
# PgBouncer Deep Dive

Karen Jex | Senior Solutions Architect | Crunchy Data

June 2023

# Agenda

- What is PgBouncer?
- Connection Pooling Overview
- Getting Started with PgBouncer
- PgBouncer in Crunchy Data Products
- More PgBouncer Options

# What is PgBouncer?

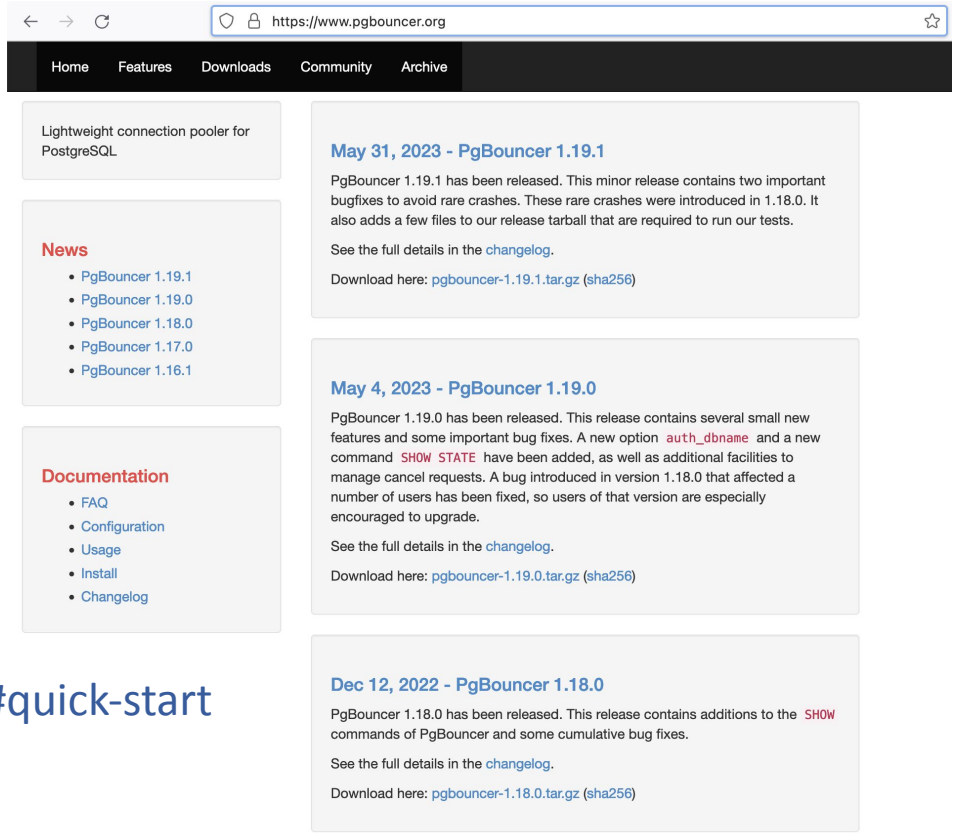
- What does PgBouncer do?
- Why should I use it?

# What is PgBouncer: What does PgBouncer do?

"lightweight, easy to configure connection pooler"

