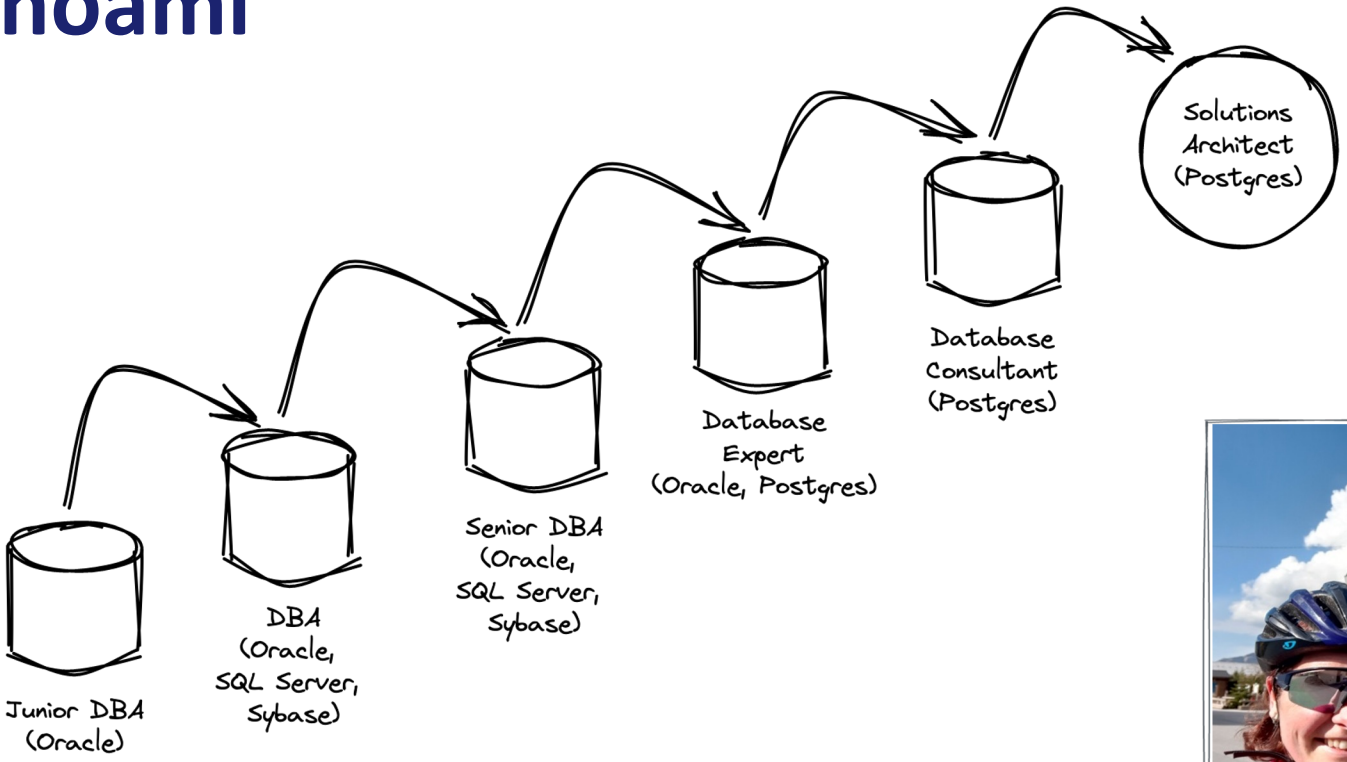
How to Keep your Database Happy

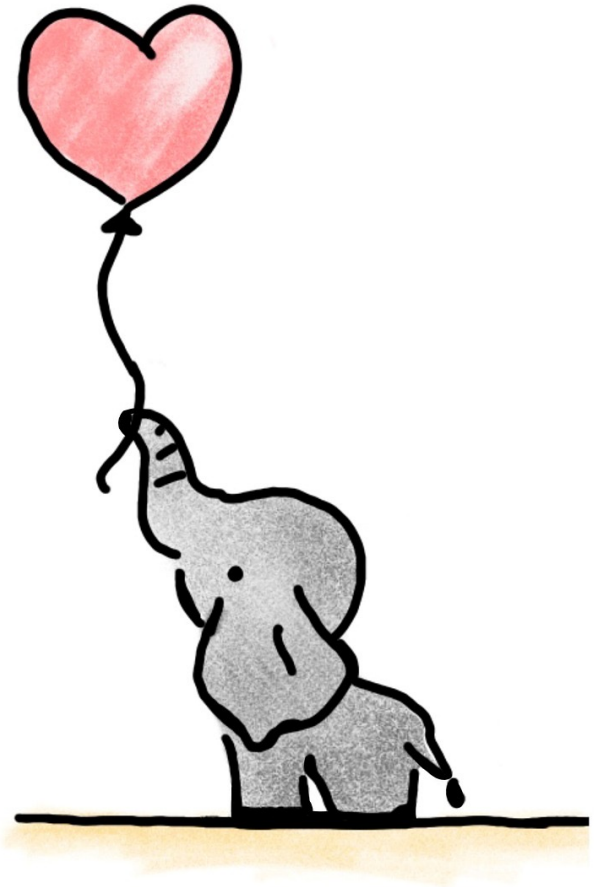
Karen Jex | Senior Solutions Architect @ Crunchy Data

PyCon UK | Cardiff | Sept 2023

whoami


Junior DBA
(Oracle)
DBA
(Oracle,
SQL Server,
Sybase)
Senior DBA
(Oracle,
SQL Server,
Sybase)
Database
Expert
(Oracle, Postgres)
Database
Consultant
(Postgres)
Solutions
Architect
(Postgres)

Introduction



Agenda

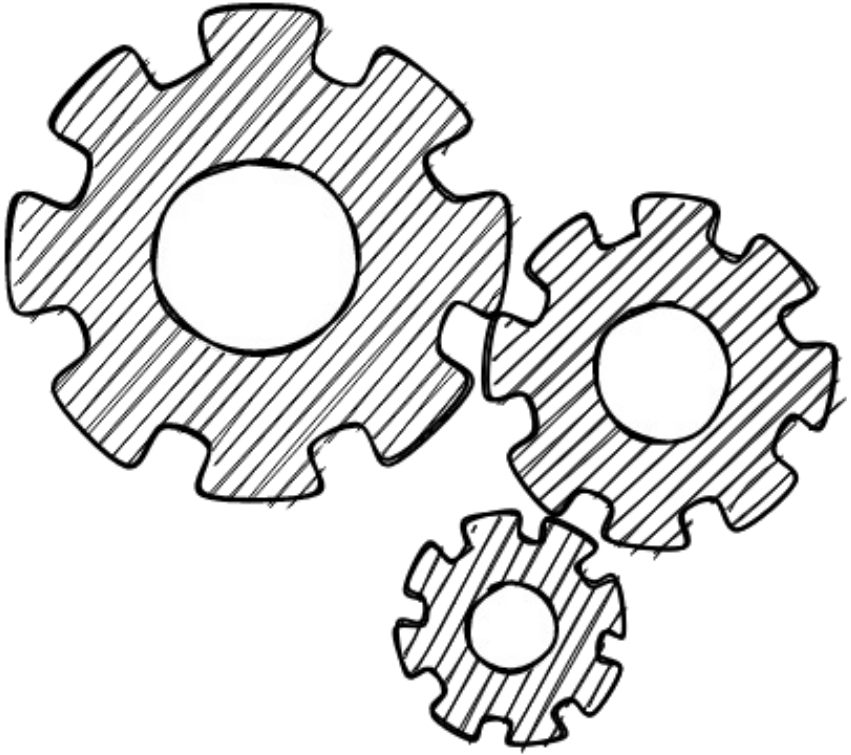
1. Set **Memory & Performance Parameters**
2. Schedule (and test) **Backups**
3. Implement **High Availability**
4. Configure **Connections**
5. Put in place **Logging, Monitoring & Alerting**

