copy

```yaml
apiVersion: 1

providers:
  - name: dashboards
    type: file
    updateIntervalSeconds: 30
    options:
      path: /etc/dashboards
      foldersFromFilesStructure: true
```

`server` and `application` will become new folders in Grafana menu.

NOTE

`folder` and `folderUid` options should be empty or missing to make `foldersFromFilesStructure` work.

NOTE

To provision dashboards to the root level, store them in the root of your `path`.

NOTE

This feature doesn't currently allow you to create nested folder structures, that is, where you have folders within folders.

Alerting

For information on provisioning Grafana Alerting, refer to Provision Grafana Alerting resources.

Alert Notification Channels

> **NOTE**
>
> Alert Notification Channels are part of legacy alerting, which is deprecated and will be removed in Grafana 10. Use the Provision contact points section in Create and manage alerting resources using file provisioning.

Alert Notification Channels can be provisioned by adding one or more YAML config files in the `provisioning/notifiers` directory.

Each config file can contain the following top-level fields:

- `notifiers`, a list of alert notifications that will be added or updated during start up. If the notification channel already exists, Grafana will update it to match the configuration file.
- `delete_notifiers`, a list of alert notifications to be deleted before inserting/updating those in the `notifiers` list.

Provisioning looks up alert notifications by uid, and will update any existing notification with the provided uid.

By default, exporting a dashboard as JSON will use a sequential identifier to refer to alert notifications. The field `uid` can be optionally specified to specify a string identifier for the alert name.

json Copy

```
{
  ...
      "alert": {
        ...,
        "conditions": [...],
        "frequency": "24h",
        "noDataState": "ok",
        "notifications": [
           {"uid": "notifier1"},
           {"uid": "notifier2"},
        ]
      }
```

```
  ...
}
```

Example Alert Notification Channels Config File

yaml

Copy

```yaml
notifiers:
  - name: notification-channel-1
    type: slack
    uid: notifier1
    # either
    org_id: 2
    # or
    org_name: Main Org.
    is_default: true
    send_reminder: true
    frequency: 1h
    disable_resolve_message: false
    # See `Supported Settings` section for settings supported for each
    # alert notification type.
    settings:
      recipient: 'XXX'
      uploadImage: true
      token: 'xoxb' # legacy setting since Grafana v7.2 (stored non-encrypt
      url: https://slack.com # legacy setting since Grafana v7.2 (stored non-encrypted)
```

Expand code

Supported Settings

The following sections detail the supported settings and secure settings for each alert notification type. Secure settings are stored encrypted in the database and you add them to `secure_settings` in the YAML file instead of `settings`.

> **NOTE**
>
> Secure settings is supported since Grafana v7.2.

Alert notification `pushover`

Name	Secure setting
apiToken	yes
userKey	yes
device	
priority	
okPriority	
retry	
expire	
sound	
okSound	

Alert notification `discord`

Name	Secure setting
url	yes
avatar_url	
content	
use_discord_username	

Alert notification `slack`

Name	Secure setting
url	yes
recipient	
username	
icon_emoji	
icon_url	
uploadImage	
mentionUsers	
mentionGroups	

Name	Secure setting
mentionChannel	
token	yes

Alert notification `victorops`

Name
url
autoResolve

Alert notification `kafka`

Name
kafkaRestProxy
kafkaTopic

Alert notification `LINE`

Name	Secure setting
token	yes

Alert notification `pagerduty`

Name	Secure setting
integrationKey	yes
autoResolve	

Alert notification `sensu`

Name	Secure setting
url	
source	
handler	
username	
password	yes

Alert notification `sensugo`

Name	Secure setting
url	
apikey	yes
entity	
check	
handler	
namespace	

Alert notification `prometheus-alertmanager`

Name	Secure setting
url	
basicAuthUser	
basicAuthPassword	yes

Alert notification `teams`

Name
url

Alert notification `dingding`

Name
url

Alert notification `email`

Name
singleEmail
addresses

Alert notification `hipchat`

Name
url
apikey
roomid

Alert notification `opsgenie`

Name	Secure setting
apiKey	yes
apiUrl	
autoClose	
overridePriority	
sendTagsAs	

Alert notification `telegram`

Name	Secure setting
bottoken	yes
chatid	
uploadImage	

Alert notification `threema`

Name	Secure setting
gateway_id	
recipient_id	
api_secret	yes

Alert notification `webhook`

Name	Secure setting
url	
httpMethod	
username	
password	yes

Alert notification `googlechat`

Name
url

Alert notification `Cisco Webex Teams`

Name	Secure setting
message	
room_id	
api_url	
bot_token	yes

Grafana Enterprise

Grafana Enterprise supports:

- Provisioning role-based access control with Grafana
- Provisioning role-based access control with Terraform

What's new in Grafana v10.4

Welcome to Grafana 10.4! This minor release contains some notable improvements in its own right, as well as early previews of functionality we intend to turn on by default in Grafana v11. Read on to learn about a quicker way to set up alert notifications, an all-new UI for configuring single sign-on, and improvements to our Canvas, Geomap, and Table panels.

For even more detail about all the changes in this release, refer to the changelog. For the specific steps we recommend when you upgrade to v10.4, check out our Upgrade Guide.

Dashboards and visualizations

AngularJS plugin warnings in dashboards

Generally available in all editions of Grafana

AngularJS support in Grafana was deprecated in v9 and will be turned off by default in Grafana v11. When this happens, any plugin which depended on AngularJS will not load, and dashboard panels will be unable to show data.

To help you understand where you may be impacted, Grafana now displays a warning banner in any dashboard with a dependency on an AngularJS plugin. Additionally, warning icons are present in any panel where the panel plugin or underlying data source plugin has an AngularJS dependency.

This complements the existing warnings already present on the **Plugins** page under the administration menu.

In addition, you can use our detect-angular-dashboards open source tool, which can be run against any Grafana instance to generate a report listing all dashboards that have a dependency on an AngularJS plugin, as well as which plugins are in use. This tool also supports the detection of private plugins that are dependent on AngularJS, however this particular feature requires Grafana v10.1.0 or higher.

Use the aforementioned tooling and warnings to plan migrations to React based visualizations and data sources included in Grafana or from the Grafana plugins catalog.

To learn more, refer to the Angular support deprecation, which includes recommended alternative plugins.

Documentation

Data visualization quality of life improvements

Generally available in all editions of Grafana

We've made a number of small improvements to the data visualization experience in Grafana.

Geomap geojson layer now supports styling

You can now visualize geojson styles such as polygons, point color/size, and line strings. To learn more, refer to the documentation.

Canvas elements now support snapping and aligning

You can precisely place elements in a canvas with ease as elements now snap into place and align with one another.

Canvas element snapping and alignment

View data links inline in table visualizations

You can now view your data links inline to help you keep your tables visually streamlined.

Create subtables in table visualizations with Group to nested tables

Available in public preview in all editions of Grafana

You can now create subtables out of your data using the new **Group to nested tables** transformation. To use this feature, enable the `groupToNestedTableTransformation` feature toggle.

Group to nested tables transformation

Set library panel permissions with RBAC

Generally available in Grafana Enterprise and Grafana Cloud

We've added the option to manage library panel permissions through role-based access control (RBAC). With this feature, you can choose who can create, edit, and read library panels. RBAC provides a standardized way of granting, changing, and revoking access when it comes to viewing and modifying Grafana resources, such as dashboards, reports, and administrative settings.

Documentation

Tooltip improvements

Available in public preview in all editions of Grafana

We've made a number of small improvements to the way tooltips work in Grafana. To try out the new tooltips, enable the `newVizTooltips` feature toggle.

Copy on click support

You can now copy the content from within a tooltip by clicking on the text.

Scrollable content

You can now scroll the content of a tooltip, which allows you to view long lists. This is currently supported in the time series, candlestick, and trend visualizations. We'll add more improvements to the scrolling functionality in a future version.

Added tooltip options for candlestick visualization

The default tooltip options are now also visible in candlestick visualizations.

Hover proximity option in time series

We've added a tooltip hover proximity limit option (in pixels), which makes it possible to reduce the number of hovered-over data points under the cursor when two datasets are not aligned in time.

Return to previous

Available in public preview in all editions of Grafana

When you're browsing Grafana - for example, exploring the dashboard and metrics related to an alert - it's easy to end up far from where you started and hard get back to where you came from.

The ‘Return to previous’ button is an easy way to go back to the previous context, like the alert rule that kicked off your exploration. This first release works for Alerts, and we plan to expand to other apps and features in Grafana in future releases to make it easier to navigate around.

Return to Previous is rolling out across Grafana Cloud now. To try Return to Previous in self-managed Grafana, turn on the `returnToPrevious` feature toggle in Grafana v10.4 or newer.

> **NOTE**
>
> The term **context** refers to applications in Grafana like Incident and OnCall, as well as core features like Explore and Dashboards.
>
> To notice a change in your context, look at Grafana’s breadcrumbs. If you go from *Home* > ***Dashboards*** to *Home* > ***Explore***, you’ve changed context. If you go from *Home* > ***Dashboards*** > *Playlist* > *Edit playlist* to *Home* > ***Dashboards*** > *Reporting* > *Settings*, you are in the same context.

Alerting

Simplified Alert Notification Routing

Generally available in all editions of Grafana

This feature simplifies your options for configuring where your notifications are sent when an alert rule fires. Choose an existing contact point directly from within the alert rule creation form without the need to label match notification policies. You can also set optional muting, grouping, and timing settings directly in the alert rule.

Simplified routing inherits the alert rule RBAC, increasing control over notification routing while preventing accidental notification policy updates, ensuring critical notifications make it to their intended contact point destination.

To try out Simplified Alert Notification Routing enable the `alertingSimplifiedRouting` feature toggle.

Grafana Alerting upgrade with rule preview

Generally available in all editions of Grafana

Users looking to migrate to the new Grafana Alerting product can do so with confidence with the Grafana Alerting migration preview tool. The migration preview tool allows users to view, edit, and delete migrated rules prior cutting over, with the option to roll back to Legacy Alerting.

Documentation

Rule evaluation spread over the entire evaluation interval

Generally available in all editions of Grafana

Grafana Alerting previously evaluated rules at the start of the evaluation interval. This created a sudden spike of resource utilization, impacting data sources. Rule evaluation is now spread over the entire interval for smoother performance utilization of data sources.

UTF-8 Support for Prometheus and Mimir Alertmanagers

Generally available in all editions of Grafana

Grafana can now be used to manage both Prometheus and Mimir Alertmanagers with UTF-8 configurations. For more information, please see the release notes for Alertmanager 0.27.0.

Authentication and authorization

SSO Settings UI and Terraform resource for configuring OAuth providers

Available in public preview in all editions of Grafana

Configuring OAuth providers was a bit cumbersome in Grafana: Grafana Cloud users had to reach out to Grafana Support, self-hosted users had to manually edit the configuration file, set up environment variables, and then they had to restart Grafana. On Cloud, the Advanced Auth page is there to configure some of the providers, but configuring Generic OAuth hasn't been available until now and there was no way to manage the settings through the Grafana UI, nor was there a way to manage the settings through Terraform or the Grafana API.

Our goal is to make setting up SSO for your Grafana instance simple and fast.

To get there, we are introducing easier self-serve configuration options for OAuth in Grafana. All of the currently supported OAuth providers are now available for configuration through the Grafana UI, Terraform and via the API. From the UI, you can also now manage all of the settings for the Generic OAuth provider.

We are working on adding complete support for configuring all other supported OAuth providers as well, such as GitHub, GitLab, Google, Microsoft Azure AD and Okta. You can already manage some of these settings via the new self-serve configuration options, and we're working on adding more at the moment.

Documentation

Data sources

> **NOTE**
>
> The following data sources are released separately from Grafana itself. They are included here for extra visibility.

PagerDuty enterprise data source for Grafana

Generally available in Grafana Enterprise and Grafana Cloud

PagerDuty enterprise data source plugin for Grafana allows you to query incidents data or visualize incidents using annotations.

> **NOTE**
>
> Plugin is currently in a preview phase.

You can find more information and how to configure the plugin in the documentation.

Screenshots:

PagerDuty data source annotation editor

Incidents annotations from PagerDuty data source on a dashboard panel

SurrealDB Data Source

Experimental in all editions of Grafana

A SurrealDB data source has been added to the Plugin Catalog, enabling the integration of SurrealDB, a real-time, multi-model database, with Grafana's visualization capabilities. This datasource allows users to directly query and visualize data from SurrealDB within Grafana, using SurrealDB's query language.

The SurrealDB data source launches with just the basics today. You can write queries in SurrealQL using the built-in query editor, although many Grafana features like macros are not supported for now.

