Contents

Jump to...

PgBouncer FAQ

How to connect to PgBouncer?

PgBouncer acts as a Postgres server, so simply point your client to the PgBouncer port.

How to load-balance queries between several servers?

PgBouncer does not have an internal multi-host configuration. It is possible via external tools:

1. DNS round-robin. Use several IPs behind one DNS name. PgBouncer does not look up DNS each time a new connection is launched. Instead, it caches all IPs and does round-robin internally. Note: if there are more than 8 IPs behind one name, the DNS backend must support the EDNS0 protocol. See README for details.

2. Use a TCP connection load-balancer. Either LVS or HAProxy seem to be good choices. On the PgBouncer side it may be a good idea to make `server_lifetime` smaller and also turn `server_round_robin` on: by default, idle connections are reused by a LIFO algorithm, which may work not so well when load-balancing is needed.

How to failover

1. DNS reconfiguration: When the IP address behind a DNS name is reconfigured, PgBouncer will reconnect to the new server. This behaviour can be tuned by two configuration parameters: `dns_max_ttl` tunes the lifetime for one host name, and `dns_zone_check_period` tunes how often a zone SOA will be queried for changes. If a zone SOA record has changed, PgBouncer will re-query all host names under that zone.

2. Write a new host to the configuration and let PgBouncer reload it: send SIGHUP or use the `RELOAD` command on the console. PgBouncer will detect a changed host configuration and reconnect to the new server.

3. Use the `RECONNECT` command. This is meant for situations where neither of the two options above are applicable, for example when you use the aforementioned HAProxy to route connections downstream from PgBouncer. `RECONNECT` simply causes all server connections to be reopened. So run that after that other component has changed its connection routing information.

How to use prepared statements with session pooling?

In session pooling mode, the reset query must clean old prepared statements. This can be achieved by `server_reset_query = DISCARD ALL;` or at least to `DEALLOCATE ALL;`

How to use prepared statements with transaction pooling?

Since version 1.21.0 PgBouncer can track prepared statements in transaction pooling mode and make sure they get prepared on-the-fly on the linked server connection. To enable this feature, `max_prepared_statements` needs to be set to a non-zero value. See the docs for `max_prepared_statements` for more details.

Due to the way PHP/PDO uses prepared statements (#991) the prepared statement support in PgBouncer 1.21.0 does not work for PHP/PDO. So for PHP/PDO and PgBouncer versions before 1.21.0 the only work-around is to disable prepared statements in the client side.

prepareThreshold=0 parameter to the connection string.

Disabling prepared statements in PHP/PDO

To disable use of server-side prepared statements, the PDO attribute `PDO::ATTR_EMULATE_PREPARES` must be set to `true`. Either at connect-time:

```
$db = new PDO("dsn", "user", "pass", array(PDO::ATTR_EMULATE_
```

or later:

```
$db->setAttribute(PDO::ATTR_EMULATE_PREPARES, true);
```

How to upgrade PgBouncer without dropping connections?

DEPRECATED: Instead of this option use a rolling restart with multiple pgbouncer processes listening on the same port using so_reuseport instead

This is as easy as launching a new PgBouncer process with the `-R` switch and the same configuration:

```
$ pgbouncer -R -d config.ini
```

The `-R` (reboot) switch makes the new process connect to the console of the old process (dbname=pgbouncer) via the Unix socket and issue the following commands:

```
SUSPEND;
SHOW FDS;
SHUTDOWN;
```

After that, if the new one notices that the old one is gone, it resumes work with the old connections. The magic happens during the `SHOW FDS` command which transports the actual file descriptors to new process.

If the takeover does not work for whatever reason, the new process can be simply killed. The old one notices this and resumes work.

How to know which client is on which server connection?

Use the `SHOW CLIENTS` and `SHOW SERVERS` commands on the

1. Use `ptr` and `link` to map local client connection to server connection.
2. Use `addr` and `port` of client connection to identify TCP connection from client.
3. Use `local_addr` and `local_port` to identify TCP connection to server.

Should PgBouncer be installed on the web server or database server?

It depends.

Installing PgBouncer on the web server is good when short-lived connections are used. Then the connection setup latency is minimised. (TCP requires a couple of packet roundtrips before a connection is usable.) Installing PgBouncer on the database server is good when there are many different hosts (e.g., web servers) connecting to it. Then their connections can be optimised together.

It is also possible to install PgBouncer on both web server and database server. One negative aspect of that is that each PgBouncer hop adds a small amount of latency to each query.

In the end, you will need to test which model works best for your performance needs. You should also consider how installing PgBouncer will affect the failover of your applications in the event of a web server vs. database server going away.

pgbouncer.ini

Description

The configuration file is in "ini" format. Section names are between "[" and "]". Lines starting with ";" or "#" are taken as comments and ignored. The characters ";" and "#" are not recognized as special when they appear later in the line.

Generic settings

logfile

Specifies the log file. For daemonization (`-d`), either this or `syslog` need to be set.

The log file is kept open, so after rotation, `kill -HUP` or on console `RELOAD;` should be done. On Windows, the service must be stopped and started.

Note that setting `logfile` does not by itself turn off logging to stderr. Use the command-line option `-q` or `-d` for that.

Default: not set

pidfile

Specifies the PID file. Without `pidfile` set, daemonization (`-d`) is not allowed.

Default: not set

listen_addr

Specifies a list (comma-separated) of addresses where to listen for TCP connections. You may also use `*` meaning "listen on all addresses". When not set, only Unix socket connections are accepted.

Addresses can be specified numerically (IPv4/IPv6) or by name.

Default: not set

listen_port

Which port to listen on. Applies to both TCP and Unix sockets.

Default: 6432

unix_socket_dir

Specifies the location for Unix sockets. Applies to both the listening socket and to server connections. If set to an empty string, Unix sockets are disabled. A value that starts with `@` specifies that a Unix socket in the abstract namespace should be created (currently supported on Linux and Windows).

For online reboot (`-R`) to work, a Unix socket needs to be configured, and it needs to be in the file-system namespace.

Default: `/tmp` (empty on Windows)

unix_socket_mode

File system mode for Unix socket. Ignored for sockets in the abstract namespace. Not supported on Windows.

Default: 0777

unix_socket_group

Group name to use for Unix socket. Ignored for sockets in the abstract namespace. Not supported on Windows.

Default: not set

user

If set, specifies the Unix user to change to after startup. Works only if PgBouncer is started as root or if it's already running as the given user. Not supported on Windows.

Default: not set

pool_mode

Specifies when a server connection can be reused by other clients.

session
: Server is released back to pool after client disconnects. Default.

transaction
: Server is released back to pool after transaction finishes.

statement
: Server is released back to pool after query finishes. Transactions spanning multiple statements are disallowed in this mode.

max_client_conn

Maximum number of client connections allowed.

When this setting is increased, then the file descriptor limits in the operating system might also have to be increased. Note that the number of file descriptors potentially used is more than `max_client_conn`. If each user connects under its own user name to the server, the theoretical maximum used is:

```
max_client_conn + (max pool_size * total databases * total users)
```

If a database user is specified in the connection string (all users connect under the same user name), the theoretical maximum is:

```
max_client_conn + (max pool_size * total databases)
```

The theoretical maximum should never be reached, unless somebody deliberately crafts a special load for it. Still, it means you should set the number of file descriptors to a safely high number.

Search for `ulimit` in your favorite shell man page. Note: `ulimit` does not apply in a Windows environment.

Default: 100

default_pool_size

How many server connections to allow per user/database pair. Can be overridden in the per-database configuration.

Default: 20

min_pool_size

Add more server connections to pool if below this number. Improves behavior when the normal load suddenly comes back after a period of total inactivity. The value is effectively capped at the pool size.

Only enforced for pools where at least one of the following is true:

- the entry in the `[database]` section for the pool has a value set for the `user` key (aka forced user)
- there is at least one client connected to the pool

Default: 0 (disabled)

reserve_pool_size

How many additional connections to allow to a pool (see `reserve_pool_timeout`). 0 disables.

Default: 0 (disabled)

reserve_pool_timeout

If a client has not been serviced in this time, use additional connections from the reserve pool. 0 disables. [seconds]

Default: 5.0

max_db_connections

Do not allow more than this many server connections per database (regardless of user). This considers the PgBouncer database that the client has connected to, not the PostgreSQL database of the outgoing connection.

This can also be set per database in the `[databases]` section.

Note that when you hit the limit, closing a client connection to one pool will not immediately allow a server connection to be established for another pool, because the server connection for the first pool is still open. Once the server connection closes (due to idle timeout), a new server connection will immediately be opened for the waiting pool.

Default: 0 (unlimited)

max_user_connections

Do not allow more than this many server connections per user (regardless of database). This considers the PgBouncer user that is associated with a pool, which is either the user specified for the server connection or in absence of that the user the client has connected as.

This can also be set per user in the `[users]` section.

Note that when you hit the limit, closing a client connection to one pool will not immediately allow a server connection to be established for another pool, because the server connection for the first pool is still open. Once the server connection closes (due to idle timeout), a new server connection will immediately be opened for the waiting pool.

Default: 0 (unlimited)

server_round_robin

By default, PgBouncer reuses server connections in LIFO (last-in, first-out) manner, so that few connections get the most load. This gives best performance if you have a single server serving a database. But if there is a round-robin system behind a database address (TCP, DNS, or host list), then it is better if PgBouncer also uses connections in that manner, thus achieving uniform load.

Default: 0

track_extra_parameters

By default, PgBouncer tracks `client_encoding`, `datestyle`, `timezone`, `standard_conforming_strings` and `application_name` parameters per client. To allow other parameters to be tracked, they can be specified here, so that PgBouncer knows that they should be maintained in the client variable cache and restored in the server whenever the client becomes active.

If you need to specify multiple values, use a comma-separated list (e.g. `default_transaction_readonly, IntervalStyle`)

Note: Most parameters cannot be tracked this way. The only parameters that can be tracked are ones that Postgres reports to the client. Postgres has an official list of parameters that it reports to the client (https://www.postgresql.org/docs/15/protocol-flow.html#PROTOCOL-ASYNC). Postgres extensions can change this list though, they can add parameters themselves that they also report, and they can start reporting already existing parameters that Postgres does not report. Notably Citus 12.0+ causes Postgres to also report `search_path`.

The Postgres protocol allows specifying parameters settings, both directly as a parameter in the startup packet, or inside the `options` startup packet (https://www.postgresql.org/docs/current/libpq-connect.html#LIBPQ-CONNECT-OPTIONS). Parameters specified using both of these methods are supported by `track_extra_parameters`. However, it's not possible to include `options` itself in `track_extra_parameters`, only the parameters contained in `options`.

Default: IntervalStyle

ignore_startup_parameters

By default, PgBouncer allows only parameters it can keep track of in startup packets: `client_encoding`, `datestyle`, `timezone` and `standard_conforming_strings`. All others parameters will raise an error. To allow others parameters, they can be specified here, so that PgBouncer knows that they are handled by the admin and it can ignore them.

If you need to specify multiple values, use a comma-separated list (e.g. `options,extra_float_digits`)

The Postgres protocol allows specifying parameters settings, both directly as a parameter in the startup packet, or inside the `options` startup packet (https://www.postgresql.org/docs/current/libpq-connect.html#LIBPQ-CONNECT-OPTIONS). Parameters specified using both of these methods are supported by `ignore_startup_parameters`. It's even possible to include `options` itself in `track_extra_parameters`, which results in any unknown parameters contained inside `options` to be ignored.

Default: empty

peer_id

The peer id used to identify this PgBouncer process in a group of PgBouncer processes that are peered together. The `peer_id` value should be unique within a group of peered PgBouncer processes. When set to 0 pgbouncer peering is disabled. See the docs for the `[peers]` section for more information. The maximum value that can be used for the `peer_id` is 16383.

Default: 0

disable_pqexec

Disable the Simple Query protocol (PQexec). Unlike the Extended Query protocol, Simple Query allows multiple queries in one packet, which allows some classes of SQL-injection attacks. Disabling it can improve security. Obviously, this means only clients that exclusively use the Extended Query protocol will stay working.

Default: 0

application_name_add_host

Add the client host address and port to the application name setting set on connection start. This helps in identifying the source of bad queries etc. This logic applies only at the start of a connection. If `application_name` is later changed with `SET`, PgBouncer does not change it again.

Default: 0

conffile

Show location of current config file. Changing it will make PgBouncer use another config file for next `RELOAD` / `SIGHUP`.

Default: file from command line

service_name

Used on win32 service registration.

Default: `pgbouncer`

job_name

Alias for `service_name`.

stats_period

Sets how often the averages shown in various `SHOW` commands are updated and how often aggregated statistics are written to the log (but see `log_stats`). [seconds]

Default: 60

max_prepared_statements

When this is set to a non-zero value PgBouncer tracks protocol-level named prepared statements related commands sent by the client in transaction and statement pooling mode. PgBouncer makes sure that any statement prepared by a client is available on the backing server connection. Even when the statement was originally prepared on another server connection.

PgBouncer internally examines all the queries that are sent as a prepared statement by clients and gives each unique query string an internal name with the format `PGBOUNCER_{unique_id}`. If the same query string is prepared multiple times (possibly by different clients), then these queries share the same internal name. PgBouncer only prepares the statement on the actual PostgreSQL server using the internal name (so not the name provided by the client). PgBouncer keeps track of the name that the client gave to each prepared statement. It then rewrites each command that uses a prepared statement to by replacing the client side name with the the internal name (e.g. replacing `my_prepared_stamenent` with `PGBOUNCER_123`) before forwarding that command to the server. More importantly, if the prepared statement that the client wants to execute is not yet prepared on the server (e.g. because a different server is now assigned to the client then when the client prepared the statement), then PgBouncer transparrently prepares the statement before executing it.

Note: This tracking and rewriting of prepared statement commands does not work for SQL-level prepared statement commands, so `PREPARE`, `EXECUTE` and `DEALLOCATE` are forwarded straight to Postgres. The exception to this rule are the `DEALLOCATE ALL` and `DISCARD ALL` commands, these do work as expected and will clear the prepared statements that PgBouncer tracked for the client that sends this command.

The actual value of this setting controls the number of prepared statements kept active in an LRU cache on a single server connection. When the setting is set to 0 prepared statement support for transaction and statement pooling is disabled. To get the best performance you should try to make sure that this setting is larger than the amount of commonly used prepared statements in your application. Keep in mind that the higher this value, the larger the memory footprint of each PgBouncer connection will be on your PostgreSQL server, because it will keep more queries prepared on those connections. It also increases the memory footprint of PgBouncer itself, because it now needs to keep track of query strings.

The impact on PgBouncer memory usage is not that big though:

- Each unique query is stored once in a global query cache.
- Each client connection keeps a buffer that it uses to rewrite packets. This is at most 4 times the size of `pkt_buf`. This limit is often not reached though, it only happens when the queries in your prepared statements are between 2 and 4 times the size of `pkt_buf`.

