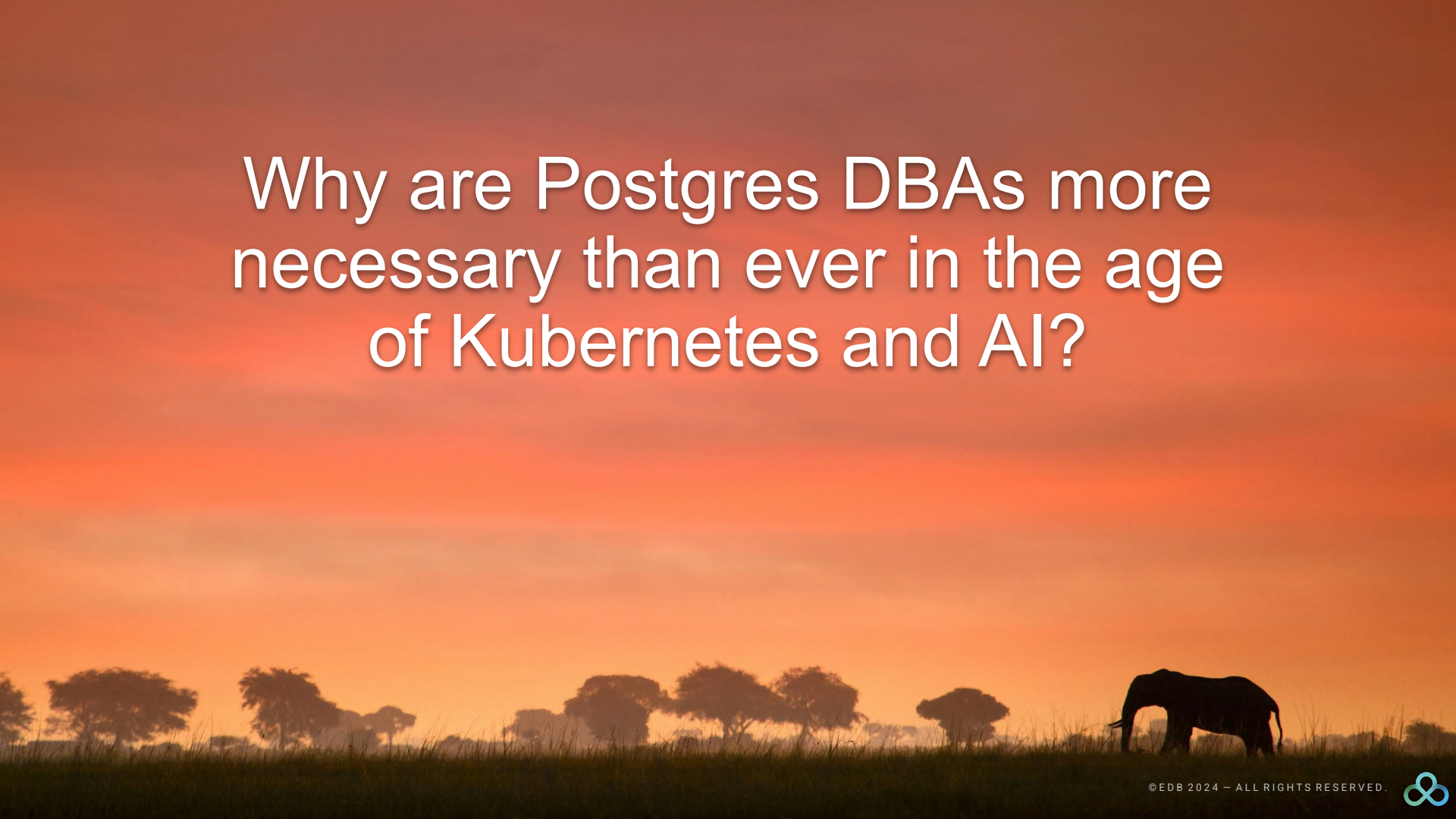
Why are Postgres DBAs more necessary than ever in the age of Kubernetes and AI?



How is Kubernetes the next logical step in your Postgres DBA career?

EDB
Postgres® for the AI Generation

Migrating PostgreSQL to Kubernetes

Gabriele Bartolini
VP, Chief Architect, Kubernetes at EDB
Cloud Native Days Italy 2025, Bologna, 24 June

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

- The Context
- The human side
- How the infrastructure changes
- How the database changes
- How the application changes
- Conclusions

EDB
Postgres® for the AI Generation

The Context

DISCLAIMER

There are several open-source operators for Postgres. As a founder and maintainer, I will cover only CloudNativePG today.

Kubernetes vs Virtual Machines

Kubernetes

Infrastructure and containerised apps

Containers share the same OS

Immutable Application Containers

Portable on any cloud, standard

Day 2 operations on applications

VMs

Operating systems

VMs have their own OS

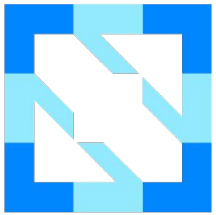
Mutable (“dnf update”)

Indirectly portable (Day 1, Terraform)

No Day 2 operations on applications

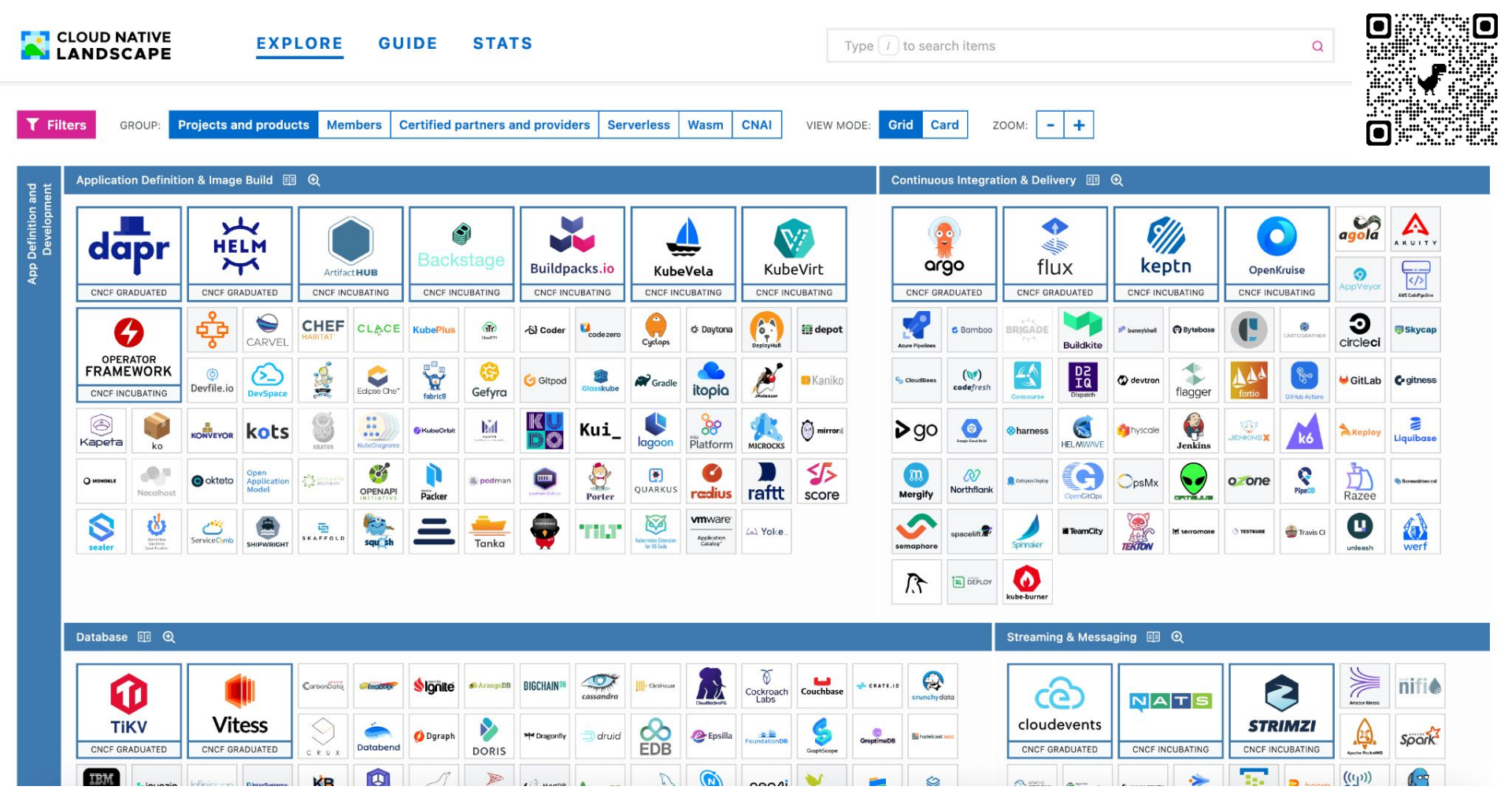
Kubernetes is complex by nature. However, its modular design shifts the **cognitive load** from **individuals** to **teams**.