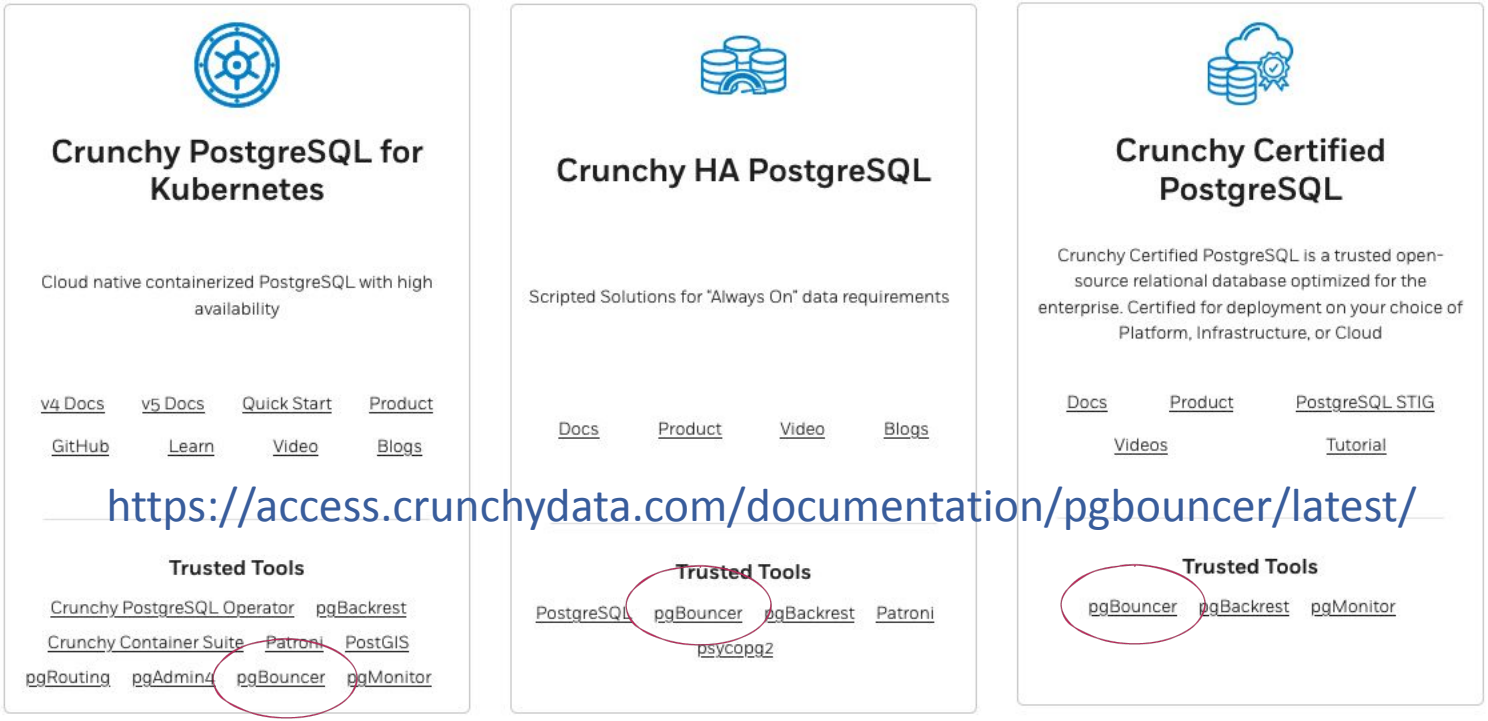
Image par Christine Sponchia de Pixabay

# What is PgBouncer: PgBouncer Docs




https://www.pgbouncer.org/usage.html#quick-start

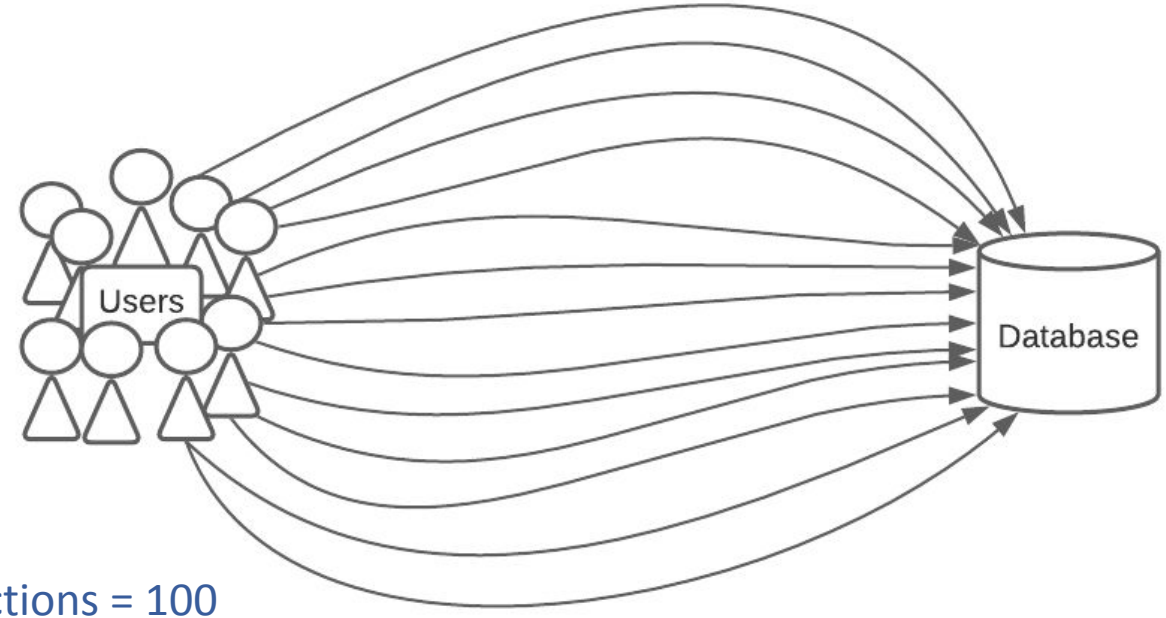
# What is PgBouncer: Why should I use PgBouncer?