So if you consider the following as an example scenario:

- There are 1000 active clients
- The clients prepare 200 unique queries
- The average size of a query is 5kB
- `pkt_buf` parameter is set to the default of 4096 (4kB)

Then PgBouncer needs at most the following amount of memory to handle these prepared statements:

200 x 5kB + 1000 x 4 x 4kB = ~17MB of memory.

Tracking prepared statements does not only come with a memory cost, but also with increased CPU usage, because PgBouncer needs to inspect and rewrite the queries. Multiple PgBouncer instances can listen on the same port to use more than one core for processing, see the documentation for the `so_reuseport` option (/config.html#so_reuseport) for details.

But of course there are also performance benefits to prepared statements. Just as when connecting to PostgreSQL directly, by preparing a query that is executed many times, it reduces the total amount of parsing and planning that needs to be done. The way that PgBouncer tracks prepared statements is especially beneficial to performance when multiple clients prepare the same queries. Because client connections automatically reuse a prepared statement on a server connection even if it was prepared by another client. As an example if you have a `pool_size` of 20 and you have 100 clients that all prepare the exact same query, then the query is prepared (and thus parsed) only 20 times on the PostgreSQL server.

The reuse of prepared statements has one downside. If the return or argument types of a prepared statement changes across executions then PostgreSQL currently throws an error such as:

```
ERROR:  cached plan must not change result type
```

You can avoid such errors by not having multiple clients that use the exact same query string in a prepared statement, but expecting different argument or result types. One of the most common ways of running into this issue is during a DDL migration where you add a new column or change a column type on an existing table. In those cases you can run `RECONNECT` on the PgBouncer admin console after doing the migration to force a re-prepare of the query and make the error goes away.

Default: 0

Authentication settings

PgBouncer handles its own client authentication and has its own database of users. These settings control this.

auth_type

How to authenticate users.

cert
: Client must connect over TLS connection with a valid client certificate. The user name is then taken from the CommonName field from the certificate.

md5
: Use MD5-based password check. This is the default authentication method. `auth_file` may contain both MD5-encrypted and plain-text passwords. If `md5` is configured and a user has a SCRAM secret, then SCRAM authentication is used automatically instead.

scram-sha-256
: Use password check with SCRAM-SHA-256. `auth_file` has to contain SCRAM secrets or plain-text passwords.

plain
: The clear-text password is sent over the wire. Deprecated.

trust
: No authentication is done. The user name must still exist in `auth_file`.

any
: Like the `trust` method, but the user name given is ignored. Requires that all databases are configured to log in as a specific user. Additionally, the console database allows any user to log in as admin.

hba
: The actual authentication type is loaded from `auth_hba_file`. This allows different authentication methods for different access paths, for example: connections over Unix socket use the `peer` auth method, connections over TCP must use TLS.

pam
: PAM is used to authenticate users, `auth_file` is ignored. This method is not compatible with databases using the `auth_user` option. The service name reported to PAM is "pgbouncer". `pam` is not supported in the HBA configuration file.

auth_hba_file

HBA configuration file to use when `auth_type` is `hba`.

Default: not set

auth_file

The name of the file to load user names and passwords from. See section Authentication file format below about details.

Most authentication types (see above) require that either `auth_file` or `auth_user` be set; otherwise there would be no users defined.

Default: not set

auth_user

If `auth_user` is set, then any user not specified in `auth_file` will be queried through the `auth_query` query from pg_shadow in the database, using `auth_user`. The password of `auth_user` will be taken from `auth_file`. (If the `auth_user` does not require a password then it does not need to be defined in `auth_file`.)

Direct access to pg_shadow requires admin rights. It's preferable to use a non-superuser that calls a SECURITY DEFINER function instead.

Default: not set

auth_query

Query to load user's password from database.

Direct access to pg_shadow requires admin rights. It's preferable to use a non-superuser that calls a SECURITY DEFINER function instead.

Note that the query is run inside the target database. So if a function is used, it needs to be installed into each database.

Default: `SELECT usename, passwd FROM pg_shadow WHERE usename=$1`

auth_dbname

Database name in the `[database]` section to be used for authentication purposes. This option can be either global or overridden in the connection string if this parameter is specified.

Log settings

syslog

Toggles syslog on/off. On Windows, the event log is used instead.

Default: 0

syslog_ident

Under what name to send logs to syslog.

Default: `pgbouncer` (program name)

syslog_facility

Under what facility to send logs to syslog. Possibilities: `auth`, `authpriv`, `daemon`, `user`, `local0-7`.

Default: `daemon`

log_connections

Log successful logins.

Default: 1

log_disconnections

Log disconnections with reasons.

Default: 1

log_pooler_errors

Log error messages the pooler sends to clients.

Default: 1

log_stats

Write aggregated statistics into the log, every `stats_period`. This can be disabled if external monitoring tools are used to grab the same data from `SHOW` commands.

Default: 1

verbose

Increase verbosity. Mirrors the "-v" switch on the command line. For example, using "-v -v" on the command line is the same as `verbose=2`.

Default: 0

Console access control

admin_users

Comma-separated list of database users that are allowed to connect and run all commands on the console. Ignored when `auth_type` is `any`, in which case any user name is allowed in as admin.

Default: empty

stats_users

Comma-separated list of database users that are allowed to connect and run read-only queries on the console. That means all `SHOW` commands except `SHOW FDS`.

Default: empty

Connection sanity checks, timeouts

server_reset_query

Query sent to server on connection release, before making it available to other clients. At that moment no transaction is in progress, so the value should not include `ABORT` or `ROLLBACK`.

The query is supposed to clean any changes made to the database session so that the next client gets the connection in a well-defined state. The default is `DISCARD ALL`, which cleans everything, but that leaves the next client no pre-cached state. It can be made lighter, e.g. `DEALLOCATE ALL` to just drop prepared statements, if the application does not break when some state is kept around.

When transaction pooling is used, the `server_reset_query` is not used, because in that mode, clients must not use any session-based features, since each transaction ends up in a different connection and thus gets a different session state.

Default: `DISCARD ALL`

server_reset_query_always

Whether `server_reset_query` should be run in all pooling modes. When this setting is off (default), the `server_reset_query` will be run only in pools that are in sessions-pooling mode. Connections in transaction-pooling mode should not have any need for a reset query.

This setting is for working around broken setups that run applications that use session features over a transaction-pooled PgBouncer. It changes non-deterministic breakage to deterministic breakage: Clients always lose their state after each transaction.

Default: 0

server_check_delay

How long to keep released connections available for immediate re-use, without running `server_check_query` on it. If 0 then the check is always run.

Default: 30.0

server_check_query

Simple do-nothing query to check if the server connection is alive.

If an empty string, then sanity checking is disabled.

Default: `select 1`

server_fast_close

Disconnect a server in session pooling mode immediately or after the end of the current transaction if it is in "close_needed" mode (set by `RECONNECT`, `RELOAD` that changes connection settings, or DNS change), rather than waiting for the session end. In statement or transaction pooling mode, this has no effect since that is the default behavior there.

If because of this setting a server connection is closed before the end of the client session, the client connection is also closed. This ensures that the client notices that the session has been interrupted.

This setting makes connection configuration changes take effect sooner if session pooling and long-running sessions are used. The downside is that client sessions are liable to be interrupted by a configuration change, so client applications will need logic to reconnect and reestablish session state. But note that no transactions will be lost, because running transactions are not interrupted, only idle sessions.

Default: 0

server_lifetime

The pooler will close an unused (not currently linked to any client connection) server connection that has been connected longer than this. Setting it to 0 means the connection is to be used only once, then closed. [seconds]

Default: 3600.0

server_idle_timeout

If a server connection has been idle more than this many seconds it will be closed. If 0 then this timeout is disabled. [seconds]

Default: 600.0

server_connect_timeout

If connection and login don't finish in this amount of time, the connection will be closed. [seconds]

Default: 15.0

server_login_retry

If login to the server failed, because of failure to connect or from authentication, the pooler waits this much before retrying to connect. During the waiting interval, new clients trying to connect to the failing server will get an error immediately without another connection attempt. [seconds]

The purpose of this behavior is that clients don't unnecessarily queue up waiting for a server connection to become available if the server is not working. However, it also means that if a server is momentarily failing, for example during a restart or if the configuration was erroneous, then it will take at least this long until the pooler will consider connecting to it again. Planned events such as restarts should normally be managed using the `PAUSE` command to avoid this.

Default: 15.0

client_login_timeout

If a client connects but does not manage to log in in this amount of time, it will be disconnected. Mainly needed to avoid dead connections stalling `SUSPEND` and thus online restart. [seconds]

Default: 60.0

autodb_idle_timeout

If the automatically created (via "*") database pools have been unused this many seconds, they are freed. The negative aspect of that is that their statistics are also forgotten. [seconds]

Default: 3600.0

dns_max_ttl

How long DNS lookups can be cached. The actual DNS TTL is ignored. [seconds]

Default: 15.0

dns_nxdomain_ttl

How long DNS errors and NXDOMAIN DNS lookups can be cached. [seconds]

Default: 15.0

dns_zone_check_period

Period to check if a zone serial has changed.

PgBouncer can collect DNS zones from host names (everything after first dot) and then periodically check if the zone serial changes. If it notices changes, all host names under that zone are looked up again. If any host IP changes, its connections are invalidated.

Works only with c-ares backend (`configure` option `--with-cares`).

Default: 0.0 (disabled)

resolv_conf

The location of a custom `resolv.conf` file. This is to allow specifying custom DNS servers and perhaps other name resolution options, independent of the global operating system configuration.

Requires evdns (>= 2.0.3) or c-ares (>= 1.15.0) backend.

The parsing of the file is done by the DNS backend library, not PgBouncer, so see the library's documentation for details on allowed syntax and directives.

Default: empty (use operating system defaults)

TLS settings

client_tls_sslmode

TLS mode to use for connections from clients. TLS connections are disabled by default. When enabled, `client_tls_key_file` and `client_tls_cert_file` must be also configured to set up the key and certificate PgBouncer uses to accept client connections.

disable
: Plain TCP. If client requests TLS, it's ignored. Default.

allow
: If client requests TLS, it is used. If not, plain TCP is used. If the client presents a client certificate, it is not validated.

prefer
: Same as `allow`.

require
: Client must use TLS. If not, the client connection is rejected. If the client presents a client certificate, it is not validated.

verify-ca
: Client must use TLS with valid client certificate.

verify-full
: Same as `verify-ca`.

client_tls_key_file

Private key for PgBouncer to accept client connections.

Default: not set

client_tls_cert_file

Certificate for private key. Clients can validate it.

Default: not set

client_tls_ca_file

Root certificate file to validate client certificates.

Default: not set

client_tls_protocols

Which TLS protocol versions are allowed. Allowed values: `tlsv1.0`, `tlsv1.1`, `tlsv1.2`, `tlsv1.3`. Shortcuts: `all` (tlsv1.0,tlsv1.1,tlsv1.2,tlsv1.3), `secure` (tlsv1.2,tlsv1.3), `legacy` (all).

Default: `secure`

client_tls_ciphers

Allowed TLS ciphers, in OpenSSL syntax. Shortcuts:

- `default`/`secure`/`fast`/`normal` (these all use system wide OpenSSL defaults)
- `all` (enables all ciphers, not recommended)

Only connections using TLS version 1.2 and lower are affected. There is currently no setting that controls the cipher choices used by TLS version 1.3 connections.

Default: `default`

client_tls_ecdhcurve

Elliptic Curve name to use for ECDH key exchanges.

Allowed values: `none` (DH is disabled), `auto` (256-bit ECDH), curve name

Default: `auto`

client_tls_dheparams

DHE key exchange type.

Allowed values: `none` (DH is disabled), `auto` (2048-bit DH), `legacy` (1024-bit DH)

Default: `auto`

server_tls_sslmode

TLS mode to use for connections to PostgreSQL servers. The default mode is `prefer`.

disable
: Plain TCP. TLS is not even requested from the server.

allow
: FIXME: if server rejects plain, try TLS?

prefer
: TLS connection is always requested first from PostgreSQL. If refused, the connection will be established over plain TCP. Server certificate is not validated. Default

require

Connection must go over TLS. If server rejects it, plain TCP is not attempted. Server certificate is not validated.

verify-ca

Connection must go over TLS and server certificate must be valid according to `server_tls_ca_file`. Server host name is not checked against certificate.

verify-full

Connection must go over TLS and server certificate must be valid according to `server_tls_ca_file`. Server host name must match certificate information.

server_tls_ca_file

Root certificate file to validate PostgreSQL server certificates.

Default: not set

server_tls_key_file

Private key for PgBouncer to authenticate against PostgreSQL server.

Default: not set

server_tls_cert_file

Certificate for private key. PostgreSQL server can validate it.

Default: not set

server_tls_protocols

Which TLS protocol versions are allowed. Allowed values: `tlsv1.0`, `tlsv1.1`, `tlsv1.2`, `tlsv1.3`. Shortcuts: `all` (tlsv1.0,tlsv1.1,tlsv1.2,tlsv1.3), `secure` (tlsv1.2,tlsv1.3), `legacy` (all).

Default: `secure`

server_tls_ciphers

Allowed TLS ciphers, in OpenSSL syntax. Shortcuts:

- `default`/`secure`/`fast`/`normal` (these all use system wide OpenSSL defaults)
- `all` (enables all ciphers, not recommended)

Only connections using TLS version 1.2 and lower are affected. There is currently no setting that controls the cipher choices used by TLS version 1.3 connections.

Default: `default`

Dangerous timeouts

Setting the following timeouts can cause unexpected errors.

query_timeout

Queries running longer than that are canceled. This should be used only with a slightly smaller server-side `statement_timeout`, to apply only for network problems. [seconds]

Default: 0.0 (disabled)

query_wait_timeout

Maximum time queries are allowed to spend waiting for execution. If the query is not assigned to a server during that time, the client is disconnected. 0 disables. If this is disabled, clients will be queued indefinitely. [seconds]

This setting is used to prevent unresponsive servers from grabbing up connections. It also helps when the server is down or rejects connections for any reason.

Default: 120.0

cancel_wait_timeout

Maximum time cancellation requests are allowed to spend waiting for execution. If the cancel request is not assigned to a server during that time, the client is disconnected. 0 disables. If this is disabled, cancel requests will be queued indefinitely. [seconds]

This setting is used to prevent a client locking up when a cancel cannot be forwarded due to the server being down.

Default: 10.0

client_idle_timeout

Client connections idling longer than this many seconds are closed. This should be larger than the client-side connection lifetime settings, and only used for network problems. [seconds]

Default: 0.0 (disabled)

idle_transaction_timeout

If a client has been in "idle in transaction" state longer, it will be disconnected. [seconds]

Default: 0.0 (disabled)

suspend_timeout

How long to wait for buffer flush during `SUSPEND` or reboot (`-R`). A connection is dropped if the flush does not succeed. [seconds]

Default: 10

Low-level network settings

pkt_buf

Internal buffer size for packets. Affects size of TCP packets sent and general memory usage. Actual libpq packets can be larger than this, so no need to set it large.

Default: 4096

max_packet_size

Maximum size for PostgreSQL packets that PgBouncer allows through. One packet is either one query or one result set row. The full result set can be larger.

