

Untangling the PostgreSQL upgrade

Martín Marqués

2ndQuadrant

4 de agosto de 2018

Contents

- 1 Introduction
- 2 Point Releases
- 3 Major upgrades
- 4 Zero downtime
- 5 Conclusion
- 6 Questions?

Upgrades

- ▷ Point release upgrades
- ▷ Major version upgrade

A note on versions

- ▷ Point releases:
 - ▷ 8.4.x, 9.1.x or 9.6.x (before version 10)
 - ▷ 10.x

A note on versions

- ▷ Point releases:
 - ▷ 8.4.x, 9.1.x or 9.6.x (before version 10)
 - ▷ 10.x
- ▷ Major version upgrade
 - ▷ 8.2, 9.3, 9.6, 10, 11

Upgrades

- ▷ Point release upgrades

- ▷ 9.6.6 → 9.6.9

- ▷ 10.3 → 10.4

Upgrades

- ▷ Point release upgrades

- ▷ 9.6.6 → 9.6.9

- ▷ 10.3 → 10.4

- ▷ Major version upgrade

- ▷ 9.2.20 → 9.6.9

- ▷ 9.4.18 → 10.4

Contents

- 1 Introduction
- 2 Point Releases
- 3 Major upgrades
- 4 Zero downtime
- 5 Conclusion
- 6 Questions?

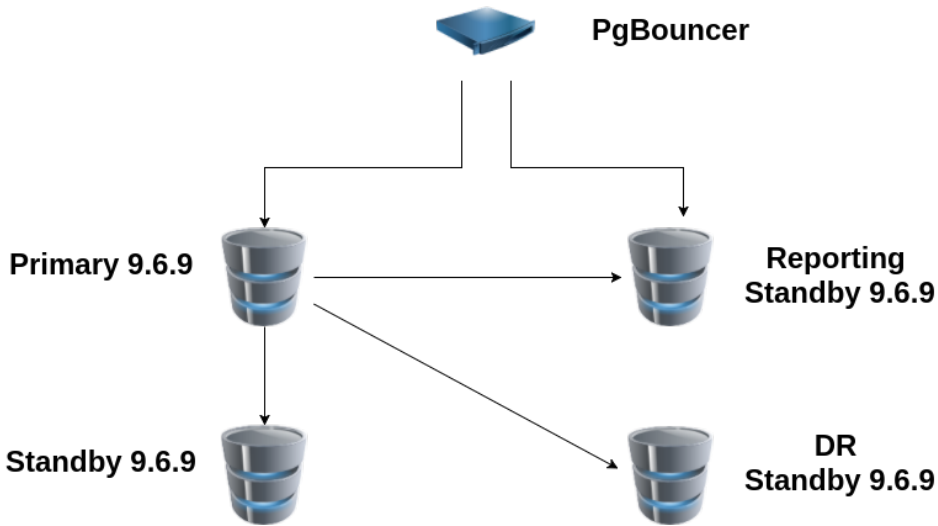
Upgrades for Point Releases

- ▷ Why?
- ▷ How?

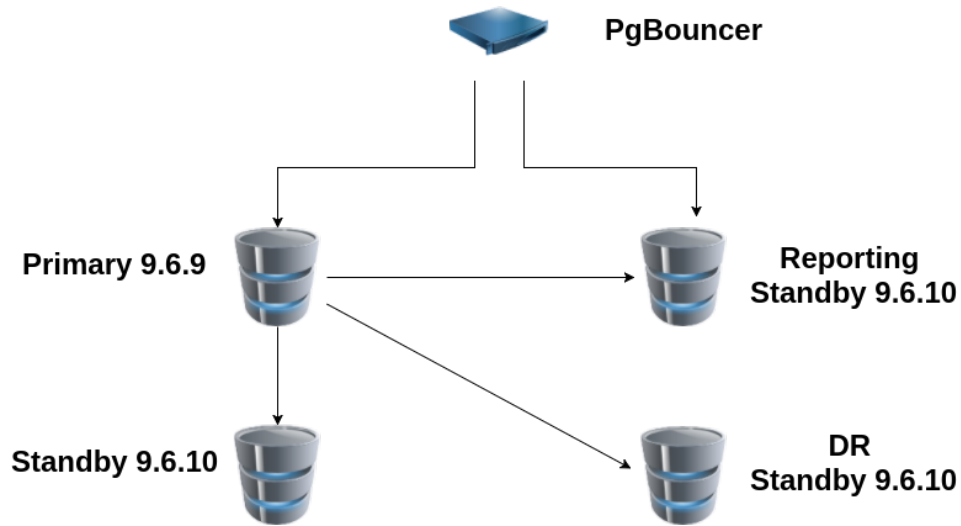
Upgrades for Point Releases

- ▷ Why?
- ▷ How?
 - ▷ Upgrade standbys 1 by 1
 - ▷ Switchover master/standby
 - ▷ Upgrade remaining node
 - ▷ Failback

