

Your remote PostgreSQL DBA Team

data egret

An ultimate guide to upgrading your PostgreSQL installation



Ilya Kosmodemiansky

ik@dataegret.com



SECURING YOUR DATABASE AVAILABILITY, SO THAT YOUR TEAM CAN FOCUS ON NEW FEATURE DEVELOPMENT.

- Migrations
- DB audit
- Performance optimisation
- Backup & restore
- Architectural review
- Advising Data Science teams
- Developer training

on premise & cloud



EXPERTISE

Senior DBA with **10+ years**
of PostgreSQL
administration **experience**



DEVELOPMENT

Involved in
**new feature and
extension development**



TAILORED APPROACH

Felxible approach and
dedicated team focused
on success of your project



COMMUNITY

Contributing Sponsor.
Deeply involved in the
PostgreSQL community

Data Egret GmbH | 66583 Spiesen-Elversberg | Germany

contact@DataEgret.com



Why this talk?

Upgrading your PostgreSQL is not a rocket science!

- ...but there are **lots** of small details
- An unsuccessful upgrade can ruin your data
- Or at least cause an unacceptable downtime
- Upgrade requires good knowledge of your system and substantial preparation time



Because of that

- DBAs do not like upgrades
- They are attached to outdated versions
- They manage to screw up an upgrade when they finally decide to perform one

Why do you need to upgrade?

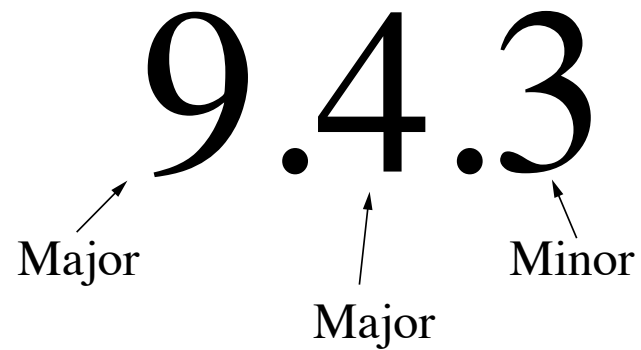
- Bugfixes, security fixes
- Many performance improvements and new features over the last years
- Upgrading on time makes it easier
- Running 7.* (or even 9.*) in 2024 would make consultants too happy

PostgreSQL version numbering

`<= 9.6.*`

9.4.3

Major Major Minor

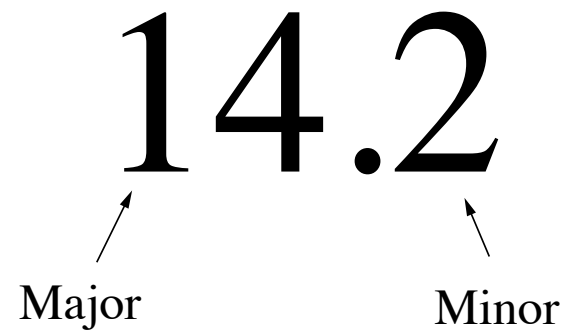
The diagram shows the version number 9.4.3. Three arrows point to the digits: one from the word 'Major' to the digit '9', one from the word 'Major' to the digit '4', and one from the word 'Minor' to the digit '3'. The dots are not labeled.

PostgreSQL version numbering

> 9.6.*

14.2

Major Minor



Types of upgrades

- Minor
 - New versions' binaries can run on old version datafiles
- Major
 - New versions' binaries can run on old version datafiles, but require new system tables and internal data format may change
- Major with special requirements

Before any upgrade

- Read carefully version specific release notes
- Play with chosen upgrade method in test environment
- Align with your application development team
- **Make a backup and check it by test recovery**



Minor upgrades are easy

- You simply install new binaries and start new database server on the old data directory
- There are no new features between minor versions
- Still - keep an eye on updating all PostgreSQL-related packages you use