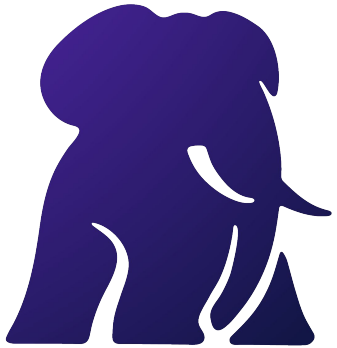
What is the CNCF?



CLOUD NATIVE
COMPUTING FOUNDATION

cncf.io


CLOUD NATIVE LANDSCAPE
EXPLORE
GUIDE
STATS
Type / to search items
Filters
GROUP:
Projects and products
Members
Certified partners and providers
Serverless
Wasm
CNAI
VIEW MODE:
Grid
Card
ZOOM:
App Definition and Development
Application Definition & Image Build
Continuous Integration & Delivery
Database
Streaming & Messaging
dapr
HELM
Artifact HUB
Backstage
Buildpacks.io
KubeVela
KubeVirt
argo
flux
keptn
OpenKruise
agola
AKUITY
AppVeyor
OPERATOR FRAMEWORK
CHEF
CARVEL
CLACE
KubePlus
Coder
codezero
Cyclops
Daytona
DeployHub
depot
Devfile.io
DevSpace
Eclipse Che
fabric8
Gefyra
Gitpod
Glasskube
Gradle
itopia
Jelastic
Kaniko
Kapeta
ko
KONVEYOR
kots
KRATOR
KubeDiagrams
KubeOrbit
KUDO
Kui
lagoon
Platform
MICROCKS
mirror
MONOKLE
Nocalhost
okteto
Open Application Model
OPENAPI INITIATIVE
Packer
podman
Porter
QUARKUS
radius
rafttt
score
sealer
ServiceComb
SHIPWRIGHT
SKAFFOLD
squash
Tanka
TILT
vmware
Application Catalog
Yoke
Azure Pipelines
Bamboo
BRIGADE
Buildkite
bunnyshell
Bytebase
CARTOGRAPHER
circleci
Skycap
CloudBees
codefresh
Concourse
D2 IQ
Dispatch
devtron
flagger
fortio
GitHub Actions
GitLab
gitness
go
harness
HELM/WAVE
hyscale
Jenkins
JENKINS X
k6
Keploy
Liquibase
Mergify
Northflank
Octopus Deploy
OpenGitOps
OpsMx
ozone
PipeCD
Razee
Screwdriver.cd
semaphore
spaceliftt
Spinnaker
TeamCity
TEKTON
terramate
TESTKUBE
Travis CI
unleash
werf
DEPLOY
kube-burner
TiKV
Vitess
CarbonData
Ignite
ArangoDB
BIGCHAINDB
cassandra
ClickHouse
CloudNativePG
Cockroach Labs
Couchbase
CRATE.IO
crunchy data
CRUX
Databend
Dgraph
DORIS
Dragonfly
druid
EDB
Epsilla
FoundationDB
GraphScope
GreptimeDB
hazelcast
cloudevents
NATS
STRIMZI
nifi
Spark
©EDB 2024 — ALL RIGHTS RESERVED.
https://landscape.cncf.io



CloudNativePG

is a CNCF Sandbox Project

the first relational database to enter since 2018
the first ever for PostgreSQL

Key adopters

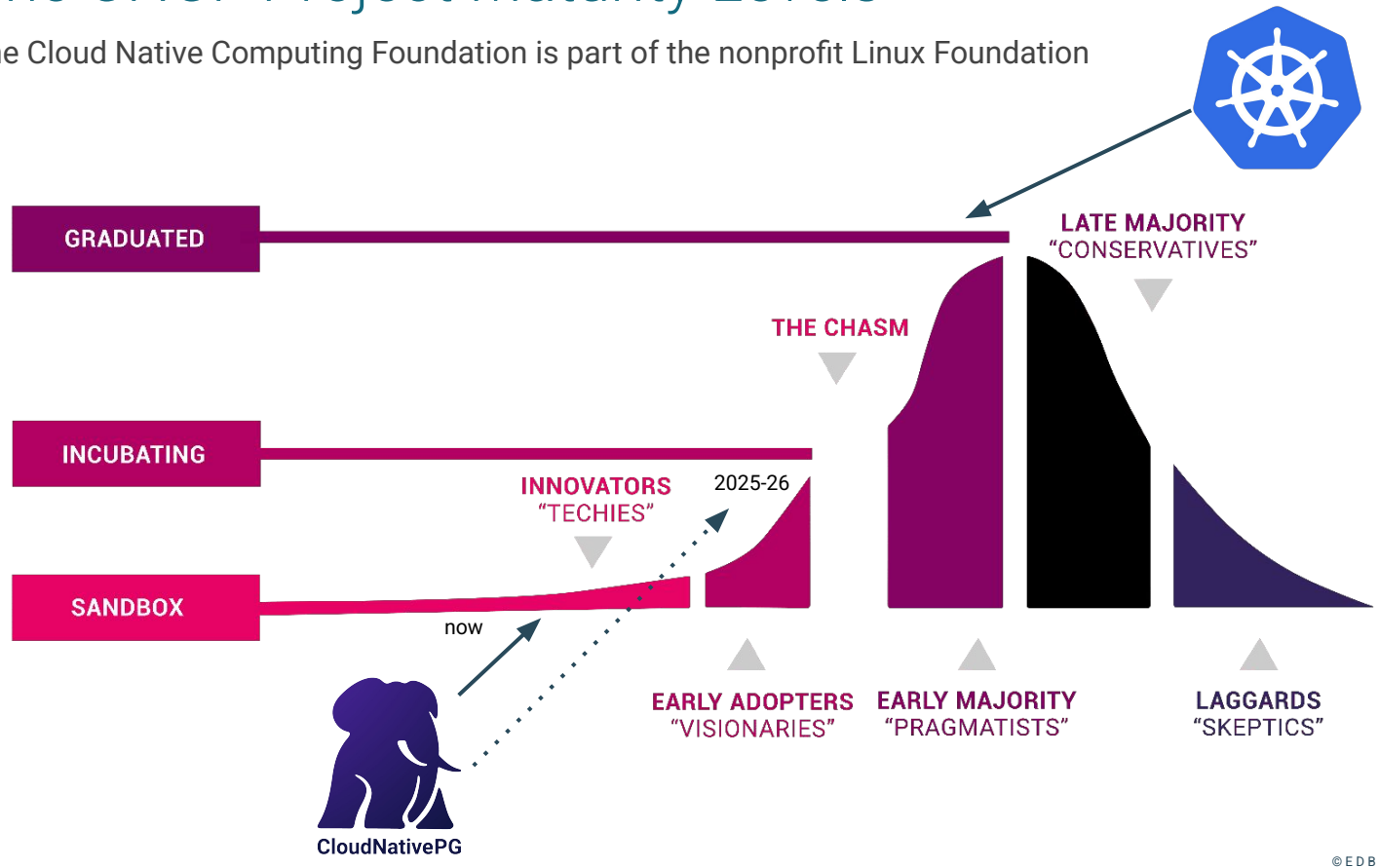
IBM Cloud Pak, Google Cloud, Azure, Bitnami, Akamai, Novo Nordisk, Hitachi, Belastingdienst

