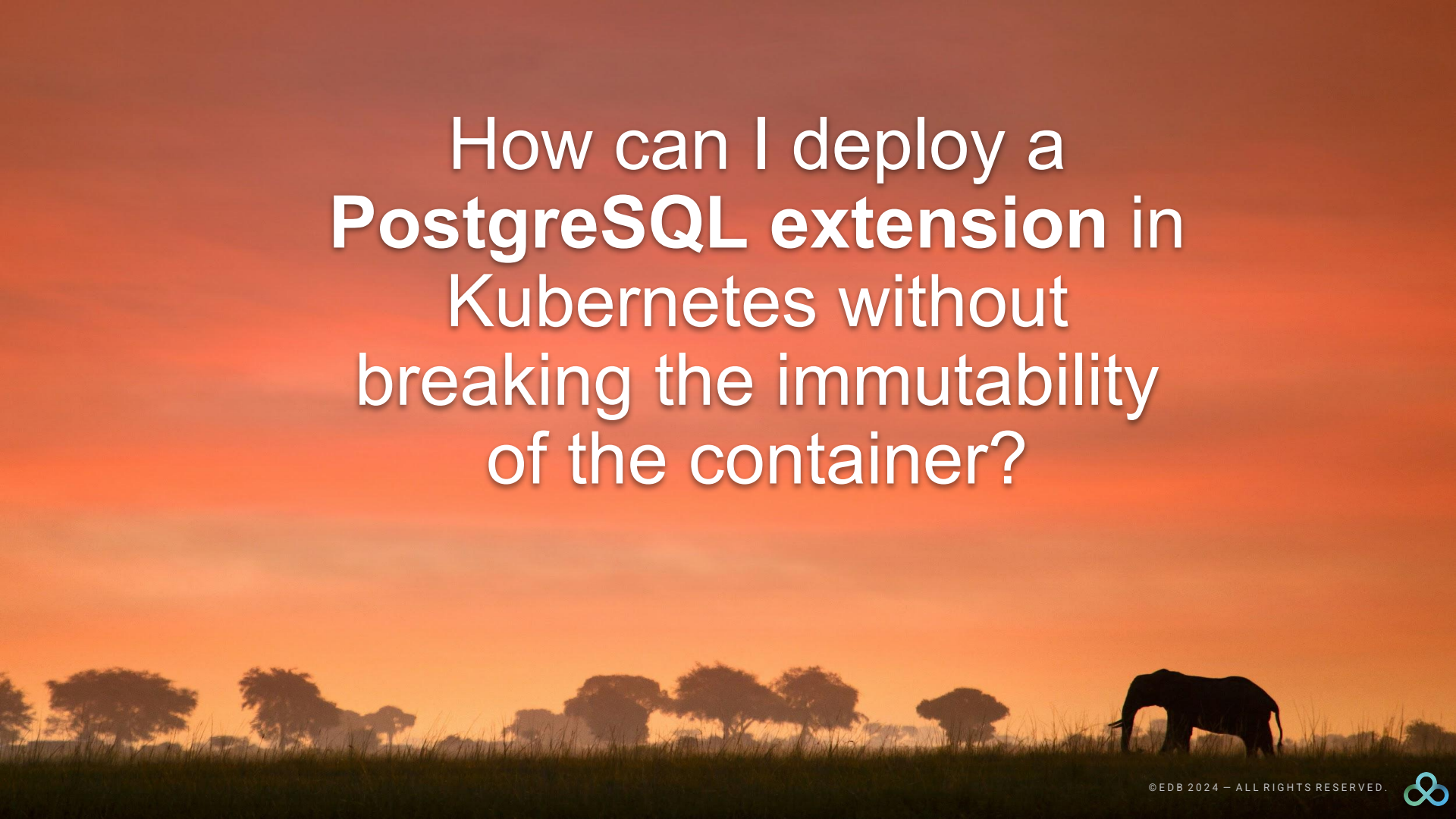


EDB
Postgres® for the AI Generation

Improving the PostgreSQL Extensions Experience in Kubernetes with CloudNativePG

Postgres Extensions Ecosystem Virtual Mini-Summits

Wednesday 7 May, 2025

Gabriele Bartolini
VP, Chief Architect, Kubernetes at EDB

Gabriele Bartolini

VP, Chief Architect of Kubernetes at EDB

PostgreSQL user since ~2000

Ex 2ndQuadrant (co-founder)

PostgreSQL Contributor

DoK Ambassador

DevOps evangelist

Open source contributor

- Barman (2011)
- CloudNativePG (2022)

Blog: gabrielebartolini.it @_GBartolini_

#1 Contributors to 

Creators of  CloudNativePG

enterprisedb.com

cloudnative-pg.io

postgresql.org

EDB
Postgres® for the AI Generation

Agenda

- Setting the context
- Immutable Application Containers
- PostgreSQL Extensions in Kubernetes with CNPG
- The (immutable) future
- Conclusions

EDB
Postgres® for the AI Generation

Setting the context

What is Kubernetes?