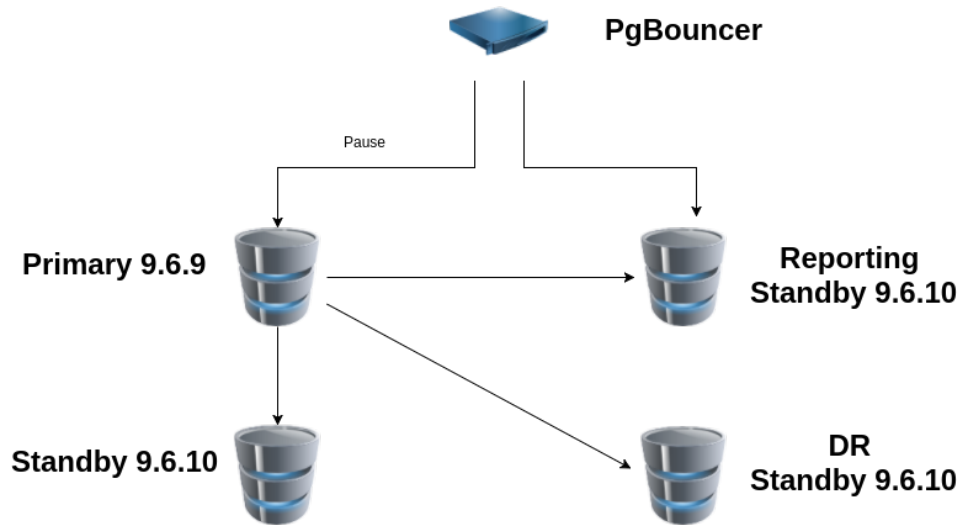
Initial setup of the cluster



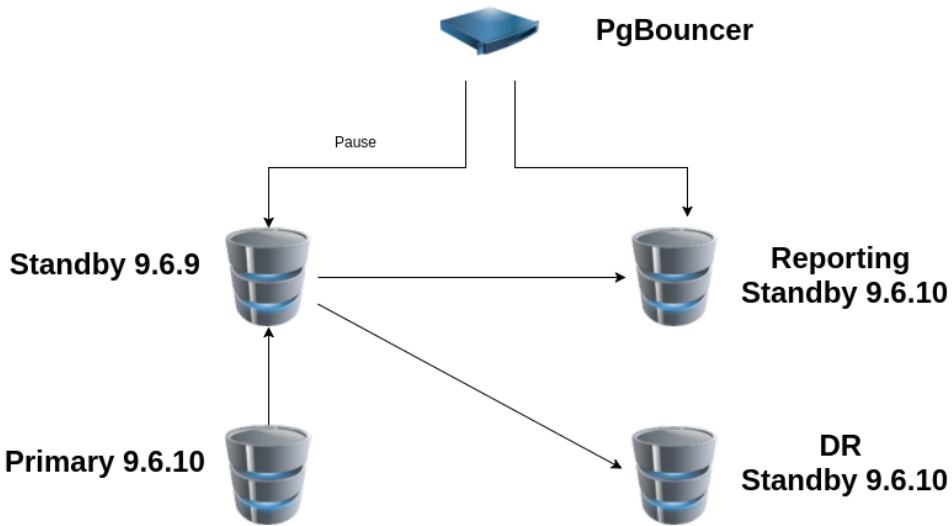
Cluster with Standbys upgraded



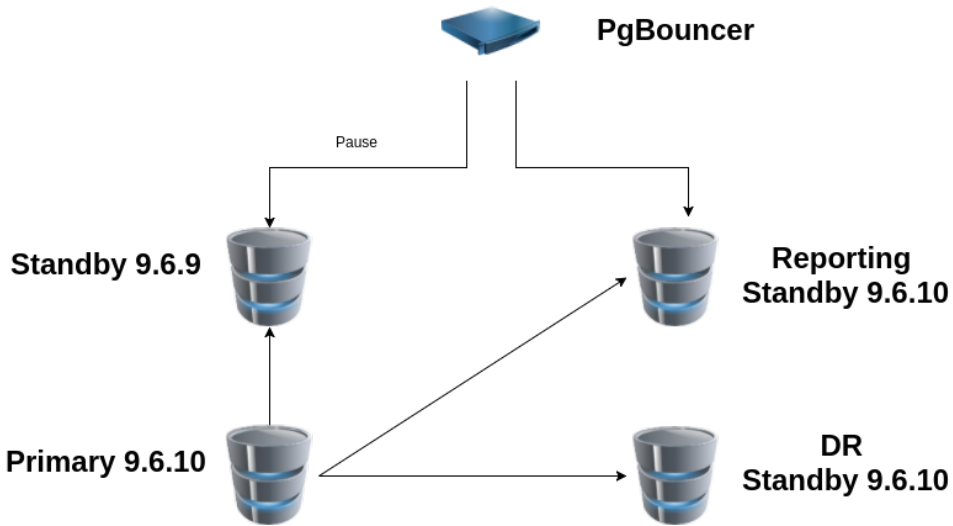
Switchover



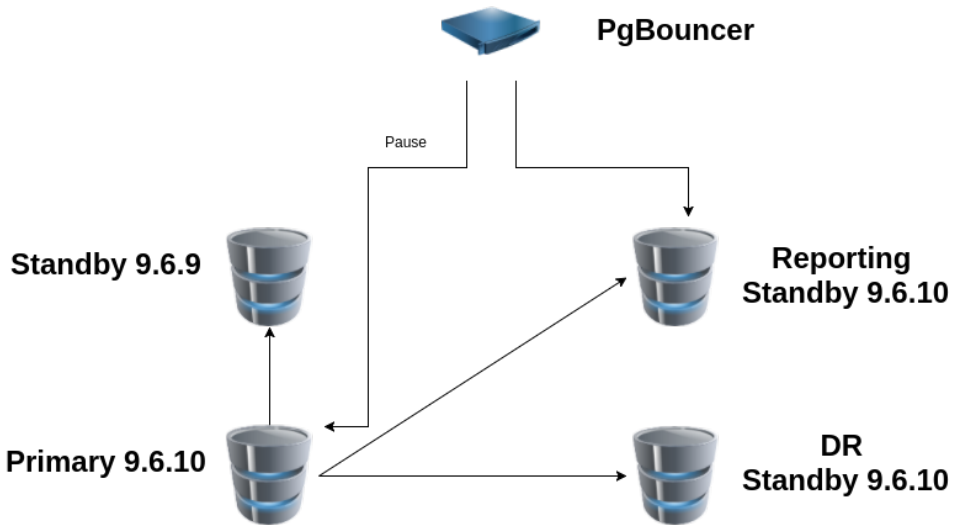
Switchover