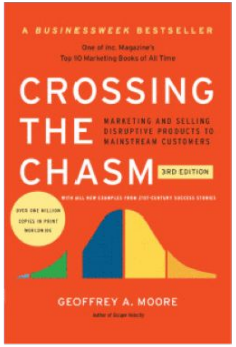
Target: CNCF Incubation in 2025-2026

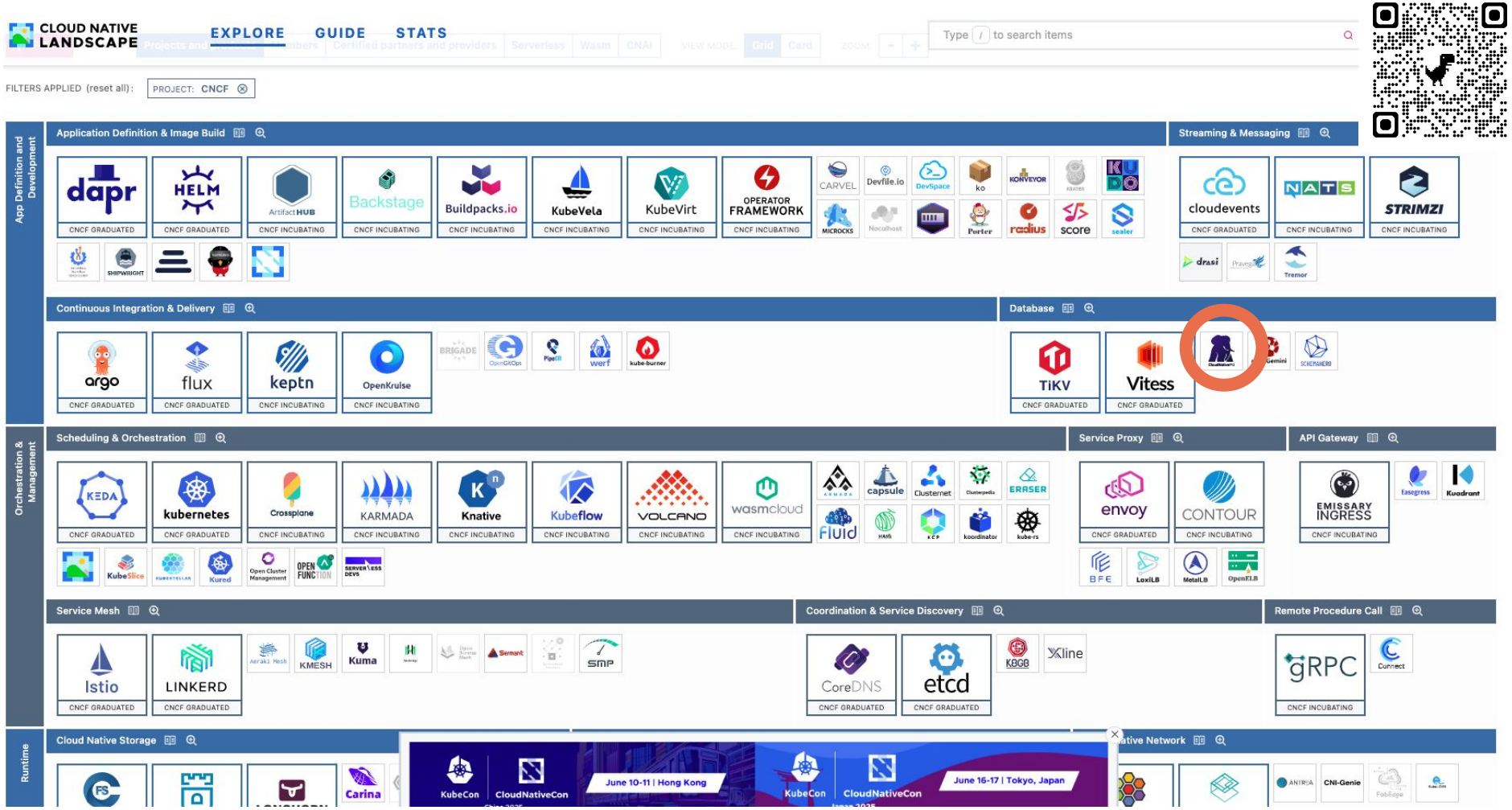
github.com/cloudnative-pg

The CNCF Project Maturity Levels

The Cloud Native Computing Foundation is part of the nonprofit Linux Foundation







https://landscape.cncf.io

Evolution of PostgreSQL in containers

From Docker system containers to Kubernetes native databases with CloudNativePG

- **2013/3**: Docker is released. Postgres runs mainly for testing in system containers
- **2015/7**: Kubernetes 1.0 is released. Stateless applications only.
- **2016/11**: Operator pattern by CoreOS
- **2017/3**: Crunchy Data releases the first Postgres operator based on Patroni
- **2017/12**: Statefulsets are introduced in Kubernetes 1.9 (*1 year after beta in 1.5*)
- **2018/8**: Zalando releases their operator
- **2019/4**: Local persistent volumes are introduced in Kubernetes 1.14
- **2019/8**: The Cloud Native initiative at EDB (2ndQuadrant at that time) begins
- **2021/2:** EDB launches Cloud Native Postgres
- **2022/5: EDB open sources CloudNativePG**
- **2024/10**: CloudNativePG reaches 4500 stars on GitHub (#1 Postgres operator)
- **2025/1**: **CloudNativePG** becomes a **CNCF** project entering the **Sandbox**

Kubernetes operators for PostgreSQL

Operator	Year	Technology	License	Copyright	Governance	Commits year avg	Stars
Crunchy PGO	2017	Based on **Patroni** and **Statefulsets**	Apache 2.0	Company	Company	~ 165	4.2k
Zalando	2018	Based on **Patroni** and **Statefulsets**	MIT	Company	Company	~ 100	4.8k
Stackgres	2020	Based on **Stolon** and **Statefulsets**	AGPL 3.0	Company	Company	~ 615	1.2k
Percona *(forked from Crunchy)*	2021	Based on **Patroni** and **Statefulsets**	Apache 2.0	Company	Company	~ 250	317
Kubegres	2021	Kubernetes **native** with **Statefulsets**	Apache 2.0	Company	Company	~ 10	1.3k
CloudNativePG	2022*	**Kubernetes native with PVC**	**Apache 2.0**	**CNCF**	**Vendor Neutral**	**~ 850**	**6.2k**
Cybertec *(forked from Zalando)*	2023	Based on **Patroni** and **Statefulsets**	Apache 2.0	Company	Company	~ 270	15