Agenda

1. **Set Memory & Performance Parameters**
2. Schedule (and test) Backups
3. Implement High Availability
4. Configure Connections
5. Put in place Logging, Monitoring & Alerting

Memory & Performance Settings

- Small footprint by default
- May not be right for prod
- Over 350 parameters
  - Memory allocation
  - WAL and Checkpoints



Memory Allocation

https://www.postgresql.org/docs/current/runtime-config-resource.html

```
postgres=# alter system set shared_buffers =
```

~ 25-40% of RAM

Memory dedicated to Postgres to use for caching data

Memory Allocation

https://www.postgresql.org/docs/current/runtime-config-resource.html

```
postgres=# alter system set shared_buffers =
```

~ 25-40% of RAM

```
postgres=# alter system set work_mem =
```

~ 10MB
or increase for specific sessions

max memory used by a query operation before spilling to disk

use “log_temp_files” to see if temp files are created

Memory Allocation

https://www.postgresql.org/docs/current/runtime-config-resource.html

```
postgres=# alter system set shared_buffers =
```

~ 25-40% of RAM

```
postgres=# alter system set work_mem =
```

~ 10MB
or increase for specific sessions

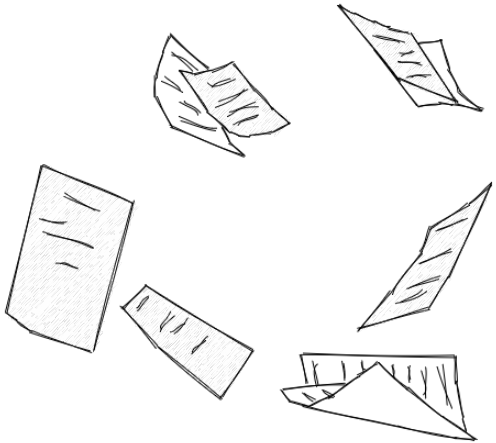
```
postgres=# alter system set maintenance_work_mem =
```

~ 5% of RAM

Memory that can be used by maintenance operations

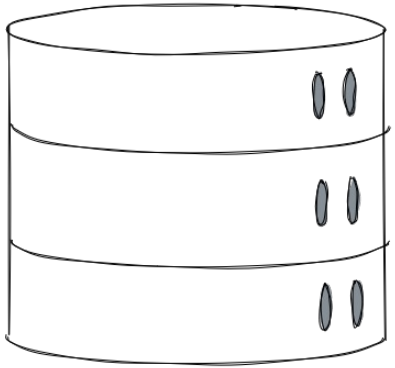
WAL/Checkpoint Tuning

https://www.postgresql.org/docs/current/runtime-config-wal.html



Checkpoint:

Dirty data pages flushed to disk



Checkpoints are expensive!

WAL/Checkpoint Tuning

https://www.postgresql.org/docs/current/runtime-config-wal.html

```
postgres=# alter system set wal_buffers = 32MB
```

~3% shared_buffers, up to 16MB

memory available for WAL before it's synced to disk

WAL/Checkpoint Tuning

https://www.postgresql.org/docs/current/runtime-config-wal.html

```
postgres=# alter system set wal_buffers = 32MB

postgres=# alter system set checkpoint_timeout = 10 - 30 mins
```

A checkpoint will be triggered if
one hasn't taken place within checkpoint_timeout

WAL/Checkpoint Tuning

https://www.postgresql.org/docs/current/runtime-config-wal.html

```
postgres=# alter system set wal_buffers = 32MB

postgres=# alter system set checkpoint_timeout = 10 - 30 mins

postgres=# alter system set max_wal_size = 1/2 to 2/3 available space for WAL
```

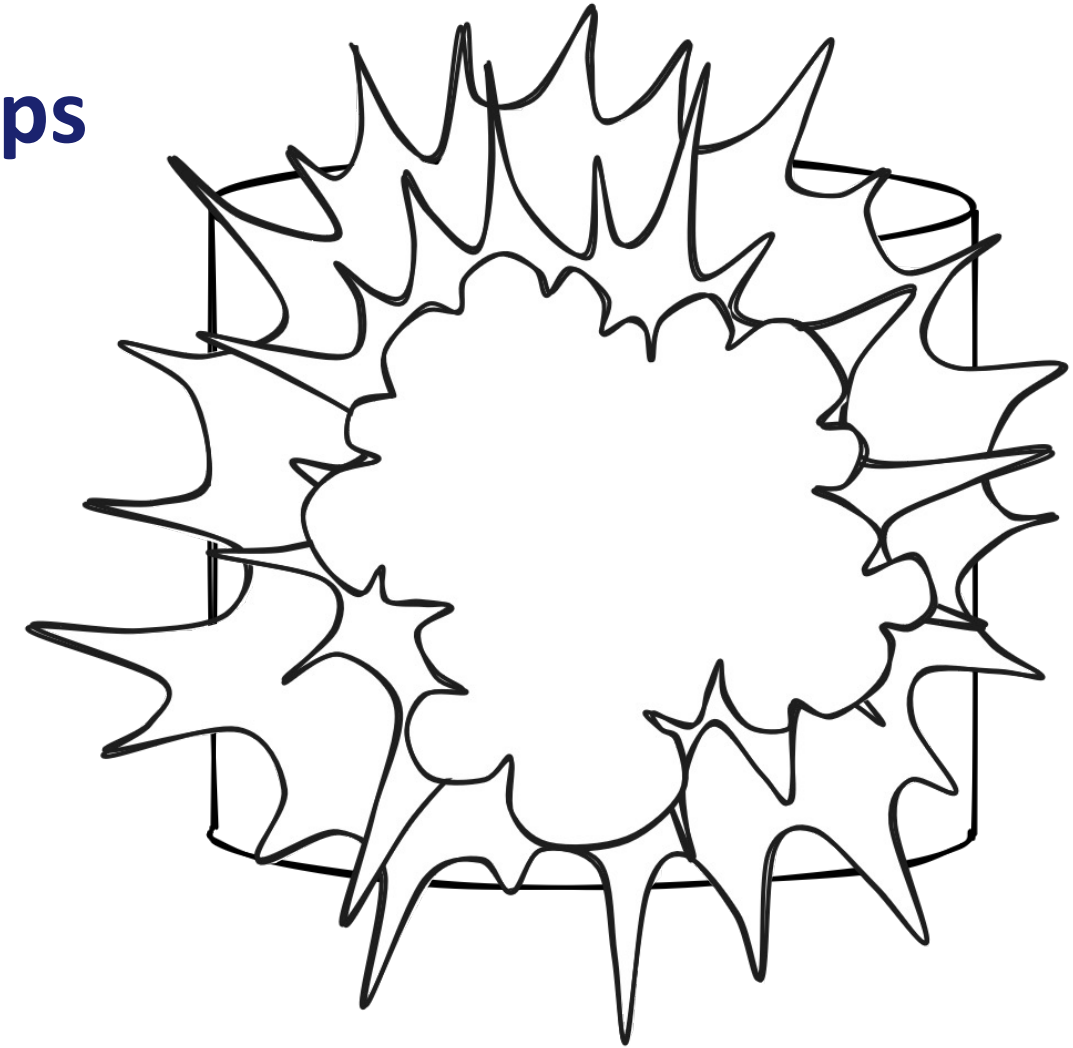
A checkpoint will be triggered if this amount of WAL is generated