https://access.crunchydata.com/documentation/pgbouncer/latest/

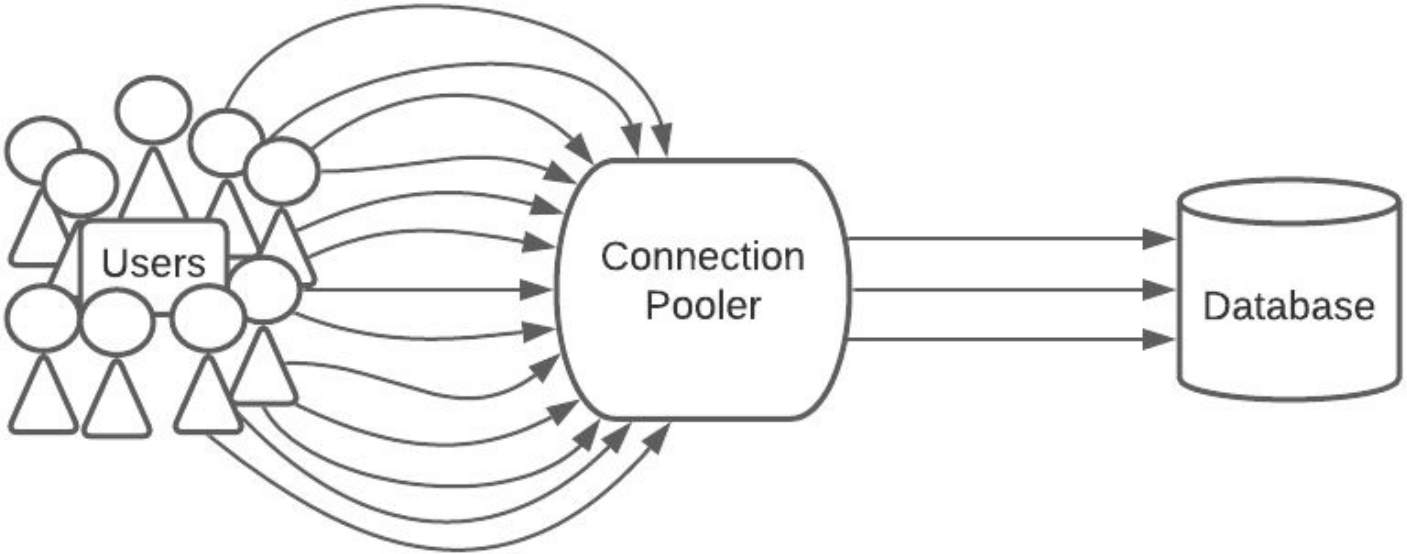
# Connection Pooling Overview

# Connection Pooling Overview




max_connections = 100

# Connection Pooling Overview



## Connection Pooling Overview:

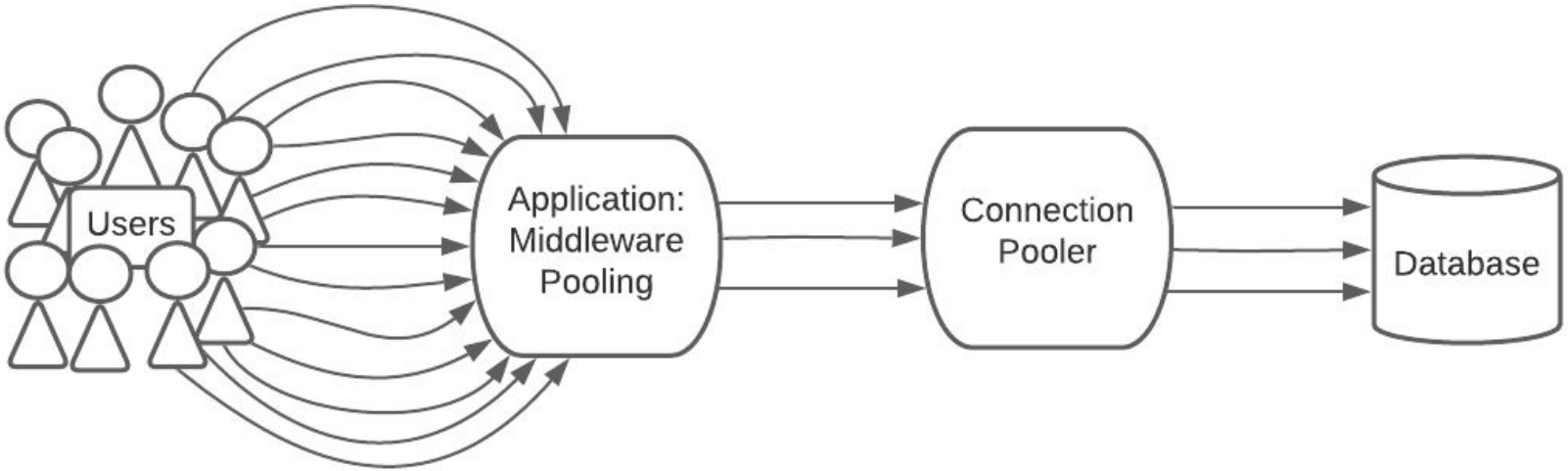
# When not to use Connection Pooling

Consider potential disadvantages and whether or not it is required

- Additional latency
- Additional complexity
- Additional resources
- Security implications