You can find more information and how to configure the plugin on Github.

Documentation

Table Visualization for Logs

Generally available in all editions of Grafana

The table visualization for logs, announced in public preview for Grafana 10.3, is generally available in Cloud (all editions) and with Grafana 10.4.

New to the table visualization with 10.4:

- the ability to sort columns
- data type autodetection of fields
- autodetection and clean formatting of json fields

Try it out today!

Graphite data source

Grafana includes built-in support for Graphite. This topic explains options, variables, querying, and other features specific to the Graphite data source, which include its feature-rich query editor.

For instructions on how to add a data source to Grafana, refer to the administration documentation. Only users with the organization administrator role can add data sources.

Once you've added the Graphite data source, you can configure it so that your Grafana instance's users can create queries in its query editor when they build dashboards and use Explore.

Give it a try using Grafana Play

With Grafana Play, you can explore and see how it works, learning from practical examples to accelerate your development. This feature can be seen on Graphite: Sample Website Dashboard.

Configure the data source

To configure basic settings for the data source, complete the following steps:

1. Click **Connections** in the left-side menu.
2. Under Your connections, click **Data sources**.
3. Enter `Graphite` in the search bar.
4. Click **Graphite**.

   The **Settings** tab of the data source is displayed.
5. Set the data source's basic configuration options:

Name	Description
Name	Sets the name you use to refer to the data source in panels and queries.
Default	Sets whether the data source is pre-selected for new panels. You can set only one default data source per organization.
URL	Sets the HTTP protocol, IP, and port of your graphite-web or graphite-api installation.
Auth	For details, refer to Configure Authentication.
Basic Auth	Enables basic authentication to the data source.
User	Sets the user name for basic authentication.
Password	Sets the password for basic authentication.
Custom HTTP Headers	Click **Add header** to add a custom HTTP header.
Header	Defines the custom header name.
Value	Defines the custom header value.

You can also configure settings specific to the Graphite data source:

Name	Description
Version	Select your version of Graphite. If you are using Grafana Cloud Graphite, this should be set to `1.1.x`.
Type	Select your type of Graphite. If you are using Grafana Cloud Graphite, this should be set to `Default`.

Integrate with Loki

When you change the data source selection in Explore, Graphite queries are converted to Loki queries. Grafana extracts Loki label names and values from the Graphite queries according to mappings provided in the Graphite data source configuration. Queries using tags with `seriesByTags()` are also transformed without any additional setup.

Provision the data source

You can define and configure the data source in YAML files as part of Grafana's provisioning system. For more information about provisioning, and for lists of common configuration options and JSON data options, refer to Provisioning data sources.

Provisioning example

yaml Copy

```yaml
apiVersion: 1

datasources:
  - name: Graphite
    type: graphite
    access: proxy
    url: http://localhost:8080
    jsonData:
      graphiteVersion: '1.1'
```

Query the data source

Grafana includes a Graphite-specific query editor to help you build queries. The query editor helps you quickly navigate the metric space, add functions, and change function parameters. It can handle all types of Graphite queries, including complex nested queries through the use of query references.

For details, refer to the query editor documentation.

Use template variables

Instead of hard-coding details such as server, application, and sensor names in metric queries, you can use variables. Grafana lists these variables in dropdown select boxes at the top of the dashboard to help you change the data displayed in your dashboard. Grafana refers to such variables as template variables.

For details, see the template variables documentation.

Get Grafana metrics into Graphite

Grafana exposes metrics for Graphite on the `/metrics` endpoint. For detailed instructions, refer to Internal Grafana metrics.

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Elasticsearch data source

Elasticsearch is a search and analytics engine used for a variety of use cases. You can create many types of queries to visualize logs or metrics stored in Elasticsearch, and annotate graphs with log events stored in Elasticsearch.

The following will help you get started working with Elasticsearch and Grafana:

- What is Elasticsearch?
- Configure the Elasticsearch data source
- Elasticsearch query editor
- Elasticsearch template variables

Supported Elasticsearch versions

This data source supports these versions of Elasticsearch:

- v7.16+
- v8.x

Our maintenance policy for Elasticsearch data source is aligned with the Elastic Product End of Life Dates and we ensure proper functionality for supported versions. If you are using an Elasticsearch with version that is past its end-of-life (EOL), you can still execute queries, but you will receive a notification in the query builder indicating that the version of Elasticsearch you are using is no longer supported. It's important to note that in such cases, we do not guarantee the correctness of the functionality, and we will not be addressing any related issues.

Provision the data source

You can define and configure the data source in YAML files as part of Grafana's provisioning system. For more information about provisioning, and for available configuration options, refer to

Provisioning Grafana.

> **NOTE**
>
> The previously used `database` field has now been deprecated. You should now use the `index` field in `jsonData` to store the index name. Please see the examples below.

Provisioning examples

Basic provisioning

yaml Copy

```yaml
apiVersion: 1

datasources:
  - name: Elastic
    type: elasticsearch
    access: proxy
    url: http://localhost:9200
    jsonData:
      index: '[metrics-]YYYY.MM.DD'
      interval: Daily
      timeField: '@timestamp'
```

Provision for logs

yaml Copy

```yaml
apiVersion: 1

datasources:
  - name: elasticsearch-v7-filebeat
    type: elasticsearch
    access: proxy
    url: http://localhost:9200
    jsonData:
      index: '[filebeat-]YYYY.MM.DD'
      interval: Daily
      timeField: '@timestamp'
      logMessageField: message
      logLevelField: fields.level
      dataLinks:
```

```
      - datasourceUid: my_jaeger_uid # Target UID needs to be known
        field: traceID
        url: '$${__value.raw}' # Careful about the double "$$" because of env var expansion
```

Configure Amazon Elasticsearch Service

If you use Amazon Elasticsearch Service, you can use Grafana's Elasticsearch data source to visualize data from it.

If you use an AWS Identity and Access Management (IAM) policy to control access to your Amazon Elasticsearch Service domain, you must use AWS Signature Version 4 (AWS SigV4) to sign all requests to that domain.

For details on AWS SigV4, refer to the AWS documentation.

AWS Signature Version 4 authentication

> **NOTE**
>
> Available in Grafana v7.3 and higher.

To sign requests to your Amazon Elasticsearch Service domain, you can enable SigV4 in Grafana's configuration.

Once AWS SigV4 is enabled, you can configure it on the Elasticsearch data source configuration page. For more information about AWS authentication options, refer to AWS authentication.

SigV4 configuration for AWS Elasticsearch Service

Query the data source

You can select multiple metrics and group by multiple terms or filters when using the Elasticsearch query editor.

For details, see the query editor documentation.

Use template variables

Instead of hard-coding details such as server, application, and sensor names in metric queries, you can use variables. Grafana lists these variables in dropdown select boxes at the top of the dashboard to help you change the data displayed in your dashboard. Grafana refers to such variables as template variables.

For details, see the template variables documentation.

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

InfluxDB data source

InfluxDB is an open-source time series database (TSDB) developed by InfluxData. It is optimized for fast, high-availability storage and retrieval of time series data in fields such as operations monitoring, application metrics, IoT sensor data, and real-time analytics.

Grafana includes built-in support for InfluxDB. This topic explains options, variables, querying, and other features specific to the InfluxDB data source, which include its feature-rich code editor for queries and visual query builder.

For instructions on how to add a data source to Grafana, refer to the administration documentation. Only users with the organization administrator role can add data sources. Administrators can also configure the data source via YAML with Grafana's provisioning system.

Once you've added the InfluxDB data source, you can configure it so that your Grafana instance's users can create queries in its query editor when they build dashboards and use Explore.

Configure the data source

To configure basic settings for the data source, complete the following steps:

1. Click **Connections** in the left-side menu.
2. Under Your connections, click **Data sources**.
3. Enter `InfluxDB` in the search bar.
4. Select **InfluxDB**.

   The **Settings** tab of the data source is displayed.
5. Set the data source's basic configuration options carefully:

Name	Description
Name	Sets the name you use to refer to the data source in panels and queries. We recommend something like `InfluxDB-InfluxQL`.
Default	Sets whether the data source is pre-selected for new panels.
URL	The HTTP protocol, IP address, and port of your InfluxDB API. InfluxDB's default API port is 8086.
Min time interval	*(Optional)* Refer to Min time interval.
Max series	*(Optional)* Limits the number of series and tables that Grafana processes. Lower this number to prevent abuse, and increase it if you have many small time series and not all are shown. Defaults to 1,000.

You can also configure settings specific to the InfluxDB data source. These options are described in the sections below.

Min time interval

The **Min time interval** setting defines a lower limit for the auto group-by time interval.

This value *must* be formatted as a number followed by a valid time identifier:

Identifier	Description
`y`	year
`M`	month
`w`	week
`d`	day
`h`	hour
`m`	minute
`s`	second
`ms`	millisecond

We recommend setting this value to match your InfluxDB write frequency. For example, use `1m` if InfluxDB writes data every minute.

You can also override this setting in a dashboard panel under its data source options.

Select a query language

InfluxDB data source options differ depending on which query language you select:

- InfluxQL, a SQL-like language for querying InfluxDB, with statements such as SELECT, FROM, WHERE, and GROUP BY that are familiar to SQL users. InfluxQL is available in InfluxDB 1.0 onwards.
- SQL native SQL language with support FlightSQL.
- Flux, which provides significantly broader functionality than InfluxQL. It supports not only queries but also built-in functions for data shaping, string manipulation, and joining to non-InfluxDB data sources, but also processing time-series data. It's similar to JavaScript with a functional style.

To help choose the best language for your needs, refer to a comparison of Flux vs InfluxQL and why InfluxData created Flux.

> **NOTE**
>
> Though not required, we recommend that you append your query language choice to the data source's **Name** setting:
>
> - InfluxDB-InfluxQL
> - InfluxDB-SQL
> - InfluxDB-Flux

Configure InfluxQL

Configure these options if you select the InfluxQL (classic InfluxDB) query language:

Name	Description
Allowed cookies	Defines which cookies are forwarded to the data source. All other cookies are deleted.
Database	Sets the ID of the bucket to query. Copy this from the Buckets page of the InfluxDB UI.
User	Sets the username to sign into InfluxDB.
Password	Defines the token you use to query the bucket defined in **Database**. Copy this from the Tokens page of the InfluxDB UI.
HTTP mode	Sets the HTTP method used to query your data source. The POST verb allows for larger queries that would return an error using the GET verb. Defaults to GET.

Configure SQL

Configure these options if you select the SQL query language:

Name	Description
Database	Sets the ID of the bucket to query. Copy this from the Buckets page of the InfluxDB UI.
Token	API token used for SQL queries. It can be generated on InfluxDB Cloud dashboard under Load Data > API Tokens menu.

Configure Flux

Configure these options if you select the Flux query language:

Name	Description
Organization	The Influx organization that will be used for Flux queries. This is also used to for the `v.organization` query macro.
Token	The authentication token used for Flux queries. With Influx 2.0, use the influx authentication token to function. Token must be set as `Authorization` header with the value `Token <geenrated-token>`. For influx 1.8, the token is `username:password`.
Default bucket	*(Optional)* The Influx bucket that will be used for the `v.defaultBucket` macro in Flux queries.

Provision the data source

You can define and configure the data source in YAML files as part of Grafana's provisioning system. For more information about provisioning, and for available configuration options, refer to [Provisioning Grafana][provisioning-data-sources].

> **NOTE**
>
> `database` field is deprecated. We suggest to use `dbName` field in `jsonData`. Please see the examples below. No need to change existing provisioning settings.

Provisioning examples

InfluxDB 1.x example:

yaml Copy

```yaml
apiVersion: 1

datasources:
  - name: InfluxDB_v1
```

```
    type: influxdb
    access: proxy
    user: grafana
    url: http://localhost:8086
    jsonData:
      dbName: site
      httpMode: GET
    secureJsonData:
      password: grafana
```

InfluxDB 2.x for Flux example:

yaml Copy

```yaml
apiVersion: 1

datasources:
  - name: InfluxDB_v2_Flux
    type: influxdb
    access: proxy
    url: http://localhost:8086
    jsonData:
      version: Flux
      organization: organization
      defaultBucket: bucket
      tlsSkipVerify: true
    secureJsonData:
      token: token
```

InfluxDB 2.x for InfluxQL example:

yaml Copy

```yaml
apiVersion: 1

datasources:
  - name: InfluxDB_v2_InfluxQL
    type: influxdb
    access: proxy
    url: http://localhost:8086
    jsonData:
      version: SQL
      metadata:
        - database: <bucket-name>
```

```
    secureJsonData:
      httpHeaderValue1: 'Token <token>'
```

InfluxDB 3.x for SQL example:

yaml Copy

```yaml
apiVersion: 1

datasources:
  - name: InfluxDB_v2_InfluxQL
    type: influxdb
    access: proxy
    url: http://localhost:8086
    jsonData:
      dbName: site
      httpMode: POST
    secureJsonData:
      token: '<api-token>'
```

Query the data source

The InfluxDB data source's query editor has two modes, InfluxQL and Flux, depending on your choice of query language in the data source configuration:

For details, refer to the query editor documentation.

