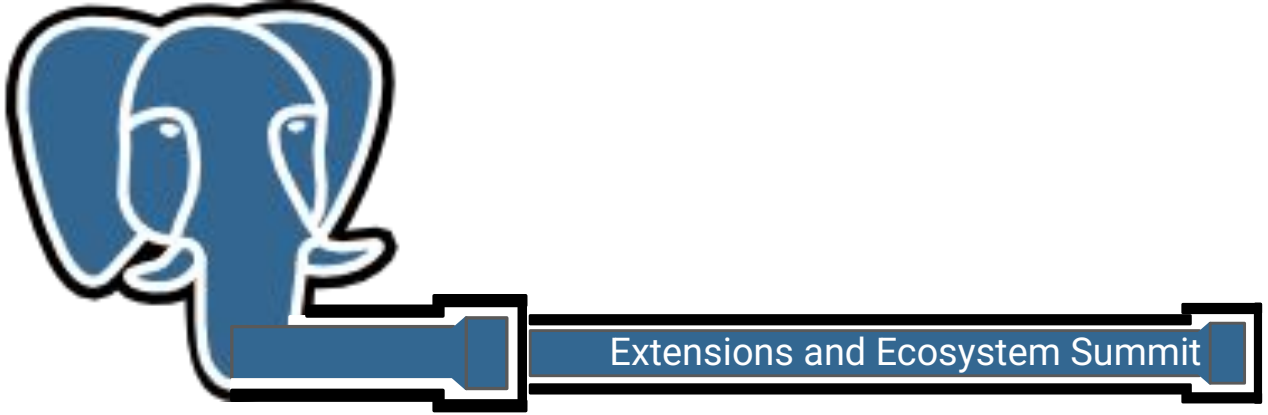

How immutability challenges extension packaging and distribution

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

Immutable Infrastructure

- Key concept of cloud native computing
- Services are replaced, not changed
- Great for stateless services
- Challenge configurable services
- Relevant example: Extension deployment

Extensions in OCI images

- OCI Images immutable
- Extensions baked into image
- How to add an extension?
- Bake new image & restart
- Doesn’t scale for cloud services
- New image for every customer + extension?
- Combinatoric explosion

CNPG and extensions

- CloudNativePG containers immutable
- Extensions must be baked in
- Postgres 18 adds extension_control_path
- Kubernetes 1.36 image volumes
- CNPG 1.26: immutable container extensions

Extension deployment with CNPG and image volumes

- Triggers rolling update, replicas first
- Mounts each extension as read-only volume

  `/extensions/pgvector`
- Updates GUCs

  extension_control_path = '$system:/extensions/pgvector/share'

  dynamic_library_path = '$libdir:/extensions/pgvector/lib'

This works, but…

- Volumes resolved at pod startup
- Updating GUCs requires restart
- Cannot use single prefix for all image volumes
- Each operates like full server prefix

Configuration sprawl from multiple extensions

```yaml
extensions:
  - name: pgvector
    image:
      reference: ghcr.io/cloudnative-pg/pgvector-18-testing:latest
  - name: semver
    image:
      reference: ghcr.io/example/semver:0.40.0
  - name: auto_explain
    image:
      reference: ghcr.io/example/auto_explain:18
  - name: bloom
    image:
      reference: ghcr.io/example/bloom:18
  - name: postgis
    image:
      reference: ghcr.io/example/postgis:18
```

Config sprawl from multiple extensions

```
extension_control_path =
'$system:/extensions/pgvector/share:/extensions/se
mver/share:/extensions/auto_explain/share:/extensi
ons/bloom/share:/extensions/postgis/share'

dynamic_library_path    =
'$libdir:/extensions/pgvector/lib:/extensions/semv
er/lib:/extensions/auto_explain/lib:/extensions/bl
oom/lib:/extensions/postgis/lib'
```

What would the ideal look like?

- Add extensions by patching manifest
- Avoid rolling restarts
  - Requires ImageVolume without pod start
- Avoid updating GUCs
  - Requires different file layout
- `shared_preload_libraries` will create exceptions
- Would also simplify non-CNPG installs

For open discussion

- 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