Default: 2147483647

listen_backlog

Backlog argument for listen(2). Determines how many new unanswered connection attempts are kept in the queue. When the queue is full, further new connections are dropped.

Default: 128

sbuf_loopcnt

How many times to process data on one connection, before proceeding. Without this limit, one connection with a big result set can stall PgBouncer for a long time. One loop processes one `pkt_buf` amount of data. 0 means no limit.

Default: 5

so_reuseport

Specifies whether to set the socket option `SO_REUSEPORT` on TCP listening sockets. On some operating systems, this allows running multiple PgBouncer instances on the same host listening on the same port and having the kernel distribute the connections automatically. This option is a way to get PgBouncer to use more CPU cores. (PgBouncer is single-threaded and uses one CPU core per instance.)

The behavior in detail depends on the operating system kernel. As of this writing, this setting has the desired effect on (sufficiently recent versions of) Linux, DragonFlyBSD, and FreeBSD. (On FreeBSD, it applies the socket option `SO_REUSEPORT_LB` instead.) Some other operating systems support the socket option but it won't have the desired effect: It will allow multiple processes to bind to the same port but only one of them will get the connections. See your operating system's setsockopt() documentation for details.

On systems that don't support the socket option at all, turning this setting on will result in an error.

Each PgBouncer instance on the same host needs different settings for at least `unix_socket_dir` and `pidfile`, as well as `logfile` if that is used. Also note that if you make use of this option, you can no longer connect to a specific PgBouncer instance via TCP/IP, which might have implications for monitoring and metrics collection.

To make sure query cancellations keep working, you should set up PgBouncer peering between the different PgBouncer processes. For details look at docs for the `peer_id` configuration option and the `peers` configuration section. There's also an example that uses peering and so_reuseport in the example section of these docs.

Default: 0

tcp_defer_accept

Sets the `TCP_DEFER_ACCEPT` socket option; see `man 7 tcp` for details. (This is a Boolean option: 1 means enabled. The actual value set if enabled is currently hardcoded to 45 seconds.)

This is currently only supported on Linux.

Default: 1 on Linux, otherwise 0

tcp_socket_buffer

Default: not set

tcp_keepalive

Turns on basic keepalive with OS defaults.

On Linux, the system defaults are tcp_keepidle=7200, tcp_keepintvl=75, tcp_keepcnt=9. They are probably similar on other operating systems.

Default: 1

tcp_keepcnt

Default: not set

tcp_keepidle

Default: not set

tcp_keepintvl

Default: not set

tcp_user_timeout

Sets the `TCP_USER_TIMEOUT` socket option. This specifies the maximum amount of time in milliseconds that transmitted data may remain unacknowledged before the TCP connection is forcibly closed. If set to 0, then operating system's default is used.

This is currently only supported on Linux.

Default: 0

Section [databases]

The section `[databases]` defines the names of the databases that clients of PgBouncer can connect to and specifies where those connections will be routed. The section contains key=value lines like

```
dbname = connection string
```

where the key will be taken as a database name and the value as a connection string, consisting of key=value pairs of connection parameters, described below (similar to libpq, but the actual libpq is not used and the set of available features is different). Example:

```
foodb = host=host1.example.com port=5432
bardb = host=localhost dbname=bazdb
```

The database name can contain characters `_0-9A-Za-z` without quoting. Names that contain other characters need to be quoted with standard SQL identifier quoting: double quotes, with "" for a single instance of a double quote.

The database name "pgbouncer" is reserved for the admin console and cannot be used as a key here.

"*" acts as a fallback database: If the exact name does not exist, its value is taken as connection string for the requested database. For example, if there is an entry (and no other overriding entries)

```
* = host=foo
```

then a connection to PgBouncer specifying a database "bar" will effectively behave as if an entry

```
bar = host=foo dbname=bar
```

exists (taking advantage of the default for `dbname` being the client-side database name; see below).

Such automatically created database entries are cleaned up if they stay idle longer than the time specified by the `autodb_idle_timeout` parameter.

dbname

Destination database name.

Default: same as client-side database name

host

Host name or IP address to connect to. Host names are resolved at connection time, the result is cached per `dns_max_ttl` parameter. When a host name's resolution changes, existing server connections are automatically closed when they are released (according to the pooling mode), and new server connections immediately use the new resolution. If DNS returns several results, they are used in a round-robin manner.

If the value begins with `/`, then a Unix socket in the file-system namespace is used. If the value begins with `@`, then a Unix socket in the abstract namespace is used.

A comma-separated list of host names or addresses can be specified. In that case, connections are made in a round-robin manner. (If a host list contains host names that in turn resolve via DNS to multiple addresses, the round-robin systems operate independently. This is an implementation dependency that is subject to change.) Note that in a list, all hosts must be available at all times: There are no mechanisms to skip unreachable hosts or to select only available hosts from a list or similar. (This is different from what a host list in libpq means.) Also note that this only affects how the destinations of new connections are chosen. See also the setting `server_round_robin` for how clients are assigned to already established server connections.

Examples:

```
host=localhost
host=127.0.0.1
host=2001:0db8:85a3:0000:0000:8a2e:0370:7334
host=/var/run/postgresql
host=192.168.0.1,192.168.0.2,192.168.0.3
```

Default: not set, meaning to use a Unix socket

port

Default: 5432

user

If `user=` is set, all connections to the destination database will be done with the specified user, meaning that there will be only one pool for this database.

Otherwise, PgBouncer logs into the destination database with the client user name, meaning that there will be one pool per user.

password

If no password is specified here, the password from the `auth_file` or `auth_query` will be used.

auth_user

Override of the global `auth_user` setting, if specified.

auth_query

Override of the global `auth_query` setting, if specified. The entire SQL statement needs to be enclosed in single quotes.

auth_dbname

Override of the global `auth_dbname` setting, if specified.

pool_size

Set the maximum size of pools for this database. If not set, the `default_pool_size` is used.

min_pool_size

Set the minimum pool size for this database. If not set, the global `min_pool_size` is used.

Only enforced if at least one of the following is true:

- this entry in the `[database]` section has a value set for the `user` key (aka forced user)
- there is at least one client connected to the pool

reserve_pool

Set additional connections for this database. If not set, `reserve_pool_size` is used.

connect_query

Query to be executed after a connection is established, but before allowing the connection to be used by any clients. If the query raises errors, they are logged but ignored otherwise.

pool_mode

Set the pool mode specific to this database. If not set, the default `pool_mode` is used.

max_db_connections

Configure a database-wide maximum (i.e. all pools within the database will not have more than this many server connections).

client_encoding

Ask specific `client_encoding` from server.

datestyle

Ask specific `datestyle` from server.

timezone

Ask specific `timezone` from server.

Section [users]

This section contains key=value lines like

```
user1 = settings
```

where the key will be taken as a user name and the value as a list of key=value pairs of configuration settings specific for this user. Example:

```
user1 = pool_mode=session
```

Only a few settings are available here.

pool_mode

Set the pool mode to be used for all connections from this user. If not set, the database or default `pool_mode` is used.

max_user_connections

Configure a maximum for the user (i.e. all pools with the user will not have more than this many server connections).

Section [peers]

The section `[peers]` defines the peers that PgBouncer can forward cancellation requests to and where those cancellation requests will be routed.

PgBouncer processes can be peered together in a group by defining a `peer_id` value and a `[peers]` section in the configs of all the PgBouncer processes. These PgBouncer processes can then forward cancellations requests to the process that it originated from. This is needed to make cancellations work when multiple PgBouncer processes (possibly on different servers) are behind the same TCP load balancer. Cancellation requests are sent over different TCP connections than the query they are cancelling, so a TCP load balancer might send the cancellation request connection to a different process than the one that it was meant for. By peering them these cancellation requests eventually end up at the right process. A more in-depth explanation is provided in this recording of a conference talk (https://www.youtube.com/watch?v=M585FfbboNA).

The section contains key=value lines like

```
peer_id = connection string
```

Where the key will be taken as a `peer_id` and the value as a connection string, consisting of key=value pairs of connection parameters, described below (similar to libpq, but the actual libpq is not used and the set of available features is different). Example:

```
1 = host=host1.example.com
2 = host=/tmp/pgbouncer-2  port=5555
```

Note 1: For peering to work, the `peer_id` of each PgBouncer process in the group must be unique within the peered group. And the `[peers]` section should contain entries for each of those peer ids. An example can be found in the examples section of these docs. It **is** allowed, but not necessary, for the `[peers]` section to contain the `peer_id` of the PgBouncer that the config is for. Such an entry will be ignored, but it is allowed to config management easy. Because it allows using the exact same `[peers]` section for multiple configs.

Note 2: Cross-version peering is supported as long as all peers are on the same side of the v1.21.0 version boundary. In v1.21.0 some breaking changes were made in how we encode the cancellation tokens that made them incompatible with the ones created by earlier versions.

host

Host name or IP address to connect to. Host names are resolved at connection time, the result is cached per `dns_max_ttl` parameter. If DNS returns several results, they are used in a round-robin manner. But in general it's not recommended to use a hostname that resolves to multiple IPs, because then the cancel request might still be forwarded to the wrong node and it would need to be forwarded again (which is only allowed up to three times).

If the value begins with `/`, then a Unix socket in the file-system namespace is used. If the value begins with `@`, then a Unix socket in the abstract namespace is used.

Examples:

```
host=localhost
host=127.0.0.1
host=2001:0db8:85a3:0000:0000:8a2e:0370:7334
host=/var/run/pgbouncer-1
```

port

Default: 6432

pool_size

Set the maximum number of cancel requests that can be in flight to the peer at the same time. It's quite normal for cancel requests to arrive in bursts, e.g. when the backing Postgres server slow or down. So it's important for `pool_size` to not be so low that it cannot handle these bursts.

If not set, the `default_pool_size` is used.

Include directive

The PgBouncer configuration file can contain include directives, which specify another configuration file to read and process. This allows splitting the configuration file into physically separate parts. The include directives look like this:

```
%include filename
```

If the file name is not an absolute path, it is taken as relative to the current working directory.

Authentication file format

This section describes the format of the file specified by the `auth_file` setting. It is a text file in the following format:

```
"username1" "password" ...
"username2" "md5abcdef012342345" ...
"username2" "SCRAM-SHA-256$<iterations>:<salt>$<storedkey>:<serverkey>"
```

There should be at least 2 fields, surrounded by double quotes. The first field is the user name and the second is either a plain-text, a MD5-hashed password, or a SCRAM secret. PgBouncer ignores the rest of the line. Double quotes in a field value can be escaped by writing two double quotes.

PostgreSQL MD5-hashed password format:

```
"md5" + md5(password + username)
```

So user `admin` with password `1234` will have MD5-hashed password `md545f2603610af569b6155c45067268c6b`.

PostgreSQL SCRAM secret format:

```
SCRAM-SHA-256$<iterations>:<salt>$<storedkey>:<serverkey>
```

See the PostgreSQL documentation and RFC 5803 for details on this.

The passwords or secrets stored in the authentication file serve two purposes. First, they are used to verify the passwords of incoming client connections, if a password-based authentication method is configured. Second, they are used as the passwords for outgoing connections to the backend server, if the backend server requires password-based authentication (unless the password is specified directly in the database's connection string). The latter works if the password is stored in plain text or MD5-hashed. SCRAM secrets can only be used for logging into a server if the client authentication also uses SCRAM, the PgBouncer database definition does not specify a user name, and the SCRAM secrets are identical in PgBouncer and the PostgreSQL server (same salt and iterations, not merely the same password). This is due to an inherent security property of SCRAM: The stored SCRAM secret cannot by itself be used for deriving login credentials.

The authentication file can be written by hand, but it's also useful to generate it from some other list of users and passwords. See `./etc/mkauth.py` for a sample script to generate the authentication file from the `pg_shadow` system table. Alternatively, use `auth_query` instead of `auth_file` to avoid having to maintain a separate authentication file.

HBA file format

The location of the HBA file is specified by the setting `auth_hba_file`. It is only used if `auth_type` is set to `hba`.

The file follows the format of the PostgreSQL `pg_hba.conf` file (see https://www.postgresql.org/docs/current/auth-pg-hba-conf.html (https://www.postgresql.org/docs/current/auth-pg-hba-conf.html)).

- Supported record types: `local`, `host`, `hostssl`, `hostnossl`.
- Database field: Supports `all`, `sameuser`, `@file`, multiple names. Not supported: `replication`, `samerole`, `samegroup`.
- User name field: Supports `all`, `@file`, multiple names. Not supported: `+groupname`.
- Address field: Supports IPv4, IPv6. Not supported: DNS names, domain prefixes.
- Auth-method field: Only methods supported by PgBouncer's `auth_type` are supported, plus `peer` and `reject`, but except `any` and `pam`, which only work globally. User name map (`map=`) parameter is not supported.

Examples

Small example configuration:

```
[databases]
template1 = host=localhost dbname=template1 auth_user=someuser

[pgbouncer]
pool_mode = session
listen_port = 6432
listen_addr = localhost
auth_type = md5
auth_file = users.txt
logfile = pgbouncer.log
pidfile = pgbouncer.pid
admin_users = someuser
stats_users = stat_collector
```

Database examples:

```
[databases]

; foodb over Unix socket
foodb =

; redirect bardb to bazdb on localhost
bardb = host=localhost dbname=bazdb

; access to destination database will go with single user
forcedb = host=localhost port=300 user=baz password=foo client_encoding=UNICODE datestyle=ISO
```

Example of a secure function for `auth_query`:

```
CREATE OR REPLACE FUNCTION pgbouncer.user_lookup(in i_username text, out uname text, out phash text)
RETURNS record AS $$
BEGIN
    SELECT usename, passwd FROM pg_catalog.pg_shadow
    WHERE usename = i_username INTO uname, phash;
    RETURN;
END;
$$ LANGUAGE plpgsql SECURITY DEFINER;
REVOKE ALL ON FUNCTION pgbouncer.user_lookup(text) FROM public, pgbouncer;
GRANT EXECUTE ON FUNCTION pgbouncer.user_lookup(text) TO pgbouncer;
```

Example configs for 2 peered PgBouncer processes to create a multi-core PgBouncer setup using `so_reuseport`. The config for the first process:

```
[databases]
postgres = host=localhost dbname=postgres

[peers]
1 = host=/tmp/pgbouncer1
2 = host=/tmp/pgbouncer2

[pgbouncer]
listen_addr=127.0.0.1
auth_file=auth_file.conf
so_reuseport=1
unix_socket_dir=/tmp/pgbouncer1
peer_id=1
```

The config for the second process:

```
[databases]
postgres = host=localhost dbname=postgres

[peers]
1 = host=/tmp/pgbouncer1
2 = host=/tmp/pgbouncer2

[pgbouncer]
listen_addr=127.0.0.1
auth_file=auth_file.conf
so_reuseport=1
; only unix_socket_dir and peer_id are different
unix_socket_dir=/tmp/pgbouncer2
peer_id=2
```

See also

pgbouncer(1) - man page for general usage, console commands

https://www.pgbouncer.org/ (https://www.pgbouncer.org/)

% PGBOUNCER.INI(5) @PACKAGE_VERSION@ | Databases

Name

pgbouncer.ini - configuration file for pgbouncer

% PGBOUNCER(1) @PACKAGE_VERSION@ | Databases

Name

pgbouncer - lightweight connection pooler for PostgreSQL

pgbouncer

Synopsis

```
pgbouncer [-d][-R][-v][-u user] <pgbouncer.ini>
pgbouncer -V|-h
```

On Windows, the options are:

```
pgbouncer.exe [-v][-u user] <pgbouncer.ini>
pgbouncer.exe -V|-h
```

Additional options for setting up a Windows service:

```
pgbouncer.exe --regservice   <pgbouncer.ini>
pgbouncer.exe --unregservice <pgbouncer.ini>
```

Description

pgbouncer is a PostgreSQL connection pooler. Any target application can be connected to **pgbouncer** as if it were a PostgreSQL server, and **pgbouncer** will create a connection to the actual server, or it will reuse one of its existing connections.

