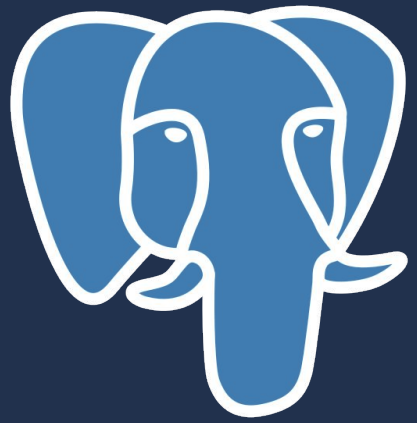
pg_upgrade 9.1 → 9.5

How we managed to upgrade with (almost) no downtime

Paul Bonaud
Developer / SysOps
@paulrb_r

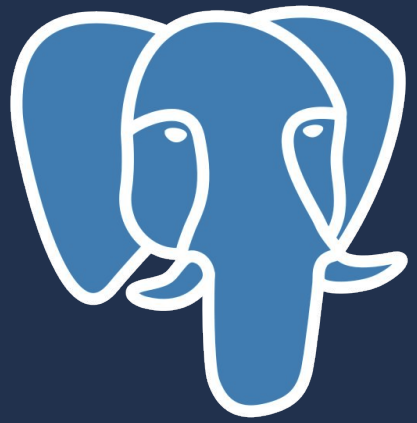
Théophile Helleboid
Developer / SysOps
@chtitux

trainline + 

Paul Bonaud
Developer / SysOps
@paulrb_r

Théophile Helleboid
Developer / SysOps
@chtitux

trainline.eu +



Paul Bonaud
Developer / SysOps
@paulrb_r

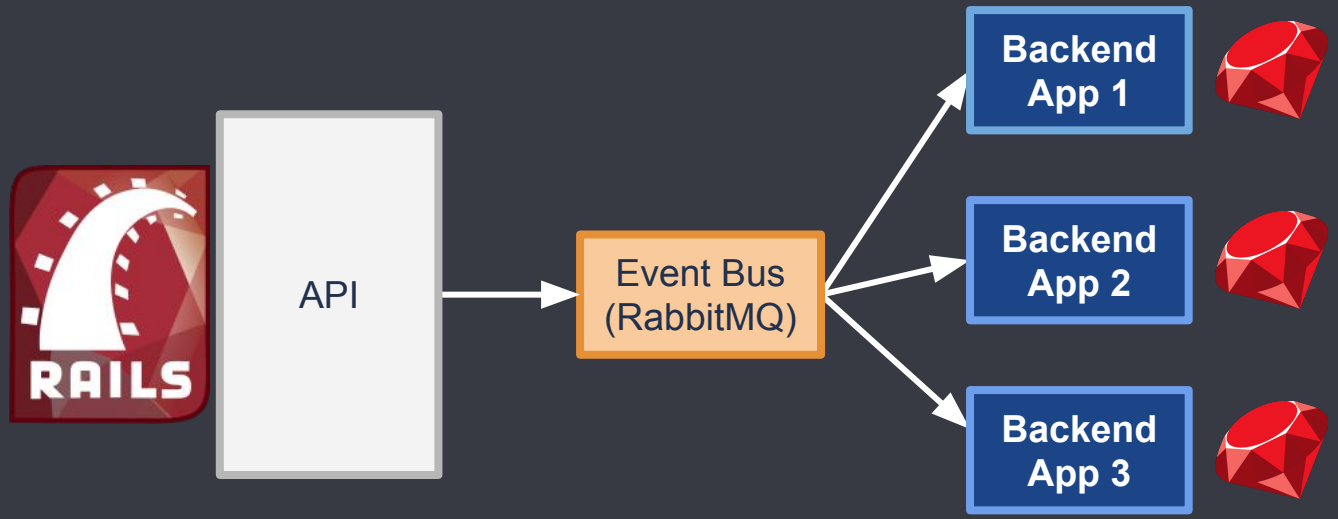
Théophile Helleboid
Developer / SysOps
@chtitux

