pgBouncer Foreign Data Wrapper

Introduction

pgbouncer_fdw provides a direct SQL interface to the pgbouncer SHOW commands. It takes advantage of the dblink_fdw feature to provide a more typical, table-like interface to the current status of your pgbouncer server(s). This makes it easier to set up monitoring or other services that require direct access to pgbouncer statistics. # PgBouncer Foreign Data Wrapper

Introduction

pgbouncer_fdw provides a direct SQL interface to the PgBouncer SHOW commands. It takes advantage of the dblink_fdw feature to provide a more typical, table-like interface to the current status of your PgBouncer server(s). This makes it easier to set up monitoring or other services that require direct access to PgBouncer statistics.

Requirements

- PostgreSQL 11+ - https://www.postgresql.org
- dblink (contrib module) - https://www.postgresql.org/docs/current/dblink.html
- PgBouncer 1.17+ - https://pgbouncer.github.io

Installation

Database Users

For basic monitoring of statistics, whichever database role(s) you will be using in the user mapping below will have to be added to the `stats_users` list in the PgBouncer configuration (pgbouncer.ini). You will also need to add any of these roles to the PgBouncer `auth_users` file. The auth_query method in PgBouncer cannot be used to connect to the special `pgbouncer` database where the SHOW commands must be run. Ensure the role(s) used are able to connect to the special `pgbouncer` database and run the SHOW commands before setting up the FDW.

For running of the command functions, roles will have to be added to the `admin_users` list in the PgBouncer configuration. It is not recommended that your monitoring roles also be given admin console access. It is recommended to have a separate database role for a separate user mapping to allow access to the PgBouncer to run these commands.

Extension Setup

1. If installing from source, run make from the source directory

```
make install
```

2. The dblink extension must be created in a schema that is within the search path of the role that will be used for the user mapping below. A default location of the PUBLIC schema is the easiest.

```
CREATE EXTENSION dblink;
```

3. Create the extension in the monitoring database. PgBouncer statistics are global so it only needs to be monitored from a single database. If you have multiple databases in your cluster, it is recommended to just install it to the default `postgres` database, or whichever one is being used as a global database that will never be dropped.

```
CREATE EXTENSION pgbouncer_fdw;
```

4. Create one or more FDW servers in the same database where the extension was installed.

    a. If only a single PgBouncer server is the target, leave FDW the server name as `pgbouncer` to use the default configuration. This avoids needing to use the configuration table at all. Set the port(s) to whichever one PgBouncer itself is running on, NOT the postgres database.

    ```
    CREATE SERVER pgbouncer FOREIGN DATA WRAPPER dblink_fdw OPTIONS (host 'localhost',
                                                                     port '6432',
                                                                     dbname 'pgbouncer');
    ```

    b. If more than one PgBouncer needs to be targeted, give each FDW server a unique name and add those names to the `pgbouncer_fdw_targets` configuration table.

    ```
    CREATE SERVER pgbouncer1 FOREIGN DATA WRAPPER dblink_fdw OPTIONS (host '192.168.122.12'
                                                                     port '6432',
                                                                     dbname 'pgbouncer');

    CREATE SERVER pgbouncer2 FOREIGN DATA WRAPPER dblink_fdw OPTIONS (host '192.168.122.13'
                                                                     port '6432',
                                                                     dbname 'pgbouncer');

    INSERT INTO pgbouncer_fdw_targets (target_host) VALUES ('pgbouncer1'),('pgbouncer2');
    ```

    If you do not have an FDW server named `pgbouncer`, be sure to deactivate or remove that default entry in the configuration table. `UPDATE pgbouncer_fdw_targets SET active = false WHERE target_host = 'pgbouncer';`

5. Create user mapping(s) with your preferred credentials in the same database as the FDW(s).

```
CREATE USER MAPPING FOR PUBLIC SERVER pgbouncer OPTIONS (user 'ccp_monitoring', password 'my
```

If you've configured multiple pgbouncer targets, ensure you've also set the user mappings for all PgBouncer targets.

