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 9.4+ - https://www.postgresql.org
- dblink (contrib module) - https://www.postgresql.org/docs/current/dblink.html
- pgbouncer 1.10+ - https://pgbouncer.github.io

Setup

Whichever database role 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 this role to the `auth_users` file (see NOTE below). Ensure the role used below is able to connect to the special pgbouncer database and run the SHOW commands before setting up the FDW.

If installing from source, run make from the source directory

```
make install
```

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;
```

Create an fdw server & user mapping manually first with your preferred credentials. Leave server name as "pgbouncer". Set the port to whichever port pgbouncer itself is running on, NOT the postgres 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.

NOTE: The database role used for the user mapping must have an explicit entry in the pgbouncer auth_file. The auth_query method in pgbouncer cannot be used to connect to the special `pgbouncer` database where the SHOW commands must be run.

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

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

CREATE EXTENSION pgbouncer_fdw;
```

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;
```

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 either due to recommendations from author (FDS) or duplication of other view data already provided (STATS_TOTALS, STATS_AVERAGES, etc).

```
postgres=> SELECT * FROM pgbouncer_pools;
-[ RECORD 1 ]---------
database   | pgbouncer
user       | pgbouncer
cl_active  | 1
cl_waiting | 0
sv_active  | 0
sv_idle    | 0
sv_used    | 0
sv_tested  | 0
sv_login   | 0
maxwait    | 0
```

```
 maxwait_us | 0
 pool_mode  | statement
-[ RECORD 2 ]---------
 database   | postgres
 user       | postgres
 cl_active  | 1
 cl_waiting | 0
 sv_active  | 1
 sv_idle    | 0
 sv_used    | 0
 sv_tested  | 0
 sv_login   | 0
 maxwait    | 0
 maxwait_us | 0
 pool_mode  | session
```