## Connection Pooling Overview:

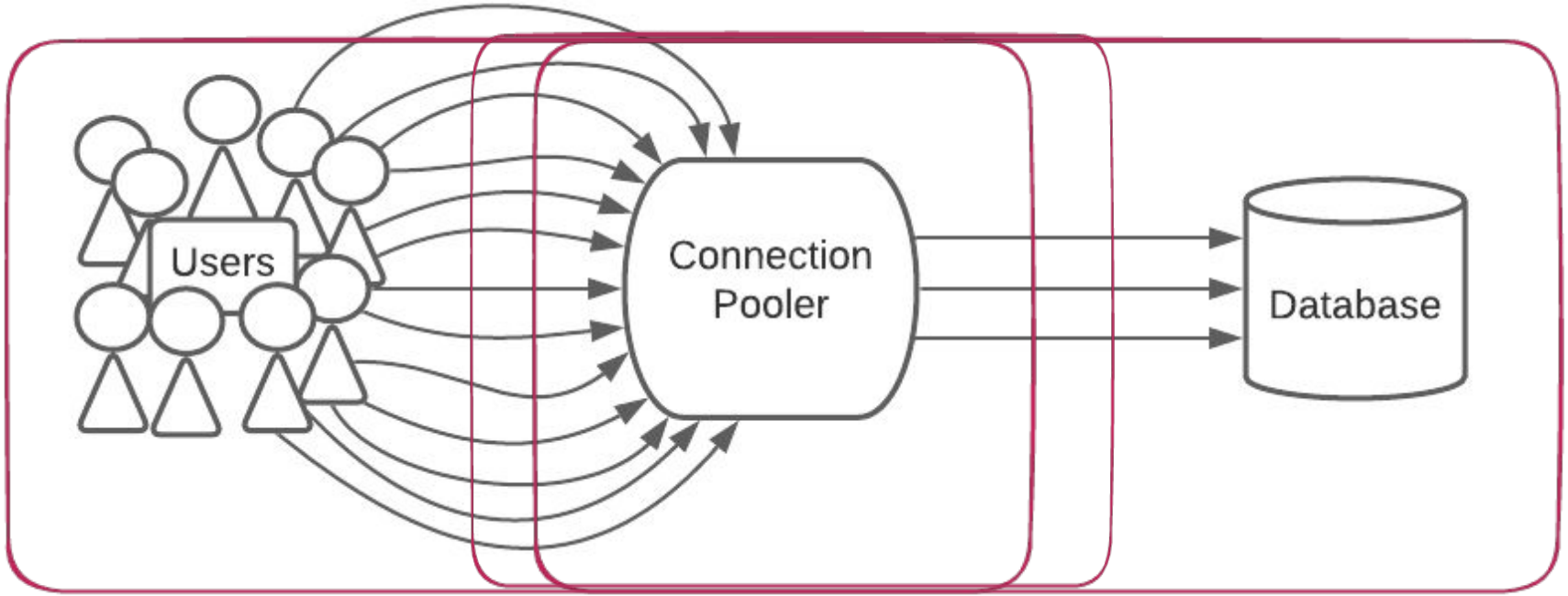
# When not to use Connection Pooling



# Getting Started with PgBouncer

- Installing
- Configuring
- Monitoring

# Getting Started with PgBouncer: Installing PgBouncer




http://www.pgbouncer.org/faq.html#should-pgbouncer-be-installed-on-the-web-server-or-database-server

# Getting Started with PgBouncer: Installing PgBouncer

https://www.pgbouncer.org/downloads/

RPM: yum.postgresql.org

Deb: apt.postgresql.org

https://github.com/pgbouncer/pgbouncer.git

# Getting Started with PgBouncer: Installing PgBouncer

```
[karen_jex@rocky9 ~]$ sudo dnf -y install pgbouncer
...
================================================================================
Package         Architecture   Version              Repository          Size
================================================================================
Installing:
pgbouncer        x86_64        1.19.1-42.rhel9      pgdg-common         215 k
Installing dependencies:
 python3-psycopg2 x86_64        2.9.5-1.rhel9        pgdg-common         188 k
...
Installed:
  pgbouncer-1.19.1-42.rhel9.x86_64        python3-psycopg2-2.9.5-1.rhel9.x86_64

Complete!
```

# Getting Started with PgBouncer: Configuring PgBouncer

```
$ cat /etc/pgbouncer/pgbouncer.ini
;;;
;;; PgBouncer configuration file
;;;

;; database name = connect string
;;
;; connect string params:
;;   dbname= host= port= user= password= auth_user= client_encoding= datestyle= timezone=
;;   pool_size= reserve_pool= max_db_connections= pool_mode= connect_query= application_name=
...
;; Read additional config from other file
;%include /etc/pgbouncer/pgbouncer-other.ini
```

# Getting Started with PgBouncer: Configuring PgBouncer

## databases section

```
[databases]

charleston = host=localhost dbname=charleston

;charleston = dbname=charleston host=192.168.152.128 port=5433

;ch = dbname=charleston host=localhost
```