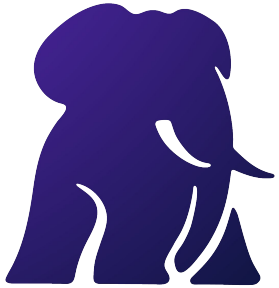
Greek for helmsman. Also known as **k8s**. URL: kubernetes.io

CLOUD NATIVE
COMPUTING FOUNDATION

- Orchestration system for containers
- Automates deployment, administration and scaling of:
  - **infrastructure**
  - **cloud native applications** - also known as **workloads**
    - In Kubernetes, databases are workloads
- Open Source (Apache Licence 2.0) since 2014
  - v1.0 released in 2015
- Owned by the Cloud Native Computing Foundation (CNCF, cncf.io)
  - First project within the CNCF to become “Graduated”

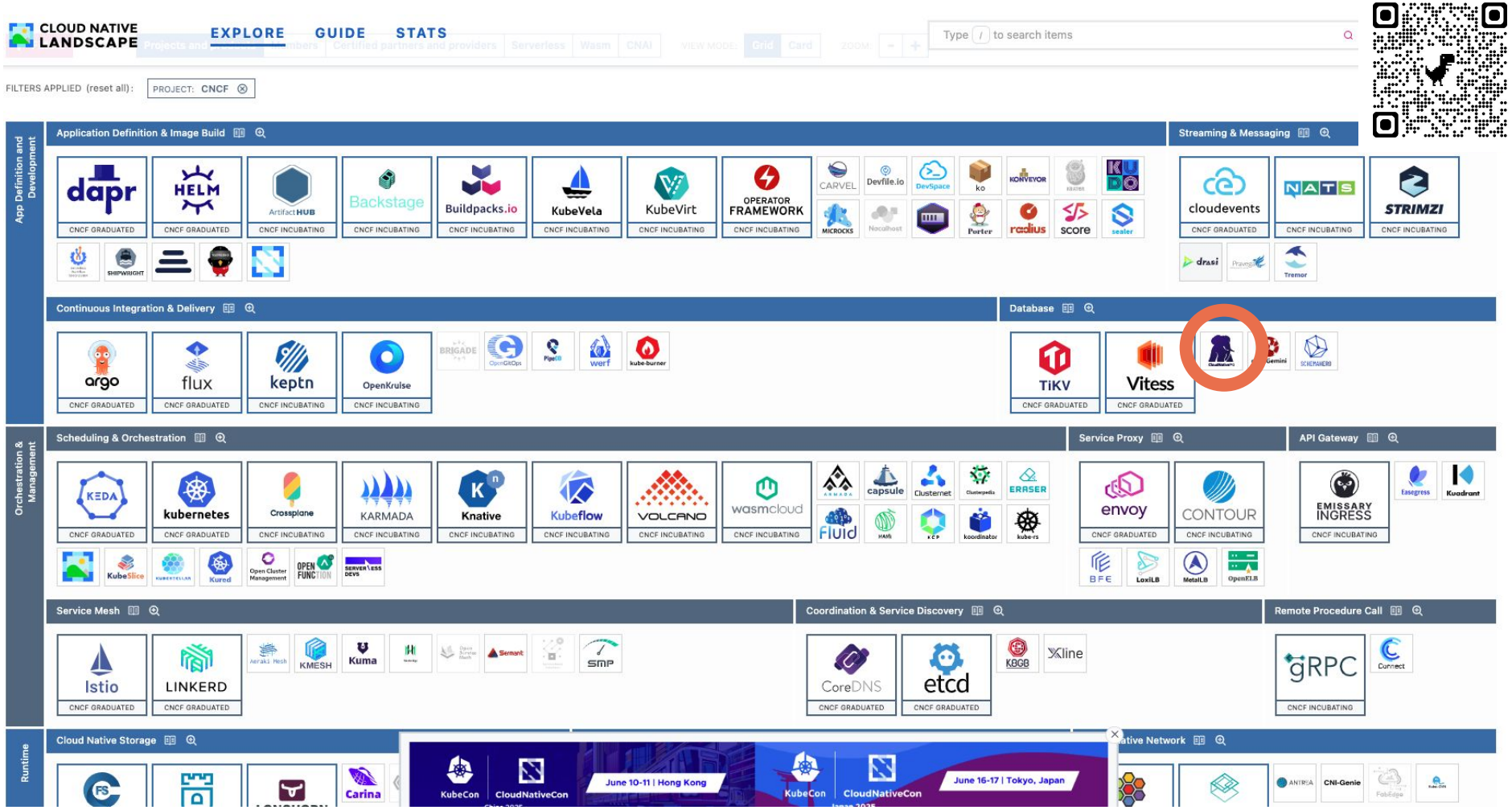
What is CloudNativePG?