Increase if checkpoints frequently triggered by max_wal_size

Agenda

1. Set Memory & Performance Parameters
2. **Schedule (and test) Backups**
3. Implement High Availability
4. Configure Connections
5. Put in place Logging, Monitoring & Alerting

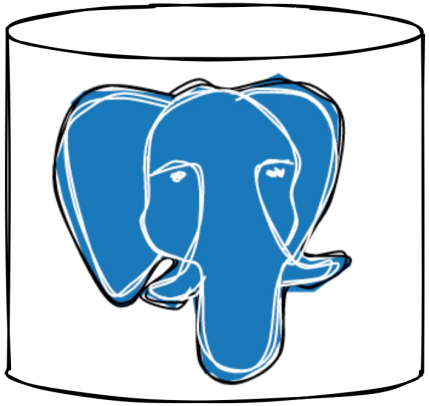
Backups

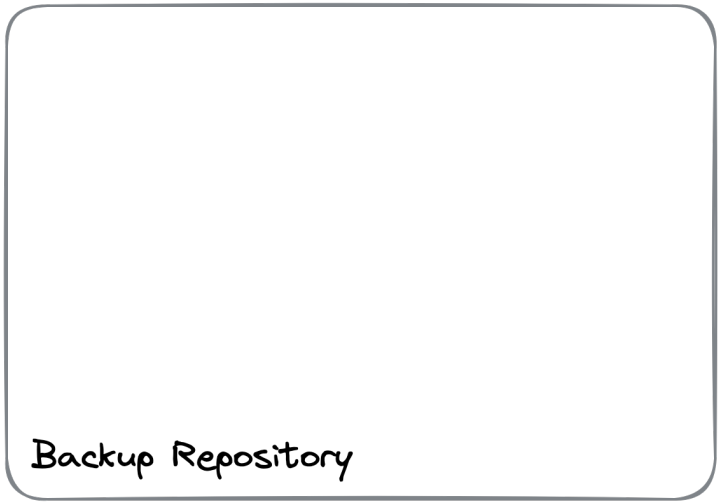


Backups

Area of disk:

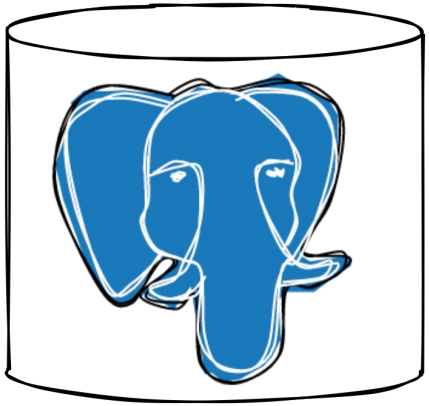
- on DB server
- on another server
- Network storage
- Cloud storage