# Getting Started with PgBouncer: Configuring PgBouncer

## Authentication settings

```
;;;
;;; Authentication settings
;;;

;; any, trust, plain, md5, cert, hba, pam
auth_type = scram-sha-256

;auth_hba_file =
```

# Getting Started with PgBouncer: Configuring PgBouncer

## Authentication settings

**PgBouncer** handles its own client authentication and **has its own database of users**.

```
auth_file = /etc/pgbouncer/userlist.txt
```

```
$ echo '"karen" "mysupersecretpassword"' > /etc/pgbouncer/userlist.txt"
```

# Getting Started with PgBouncer: Configuring PgBouncer

## Authentication settings

```
[databases]

charleston = host=localhost dbname=Charleston auth_user=karen auth_dbname=authdb

[pgbouncer]

;; Authentication database that can be set globally to run "auth_query".
auth_dbname = authdb

;; Query to use to fetch username and password hash from database.
auth_query = SELECT usename, passwd FROM pg_shadow WHERE usename=$1
```

# Getting Started with PgBouncer: Configuring PgBouncer

## Admin User

```
;;;
;;; Users allowed into database 'pgbouncer'
;;;

;; comma-separated list of users who are allowed to change settings
admin_users = postgres
admin_users = karen
```

# Getting Started with PgBouncer: Configuring PgBouncer

## Type of Pooling

When can a server connection be returned to the pool for reuse?

```
pool_mode = session | transaction | statement
```

# Getting Started with PgBouncer: Configuring PgBouncer

## pool_mode = session (default)

The connection is released back to the pool after the client disconnects

# Getting Started with PgBouncer: Configuring PgBouncer

## pool_mode = transaction

The connection is released back to the pool after the transaction finishes

# Getting Started with PgBouncer: Configuring PgBouncer

## pool_mode = statement

The connection is released back to the pool after the query finishes

Transactions spanning multiple statements not allowed

Tends to break applications, especially if they've not been designed with this in mind

# Getting Started with PgBouncer: Configuring PgBouncer

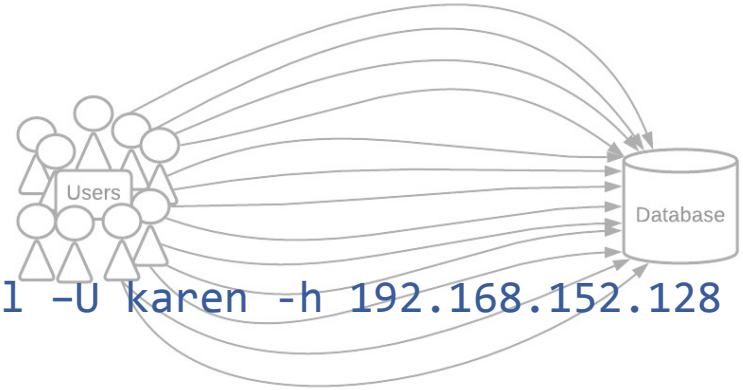
## Pool Sizing

| Parameter | Description | Default value |
|---|---|---|
| max_client_conn | Max number of client connections allowed | 100 |
| default_pool_size | Number of server connections per user/database pair | 20 |
| min_pool_size | Add server connections to pool if below this number | 0 (disabled) |
| reserve_pool_size | How many additional connections to allow to a pool | 0 (disabled) |
| reserve_pool_timeout | Use connection from the reserve pool if client waits this long (s) | 5.0 |
| max_db_connections | Max number of connections allowed per database | 0 (unlimited) |
| max_user_connections | Max number of connections allowed per user | 0 (unlimited) |

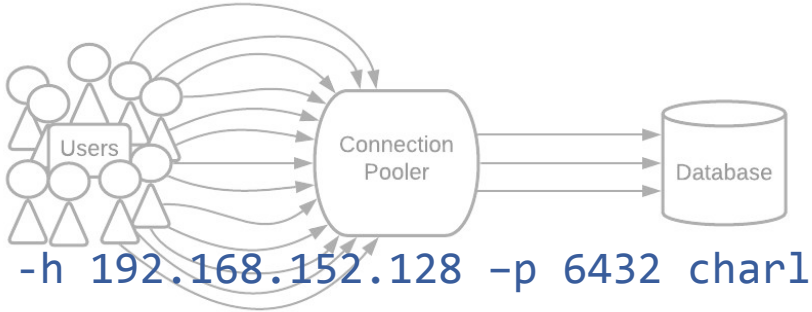
# Getting Started with PgBouncer: Starting PgBouncer

```
$ sudo systemctl start pgbouncer

or

$ pgbouncer -d /etc/pgbouncer/pgbouncer.ini
```

# Getting Started with PgBouncer: Connecting via PgBouncer




```
psql -U karen -h 192.168.152.128 -p 5432 charleston
```




```
psql -U karen -h 192.168.152.128 -p 6432 charleston
```

