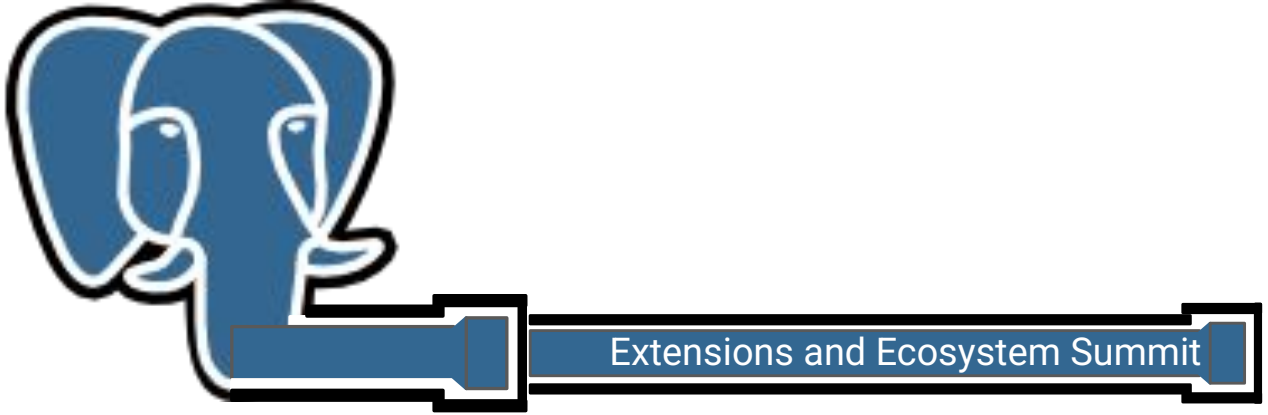

Extensions and upgrades

Extensions and Ecosystem Summit @ PGConf.dev 2026

You are Here

- How immutability challenges extension packaging and distribution
- Extending Authorisation and Authentication
- Extensions and upgrades
- Hooks and APIs: What to expose in the core
- To extend, to contribute, or to petition to core?
- Extensions summit readout

Upgrades

- Extensions support upgrades
- `ALTER EXTENSION UPDATE`
- How to cover all possible versions?
- Upgrade file proliferation
- Code duplication
- Paths to downgrades too?

Shared Library Challenges

- New version changes function signature
- Binary installed
- Breaks existing function till ALTER EXTENSION UPDATE
- Per-version C functions difficult to maintain
- Even harder with shared_preload_libraries

Why does it hurt when I do this?

- How to cover all relevant versions?
  - Currently supported by PGDG?
- Upgrade file proliferation
  - Install n-lots and roll forward?
- Forcing reinstalls can have dependency problems for extensions with types

Workarounds

- pgAudit:
  - Minor releases have scripts
  - Major releases force reinstall

- PostGIS:
  - Minor release provide alt C functions
  - Major releases require reinstall

Other issues

- catalog tables fixed once created
- upgrade and dump/restore go through unintuitively different paths
- poor visibility into/handling of version information

Other workarounds/compensations considered

- Stop (lock) the world for extension upgrades
  - Maybe only at session level?
- Make catalog aware of extension versions
- Provide macros to handle variable args
- Abstract the shared objects behind a “module”

For open discussion

- What other challenges are there?
- What's the ideal?
- What should change?
- What should not?
- What changes to propose?
- How do we work toward the ideal?

Notes will be at:

https://wiki.postgresql.org/wiki/PGConf.dev_2026_Extension_Summit