Why you need to upgrade

Jimmy Angelakos
Postgres London 2022-06-24
We’ll be looking at:

• The PostgreSQL development cycle
• Why you need to upgrade regularly
• Minor – Major version upgrades
• What works great or better than other DBs
• Pain points & suggested improvements
Which Postgres version are you on?
NEVER UPGRADE. NEVER SURRENDER!
Fear of upgrading

• “It works fine now”
  • What about tomorrow?

• Afraid it’s going to cause problems
  • Upgrade may contain a bug
  • Incompatibility with application
  • “Don’t touch it, you might break it” – how well do you know your system?

• Change process laborious
  • Change request / approval process
  • Downtime / co-ordination with other departments

Result: Ignore the new version
The PostgreSQL development cycle
PostgreSQL development

Postgres has over 30 years of active development

• Strong reputation for:
  • Reliability & data integrity
  • Proven architecture & feature robustness
  • Extensibility
  • Dedication of the open source community

• Open source development model
  • Non-trivial changes are discussed (pgsql-hackers)
  • Submit patch, tests & documentation
  • Add to commitfest
  • Review & discussion
Minor – Major releases

Minor releases

• At least one minor release every quarter
• Versioning: last part i.e. 14.4
• Bugfixes
• Security issues
• Data corruption issues
• No internal format changes, no new features
• The community considers not upgrading to be riskier than upgrading

Major releases

• New major version about once a year
• Versioning: first part (post-PG10) i.e. 14.4
• New features (thoroughly vetted)
• Usually change internal format of system tables, data files
• No backward compatibility of stored data
• Supported for 5 years after initial release
• What keeps Postgres moving forward
Backward compatibility

Why something that was written for PG 9.3 will work with PG 15*

*Usually!
  - SQL is SQL
  - Upgrades are reliable

- User-visible changes listed in release notes
  - Section “Migration”
  - May require application changes (rare)

- Can upgrade from one major version to another w/o upgrading to intervening versions
- Read the major release notes of all intervening versions
  - https://www.postgresql.org/docs/current/runtime-config-compatible.html
Why you need to upgrade regularly
Upgrade regularly

Why?

• Benefits of open source
  • Unmatched ability to issue updates rapidly
  • For bugs and security vulnerabilities

• Bugfixes keep streaming in
  • Recently introduced regressions
  • Long-standing bugs undetected for years
  • Contributed by a wide range of people
    • Professionals paid to enhance Postgres
    • Individual contributors spotting an issue

• Security updates eliminate threats
  • Known to roll out in a matter of hours
  • Unheard of in proprietary software
False sense of stability

“A tested and monitored system is stable”

• Reality
  • According to our insights
  • To the best of our ability
  • Not objectively

• Software considered to be extremely stable
  • Discovery of latent bugs
  • Triggering of unexpected behaviours in software
  • New security vulnerabilities

• Don’t forget about extensions!
  • They get bugfixes too
  • But they have their own release schedules
Fear of upgrading

Associated with:

• No zero-downtime procedure defined
  • Or even upgrade procedure defined (!)

• Reluctance to schedule maintenance windows
  • Organisational issues

• Too much faith in our own tests
  • Not enough in others'

• Cumbersome deployment procedure
  • Discourages from deploying often

• Lack of QA system for testing upgrades frequently
FOMO

Fear Of Missing Out

• Missing out on newer features

• Stayed on PG 9.6, didn’t get:
  • Native partitioning
  • Logical replication

• Stayed on PG11, didn’t get:
  • pg_checksums
  • Generated columns

• Stayed on PG13, didn’t get:
  • Throughput improvement for large numbers of connections
  • Streaming of large transactions
  • libpq pipelining
Minor – Major version upgrades
Minor release upgrades

What they involve

• Always recommended to run latest available minor release for major version used
  • No new features / deprecations
  • Only necessary bugfixes / security issues
  • Minimal risk

• Upgrading to a minor release does not touch stored data files

• Quick and easy to perform

• For some releases, manual changes may be required to complete the upgrade
  • Always read the release notes before upgrading
Minor release upgrades

What to test

• Normally no need to re-test application
• Read the fine release notes!
  • PG 14.4 index corruption issue
  • Needed REINDEX for some cases
• PG 14.4 security vuln fix
  • Could affect \texttt{pg\_trgm} extension users
Minor release upgrades

How to upgrade

• No dump / restore needed
• (Minimal) downtime required
• Stop Postgres server
• Install new binaries
  • Usually via distribution’s package manager
• Restart Postgres
Major version upgrades