Major upgrade prerequisites

- Install new versions for all PostgreSQL-related packages
- Read carefully all release notes
- Know your PostgreSQL installation
- Choose the method and carefully read the documentation for this method
- Align with your application development team
- **Do a backup and check it by doing a test recovery**



Major upgrade methods

- Good old dump/restore
- pg_upgrade
- Replication-based methods

Major upgrade using `pg_dump`

- Difficult to make without downtime if you have a large, heavy loaded database
- Requires additional disk space
- Works with any PostgreSQL version since 7.0
- `pg_dump -Fc` - custom format, `-Z` - compression

Major upgrade using `pg_dump`

- `pg_dump -Fd --jobs` can be a good option in terms of speed and downtime
- But if using `-j` you can't do things like that: `pg_dumpall -p 5432 | psql -d postgres -p 5433`
- If your installation can be upgraded easily by dump/restore, you are lucky!



Major upgrade using `pg_dump` - procedure

- Install new binaries
- Initialize new cluster. Don't forget about locale
- Change config files appropriately
- It can be a good idea to use newer version of `pg_dump`, but be careful if running on pre-9.2 server
- Restore the dump, try to figure out if everything looks good
- Switch your application to the new cluster

Install new binaries

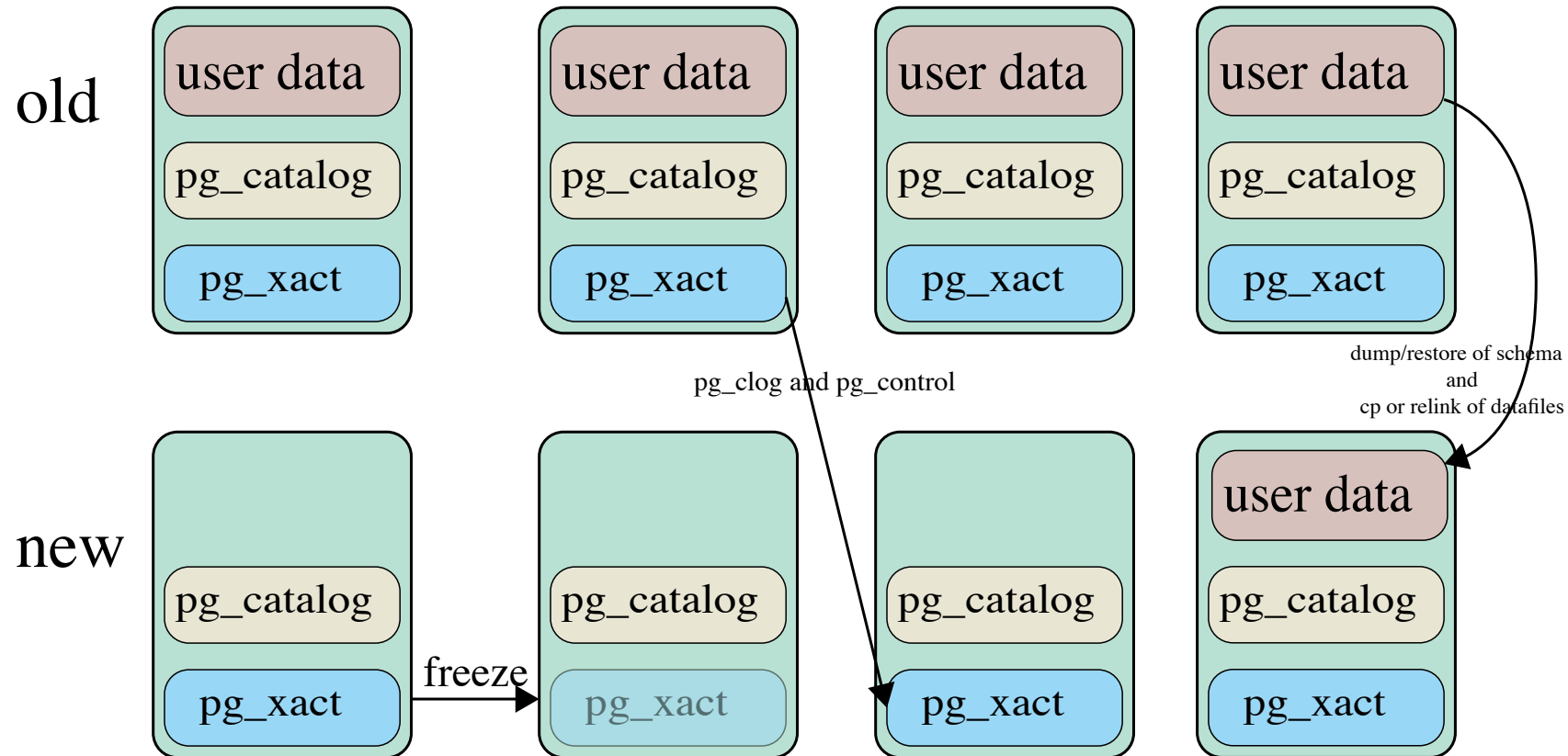
- Know your packet manager!
- Debian/Ubuntu tweaks:
 - in `/etc/postgresql/14/main/start.conf` change *auto* to *manual*
 - remember about `--download-only`



