How to Set Up PgBouncer Connection Pooling for Postgres Plus Standard Server(R)

A Postgres Evaluation Quick Tutorial From EnterpriseDB

January 22, 2010

EnterpriseDB Corporation, 235 Littleton Road, Westford, MA 01866, USA
T +1 978 589 5700 **F** +1 978 589 5701 **E** info@enterprisedb.com **www**.enterprisedb.com

Introduction

PgBouncer is a lightweight connection pooler for Postgres Plus that dramatically reduces the processing time and resources for maintaining a large number of client connections to one or more databases. PgBouncer is typically used to increase the number of user connections that can be handled in a high performance environment. This tutorial shows how to effectively setup and configure a pgBouncer environment for Postgres Plus Standard Server .

This EnterpriseDB Quick Tutorial helps you get started with the Postgres Plus Standard Server database product in a Linux or Windows environment. It is assumed that you have already downloaded and installed Postgres Plus Standard Server on your desktop or laptop computer.

This Quick Tutorial is designed to help you expedite your Technical Evaluation of Postgres Plus Standard Server. For more informational assets on conducting your evaluation of Postgres Plus, visit the self-service web site, Postgres Plus Open Source Adoption.

In this Quick Tutorial you will learn how to do the following:

- install and setup PgBouncer
- key concepts of the client connection process
- how to configure PgBouncer for Linux or Windows
- set global configuration parameters
- set up authentication through PgBouncer
- use the admin console to monitor connection pools

Feature Description

PgBouncer is a pre-bundled enterprise module installed by default with Postgres Plus Standard Server.

PgBouncer is a lightweight connection pooler for Postgres Plus. Connection pooling dramatically reduces the processing time and resources for maintaining a large number of client connections to one or more databases. PgBouncer is typically used to dramatically increase the number of user connections that can be handled in a high performance environment.

PgBouncer Concepts

PgBouncer reduces the impact of opening new client connections to Postgres Plus

databases by maintaining and using a cache of database connections called a *connection pool*. An application connects to PgBouncer as if it were a Postgres Plus database. PgBouncer then creates a connection to the actual database server, or it reuses one of the existing connections from the pool.

A connection pool is established for a unique combination of the PgBouncer database alias name (typically equates to a Postgres Plus database name) and the database server user name connecting to the database. The database server user name used to connect to the database may be either the user name supplied by the client application, or it may be a user name configured with PgBouncer that overrides the client supplied user name.

The client connection process occurs in the following sequence:

Step 1: The client application attempts to connect to PgBouncer using the same database connection interface that it uses to connect to any Postgres Plus database. However the client application supplies the IP address of the host running PgBouncer and the port number on which PgBouncer is listening for connections (default port number is `6432`) instead of the respective values for the Postgres Plus database server.

Step 2: The database name supplied by the client application during the connection attempt must match one of a list of PgBouncer database alias names that are maintained in a plain-text configuration file accessed by PgBouncer. If there is no match, the client connection is rejected with an error message.

Step 3: The user name and password supplied by the client application during the connection attempt must match one of a list of user name and password pairs that are maintained in a plain-text authentication file accessed by PgBouncer. If there is no match, the client connection is rejected with an error message.

Step 4: If there is already an existing connection pool for the combination of database alias name passed by the client application and the database server user name to be used for the connection, then an available connection from this pool is assigned to the client application. The client can now access the database.

If there is no available connection in the pool, then one is created (see Step 6) provided the pool's connection limit has not been reached. If the pool has reached its connection limit, the client must wait until a connection in the pool becomes available.

Step 5: If there is no existing pool, then one is created for the combination of database alias name and database server user name.

Step 6: For each database alias name in the PgBouncer configuration file, there is a connection string containing parameter/value pairs PgBouncer uses these pairs to connect to a Postgres Plus database. If certain parameters are omitted from the connection string, the values supplied by the client application are used instead. The database server user name and password, and the database name, itself, are examples of connection

parameters that may come from the client application if they are not supplied in the connection string.

PgBouncer attempts to establish a connection to the database using the parameters in the connection string, possibly supplemented by values passed by the client application. The usual Postgres Plus connection authentication process occurs. Checks are made using the `pg_hba.conf` file and the database server's user names and passwords. If Postgres Plus authentication fails, then the client connection is rejected with an error message. If authentication succeeds, then a database connection is established, becomes part of the pool, and is assigned to the client application. The client can now access the database.

Step 7: Once PgBouncer determines a client is finished with a connection, the connection is returned to the pool and becomes available for use by other clients without the additional overhead of establishing a new connection.