- “Level 5”, Production ready
  - IBM Cloud Pak, Google Cloud, Azure, Akamai, Tembo, …
- **CNCF Project (Sandbox)**
  - Apache License 2.0
  - Owned by the Community (CNCF)
  - Vendor neutral, openly governed, **always free**
  - Originally created by EDB (2019 by 2ndQuadrant)
- Multiple installation methods:
  - K8s manifests
  - Helm chart
  - OperatorHub.io (OLM)
- 4.0k commits
- 5.9k stars on Github (#1 Postgres operator)
- **~78M** downloads on Github (~3 years)



CloudNativePG

github.com/cloudnative-pg



https://landscape.cncf.io

What does cloud native means to us?

The three pillars of cloud native based on our experience since 2019

- **People and a “DevOps culture”**
  - Immutable Infrastructure
  - Infrastructure abstraction
  - Version Control
  - Declarative Configuration
  - Shift left on Security
  - Continuous Delivery
  - …
- **Immutable Application Containers**
- **Kubernetes** as the standard way to orchestrate them through IaC

EDB
Postgres® for the AI Generation

Immutable Application Containers

Mutable infrastructure

Prevalent mindset

- **Main artifacts: Packages**
- Long term infrastructure
- Changes happen as incremental updates on an existing infrastructure
  - `dnf update` or `apt update`
- Current state is represented by changes over time:
  - configuration management and operations
- Systems are healed after failures (*pets*)

Immutable infrastructure

Emerging mindset from DevOps and Platform Engineering

- **Main artifacts: OCI Container Images**
- Founded on Continuous Delivery (GitOps)
- Releasing a new version of an application means:
  - automatically build a new container image
  - publish it on a public/private registry
- Changes happen in atomic operations:
  - the new version image is pulled from the registry
  - the existing image is replaced “almost” instantaneously

Kubernetes: mutable or immutable?

Immutability requires a mindshift

- **There are no incremental updates**
- Data volumes are attached to containers (**stateful**)
- Containers are not changed (**immutable**)
  - A single change to the content, requires a new version of the image
- **Kubernetes works with immutable containers**
- **Security and Change Management Policies require this approach**
  - At any time in Kubernetes you know what software is running in your infrastructure, and it is in a version control system
  - Very important to control CVEs

The lifecycle of a container image

Simplified view

Immutable Application Containers

From day 1 of CloudNativePG we have placed this idea at the core

- The content of a running container must not change
  - File system is read-only to prevent any unauthorized change
    - “apt update” or “dnf update” are not possible
- Only a single application is running
  - PID 1 or entrypoint
- CloudNativePG and Immutable Application Containers:
  - Database files are in separate persistent volumes
    - PGDATA and optionally WAL files and tablespaces
  - Orchestrates “operand images” containing PostgreSQL binaries
    - The “instance manager” is the PID 1

postgres-containers

Public

Edit Pins | Unwatch 6 | Fork 42 | Starred 77

main | 6 Branches | 0 Tags | Go to file t | Add file | Code

dependabot[bot] Bump sigstore/cosign-installer from 3.8.1 to 3.8.2 (#182) 1a8f19f · 4 hours ago 1,038 Commits

.github	Bump sigstore/cosign-installer from 3.8.1 to 3.8.2 (#182)	4 hours ago
Debian	Automatic ClusterImageCatalog update	3 days ago
lib	chore: update barman-cloud to version 3.13.3 (#177)	2 days ago
.gitignore	chore: add PostgreSQL images 15beta (#18)	3 years ago
BUILD.md	feat: add cosign to sign the images (#137)	4 months ago
CODEOWNERS	Initial Commit	3 years ago
CODE_OF_CONDUCT.md	Initial Commit	3 years ago
Dockerfile	fix: Dockerfile to reduce vulnerabilities (#167)	3 weeks ago
GOVERNANCE.md	Initial Commit	3 years ago
LICENSE	Initial Commit	3 years ago
README.md	feat: add cosign to sign the images (#137)	4 months ago
docker-bake.hcl	chore(deps): update debian base images (#174)	last week
renovate.json	Enable pinGitHubActionDigests for renovatebot (#165)	last week