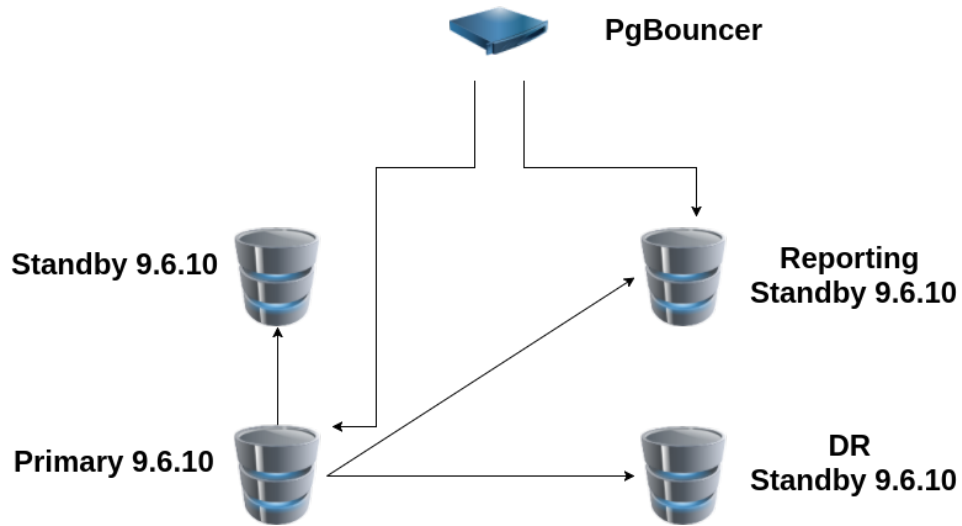
Switchover



Switchover



Switchover



Point release upgrade homework

- ▷ Write an Ansible playbook

Point-release upgrade homework



Contents

- 1 Introduction
- 2 Point Releases
- 3 Major upgrades**
- 4 Zero downtime
- 5 Conclusion
- 6 Questions?