The aim of **pgbouncer** is to lower the performance impact of opening new connections to PostgreSQL.

In order not to compromise transaction semantics for connection pooling, **pgbouncer** supports several types of pooling when rotating connections:

Session pooling
: Most polite method. When a client connects, a server connection will be assigned to it for the whole duration the client stays connected. When the client disconnects, the server connection will be put back into the pool. This is the default method.

Transaction pooling

A server connection is assigned to a client only during a transaction. When PgBouncer notices that transaction is over, the server connection will be put back into the pool.

Statement pooling

Most aggressive method. The server connection will be put back into the pool immediately after a query completes. Multi-statement transactions are disallowed in this mode as they would break.

The administration interface of **pgbouncer** consists of some new `SHOW` commands available when connected to a special "virtual" database **pgbouncer**.

Quick-start

Basic setup and usage is as follows.

1. Create a pgbouncer.ini file. Details in **pgbouncer(5)**. Simple example:

```
[databases]
template1 = host=localhost port=5432 dbname=template1

[pgbouncer]
listen_port = 6432
listen_addr = localhost
auth_type = md5
auth_file = userlist.txt
logfile = pgbouncer.log
pidfile = pgbouncer.pid
admin_users = someuser
```

2. Create a `userlist.txt` file that contains the users allowed in:

```
"someuser" "same_password_as_in_server"
```

3. Launch **pgbouncer**:

```
$ pgbouncer -d pgbouncer.ini
```

4. Have your application (or the **psql** client) connect to **pgbouncer** instead of directly to the PostgreSQL server:

```
$ psql -p 6432 -U someuser template1
```

5. Manage **pgbouncer** by connecting to the special administration database **pgbouncer** and issuing `SHOW HELP;` to begin:

```
$ psql -p 6432 -U someuser pgbouncer
pgbouncer=# SHOW HELP;
NOTICE:  Console usage
DETAIL:
  SHOW [HELP|CONFIG|DATABASES|FDS|POOLS|CLIENTS|SERVERS|SOCKETS|LISTS|VERSION|...]
  SET key = arg
  RELOAD
  PAUSE
  SUSPEND
  RESUME
  SHUTDOWN
  [...]
```

6. If you made changes to the pgbouncer.ini file, you can reload it with:

```
pgbouncer=# RELOAD;
```

Command line switches

`-d`, `--daemon`

Run in the background. Without it, the process will run in the foreground.

In daemon mode, setting `pidfile` as well as `logfile` or `syslog` is required. No log messages will be written to stderr after going into the background.

Note: Does not work on Windows; **pgbouncer** need to run as service there.

`-R`, `--reboot`

DEPRECATED: Instead of this option use a rolling restart with multiple pgbouncer processes listening on the same port using so_reuseport instead Do an online restart. That means connecting to the running process, loading the open sockets from it, and then using them. If there is no active process, boot normally. Note: Works only if OS supports Unix sockets and the `unix_socket_dir` is not disabled in configuration. Does not work on Windows. Does not work with TLS connections, they are dropped.

`-u` *USERNAME*, `--user=`*USERNAME*

Switch to the given user on startup.

`-v`, `--verbose`

Increase verbosity. Can be used multiple times.

`-q`, `--quiet`

Be quiet: do not log to stderr. This does not affect logging verbosity, only that stderr is not to be used. For use in init.d scripts.

`-V`, `--version`

Show version.

`-h`, `--help`

Show short help.

`--regservice`

Win32: Register pgbouncer to run as Windows service. The **service_name** configuration parameter value is used as the name to register under.

`--unregservice`

Win32: Unregister Windows service.

Admin console

The console is available by connecting as normal to the database **pgbouncer**:

```
$ psql -p 6432 pgbouncer
```

Only users listed in the configuration parameters **admin_users** or **stats_users** are allowed to log in to the console. (Except when `auth_type=any`, then any user is allowed in as a stats_user.)

Additionally, the user name **pgbouncer** is allowed to log in without password, if the login comes via the Unix socket and the client has same Unix user UID as the running process.

The admin console currently only supports the simple query protocol. Some drivers use the extended query protocol for all commands; these drivers will not work for this.

Show commands

The **SHOW** commands output information. Each command is described below.

SHOW STATS

Shows statistics. In this and related commands, the total figures are since process start, the averages are updated every `stats_period`.

database
: Statistics are presented per database.

total_xact_count
: Total number of SQL transactions pooled by **pgbouncer**.

total_query_count
: Total number of SQL commands pooled by **pgbouncer**.

total_received
: Total volume in bytes of network traffic received by **pgbouncer**.

total_sent
: Total volume in bytes of network traffic sent by **pgbouncer**.

total_xact_time
: Total number of microseconds spent by **pgbouncer** when connected to PostgreSQL in a transaction, either idle in transaction or executing queries.

total_query_time
: Total number of microseconds spent by **pgbouncer** when actively connected to PostgreSQL, executing queries.

total_wait_time
: Time spent by clients waiting for a server, in microseconds. Updated when a client connection is assigned a backend connection.

avg_xact_count
: Average transactions per second in last stat period.

avg_query_count
: Average queries per second in last stat period.

avg_recv
: Average received (from clients) bytes per second.

avg_sent
: Average sent (to clients) bytes per second.

avg_xact_time
: Average transaction duration, in microseconds.

avg_query_time

Average query duration, in microseconds.

avg_wait_time

Average time spent by clients waiting for a server that were assigned a backend connection within the current `stats_period`, in microseconds (averaged per second within that period).

SHOW STATS_TOTALS

Subset of **SHOW STATS** showing the total values (**total_**).

SHOW STATS_AVERAGES

Subset of **SHOW STATS** showing the average values (**avg_**).

SHOW TOTALS

Like **SHOW STATS** but aggregated across all databases.

SHOW SERVERS

type

S, for server.

user

User name **pgbouncer** uses to connect to server.

database

Database name.

state

State of the pgbouncer server connection, one of **active**, **idle**, **used**, **tested**, **new**, **active_cancel**, **being_canceled**.

addr

IP address of PostgreSQL server.

port

Port of PostgreSQL server.

local_addr

Connection start address on local machine.

local_port

Connection start port on local machine.

connect_time

When the connection was made.

request_time

When last request was issued.

wait

Not used for server connections.

wait_us

Not used for server connections.

close_needed

1 if the connection will be closed as soon as possible, because a configuration file reload or DNS update changed the connection information or **RECONNECT** was issued.

ptr

Address of internal object for this connection. Used as unique ID.

link

Address of client connection the server is paired with.

remote_pid

PID of backend server process. In case connection is made over Unix socket and OS supports getting process ID info, its OS PID. Otherwise it's extracted from cancel packet the server sent, which should be the PID in case the server is PostgreSQL, but it's a random number in case the server it is another PgBouncer.

tls

A string with TLS connection information, or empty if not using TLS.

application_name

A string containing the `application_name` set on the linked client connection, or empty if this is not set, or if there is no linked connection.

prepared_statements : The amount of prepared statements that are prepared on the server. This number is limited by the `max_prepared_statements` setting.

SHOW CLIENTS

type

C, for client.

user

Client connected user.

database

Database name.

state

State of the client connection, one of **active**, **waiting**, **active_cancel_req**, or **waiting_cancel_req**.

addr

IP address of client.

port

Source port of client.

local_addr

Connection end address on local machine.

local_port

Connection end port on local machine.

connect_time

Timestamp of connect time.

request_time

Timestamp of latest client request.

wait

Current waiting time in seconds.

wait_us

Microsecond part of the current waiting time.

close_needed

not used for clients

ptr

Address of internal object for this connection. Used as unique ID.

link

Address of server connection the client is paired with.

remote_pid

Process ID, in case client connects over Unix socket and OS supports getting it.

tls

A string with TLS connection information, or empty if not using TLS.

application_name

A string containing the `application_name` set by the client for this connection, or empty if this was not set.

prepared_statements : The amount of prepared statements that the client has prepared

SHOW POOLS

A new pool entry is made for each couple of (database, user).

database
Database name.
user
User name.
cl_active
Client connections that are either linked to server connections or are idle with no queries waiting to be processed.
cl_waiting
Client connections that have sent queries but have not yet got a server connection.
cl_active_cancel_req
Client connections that have forwarded query cancellations to the server and are waiting for the server response.
cl_waiting_cancel_req
Client connections that have not forwarded query cancellations to the server yet.
sv_active
Server connections that are linked to a client.
sv_active_cancel
Server connections that are currently forwarding a cancel request.
sv_being_canceled
Servers that normally could become idle but are waiting to do so until all in-flight cancel requests have completed that were sent to cancel a query on this server.
sv_idle
Server connections that are unused and immediately usable for client queries.
sv_used
Server connections that have been idle for more than `server_check_delay`, so they need `server_check_query` to run on them before they can be used again.
sv_tested
Server connections that are currently running either `server_reset_query` or `server_check_query`.
sv_login
Server connections currently in the process of logging in.
maxwait
How long the first (oldest) client in the queue has waited, in seconds. If this starts increasing, then the current pool of servers does not handle requests quickly enough. The reason may be either an overloaded server or just too small of a **pool_size** setting.
maxwait_us
Microsecond part of the maximum waiting time.
pool_mode
The pooling mode in use.

SHOW PEER_POOLS

A new peer_pool entry is made for each configured peer.

database
ID of the configured peer entry.
cl_active_cancel_req
Client connections that have forwarded query cancellations to the server and are waiting for the server response.
cl_waiting_cancel_req

Client connections that have not forwarded query cancellations to the server yet.

sv_active_cancel

Server connections that are currently forwarding a cancel request.

sv_login

Server connections currently in the process of logging in.

SHOW LISTS

Show following internal information, in columns (not rows):

databases

Count of databases.

users

Count of users.

pools

Count of pools.

free_clients

Count of free clients. These are clients that are disconnected, but PgBouncer keeps the memory around that was allocated for them so it can be reused for a future clients to avoid allocations.

used_clients

Count of used clients.

login_clients

Count of clients in **login** state.

free_servers

Count of free servers. These are servers that are disconnected, but PgBouncer keeps the memory around that was allocated for them so it can be reused for a future servers to avoid allocations.

used_servers

Count of used servers.

dns_names

Count of DNS names in the cache.

dns_zones

Count of DNS zones in the cache.

dns_queries

Count of in-flight DNS queries.

dns_pending

not used

SHOW USERS

name

The user name

pool_mode

The user's override pool_mode, or NULL if the default will be used instead.

SHOW DATABASES

name

Name of configured database entry.

host

Host pgbouncer connects to.

port

Port pgbouncer connects to.

database

Actual database name pgbouncer connects to.

force_user

When the user is part of the connection string, the connection between pgbouncer and PostgreSQL is forced to the given user, whatever the client user.

pool_size

Maximum number of server connections.

min_pool_size

Minimum number of server connections.

reserve_pool

Maximum number of additional connections for this database.

pool_mode

The database's override pool_mode, or NULL if the default will be used instead.

max_connections

Maximum number of allowed connections for this database, as set by **max_db_connections**, either globally or per database.

current_connections

Current number of connections for this database.

paused

1 if this database is currently paused, else 0.

disabled

1 if this database is currently disabled, else 0.

SHOW PEERS

peer_id

ID of the configured peer entry.

host

Host pgbouncer connects to.

port

Port pgbouncer connects to.

pool_size

Maximum number of server connections that can be made to this peer

SHOW FDS

Internal command - shows list of file descriptors in use with internal state attached to them.

When the connected user has the user name "pgbouncer", connects through the Unix socket and has same the UID as the running process, the actual FDs are passed over the connection. This mechanism is used to do an online restart. Note: This does not work on Windows.

This command also blocks the internal event loop, so it should not be used while PgBouncer is in use.

fd

File descriptor numeric value.

task

One of **pooler**, **client** or **server**.

user

User of the connection using the FD.

database

Database of the connection using the FD.

addr

IP address of the connection using the FD, **unix** if a Unix socket is used.

port

Port used by the connection using the FD.

cancel

Cancel key for this connection.

link

fd for corresponding server/client. NULL if idle.

SHOW SOCKETS, SHOW ACTIVE_SOCKETS

Shows low-level information about sockets or only active sockets. This includes the information shown under **SHOW CLIENTS** and **SHOW SERVERS** as well as other more low-level information.

SHOW CONFIG

Show the current configuration settings, one per row, with the following columns:

key

Configuration variable name

value

Configuration value

default

Configuration default value

changeable

Either **yes** or **no**, shows if the variable can be changed while running. If **no**, the variable can be changed only at boot time. Use **SET** to change a variable at run time.

SHOW MEM

Shows low-level information about the current sizes of various internal memory allocations. The information presented is subject to change.

SHOW DNS_HOSTS

Show host names in DNS cache.

hostname

Host name.

ttl

How many seconds until next lookup.

addrs

Comma separated list of addresses.

SHOW DNS_ZONES

Show DNS zones in cache.

zonename

Zone name.

serial

Current serial.

count

Host names belonging to this zone.

SHOW VERSION

Show the PgBouncer version string.

SHOW STATE

Show the PgBouncer state settings. Current states are active, paused and suspended.

Process controlling commands

PAUSE [db]

PgBouncer tries to disconnect from all servers. Disconnecting each server connection waits for that server connection to be released according to the server pool's pooling mode (in transaction pooling mode, the transaction must complete, in statement mode, the statement must complete, and in session pooling mode the client must disconnect). The command will not return before all server connections have been disconnected. To be used at the time of database restart.

If database name is given, only that database will be paused.

New client connections to a paused database will wait until **RESUME** is called.

DISABLE db

Reject all new client connections on the given database.

ENABLE db

Allow new client connections after a previous **DISABLE** command.

RECONNECT [db]

Close each open server connection for the given database, or all databases, after it is released (according to the pooling mode), even if its lifetime is not up yet. New server connections can be made immediately and will connect as necessary according to the pool size settings.

This command is useful when the server connection setup has changed, for example to perform a gradual switchover to a new server. It is *not* necessary to run this command when the connection string in pgbouncer.ini has been changed and reloaded (see **RELOAD**) or when DNS resolution has changed, because then the equivalent of this command will be run automatically. This command is only necessary if something downstream of PgBouncer routes the connections.