It’s about time

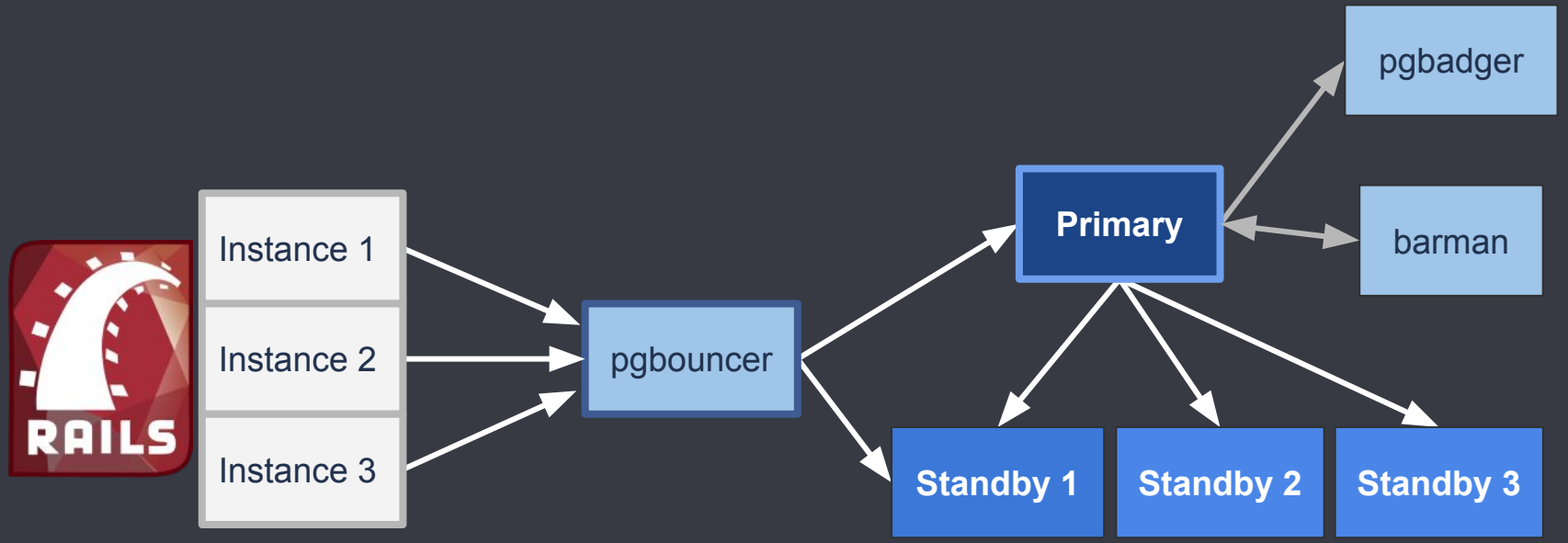
End Of Life dates (https://www.postgresql.org/support/versioning/)

Version	Current minor	Supported	First release date	EOL date
9.1	9.1.24	**No**	September 2011	**September 2016**

Application architecture



System architecture of one App



Methods for upgrading
SQL dump+restore

- “no brain migration”
- Pros
  - Very simple process: `pg_dumpall | psql -p5433`
  - Rollback is easy
  - Bonus: Cluster is garbage-collected
- Cons
  - Long downtime with big database

Methods for upgrading
`pg_upgrade`

- “PostgreSQL has powerful tools, use them!”. postgresql.org/docs/9.6/static/pgupgrade.html
- 2 usages
  - Copy the data and upgrade: longer but rollback is easy
  - Upgrade “in place” (aka “hard links”): lightning fast but no rollback after upgrade
- Pros
  - Almost no downtime
  - Upgrade “in place” has a probably very acceptable downtime
- Cons
  - Not the simplest PostgreSQL tool
  - No Rollback with “in place” upgrade
  - Extensions must be in the same versions

Methods for upgrading
logical replication

- “trigger based replication”
- An external tool replicates the changes logically to an up-to-date cluster
  - Slony
  - pglogical (PostgreSQL >= 9.4)
- Pro
  - Virtually no downtime
  - Rollback is easy
  - Bonus: the cluster is garbage-collected
- Cons
  - Complex to setup
  - Risk of split brain during migration

Methods for upgrading

SQL dump+restore

`pg_upgrade`

logical replication

pg_upgrade @ trainline.eu

Upgrade
RTFM, test & train

Read - Releases notes, tools man pages, etc.

Test - The app, the upgraded Cluster, the updated clients

Train - PostgreSQL tools, upgrade process, etc.