README | Code of conduct | Apache-2.0 license

About

Operand images for CloudNativePG containing all community supported version PostgreSQL

postgres postgresql container-images cloudnativepg

Readme
Apache-2.0 license
Code of conduct
Activity
Custom properties
77 stars
6 watching
42 forks
Report repository

Releases

No releases published
Create a new release

Packages 2

postgresql
postgresql-testing

Mmhh …
What about PostgreSQL extensions?

EDB
Postgres® for the AI Generation

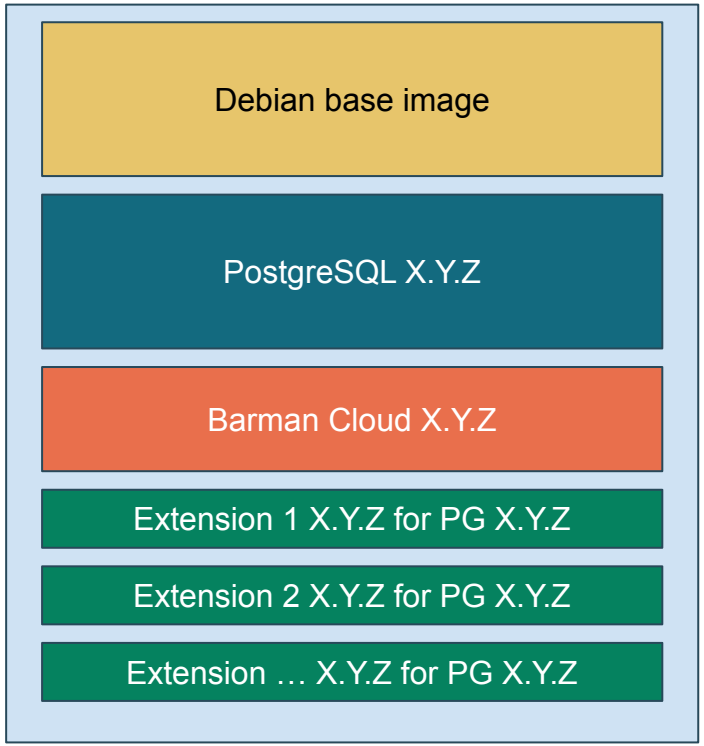
PostgreSQL Extensions in Kubernetes with CloudNativePG

Possible scenarios for extensions

- Place them on-demand in a mutable volume
  - We immediately discarded that as it breaks immutability
- **Place them in the operand image**
  - Extensions must be included at build time in the operand image
  - A few approaches:
    - The “Spilo” way (images containing “the lot”)
    - **Create a few different flavours**
    - **Define a way to build custom images**
      - Container image requirements

PostgreSQL Operand Images now

How we distribute PostgreSQL container images in CloudNativePG



Every additional layer makes the image heavier

*Images are normally rebuilt every week in case some element in the stack changes **

Extensions are normally installed through packages

The problem with extensions

Remember: we cannot break immutability

- “Can I use extension WHATEVER in my CloudNativePG cluster?”
  - We add the extension to the container image
  - We could create a new flavor (e.g. PostGIS, TimeScale, …)
  - Users ultimately need to build their own images
- Very large images!
  - System dependencies accentuate this problem
  - Increase the risk of CVEs being present in the system
    - Require more frequent updates (in terms of rebuild!)

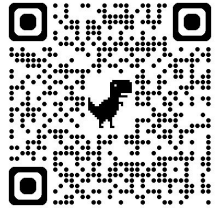
Technology limitations

Both for PostgreSQL and Kubernetes

- PostgreSQL extensions must be in a single folder
  - e.g. /usr/share/postgresql/17/extension
  - It is not possible to define multiple locations
    - **Until PostgreSQL 18 with “extension_control_path”!!!**
- Kubernetes doesn’t allow mount OCI artifacts as read-only volumes
  - Until Kubernetes 1.33 (released April 23, 2025)
    - Support for mounting images as volumes