Note: Once a pool of database server connections is established, a change to the `pg_hba.conf` file or the database server's user names and passwords may not prevent a client from connecting using what may now be an invalid user name and password combination. If the authentication described in steps 2 and 3 succeed, and the condition described in Step 4 is true, PgBouncer connects the client using an existing connection from the pool. Stop and restart PgBouncer to remove all connection pools so new clients are forced to go through the Postgres Plus authentication process as well as the PgBouncer authentication steps.

PgBouncer supports three types of pooling when rotating connections:

- **Session pooling.** A server connection is assigned to the client application for the life of the client connection. PgBouncer releases the server connection back into the pool once the client application disconnects. This is the default method.
- **Transaction pooling.** A server connection is assigned to the client application for the duration of a transaction. When PgBouncer detects the completion of the transaction, it releases the server connection back into the pool.
- **Statement pooling.** A server connection is assigned to the client application for each statement. When the statement completes, the server connection is returned back into the pool. Multi-statement transactions are not permitted for this mode.

Administration of PgBouncer is done through the PgBouncer Admin Console. The Admin Console is accessed through the `psql` utility program by connecting to PgBouncer with a special "virtual" database named `pgbouncer`. Once logged into the Admin Console, use the `SHOW` command to display information and statistics on connection pool usage.

The PgBouncer Index page is an index to the complete PgBouncer documentation.

Additional information about PgBouncer and the PgBouncer project can be found on the

Postgres Community Projects page of the EnterpriseDB web site.

Tutorial Steps

Configuring PgBouncer

Step 1: Verify that PgBouncer is installed on the database server.

You should see a subdirectory named `pgbouncer` under the Postgres Plus home directory.

Note: When installing Standard Server, if you de-selected the PgBouncer component, the subdirectory `pgbouncer` and its contents are not installed. If you did not install PgBouncer, you can use StackBuilder Plus to add PgBouncer to your Standard Server configuration.

Once PgBouncer is installed, default configuration files are created and PgBouncer is set up to automatically start whenever you start your computer. PgBouncer runs as a daemon on Linux and as a service on Microsoft Windows® systems. The daemon or service is named `pgbouncer`.

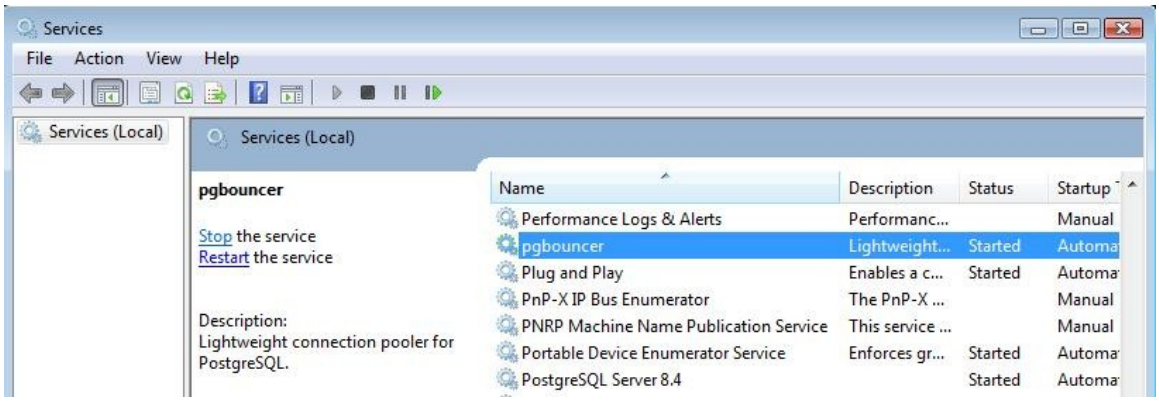
On Linux hosts you can verify the PgBouncer daemon is running by using the following command:

```
ps aux | grep pgbouncer
```

This is shown by the following:

```
$ ps aux | grep pgbouncer
postgres  3162  0.0  0.0   2472   856 ?        S    12:45   0:00 ./pgbouncer
-d /opt/PostgresPlus/8.4SS/pgbouncer/share/pgbouncer.ini
postgres 22540  0.0  0.0   2456   668 ?        S    12:58   0:00
/opt/PostgresPlus/8.4SS/pgbouncer/bin/pgbouncer-d
/opt/PostgresPlus/8.4SS/pgbouncer/share/pgbouncer.ini
```

For Windows hosts open Control Panel, Administrative Tools, and then Services. The `pgbouncer` service is one of the running services:



The following table lists the names and directory locations of certain PgBouncer files that will be used in these instructions:

File Name	Location	Description
`pgbouncer`	`pgbouncer/bin`	Program executable file
`pgbouncer.ini`	`pgbouncer/share`	Configuration file
`userlist.txt`	`pgbouncer/etc`	Authentication file
`pgbouncer.log` (Linux)	`/var/log/pgbouncer`	Log file
`pgbouncer.log` (Windows)	`pgbouncer\log`	Log file

Note: If PgBouncer is installed during Postgres Plus product installation, the `pgbouncer` subdirectory is located under the Postgres Plus home directory. If PgBouncer is installed using StackBuilder Plus, you choose the location to install the `pgbouncer` subdirectory.