Use template variables

Instead of hard-coding details such as server, application, and sensor names in metric queries, you can use variables. Grafana lists these variables in dropdown select boxes at the top of the dashboard to help you change the data displayed in your dashboard. Grafana refers to such variables as template variables.

For details, see the template variables documentation.

Prometheus data source

Prometheus is an open-source database that uses a telemetry collector agent to scrape and store metrics used for monitoring and alerting. If you are just getting started with Prometheus, see What is Prometheus?.

Grafana provides native support for Prometheus. For instructions on downloading Prometheus see Get started with Grafana and Prometheus.

For instructions on how to add a data source to Grafana, refer to the administration documentation. Only users with the organization `administrator` role can add data sources and edit existing data sources. Administrators can also configure the data source via YAML with Grafana's provisioning system.

Once you've added the Prometheus data source, you can configure it so that your Grafana instance's users can create queries in its query editor when they build dashboards, use Explore, and [annotate visualizations][annotate visualizations].

The following guides will help you get started with the Prometheus data source:

- Configure the Prometheus data source
- Prometheus query editor
- Template variables

Prometheus API

The Prometheus data source also works with other projects that implement the Prometheus querying API.

For more information on how to query other Prometheus-compatible projects from Grafana, refer to the specific project's documentation:

- Grafana Mimir

- Thanos

Provision the data source

You can define and configure the data source in YAML files as part of Grafana's provisioning system. For more information about provisioning, and for available configuration options, refer to Provisioning Grafana.

> **NOTE**
>
> Once you have provisioned a data source you cannot edit it.

Provisioning example

yaml Copy

```yaml
apiVersion: 1

datasources:
  - name: Prometheus
    type: prometheus
    access: proxy
    # Access mode - proxy (server in the UI) or direct (browser in the UI).
    url: http://localhost:9090
    jsonData:
      httpMethod: POST
      manageAlerts: true
      prometheusType: Prometheus
      prometheusVersion: 2.44.0
      cacheLevel: 'High'
      disableRecordingRules: false
      incrementalQueryOverlapWindow: 10m
      exemplarTraceIdDestinations:
        # Field with internal link pointing to data source in Grafana.
        # datasourceUid value can be anything, but it should be unique across all defined data so
```

Expand code

View Grafana metrics with Prometheus

Grafana exposes metrics for Prometheus on the `/metrics` endpoint. We also bundle a dashboard within Grafana so you can start viewing your metrics faster.

To import the bundled dashboard:

1. Navigate to the data source's configuration page.
2. Select the **Dashboards** tab.

This displays dashboards for Grafana and Prometheus.

1. Select **Import** for the dashboard to import.

For details about these metrics, refer to Internal Grafana metrics.

Amazon Managed Service for Prometheus

The Prometheus data source works with Amazon Managed Service for Prometheus.

If you use an AWS Identity and Access Management (IAM) policy to control access to your Amazon Elasticsearch Service domain, you must use AWS Signature Version 4 (AWS SigV4) to sign all requests to that domain.

For details on AWS SigV4, refer to the AWS documentation.

AWS Signature Version 4 authentication

> **NOTE**
>
> Available in Grafana v7.3.5 and higher.

To connect the Prometheus data source to Amazon Managed Service for Prometheus using SigV4 authentication, refer to the AWS guide to Set up Grafana open source or Grafana Enterprise for use with AMP.

If you run Grafana in an Amazon EKS cluster, follow the AWS guide to Query using Grafana running in an Amazon EKS cluster.

Azure authentication settings

The Prometheus data source works with Azure authentication. To configure Azure authentication see Configure Azure Active Directory (AD) authentication.

In Grafana Enterprise, update the .ini configuration file: Configure Grafana. Depending on your setup, the .ini file is located here. Add the following setting in the **[auth]** section :

bash Copy

```
[auth]
azure_auth_enabled = true
```

> **NOTE**

If you are using Azure authentication settings do not enable `Forward OAuth identity`. Both use the same HTTP authorization headers. Azure settings will get overwritten by the Oauth token.

Exemplars

Exemplars associate higher-cardinality metadata from a specific event with traditional time series data. See Introduction to exemplars in Prometheus documentation for detailed information on how they work.

> **NOTE**
>
> Available in Prometheus v2.26 and higher with Grafana v7.4 and higher.

Grafana 7.4 and higher can show exemplars data alongside a metric both in Explore and in Dashboards.

Screenshot showing the detail window of an Exemplar

See the Exemplars section in Configure Prometheus data source.

Exemplars

Incremental dashboard queries (beta)

As of Grafana 10, the Prometheus data source can be configured to query live dashboards incrementally, instead of re-querying the entire duration on each dashboard refresh.

This can be toggled on or off in the data source configuration or provisioning file (under `incrementalQuerying` in jsonData). Additionally, the amount of overlap between incremental queries can be configured using the `incrementalQueryOverlapWindow` jsonData field, the default value is `10m` (10 minutes).

Increasing the duration of the `incrementalQueryOverlapWindow` will increase the size of every incremental query, but might be helpful for instances that have inconsistent results for recent data.

Recording Rules (beta)

The Prometheus data source can be configured to disable recording rules under the data source configuration or provisioning file (under `disableRecordingRules` in jsonData).

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Google Cloud Monitoring data source

Grafana ships with built-in support for Google Cloud Monitoring. This topic describes queries, templates, variables, and other configuration specific to the Google Cloud Monitoring data source.

NOTE

Before Grafana v7.1, Google Cloud Monitoring was referred to as Google Stackdriver.

For instructions on how to add a data source to Grafana, refer to the administration documentation. Only users with the organization administrator role can add data sources.

Once you've added the Google Cloud Monitoring data source, you can configure it so that your Grafana instance's users can create queries in its query editor and apply annotations when they build dashboards and use Explore.

Configure the data source

To configure basic settings for the data source, complete the following steps:

1. Click **Connections** in the left-side menu.
2. Under Your connections, click **Data sources**.
3. Enter `Google Cloud Monitoring` in the search bar.
4. Click **Google Cloud Monitoring**.

   The **Settings** tab of the data source is displayed.
5. Set the data source's basic configuration options:

Name	Description
Name	Sets the name you use to refer to the data source in panels and queries.
Default	Sets whether the data source is pre-selected for new panels.

Configure Google authentication

Before you can request data from Google Cloud Monitoring, you must configure authentication. All requests to Google APIs are performed on the server-side by the Grafana backend.

For authentication options and configuration details, refer to Google authentication.

When configuring Google authentication, note these additional Google Cloud Monitoring-specific steps:

Configure a GCP Service Account

When you create a Google Cloud Platform (GCP) Service Account and key file, the Service Account must have the **Monitoring Viewer** role (**Role > Select a role > Monitoring > Monitoring Viewer**):

Choose role

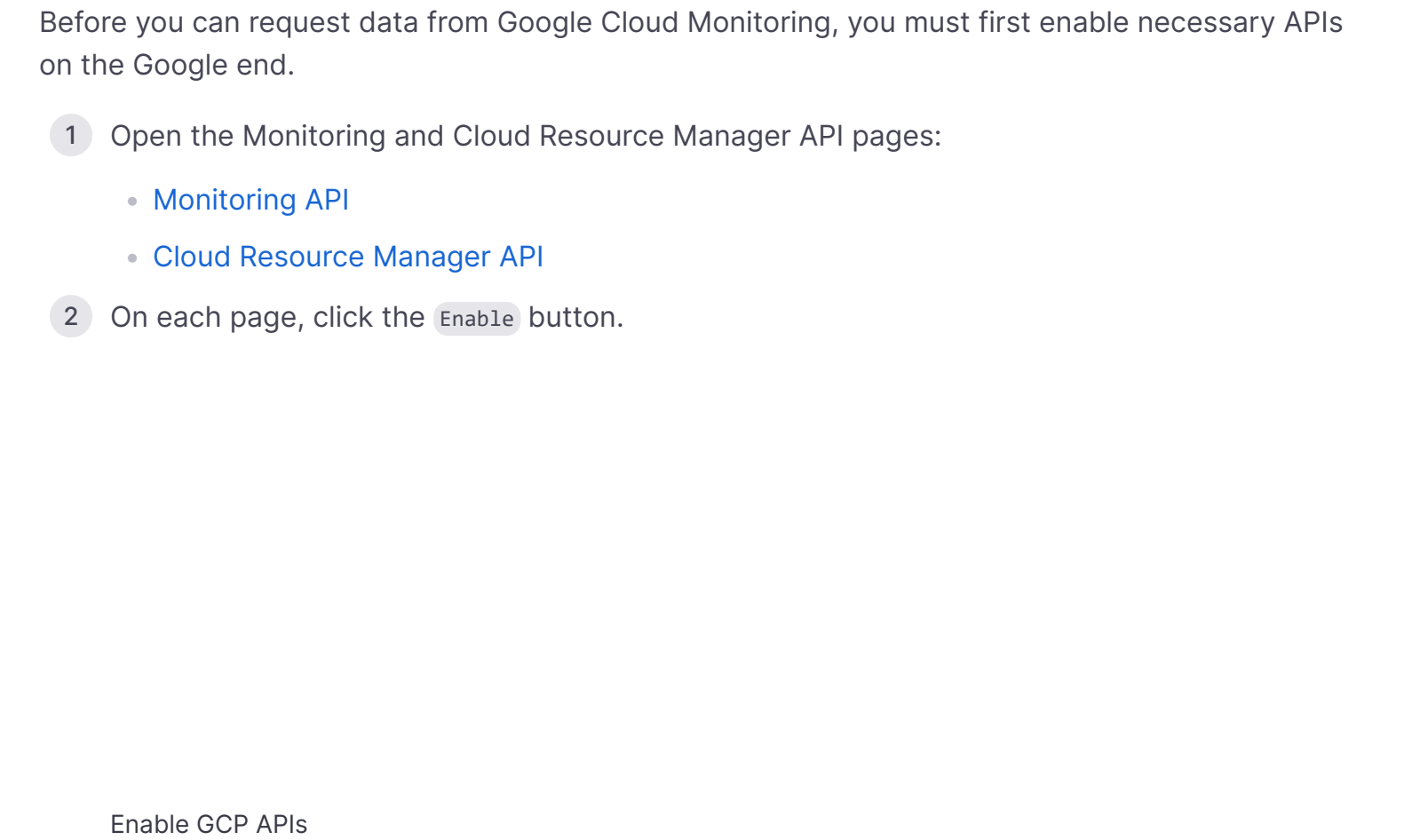
Grant the GCE Default Service Account scope

If Grafana is running on a Google Compute Engine (GCE) virtual machine, then when you Configure a GCE Default Service Account, you must also grant that Service Account access to the “Cloud Monitoring API” scope.

Enable necessary Google Cloud Platform APIs

Before you can request data from Google Cloud Monitoring, you must first enable necessary APIs on the Google end.

1. Open the Monitoring and Cloud Resource Manager API pages:
   - Monitoring API
   - Cloud Resource Manager API
2. On each page, click the `Enable` button.



Enable GCP APIs

Provision the data source

You can define and configure the data source in YAML files as part of Grafana's provisioning system. For more information about provisioning, and for available configuration options, refer to Provisioning Grafana.

Provisioning examples

Using the JWT (Service Account key file) authentication type:

yaml Copy

```yaml
apiVersion: 1

datasources:
  - name: Google Cloud Monitoring
    type: stackdriver
    access: proxy
    jsonData:
      tokenUri: https://oauth2.googleapis.com/token
      clientEmail: stackdriver@myproject.iam.gserviceaccount.com
      authenticationType: jwt
      defaultProject: my-project-name
    secureJsonData:
      privateKey: |
```

```
-----BEGIN PRIVATE KEY-----
POSEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQCb1u1Srw8ICYHS
...
yA+23427282348234=
-----END PRIVATE KEY-----
```

Using the JWT (Service Account private key path) authentication type:

yaml Copy

```yaml
apiVersion: 1

datasources:
  - name: Google Cloud Monitoring
    type: stackdriver
    access: proxy
    jsonData:
      tokenUri: https://oauth2.googleapis.com/token
      clientEmail: stackdriver@myproject.iam.gserviceaccount.com
      authenticationType: jwt
      defaultProject: my-project-name
      privateKeyPath: /etc/secrets/gce.pem
```

Using GCE Default Service Account authentication:

yaml Copy

```yaml
apiVersion: 1

datasources:
  - name: Google Cloud Monitoring
    type: stackdriver
    access: proxy
    jsonData:
      authenticationType: gce
```

Import pre-configured dashboards

The Google Cloud Monitoring data source ships with pre-configured dashboards for some of the most popular GCP services. These curated dashboards are based on similar dashboards in the GCP dashboard samples repository.