Major upgrades

- ▷ Why?
 - ▷ New features
 - ▷ Some code path changes
 - ▷ Stay on a supported version

Major upgrades

- ▷ Why?
 - ▷ New features
 - ▷ Some code path changes
 - ▷ Stay on a supported version
- ▷ How?
 - ▷ logical upgrade with downtime
 - ▷ in-place upgrade
 - ▷ logical upgrade with near-zero downtime

Major upgrades

- ▷ Why?
 - ▷ New features
 - ▷ Some code path changes
 - ▷ Stay on a supported version
- ▷ How?
 - ▷ logical upgrade with downtime → `pg_dump` && `pg_restore`
 - ▷ in-place upgrade
 - ▷ logical upgrade with near-zero downtime

Major upgrades

- ▷ Why?
 - ▷ New features
 - ▷ Some code path changes
 - ▷ Stay on a supported version
- ▷ How?
 - ▷ logical upgrade with downtime → `pg_dump` && `pg_restore`
 - ▷ in-place upgrade → `pg_upgrade`
 - ▷ logical upgrade with near-zero downtime

Major upgrades

- ▷ Why?
 - ▷ New features
 - ▷ Some code path changes
 - ▷ Stay on a supported version
- ▷ How?
 - ▷ logical upgrade with downtime → `pg_dump` && `pg_restore`
 - ▷ in-place upgrade → `pg_upgrade`
 - ▷ logical upgrade with near-zero downtime → Stay tuned

pg_dump

- ▷ Pros:
 - ▷ End with cluster clean from bloat
 - ▷ Well tested
 - ▷ Can dump in parallel and restore in parallel
 - ▷ Easy to deploy with hardware upgrade

pg_dump

- ▷ Pros:
 - ▷ End with cluster clean from bloat
 - ▷ Well tested
 - ▷ Can dump in parallel and restore in parallel
 - ▷ Easy to deploy with hardware upgrade
- ▷ Cons:
 - ▷ Doesn't scale well

pg_dump Tips

- ▷ Use directory format and as much jobs as CPUs
- ▷ Turn off any unneeded parameter in postgresql.conf
 - ▷ `archive_command='/bin/true'`
 - ▷ `autovacuum = off`
 - ▷ `synchronous_commit = off`
- ▷ Increase read-ahead on source, and test with various scheduler settings

pg_upgrade in link mode

- ▷ Pros:
 - ▷ Much faster than pg_dump
 - ▷ Doesn't need double disk space

pg_upgrade in link mode

- ▷ Pros:
 - ▷ Much faster than pg_dump
 - ▷ Doesn't need double disk space
- ▷ Cons:
 - ▷ Can take long with big schemas
 - ▷ May have problems if skipping versions
 - ▷ No going back after starting up with new version

pg_upgrade in link mode

▷ Pros:

- ▷ Much faster than pg_dump
- ▷ Doesn't need double disk space

▷ Cons:

- ▷ Can take long with big schemas
- ▷ May have problems if skipping versions
- ▷ No going back after starting up with new version

▷ Gotcha:

- ▷ PG $\leq 9.5 \rightarrow$ PG ≥ 9.6 takes longer

Contents

- 1 Introduction
- 2 Point Releases
- 3 Major upgrades
- 4 Zero downtime**
- 5 Conclusion
- 6 Questions?

Logical Replication

▷ Before **9.4**

- ▷ Trigger based → performance impact
- ▷ All tables replicated need a *Primary Key*

▷ After **9.4**

- ▷ Logical decoding → Uses WALs → No overhead
- ▷ Doesn't need *Primary Key* on all tables

Logical Replication

- ▷ Trigger based
 - ▷ Londiste
 - ▷ Slony-I
 - ▷ Bucardo
- ▷ Logical decoding (9.4+)
 - ▷ pglogical
 - ▷ PG 10 Logical replication

Logical Replication

- ▷ Trigger based
 - ▷ Londiste
 - ▷ Slony-I
 - ▷ Bucardo
- ▷ Logical decoding (9.4+)
 - ▷ pglogical
 - ▷ PG 10 Logical replication → pglogical