After this command is run, there could be an extended period where some server connections go to an old destination and some server connections go to a new destination. This is likely only sensible when switching read-only traffic between read-only replicas, or when switching between nodes of a multimaster replication setup. If all connections need to be switched at the same time, **PAUSE** is recommended instead. To close server connections without waiting (for example, in emergency failover rather than gradual switchover scenarios), also consider **KILL**.

KILL db

Immediately drop all client and server connections on given database.

New client connections to a killed database will wait until **RESUME** is called.

SUSPEND

All socket buffers are flushed and PgBouncer stops listening for data on them. The command will not return before all buffers are empty. To be used at the time of PgBouncer online reboot.

New client connections to a suspended database will wait until **RESUME** is called.

RESUME [db]

Resume work from previous **KILL**, **PAUSE**, or **SUSPEND** command.

SHUTDOWN

The PgBouncer process will exit.

RELOAD

The PgBouncer process will reload its configuration files and update changeable settings. This includes the main configuration file as well as the files specified by the settings `auth_file` and `auth_hba_file`.

PgBouncer notices when a configuration file reload changes the connection parameters of a database definition. An existing server connection to the old destination will be closed when the server connection is next released (according to the pooling mode), and new server connections will immediately use the updated connection parameters.

WAIT_CLOSE [db]

Wait until all server connections, either of the specified database or of all databases, have cleared the "close_needed" state (see **SHOW SERVERS**). This can be called after a **RECONNECT** or **RELOAD** to wait until the respective configuration change has been fully activated, for example in switchover scripts.

Other commands

SET key = arg

Changes a configuration setting (see also **SHOW CONFIG**). For example:

```
SET log_connections = 1;
SET server_check_query = 'select 2';
```

(Note that this command is run on the PgBouncer admin console and sets PgBouncer settings. A **SET** command run on another database will be passed to the PostgreSQL backend like any other SQL command.)

Signals

SIGHUP
Reload config. Same as issuing the command **RELOAD** on the console.
SIGINT
Safe shutdown. Same as issuing **PAUSE** and **SHUTDOWN** on the console.
SIGTERM
Immediate shutdown. Same as issuing **SHUTDOWN** on the console.
SIGUSR1
Same as issuing **PAUSE** on the console.
SIGUSR2
Same as issuing **RESUME** on the console.

Libevent settings

From the Libevent documentation:

> *It is possible to disable support for epoll, kqueue, devpoll, poll or select by setting the environment variable EVENT_NOEPOLL, EVENT_NOKQUEUE, EVENT_NODEVPOLL, EVENT_NOPOLL or EVENT_NOSELECT, respectively.*
>
> *By setting the environment variable EVENT_SHOW_METHOD, libevent displays the kernel notification method that it uses.*

See also

pgbouncer(5) - man page of configuration settings descriptions

https://www.pgbouncer.org/ (https://www.pgbouncer.org/)

Contents

Jump to...

PgBouncer compilation and installation

Building

PgBouncer depends on few things to get compiled:

- GNU Make 3.81+
- Libevent 2.0+
- pkg-config
- OpenSSL 1.0.1+ for TLS support
- (optional) c-ares as alternative to Libevent’s evdns
- (optional) PAM libraries

When dependencies are installed just run:

```
$ ./configure --prefix=/usr/local
$ make
$ make install
```

If you are building from Git, or are building for Windows, please see separate build instructions below.

DNS lookup support

PgBouncer does host name lookups at connect time instead of just once at configuration load time. This requires an asynchronous DNS implementation. The following table shows supported backends and their probing order:

backend	parallel	EDNS0	/etc/hosts	SOA lookup	note

				(2)	
c-ares	yes	yes	yes	yes	buggy in <=1.10
evdns, libevent 2.x	yes	no	yes	no	does not check /etc/hosts updates
getaddrinfo_a, glibc 2.9+	yes	yes (3)	yes	no	N/A on non-glibc
getaddrinfo, libc	no	yes (3)	yes	no	requires pthreads

1. EDNS0 is required to have more than 8 addresses behind one host name.
2. SOA lookup is needed to re-check host names on zone serial change.
3. To enable EDNS0, add `options edns0` to `/etc/resolv.conf`.

c-ares is the most fully-featured implementation and is recommended for most uses and binary packaging (if a sufficiently new version is available). Libevent's built-in evdns is also suitable for many uses, with the listed restrictions. The other backends are mostly legacy options at this point and don't receive much testing anymore.

By default, c-ares is used if it can be found. Its use can be forced with `configure --with-cares` or disabled with `--without-cares`. If c-ares is not used (not found or disabled), then Libevent is used. Specify `--disable-evdns` to disable the use of Libevent's evdns and fall back to a libc-based implementation.

PAM authentication

To enable PAM authentication, `./configure` has a flag `--with-pam` (default value is no). When compiled with PAM support, a new global authentication type `pam` is available to validate users through PAM.

systemd integration

To enable systemd integration, use the `configure` option `--with-systemd`. This allows using `Type=notify` service units as well as socket activation. See `etc/pgbouncer.service` and `etc/pgbouncer.socket` for examples.

Building from Git

configuration files before you can run configure.

```
$ git clone https://github.com/pgbouncer/pgbouncer.git
$ cd pgbouncer
$ git submodule init
$ git submodule update
$ ./autogen.sh
$ ./configure
$ make
$ make install
```

All files will be installed under `/usr/local` by default. You can supply one or more command-line options to `configure`. Run

`./configure --help` to list the available options and the environment variables that customizes the configuration.

Additional packages required: autoconf, automake, libtool, pandoc

Testing

See the `README.md` file in the test directory on how to run the tests.

Building on Windows

The only supported build environment on Windows is MinGW. Cygwin and Visual $ANYTHING are not supported.

To build on MinGW, do the usual:

```
$ ./configure
$ make
```

If cross-compiling from Unix:

```
$ ./configure --host=i586-mingw32msvc
```

Running on Windows

Running from the command line goes as usual, except that the `-d` (daemonize), `-R` (reboot), and `-u` (switch user) switches will not work.

To run PgBouncer as a Windows service, you need to configure the `service_name` parameter to set a name for the service. Then:

```
$ pgbouncer -regservice config.ini
```

To uninstall the service:

configuration file. But before that, you need to register `pgbevent.dll`:

```
$ regsvr32 pgbevent.dll
```

To unregister it, do:

```
$ regsvr32 /u pgbevent.dll
```

Documentation

- FAQ
- Configuration
- Usage
- Install
- Changelog

Contents

Jump to...

PgBouncer changelog

PgBouncer 1.23.x

2024-08-02 - PgBouncer 1.23.1 - "Everything is put back in order"

- Fixes
  - Fix a possible segmentation fault after PgBouncer reloads its configuration. (#1105) (bug introduced in 1.23.0)
  - Fix all known put_in_order crashes. (#1120) (new crashes were introduced in 1.23.0)
  - Add missing files to release tarball that are required for testing. (#1124) (missing files were introduced in 1.23.0)

2024-07-03 - PgBouncer 1.23.0 - "Into the new beginnings"

- Features
  - Add support for rolling restarts. SIGTERM doesn't cause immediate shutdown of the PgBouncer process anymore. It now does a "super safe shutdown": waiting for all clients to disconnect before shutting down. The new SIGTERM behaviour allows rolling restarts of multiple PgBouncer processes behind a load balancer, or listening on the same port using `so_reuseport`. This is a **minor breaking change**. If you relied on the old behaviour of SIGTERM in your Dockerfile or Systemd service file you should now use SIGQUIT. (#902)
  - Add support for user name maps for `cert` and `peer` authentication methods. This feature provides the flexibility that the user initiating the connection does not have to be the database user. PgBouncer support for user name maps works very similar to the postgres with the exceptions listed in the docs. (#996)

- Improve `SHOW USERS` output listing the connections. (#1040)
- Allow `pool_size` configuration per user. (#1049)
- Allow `server_lifetime` configuration per database. (#1057)
- Add support for listing dynamically created users in the output of `SHOW USERS`. (#1052)
- Add support for `all` address type in hba configuration. (#1078)
- Add support for automatically restarting when using systemd. (#1080)
- Increase c-ares minimum version requirement to 1.9.0 (#1076)

- Fixes
  - Fix issues handling large and partial startup packets. (#1058)
  - Add support for `--config=value` format in options startup parameter. (#1064)
  - Fix `avg_wait_time` metric calculation. (#727)
  - Add support for negotiating the postgres protocol version with the client. (#1007)
  - Add outstanding request for `auth_query`. (#1034)
  - Multiple documentation and CI improvements.

PgBouncer 1.22.x

2024-03-04 - PgBouncer 1.22.1 - “It’s summer in Bangalore”

- Fixes
  - Fix issues caused by some clients using `COPY FROM STDIN` queries. Such queries could introduce memory leaks, performance regressions and prepared statement misbehavior. (#1025) (bug introduced in 1.21.0)
  - Add missing tests to release tarball (#1026) (missing tests were introduced in 1.19.0 & 1.21.0)

2024-01-31 - PgBouncer 1.22.0 - “DEALLOCATE ALL”

- Features

  - Support configuring `auth_query` per database (#979)
- Changes
  - Improve settings in the recommended systemd unit file (#983)
  - Make fail fast logic handle all scenarios where no working connections to the database exist anymore and none can be established (#998)
  - Multiple documentation improvements
- Fixes
  - Fix issue in PG14+ where PgBouncer would send `SET DateStyle='ISO'` for every transaction (#879)
  - Fix handling of empty `application_name` (#999)
  - Fix building on Windows with OpenSSL 3.2.0 (#1009)

PgBouncer 1.21.x

2023-10-16 - PgBouncer 1.21.0 - "The one with prepared statements"

- Features
  - Add support for protocol-level named prepared statements! This is probably one of the most requested features for PgBouncer. Using prepared statements together with PgBouncer can reduce the CPU load on your system a lot (both at the PgBouncer side and the PostgreSQL side). In synthetic benchmarks this feature was able to increase query throughput anywhere from 15% to 250%, depending on the workload. To benefit from this new feature you need to change the new `max_prepared_statements` setting to a non-zero value (the exact value depends on your workload, but 100 is probably reasonable). See the docs on `max_prepared_statements` for details on how the feature works, its limitations, and how to tune the value. After doing that you need to make sure your client library actually uses prepared statements. How to do that differs for each client, so you should look at the docs for the client you're using. This feature has been tested very well

- Changes
    - Improve security of OpenSSL settings, the defaults used were VERY outdated. With this release the defaults are now the same as the OpenSSL defaults of the system that runs PgBouncer. (#948 & libusual/#41)
    - PgBouncer now uses OpenSSL to calculate MD5 hashes when possible. This is necessary to use PgBouncer in a FIPS compliant way. (#949)
    - Maintain `min_pool_size` for pools with a forced user even if no clients are connected to PgBouncer (#947)
    - The way a `peer_id` is encoded in the cancellation token by PgBouncer has changed, this means that peering between different PgBouncer versions will not work if not all of them are on the same side of the v1.21.0 version boundary. (#945)
- Fixes
    - Fix crash with error message: "FATAL in function client_proto(): bad client state: 6/7" (#928) (bug introduced in 1.18.0)
    - Fix crash with error message: "FATAL in function server_proto(): server in bad state: 11" (#927) (bug introduced in 1.18.0)
    - Reduce cancellation sending log level (#903)
    - Fix slog log prefix for peers (#922)
    - Fix typos in docs (#932)
    - Fix errors pointed out by static analyzer (#943)
    - Don't kill all waiting clients on temporary FATAL errors during login (#946)
    - Use auto-database when database in `auth_dbname` is not explicitly configured (#921)
- Cleanup
    - Remove support for udns (#938)

PgBouncer 1.20.x

2023-08-09 - PgBouncer 1.20.1 - "Optional options"

unknown parameters inside the `options` startup parameter anymore. (#908) (regression was introduced in 1.20.0)
- Fix confusing typo in the docs (#917)

2023-07-20 - PgBouncer 1.20.0 - “A funny name goes here”

- Deprecations
    - Online restart option is now considered deprecated. The feature has received very little love in recent years. There are multiple known issues with it and newly added features often don’t support it. The recommended method to do online restarts these days is using the `so_reuseport` and `peers` feature. That way you can have multiple different PgBouncer processes running on the same port. Then by restarting those processes one-by-one, you can make sure there’s always a PgBouncer process listening on the desired port. (#894)
- Features
    - Introduce the `track_extra_parameters` which allows tracking of more parameters in transaction pooling mode. Previously, PgBouncer only tracked `application_name`, `DateStyle`, `TimeZone` and `standard_conforming_strings`. Now PgBouncer also tracks `IntervalStyle` by default. And by changing `track_extra_parameters` you can track even more settings, but only ones that PostgreSQL reports back to the client. If you’re using Citus 12.0+, then Citus will make sure that PostgreSQL also reports `search_path` back to the client. So if you use Citus you can add `search_path` to the `track_extra_parameters` setting. (#867)
    - Forward SQLSTATE in authentication phase. This allows the detection of database not existing, which is done by Npgsql (a .NET data provider for PostgreSQL). (#814)
    - Change default `server_tls_sslmode` to `prefer`. (#866)
    - Add support for the `options` startup parameter. This allows usage of the `PGOPTIONS` environment variable that `psql` and `libpq` know about. Using this variable you can set any

- Fixes
  - Don’t crash when the `pgbouncer` admin database is used as auth_dbname. It’s still not supported, but this now gives a clear error instead of crashing. (#817)
  - Fix name of `peer_cache` in `SHOW MEM`. It was incorrectly showing up as `db_cache` before. (#864)
  - Fix src/dst confusion in log. PgBouncer was logging a source IP when it meant to log the destination IP. (#880)
  - Only log admin connections over unix sockets when `log_connections` is set to `1`. (#883)

PgBouncer 1.19.x

2023-05-31 - PgBouncer 1.19.1 - “Sunny Spring”

This is a minor release that fixes a few recently introduced bugs:

- Fixes
  - Fix: FATAL in function disconnect_client(): bad client state: 0 (#846) (bug introduced in 1.18.0)
  - Fix: FATAL in function server_proto(): server in bad state: 14 (#849) (bug introduced in 1.18.0)
  - Add files required to run python based tests to release tarball (#852) (new tests introduced in 1.19.0)

2023-05-04 - PgBouncer 1.19.0 - “The old-fashioned, human-generated kind”

- Features
  - Add `auth_dbname` option, which specifies against which database to run the `auth_query`. (#764)

  - Add support for peering between PgBouncer processes. This allows configuring PgBouncer such that cancellation requests continue to work when multiple different PgBouncer processes are behind a single load balancer. (#666)
  - Add a dedicated `cancel_wait_timeout` setting, which determines after how long to give up on forwarding a cancel request. Default is 10 seconds. (#833)
  - New testing framework (#792)
- Fixes
  - Fix possible memory leak on TLS handshake failure. (#796)
  - Give more accurate error messages for unsupported command-line options on Windows. (#620)
  - Fix calling `disconnect_server` on a server in `BEING_CANCELED` state. (#815) (introduced in 1.18.0)
  - Don't exit with a non-zero status when a `SIGTERM` is received. (#834)
  - Fail hard during startup when a socket could not be created in `unix_socket_dir`. (#830)
  - Fail hard during startup when none of the addresses in `listen_addr` could be listened on. (#838)
  - Give more warning messages with more information when `sbuf_connect` fails. This is especially useful when failing to create Unix sockets. (#837)
- Cleanups
  - Various CI updates for better performance
  - Removed AppVeyor

PgBouncer 1.18.x

2022-12-12 - PgBouncer 1.18.0 - "No real mystery"

- Features
  - Add `application_name` to `SHOW CLIENTS`/`SERVERS`/`SOCKETS` output (#449)

- Fail `sbuf_send_pending` operation if destination socket is closed (#652)
- Fix a few possible crashes (#700, #730)
- Fix for overflow bug in comma-separated host list feature, causing connection to get re-routed to Unix socket (#747)
- Don't evict connections to achieve `min_pool_size` (#648)
- Fix `SHOW HELP` with PostgreSQL 15 (#769)
- Fix race condition in query cancellation handling. It was possible that a query cancellation for one client canceled a query for another one. This could happen when a cancel request was received by PgBouncer when the query it was meant to cancel already completed by itself. (#717)

- Cleanups
  - Various CI updates

PgBouncer 1.17.x

2022-03-23 - PgBouncer 1.17.0 - "A line has been drawn"

- Features
  - A database definition can specify a comma-separated host list. The hosts will be connected to in a round-robin manner.
  - When connecting to a non-existing database, the error ("no such database") is now reported after authentication. This prevents unauthenticated clients from probing what databases exist. (This is similar to the change in version 1.15.0 to report missing users after authentication.)
  - Don't send server disconnect errors to the client before login. This could reveal not-quite-public information, such as configuration details, to a client that is not logged in yet.
  - Increase maximum password length again. Apparently, the last increase wasn't enough for long enough.