What they involve

• More complex!
  • Risk to application availability, behaviour
• Changes to internal system tables
  • pg_catalog
• Changes to data files format
  • Not binary compatible
• Don’t upgrade to a release that is not current
  • Unless there are compatibility issues with the application that can’t be addressed yet
• Read! https://www.postgresql.org/docs/current/upgrading.html
Major version upgrades

What to test

• Extensions used
  • For compatibility with the new major version

• Deprecated / removed features
  • If your application depends on them

• Beware of changes to pg_catalog
  • information_schema however is stable (SQL standard)
  • Mostly used by database management tools

• How long the upgrade will take to complete
  • pg_upgrade is highly dependent on number of DB objects
  • For pg_dump(all) factor in the data copy time

• Set up application environments with old and new PG versions
Major version upgrades

**pg_dumpall**

- Dump / reload
- Quiesce writes (for obvious reasons)
- `pg_dumpall > dumpfile` from old version
- `psql -d postgres -f dumpfile` in new version
- Upgrade via logical backup
- Needs disk space
- Takes time to export / import data
Major version upgrades

`pg_upgrade`

- In-place upgrade
- Stop Postgres server
- `pg_upgrade -b oldbindir -d oldconfigdir -D newconfigdir`
- Has `--check` option to perform a rudimentary dry run
- Can be performed in minutes or even seconds when using `--link`
- `--jobs` option for number of parallel jobs
Major version upgrades

Logical replication upgrades

• Replicate from old version directly to new version server
  • If switchover carefully orchestrated with pgbouncer, etc.

• Near-zero downtime upgrade

• No binary compatibility needed

• Can be performed with “native” LR or external solutions
  • pglogical, Slony-I, Londiste, Bucardo, ...
What works great or better than other DBs
What works great

Or better than other DBs

• Minor release upgrades are extremely reliable
• So are `pg_upgrades` (provided your system doesn’t have much weirdness)
• `--link` or `--clone` can be significantly faster than other DBs’ upgrade methods
• Full backward compatibility is supported everywhere
  • Clients, tools, and libraries can always connect to a previous version with no problems
  • Sometimes even forward compatibility is possible (but taking care to avoid new features)
• Logical replication gives an alternate upgrade path, and can be near-zero downtime
• Oracle upgrades also change internal table structure
• Oracle upgradability is between arbitrary close versions, and skipping is strongly discouraged
• MySQL upgrades cannot skip versions, upgrading from development (non-GA) versions not possible
• Mongo, MySQL, Oracle and other DBs suffer from driver compatibility issues between versions
Pain points and suggested improvements
## Pain points

### pg_dumpall
- Need to recreate all indexes
- Need to **ANALYZE** all tables

### pg_upgrade
- Loses optimizer statistics so you need to **ANALYZE** tables
- Breaks all logical replication slots so you need to orchestrate and recreate
- Doesn’t always detect all errors with **-- check** or verify that all extensions are upgradable

### Logical replication upgrades
- **walsender** is single threaded (consider heavy traffic DB)
- Apply process also single threaded

### Other general issues
- Upgrading systems which have physical streaming replicas
- Cluster awareness (in a HA context)
Suggested improvements

- Upgrade tool warns you about incompatibilities
- Upgrade tool explains the steps required for minor release upgrade
  - Or even takes you automatically through them
- Upgrade tool gives you estimated time and disk space required
- True zero-downtime upgrades not possible with vanilla Postgres
- Avoid wait for flushing buffers at checkpoint to perform server restart?
- Keep queries running during upgrade?
- Cluster awareness and information tooling?
Acknowledgements

Special thanks to:

- The PostgreSQL community panel on Upgradability at Postgres Vision 2022 conference, for accepting my invitation to come discuss the issue of upgrading PG publicly
  - Moderator: Bruce Momjian (VP, PG Evangelist, EDB)
  - Nikolay Samokhvalov (Founder, Postgres.ai - Database Lab)
  - Lætitia Avrot (Field CTO, EDB)
  - Andreas Scherbaum (Head of Databases, Adjust GmbH)

- Go watch their panel discussion!
  https://youtu.be/kgO_ms0o22E
Finally...

- Keep your Postgres updated!
- New features and improvements make it worth it
- Upgrades are reliable and mostly straightforward
- The alternative is worse!
- Postgres needs improved tooling, that’s easier to use
- Don’t forget to upgrade your OS as well
Thank you!

Find me on Twitter: @vyruss

- Keep your Postgres updated!
- New features and improvements make it worth it
- Upgrades are reliable and mostly straightforward
- The alternative is worse!
- Postgres needs improved tooling, that’s easier to use
- Don’t forget to upgrade your OS as well

Photo: Sango Bay, NW Highlands, Scotland