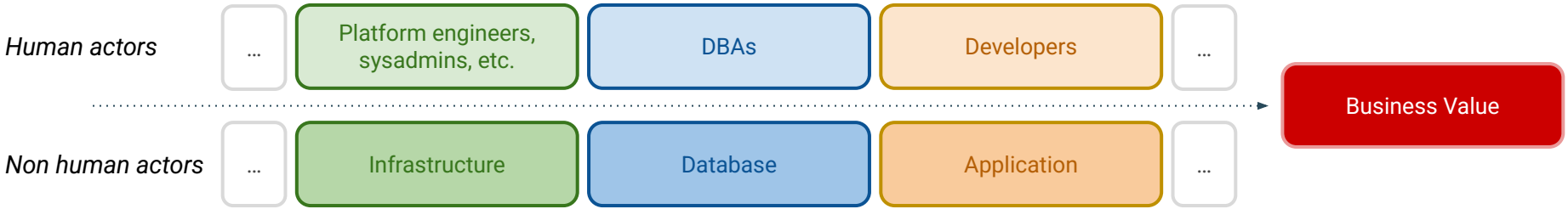
EDB
Postgres® for the AI Generation

The human side

aka *non-artificial intelligence*

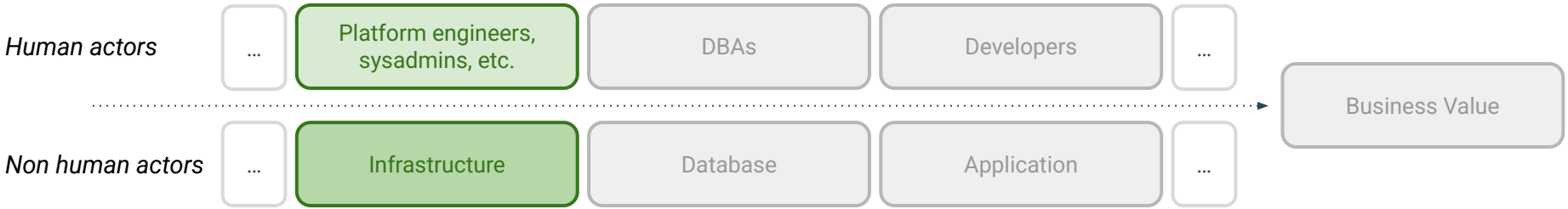
Postgres & Kubernetes

The main actors in an organisation



From traditional (VMs/BM) to Kubernetes

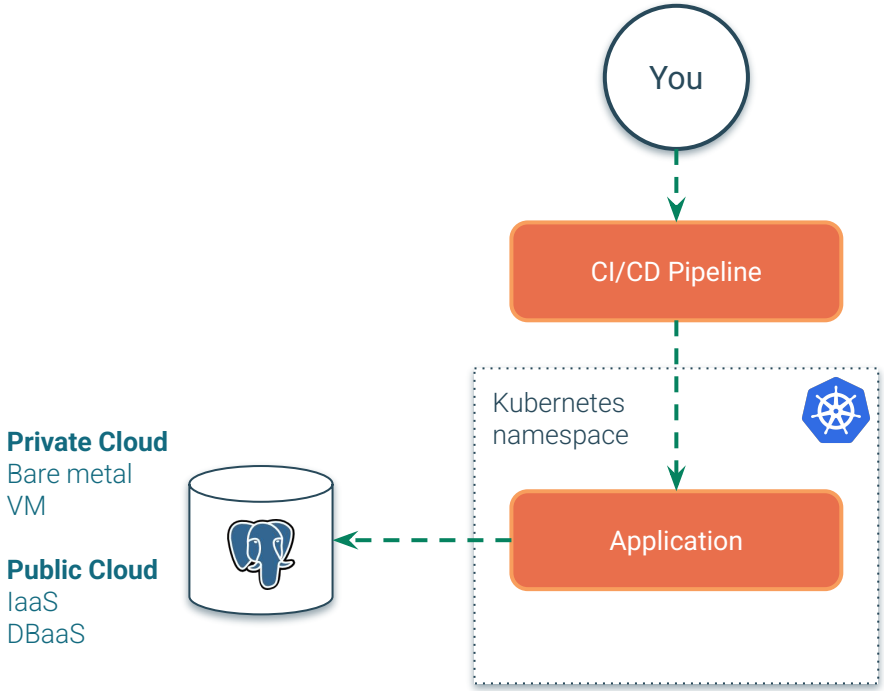
How change affects the infrastructure team



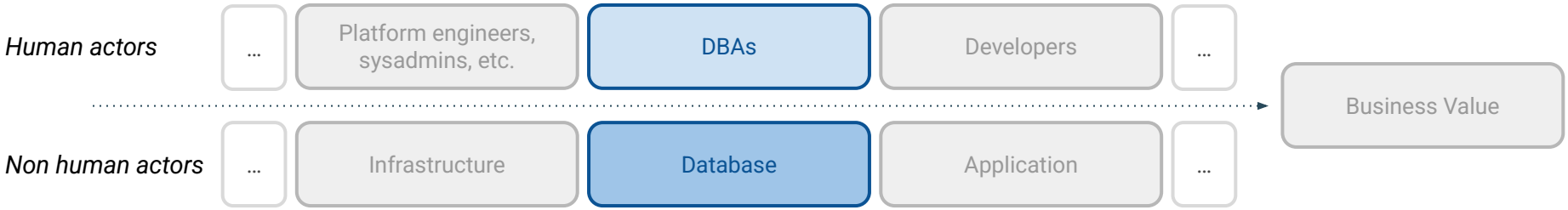
Infrastructure teams have driven **Kubernetes adoption** within their organisations.

Those not up-to-date with Kubernetes often believe databases should remain outside, shaped by early limitations.
First impressions last.

Database outside your Kubernetes cluster
(Bare metal, VM, IaaS, and DBaaS approaches)



From traditional (VMs/BM) to Kubernetes
How change affects the database team

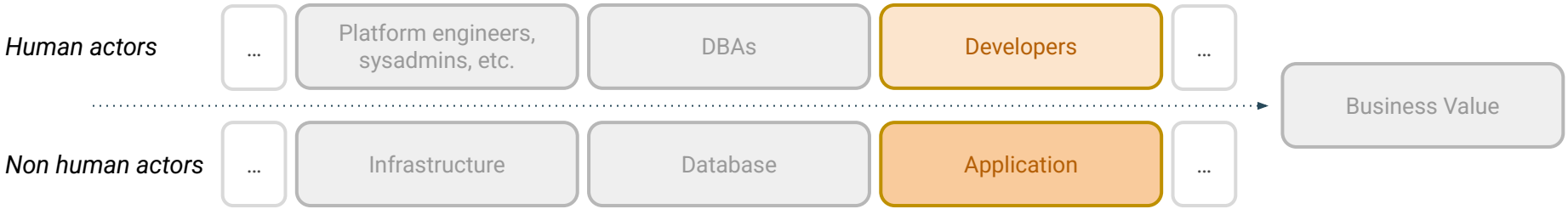


DBAs often have limited influence over Kubernetes adoption decisions, which can feel like an **imposed change** and trigger **self-defense** mechanisms (**TWWADI**).

Despite growing interest, the perception that "**Kubernetes isn't for databases**" remains widespread, reinforced by the perceived **steep learning curve** of Kubernetes.