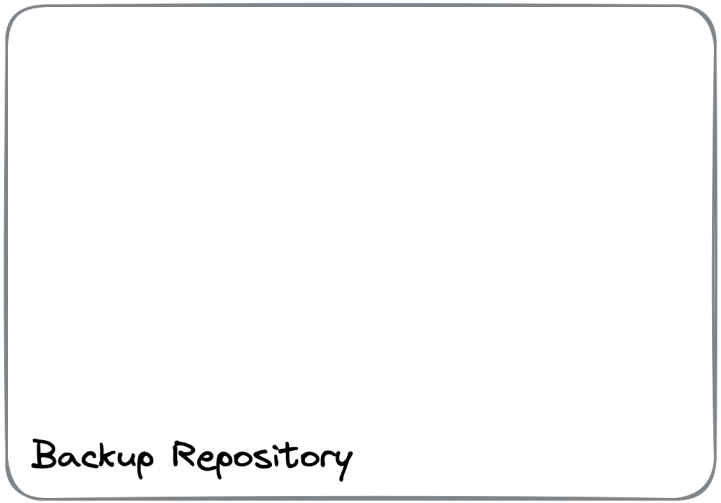




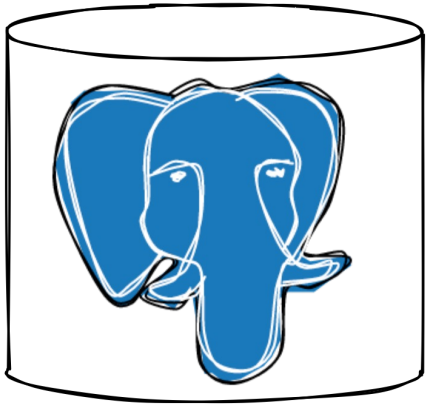
Backups

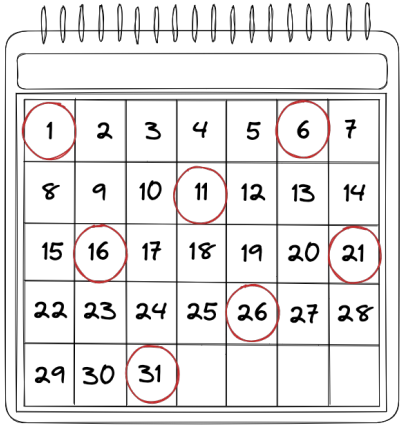
- pgBackRest
- Barman





Backups



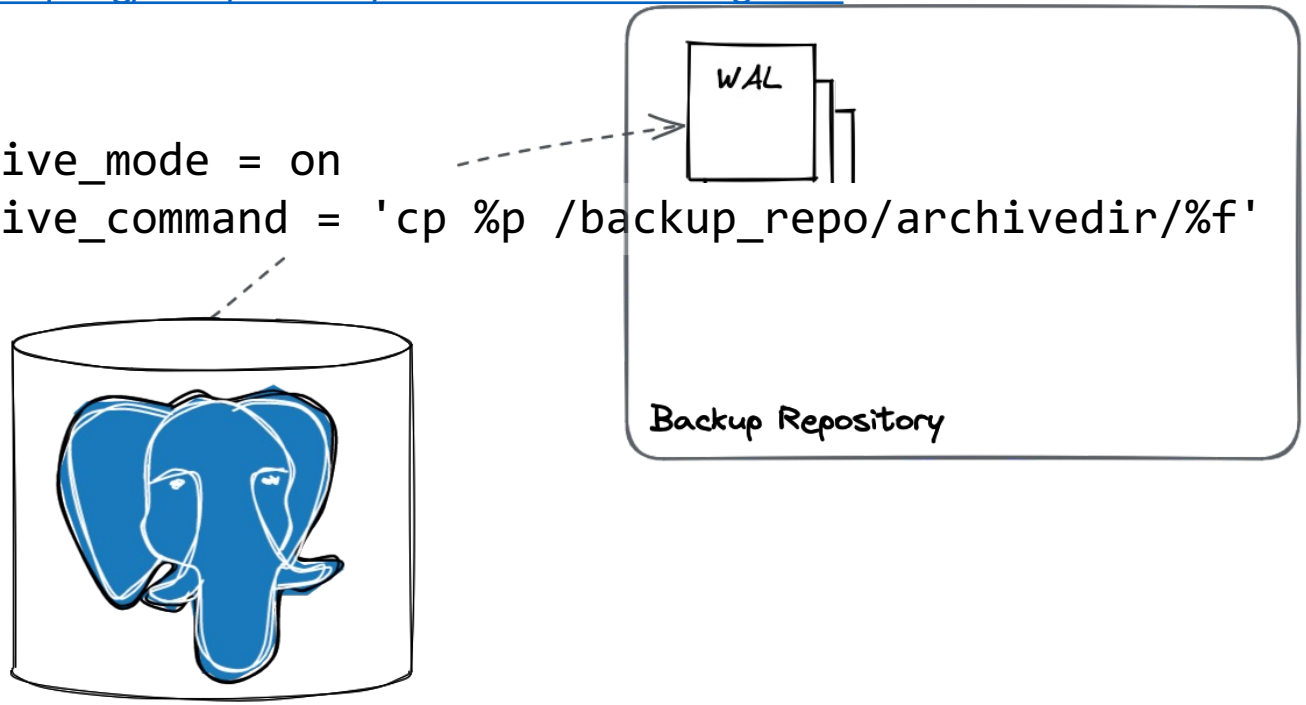


Backup Repository

Backups

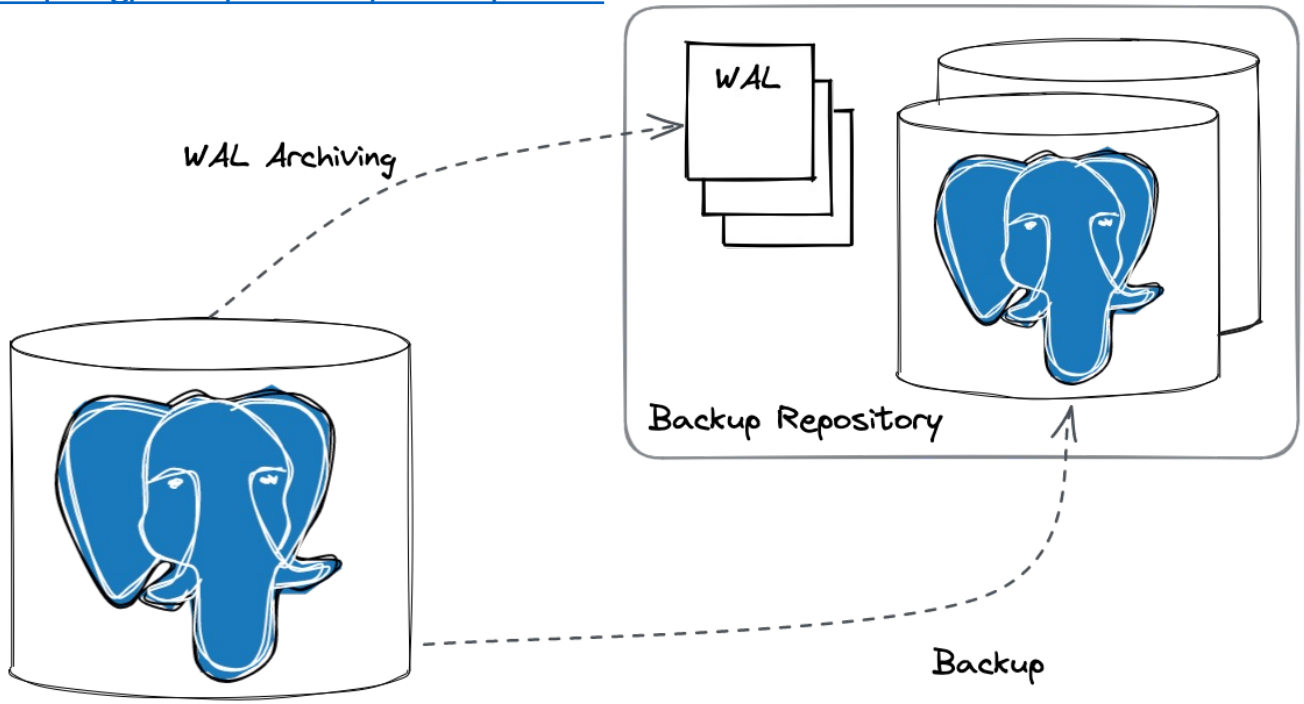
https://www.postgresql.org/docs/current/continuous-archiving.html

```
archive_mode = on
archive_command = 'cp %p /backup_repo/archivedir/%f'
```



Backups

https://www.postgresql.org/docs/current/backup.html



Configuring Backups with pgBackRest

https://pgbackrest.org/user-guide.html#quickstart

```
karen$ sudo apt install -y pgbackrest

postgres$ vi /etc/pgbackrest.conf

[global]
repo1-path=/backup_repo/pgbackrest
repo1-retention-full=2

[main]
pg1-path=/var/lib/postgresql/16/main
```

Configuring Backups with pgBackRest

https://pgbackrest.org/user-guide.html#quickstart

```
postgres$ vi /etc/postgresql/16/main/postgresql.conf

        archive_command = 'pgbackrest --stanza=main archive-push %p'

karen$ sudo systemctl restart postgresql

postgres$ pgbackrest --stanza=main --log_level_console=info stanza-create

postgres$ pgbackrest --stanza=main --log_level_console=info check
```

Configuring Backups with pgBackRest

https://pgbackrest.org/user-guide.html#quickstart

```
postgres$ pgbackrest --stanza=main --type=full backup

postgres$ crontab -e

        #m h  dom mon dow    command

        30 06  *   *   0     pgbackrest --type=full --stanza=main backup
        30 06  *   *   1-6   pgbackrest --type=diff --stanza=main backup

postgres$ pgbackrest --stanza=main info
```

Schrödinger's Backup

"The condition of any backup is unknown until a restore is attempted."