Commit `4f7f7b0`

 **petere** and **mattheusv** committed 2 days ago · ✓ 7 / 10



```
extension_control_path

The new GUC extension_control_path specifies a path to look for
extension control files.  The default value is $system, which looks in
the compiled-in location, as before.

The path search uses the same code and works in the same way as
dynamic_library_path.

Some use cases of this are: (1) testing extensions during package
builds, (2) installing extensions outside security-restricted
containers like Python.app (on macOS), (3) adding extensions to
PostgreSQL running in a Kubernetes environment using operators such as
CloudNativePG without having to rebuild the base image for each new
extension.

There is also a tweak in Makefile.global so that it is possible to
install extensions using PGXS into an different directory than the
default, using 'make install prefix=/else/where'.  This previously
only worked when specifying the subdirectories, like 'make install
datadir=/else/where/share pkglibdir=/else/where/lib', for purely
implementation reasons.  (Of course, without the path feature,
installing elsewhere was rarely useful.)

Author: Peter Eisentraut <peter@eisentraut.org>
Co-authored-by: Matheus Alcantara <matheusssilv97@gmail.com>
Reviewed-by: David E. Wheeler <david@justatheory.com>
Reviewed-by: Gabriele Bartolini <gabriele.bartolini@enterprisedb.com>
Reviewed-by: Marco Nenciarini <marco.nenciarini@enterprisedb.com>
Reviewed-by: Niccolò Fei <niccolo.fei@enterprisedb.com>
Discussion: https://www.postgresql.org/message-id/flat/E7C7BFFB-8857-48D4-A71F-88B359FADCFD@justatheory.com
```

kubernetes / enhancements

Type / to search

Code | Issues 264 | Pull requests 108 | Actions | Projects | Security | Insights

VolumeSource: OCI Artifact and/or Image #4639

New issue

Open

sallyom opened on May 16, 2024 · edited by saschagrunert

Edits

Enhancement Description

- One-line enhancement description (can be used as a release note): VolumeSource: OCI Artifact or OCI Image
- Kubernetes Enhancement Proposal: https://github.com/kubernetes/enhancements/tree/master/keps/sig-node/4639-oci-volume-source
- Discussion Link:
- Primary contact (assignee): sgrunert@redhat.com, somalley@redhat.com
- Responsible SIGs: sig-node, sig-storage
- Enhancement target (which target equals to which milestone):
  - Alpha release target (x.y): 1.31
  - Beta release target (x.y): 1.33
  - Stable release target (x.y):

- [x] Alpha
  - [x] KEP (`k/enhancements`) update PR(s):
    - KEP-4639: adding OCI VolumeSource #4642
  - [x] Code (`k/k`) update PR(s):
    - [KEP-4639] Add OCI VolumeSource CRI API kubernetes#125659
      - [KEP-4639] Update CRI API and workflow #4751
    - [KEP-4639] Add `ImageVolumeSource` API kubernetes#125660
    - [KEP-4639] Add `ImageVolumeSource` implementation kubernetes#125663
    - **Nice to have:**
      - [KEP-4639] Mention that `fsGroupChangePolicy` has no effect kubernetes#126281
      - Fix runtime panic in imagevolume `CanSupport` method kubernetes#126323
      - [KEP-4639] Add `ImageVolumeSource` node e2e tests kubernetes#126220
  - [x] Out of tree update PR(s):
    - Add OCI Volume Source support to `crictl [create|run] --with-pull` kubernetes-sigs/cri-tools#1464

Assignees

sallyom

saschagrunert

Labels

kind/design | lead-opted-in | sig/node | sig/storage | stage/beta | tracked/yes

Type

No type

Projects

1.33 Enhancements Tracking

Status: Tracked for Docs Freeze

SIG Node 1.33 KEPs planning

Status: Implemented

3 closed projects

Milestone

v1.33

No due date

Relationships

None yet

EDB
Postgres® for the AI Generation

The (immutable) future

What's new in PostgreSQL 18

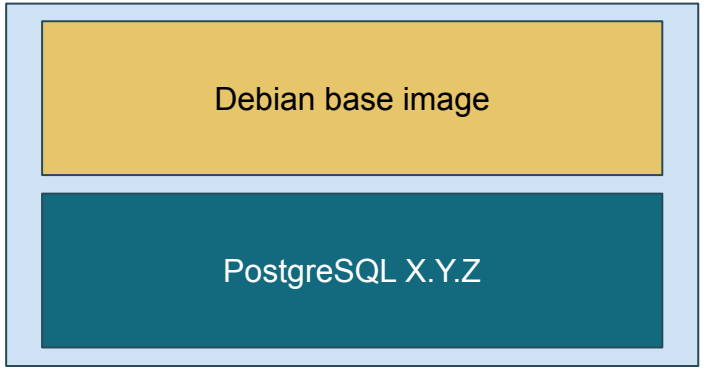
Introduce “extension_control_path”