# Getting Started with PgBouncer: Connecting via PgBouncer

```
$ psql -U [username] -h [pgbouncer_host] -p [pgbouncer_port] [pgbouncer_db]

$ psql –U karen -h 192.168.152.128 –p 6432 charleston
```

PgBouncer Console

```
$ psql -U postgres -h 127.0.0.1 -p 6432 pgbouncer
```

# Getting Started with PgBouncer: Monitoring

```
pgbouncer=# SHOW HELP;

SHOW HELP|CONFIG|DATABASES|POOLS|CLIENTS|SERVERS|USERS|VERSION
SHOW PEERS|PEER_POOLS
SHOW FDS|SOCKETS|ACTIVE_SOCKETS|LISTS|MEM|STATE
SHOW DNS_HOSTS|DNS_ZONES
SHOW STATS|STATS_TOTALS|STATS_AVERAGES|TOTALS

...
```

# Getting Started with PgBouncer: Monitoring

```
pgbouncer=# SHOW clients;

type | user   |  database   | state   | port |       connect_time        | application_name

-----+-------+------------+--------+------+---------------------------+-----------------

C    | karen | charleston | active | 6432 | 2023-06-12 15:15:05 CEST | psql
C    | karen | pgbouncer  | active | 6432 | 2023-06-12 15:14:06 CEST | psql
```

# Getting Started with PgBouncer: pgbouncer_fdw

“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.”

https://www.crunchydata.com/blog/making-pgbouncer-easier-to-monitor

https://github.com/CrunchyData/pgbouncer_fdw

# Getting Started with PgBouncer: Admin

```
pgbouncer=# SHOW HELP;
...
RELOAD
PAUSE [<db>]
RESUME [<db>]
DISABLE <db>
ENABLE <db>
RECONNECT [<db>]
KILL <db>
SUSPEND
SHUTDOWN
WAIT_CLOSE [<db>]
```

# PgBouncer in Crunchy Data Products

- **Crunchy Postgres**

  full service scripted solution for deploying Production PostgreSQL anywhere you want to run it

- **Crunchy Postgres for Kubernetes**

  Cloud Native Postgres on Kubernetes
  powered by Crunchy Postgres Operator (PGO)

- **Crunchy Bridge**

  Fully managed Postgres on your choice of Cloud provider

# PgBouncer in Crunchy Data Products: Crunchy Postgres

Implemented via the pgbouncer role (extracts from docs)

### collect_pgbouncer_metrics

| | |
|---|---|
| **VARIABLE NAME:** | collect_pgbouncer_metrics |
| **DESCRIPTION:** | Whether to collect PgBouncer metrics for pgMonitor |
| **DEFAULT VALUE:** | `false` |
| **TYPE:** | boolean |
| **REQUIRED:** | false |
| **EXAMPLE VALUE(S):** | N/A |
| **NOTES:** | When **`true`**, the **`pgbouncer`** role will assume that the 'dblink' and 'pgbouncer_fdw' extensions are installed in the target database (the **`pgmonitor`** role will do this if **`collect_pgbouncer_metrics: true`** is set there) and create the Foreign Data Wrapper server and user mappings for the **`pgmonitor_db_user`** (usually **`ccp_monitoring`**). This presents a set of views in the PostgreSQL database that pgMonitor will use to collect metric data for monitoring PgBouncer itself. |

### databases

| | |
|---|---|
| **VARIABLE NAME:** | databases |
| **DESCRIPTION:** | Configured user databases for PgBouncer |
| **DEFAULT VALUE:** | `[]` |
| **TYPE:** | list of dictionaries |
| **REQUIRED:** | false |
| **EXAMPLE VALUE(S):** | ▸ Complex value; click to see in full |

## PgBouncer in Crunchy Data Products:

# Crunchy Postgres for Kubernetes

https://access.crunchydata.com/documentation/postgres-operator/latest/quickstart/#connect-to-the-postgres-cluster

https://access.crunchydata.com/documentation/postgres-operator/latest/tutorial/connection-pooling/