@nixcraft

Agenda

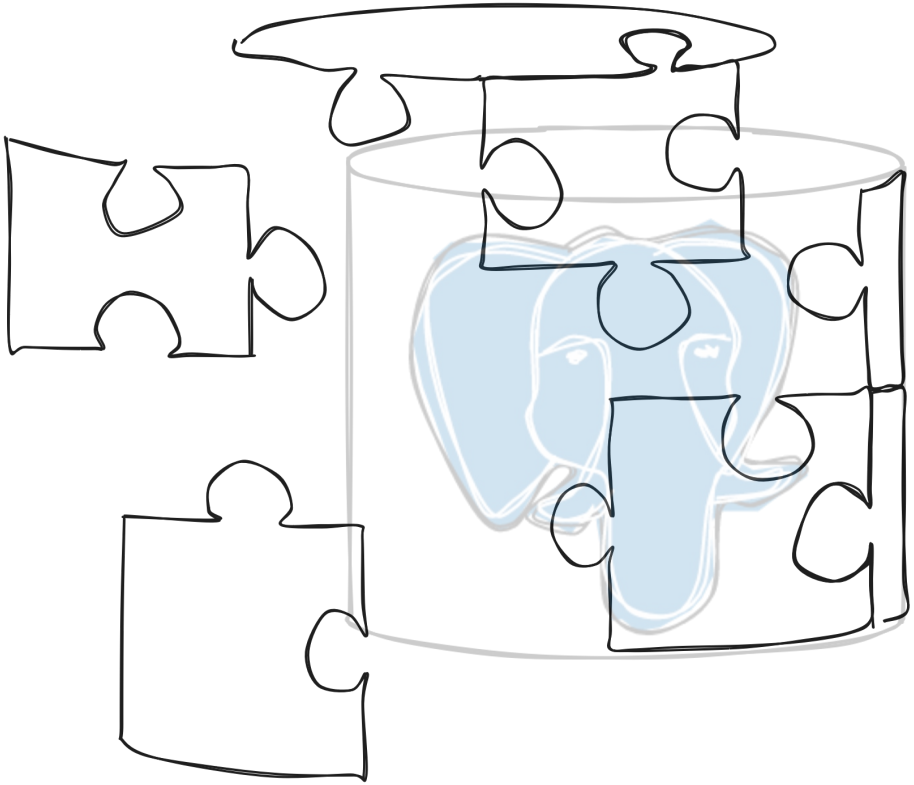
1. Set Memory & Performance Parameters
2. Schedule (and test) Backups
3. **Implement High Availability**
4. Configure Connections
5. Put in place Logging, Monitoring & Alerting

High Availability

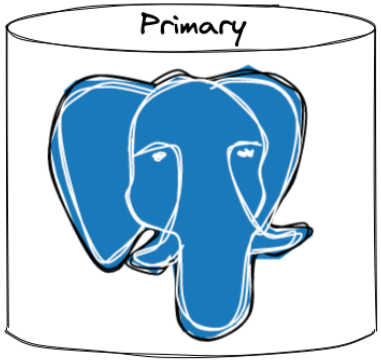
- If unable to wait for a restore
- Implement high availability