From traditional (VMs/BM) to Kubernetes

How change affects the development team

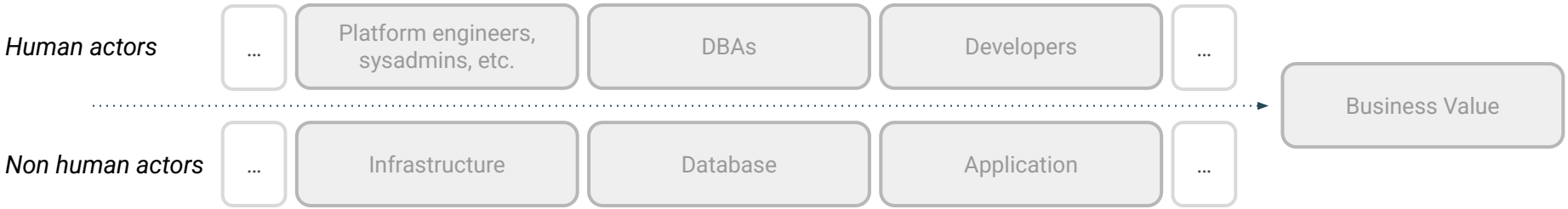


For application developers, the move to Kubernetes brings **minimal disruption**. Most changes involve adapting GitOps pipelines to build, test, and deploy **containerised applications**. Major benefits: improve velocity and reliability.

They **don't care** if their database runs inside Kubernetes, but they would benefit from the **microservice database**.

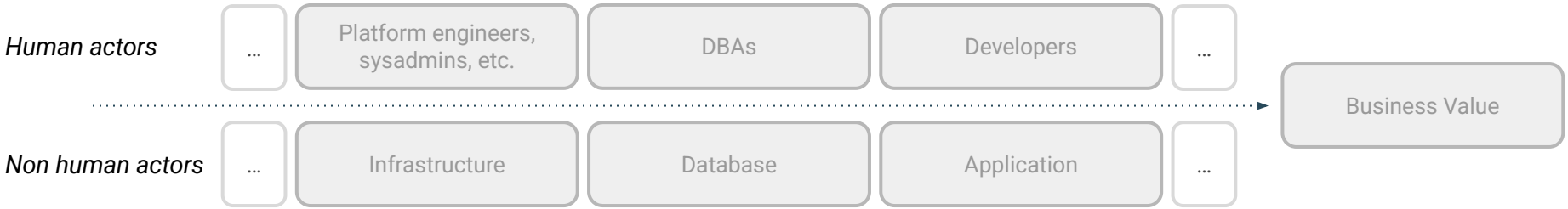
From traditional (VMs/BM) to Kubernetes

Infrastructure, DB, Dev teams are not “pushing” to run Postgres on K8s



Inertia leads to basic Postgres adoption in Kubernetes (“cattle”), DBaaS adoption or traditional VM/BM databases outside Kubernetes (TWWADI, comfort zone).

From traditional (VMs/BM) to Kubernetes
Infrastructure, DB, Dev teams are generally not aware



Databases are the **#1 workload** in Kubernetes.

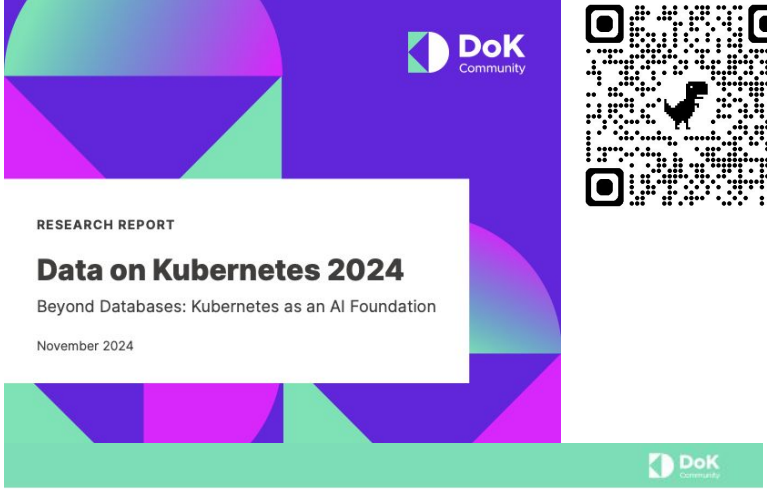
You can run Postgres on **bare-metal Kubernetes nodes,** without an hypervisor (and its licence).

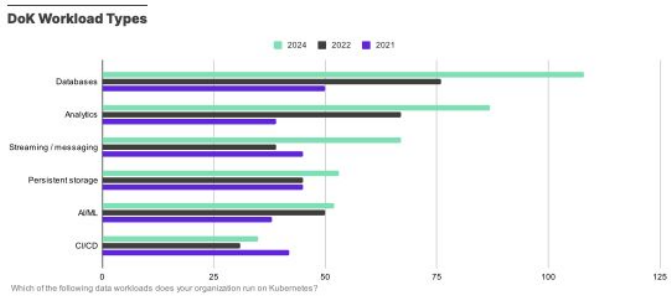
You can use **persistent local storage solutions**, including LVM.

Data on Kubernetes Community

Databases are #1 workload in Kubernetes

The Future of DBaaS on Kubernetes - M. Logan, S. Pronin, D. Sigireddi, G. Bartolini
Kubecon NA 2024 - Panel





Database Workloads: The Steady Foundation

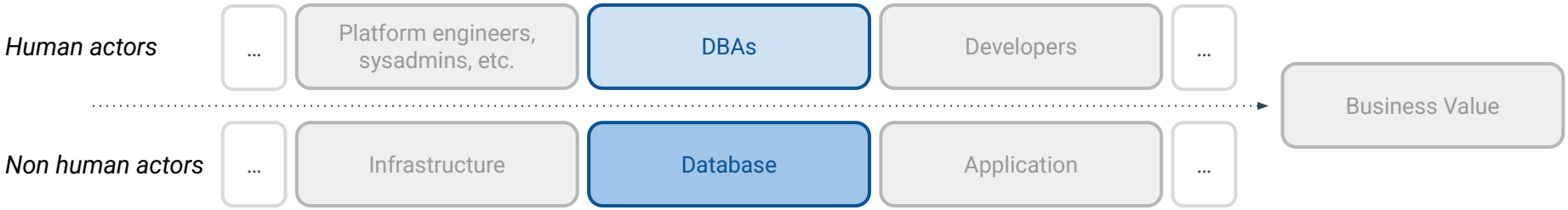
Databases continue to be the cornerstone of DoK deployments. For the third consecutive year, databases remain the most common DoK workload, demonstrating the platform's reliability for critical data services. The consistency in database workload adoption demonstrates:

1. **Platform Reliability:** Organizations trust Kubernetes for critical data services.
2. **Operational Standardization:** Growing comfort with running databases on Kubernetes.
3. **Deployment Confidence:** Increased willingness to run production database workloads.

Today, DBAs stand at a crossroads:
Is Kubernetes a threat, or is it an opportunity?

From traditional (VMs/BM) to Kubernetes

The Kubernetes opportunity for Postgres DBAs

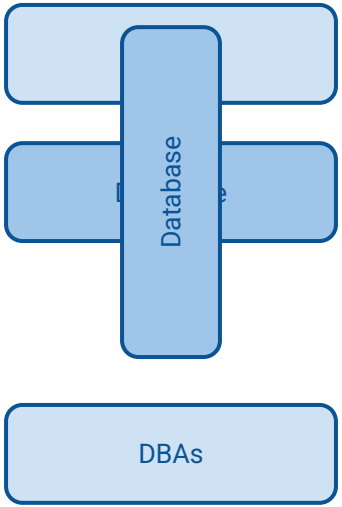


DBAs have a choice: they can **advocate** for **Postgres on Kubernetes** within their organisation by collaborating with their peers—infrastructure and development teams.