Curated dashboards for Google Cloud Monitoring

To import curated dashboards:

1. Navigate to the data source's configuration page.
2. Select the **Dashboards** tab.

   This displays the curated selection of importable dashboards.
3. Select **Import** for the dashboard to import.

The dashboards include a template variable populated with the projects accessible by the configured Service Account each time you load the dashboard. After Grafana loads the dashboard, you can select a project from the dropdown list.

To customize an imported dashboard:

To customize one of these dashboards, we recommend that you save it under a different name. If you don't, upgrading Grafana can overwrite the customized dashboard with the new version.

Query the data source

The Google Cloud Monitoring query editor helps you build two types of queries: **Metric** and **Service Level Objective (SLO)**.

For details, refer to the query editor documentation.

Use template variables

Instead of hard-coding details such as server, application, and sensor names in metric queries, you can use variables. Grafana lists these variables in dropdown select boxes at the top of the dashboard to help you change the data displayed in your dashboard. Grafana refers to such variables as template variables.

For details, see the template variables documentation.

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Amazon CloudWatch data source

Grafana ships with built-in support for Amazon CloudWatch. This topic describes queries, templates, variables, and other configuration specific to the CloudWatch data source.

For instructions on how to add a data source to Grafana, refer to the administration documentation. Only users with the organization administrator role can add data sources. Administrators can also provision the data source with Grafana's provisioning system, and should control pricing and manage service quotas accordingly.

Once you've added the data source, you can configure it so that your Grafana instance's users can create queries in its query editor when they build dashboards and use Explore.

NOTE

To troubleshoot issues while setting up the CloudWatch data source, check the `/var/log/grafana/grafana.log` file.

Configure the data source

1. Click **Connections** in the left-side menu.
2. Under Your connections, click **Data sources**.
3. Enter `CloudWatch` in the search bar.
4. Click **CloudWatch**.

   The **Settings** tab of the data source is displayed.

Configure AWS authentication

A Grafana plugin's requests to AWS are made on behalf of an AWS Identity and Access Management (IAM) role or IAM user. The IAM user or IAM role must have the associated policies to perform certain API actions.

For authentication options and configuration details, refer to AWS authentication.

IAM policy examples

To read CloudWatch metrics and EC2 tags, instances, regions, and alarms, you must grant Grafana permissions via IAM. You can attach these permissions to the IAM role or IAM user you configured in AWS authentication.

Metrics-only permissions

json Copy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowReadingMetricsFromCloudWatch",
      "Effect": "Allow",
      "Action": [
        "cloudwatch:DescribeAlarmsForMetric",
        "cloudwatch:DescribeAlarmHistory",
        "cloudwatch:DescribeAlarms",
        "cloudwatch:ListMetrics",
        "cloudwatch:GetMetricData",
        "cloudwatch:GetInsightRuleReport"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowReadingTagsInstancesRegionsFromEC2",
      "Effect": "Allow",
```

Expand code

Logs-only permissions

json Copy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowReadingLogsFromCloudWatch",
      "Effect": "Allow",
      "Action": [
```

```
        "logs:DescribeLogGroups",
        "logs:GetLogGroupFields",
        "logs:StartQuery",
        "logs:StopQuery",
        "logs:GetQueryResults",
        "logs:GetLogEvents"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowReadingTagsInstancesRegionsFromEC2",
```

Expand code

Metrics and logs permissions

json

Copy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowReadingMetricsFromCloudWatch",
      "Effect": "Allow",
      "Action": [
        "cloudwatch:DescribeAlarmsForMetric",
        "cloudwatch:DescribeAlarmHistory",
        "cloudwatch:DescribeAlarms",
        "cloudwatch:ListMetrics",
        "cloudwatch:GetMetricData",
        "cloudwatch:GetInsightRuleReport"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowReadingLogsFromCloudWatch",
      "Effect": "Allow",
```

Expand code

Cross-account observability permissions

json

Copy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": ["oam:ListSinks", "oam:ListAttachedLinks"],
      "Effect": "Allow",
```

```
      "Resource": "*"
    }
  ]
}
```

Configure CloudWatch settings

Namespaces of Custom Metrics

Grafana can't load custom namespaces through the CloudWatch GetMetricData API.

To make custom metrics appear in the data source's query editor fields, specify the names of the namespaces containing the custom metrics in the data source configuration's *Namespaces of Custom Metrics* field. The field accepts multiple namespaces separated by commas.

Timeout

Configure the timeout specifically for CloudWatch Logs queries.

Log queries don't keep a single request open, and instead periodically poll for results. Therefore, they don't recognize the standard Grafana query timeout. Because of limits on concurrently running queries in CloudWatch, they can also take longer to finish.

X-Ray trace links

To automatically add links in your logs when the log contains the `@xrayTraceId` field, link an X-Ray data source in the "X-Ray trace link" section of the data source configuration.

Trace link configuration

The data source select contains only existing data source instances of type X-Ray. To use this feature, you must already have an X-Ray data source configured. For details, see the X-Ray data source docs.

To view the X-Ray link, select the log row in either the Explore view or dashboard Logs panel to view the log details section.

To log the `@xrayTraceId`, see the AWS X-Ray documentation.

To provide the field to Grafana, your log queries must also contain the `@xrayTraceId` field, for example by using the query `fields @message, @xrayTraceId`.

Trace link in log details

Configure the data source with grafana.ini

The Grafana configuration file includes an `AWS` section where you can configure data source options:

Configuration option	Description
`allowed_auth_providers`	Specifies which authentication providers are allowed for the CloudWatch data source. The following providers are enabled by default in open-source Grafana: `default` (AWS SDK default), keys (Access and secret key), credentials (Credentials file), ec2_IAM_role (EC2 IAM role).
`assume_role_enabled`	Allows you to disable `assume role (ARN)` in the CloudWatch data source. The assume role (ARN) is enabled by default in open-source Grafana.
`list_metrics_page_limit`	Sets the limit of List Metrics API pages. When a custom namespace is specified in the query editor, the List Metrics API populates the *Metrics* field and *Dimension* fields. The API is paginated and returns up to 500 results per page, and the data source also limits the number of pages to 500 by default. This setting customizes that limit.

Provision the data source

You can define and configure the data source in YAML files as part of Grafana's provisioning system. For more information about provisioning, and for available configuration options, refer to Provisioning Grafana.

Provisioning examples

Using AWS SDK (default)

yaml Copy

```yaml
apiVersion: 1
datasources:
```

```yaml
  - name: CloudWatch
    type: cloudwatch
    jsonData:
      authType: default
      defaultRegion: eu-west-2
```

Using credentials' profile name (non-default)

yaml Copy

```yaml
apiVersion: 1

datasources:
  - name: CloudWatch
    type: cloudwatch
    jsonData:
      authType: credentials
      defaultRegion: eu-west-2
      customMetricsNamespaces: 'CWAgent,CustomNameSpace'
      profile: secondary
```

Using accessKey and secretKey

yaml Copy

```yaml
apiVersion: 1

datasources:
  - name: CloudWatch
    type: cloudwatch
    jsonData:
      authType: keys
      defaultRegion: eu-west-2
    secureJsonData:
      accessKey: '<your access key>'
      secretKey: '<your secret key>'
```

Using AWS SDK Default and ARN of IAM Role to Assume

yaml Copy

```yaml
apiVersion: 1
datasources:
  - name: CloudWatch
```

```
    type: cloudwatch
    jsonData:
      authType: default
      assumeRoleArn: arn:aws:iam::123456789012:root
      defaultRegion: eu-west-2
```

Query the data source

The CloudWatch data source can query data from both CloudWatch metrics and CloudWatch Logs APIs, each with its own specialized query editor.

For details, see the query editor documentation.

Use template variables

Instead of hard-coding details such as server, application, and sensor names in metric queries, you can use variables. Grafana lists these variables in dropdown select boxes at the top of the dashboard to help you change the data displayed in your dashboard. Grafana refers to such variables as template variables.

For details, see the template variables documentation.

Import pre-configured dashboards

The CloudWatch data source ships with curated and pre-configured dashboards for five of the most popular AWS services:

- **Amazon Elastic Compute Cloud:** `Amazon EC2`
- **Amazon Elastic Block Store:** `Amazon EBS`
- **AWS Lambda:** `AWS Lambda`
- **Amazon CloudWatch Logs:** `Amazon CloudWatch Logs`
- **Amazon Relational Database Service:** `Amazon RDS`

To import curated dashboards:

1. Navigate to the data source's configuration page.
2. Select the **Dashboards** tab.

   This displays the curated selection of importable dashboards.
3. Select **Import** for the dashboard to import.

CloudWatch dashboard import

To customize an imported dashboard:

To customize one of these dashboards, we recommend that you save it under a different name. If you don't, upgrading Grafana can overwrite the customized dashboard with the new version.

Create queries for alerting

Alerting requires queries that return numeric data, which CloudWatch Logs support. For example, you can enable alerts through the use of the `stats` command.

This is also a valid query for alerting on messages that include the text "Exception":

```
filter @message like /Exception/
    | stats count(*) as exceptionCount by bin(1h)
    | sort exceptionCount desc
```

Copy

> **NOTE**
>
> If you receive an error like `input data must be a wide series but got ...` when trying to alert on a query, make sure that your query returns valid numeric data that can be output to a Time series panel.

For more information on Grafana alerts, refer to Alerting.

Control pricing

The Amazon CloudWatch data source for Grafana uses `ListMetrics` and `GetMetricData` CloudWatch API calls to list and retrieve metrics. Pricing for CloudWatch Logs is based on the amount of data ingested, archived, and analyzed via CloudWatch Logs Insights queries. Each time you select a

dimension in the query editor, Grafana issues a `ListMetrics` API request. Each time you change queries in the query editor, Grafana issues a new request to the `GetMetricData` API.

> **NOTE**
>
> Grafana v6.5 and higher replaced all `GetMetricStatistics` API requests with calls to GetMetricData to provide better support for CloudWatch metric math, and enables the automatic generation of search expressions when using wildcards or disabling the `Match Exact` option. The `GetMetricStatistics` API qualified for the CloudWatch API free tier, but `GetMetricData` calls don't.

For more information, refer to the CloudWatch pricing page.

Manage service quotas

AWS defines quotas, or limits, for resources, actions, and items in your AWS account. Depending on the number of queries in your dashboard and the number of users accessing the dashboard, you might reach the usage limits for various CloudWatch and CloudWatch Logs resources. Quotas are defined per account and per region.

If you use multiple regions or configured more than one CloudWatch data source to query against multiple accounts, you must request a quota increase for each account and region in which you reach the limit.

To request a quota increase, visit the AWS Service Quotas console. For more information, refer to the AWS documentation for Service Quotas and CloudWatch limits.

Cross-account observability

The CloudWatch plugin enables you to monitor and troubleshoot applications across multiple regional accounts. Using cross-account observability, you can seamlessly search, visualize and analyze metrics and logs without worrying about account boundaries.

To use this feature, configure in the AWS console under Cloudwatch Settings, a monitoring and source account, and then add the necessary IAM permissions as described above.

CloudWatch Logs data protection

CloudWatch Logs can safeguard data by using log group data protection policies. If you have data protection enabled for a log group, then any sensitive data that matches the data identifiers you've selected will be masked. In order to view masked data you will need to have the `logs:Unmask` IAM permission enabled. See the AWS documentation on how to help protect sensitive log data with masking to learn more about this.

Loki data source

Grafana Loki is a set of components that can be combined into a fully featured logging stack. Unlike other logging systems, Loki is built around the idea of only indexing metadata about your logs: labels (just like Prometheus labels). Log data itself is then compressed and stored in chunks in object stores such as S3 or GCS, or even locally on a filesystem.

The following guides will help you get started with Loki:

- Getting started with Loki
- Install Loki
- Loki best practices
- Configure the Loki data source
- LogQL
- Loki query editor

Adding a data source

For instructions on how to add a data source to Grafana, refer to the administration documentation Only users with the organization administrator role can add data sources. Administrators can also configure the data source via YAML with Grafana's provisioning system.

Once you've added the Loki data source, you can configure it so that your Grafana instance's users can create queries in its query editor when they build dashboards, use Explore, and annotate visualizations.