pg_upgrade - outline

- How it works?
- Procedure
 - Simple case - standalone server
 - How to minimize downtime?
 - Upgrading hot-standby cluster
- Details

pg_upgrade - How it works?



pg_upgrade - preparations

- check for replication slots (we most likely need to recreate them after upgrade)
- Read release notes
 - pg_upgrade documentation
 - incompatibilities section
 - check list of your extensions and their compatibility
- Discuss procedure with your Dev Team
- **make an extra backup**
- `pg_dumpall -s`



pg_upgrade - procedure

- Create empty database for new version of PostgreSQL
- Dry run pg_upgrade with `--check`
- Stop database with old PostgreSQL version
- Start upgrade procedure with pg_upgrade command
- Start database that runs on new PostgreSQL version

pg_upgrade - procedure

- Start collecting statistics (pg_upgrade does not transfer optimizer statistics)
- When statistic collection started, you can open database for your application
- Depending on your database, you can achieve 1-10 min downtime target

pg_upgrade --check

```
sudo -iu postgres /usr/lib/postgresql/17/bin/pg_upgrade \  
--old-bindir /usr/lib/postgresql/16/bin/ --new-bindir /usr/lib/postgresql/17/bin/ \  
--old-datadir /var/lib/postgresql/16/main/ --new-datadir /var/lib/postgresql/17/main/ \  
--old-options " -c config_file=/etc/postgresql/16/main/postgresql.conf" \  
--new-options " -c config_file=/etc/postgresql/17/main/postgresql.conf" \  
--new-options " -c archive_mode=off" --retain --jobs 4 --link --check  
Performing Consistency Checks on Old Live Server
```

```
-----  
Checking cluster versions                                ok  
Checking database user is the install user            ok  
Checking database connection settings                  ok  
Checking for prepared transactions                    ok  
Checking for contrib/isn with bigint-passing mismatch ok  
Checking data type usage                              ok  
Checking for presence of required libraries           ok  
Checking database user is the install user            ok  
Checking for prepared transactions                    ok  
Checking for new cluster tablespace directories       ok
```

Clusters are compatible



pg_upgrade - minimizing downtime

- Use pgbouncer
 - *PAUSE/RESUME*
 - Issue *CHECKPOINT*; on old server before you start, to make shutdown process faster
- Use *-k (--link)* to use hard links instead of copy (**but carefully!**)

pg_upgrade - hot-standby replica

- Upgrade primary as a standalone server
- Keep replica intact to failover if something goes wrong
- Reinstantiate your replica
- Procedure
 - Pause pgbouncer on standby or stop it
 - Clone a replica from upgraded primary using pg_basebackup
 - Start replica with new binaries
 - Resume pgbouncer connections or start pgbouncer.