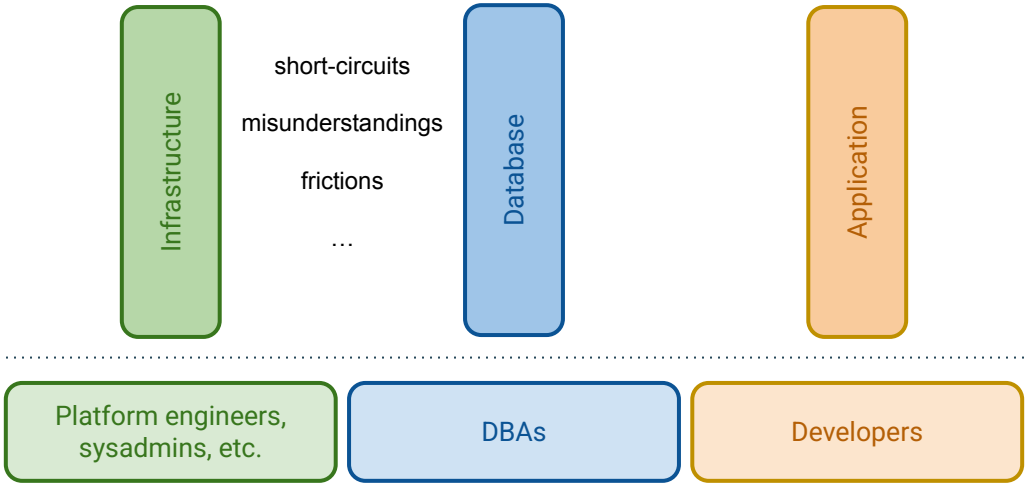
But how can they do this effectively?
The answer is: **“T”**

(Hint: T-shaped profiles. More on this next.)

I-shaped
(specialist)

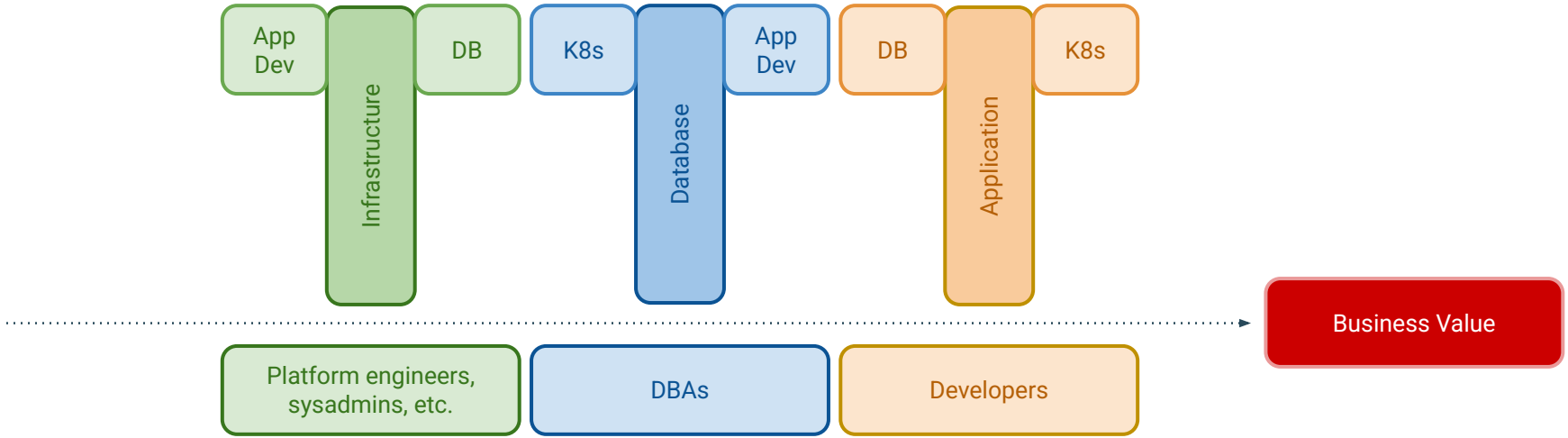


I-shaped
(specialist)



Silos
(are you familiar with DevOps?)

T-shaped
(specialist + generalist)



Breaking the silos
(Collaboration through constructive dissent)

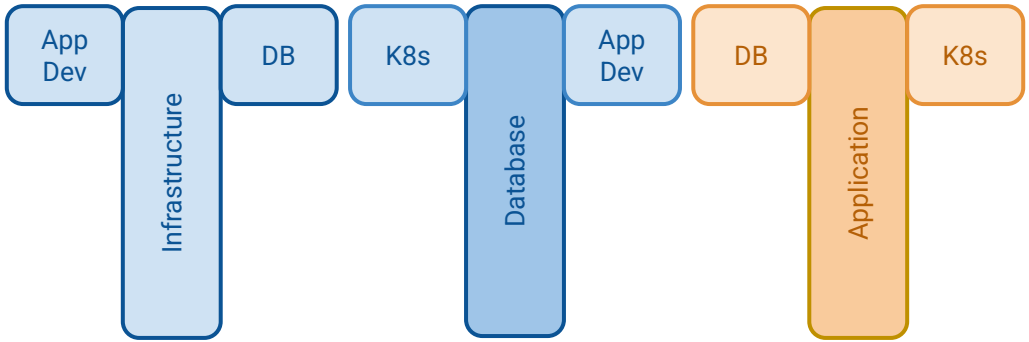
From infrastructure to applications

Fundamental concepts for DBAs to better understand Kubernetes (T-shaped profile)

- **Region (Kubernetes Cluster)**
- **Data Centre (Availability Zone)**
- **Nodes:**
  - Linux with K8s
  - VMs or Bare Metal
- **Network:**
  - The **Service** resource
    - **ClusterIP** type
    - **LoadBalancer** type
- **Storage:**
  - Network vs Local
  - Container Storage Interface (CSI)
  - The **StorageClass** Resource
- **Applications (workloads):**
  - The **Deployment** resource
  - Databases are applications

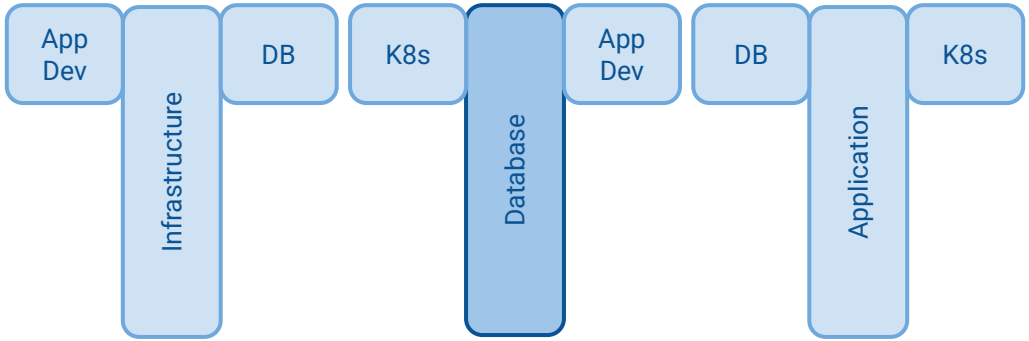
There's more …

Pi-shaped
(specialist + generalist + specialist)



There's more …

Comb-shaped
(specialist + generalist + specialist + specialist + …)



Your opportunity as a DBA today?

Learn enough Kubernetes to build a T-shaped profile— so you can start having smart, meaningful conversations across the organisation, not just with your peers.