NOTE

To troubleshoot configuration and other issues, check the log file located at `/var/log/grafana/grafana.log` on Unix systems, or in `<grafana_install_dir>/data/log` on other platforms and manual installations.

Provision the data source

You can define and configure the data source in YAML files as part of Grafana's provisioning system. For more information about provisioning, and for available configuration options, refer to Provisioning Grafana.

Provisioning examples

yaml Copy

```yaml
apiVersion: 1

datasources:
  - name: Loki
    type: loki
    access: proxy
    url: http://localhost:3100
    jsonData:
      timeout: 60
      maxLines: 1000
```

Using basic authorization and a derived field:

You must escape the dollar (`$`) character in YAML values because it can be used to interpolate environment variables:

yaml Copy

```yaml
apiVersion: 1

datasources:
  - name: Loki
    type: loki
    access: proxy
    url: http://localhost:3100
    basicAuth: true
    basicAuthUser: my_user
    jsonData:
      maxLines: 1000
      derivedFields:
        # Field with internal link pointing to data source in Grafana.
        # datasourceUid value can be anything, but it should be unique across all defined data sou
        - datasourceUid: my_jaeger_uid
```

```
        matcherRegex: "traceID=(\\w+)"
        name: TraceID
        # url will be interpreted as query for the datasource
```

Expand code

Using a Jaeger data source:

In this example, the Jaeger data source's `uid` value should match the Loki data source's `datasourceUid` value.

```
datasources:
    - name: Jaeger
      type: jaeger
      url: http://jaeger-tracing-query:16686/
      access: proxy
      # UID should match the datasourceUid in derivedFields.
      uid: my_jaeger_uid
```

Copy

Query the data source

The Loki data source's query editor helps you create log and metric queries that use Loki's query language, LogQL.

For details, refer to the query editor documentation.

Use template variables

Instead of hard-coding details such as server, application, and sensor names in metric queries, you can use variables. Grafana lists these variables in dropdown select boxes at the top of the dashboard to help you change the data displayed in your dashboard. Grafana refers to such variables as template variables.

For details, see the template variables documentation.

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

MySQL data source

Starting from Grafana v5.1 you can name the time column *time* in addition to earlier supported *time_sec*. Usage of *time_sec* will eventually be deprecated.

Grafana ships with a built-in MySQL data source plugin that allows you to query and visualize data from a MySQL compatible database like MariaDB or Percona Server.

For instructions on how to add a data source to Grafana, refer to the administration documentation. Only users with the organization administrator role can add data sources. Administrators can also configure the data source via YAML with Grafana's provisioning system.

Give it a try using Grafana Play

With Grafana Play, you can explore and see how it works, learning from practical examples to accelerate your development. This feature can be seen on MySQL: Cities of the World Sample Data Set.

Configure the data source

To access the data source configuration page:

1. Click **Connections** in the left-side menu.
2. Under Your connections, click **Data sources**.
3. Enter `MySQL` in the search bar.
4. Select **MySQL**.

   The **Settings** tab of the data source is displayed.

5 Set the data source's basic configuration options.

Name	Description
Name	The data source name. This is how you refer to the data source in panels and queries.
Default	Default data source means that it will be pre-selected for new panels.
Host	The IP address/hostname and optional port of your MySQL instance.
Database	Name of your MySQL database.
User	Database user's login/username
Password	Database user's password
Session Timezone	Specifies the timezone used in the database session, such as `Europe/Berlin` or `+02:00`. Required if the timezone of the database (or the host of the database) is set to something other than UTC. Set this to `+00:00` so Grafana can handle times properly. Set the value used in the session with `SET time_zone='...'`. If you leave this field empty, the timezone will not be updated. For more information, refer to MySQL Server Time Zone Support.
Max open	The maximum number of open connections to the database, default `100` (Grafana v5.4+).
Max idle	The maximum number of connections in the idle connection pool, default `100` (Grafana v5.4+).
Auto (max idle)	Toggle to set the maximum number of idle connections to the number of maximum open connections (available in Grafana v9.5.1+). Default is `true`.
Allow cleartext passwords	Allows the use of the cleartext client side plugin as required by a specific type of account, such as one defined with the PAM authentication plugin. **Sending passwords in clear text may be a security problem in some configurations.** To avoid password issues, it is recommended that clients connect to a MySQL server using a method that protects the password. Possibilities include TLS / SSL, IPsec, or a private network. Default is `false`.
Max lifetime	The maximum amount of time in seconds a connection may be reused. This should always be lower than configured wait_timeout in MySQL (Grafana v5.4+). The default is `14400` or 4 hours.

Min time interval

The **Min time interval** setting defines a lower limit for the `$__interval` and `$__interval_ms` variables.

This value *must* be formatted as a number followed by a valid time identifier:

Identifier	Description
`y`	year

Identifier	Description
`M`	month
`w`	week
`d`	day
`h`	hour
`m`	minute
`s`	second
`ms`	millisecond

We recommend setting this value to match your MySQL write frequency. For example, use `1m` if MySQL writes data every minute.

You can also override this setting in a dashboard panel under its data source options.

Database User Permissions (Important!)

The database user you specify when you add the data source should only be granted SELECT permissions on the specified database and tables you want to query. Grafana does not validate that the query is safe. The query could include any SQL statement. For example, statements like `USE otherdb;` and `DROP TABLE user;` would be executed. To protect against this we **Highly** recommend you create a specific mysql user with restricted permissions.

Example:

sql Copy

```sql
CREATE USER 'grafanaReader' IDENTIFIED BY 'password';
GRANT SELECT ON mydatabase.mytable TO 'grafanaReader';
```

You can use wildcards (`*`) in place of database or table if you want to grant access to more databases and tables.

Provision the data source

You can define and configure the data source in YAML files as part of Grafana's provisioning system. For more information about provisioning, and for available configuration options, refer to Provisioning Grafana.

Provisioning examples

Basic Provisioning

yaml Copy

```yaml
apiVersion: 1

datasources:
  - name: MySQL
    type: mysql
    url: localhost:3306
    user: grafana
    jsonData:
      database: grafana
      maxOpenConns: 100 # Grafana v5.4+
      maxIdleConns: 100 # Grafana v5.4+
      maxIdleConnsAuto: true # Grafana v9.5.1+
      connMaxLifetime: 14400 # Grafana v5.4+
    secureJsonData:
      password: ${GRAFANA_MYSQL_PASSWORD}
```

Using TLS verification

yaml Copy

```yaml
apiVersion: 1

datasources:
  - name: MySQL
    type: mysql
    url: localhost:3306
    user: grafana
    jsonData:
      tlsAuth: true
      database: grafana
      maxOpenConns: 100 # Grafana v5.4+
      maxIdleConns: 100 # Grafana v5.4+
      maxIdleConnsAuto: true # Grafana v9.5.1+
      connMaxLifetime: 14400 # Grafana v5.4+
    secureJsonData:
      password: ${GRAFANA_MYSQL_PASSWORD}
      tlsClientCert: ${GRAFANA_TLS_CLIENT_CERT}
      tlsCACert: ${GRAFANA_TLS_CA_CERT}
```

Use TLS and Skip Certificate Verification

yaml Copy

```yaml
apiVersion: 1

datasources:
  - name: MySQL
    type: mysql
    url: localhost:3306
    user: grafana
    jsonData:
      tlsAuth: true
      tlsSkipVerify: true
      database: grafana
      maxOpenConns: 100 # Grafana v5.4+
      maxIdleConns: 100 # Grafana v5.4+
      maxIdleConnsAuto: true # Grafana v9.5.1+
      connMaxLifetime: 14400 # Grafana v5.4+
    secureJsonData:
      password: ${GRAFANA_MYSQL_PASSWORD}
      tlsClientCert: ${GRAFANA_TLS_CLIENT_CERT}
      tlsCACert: ${GRAFANA_TLS_CA_CERT}
```

Expand code

Query builder

The MySQL query builder is available when editing a panel using a MySQL data source.

This topic explains querying specific to the MySQL data source. For general documentation on querying data sources in Grafana, see Query and transform data.

You can run the built query by pressing the `Run query` button in the top right corner of the editor.

Format

The response from MySQL can be formatted as either a table or as a time series. To use the time series format one of the columns must be named `time`.

Dataset and Table selection

> **NOTE**
>
> If your table or database name contains a reserved word or a prohibited character the editor will put quotes around the name. For example, the name `table-name` will be quoted with backticks - `` `table-name` ``.

In the dataset dropdown, choose the MySQL database to query. The dropdown is be populated with the databases that the user has access to. When the dataset is selected, the table dropdown is populated with the tables that are available.

Note: If a default database has been configured through the Data Source Configuration page (or through a provisioning configuration file), the user will only be able to use that single preconfigured database for querying.

Columns and Aggregation functions (SELECT)

Using the dropdown, select a column to include in the data. You can also specify an optional aggregation function.

Add further value columns by clicking the plus button and another column dropdown appears.

Filter data (WHERE)

To add a filter, toggle the **Filter** switch at the top of the editor. This reveals a **Filter by column value** section with two dropdown selectors.

Use the first dropdown to choose whether all of the filters need to match (`AND`), or if only one of the filters needs to match (`OR`). Use the second dropdown to choose a filter.

To filter on more columns, click the plus (`+`) button to the right of the condition dropdown.

To remove a filter, click the `x` button next to that filter's dropdown.

After selecting a date type column, you can choose Macros from the operators list and select timeFilter which will add the $__timeFilter macro to the query with the selected date column.

Group By

To group the results by column, flip the group switch at the top of the editor. You can then choose which column to group the results by. The group by clause can be removed by pressing the X

button.

Preview

By flipping the preview switch at the top of the editor, you can get a preview of the SQL query generated by the query builder.

Code editor

To make advanced queries, switch to the code editor by clicking `code` in the top right corner of the editor. The code editor support autocompletion of tables, columns, SQL keywords, standard sql functions, Grafana template variables and Grafana macros. Columns cannot be completed before a table has been specified.

You can expand the code editor by pressing the `chevron` pointing downwards in the lower right corner of the code editor.

`CTRL/CMD + Return` works as a keyboard shortcut to run the query.

Macros

To simplify syntax and to allow for dynamic parts, like date range filters, the query can contain macros.

Macro example	Description
`$__time(dateColumn)`	Will be replaced by an expression to convert to a UNIX timestamp and rename the column to `time_sec`. For example, *UNIX_TIMESTAMP(dateColumn) as time_sec*
`$__timeEpoch(dateColumn)`	Will be replaced by an expression to convert to a UNIX timestamp and rename the column to `time_sec`. For example, *UNIX_TIMESTAMP(dateColumn) as time_sec*
`$__timeFilter(dateColumn)`	Will be replaced by a time range filter using the specified column name. For example, *dateColumn BETWEEN*

Macro example	Description
	FROM_UNIXTIME(1494410783) AND FROM_UNIXTIME(1494410983)
`$__timeFrom()`	Will be replaced by the start of the currently active time selection. For example, *FROM_UNIXTIME(1494410783)*
`$__timeTo()`	Will be replaced by the end of the currently active time selection. For example, *FROM_UNIXTIME(1494410983)*
`$__timeGroup(dateColumn,'5m')`	Will be replaced by an expression usable in GROUP BY clause. For example, *cast(cast(UNIX_TIMESTAMP(dateColumn)/(300) as signed)*300 as signed),*
`$__timeGroup(dateColumn,'5m', 0)`	Same as above but with a fill parameter so missing points in that series will be added by grafana and 0 will be used as value (only works with time series queries).
`$__timeGroup(dateColumn,'5m', NULL)`	Same as above but NULL will be used as value for missing points (only works with time series queries).
`$__timeGroup(dateColumn,'5m', previous)`	Same as above but the previous value in that series will be used as fill value if no value has been seen yet NULL will be used (only works with time series queries).
`$__timeGroupAlias(dateColumn,'5m')`	Will be replaced identical to $__timeGroup but with an added column alias (only available in Grafana 5.3+).
`$__unixEpochFilter(dateColumn)`	Will be replaced by a time range filter using the specified column name with times represented as Unix timestamp. For example, *dateColumn > 1494410783 AND dateColumn < 1494497183*
`$__unixEpochFrom()`	Will be replaced by the start of the currently active time selection as Unix timestamp. For example, *1494410783*
`$__unixEpochTo()`	Will be replaced by the end of the currently active time selection as Unix timestamp. For example, *1494497183*
`$__unixEpochNanoFilter(dateColumn)`	Will be replaced by a time range filter using the specified column name with times represented as nanosecond timestamp. For example, *dateColumn > 1494410783152415214 AND dateColumn < 1494497183142514872*
`$__unixEpochNanoFrom()`	Will be replaced by the start of the currently active time selection as nanosecond timestamp. For example, *1494410783152415214*
`$__unixEpochNanoTo()`	Will be replaced by the end of the currently active time selection as nanosecond timestamp. For example, *1494497183142514872*
`$__unixEpochGroup(dateColumn,'5m', [fillmode])`	Same as $__timeGroup but for times stored as Unix timestamp (`fillMode` only works with time series queries).
`$__unixEpochGroupAlias(dateColumn,'5m', [fillmode])`	Same as above but also adds a column alias (`fillMode` only works with time series queries).