- Path to search for extension control files (i.e. extensions)
- List of absolute directory paths
- Works with “dynamic_library_path” for additional libraries

PostgreSQL Operand Images in the future

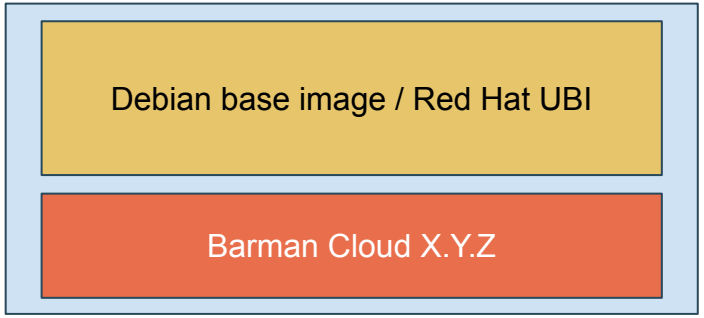
How we will distribute PostgreSQL and extensions in the future

PostgreSQL (minimal)

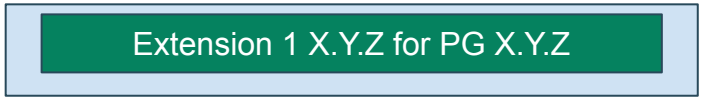


Every image will be independent and updated only when that software changes, or the underlying image (e.g. CVEs).

Barman Cloud Plugin sidecar



Any extension

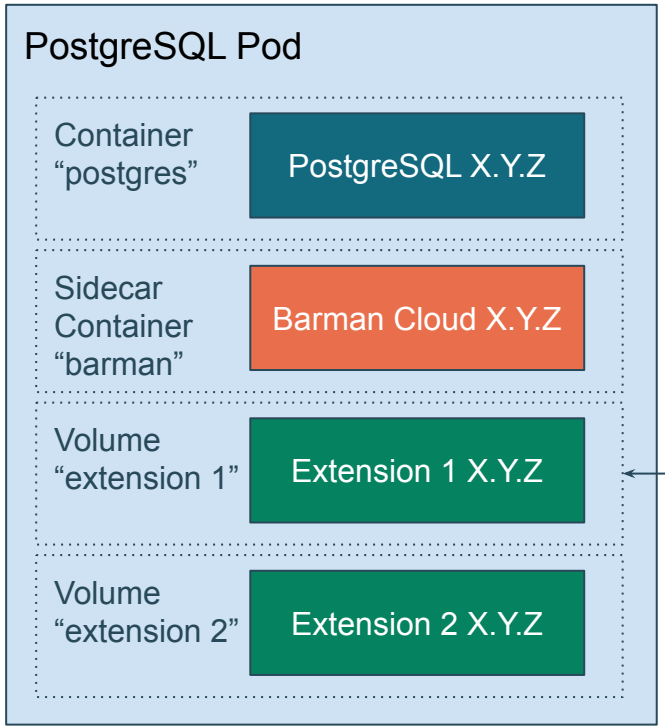


***Note**: no base image, just the actual files.*

Obviously they must be coherent with the OS of the main PostgreSQL image (e.g. if Debian, extension should be built from the Debian package content or compiled)

What will change at runtime (without breaking immutability)

Extensions can be added at any time, without having to prebuild images with all needed combinations