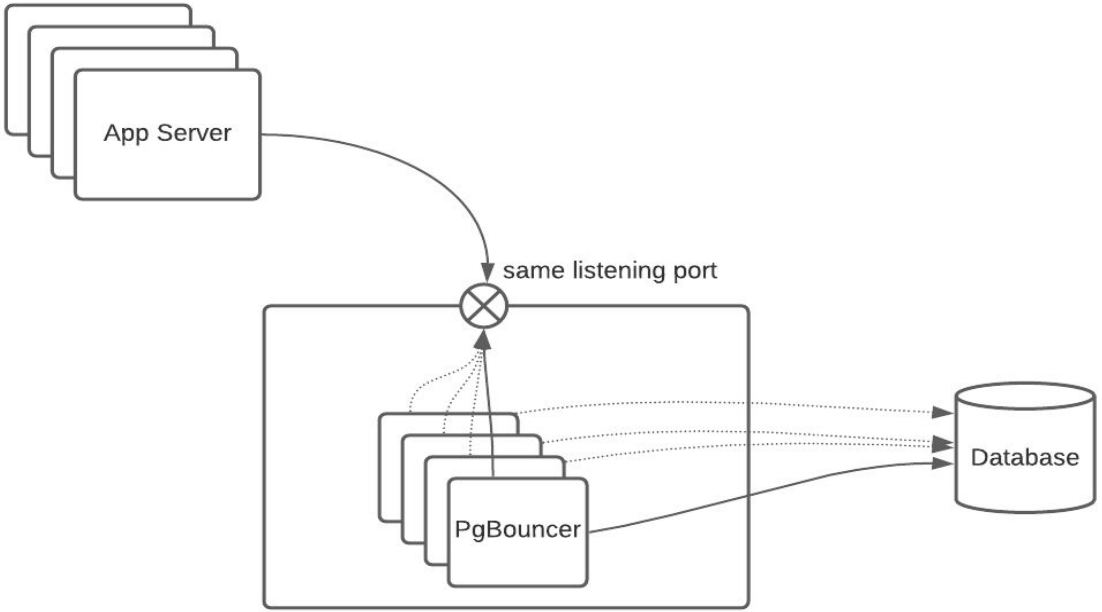
```
spec:
  proxy:
    pgBouncer:
      config:
        global:
          pool_mode: transaction
```

# More PgBouncer Options

- Multiple Instances
- Read-Write / Read Only Routing
- Load Balancing
- Fault Tolerance
- Taming ”Badly Behaved” Applications

# More PgBouncer Options: Multiple Instances

## SO_REUSEPORT



# More PgBouncer Options: Multiple Instances

- Create multiple PgBouncer instances
- Use SO_REUSEPORT so multiple PgBouncer instances listen on the same port
- Use more than one thread
- easy scaling!
- so_reuseport info: https://www.pgbouncer.org/config.html#low-level-network-settings

# More PgBouncer Options: Multiple Instances

```
[pgbouncer]
...
so_reuseport = 1
```

```
$ cp /etc/pgbouncer.service /etc/systemd/system/pgbouncer@.service

$ daemon reload

$ systemctl start pgbouncer@1
$ systemctl start pgbouncer@2
$ etc
```

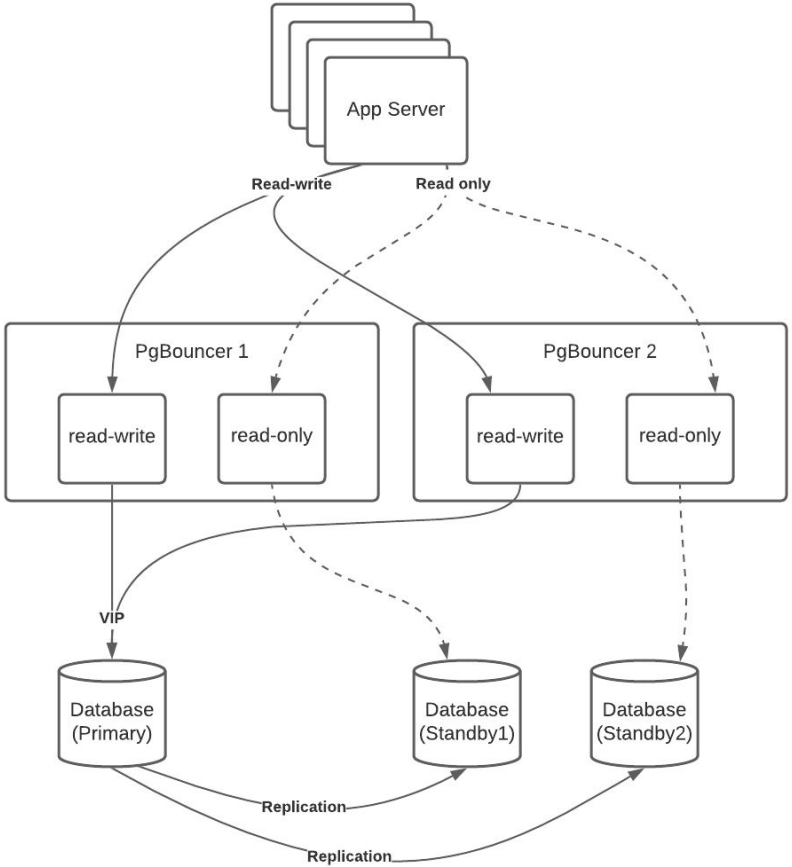
# More PgBouncer Options: Read-Write / Read Only Routing

Use different pgBouncer databases to:

- Route read-write traffic to the current primary database
- Route read-only traffic to a standby database

```
[pgbouncer]
...
charleston_rw = dbname=charleston host=<virtual_IP>
charleston_ro = dbname=charleston host=<standby_ip>
```