pg_upgrade - Details

- **pg_upgrade does not transfer optimizer statistics**
- instead, it suggests to run
 - `/usr/lib/postgresql/17/bin/vacuumdb --all --analyze-in-stages`
 - In some cases you can run `vacuumdb --all --analyze-only`
 - You can vacuumdb run in parallel (`-j 20`)
 - We usually use `vacuumdb --all --analyze-in-stages` and open database for application after medium optimizer statistics (10 targets) are generated

pg_upgrade - Details

- Documentation suggests to use rsync to reinstantiate standby
- `rsync --archive --delete --hard-links --size-only old_pgdata new_pgdata remote_dir`
- rsync allows you to save a lot of network traffic in that case
- ...and provides **lots** of opportunities to shoot yourself in the foot
- **pg_basebackup** is generally safer
 - `pg_basebackup -v -P -R -c fast -h IP -U replica -D /var/lib/postgresql/14/main --wal-method=stream`



pg_upgrade - Details

- **Debian/Ubuntu follow their own way**
- Wrappers, like *pg_ctlcluster* are designed to manipulate PostgreSQL cluster in a Debian way
- `pg_upgradecluster -v 9.5 9.3 main1 -m upgrade -k` supposed to be a make-me-happy button for a DBA
- It even takes care of converting postgresql.conf parameters for you
- **But I strongly recommend to do this manually**

pg_upgrade - Details

- Extensions can surprise you
- pg_upgrade keeps old versions of extensions
- We advise to cycle through all extensions and perform *alter extension EXTENSION_NAME update;*
- Some extensions need special care: for example PostGIS should be updated before an upgrade

pg_upgrade - Checksums

- checksums were not default before a recent patch
- pg_upgrade makes a physical copy of the database, so if it was initialized without checksums, you need to enable them with `pg_checksums -D /var/lib/postgresql/16/main --enable --verbose`
- this is an offline operation, so good idea is to enable it on replica, than do an upgrade of it with performing a switchover



pg_upgrade - work in progress

- There was discussion of pg_upgrade at Developer Unconference 2023 in Ottawa
 - no statistic in the new cluster is the biggest problem
 - a nice idea is to sample old cluster and transfer that statistics
 - implementation of that could be not easy
- Currently several people are working on implementing it, see [this](#) thread

pg_upgrade - notes on pgBackRest

- check version compatibility
- right after pg_upgrade
 - change path in pgbackrest.conf
 - `pgbackrest --stanza=$STANZA stanza-upgrade --no-online`

Using replication to upgrade PostgreSQL

- Streaming replication doesn't work between versions
- But some replication methods can do that
 - Logical replication
 - Slony-I (Yes, even in 2024!)
 - Londiste (maybe not in 2024...)
- Procedure
 - Setup new database cluster
 - Setup replication from old one to a new one
 - Perform failover

Logical replication

- a desired method for many, because *promises* zero downtime
- Could be very complex because of schema design
- Could require more downtime or service degradation than pg_upgrade

Conclusion

Method	downtime	extra disk space	complexity	riskiness
dump/restore	high	double	low	low
pg_upgrade (copy)	high	double	high	low
pg_upgrade (link)	low	low	high	very high
Logical replication	low	double	high	low
Slony-I	low	double	medium	low
Londiste	low	double	medium	low

Reading list

- http://momjian.us/main/writings/pgsql/pg_upgrade.pdf
- <https://blog.2ndquadrant.com/untangling-the-postgresql-upgrade/>
- <http://blog.endpoint.com/2016/12/postgres-statistics-and-pain-of-analyze.html>
- <https://www.depesz.com/2016/11/08/major-version-upgrading-with-minimal-downtime/>

Questions?

ik@dataegret.com