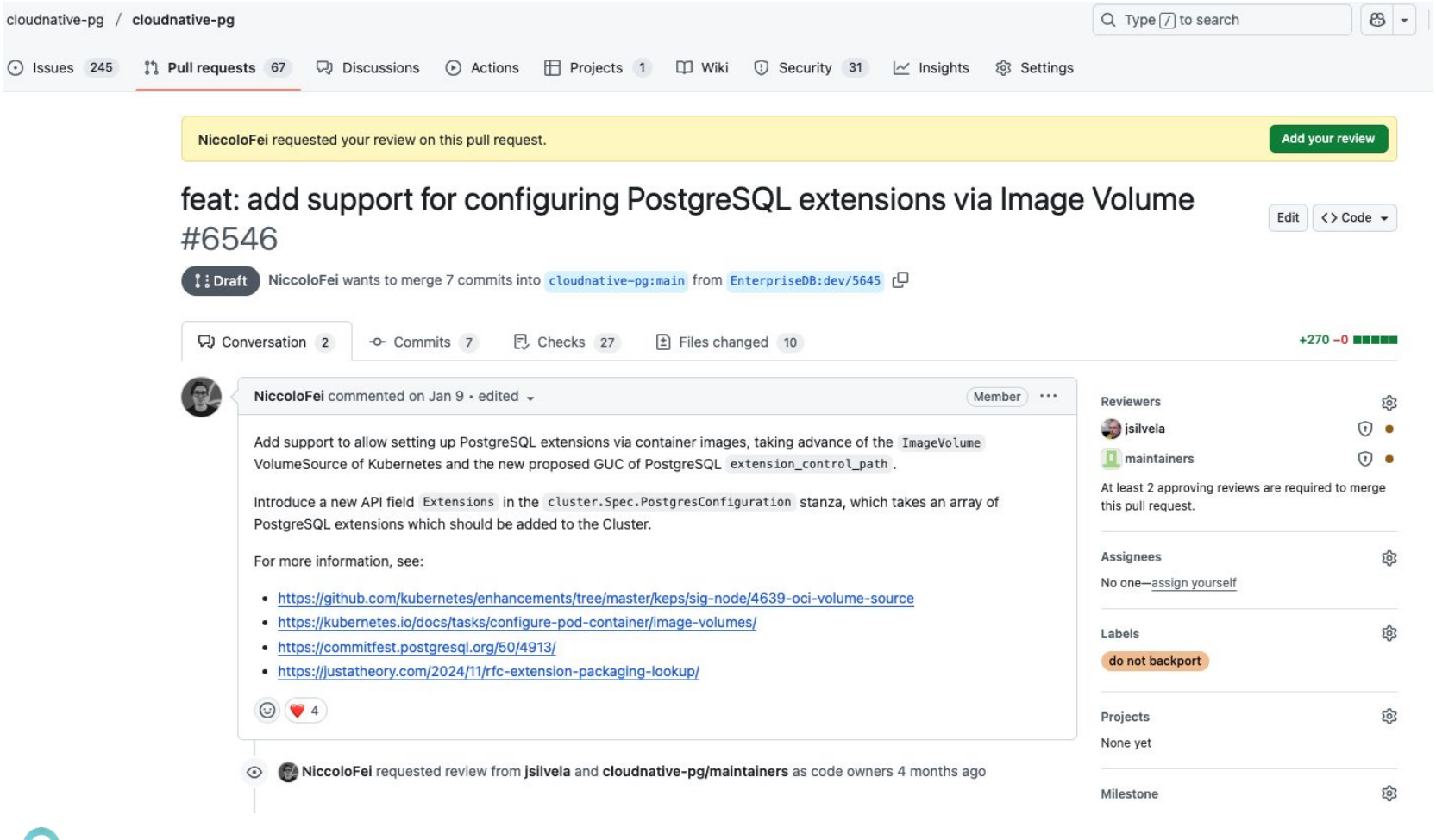
`extension_control_path` could list all the extension volumes loaded via Kubernetes VolumeSource

This could be `/extensions/pgvector/extension`

Fail-fast pilot project

Introduce extensions management in CloudNativePG

- Exploratory patch in the CloudNativePG operator:
  - Declarative management of extensions for a “Cluster” resource
    - Introduce the “.spec.postgresql.extensions” stanza
- PostgreSQL image with “extension_control_path”
  - Built from the “postgres-trunk-containers” project
- Bare container images for a few extensions, including:
  - pgvector
  - postgis
- *Many thanks to Niccolò Fei and Marco Nenciarini for working on the pilot*



Our goal for CloudNativePG 1.27

```
apiVersion: postgresql.cnpg.io/v1
kind: Cluster
metadata:
  name: postgresql-with-extensions
spec:
  instances: 1
  imageName: ghcr.io/cloudnative-pg/postgresql-trunk:18-devel
  postgresql:
    extensions:
      - name: pgvector
        image:
          reference: ghcr.io/cloudnative-pg/pgvector-18-testing:latest
  storage:
    storageClass: standard
    size: 1Gi
```