Step 2: Add additional databases to the PgBouncer configuration file

If desired, add additional database alias names and their connection strings to the PgBouncer configuration file, `pgbouncer.ini`, located in the `pgbouncer/share` subdirectory. When PgBouncer is installed, a database connection is pre-configured for the `postgres` database.

If you wish to add additional database connections, add an entry with the following syntax in the section of the configuration file labeled `[databases]`.

```
database_alias_name = connection_string
```

When a client application makes a connection, it specifies the database alias name. PgBouncer uses the connection string associated with the alias name to make the database server connection.

The connection string contains space-delimited, `libpq` style *`parameter=value`* pairs. These parameters include the following:

- **dbname.** Name of the database to which the connection is to be made. If omitted, defaults to the database with the same name as *database_alias_name*.
- **host.** IP address of the database server. This parameter is mandatory for Windows hosts. If omitted on Linux hosts, defaults to the host running PgBouncer. PgBouncer uses the Unix domain socket to make the connection if the `host` parameter is omitted in which case the `pg_hba.conf` file of the database server must have a `local` connection type. (This entry already exists in default Postgres Plus installations.)
- **port.** Port number on which the database server is listening for connections. If omitted, defaults to 5432.
- **user.** Database server user name to be used to establish the database connection. If specified, then all database connections made as a result of references to this alias name use the user name given by the `user` parameter. This results in one pool used by all such clients connecting with this alias name. If the `user` parameter is omitted, the user name given by the client to connect to PgBouncer is also used to make the database connection. In this case, a separate pool is established per user name.
- **password.** Password for the database user name. If the `user` parameter is specified, the `password` parameter should also be specified. If the `user` parameter is specified, but the `password` parameter is omitted, no password is supplied to the database server when the connection attempt is made.

Note: The host-based authentication file, `pg_hba.conf`, must have the appropriate entries to allow connections using the connection strings in the configuration file. The default location of the `pg_hba.conf` file is in the `data` subdirectory of the Postgres Plus home directory.

The following example shows the connection strings for three aliases:

```
[databases]
postgres = host=127.0.0.1 port=5432
example = host=127.0.0.1 port=5432 dbname=sample
remote = host=192.168.10.101 port=5432 user=postgres password=my_password
```

In the entry with alias name `postgres`, a server connection is made to the `postgres` database in a database server running on the same host as PgBouncer. The same user name and password given by the client application to connect to PgBouncer are used to connect to the database.

In the entry with alias name `example`, a server connection is made to a database named `sample` in a database server running on the same host as PgBouncer. The same user name and password given by the client application to connect to PgBouncer are used to connect to the database.

In the entry with alias name `remote`, a server connection is made to a database named `remote` in a database server running on the host with IP address `192.168.10.101`. The

user name `postgres` with password `my_password` are used to connect to the database.

Step 3: If desired, make adjustments to the global configuration of PgBouncer.

The section of the configuration file labeled `[pgbouncer]` contains global configuration information. When PgBouncer is installed, this portion of the configuration file is created with ready-to-use settings.

The following lists some of the parameters that can be adjusted:

- **auth_type.** Method used by PgBouncer to authenticate client connections to PgBouncer. Values may be `md5`, `crypt`, `plain`, `trust`, or `any`. The default setting is `md5`.
- **auth_file.** Directory path to the authentication file containing user names and passwords that clients must use to connect to PgBouncer.
- **listen_addr.** IP address on which PgBouncer listens for client connections. If omitted, only Unix socket connections are allowed. (That is, the client must be on the same host as PgBouncer and must not supply a host IP address when it connects to PgBouncer.) PgBouncer is installed with a setting of "`*`", which means listen on all addresses.
- **listen_port.** Port on which PgBouncer listens for client connections. The default setting is `6432`.
- **logfile.** Directory path to the PgBouncer log file.
- **pidfile.** Directory path to the process ID file.
- **admin_users.** Comma-separated list of users allowed access to the Admin Console who can then perform connection pool management operations and obtain information about the connection pools. PgBouncer is installed with `postgres` as an admin user.
- **pool_mode.** Specifies when the server connection can be released back into the pool. Values may be `session`, `transaction`, or `statement`. The default is `session`.