- The Windows build now includes a version-information resource file.
- The Windows builds created on CI are now statically linked, so they can be used directly without requiring any dependencies.

- Fixes
  - OpenSSL 3 support has been fixed. Previous releases would crash.
  - Don’t apply fast-fail at connect time. This is part of the above-mentioned change to not report server errors before authentication. It also fixes a particular situation with SCRAM pass-through authentication, where we need to allow the client-side authentication exchange in order to be able to fix the server-side connection by re-authenticating. The fast-fail mechanism still applies right after authentication, so the effective observed behavior will be the same in most situations.
  - Change `auth_type` in sample `pgbouncer.ini` to `md5` to match the built-in default. Some deploy this file as the default configuration file, so check if this changed configuration still makes sense for you.
  - Fix crash at exit in assert-enabled builds.
  - Improve `tcp_defer_accept` documentation and behavior. The documentation was incorrect and misleading about the default. In some cases the wrong value was showing in “show config”. Also, if it’s set but not supported, give an error instead of ignoring, similar to how other platform-specific socket options are handled.
  - Fix build with c-ares on Windows. c-ares >=1.18.0 is now required on Windows.
- Cleanups
  - Most deprecation warnings from Autoconf >=2.70 have been cleaned up. Older Autoconf versions are still supported.
  - Cirrus CI use has been expanded to more platforms.
  - Travis CI support has been removed.
  - Update locations to search for default root CA file, to cover more platforms, such as Fedora/RHEL/CentOS.

- The test suite scripts use `command -v` instead of `which`, which is deprecated.
- Several error messages have been reworded to make it clearer which command or configuration setting they relate to.
- The test suite scripts no longer require GNU sed.
- `make check` now works on Windows (but not the SSL test suite yet).
- Document that the admin console only supports the simple query protocol, and give better error messages about this.

PgBouncer 1.16.x

2021-11-11 - PgBouncer 1.16.1 - “Test of depth against quiet efficiency”

This is a minor release with a security fix.

- Make PgBouncer acting as a server reject extraneous data after an SSL or GSS encryption handshake.

  A man-in-the-middle with the ability to inject data into the TCP connection could stuff some cleartext data into the start of a supposedly encryption-protected database session. This could be abused to send faked SQL commands to the server, although that would only work if PgBouncer did not demand any authentication data. (However, a PgBouncer setup relying on SSL certificate authentication might well not do so.) (CVE-2021-3935)

2021-08-09 - PgBouncer 1.16.0 - “Fended off a jaguar”

- Features
  - Support hot reloading of TLS settings. When the configuration file is reloaded, changed TLS settings automatically take effect.
  - Add support for abstract Unix-domain sockets. Prefix a Unix-domain socket path with `@` to use a socket in the abstract namespace. This matches the corresponding PostgreSQL

respectively. Various cloud services require this.
  - The minimum pool size can now be set per database, similar to the regular pool size and the reserve pool size.
  - The number of pending query cancellations is shown in `SHOW POOLS`.
- Fixes
  - Configuration parsing now has tighter error handling in many places. Where previously it might have logged an error and proceeded, those configuration errors would now result in startup failures. This is what always should have happened, but some code didn't do this right. Some users might discover that their configurations have been faulty all along and will not work anymore.
  - Query cancel handling has been fixed. Under some circumstances, cancel requests would seemingly get stuck for a long time. This should no longer happen. In fact, cancel requests can now exceed the pool size by a factor of two, so they really shouldn't get stuck anymore. (#542, #543)
  - Mixed use of md5 and scram via hba has been fixed.
  - The build with c-ares on Windows has been fixed.
  - The dreaded "FIXME: query end, but query_start == 0" messages have been fixed. We now know why they happen, and you shouldn't see them anymore. (#565)
  - Fix reloading of `default_pool_size`, `min_pool_size`, and `res_pool_size`. Reloading these settings previously didn't work.
- Cleanups
  - Cirrus CI is now used instead of Travis CI.
  - As usual, many tests have been added.
  - The "unclean server" log message has been clarified a bit. It now says "client disconnect while server was not ready" or "client disconnect before everything was sent to the server". The former can happen if the client connection is closed when the server has a transaction block open, which confused some users.

now explicitly prohibited.
- Errors sent to clients before the connection is closed are now labeled as FATAL instead of just ERROR. Some clients were confused otherwise. (#564)
- Fix compiler warnings with GCC 11. (#623)

PgBouncer 1.15.x

2020-11-19 - PgBouncer 1.15.0 - “Ich hab noch einen Koffer in Berlin”

- Features
  - Improve authentication failure reporting. The authentication failure messages sent to the client now only state that authentication failed but give no further details. Details are available in the PgBouncer log. Also, if the requested user does not exist, the authentication is still processed to the end and will result in the same generic failure message. All this prevents clients from probing the PgBouncer instance for user names and other authentication-related insights. This is similar to how PostgreSQL behaves.
  - Don’t log anything if client disconnects immediately. This avoids log spam when monitoring systems just open a TCP/IP connection but don’t send anything before disconnecting.
  - Use systemd journal for logging when in use. When we detect that stderr is going to the systemd journal, we use systemd native functions for log output. This avoids printing duplicate timestamp and pid, thus making the log a bit cleaner. Also, this adds metadata such as the severity to the logs, so that if the journal gets sent on to syslog, the messages have useful metadata attached.
  - A subset of the test suite can now be run under Windows.
  - `SHOW CONFIG` now also shows the default values of the settings.
- Fixes

  - Repair compilation on systems with older systemd versions. This was broken in 1.14.0. (#505)
  - The makefile target to build Windows binary zip packages has been repaired.
  - Long command-line options now also work on Windows.
  - Fix the behavior of the global `auth_user` setting. The old behavior was confusing and fragile as it depended on the order in the configuration file. This is no longer the case. (#391, #393)
- Cleanups
  - Improve test stability and portability.
  - Modernize Autoconf-related code.
  - Disable deprecation compiler warnings from OpenSSL 3.0.0.

PgBouncer 1.14.x

2020-06-11 - PgBouncer 1.14.0 - “La ritrovata magia”

- Features
  - Add SCRAM authentication pass-through. This allows using encrypted SCRAM secrets in PgBouncer (either in `userlist.txt` or from `auth_query`) for logging into servers.
  - Add support for systemd socket activation. This is especially useful to let systemd handle the creation of the Unix-domain sockets on systems where access to `/var/run/postgresql` is restricted.
  - Add support for Unix-domain sockets on Windows.
- Cleanups
  - Add an alternative smaller sample configuration file `pgbouncer-minimal.ini` for testing or deployment.

PgBouncer 1.13.x

- Features
    - Add configuration setting `tcp_user_timeout`, to set the corresponding socket option.
    - `client_tls_protocols` and `server_tls_protocols` now default to `secure`, which means only TLS 1.2 and TLS 1.3 are enabled. Older versions are still supported, they are just not turned on by default.
    - Add support for systemd service notifications. Right now, this allows using `Type=notify` service units. More integration is planned for future versions.
- Fixes
    - Fix multiline log messages (libusual #24)
    - Handle null user names returned from `auth_query` properly (#340)
- Cleanups
    - The Debian packaging files under `debian` have been removed. It is recommended to use the packages from https://apt.postgresql.org/.
    - Numerous fixes and improvements in the test suite
    - The tests no longer try to use sudo by default. This can now be activated explicitly by setting the environment variable `USE_SUDO`.
    - The libevent API use was updated to use version 2 style interfaces and to no longer use deprecated interfaces from version 1.

PgBouncer 1.12.x

2019-10-17 - PgBouncer 1.12.0 - “It’s about learning and getting better”

This release contains a variety of minor enhancements and fixes.

- Features
    - Add a setting to turn on the `SO_REUSEPORT` socket option. On some operating systems, this allows running multiple PgBouncer instances on the same host listening on the same port and having the kernel distribute the connections automatically.

- Send the output of `SHOW VERSION` as a normal result row instead of a NOTICE message. This makes it easier to consume and is consistent with other `SHOW` commands.

- Fixes
  - Send statistics columns as `numeric` instead of `bigint`. This avoids some client libraries failing on values that overflow the `bigint` range. (#360, #401)
  - Fix issue with PAM users losing their password. (#285)
  - Accept SCRAM channel binding enabled clients. Previously, a client supporting channel binding (that is, PostgreSQL 11+) would get a connection failure when connecting to PgBouncer in certain situations. (PgBouncer does not support channel binding. This change just fixes support for clients that offer it.)
  - Fix compilation with newer versions of musl-libc (used by Alpine Linux).
- Cleanups
  - Add `make check` target. This allows running all the tests from a single command.
  - Remove references to the PostgreSQL wiki. All information is now either in the PgBouncer documentation or on the web site.
  - Remove support for Libevent version 1.x. Libevent 2.x is now required. Libevent is now detected using pkg-config.
  - Fix compiler warnings on macOS and Windows. The build on these platforms should now be free of warnings.
  - Fix some warnings from LLVM scan-build.

PgBouncer 1.11.x

2019-08-27 - PgBouncer 1.11.0 - “Instinct for Greatness”

- Features
  - Add support for SCRAM authentication for clients and servers. A new authentication type `scram-sha-256` is added.

- Add option `log_stats` to disable printing stats to log. (#287)
- Add time zone to log timestamps.
- Put PID into [brackets] in log prefix.

- Fixes
  - Fix OpenSSL configure test when running against newer OpenSSL with `-Werror`.
  - Fix wait time computation with `auth_user`. This would either crash or report garbage values for wait time. (#393)
  - Handle GSSENCRequest packet, added in PostgreSQL 12. It doesn't do anything right now, but it avoids confusing error messages about "bad packet header".
- Cleanups
  - Many improvements in the test suite and several new tests
  - Fix several compiler warnings on Windows.
  - Expand documentation of the `[users]` section and add to example config file. (#330)

PgBouncer 1.10.x

2019-07-01 - PgBouncer 1.10.0 - "Afraid of the World"

- Features
  - Add support for enabling and disabling TLS 1.3. (TLS 1.3 was already supported, depending on the OpenSSL library, but now the configuration settings to pick the TLS protocol versions also support it.)
- Fixes
  - Fix TLS 1.3 support. This was broken with OpenSSL 1.1.1 and 1.1.1a (but not before or after).
  - Fix a rare crash in `SHOW FDS` (#311).
  - Fix an issue that could lead to prolonged downtime if many cancel requests arrive (#329).
  - Avoid "unexpected response from login query" after a postgres reload (#220).

- Cleanups
  - Make various log and error messages more precise.
  - Fix issues found by Coverity (none had a significant impact in practice).
  - Improve and document all test scripts.
  - Add additional SHOW commands to the documentation.
  - Convert the documentation from rst to Markdown.
  - Python scripts in the source tree are all compatible with Python 3 now.

PgBouncer 1.9.x

2018-08-13 - PgBouncer 1.9.0 - "Chaos Survival"

- Features
  - RECONNECT command
  - WAIT_CLOSE command
  - Fast close - Disconnect a server in session pool mode immediately if it is in "close_needed" (reconnect) mode.
  - Add close_needed column to SHOW SERVERS
- Fixes
  - Avoid double-free in parse_filename
  - Avoid NULL pointer deref in parse_line
- Cleanups
  - Port mkauth.py to Python 3
  - Improve signals documentation
  - Improve quick start documentation
  - Document SET command
  - Correct list of required software
  - Fix -Wimplicit-fallthrough warnings

- Document that KILL requires RESUME afterwards
- Clarify documentation of server_lifetime
- Typos and capitalization fixes in messages and docs
- Fix psql invocation in tests
- Various other test setup improvements

PgBouncer 1.8.x

2017-12-20 - PgBouncer 1.8.1 - “Ground-and-pound Mentality”

- Fixes
  - Include file `include/pam.h` into distribution tarball. This prevented the 1.8 tarball from building at all.

2017-12-19 - PgBouncer 1.8 - “Confident at the Helm”

- Features
  - Support PAM authentication. (Enable with `--with-pam`.)
  - Add `paused` and `disabled` fields to `SHOW DATABASES` output.
  - Add `maxwait_us` field to `SHOW POOLS` output.
  - Add `wait` and `wait_us` fields to `SHOW` commands output.
  - Add new commands `SHOW STATS_TOTALS` and `SHOW STATS_AVERAGES`.
  - Track queries and transactions separately in `SHOW STATS`. The fields `total_requests`, `avg_req`, and `avg_query` have been replaced by new fields.
  - Add `wait_time` to `SHOW STATS`.
- Fixes
  - Updated libusual supports OpenSSL 1.1.
  - Do not attempt to use TLS on Unix sockets.

- Several other hba parsing fixes.
- Fix race condition when canceling query. (#141)

- Cleanups
  - `auth_user` setting is now also allowed globally, not only per database. (#142)
  - Set console client and server encoding to `UTF8`.