We plan to add many more macros. If you have suggestions for what macros you would like to see, please open an issue in our GitHub repo.

The query editor has a link named `Generated SQL` that shows up after a query has been executed, while in panel edit mode. Click on it and it will expand and show the raw interpolated SQL string that was executed.

Table queries

If the `Format as` query option is set to `Table` then you can basically do any type of SQL query. The table panel will automatically show the results of whatever columns and rows your query returns.

Query editor with example query:

The query:

```sql
SELECT
  title as 'Title',
  user.login as 'Created By' ,
  dashboard.created as 'Created On'
 FROM dashboard
INNER JOIN user on user.id = dashboard.created_by
WHERE $__timeFilter(dashboard.created)
```

You can control the name of the Table panel columns by using regular `as` SQL column selection syntax.

The resulting table panel:

Time series queries

The examples in this section query the following table:

text

Copy

```
+---------------------+--------------+---------------------+----------+
| time_date_time      | value_double | CreatedAt           | hostname |
+---------------------+--------------+---------------------+----------+
| 2020-01-02 03:05:00 | 3.0          | 2020-01-02 03:05:00 | 10.0.1.1 |
| 2020-01-02 03:06:00 | 4.0          | 2020-01-02 03:06:00 | 10.0.1.2 |
| 2020-01-02 03:10:00 | 6.0          | 2020-01-02 03:10:00 | 10.0.1.1 |
| 2020-01-02 03:11:00 | 7.0          | 2020-01-02 03:11:00 | 10.0.1.2 |
| 2020-01-02 03:20:00 | 5.0          | 2020-01-02 03:20:00 | 10.0.1.2 |
+---------------------+--------------+---------------------+----------+
```

If the `Format as` query option is set to `Time Series` then the query must have a column named time that returns either a SQL datetime or any numeric datatype representing Unix epoch in seconds. In addition, result sets of time series queries must be sorted by time for panels to properly visualize the result.

A time series query result is returned in a wide data frame format. Any column except time or of type string transforms into value fields in the data frame query result. Any string column transforms into field labels in the data frame query result.

> For backward compatibility, there's an exception to the above rule for queries that return three columns including a string column named metric. Instead of transforming the metric column into field labels, it becomes the field name, and then the series name is formatted as the value of the metric column. See the example with the metric column below.

To optionally customize the default series name formatting, refer to Standard options definitions.

Example with `metric` column:

sql

Copy

```sql
SELECT
  $__timeGroupAlias(time_date_time,'5m'),
  min(value_double),
  'min' as metric
FROM test_data
WHERE $__timeFilter(time_date_time)
```

```
GROUP BY time
ORDER BY time
```

Data frame result:

text Copy

```
+--------------------+-----------------+
| Name: time         | Name: min       |
| Labels:            | Labels:         |
| Type: []time.Time  | Type: []float64 |
+--------------------+-----------------+
| 2020-01-02 03:05:00 | 3              |
| 2020-01-02 03:10:00 | 6              |
| 2020-01-02 03:20:00 | 5              |
+--------------------+-----------------+
```

Example using the fill parameter in the $__timeGroupAlias macro to convert null values to be zero instead:

sql Copy

```
SELECT
  $__timeGroupAlias(createdAt,'5m',0),
  sum(value_double) as value,
  hostname
FROM test_data
WHERE
  $__timeFilter(createdAt)
GROUP BY time, hostname
ORDER BY time
```

Given the data frame result in the following example and using the graph panel, you will get two series named *value 10.0.1.1* and *value 10.0.1.2*. To render the series with a name of *10.0.1.1* and *10.0.1.2* , use a Standard options definitions display value of `${__field.labels.hostname}`.

Data frame result:

text Copy

```
+--------------------+--------------------------+--------------------------+
| Name: time         | Name: value              | Name: value              |
| Labels:            | Labels: hostname=10.0.1.1 | Labels: hostname=10.0.1.2 |
| Type: []time.Time  | Type: []float64          | Type: []float64          |
+--------------------+--------------------------+--------------------------+
```

```
| 2020-01-02 03:05:00 | 3                    | 4                    |
| 2020-01-02 03:10:00 | 6                    | 7                    |
| 2020-01-02 03:15:00 | 0                    | 0                    |
| 2020-01-02 03:20:00 | 0                    | 5                    |
+---------------------+----------------------+----------------------+
```

Example with multiple columns:

sql Copy

```sql
SELECT
  $__timeGroupAlias(time_date_time,'5m'),
  min(value_double) as min_value,
  max(value_double) as max_value
FROM test_data
WHERE $__timeFilter(time_date_time)
GROUP BY time
ORDER BY time
```

Data frame result:

text Copy

```
+---------------------+-----------------+-----------------+
| Name: time          | Name: min_value | Name: max_value |
| Labels:             | Labels:         | Labels:         |
| Type: []time.Time   | Type: []float64 | Type: []float64 |
+---------------------+-----------------+-----------------+
| 2020-01-02 03:05:00 | 3               | 4               |
| 2020-01-02 03:10:00 | 6               | 7               |
| 2020-01-02 03:20:00 | 5               | 5               |
+---------------------+-----------------+-----------------+
```

Currently, there is no support for a dynamic group by time based on time range and panel width. This is something we plan to add.

Templating

This feature is currently available in the nightly builds and will be included in the 5.0.0 release.

Instead of hard-coding things like server, application and sensor name in your metric queries you can use variables in their place. Variables are shown as dropdown select boxes at the top of the dashboard. These dropdowns make it easy to change the data being displayed in your dashboard.

Check out the Templating documentation for an introduction to the templating feature and the different types of template variables.

Query Variable

If you add a template variable of the type `Query`, you can write a MySQL query that can return things like measurement names, key names or key values that are shown as a dropdown select box.

For example, you can have a variable that contains all values for the `hostname` column in a table if you specify a query like this in the templating variable *Query* setting.

```sql
SELECT hostname FROM my_host
```

A query can return multiple columns and Grafana will automatically create a list from them. For example, the query below will return a list with values from `hostname` and `hostname2`.

```sql
SELECT my_host.hostname, my_other_host.hostname2 FROM my_host JOIN my_other_host ON my_host.city =
```

To use time range dependent macros like `$__timeFilter(column)` in your query the refresh mode of the template variable needs to be set to *On Time Range Change*.

```sql
SELECT event_name FROM event_log WHERE $__timeFilter(time_column)
```

Another option is a query that can create a key/value variable. The query should return two columns that are named `__text` and `__value`. The `__text` column value should be unique (if it is not unique then the first value is used). The options in the dropdown will have a text and value that allows you to have a friendly name as text and an id as the value. An example query with `hostname` as the text and `id` as the value:

```sql
SELECT hostname AS __text, id AS __value FROM my_host
```

You can also create nested variables. For example if you had another variable named `region`. Then you could have the hosts variable only show hosts from the current selected region with a query like this (if `region` is a multi-value variable then use the `IN` comparison operator rather than `=` to match against multiple values):

```sql
SELECT hostname FROM my_host  WHERE region IN($region)
```

Using `__searchFilter` to filter results in Query Variable

> Available from Grafana 6.5 and above

Using `__searchFilter` in the query field will filter the query result based on what the user types in the dropdown select box. When nothing has been entered by the user the default value for `__searchFilter` is `%`.

> Important that you surround the `__searchFilter` expression with quotes as Grafana does not do this for you.

The example below shows how to use `__searchFilter` as part of the query field to enable searching for `hostname` while the user types in the dropdown select box.

Query

```sql
SELECT hostname FROM my_host  WHERE hostname LIKE '$__searchFilter'
```

Using Variables in Queries

From Grafana 4.3.0 to 4.6.0, template variables are always quoted automatically so if it is a string value do not wrap them in quotes in where clauses.

From Grafana 4.7.0, template variable values are only quoted when the template variable is a `multi-value`.

If the variable is a multi-value variable then use the `IN` comparison operator rather than `=` to match against multiple values.

There are two syntaxes:

`$<varname>` Example with a template variable named `hostname`:

```sql
SELECT
  UNIX_TIMESTAMP(atimestamp) as time,
  aint as value,
  avarchar as metric
FROM my_table
```

```
WHERE $__timeFilter(atimestamp) and hostname in($hostname)
ORDER BY atimestamp ASC
```

`[[varname]]` Example with a template variable named `hostname`:

sql Copy

```sql
SELECT
  UNIX_TIMESTAMP(atimestamp) as time,
  aint as value,
  avarchar as metric
FROM my_table
WHERE $__timeFilter(atimestamp) and hostname in([[hostname]])
ORDER BY atimestamp ASC
```

Disabling Quoting for Multi-value Variables

Grafana automatically creates a quoted, comma-separated string for multi-value variables. For example: if `server01` and `server02` are selected then it will be formatted as: `'server01', 'server02'`. To disable quoting, use the csv formatting option for variables:

`${servers:csv}`

Read more about variable formatting options in the Variables documentation.

Annotations

Annotations allow you to overlay rich event information on top of graphs. You add annotation queries via the Dashboard menu / Annotations view.

Example query using time column with epoch values:

sql Copy

```sql
SELECT
  epoch_time as time,
  metric1 as text,
  CONCAT(tag1, ',', tag2) as tags
FROM
  public.test_data
WHERE
  $__unixEpochFilter(epoch_time)
```

Example region query using time and timeend columns with epoch values:

Only available in Grafana v6.6+.

sql Copy

```sql
SELECT
  epoch_time as time,
  epoch_timeend as timeend,
  metric1 as text,
  CONCAT(tag1, ',', tag2) as tags
FROM
  public.test_data
WHERE
  $__unixEpochFilter(epoch_time)
```

Example query using time column of native SQL date/time data type:

sql Copy

```sql
SELECT
  native_date_time as time,
  metric1 as text,
  CONCAT(tag1, ',', tag2) as tags
FROM
  public.test_data
WHERE
  $__timeFilter(native_date_time)
```

Name	Description
`time`	The name of the date/time field. Could be a column with a native SQL date/time data type or epoch value.
`timeend`	Optional name of the end date/time field. Could be a column with a native SQL date/time data type or epoch value. (Grafana v6.6+)
`text`	Event description field.
`tags`	Optional field name to use for event tags as a comma separated string.

Alerting

Time series queries should work in alerting conditions. Table formatted queries are not yet supported in alert rule conditions.

PostgreSQL data source

Grafana ships with a built-in PostgreSQL data source plugin that allows you to query and visualize data from a PostgreSQL compatible database.

For instructions on how to add a data source to Grafana, refer to the administration documentation. Only users with the organization administrator role can add data sources. Administrators can also configure the data source via YAML with Grafana's provisioning system.

PostgreSQL settings

To configure basic settings for the data source, complete the following steps:

1. Click **Connections** in the left-side menu.
2. Under Your connections, click **Data sources**.
3. Enter `PostgreSQL` in the search bar.
4. Select **PostgreSQL**.

   The **Settings** tab of the data source is displayed.
5. Set the data source's basic configuration options:

Name	Description
Name	The data source name. This is how you refer to the data source in panels and queries.
Default	Default data source means that it will be pre-selected for new panels.
Host	The IP address/hostname and optional port of your PostgreSQL instance. *Do not* include the database name. The connection string for connecting to Postgres will not be correct and it may cause errors.

Name	Description
Database	Name of your PostgreSQL database.
User	Database user's login/username
Password	Database user's password
SSL Mode	Determines whether or with what priority a secure SSL TCP/IP connection will be negotiated with the server. When SSL Mode is disabled, SSL Method and Auth Details would not be visible.
SSL Auth Details Method	Determines whether the SSL Auth details will be configured as a file path or file content. Grafana v7.5+
SSL Auth Details Value	File path or file content of SSL root certificate, client certificate and client key
Max open	The maximum number of open connections to the database, default `100` (Grafana v5.4+).
Max idle	The maximum number of connections in the idle connection pool, default `100` (Grafana v5.4+).
Auto (max idle)	If set will set the maximum number of idle connections to the number of maximum open connections (Grafana v9.5.1+). Default is `true`.
Max lifetime	The maximum amount of time in seconds a connection may be reused, default `14400`/4 hours (Grafana v5.4+).
Version	Determines which functions are available in the query builder (only available in Grafana 5.3+).
TimescaleDB	A time-series database built as a PostgreSQL extension. When enabled, Grafana uses `time_bucket` in the `$__timeGroup` macro to display TimescaleDB specific aggregate functions in the query builder (only available in Grafana 5.3+). For more information, see TimescaleDB documentation.