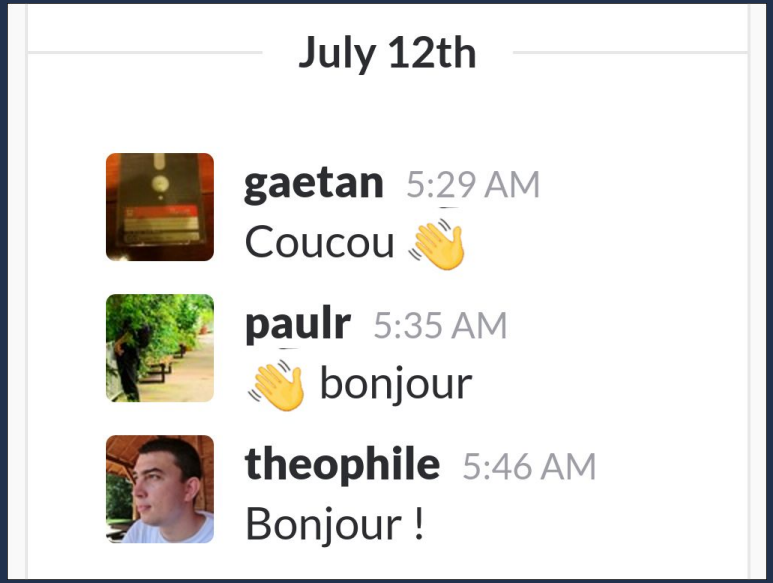
Upgrade

Write everything, prepare for the worst

- Write down every commands
  - In your documentation
  - In a shell script
- Take decisions **before** upgrade
  - Who launch the script?
  - When do we decide we need to rollback ?
  - What to do if we need to rollback?
- Inform your staff
- **Have fun!**

Upgrade

Here we go!



Upgrade
Here we go!

- (expected) Steps:
  - Stop the application
  - Stop the SQL clusters
  - **`pg_upgrade`**
  - rsync on the standbys
  - Start the primary, the standbys and the application

```
/usr/lib/postgresql/9.5/bin/pg_upgrade                --link       \
      -b /usr/lib/postgresql/9.1/bin                               \
      -B /usr/lib/postgresql/9.5/bin                               \
      -d /var/lib/postgresql/9.1/main                              \
      -D /var/lib/postgresql/9.5/main                              \
      -o ' -c config_file=/etc/postgresql/9.1/main/postgresql.conf' \
      -O ' -c config_file=/etc/postgresql/9.5/main/postgresql.conf'
```

- Downtime
  - Announced : 30 minutes
  - Expected : 5 minutes
  - Experienced : 25 minutes

Aftermath
Bad things happen

- Replication info was incorrect. *Missing details?*
  - *Verify that the "Latest checkpoint location" values match in all clusters with* `pg_controldata`

- Upgrade of standby with rsync+hard links does not work well *(it was very slow)*
  - *Resignation on upgrading standbys (risk taken to have 24h without any standbys)*
  - *Full restore in the aftermath*

Aftermath

Fix things that broke

Barman has a bug with freshly upgraded clusters

Log files changed path so **pgbadger** did not work anymore

Different projects on same integration cluster

→ We need a new cluster!

Next time
… was actually last time

- 2nd `pg_upgrade` at Trainline:
  - 30 seconds of downtime
  - Everything in a bash script
  - Application paused, not stopped
- Better experience
- Less bugs in our applications
  - “App always use the default port” now fixed

Next time
Modern tools

- Towards PostgreSQL 9.6

- No downtime for real:
  pglogical to the rescue


Profit!
... and questions?

```
#!/bin/bash -xe

# Test connection to pgbouncer is OK
psql postgres://pgbouncer@pgbouncer.sql.production:6432/pgbouncer --command    'show pools;'

# Pause the databases;
psql postgres://pgbouncer@pgbouncer.sql.production:6432/pgbouncer --command    'PAUSE rails_app;'

# Stop the 9.1 server
pg_ctlcluster 9.1 main stop -m fast

# Test to upgrade the data
time /usr/lib/postgresql/9.5/bin/pg_upgrade  --check --link       \
   -b /usr/lib/postgresql/9.1/bin                                 \
   -B /usr/lib/postgresql/9.5/bin                                 \
   -d /var/lib/postgresql/9.1/main                                \
   -D /var/lib/postgresql/9.5/main                                \
   -o ' -c config_file=/etc/postgresql/9.1/main/postgresql.conf'   \
   -O ' -c config_file=/etc/postgresql/9.5/main/postgresql.conf'

# Really upgrade the data
time /usr/lib/postgresql/9.5/bin/pg_upgrade         --link         \
   -b /usr/lib/postgresql/9.1/bin                                 \
   -B /usr/lib/postgresql/9.5/bin                                 \
   -d /var/lib/postgresql/9.1/main                                \
   -D /var/lib/postgresql/9.5/main                                \
   -o ' -c config_file=/etc/postgresql/9.1/main/postgresql.conf'   \
   -O ' -c config_file=/etc/postgresql/9.5/main/postgresql.conf'

# Start the 9.5 server
pg_ctlcluster 9.5 main start

# Test cluster is accepting connections
psql --dbname=rails_app --command= "SELECT NOW();"

# Resume the connexions
psql postgres://pgbouncer@pgbouncer.sql.production:6432/pgbouncer --command    'RESUME rails_app;'

echo "DONE"
```