PgBouncer 1.7.x

2016-02-26 - PgBouncer 1.7.2 - “Finally Airborne”

- Fixes
  - Fix crash on stale pidfile removal. Problem introduced in 1.7.1.
  - Disable cleanup - it breaks takeover and is not useful for production loads. Problem introduced in 1.7.1.
  - After takeover, wait until pidfile is gone before booting. Slow shutdown due to memory cleanup exposed existing race. (#113)
- Cleanups
  - Make build reproducible by dropping DBGVER handling. (#112)
  - Antimake: Sort file list from $(wildcard), newer gmake does not sort it anymore. (#111)
  - Show libssl version in log.
  - deb: Turn on full hardening.

2016-02-18 - PgBouncer 1.7.1 - “Forward To Five Friends Or Else”

WARNING: Since version 1.7, `server_reset_query` is not executed when database is in transaction-pooling mode. Seems this was not highlighted enough in 1.7 announcement. If your apps depend on that happening, use `server_reset_query_always` to restore previous behaviour.

connection (without leaking) instead expected 20-30k. Something to keep an eye on when using TLS.

- Fixes
  - TLS: Rename sslmode “disabled” to “disable” as that is what PostgreSQL uses.
  - TLS: `client_tls_sslmode=verify-ca/-full` now reject connections without client certificate. (#104)
  - TLS: `client_tls_sslmode=allow/require` do validate client certificate if sent. Previously they left cert validation unconfigured so connections with client cert failed. (#105)
  - Fix memleak when freeing database.
  - Fix potential memleak in tls_handshake().
  - Fix EOF handling in tls_handshake().
  - Fix too small memset in asn1_time_parse compat.
  - Fix non-TLS (`--without-openssl`) build. (#101)
  - Fix various issues with Windows build. (#100)
- Cleanups
  - TLS: Use SSL_MODE_RELEASE_BUFFERS to decrease memory usage of inactive connections.
  - Clean allocated memory on exit. Helps to run memory-leak checkers.
  - Improve `server_reset_query` documentation. (#110)
  - Add TLS options to sample config.

2015-12-18 - PgBouncer 1.7 - “Colors Vary After Resurrection”

- Features
  - Support TLS connections. OpenSSL/LibreSSL is used as backend implementation.
  - Support authentication via TLS client certificate.
  - Support “peer” authentication on Unix sockets.

- Cleanups
  - Set `query_wait_timeout` to 120s by default. Current default (0) causes infinite queueing, which is not useful. That means if client has pending query and has not been assigned to server connection, the client connection will be dropped.
  - Disable `server_reset_query_always` by default. Now reset query is used only in pools that are in session mode.
  - Increase pkt_buf to 4096 bytes. Improves performance with TLS. The behaviour is probably load-specific, but it should be safe to do as since v1.2 the packet buffers are split from connections and used lazily from pool.
  - Support pipelining count expected ReadyForQuery packets. This avoids releasing server too early. Fixes #52.
  - Improved sbuf_loopcnt logic - socket is guarateed to be reprocessed even if there are no event from socket. Required for TLS as it has it's own buffering.
  - Adapt system tests to work with modern BSD and MacOS. (Eric Radman)
  - Remove **crypt** auth. It's obsolete and not supported by PostgreSQL since 8.4.
  - Fix plain "–with-cares" configure option - without argument it was broken.

PgBouncer 1.6.x

2015-09-03 - PgBouncer 1.6.1 - "Studio Audience Approves"

- Features
  - New setting: `server_reset_query_always`. When set, disables `server_reset_query` use on non-session pools. PgBouncer introduces per-pool pool_mode, but session-pooling and transaction-pooling should not use same reset query. In fact, transaction-pooling should not use any reset query.

    It is set in 1.6.x, but will be disabled in 1.7.

client asks non-existing username, client will log in as `auth_user`. Not good.

CVE-2015-6817

- Skip NoticeResponse in handle_auth_response. Otherwise verbose log levels on server cause login failures.
- console: Fill `auth_user` when auth_type=any. Otherwise logging can crash (#67).
- Various portability fixes (OpenBSD, Solaris, OSX).

2015-08-01 - PgBouncer 1.6 - “Zombies of the future”

- Features
  - Load user password hash from postgres database. New parameters:

    auth_user user to use for connecting same db and fetching user info. Can be set per-database too.

    auth_query SQL query to run under auth_user. Default: “SELECT usename, passwd FROM pg_shadow WHERE usename=$1”

    (Cody Cutrer)
  - Pooling mode can be configured both per-database and per-user. (Cody Cutrer)
  - Per-database and per-user connection limits: max_db_connections and max_user_connections. (Cody Cutrer / Pavel Stehule)
  - Add DISABLE/ENABLE commands to prevent new connections. (William Grant)

preferred backend now, and probably will be **only** backend in the future, as it's pointless to support zoo of inadequate libraries.

SNAFU: c-ares versions <= 1.10 have bug which breaks CNAME-s support when IPv6 has been enabled. (Fixed upstream.) As a workaround, c-ares <= 1.10 is used IPv4-only. So PgBouncer will drop other backends only when c-ares >1.10 (still unreleased) has been out some time…

  - Show remote_pid in SHOW CLIENTS/SERVERS. Available for clients that connect over unix sockets and both tcp and unix socket server. In case of tcp-server, the pid is taken from cancel key.
  - Add separate config param (dns_nxdomain_ttl) for controlling negative dns caching. (Cody Cutrer)
  - Add the client host IP address and port to application_name. This is enabled by a config parameter application_name_add_host which defaults to 'off'. (Andrew Dunstan)
  - Config files have '%include FILENAME' directive to allow configuration to be split into several files. (Andrew Dunstan)
- Cleanups
  - log: wrap ipv6 address with []
  - log: On connect to server, show local ip and port
  - win32: use gnu-style for long args: –foo
  - Allow numbers in hostname, always try to parse with inet_pton
  - Fix deallocate_all() in FAQ
  - Fix incorrect keyword in example config file (Magnus Hagander)
  - Allow comments (with ';') in auth files. (Guillaume Aubert)

  - fix launching new connections during maintenance (Cody Cutrer)
  - don’t load auth file twice at boot (Cody Cutrer)
  - Proper invalidation for autodbs
  - ipv6: Set IPV6_V6ONLY on listen socket.
  - win32: Don’t set SO_REUSEADDR on listen socket.
  - Fix IPv6 address memcpy
  - Fix cancellation of of waiting clients. (Mathieu Fenniak)
  - Small bug fix, must check calloc result (Heikki Linnakangas)
  - Add newline at the end of the PID file (Peter Eisentraut)
  - Don’t allow new server connections when PAUSE was issued. (Petr Jelinek)
  - Fix ‘bad packet’ during login when header is delayed. (Michał Trojnara, Marko Kreen)
  - Fix errors detected by Coverty. (Euler Taveira)
  - Disable server_idle_timeout when server count gets below min_pool (#60) (Marko Kreen)

PgBouncer 1.5.x

2015-04-09 - PgBouncer 1.5.5 - “Play Dead To Win”

- Fixes
  - Fix remote crash - invalid packet order causes lookup of NULL pointer. Not exploitable, just DoS.

2012-11-28 - PgBouncer 1.5.4 - “No Leaks, Potty-Training Successful”

- Fixes
  - DNS: Fix memory leak in getaddrinfo_a() backend.
  - DNS: Fix memory leak in udns backend.
  - DNS: Fix stats calculation.

- Fix compiler dependency support check in configure.
- Few documentation fixes.

2012-09-12 - PgBouncer 1.5.3 - "Quantum Toaster"

- Critical fix
    - Too long database names can lead to crash, which is remotely triggerable if autodbs are enabled.

      The original checks assumed all names come from config files, thus using fatal() was fine, but when autodbs are enabled

        - by '*' in [databases] section - the database name can come from network thus making remote shutdown possible.

      CVE-2012-4575
- Minor Features
    - max_packet_size - config parameter to tune maximum packet size that is allowed through. Default is kept same: (2G-1), but now it can be made smaller.
    - In case of unparsable packet header, show it in hex in log and error message.
- Fixes
    - AntiMake: it used $(relpath) and $(abspath) to manipulate pathnames, but the result was build failure when source tree path contained symlinks. The code is now changed to work on plain strings only.
    - console: now SET can be used to set empty string values.
    - config.txt: show that all timeouts can be set in floats. This is well-hidden feature introduced in 1.4.

- Fixes
    - Due to mistake, reserve_pool_timeout was taken in microseconds, not seconds, effectively activating reserve pool immediately when pool got full. Now use it as seconds, as was intended. (Noticed by Keyur Govande)

2012-04-17 - PgBouncer 1.5.1 - “Abort, Retry, Ignore?”

- Features
    - Parameters to tune permissions on unix socket: unix_socket_mode=0777, unix_socket_group=’’.
- Fixes
    - Allow empty string for server-side variable - this is needed to get “application_name” properly working, as it’s the only parameter that does not have server-side default.
    - If connect string changes, require refresh of server parameters. Previously PgBouncer continued with old parameters, which breaks in case of Postgres upgrade.
    - If autodb connect string changes, drop old connections.
    - cf_setint: Use strtol() instead atoi() to parse integer config parameters. It allows hex, octal and better error detection.
    - Use sigqueue() to detect union sigval existence - fixes compilation on HPUX.
    - Remove ‘git’ command from Makefile, it throws random errors in case of plain-tarball build.
    - Document stats_period parameter. This tunes the period for stats output.
    - Require Asciidoc >= 8.4, seems docs are not compatible with earlier versions anymore.
    - Stop trying to retry on EINTR from close().

2012-01-05 - PgBouncer 1.5 - “Bouncing Satisfied Clients Since 2007”

If you use more than 8 IPs behind one DNS name, you now need to use EDNS0 protocol to query. Only getaddrinfo_a()/getaddrinfo() and UDNS backends support it, libevent 1.x/2.x does not. To

- Features
  - Detect DNS reply changes and invalidate connections to IPs no longer present in latest reply. (Petr Jelinek)
  - DNS zone serial based hostname invalidation. When option dns_zone_check_period is set, all DNS zones will be queried for SOA, and when serial has changed, all hostnames will be queried. This is needed to get deterministic connection invalidation, because invalidation on lookup is useless when no lookups are performed. Works only with new UDNS backend.
  - New SHOW DNS_HOSTS, SHOW DNS_ZONES commands to examine DNS cache.
  - New param: min_pool_size - avoids dropping all connections when there is no load. (Filip Rembiałkowski)
  - idle_in_transaction_timeout - kill transaction if idle too long. Not set by default.
  - New libudns backend for DNS lookups. More featureful than evdns. Use –with-udns to activate. Does not work with IPv6 yet.
  - KILL command, to immediately kill all connections for one database. (Michael Tharp)
  - Move to Antimake build system to have better looking Makefiles. Now GNU Make 3.81+ is required for building.
- Fixes
  - DNS now works with IPv6 hostnames.
  - Don’t change connection state when NOTIFY arrives from server.
  - Various documentation fixes. (Dan McGee)
  - Console: Support ident quoting with “”. Originally we did not have any commands that took database names, so no quoting was needed.
  - Console: allow numbers at the start of word regex. Trying to use strict parser makes things too complex here.
  - Don’t expire auto DBs that are paused. (Michael Tharp)

- When user= without password= is in database connect string, password will be taken from userlist.
- Parse '*' properly in takeover code.
- autogen.sh: work with older autoconf/automake.
- Fix run-as-service crash on win32 due to bad basename() from mingw/msvc runtime. Now compat basename() is always used.

PgBouncer 1.4.x

2011-06-16 - PgBouncer 1.4.2 - "Strike-First Algorithm"

Affected OS-es: *BSD, Solaris, Win32.

- Portability Fixes
  - Give CFLAGS to linker. Needed when using pthread-based getaddrinfo_a() fallback.
  - lib/find_modules.sh: Replace split() with index()+substr(). This should make it work with older AWKs.
  - <usual/endian.h>: Ignore system htoX/Xtoh defines. There may be only subset of macros defined.
  - <usual/signal.h>: Separate compat sigval from compat sigevent
  - <usual/socket.h>: Include <sys/uio.h> to get iovec
  - <usual/time.h>: Better function autodetection on win32
  - <usual/base_win32.h>: Remove duplicate sigval/sigevent declaration

2011-04-01 - PgBouncer 1.4.1 - "It Was All An Act"

- Features

  - Support listening/connect for IPv6 addresses. (Hannu Krosing)

- console: Send PgBouncer version as 'server_version' to client.
- Important Fixes
  - Disable getaddrinfo_a() on glibc < 2.9 as it crashes on older versions.

    Notable affected OS'es: RHEL/CentOS 5.x (glibc 2.5), Ubuntu 8.04 (glibc 2.7). Also Debian/lenny (glibc 2.7) which has non-crashing getaddrinfo_a() but we have no good way to detect it.

    Please use libevent 2.x on such OS'es, fallback getaddrinfo_a() is not meant for production systems. And read new 'DNS lookup support' section in README to see how DNS backend is picked.

    (Hubert Depesz Lubaczewski, Dominique Hermsdorff, David Sommerseth)
  - Default to –enable-evdns if libevent 2.x is used.
  - Turn on tcp_keepalive by default, as that's what Postgres also does. (Hubert Depesz Lubaczewski)
  - Set default server_reset_query to DISCARD ALL to be compatible with Postgres by default.
  - win32: Fix crashes with NULL unix socket addr. (Hiroshi Saito)
  - Fix autodb cleanup: old cleanup code was mixing up databases and pools: as soon as one empty pool was found, the database was tagged as 'idle', potentially later killing database with active users.

    Reported-By: Hubert Depesz Lubaczewski
- Fixes

  - Enable pthread compilation if compat getaddrinfo_a is used.
  - release_server missed setting ->last_lifetime_disconnect on lifetime disconnect. (Emmanuel Courreges)
  - win32: fix auth file on DOS line endings - load_file() did not take account of file shringage when loading. (Rich Schaaf)
  - <usual/endian.h>: add autoconf detection for enc/dec functions so it would not create conflicts on BSD. (James Pye)
  - Don’t crash when config file does not exist. (Lou Picciano)
  - Don’t crash on DNS lookup failure when logging on noise level (-v -v). (Hubert Depesz Lubaczewski, Dominique Hermsdorff)
  - Use backticks instead of $(cmd) in find_modules.sh to make it more portable. (Lou Picciano)
  - Use ‘awk’ instead of ‘sed’ in find_modules.sh to make it more portable. (Giorgio Valoti)
  - Log active async DNS backend info on startup.
  - Fix –disable-evdns to mean ‘no’ instead ‘yes’.
  - Mention in docs that -R requires unix_socket_dir.
  - Discuss server_reset_query in faq.txt.
  - Restore lost memset in slab allocator
  - Various minor portability fixes in libusual.

2011-01-11 - PgBouncer 1.4 - “Gore Code”

- Features

  - Async DNS lookup - instead of resolving hostnames at reload time, the names are now resolved at connect time, with configurable caching. (See dns_max_ttl parameter.)