  or

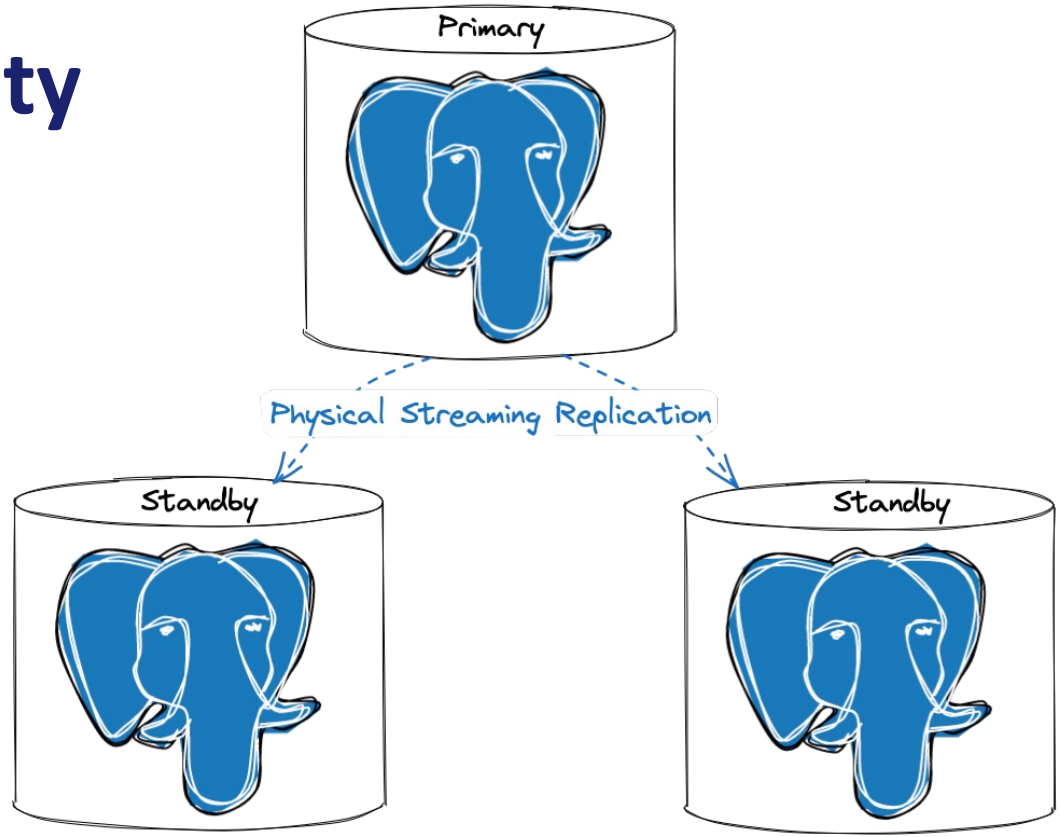
- Consider a managed service



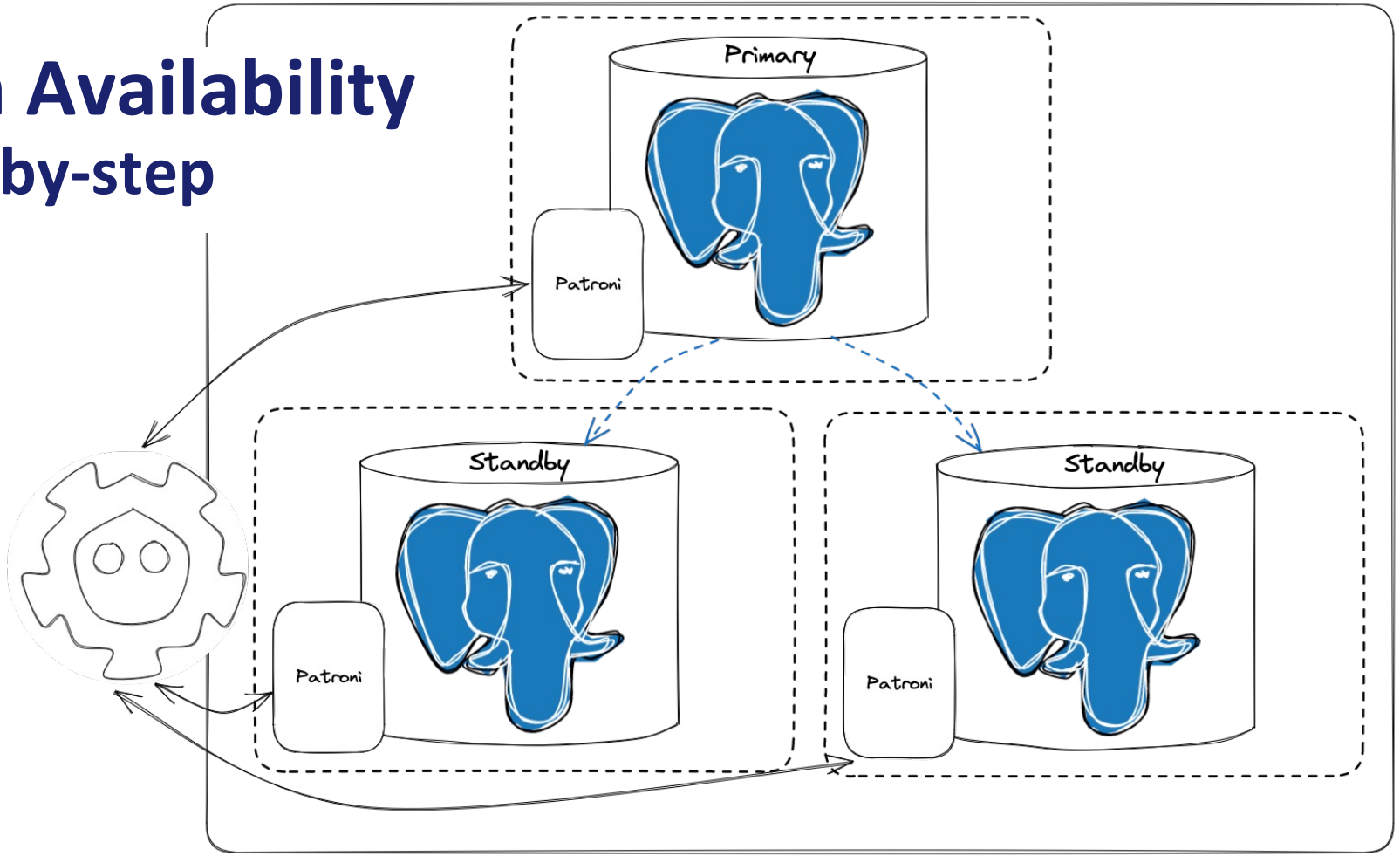
High Availability
Step-by-step



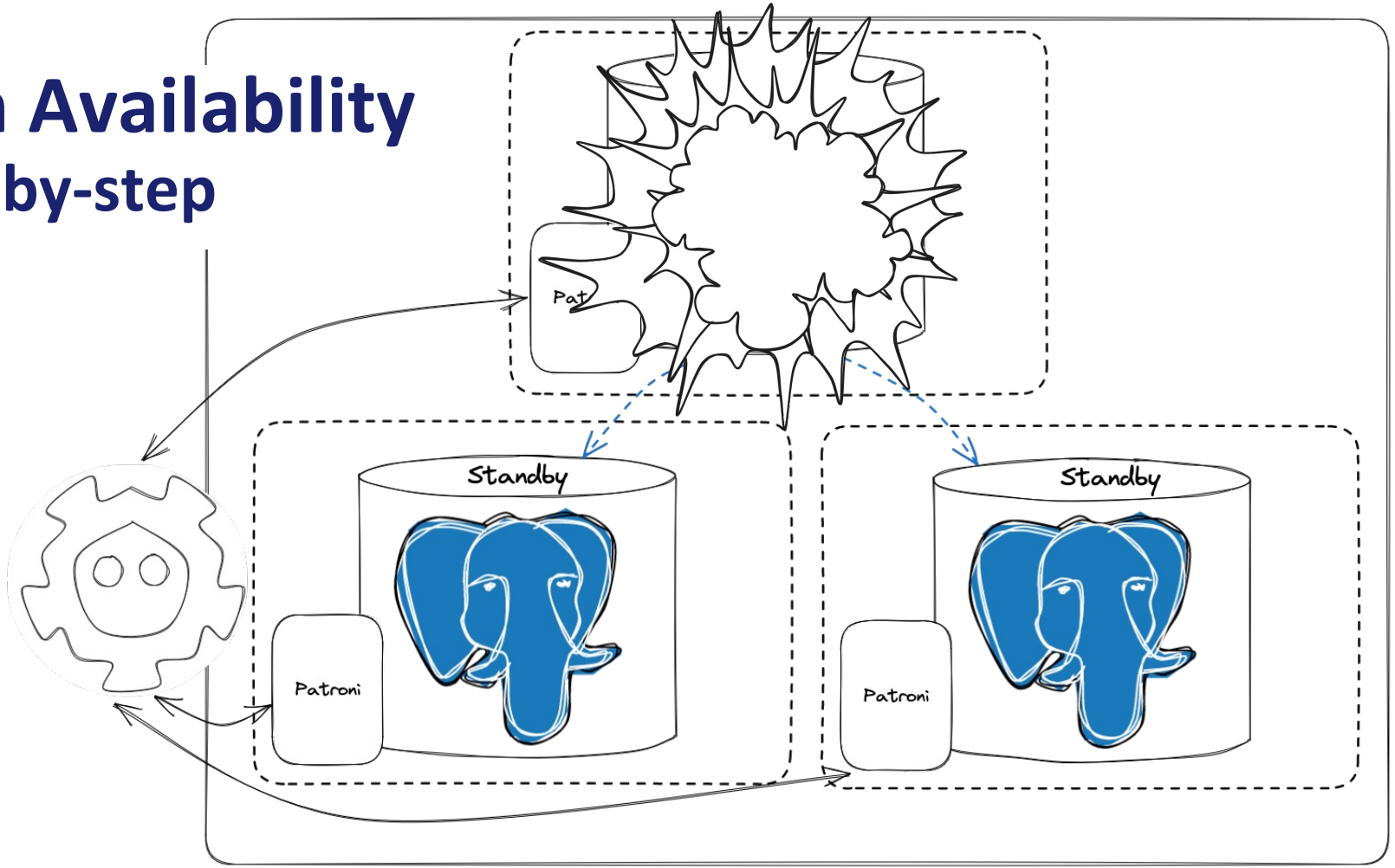
High Availability
Step-by-step



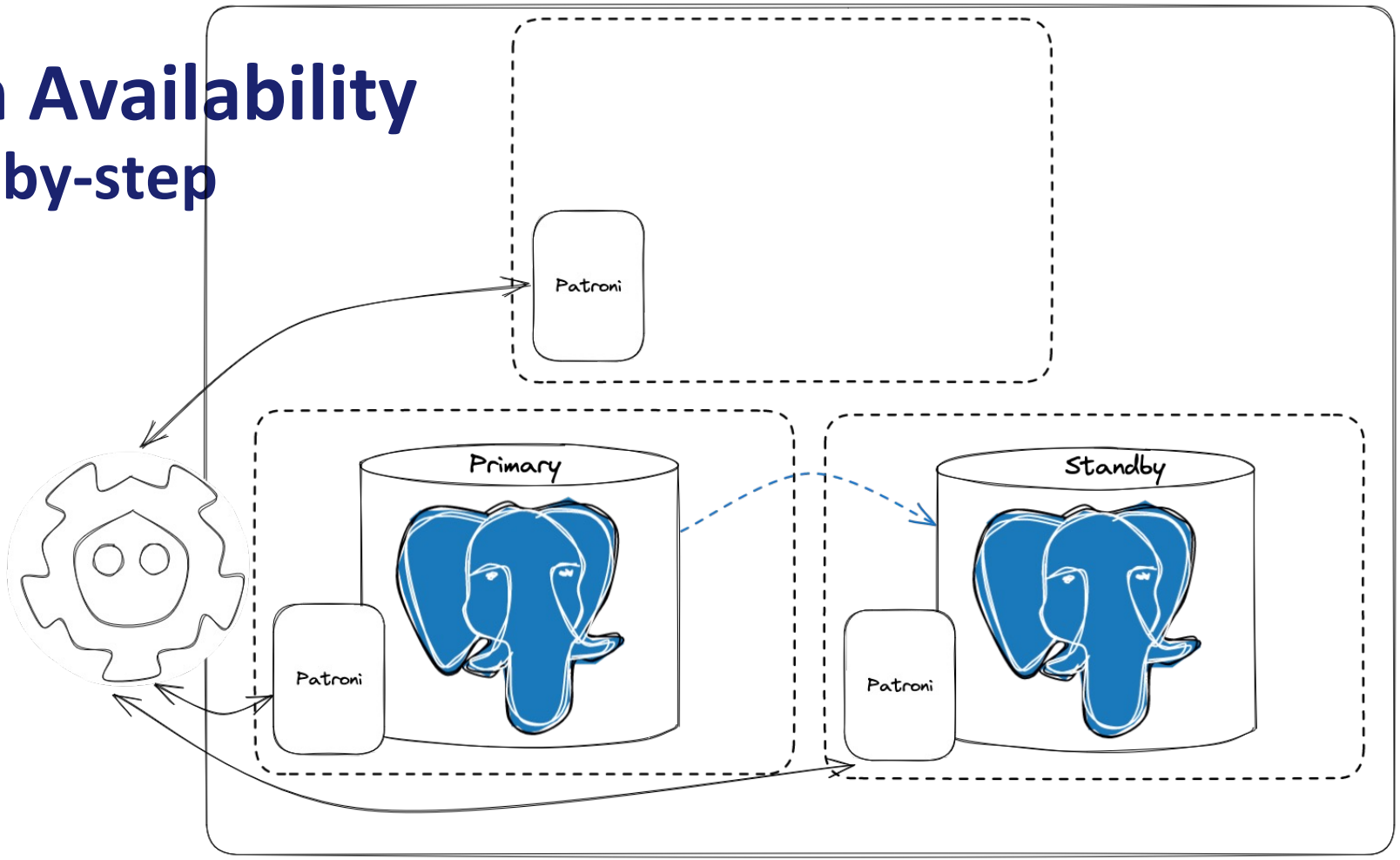
High Availability
Step-by-step



High Availability
Step-by-step



High Availability
Step-by-step



High Availability: Documentation

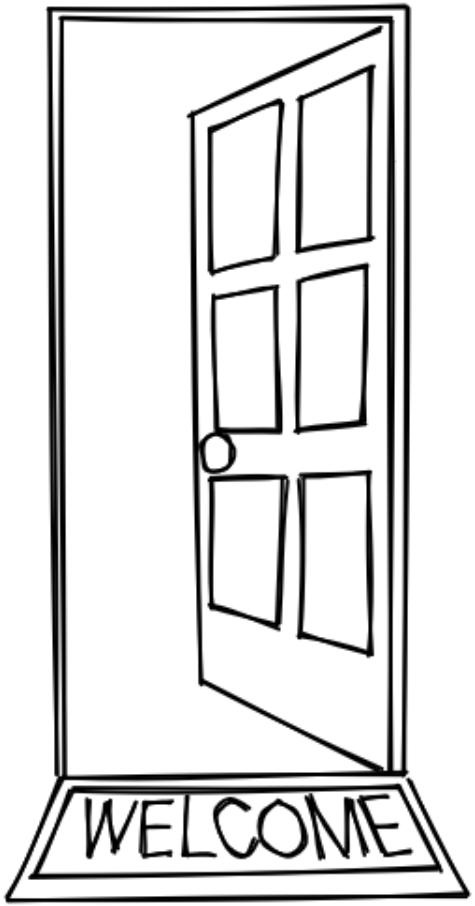
- https://www.postgresql.org/docs/current/high-availability.html
- https://www.postgresql.org/docs/current/runtime-config-replication.html
- https://etcd.io/docs/v3.5/
- https://patroni.readthedocs.io/en/latest/README.html

Agenda

1. Set Memory & Performance Parameters
2. Schedule (and test) Backups
3. Implement High Availability
4. **Configure Connections**
5. Put in place Logging, Monitoring & Alerting

Configure Connections

- Define where/who to allow connections from
- Determine how users can authenticate
- Limit number of concurrent connections
- Prevent inactive sessions blocking other users
- Create appropriate application users



Configure Connections

https://www.postgresql.org/docs/current/runtime-config-connection.htm

```
listen_addresses = '*'
```

Allow connections from all available IP interfaces

Sets the host name or IP address(es) to listen to

Configure Connections

https://www.postgresql.org/docs/current/runtime-config-connection.htm

```
listen_addresses = '*'

max_connections = no more than 500
```

May need to be much lower (<100) on a small system

maximum number of concurrent client connections allowed

Consider connection pooling above a few hundred connections

Configure Connections

https://www.postgresql.org/docs/current/runtime-config-connection.htm

```
listen_addresses = '*'

max_connections = no more than 500

idle_in_transaction_session_timeout = 30 mins
```