Optionally create a separate user mapping to allow admin command access. The example below sets the `pgb_admin` role that exists in the PostgreSQL database to connect to the PgBouncer admin console as the role `pgb_admin` which should be in the pgbouncer.ini `admin_users` list

```
CREATE USER MAPPING FOR pgb_admin SERVER pgbouncer OPTIONS (user 'pgb_admin', password 'supe
```

6. Grant necessary permissions on extension objects to the user mapping role

```
GRANT USAGE ON FOREIGN SERVER pgbouncer TO ccp_monitoring;

GRANT SELECT ON pgbouncer_clients TO ccp_monitoring;
GRANT SELECT ON pgbouncer_config TO ccp_monitoring;
GRANT SELECT ON pgbouncer_databases TO ccp_monitoring;
GRANT SELECT ON pgbouncer_dns_hosts TO ccp_monitoring;
GRANT SELECT ON pgbouncer_dns_zones TO ccp_monitoring;
GRANT SELECT ON pgbouncer_lists TO ccp_monitoring;
GRANT SELECT ON pgbouncer_pools TO ccp_monitoring;
GRANT SELECT ON pgbouncer_servers TO ccp_monitoring;
GRANT SELECT ON pgbouncer_sockets TO ccp_monitoring;
GRANT SELECT ON pgbouncer_stats TO ccp_monitoring;
GRANT SELECT ON pgbouncer_users TO ccp_monitoring;
```

Please remember that if you are monitoring multiple PgBouncers, you may need to do these grants for additional FDW servers.

For added security, execution on the PgBouncer command functions has been revoked from public by default. You will need to explicitly grant execute privileges on the command functions to your PgBouncer admin role if they are being used.

```
GRANT USAGE ON FOREIGN SERVER pgbouncer TO pgb_admin;

GRANT EXECUTE ON FUNCTION pgbouncer_command_disable(text) TO pgb_admin;
GRANT EXECUTE ON FUNCTION pgbouncer_command_enable(text) TO pgb_admin;
GRANT EXECUTE ON FUNCTION pgbouncer_command_kill(text) TO pgb_admin;
GRANT EXECUTE ON FUNCTION pgbouncer_command_pause(text) TO pgb_admin;
GRANT EXECUTE ON FUNCTION pgbouncer_command_reconnect(text) TO pgb_admin;
GRANT EXECUTE ON FUNCTION pgbouncer_command_reload() TO pgb_admin;
GRANT EXECUTE ON FUNCTION pgbouncer_command_resume(text) TO pgb_admin;
GRANT EXECUTE ON FUNCTION pgbouncer_command_set(text, text) TO pgb_admin;
GRANT EXECUTE ON FUNCTION pgbouncer_command_shutdown() TO pgb_admin;
GRANT EXECUTE ON FUNCTION pgbouncer_command_suspend() TO pgb_admin;
GRANT EXECUTE ON FUNCTION pgbouncer_command_wait_close(text) TO pgb_admin;
GRANT SELECT ON pgbouncer_clients TO pgb_admin;
GRANT SELECT ON pgbouncer_config TO pgb_admin;
GRANT SELECT ON pgbouncer_databases TO pgb_admin;
```

```
GRANT SELECT ON pgbouncer_dns_hosts TO pgb_admin;
GRANT SELECT ON pgbouncer_dns_zones TO pgb_admin;
GRANT SELECT ON pgbouncer_lists TO pgb_admin;
GRANT SELECT ON pgbouncer_pools TO pgb_admin;
GRANT SELECT ON pgbouncer_servers TO pgb_admin;
GRANT SELECT ON pgbouncer_sockets TO pgb_admin;
GRANT SELECT ON pgbouncer_stats TO pgb_admin;
GRANT SELECT ON pgbouncer_users TO pgb_admin;
```

Usage

You should be able to query any of the PgBouncer views provided. For the meaning of the views, see the PgBouncer documentation (linked above). Not all views are provided due to recommendations from upstream author (FDS) or duplication of data already provided by other views (STATS_TOTALS, STATS_AVERAGES, etc).

```
postgres=# select * from pgbouncer_pools;
-[ RECORD 1 ]---------+-----------
pgbouncer_target_host | pgbouncer
database              | pgbouncer
user                  | pgbouncer
cl_active             | 1
cl_waiting            | 0
cl_active_cancel_req  | 0
cl_waiting_cancel_req | 0
sv_active             | 0
sv_active_cancel      | 0
sv_being_canceled     | 0
sv_idle               | 0
sv_used               | 0
sv_tested             | 0
sv_login              | 0
maxwait               | 0
maxwait_us            | 0
pool_mode             | statement
-[ RECORD 2 ]---------+-----------
pgbouncer_target_host | pgbouncer2
database              | pgbouncer
user                  | pgbouncer
cl_active             | 1
cl_waiting            | 0
cl_active_cancel_req  | 0
cl_waiting_cancel_req | 0
sv_active             | 0
sv_active_cancel      | 0
```

```
sv_being_canceled     | 0
sv_idle               | 0
sv_used               | 0
sv_tested             | 0
sv_login              | 0
maxwait               | 0
maxwait_us            | 0
pool_mode             | statement
```