Min time interval

A lower limit for the `$__interval` and `$__interval_ms` variables. Recommended to be set to write frequency, for example `1m` if your data is written every minute. This option can also be overridden/configured in a dashboard panel under data source options. It's important to note that this value **needs** to be formatted as a number followed by a valid time identifier, e.g. `1m` (1 minute) or `30s` (30 seconds). The following time identifiers are supported:

Identifier	Description
`y`	year
`M`	month
`w`	week
`d`	day

Identifier	Description
`h`	hour
`m`	minute
`s`	second
`ms`	millisecond

Database user permissions (Important!)

The database user you specify when you add the data source should only be granted SELECT permissions on the specified database and tables you want to query. Grafana does not validate that the query is safe. The query could include any SQL statement. For example, statements like `DELETE FROM user;` and `DROP TABLE user;` would be executed. To protect against this we **highly** recommend you create a specific PostgreSQL user with restricted permissions.

Example:

```sql
CREATE USER grafanareader WITH PASSWORD 'password';
GRANT USAGE ON SCHEMA schema TO grafanareader;
GRANT SELECT ON schema.table TO grafanareader;
```

Make sure the user does not get any unwanted privileges from the public role.

Query builder

PostgreSQL query builder

The PostgreSQL query builder is available when editing a panel using a PostgreSQL data source. The built query can be run by pressing the `Run query` button in the top right corner of the editor.

Format

The response from PostgreSQL can be formatted as either a table or as a time series. To use the time series format one of the columns must be named `time`.

Dataset and table selection

The dataset dropdown will be populated with the configured database to which the user has access. The table dropdown is populated with the tables that are available within that database.

Columns and Aggregation functions (SELECT)

Using the dropdown, select a column to include in the data. You can also specify an optional aggregation function.

Add further value columns by clicking the plus button and another column dropdown appears.

Filter data (WHERE)

To add a filter, toggle the **Filter** switch at the top of the editor. This reveals a **Filter by column value** section with two dropdown selectors.

Use the first dropdown to choose whether all of the filters need to match (`AND`), or if only one of the filters needs to match (`OR`). Use the second dropdown to choose a filter.

To filter on more columns, click the plus (`+`) button to the right of the condition dropdown.

To remove a filter, click the `x` button next to that filter's dropdown.

After selecting a date type column, you can choose Macros from the operators list and select timeFilter which will add the $__timeFilter macro to the query with the selected date column.

Group By

To group the results by column, flip the group switch at the top of the editor. You can then choose which column to group the results by. The group by clause can be removed by pressing the X button.

Preview

By flipping the preview switch at the top of the editor, you can get a preview of the SQL query generated by the query builder.

Provision the data source

It's now possible to configure data sources using config files with Grafana's provisioning system. You can read more about how it works and all the settings you can set for data sources on the

provisioning docs page.

Provisioning example

yaml Copy

```yaml
apiVersion: 1

datasources:
  - name: Postgres
    type: postgres
    url: localhost:5432
    user: grafana
    secureJsonData:
      password: 'Password!'
    jsonData:
      database: grafana
      sslmode: 'disable' # disable/require/verify-ca/verify-full
      maxOpenConns: 100 # Grafana v5.4+
      maxIdleConns: 100 # Grafana v5.4+
      maxIdleConnsAuto: true # Grafana v9.5.1+
      connMaxLifetime: 14400 # Grafana v5.4+
      postgresVersion: 903 # 903=9.3, 904=9.4, 905=9.5, 906=9.6, 1000=10
      timescaledb: false
```

> **NOTE**
>
> In the above code, the `postgresVersion` value of `10` refers to version PostgreSQL 10 and above.

Troubleshoot provisioning

If you encounter metric request errors or other issues:

- Make sure your data source YAML file parameters exactly match the example. This includes parameter names and use of quotation marks.
- Make sure the `database` name is not included in the `url`.

Code editor

To make advanced queries, switch to the code editor by clicking `code` in the top right corner of the editor. The code editor support autocompletion of tables, columns, SQL keywords, standard sql functions, Grafana template variables and Grafana macros. Columns cannot be completed before a table has been specified.

You can expand the code editor by pressing the `chevron` pointing downwards in the lower right corner of the code editor.

`CTRL/CMD + Return` works as a keyboard shortcut to run the query.

Macros

Macros can be used within a query to simplify syntax and allow for dynamic parts.

Macro example	Description
`$__time(dateColumn)`	Will be replaced by an expression to convert to a UNIX timestamp and rename the column to `time_sec`. For example, *UNIX_TIMESTAMP(dateColumn) as time_sec*
`$__timeEpoch(dateColumn)`	Will be replaced by an expression to convert to a UNIX timestamp and rename the column to `time_sec`. For example, *UNIX_TIMESTAMP(dateColumn) as time_sec*
`$__timeFilter(dateColumn)`	Will be replaced by a time range filter using the specified column name. For example, *dateColumn BETWEEN FROM_UNIXTIME(1494410783) AND FROM_UNIXTIME(1494410983)*
`$__timeFrom()`	Will be replaced by the start of the currently active time selection. For example, *FROM_UNIXTIME(1494410783)*
`$__timeTo()`	Will be replaced by the end of the currently active time selection. For example, *FROM_UNIXTIME(1494410983)*
`$__timeGroup(dateColumn,'5m')`	Will be replaced by an expression usable in GROUP BY clause. For example, *cast(cast(UNIX_TIMESTAMP(dateColumn)/(300) as signed)*300 as signed),*
`$__timeGroup(dateColumn,'5m', 0)`	Same as above but with a fill parameter so missing points in that series will be added by grafana and 0 will be used as

Macro example	Description
	value (only works with time series queries).
`$__timeGroup(dateColumn,'5m', NULL)`	Same as above but NULL will be used as value for missing points (only works with time series queries).
`$__timeGroup(dateColumn,'5m', previous)`	Same as above but the previous value in that series will be used as fill value if no value has been seen yet NULL will be used (only works with time series queries).
`$__timeGroupAlias(dateColumn,'5m')`	Will be replaced identical to $__timeGroup but with an added column alias.
`$__unixEpochFilter(dateColumn)`	Will be replaced by a time range filter using the specified column name with times represented as Unix timestamp. For example, *dateColumn > 1494410783 AND dateColumn < 1494497183*
`$__unixEpochFrom()`	Will be replaced by the start of the currently active time selection as Unix timestamp. For example, *1494410783*
`$__unixEpochTo()`	Will be replaced by the end of the currently active time selection as Unix timestamp. For example, *1494497183*
`$__unixEpochNanoFilter(dateColumn)`	Will be replaced by a time range filter using the specified column name with times represented as nanosecond timestamp. For example, *dateColumn > 1494410783152415214 AND dateColumn < 1494497183142514872*
`$__unixEpochNanoFrom()`	Will be replaced by the start of the currently active time selection as nanosecond timestamp. For example, *1494410783152415214*
`$__unixEpochNanoTo()`	Will be replaced by the end of the currently active time selection as nanosecond timestamp. For example, *1494497183142514872*
`$__unixEpochGroup(dateColumn,'5m', [fillmode])`	Same as $__timeGroup but for times stored as Unix timestamp (`fillMode` only works with time series queries).
`$__unixEpochGroupAlias(dateColumn,'5m', [fillmode])`	Same as above but also adds a column alias (`fillMode` only works with time series queries).

We plan to add many more macros. If you have suggestions for what macros you would like to see, please open an issue in our GitHub repo.

Table queries

If the `Format as` query option is set to `Table` then you can basically do any type of SQL query. The table panel will automatically show the results of whatever columns and rows your query returns.

Query editor with example query:

The query:

```sql
SELECT
  title as "Title",
  "user".login as "Created By",
  dashboard.created as "Created On"
FROM dashboard
INNER JOIN "user" on "user".id = dashboard.created_by
WHERE $__timeFilter(dashboard.created)
```

Copy

You can control the name of the Table panel columns by using regular `as` SQL column selection syntax.

The resulting table panel:

Time series queries

If you set Format as to *Time series*, then the query must have a column named time that returns either a SQL datetime or any numeric datatype representing Unix epoch in seconds. In addition, result sets of time series queries must be sorted by time for panels to properly visualize the result.

A time series query result is returned in a wide data frame format. Any column except time or of type string transforms into value fields in the data frame query result. Any string column transforms into field labels in the data frame query result.

For backward compatibility, there's an exception to the above rule for queries that return three columns including a string column named metric. Instead of transforming the metric column into field labels, it becomes the field name, and then the series name is formatted as the value of the metric column. See the example with the metric column below.

To optionally customize the default series name formatting, refer to Standard options definitions.

Example with `metric` column:

sql Copy

```sql
SELECT
  $__timeGroupAlias("time_date_time",'5m'),
  min("value_double"),
  'min' as metric
FROM test_data
WHERE $__timeFilter("time_date_time")
GROUP BY time
ORDER BY time
```

Data frame result:

text Copy

```
+---------------------+-----------------+
| Name: time          | Name: min       |
| Labels:             | Labels:         |
| Type: []time.Time   | Type: []float64 |
+---------------------+-----------------+
| 2020-01-02 03:05:00 | 3               |
| 2020-01-02 03:10:00 | 6               |
+---------------------+-----------------+
```

Example using the fill parameter in the $__timeGroupAlias macro to convert null values to be zero instead:

sql Copy

```sql
SELECT
  $__timeGroupAlias("createdAt",'5m',0),
  sum(value) as value,
  hostname
FROM test_data
```

```
WHERE
  $__timeFilter("createdAt")
GROUP BY time, hostname
ORDER BY time
```

Given the data frame result in the following example and using the graph panel, you will get two series named *value 10.0.1.1* and *value 10.0.1.2*. To render the series with a name of *10.0.1.1* and *10.0.1.2* , use a Standard options definitions display value of `${__field.labels.hostname}`.

Data frame result:

text Copy

```
+---------------------+---------------------------+---------------------------+
| Name: time          | Name: value               | Name: value               |
| Labels:             | Labels: hostname=10.0.1.1 | Labels: hostname=10.0.1.2 |
| Type: []time.Time   | Type: []float64           | Type: []float64           |
+---------------------+---------------------------+---------------------------+
| 2020-01-02 03:05:00 | 3                         | 4                         |
| 2020-01-02 03:10:00 | 6                         | 7                         |
+---------------------+---------------------------+---------------------------+
```

Example with multiple columns:

sql Copy

```
SELECT
  $__timeGroupAlias("time_date_time",'5m'),
  min("value_double") as "min_value",
  max("value_double") as "max_value"
FROM test_data
WHERE $__timeFilter("time_date_time")
GROUP BY time
ORDER BY time
```

Data frame result:

text Copy

```
+---------------------+----------------+----------------+
| Name: time          | Name: min_value | Name: max_value |
| Labels:             | Labels:         | Labels:         |
| Type: []time.Time   | Type: []float64 | Type: []float64 |
+---------------------+----------------+----------------+
| 2020-01-02 03:04:00 | 3               | 4               |
```

```
| 2020-01-02 03:05:00 | 6                | 7                |
+---------------------+------------------+------------------+
```

Templating

Instead of hard-coding things like server, application and sensor name in your metric queries you can use variables in their place. Variables are shown as dropdown select boxes at the top of the dashboard. These dropdowns make it easy to change the data being displayed in your dashboard.

Refer to Templates and variables for an introduction to the templating feature and the different types of template variables.

Query variable

If you add a template variable of the type `Query`, you can write a PostgreSQL query that can return things like measurement names, key names or key values that are shown as a dropdown select box.

For example, you can have a variable that contains all values for the `hostname` column in a table if you specify a query like this in the templating variable *Query* setting.

```sql
SELECT hostname FROM host
```

A query can return multiple columns and Grafana will automatically create a list from them. For example, the query below will return a list with values from `hostname` and `hostname2`.

```sql
SELECT host.hostname, other_host.hostname2 FROM host JOIN other_host ON host.city = other_host.cit
```

To use time range dependent macros like `$__timeFilter(column)` in your query the refresh mode of the template variable needs to be set to *On Time Range Change*.

```sql
SELECT event_name FROM event_log WHERE $__timeFilter(time_column)
```

Another option is a query that can create a key/value variable. The query should return two columns that are named `__text` and `__value`. The `__text` column value should be unique (if it is not unique then the first value is used). The options in the dropdown will have a text and value that allows you to have a friendly name as text and an id as the value. An example query with `hostname` as the text and `id` as the value:

```sql
```

```
SELECT hostname AS __text, id AS __value FROM host
```

You can also create nested variables. Using a variable named `region`, you could have the hosts variable only show hosts from the current selected region with a query like this (if `region` is a multi-value variable then use the `IN` comparison operator rather than `=` to match against multiple values):

sql Copy

```
SELECT hostname FROM host  WHERE region IN($region)
```

Using `__searchFilter` to filter results in Query Variable

Available from Grafana 6.5 and above

Using `__searchFilter` in the query field will filter the query result based on what the user types in the dropdown select box. When nothing has been entered by the user the default value for `__searchFilter` is `%`.