Maximum allowed idle time between queries, when in a transaction

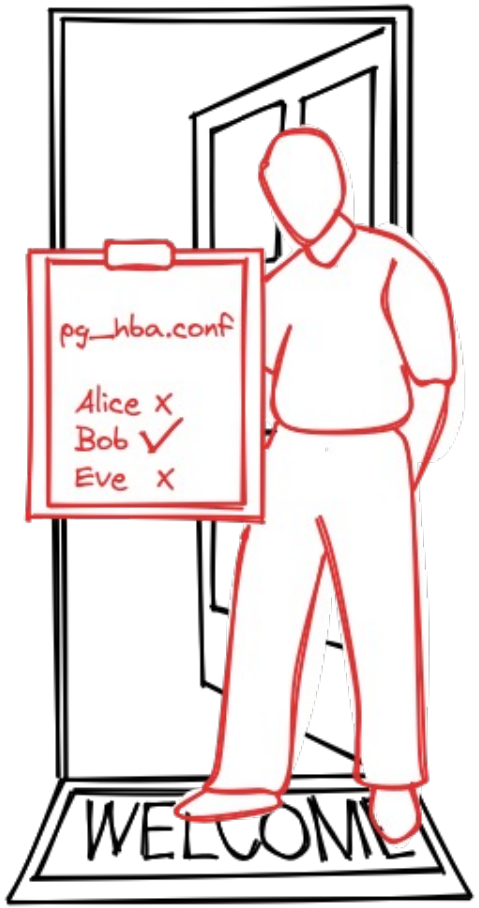
Configure Connections: pg_hba.conf

https://www.postgresql.org/docs/current/auth-pg-hba-conf.html

pg_hba.conf controls client authentication

- Users who are (or aren't) allowed to connect
- Where they can connect from
- Authentication method to use

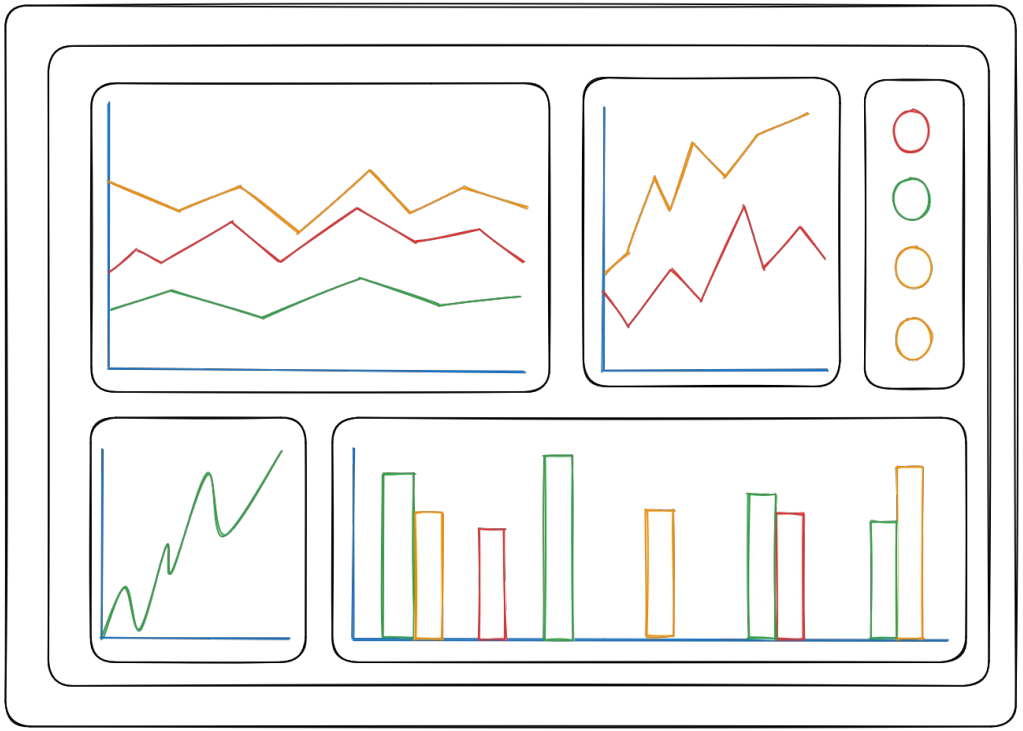
Default pg_hba.conf allows local connections only



Agenda

1. Set Memory & Performance Parameters
2. Schedule (and test) Backups
3. Implement High Availability
4. Configure Connections
5. **Put in place Logging, Monitoring & Alerting**

Logging, Monitoring & Alerting



Logging, Monitoring & Alerting

```
postgres=# show logging_collector;
 logging_collector
-------------------
 on
```

Logging, Monitoring & Alerting

```
postgres=# show log_directory;
 log_directory
---------------
 log

postgres=# show data_directory;
 data_directory
-----------------------------
 /var/lib/postgresql/16/main

postgres=# show log_filename;
 log_filename
--------------------------------
 postgresql-%Y-%m-%d_%H%M%S.log
```

Logging, Monitoring & Alerting: Parameters

https://www.postgresql.org/docs/current/runtime-config-logging.html

```
postgres=# alter system set log_min_duration_statement='1s';
```

or whatever counts as
"too long" in your system

Logging, Monitoring & Alerting: Parameters

https://www.postgresql.org/docs/current/runtime-config-logging.html

```
postgres=# alter system set log_min_duration_statement='1s';

postgres=# alter system set log_line_prefix= '%t:%r:%u@%d:[%p]: ';
```

timestamp

host connecting from

db user name

database connecting to

PID

Logging, Monitoring & Alerting: pg_stat_statements

https://www.postgresql.org/docs/current/pgstatstatements.html

```
shared_preload_libraries = 'pg_stat_statements'

postgres=# create extension pg_stat_statements;
CREATE EXTENSION

postgres=# select userid, queryid, mean_exec_time, calls from pg_stat_statements;

 userid |       queryid        | mean_exec_time | calls
--------+----------------------+----------------+-------
     10 |  3798936806175822236 |      0.1441105 |     2
     10 |  7986609124923425898 |      0.2023515 |     2
     10 | -7568047705441758065 |      0.1758495 |     2
     10 |  7338886451230691489 |       0.136604 |     2
     10 | -3691512427099624923 |         0.0287 |     2
```

Conclusions

Conclusions

- PostgreSQL really does Just Work™
- Check a few key **configuration parameters**
- Take (and test) regular **backups** of your database
- Put a **high availability** architecture in place
- Make sure the right **users/applications can connect**
- **Log activity** so you know and can react if something goes wrong
- Leave your database to look after itself

crunchydata

Thank You!

Karen Jex | @karenhjex | karen.jex@crunchydata.com

Image acknowledgements

- Elephant with balloon: Jan-Mikael Stjernberg at Pixabay
- Welcome: based on image by... at Pixabay