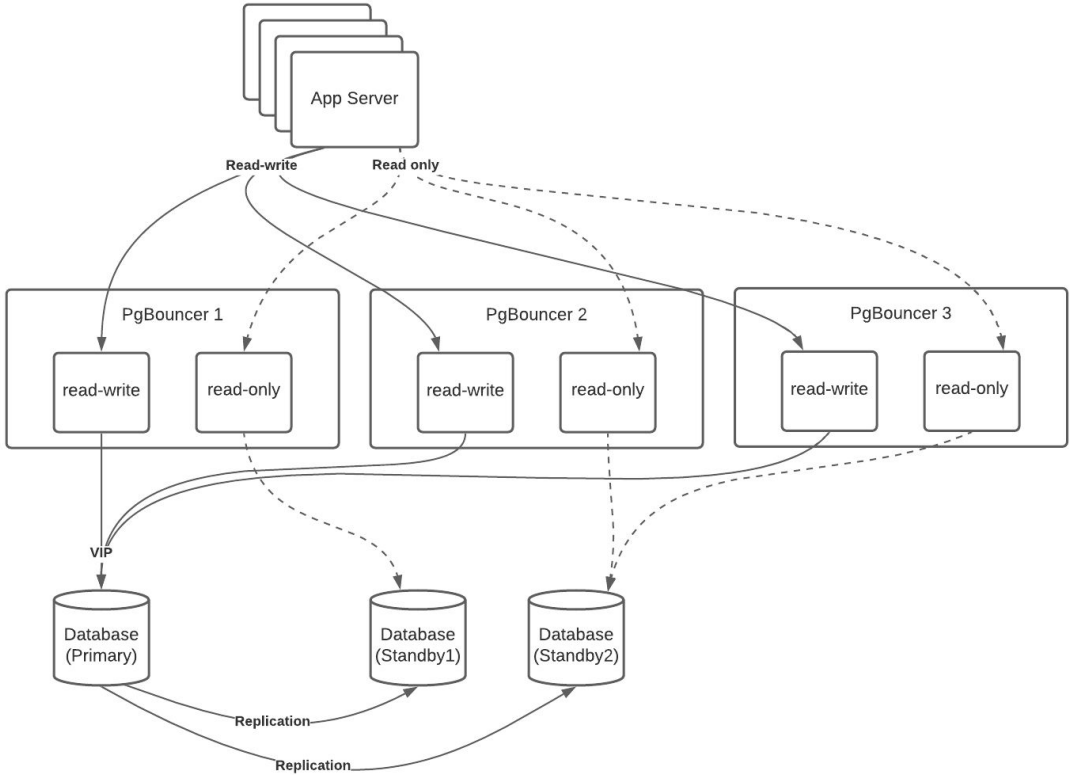
# More PgBouncer Options: Read-Write / Read Only Routing



# More PgBouncer Options: Read-Write / Read Only Routing

```
pgbouncer1.ini

    [databases]

    read_write = dbname=charleston host=<Virtual IP>
    read_only = dbname=charleston host=192.168.152.129

pgbouncer2.ini

    [databases]

    read_write = dbname=charleston host=<Virtual IP>
    read_only = dbname=charleston host=192.168.152.130
```

# More PgBouncer Options: Load Balancing



# More PgBouncer Options: Fault Tolerance

When using SO_REUSEPORT

- Connections to databases created on a round-robin basis
- Database connections are persistent

If PgBouncer is on a **database server**

If DB server/PgBouncer instance is lost:

- Connection between application and database terminated
- Round-robin begins again

If PgBouncer on an **external** machine

- if the database goes down:
  - connection to PgBouncer remains
  - PgBouncer attempts to connect to another endpoint
  - the application connection is not disrupted

# More PgBouncer Options: Taming “Badly Behaved” Applications

- Database traffic from certain application may be unpredictable
- Use PgBouncer to tame “badly behaved” applications by using distinct port numbers
- Provide database connections to deal with sudden bursts of incoming traffic in different ways
- prevent the database from becoming swamped during high activity periods

# More PgBouncer Options: Taming “Badly Behaved” Applications

```
%include /etc/pgbouncer/pgbouncer.ini

[databases]

toomuch = dbname=pgbench host=127.0.0.1

[users]

bounce = pool_mode=transaction max_user_connections=200

[pgbouncer]

listen_port = 6433

max_db_connections = 10
```

# More PgBouncer Options: Taming “Badly Behaved” Applications

```
$ /usr/bin/pgbench -h localhost -p 6433 -U bounce -C -c 200 -T 600 toomuch

connection to database "toomuch" failed:
ERROR:  no more connections allowed (max_client_conn)
client 100 aborted while establishing connection
connection to database "toomuch" failed:
ERROR:  no more connections allowed (max_client_conn)
client 100 aborted while establishing connection
connection to database "toomuch" failed:
ERROR:  no more connections allowed (max_client_conn)
client 100 aborted while establishing connection
```

crunchy data

# That’s All Folks!