The urge of “T-shaped” profiles to smooth the challenges of running Postgres in Kubernetes

12 August 2024 · 12 mins

Kubernetes K8s Cloudnativepg Postgresql Postgres Dok Data on Kubernetes Cnpg DBA Database Administrators I-Shaped Profile T-Shaped Profile TWWADI

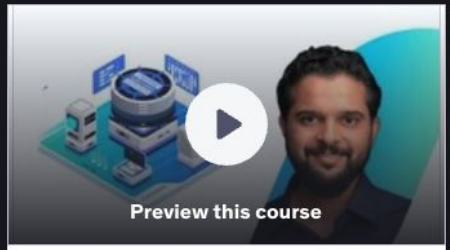
Certified Kubernetes Administrator (CKA) with Practice Tests

Prepare for the Certified Kubernetes Administrators Certification with live practice tests right in your browser - CKA

Bestseller 4.7 ★★★★☆ (82,857 ratings) 382,684 students

Created by Mumshad Mannambeth, KodeKloud Training

Last updated 06/2025 English English [CC], Arabic [Auto], 27 more



Personal Teams

Subscribe to Udemy's top courses

Get this course, plus 26,850+ of our top-rated courses, with Personal Plan. Learn more

Try Personal Plan for free

Starting at €13.00 pe

Cancel ar

or

€28.99

Add to

Prepare for your certification with this course. Learn more

CKA: Certified Kubernetes Administrator

Issued by The Linux Foundation

EDB
Postgres® for the AI Generation

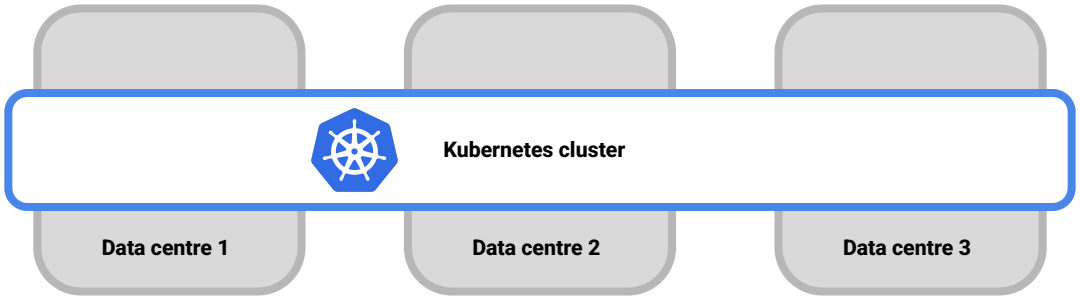
How the infrastructure changes

An architectural overview

Suggested reading from the CNCF blog

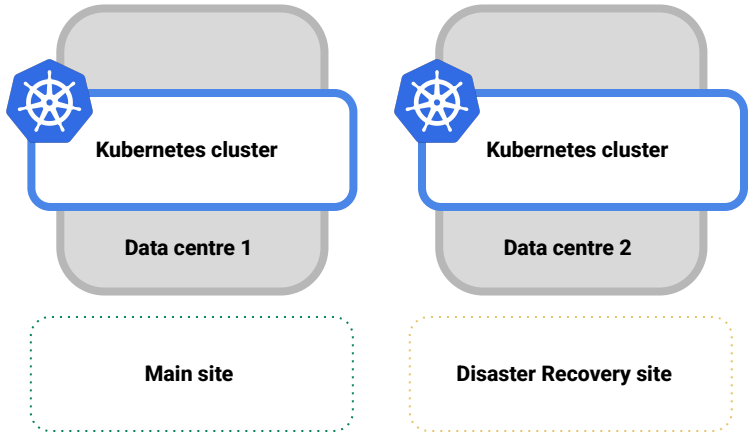
Public Cloud Environment

A typical Kubernetes cluster in cloud environments with 3+ availability zones



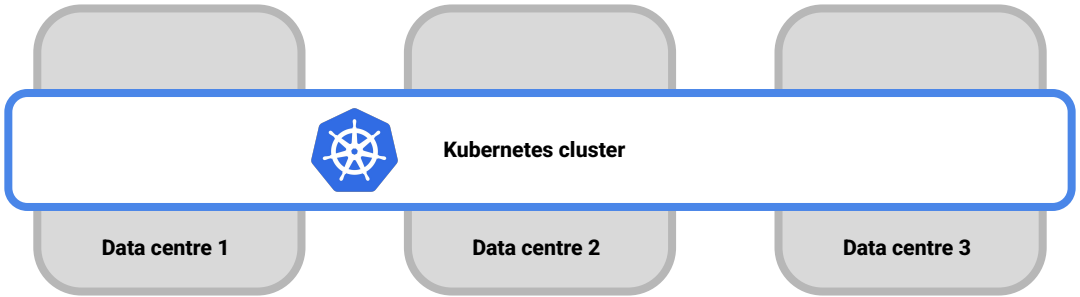
Private Cloud / On-premises Environment

A typical Kubernetes on-premises deployment with separate clusters for each data centre



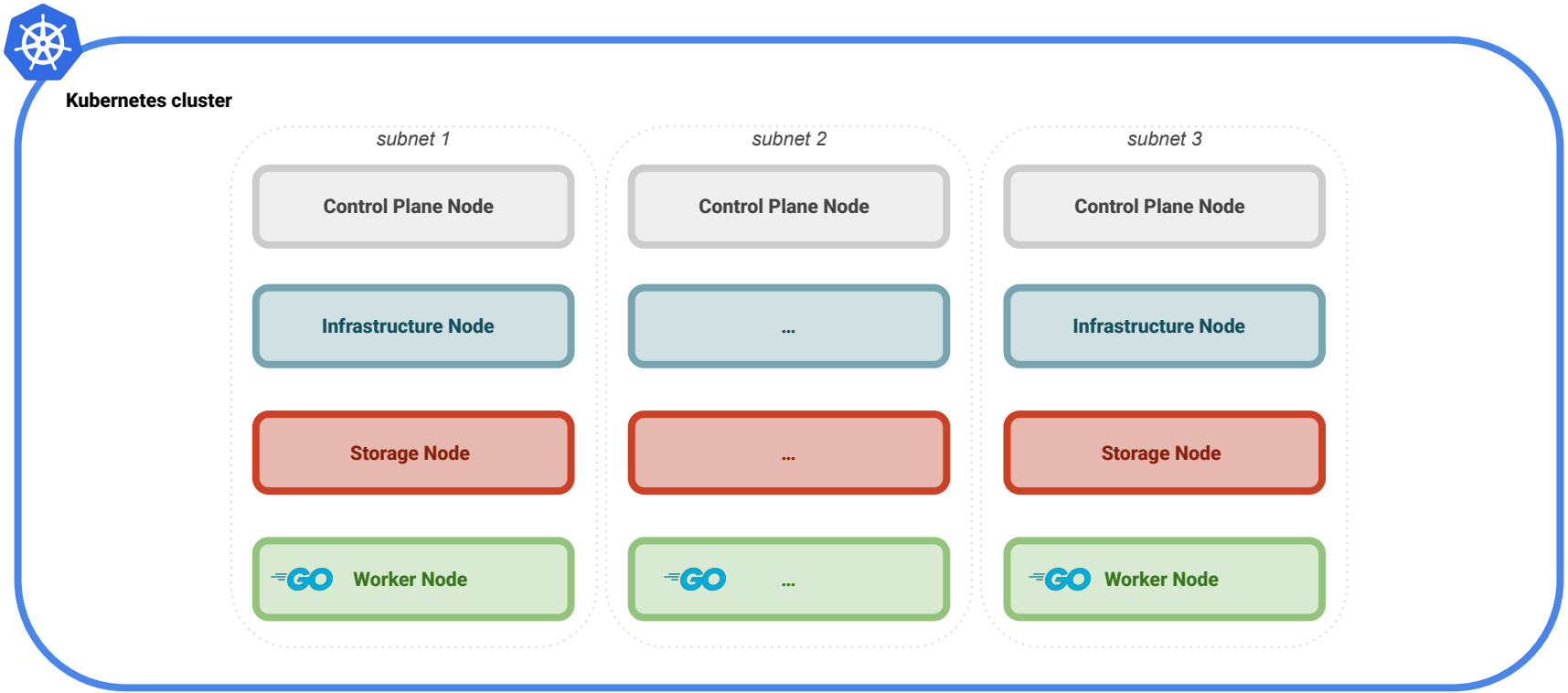
The future On-premises Environment

Data sovereignty and TCO are driving organisations to build a 3rd data centre for on-premises environments



No difference between private and public cloud environments.
Full self-healing at regional level.