The following example shows usage of these parameter settings in the configuration file:

```
[pgbouncer]
logfile = /var/log/pgbouncer/pgbouncer.log
pidfile = /var/run/pgbouncer/pgbouncer.pid
listen_addr = *
listen_port = 6432
auth_type = md5
auth_file = /opt/PostgresPlus/8.4SS/pgbouncer/etc/userlist.txt
admin_users = postgres
pool_mode = session
```

Combined with the databases section from Step 2, a complete, simple PgBouncer configuration file appears as follows:

```
[databases]
postgres = host=127.0.0.1 port=5432
```



```
example = host=127.0.0.1 port=5432 dbname=sample
remote = host=192.168.10.101 port=5432 user=postgres password=my_password

[pgbouncer]
logfile = /var/log/pgbouncer/pgbouncer.log
pidfile = /var/run/pgbouncer/pgbouncer.pid
listen_addr = *
listen_port = 6432
auth_type = md5
auth_file = /opt/PostgresPlus/8.4SS/pgbouncer/etc/userlist.txt
admin_users = postgres
pool_mode = session
```

Step 4: If desired, add additional users to the PgBouncer authentication file.

The authentication file contains pairs of double-quote enclosed user names and passwords that a client application uses to access PgBouncer. The location of the authentication file is given by the `auth_file` configuration parameter.

When PgBouncer is installed, an authentication file named `userlist.txt` is created in the `pgbouncer/etc` subdirectory with the following content:

```
"postgres" "my_password"
```

The database superuser password you supplied when you installed Postgres Plus is used as the value of *`my_password`*.

Note: The user names and passwords in the authentication file may be different from the database server user names and passwords if the connection strings in the PgBouncer configuration file include the `user` and `password` parameters. If the `user` and `password` parameters are omitted in a given connection string, then the user names and passwords in the authentication file must have matching user names and passwords in the database server in order to connect using the database alias associated with this connection string.

Step 5 (Optional): Encrypt the passwords in the authentication file.

If you do not want to use clear text passwords in the authentication file, you can replace them with MD5-encyrpted passwords. Use the `md5` function in the `psql` utility program to generate MD5-encrypted passwords. In Postgres Plus, an MD5-encrypted password consists of the string `md5` concatenated with the string returned by applying the MD5 algorithm to the concatenation of the clear text form of the password and the user name.

For example, if the clear text password of user `postgres` is `my_password`, then the `md5` function is applied to the string `my_passwordpostgres`.

Generation of the MD5-encyrpted password is shown by the following:

```
postgres=# select 'md5' || md5 ('my_password' || 'postgres');
             ?column?
```



```
-------------------------------------
 md5bd865b5255aa3a11062eb552bc89fc9c
(1 row)
```

Copy the resulting string into the `userlist.txt` file:

```
"postgres" "md5bd865b5255aa3a11062eb552bc89fc9c"
```

If you use MD5-encrypted passwords in the authentication file, be sure the `auth_type` parameter in the configuration file is set to `md5`.

Step 6: If a change to the PgBouncer configuration file was made, restart PgBouncer.

For Linux only: Log in to the computer as user `postgres`. If the `pgbouncer` daemon is currently running, find the process ID and issue the `kill` command on the process as follows:

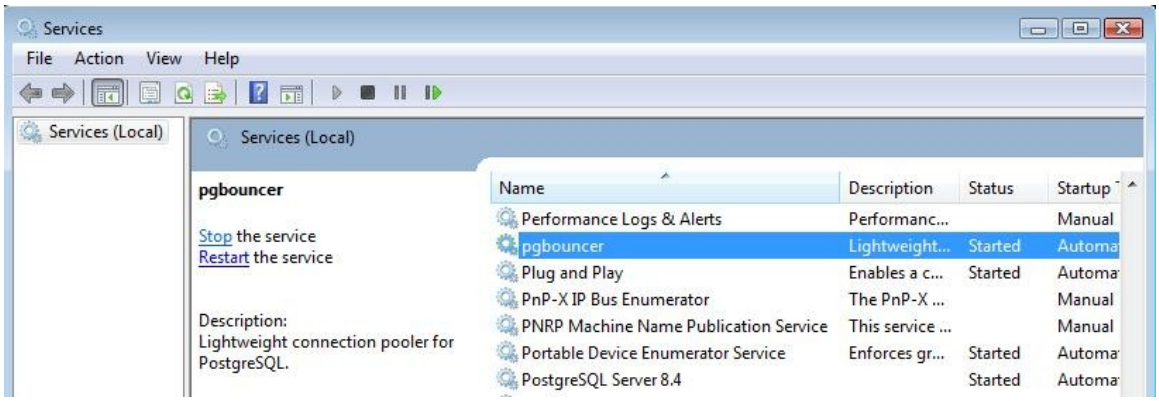
```
$ ps aux | grep pgbouncer
postgres  8185  0.0  0.0   2472   680 ?        S    16:58   0:00 ./pgbouncer
-d ../share/demobouncer.ini
postgres  8289  0.0  0.0   3064   732 pts/4    S+   17:01   0:00 grep
pgbouncer
$ kill 8185
```