Important that you surround the `__searchFilter` expression with quotes as Grafana does not do this for you.

The example below shows how to use `__searchFilter` as part of the query field to enable searching for `hostname` while the user types in the dropdown select box.

Query

sql Copy

```
SELECT hostname FROM my_host  WHERE hostname LIKE '$__searchFilter'
```

Using Variables in Queries

From Grafana 4.3.0 to 4.6.0, template variables are always quoted automatically. If your template variables are strings, do not wrap them in quotes in where clauses.

From Grafana 4.7.0, template variable values are only quoted when the template variable is a `multi-value`.

If the variable is a multi-value variable then use the `IN` comparison operator rather than `=` to match against multiple values.

There are two syntaxes:

`$<varname>` Example with a template variable named `hostname`:

```sql
SELECT
  atimestamp as time,
  aint as value
FROM table
WHERE $__timeFilter(atimestamp) and hostname in($hostname)
ORDER BY atimestamp ASC
```

`[[varname]]` Example with a template variable named `hostname`:

```sql
SELECT
  atimestamp as time,
  aint as value
FROM table
WHERE $__timeFilter(atimestamp) and hostname in([[hostname]])
ORDER BY atimestamp ASC
```

Disabling quoting for multi-value variables

Grafana automatically creates a quoted, comma-separated string for multi-value variables. For example: if `server01` and `server02` are selected then it will be formatted as: `'server01', 'server02'`. To disable quoting, use the csv formatting option for variables:

`${servers:csv}`

Read more about variable formatting options in the Variables documentation.

Annotations

Annotations allow you to overlay rich event information on top of graphs. You add annotation queries via the Dashboard menu / Annotations view.

Example query using time column with epoch values:

```sql
SELECT
  epoch_time as time,
  metric1 as text,
```

```
  concat_ws(', ', metric1::text, metric2::text) as tags
FROM
  public.test_data
WHERE
  $__unixEpochFilter(epoch_time)
```

Example region query using time and timeend columns with epoch values:

Only available in Grafana v6.6+.

sql Copy

```sql
SELECT
  epoch_time as time,
  epoch_time_end as timeend,
  metric1 as text,
  concat_ws(', ', metric1::text, metric2::text) as tags
FROM
  public.test_data
WHERE
  $__unixEpochFilter(epoch_time)
```

Example query using time column of native SQL date/time data type:

sql Copy

```sql
SELECT
  native_date_time as time,
  metric1 as text,
  concat_ws(', ', metric1::text, metric2::text) as tags
FROM
  public.test_data
WHERE
  $__timeFilter(native_date_time)
```

Name	Description
`time`	The name of the date/time field. Could be a column with a native SQL date/time data type or epoch value.

Name	Description
`timeend`	Optional name of the end date/time field. Could be a column with a native SQL date/time data type or epoch value. (Grafana v6.6+)
`text`	Event description field.
`tags`	Optional field name to use for event tags as a comma separated string.

Alerting

Time series queries should work in alerting conditions. Table formatted queries are not yet supported in alert rule conditions.

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

Microsoft SQL Server data source

Grafana ships with built-in support for Microsoft SQL Server (MS SQL). You can query and visualize data from any Microsoft SQL Server 2005 or newer, including Microsoft Azure SQL Database.

This topic explains configuration specific to the Microsoft SQL Server data source.

For instructions on how to add a data source to Grafana, refer to the administration documentation. Only users with the organization administrator role can add data sources. Administrators can also configure the data source via YAML with Grafana's provisioning system.

Once you've added the Microsoft SQL Server data source, you can configure it so that your Grafana instance's users can create queries in its query editor when they build dashboards and use Explore.

Configure the data source

To configure basic settings for the data source, complete the following steps:

1. Click **Connections** in the left-side menu.
2. Under Your connections, click **Data sources**.
3. Enter `Microsoft SQL Server` in the search bar.
4. Select **Microsoft SQL Server**.

   The **Settings** tab of the data source is displayed.
5. Set the data source's basic configuration options:

Name	Description
Name	Sets the name you use to refer to the data source in panels and queries.

Name	Description
Default	Sets the data source that's pre-selected for new panels.
Host	Sets the IP address/hostname and optional port of your MS SQL instance. Default port is 0, the driver default. You can specify multiple connection properties, such as `ApplicationIntent`, by separating each property with a semicolon (`;`).
Database	Sets the name of your MS SQL database.
Authentication	Sets the authentication mode, either using SQL Server Authentication or Windows Authentication (single sign-on for Windows users).
User	Defines the database user's username.
Password	Defines the database user's password.
Encrypt	Determines whether or to which extent a secure SSL TCP/IP connection will be negotiated with the server. Options include: `disable` - data sent between client and server is not encrypted; `false` - data sent between client and server is not encrypted beyond the login packet; `true` - data sent between client and server is encrypted. Default is `false`.
Max open	Sets the maximum number of open connections to the database. Default is `100`.
Max idle	Sets the maximum number of connections in the idle connection pool. Default is `100`.
Auto (max idle)	If set will set the maximum number of idle connections to the number of maximum open connections (Grafana v9.5.1+). Default is `true`.
Max lifetime	Sets the maximum number of seconds that the data source can reuse a connection. Default is `14400` (4 hours).

You can also configure settings specific to the Microsoft SQL Server data source. These options are described in the sections below.

Min time interval

The **Min time interval** setting defines a lower limit for the `$__interval` and [`$__interval_ms`][add-template-variables-interval_ms] variables.

This value *must* be formatted as a number followed by a valid time identifier:

Identifier	Description
`y`	year
`M`	month
`w`	week
`d`	day
`h`	hour
`m`	minute

Identifier	Description
`s`	second
`ms`	millisecond

We recommend setting this value to match your Microsoft SQL Server write frequency. For example, use `1m` if Microsoft SQL Server writes data every minute.

You can also override this setting in a dashboard panel under its data source options.

Connection timeout

The **Connection timeout** setting defines the maximum number of seconds to wait for a connection to the database before timing out. Default is 0 for no timeout.

Database user permissions

Grafana doesn't validate that a query is safe, and could include any SQL statement. For example, Microsoft SQL Server would execute destructive queries like `DELETE FROM user;` and `DROP TABLE user;` if the querying user has permission to do so.

To protect against this, we strongly recommend that you create a specific MS SQL user with restricted permissions.

Grant only `SELECT` permissions on the specified database and tables that you want to query to the database user you specified when you added the data source:

sql Copy

```sql
CREATE USER grafanareader WITH PASSWORD 'password'
GRANT SELECT ON dbo.YourTable3 TO grafanareader
```

Also, ensure that the user doesn't have any unwanted privileges from the public role.

Diagnose connection issues

If you use older versions of Microsoft SQL Server, such as 2008 and 2008R2, you might need to disable encryption before you can connect the data source.

We recommend that you use the latest available service pack for optimal compatibility.

Provision the data source

You can define and configure the data source in YAML files as part of Grafana's provisioning system. For more information about provisioning, and for available configuration options, refer to Provisioning Grafana.

Provisioning example

yaml Copy

```yaml
apiVersion: 1

datasources:
  - name: MSSQL
    type: mssql
    url: localhost:1433
    user: grafana
    jsonData:
      database: grafana
      maxOpenConns: 100 # Grafana v5.4+
      maxIdleConns: 100 # Grafana v5.4+
      maxIdleConnsAuto: true # Grafana v9.5.1+
      connMaxLifetime: 14400 # Grafana v5.4+
      connectionTimeout: 0 # Grafana v9.3+
      encrypt: 'false'
    secureJsonData:
      password: 'Password!'
```

Query the data source

You can create queries with the Microsoft SQL Server data source's query editor when editing a panel that uses a MS SQL data source.

For details, refer to the query editor documentation.

Use template variables

Instead of hard-coding details such as server, application, and sensor names in metric queries, you can use variables. Grafana lists these variables in dropdown select boxes at the top of the dashboard to help you change the data displayed in your dashboard. Grafana refers to such variables as template variables.

For details, see the template variables documentation.

Important: This documentation is about an older version. It's relevant only to the release noted, many of the features and functions have been updated or replaced. Please view the current version.

OpenTSDB data source

Grafana ships with advanced support for OpenTSDB. This topic explains configuration, variables, querying, and other features specific to the OpenTSDB data source.

For instructions on how to add a data source to Grafana, refer to the administration documentation. Only users with the organization administrator role can add data sources. Administrators can also configure the data source via YAML with Grafana's provisioning system.

OpenTSDB settings

To configure basic settings for the data source, complete the following steps:

1. Click **Connections** in the left-side menu.
2. Under Your connections, click **Data sources**.
3. Enter `OpenTSDB` in the search bar.
4. Select **OpenTSDB**.

   The **Settings** tab of the data source is displayed.
5. Set the data source's basic configuration options:

Name	Description
Name	The data source name. This is how you refer to the data source in panels and queries.
Default	Default data source that will be be pre-selected for new panels.
URL	The HTTP protocol, IP, and port of your OpenTSDB server (default port is usually 4242).

Name	Description
Allowed cookies	Listing of cookies to forward to the data source.
Version	The OpenTSDB version.
Resolution	Metrics from OpenTSDB may have data points with either second or millisecond resolution.
Lookup limit	Default is 1000.

Provision the data source

You can define and configure the data source in YAML files as part of Grafana's provisioning system. For more information about provisioning, and for available configuration options, refer to Provisioning Grafana.

Provisioning example

yaml Copy

```yaml
apiVersion: 1

datasources:
  - name: OpenTSDB
    type: opentsdb
    access: proxy
    url: http://localhost:4242
    jsonData:
      tsdbResolution: 1
      tsdbVersion: 1
```

Query editor

Open a graph in edit mode by click the title. Query editor will differ if the data source has version <=2.1 or = 2.2. In the former version, only tags can be used to query OpenTSDB. But in the latter version, filters as well as tags can be used to query OpenTSDB. Fill Policy is also introduced in OpenTSDB 2.2.

> **NOTE**
>
> While using OpenTSDB 2.2 data source, make sure you use either Filters or Tags as they are mutually exclusive. If used together, might give you weird results.

Auto complete suggestions

As soon as you start typing metric names, tag names and tag values , you should see highlighted auto complete suggestions for them. The autocomplete only works if the OpenTSDB suggest API is enabled.

Templating queries

Instead of hard-coding things like server, application and sensor name in your metric queries you can use variables in their place. Variables are shown as dropdown select boxes at the top of the dashboard. These dropdowns make it easy to change the data being displayed in your dashboard.

Check out the Templating documentation for an introduction to the templating feature and the different types of template variables.

Query variable

Grafana's OpenTSDB data source supports template variable queries. This means you can create template variables that fetch the values from OpenTSDB. For example, metric names, tag names, or tag values.

When using OpenTSDB with a template variable of `query` type you can use following syntax for lookup.

Query	Description
`metrics(prefix)`	Returns metric names with specific prefix (can be empty)
`tag_names(cpu)`	Returns tag names (i.e. keys) for a specific cpu metric
`tag_values(cpu, hostname)`	Returns tag values for metric cpu and tag key hostname
`suggest_tagk(prefix)`	Returns tag names (i.e. keys) for all metrics with specific prefix (can be empty)
`suggest_tagv(prefix)`	Returns tag values for all metrics with specific prefix (can be empty)

If you do not see template variables being populated in `Preview of values` section, you need to enable `tsd.core.meta.enable_realtime_ts` in the OpenTSDB server settings. Also, to populate metadata of the existing time series data in OpenTSDB, you need to run `tsdb uid metasync` on the OpenTSDB server.

Nested templating

One template variable can be used to filter tag values for another template variable. First parameter is the metric name, second parameter is the tag key for which you need to find tag values, and after that all other dependent template variables. Some examples are mentioned below to make nested template queries work successfully.

Query	Description
`tag_values(cpu, hostname, env=$env)`	Return tag values for cpu metric, selected env tag value and tag key hostname
`tag_values(cpu, hostname, env=$env, region=$region)`	Return tag values for cpu metric, selected env tag value, selected region tag value and tag key hostname

For details on OpenTSDB metric queries, check out the official OpenTSDB documentation