    By default it uses getaddrinfo_a() (glibc) as backend, if it does not exist, then getaddrinfo_a() is emulated via blocking(!) getaddrinfo().

libevent 2.0 seems OK.
    - New config var: syslog_ident, to tune syslog name.
    - Proper support for `application_name` startup parameter.
    - Command line long options (Guillaume Lelarge)
    - Solaris portability fixes (Hubert Depesz Lubaczewski)
    - New config var: disable_pqexec. Highly-paranoid environments can disable Simple Query Protocol with that. Requires apps that use only Extended Query Protocol.
    - Postgres compat: if database name is empty in startup packet, use user name as database.
- Fixes
    - DateStyle and TimeZone server params need to use exact case.
    - Console: send datetime, timezone and stdstr server params to client.
- Internal cleanups
    - Use libusual library for low-level utility functions.
    - Remove fixed-length limit from server params.

PgBouncer 1.3.x

2010-09-09 - PgBouncer 1.3.4 - “Bouncer is always right”

- Fixes
    - Apply fast-fail logic at connect time. So if server is failing, the clients get error when connecting.

  - Ignore application_name parameter by default. This avoids the need for all Postgres 9.0 users to add it into ignore_startup_parameters= themselves.
  - Correct pg_auth quoting. '' is not used there.
  - Better error reporting on console, show incoming query to user.
  - Support OS'es (OpenBSD) where tv_sec is not time_t.
  - Avoid too noisy warnings on gcc 4.5.

2010-05-10 - PgBouncer 1.3.3 - "NSFW"

- Improvements
  - Make listen(2) argument configurable: listen_backlog. This is useful on OS'es, where system max allowed is configurable.
  - Improve disconnect messages to show what username or dbname caused login to fail.
- Fixes
  - Move fast-fail relaunch logic around. Old one was annoying in case of permanently broken databases or users, by trying to retry even if there is no clients who want to login.
  - Make logging functions keep old errno, otherwise pgbouncer may act funny on higher loglevels and logging problems.
  - Increase the size of various startup-related buffers to handle EDB more noisy startup.
  - Detect V2 protocol startup request and give clear reason for disconnect.

2010-03-15 - PgBouncer 1.3.2 - "Boomerang Bullet"

- Fixes

  - New config var 'query_wait_timeout'. If client does not get server connection in this many seconds, it will be killed.

This together with previous fix avoids unnecessary stalls if a database has gone down.

- Track libevent state in sbuf.c to avoid double event_del(). Although it usually is safe, it does not seem to work 100%. Now we should always know whether it has been called or not.
- Disable maintenance during SUSPEND. Otherwise with short timeouts the old bouncer could close few connections after sending them over.
- Apply client_login_timeout to clients waiting for welcome packet (first server connection). Otherwise they can stay waiting infinitely, unless there is query_timeout set.
- win32: Add switch -U/-P to -regservice to let user pick account to run service under. Old automatic choice between Local Service and Local System was not reliable enough.
- console: Remove \0 from end of text columns. It was hard to notice, as C clients were fine with it.
- Documentation improvements. (Greg Sabino Mullane)
- Clarify few login-related log messages.
- Change logging level for pooler-sent errors (usually on disconnect) from INFO to WARNING, as they signify problems.
- Change log message for query_timeout to “query timeout”.

2009-07-06 - PgBouncer 1.3.1 - “Now fully conforming to NSA monitoring requirements”

- Fixes

for more data which will not appear.
  - Make database reconfigure immediate. Currently old connections could be reused after SIGHUP.
  - Fix SHOW DATABASES which was broken due to column addition.
  - Console access was disabled when “auth_type=any” as pgbouncer dropped username. Fix: if “auth_type=any”, allow any user to console as admin.
  - Fix bad CUSTOM_ALIGN macro. Luckily it’s unused if OS already defines ALIGN macro thus seems the bug has not happened in wild.
  - win32: call WSAStartup() always, not only in daemon mode as config parsing wants to resolve hosts.
  - win32: put quotes around config filename in service cmdline to allow spaces in paths. Executable path does not seem to need it due to some win32 magic.
  - Add STATS to SHOW HELP text.
  - doc/usage.txt: the time units in console results are in microseconds, not milliseconds.

2009-02-18 - PgBouncer 1.3 - “New Ki-Smash Finishing Move”

- Features

  - IANA has assigned port 6432 to be official port for PgBouncer. Thus the default port number has changed to 6432. Existing individual users do not need to change, but if you distribute packages of PgBouncer, please change the package default to official port.

  - Dynamic database creation (David Galoyan)

    Now you can define database with name “*”. If defined, it’s connect string will be used for all undefined databases. Useful mostly for test / dev environments.

  - Windows support (Hiroshi Saito)

service_name in config. Then:

```
> pgbouncer.exe config.ini -regservice
> net start SERVICE_NAME
```

To stop and unregister:

```
> net stop SERVICE_NAME
> pgbouncer.exe config.ini -unregservice
```

To use Windows Event Log, event DLL needs to be registered first:

```
> regsrv32 pgbevent.dll
```

Afterwards you can set “syslog = 1” in config.

- Minor features
    - Database names in config file can now be quoted with standard SQL ident quoting, to allow non-standard characters in db names.
    - New tunables: ‘reserve_pool_size’ and ‘reserve_pool_timeout’. In case there are clients in pool that have waited more that ‘reserve_pool_timeout’ seconds, ‘reserve_pool_size’ specifies the number of connections that can be added to pool. It can also set per-pool with ‘reserve_pool’ connection variable.
    - New tunable ‘sbuf_loopcnt’ to limit time spent on one socket.

      In some situations - eg SMP server, local Postgres and fast network - pgbouncer can run recv()->send() loop many times without blocking on either side. But that means other connections will stall for a long time. To make processing more fair, limit the times of

  Thanks to Alexander Schöcke for report and testing.
  - crypt() authentication is now optional, as it was removed from Postgres. If OS does not provide it, pgbouncer works fine without it.

  - Add milliseconds to log timestamps.

  - Replace old MD5 implementation with more compact one.

  - Update ISC licence with the FSF clarification.
- Fixes

  - In case event_del() reports failure, just proceed with cleanup. Previously pgbouncer retried it, in case the failure was due ENOMEM. But this has caused log floods with infinite repeats, so it seems libevent does not like it.

    Why event_del() report failure first time is still mystery.

  - –enable-debug now just toggles whether debug info is stripped from binary. It no longer plays with -fomit-frame-pointer as it's dangerous.

  - Fix include order, as otherwise system includes could come before internal ones. Was problem for new md5.h include file.

  - Include COPYRIGHT file in .tgz…

PgBouncer 1.2.x

2008-08-08 - PgBouncer 1.2.3 - "Carefully Selected Bytes"

- Fixes

- Use ‘$(MAKE)’ instead ‘make’ for recursion (Jørgen Austvik)
- Define _GNU_SOURCE as glibc is useless otherwise.
- Let the libevent 1.1 pass link test so we can later report “1.3b+ needed”
- Detect stale pidfile and remove it.

Thanks to Devrim GÜNDÜZ and Bjoern Metzdorf for problem reports and testing.

2008-08-06 - PgBouncer 1.2.2 - “Barf-bag Included”

- Fixes
  - Remove ‘drop_on_error’, it was a bad idea. It was added as workaround for broken plan cache behaviour in Postgres, but can cause damage in common case when some queries always return error.

2008-08-04 - PgBouncer 1.2.1 - “Waterproof”

- Features
  - New parameter ‘drop_on_error’ - if server throws error the connection will not be reused but dropped after client finished with it. This is needed to refresh plan cache. Automatic refresh does not work even in 8.3. Defaults to 1.
- Fixes
  - SHOW SOCKETS/CLIENTS/SERVERS: Don’t crash if socket has no buffer.
  - Fix infinite loop on SUSPEND if suspend_timeout triggers.
- Minor cleanups
  - Use <sys/uio.h> for ‘struct iovec’.
  - Cancel shutdown (from SIGINT) on RESUME/SIGUSR2, otherwise it will trigger on next PAUSE.
  - Proper log message if console operation is canceled.

PgBouncer 1.2 now requires libevent version 1.3b or newer. Older libevent versions crash with new restart code.

- Features
    - Command line option (-u) and config parameter (user=) to support user switching at startup. Also now pgbouncer refuses to run as root.

      (Jacob Coby)
    - More descriptive usage text (-h). (Jacob Coby)
    - New database option: connect_query to allow run a query on new connections before they are taken into use.

      (Teodor Sigaev)
    - New config var 'ignore_startup_parameters' to allow and ignore extra parameters in startup packet. By default only 'database' and 'user' are allowed, all others raise error. This is needed to tolerate overenthusiastic JDBC wanting to unconditionally set 'extra_float_digits=2' in startup packet.
    - Logging to syslog: new parameters syslog=0/1 and syslog_facility=daemon/user/local0.
    - Less scary online restart (-R)
        - Move FD loading before fork, so it logs to console and can be canceled by ^C
        - Keep SHUTDOWN after fork, so ^C would be safe
        - A connect() is attempted to unix socket to see if anyone is listening. Now -R can be used even when no previous process was running. If there is previous process, but -R

    - SHOW TOTALS that shows stats summary (as goes to log) plus mem usage.
    - SHOW ACTIVE_SOCKETS - like show sockets; but filter only active ones.
- Less visible features
  - suspend_timeout - drop stalled conns and long logins. This brings additional safety to reboot.
  - When remote database throws error on logging in, notify clients.
  - Removing a database from config and reloading works - all connections are killed and the database is removed.
  - Fake some parameters on console SHOW/SET commands to be more Postgres-like. That was needed to allow psycopg to connect to console. (client_encoding/default_transaction_isolation/datestyle/timezone)
  - Make server_lifetime=0 disconnect server connection immediately after first use. Previously “0” made PgBouncer ignore server age. As this behavior was undocumented, there should not be any users depending on it.
  - Internal improvements:
    - Packet buffers are allocated lazily and reused. This should bring huge decrease in memory usage. This also makes realistic to use big pktbuf with lot of connections.
    - Lot’s of error handling improvements, PgBouncer should now survive OOM situations gracefully.
    - Use slab allocator for memory management.

  - Only single accept() was issued per event loop which could cause connection backlog when having high amount of connection attempts. Now the listening socket is always drained fully, which should fix this.
  - Handle EINTR from connect().
  - Make configure.ac compatible with autoconf 2.59.
  - Solaris compatibility fixes (Magne Mæhre)

PgBouncer 1.1.x

2007-12-10 - PgBouncer 1.1.2 - “The Hammer”

- Features
  - Disconnects because of server_lifetime are now separated by (server_lifetime / pool_size) seconds. This avoids pgbouncer causing reconnect floods.
- Fixes
  - Online upgrade 1.0 -> 1.1 problems:
    - 1.0 does not track server parameters, so they stay NULL but 1.1 did not expect it and crashed.
    - If server params are unknown, but client ones are set, then issue a SET for them, instead complaining.
  - Remove temp debug statements that were accidentally left in code on INFO level, so they polluted logs.
  - Unbroke debian/changelog
- Cleanup
  - reorder struct SBuf fields to get better alignment for buffer.

2007-10-26 - PgBouncer 1.1.1 - “Breakdancing Bee”

This caused problem on 8.1 which does not allow touching standard_conforming_strings. (Thanks to Dimitri Fontaine for report & testing.)
  - Some doc fixes.
  - Include doc/fixman.py in .tgz.

2007-10-09 - PgBouncer 1.1 - “Mad-Hat Toolbox”

- Features
  - Keep track of following server parameters:

    ```
    client_encoding  datestyle, timezone, standard_conforming_strings
    ```

  - Database connect string enhancements:
    - Accept hostname in host=
    - Accept custom unix socket location in host=
    - Accept quoted values: password=’ asd’’foo’
  - New config var: server_reset_query, to be sent immediately after release
  - New config var: server_round_robin, to switch between LIFO and RR.
  - Cancel pkt sent for idle connection does not drop it anymore.
  - Cancel with ^C from psql works for SUSPEND / PAUSE.
  - Print FD limits on startup.
  - When suspending, try to hit packet boundary ASAP.
  - Add ‘timezone’ to database parameters.
  - Use longlived logfile fd. Reopened on SIGHUP / RELOAD;
  - Local connection endpoint info in SHOW SERVERS/CLIENTS/SOCKETS.
- Code cleanup
  - More debug log messages include socket info.

- Fixes
    - Detect invalid pkt headers better.
    - auth_file modification check was broken, which made pgbouncer reload it too often.

PgBouncer 1.0.x

2007-06-18 - PgBouncer 1.0.8 - "Undead Shovel Jutsu"

- Fixes
    - Fix crash in cancel packet handling. (^C from psql)
- Features
    - PAUSE ; RESUME ; works now.
    - Cleanup of console command parsing.
    - Disable expensive in-list assert check.

2007-04-19 - PgBouncer 1.0.7 - "With Vitamin A-Z"

- Fixes
    - Several error/notice packets with send() blocking between triggered assert. Fix it by removing flushing logic altogether. As pgbouncer does not actively buffer anything, its not needed. It was a remnant from the time when buffering was pushed to kernel with MSG_MORE.
    - Additionally avoid calling recv() logic when sending unblocks.
    - List search code for admin_users and stats_users mishandled partial finds. Fix it.
    - Standardise UNIX socket peer UID finding to getpeereid().

2007-04-12 - PgBouncer 1.0.6 - "Daily Dose"

- Fixes

- Compilation fix for FreeBSD, <sys/ucred.h> requires <sys/param.h> there. Thanks go to Robert Gogolok for report.

2007-04-11 - PgBouncer 1.0.5 - “Enough for today”

- Fixes
  - Fix online-restart bugs:
    - Set ->ready for idle servers.
    - Remove obsolete code from use_client_socket()
    - Disable maintenance during the takeover.

2007-04-11 - PgBouncer 1.0.4 - “Last ‘last’ bug”

- Fixes
  - Notice from idle server tagged server dirty. release_server() did not expect it. Fix it by dropping them.

2007-04-11 - PgBouncer 1.0.3 - “Fearless Fork”

- Fixes
  - Some error handling was missing in login path, so dying connection there could trigger asserts.
  - Cleanup of asserts in sbuf.c to catch problems earlier.
  - Create core when Assert() triggers.
- New stuff
  - New config vars: log_connections, log_disconnections, log_pooler_errors to turn on/off noise.
  - Config var: client_login_timeout to kill dead connections in login phase that could stall SUSPEND and thus online restart.

- Fixes
  - libevent may report a deleted event inside same loop. Avoid socket reuse for one loop.
  - release_server() from disconnect_client() didn’t look it the packet was actually sent.

2007-03-15 - PgBouncer 1.0.1 - “Alien technology”

- Fixes
  - Mixed usage of cached and non-cached time, plus unsigned usec_t typedef created spurious query_timeout errors.
  - Fix rare case when socket woken up from send-wait could stay stalling.
  - More fair queueing of server connections. Before, a new query could get a server connections before older one.
  - Delay server release until everything is guaranteed to be sent.
- Features
  - SHOW SOCKETS command to have detailed info about state state.
  - Put PgSocket ptr to log, to help tracking one connection.
  - In console, allow SELECT in place of SHOW.
  - Various code cleanups.

2007-03-13 - PgBouncer 1.0 - “Tuunitud bemm”

- First public release.