Change to the `pgbouncer/bin` subdirectory and start the `pgbouncer` daemon as follows:

```
$ cd /opt/PostgresPlus/8.4SS/pgbouncer/bin
$ ./pgbouncer -d /opt/PostgresPlus/8.4SS/pgbouncer/share/pgbouncer.ini
2010-01-20 14:47:47.350 24109 LOG File descriptor limit: 1024 (H:8192),
max_client_conn: 100, max fds possible: 170
```

The `-d` option runs `pgbouncer` in the background. The second parameter is the PgBouncer configuration file named `pgbouncer.ini` located in the `pgbouncer/share` subdirectory of the Postgres Plus home directory.

For Windows only: Restart the `pgbouncer` service by opening Control Panel, Administrative Tools, and then Services. Select the `pgbouncer` service and click the Restart link.



Step 7: Connect client applications to PgBouncer.

For a client application to utilize connection pooling, it must log in to PgBouncer with the following attributes:

- For a database name, the client must supply a database alias name listed in the databases section of the PgBouncer configuration file.
- For a user name and password, the client must supply a user name and password listed in the PgBouncer authentication file.
- The client must supply the port number that is given by the `listen_port` parameter in the pgbouncer section of the PgBouncer configuration file.
- If PgBouncer is not running on the same host as the client application, the client must supply the PgBouncer host IP address.

Using the `psql` utility program as your client application, you can utilize connection pooling by logging in to `psql` with the parameters shown by the following examples, assuming you use the configuration file illustrated in Step 3:

```
$ /opt/PostgresPlus/8.4SS/bin/psql -d postgres -U postgres -p 6432
Password for user postgres:
psql (8.4.1)
Type "help" for help.

postgres=#
```

The following example shows a connection to the `example` database alias:

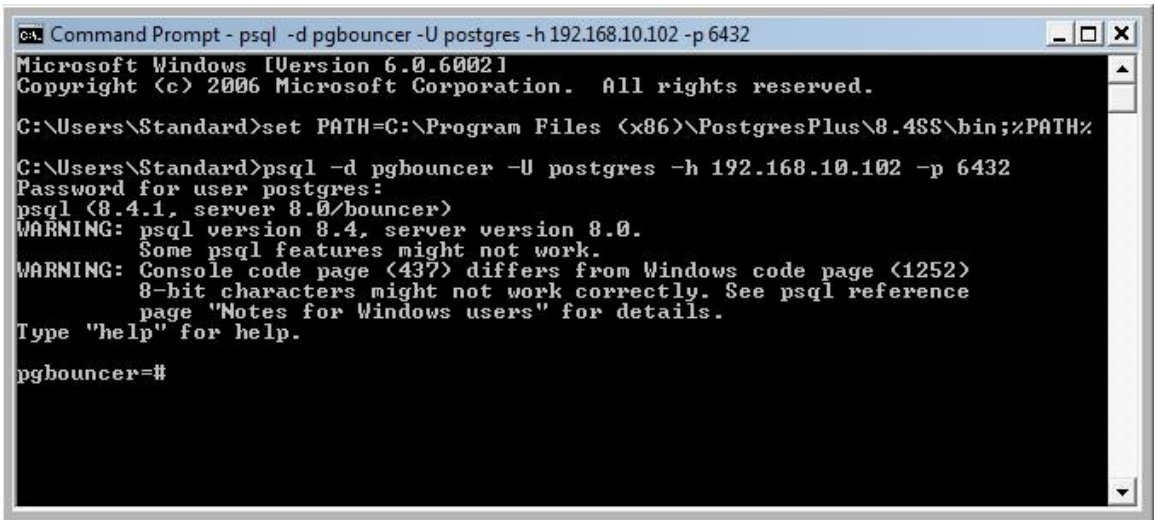
```
$ /opt/PostgresPlus/8.4SS/bin/psql -d example -U postgres -p 6432
Password for user postgres:
psql (8.4.1)
Type "help" for help.

example=#
```

The following example shows a connection to the `remote` database alias:

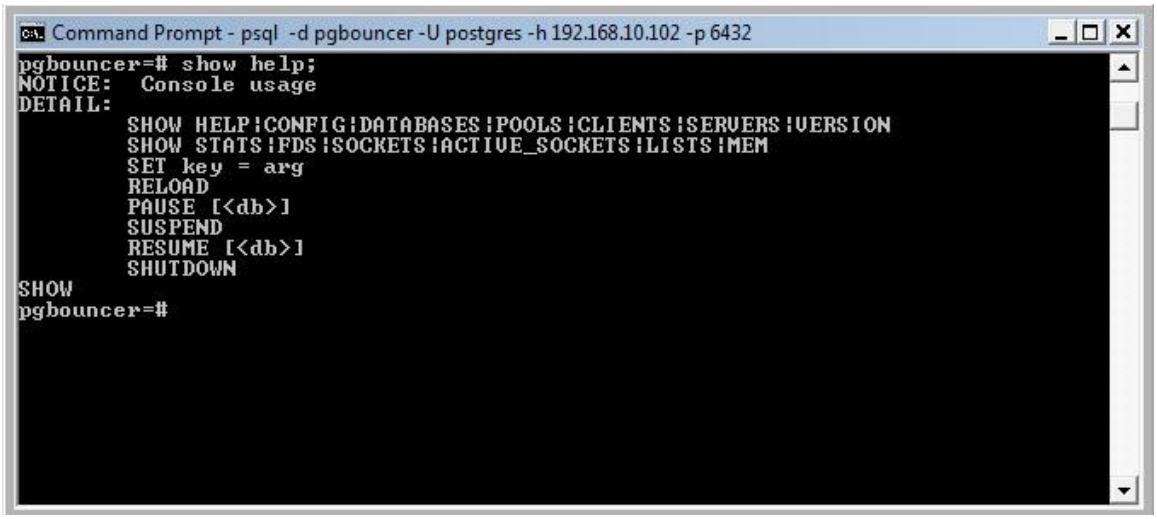

```
$ /opt/PostgresPlus/8.4SS/bin/psql -d remote -U postgres -p 6432
Password for user postgres:
psql (8.4.1)
Type "help" for help.

remote=#
```

Step 8: Use the Admin Console to observe the state of the connection pools.

Log in to the `psql` program and connect to PgBouncer using database `pgbouncer`, user name `postgres`, and port number `6432`. If PgBouncer is running on a different host than the `psql` program you are using, also specify the IP address of the host running PgBouncer as shown by the following:

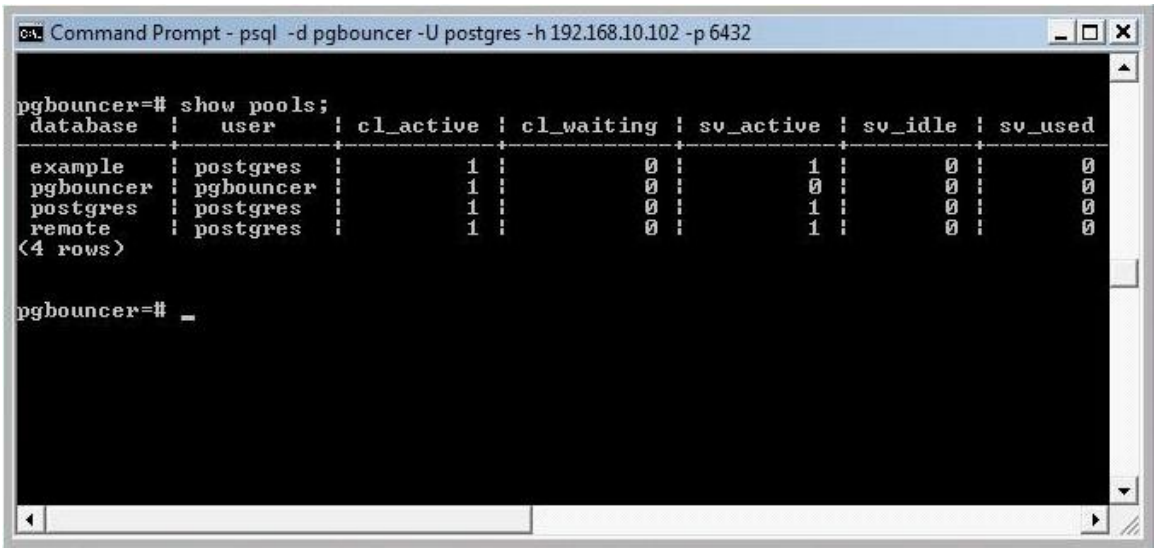


The `SHOW HELP` command lists the commands available in the Admin Console:

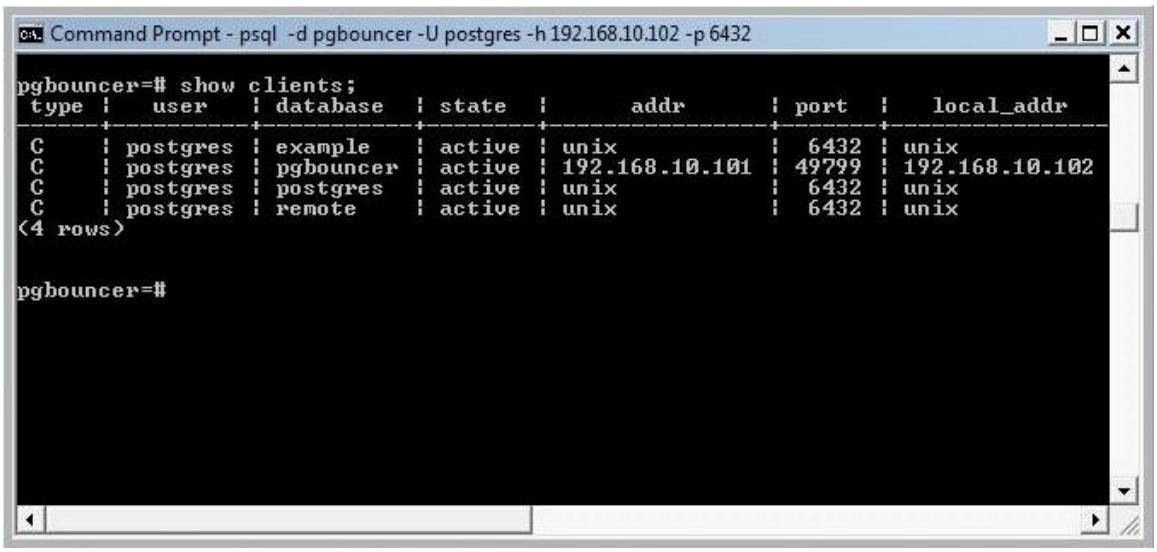


The following are examples of the output of some of the Admin Console commands.

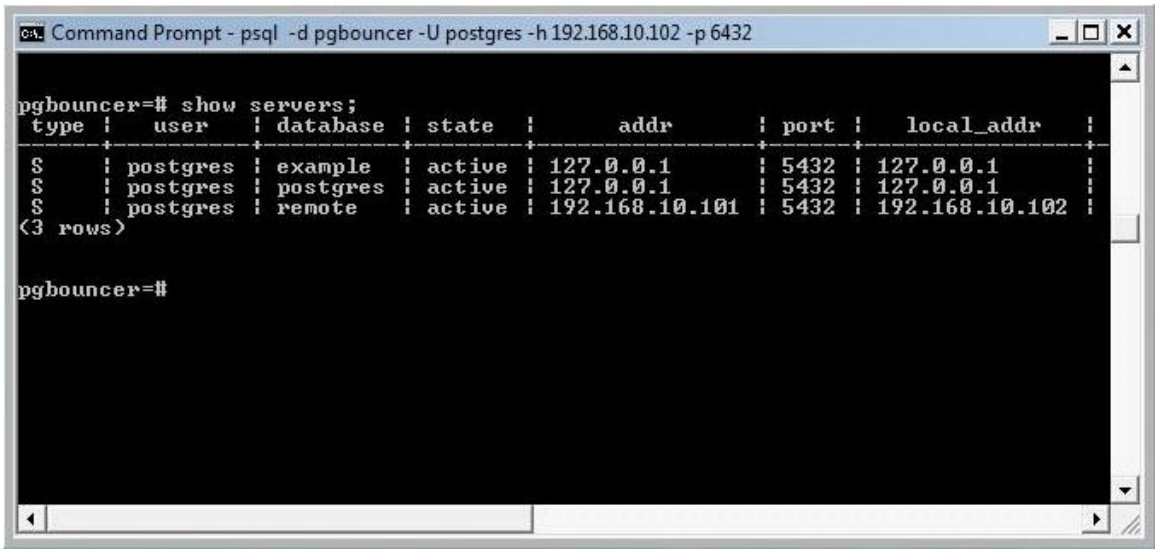
The `SHOW POOLS` command lists the connection pools:



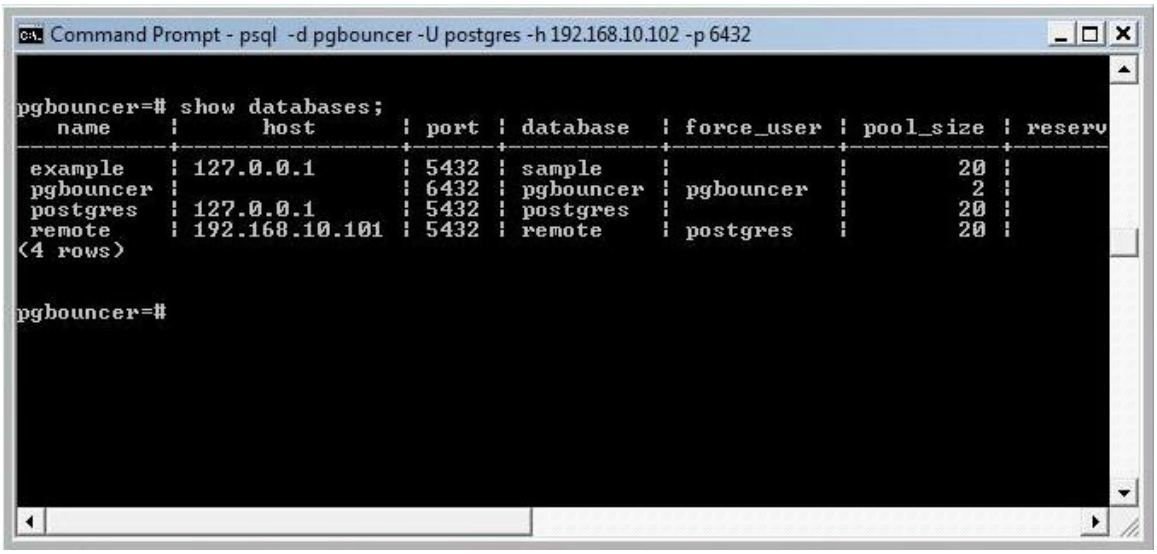
The `SHOW CLIENTS` command lists the clients and the state of their connections to PgBouncer database aliases:



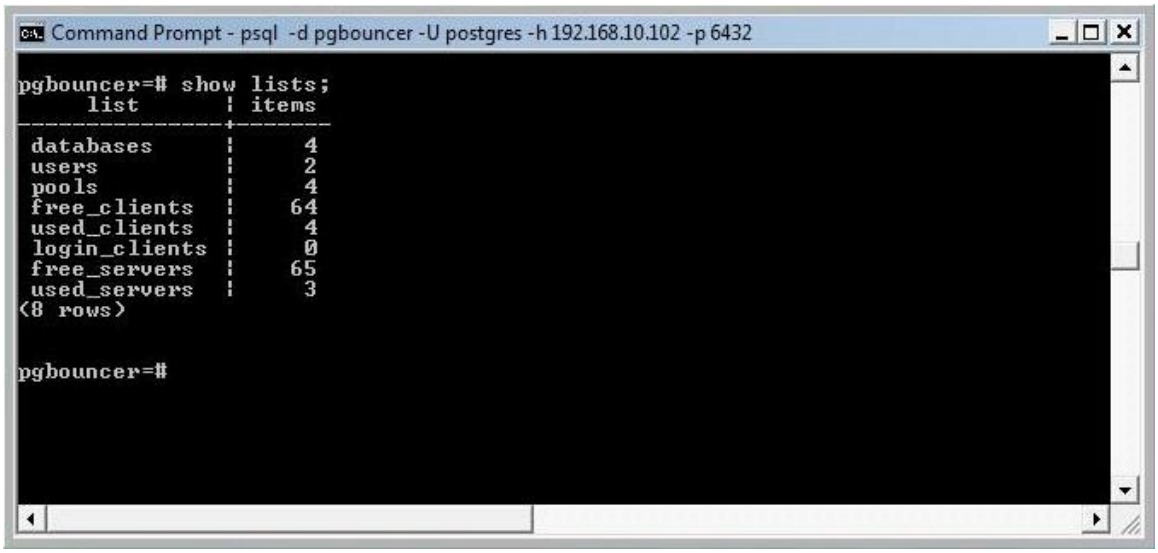
The `SHOW SERVERS` command lists the PgBouncer connections to the database servers:



The `SHOW DATABASES` command lists the PgBouncer database aliases and their associated Postgres Plus databases:



The `SHOW LISTS` command displays count information for various connection pooling components:



Conclusion

In this Quick Tutorial you learned how to set up PgBouncer on a Postgres Plus database.

You should now be able to proceed confidently with a Technical Evaluation of Postgres

Plus knowing that your Postgres Plus database can efficiently handle a high volume of client connections.

The following resources should help you move on with this step:

- Postgres Plus Technical Evaluation Guide
- Postgres Plus Getting Started resources
- Postgres Plus Quick Tutorials
- Postgres Plus User Forums
- Postgres Plus Documentation
- Postgres Plus Webinars