FAQ

Q: If connecting to multiple PgBouncer's, how does ***pgbouncer_fdw*** *handle one or more of the target hosts being down while others are up?*

A: A warning is given for each target host that cannot be connected to. The warning contains the full context of the original error message to help with debugging.

Hosts that are still up should have their metrics returned. Example with `pgbouncer2` target down.

```
postgres=# select * from pgbouncer_fdw_targets ;
 target_host | active
-------------+--------
 pgbouncer   | t
 pgbouncer2  | t
(2 rows)
```

```
postgres=# select * from pgbouncer_clients;
WARNING:  pgbouncer_fdw: Unable to establish connection to PgBouncer target host: pgbouncer2
ORIGINAL ERROR: could not establish connection
CONTEXT: SQL statement "SELECT
        v_row.target_host AS pgbouncer_target_host
        , split_part(substring(version from '\d.+'), '.', 1)::int AS version_major
        , split_part(substring(version from '\d.+'), '.', 2)::int AS version_minor
        , split_part(substring(version from '\d.+'), '.', 3)::int AS version_patch
    FROM dblink(v_row.target_host, 'show version') AS x
    (
        version text
    )"
PL/pgSQL function pgbouncer_version_func(text) line 18 at RETURN QUERY
SQL statement "SELECT version_major, version_minor
                                            FROM public.pgbouncer_version_func(v_row.targe
PL/pgSQL function pgbouncer_clients_func() line 16 at SQL statement
DETAIL: connection to server at "192.168.122.12", port 6432 failed: Connection refused
        Is the server running on that host and accepting TCP/IP connections?
HINT:
-[ RECORD 1 ]---------+--------------------------
```

```
pgbouncer_target_host | pgbouncer
type                  | C
user                  | ccp_monitoring
database              | pgbouncer
state                 | active
addr                  | 192.168.122.16
port                  | 56574
local_addr            | 192.168.122.13
local_port            | 6432
connect_time          | 2023-05-12 17:05:52-04
request_time          | 2023-05-12 17:05:52-04
wait                  | 0
wait_us               | 0
close_needed          | 0
ptr                   | 0x15d8b00
link                  |
remote_pid            | 0
tls                   |
application_name      | app - 192.168.122.16:56574
```

Q: When supporting multiple versions of PgBouncer, how are new/old/renamed columns handled?

A: pgbouncer_fdw will return all columns for all supported versions of PgBouncer. This means that there may be columns being returned that have no data because that version of PgBouncer does not have that column.

If a newer version of PgBouncer drops a column completely, pgbouncer_fdw will support it for a limited time with an empty value and evaluate a time period when support for versions with that old column will be deprecated.

For example, the application_name column will show up if you are running PgBouncer 1.17, but it will have an empty string for a value.

```
postgres=# select * from pgbouncer_clients;
-[ RECORD 1 ]---------+--------------------------
pgbouncer_target_host | pgbouncer
type                  | C
user                  | ccp_monitoring
database              | pgbouncer
state                 | active
addr                  | 192.168.122.16
port                  | 53050
local_addr            | 192.168.122.13
local_port            | 6432
connect_time          | 2023-05-12 15:07:35-04
request_time          | 2023-05-12 15:07:35-04
wait                  | 0
```

```
wait_us              | 0
close_needed         | 0
ptr                  | 0x15d8b00
link                 |
remote_pid           | 0
tls                  |
application_name     |
```

For renamed columns, the newly named column will always be returned and the old column name will not be available no matter the version of PgBouncer you are running. For example, in the `pgbouncer_pools` view for the `SHOW POOLS` command, the old `cl_cancel_req` in v1.17 was renamed to `cl_waiting_cancel_req` in 1.18. This means that for PgBouncer 1.17, you can get the value of `cl_cancel_req` by looking at the value of `cl_waiting_cancel_req`.