Logical Replication in 9.4+

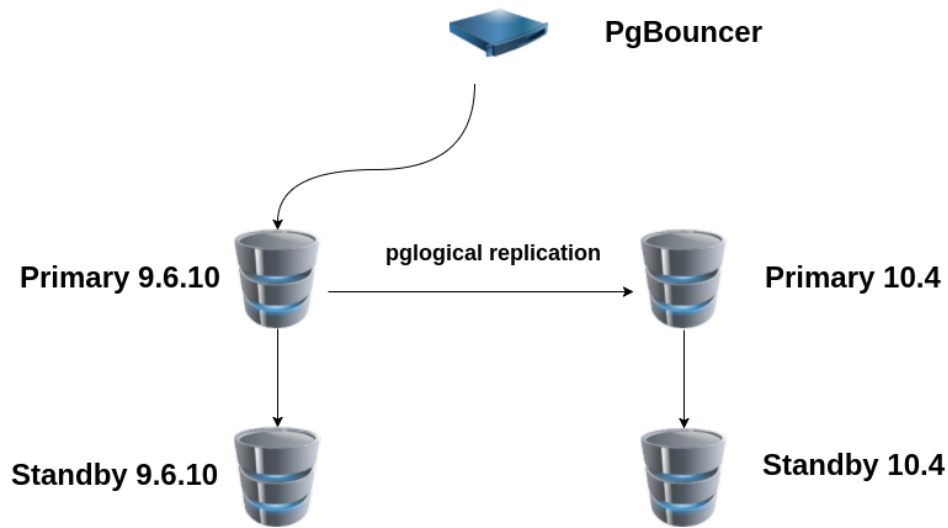
pglogical

Logical Replication in 9.4+

pglogical

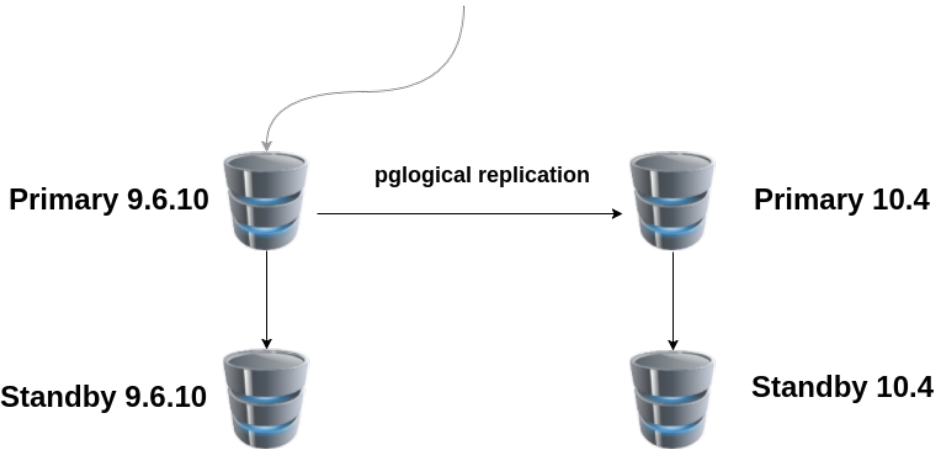
- ▷ Pros:
 - ▷ Uses logical decoding of WALs (small overhead)
 - ▷ Can upgrade the whole infrastructure
 - ▷ Can test the new cluster while replicating
- ▷ Cons:
 - ▷ Setup overhead
 - ▷ Continuous monitoring

Initial setup of the cluster

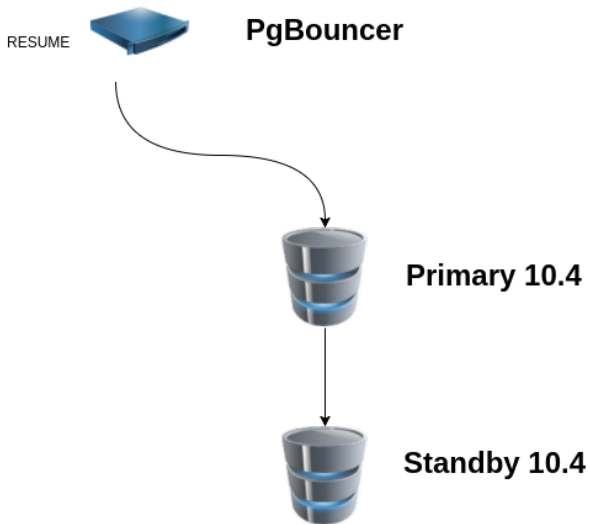


Pause pgbouncer

PAUSE  PgBouncer



Switchover pgbouncer



Contents

- 1 Introduction
- 2 Point Releases
- 3 Major upgrades
- 4 Zero downtime
- 5 Conclusion**
- 6 Questions?

Conclusion

- ▷ Plan point release upgrades as soon as available
- ▷ Stay on a community supported version
- ▷ Test your application against the upgraded version
- ▷ If enough downtime is affordable, use `pg_dump`
- ▷ It's possible to have near-zero downtime upgrade, but expensive

Contents

- 1 Introduction
- 2 Point Releases
- 3 Major upgrades
- 4 Zero downtime
- 5 Conclusion
- 6 Questions?**

Questions

?