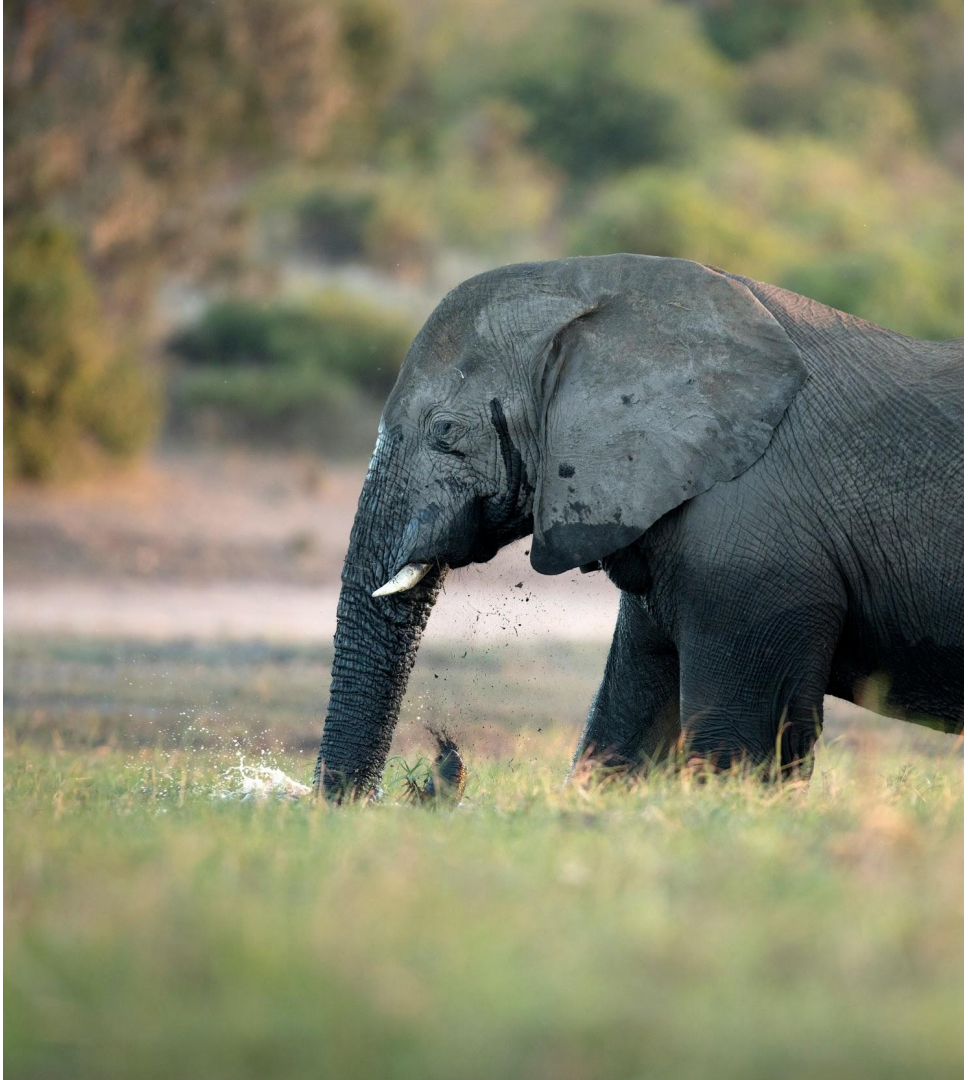
What will happen under the hood

- CNPG triggers a rolling update, starting with replicas
  - *Short downtime required*
- Each referenced image is mounted as a read-only volume via *ImageVolumes*
  - The *pgvector-18-testing:latest* image is mounted on */extensions/pgvector*.
- CNPG updates:
  - *dynamic_library_path* to include the */extensions/pgvector/lib*
  - *extension_control_path* to include */extensions/pgvector/share*

Container Images as first class artifacts

Container Images should become a standard distribution artifact for extensions, like packages

- Currently, extensions are primarily distributed as packages
- We should distribute them as OCI images
  - Epical change for PostgreSQL



EDB
Postgres® for the AI Generation

Conclusions

Suggested reading

The Immutable Future of PostgreSQL Extensions in Kubernetes with CloudNativePG

3 March 2025 · 8 mins

kubernetes k8s cloudnativepg cnpg postgresql postgres dok data on kubernetes extensions Container images sbom pgvector imagevolume extension_control_path

Key take aways

The future for PostgreSQL extensions in Kubernetes is immutable

- Container images should be seen as a standard artifact nowadays
  - Challenge the status-quo of Postgres extensions distribution
- An ecosystem based on immutable container images will bring:
  - Reduced operational complexity, easier testing and validation of extensions
  - Lighter and safer images, with less frequent need for updates and bandwidth usage
  - Full Independence for extension projects
    - Each extension will ideally build their own images and publish them
    - No need to centralize build of extensions and prebuild combinations
- Refrain from saying “This is a limitation of Kubernetes for Postgres”
  - We need to consider all the organisational benefits brought by the stack

Questions?

Creators of  **CloudNativePG**

gabrielebartolini.it @_GBartolini_

enterprisedb.com

cloudnative-pg.io

cloudnativepg@mastodon.social

cloudnativepg.bsky.social

linkedin.com/company/cloudnative-pg/