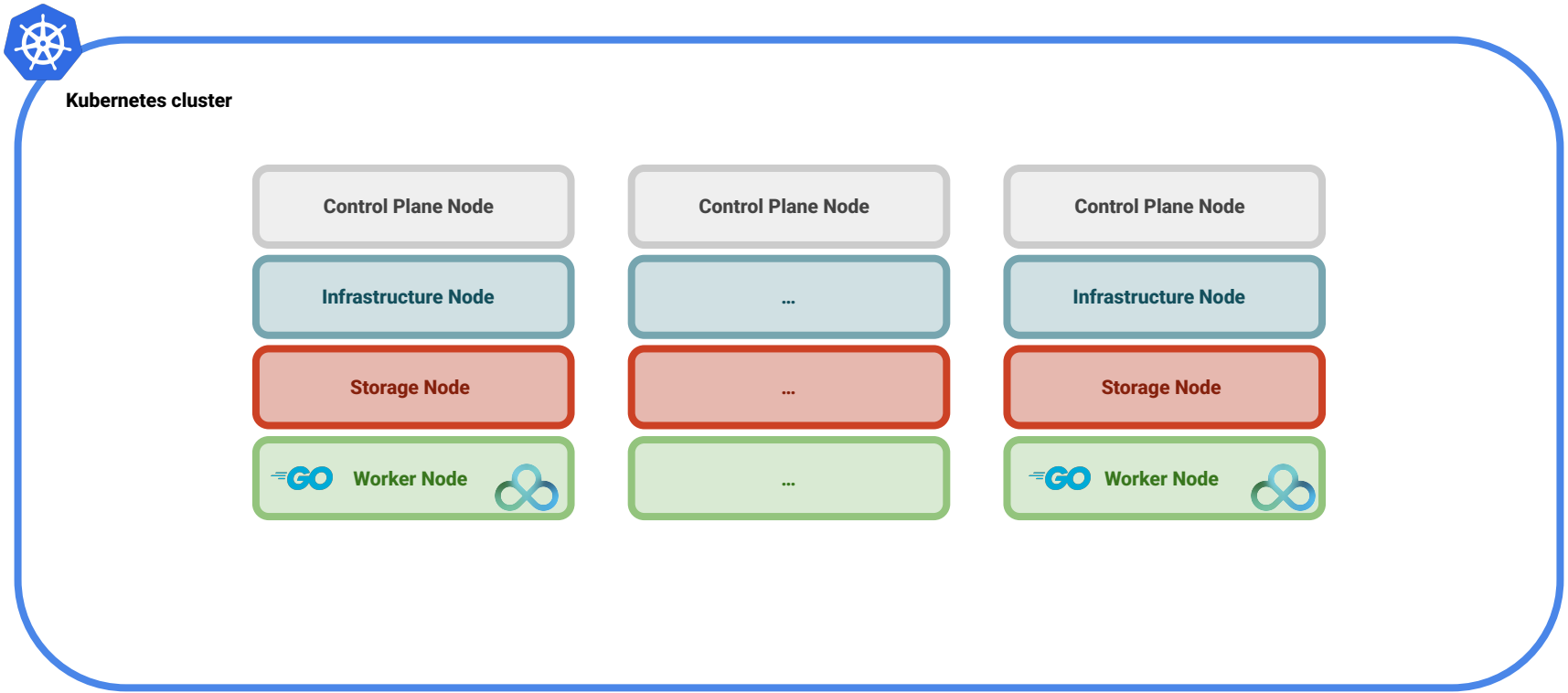
Example of hardware required in a data centre (“cattle approach”)



Kubernetes nodes are typically virtual machines

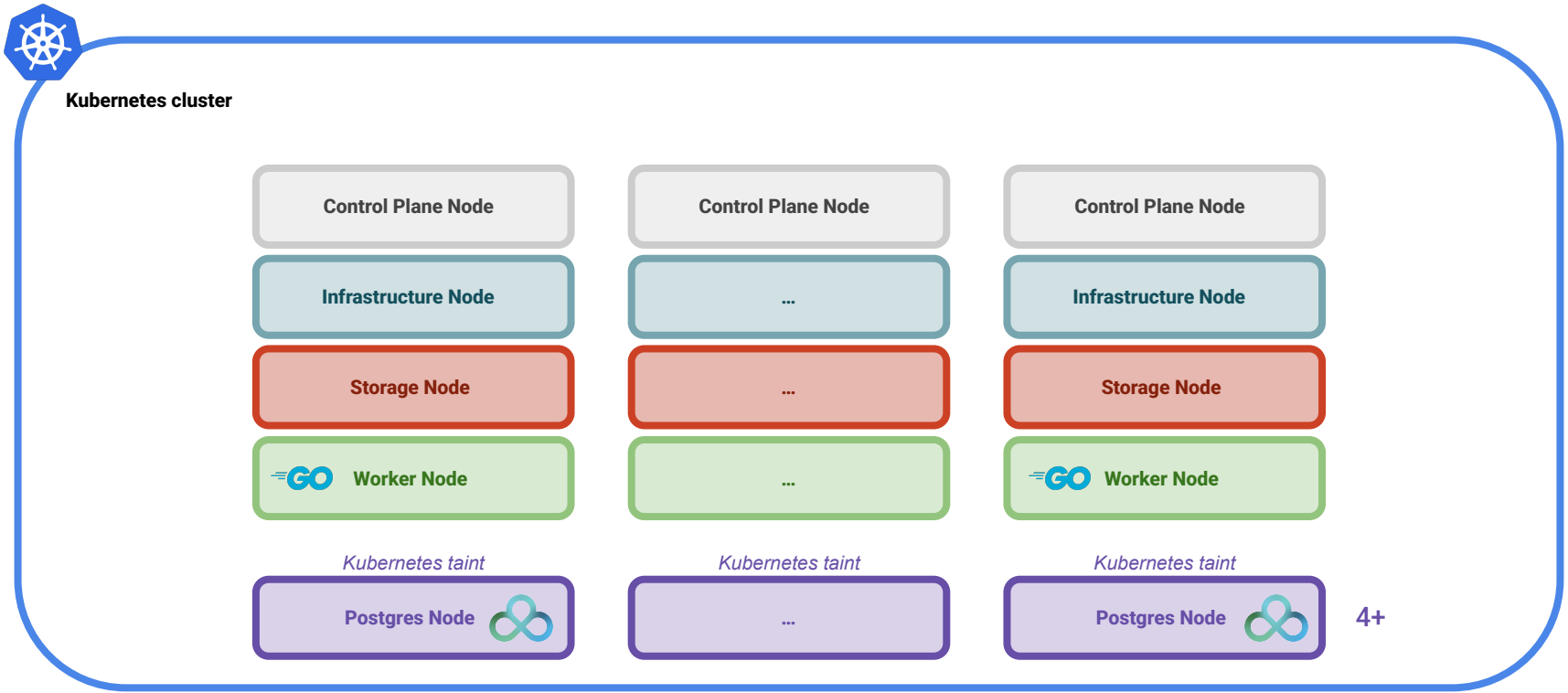
How can we add PostgreSQL in an existing Kubernetes cluster?

Common perception: shared workloads (“cattle” approach)



Kubernetes nodes are typically virtual machines

Recommendation: isolate Postgres workloads (“elephant herd” approach)



Kubernetes nodes are typically virtual machines

Cattle or **pets**? Better … **herds**

“Pinnawala orphanage provides a lifeline to the orphaned baby elephants and orphaned elephants lost in the wilderness” by Puviraj Diluckshan

From “cattle” to “elephants”: node labels and taints

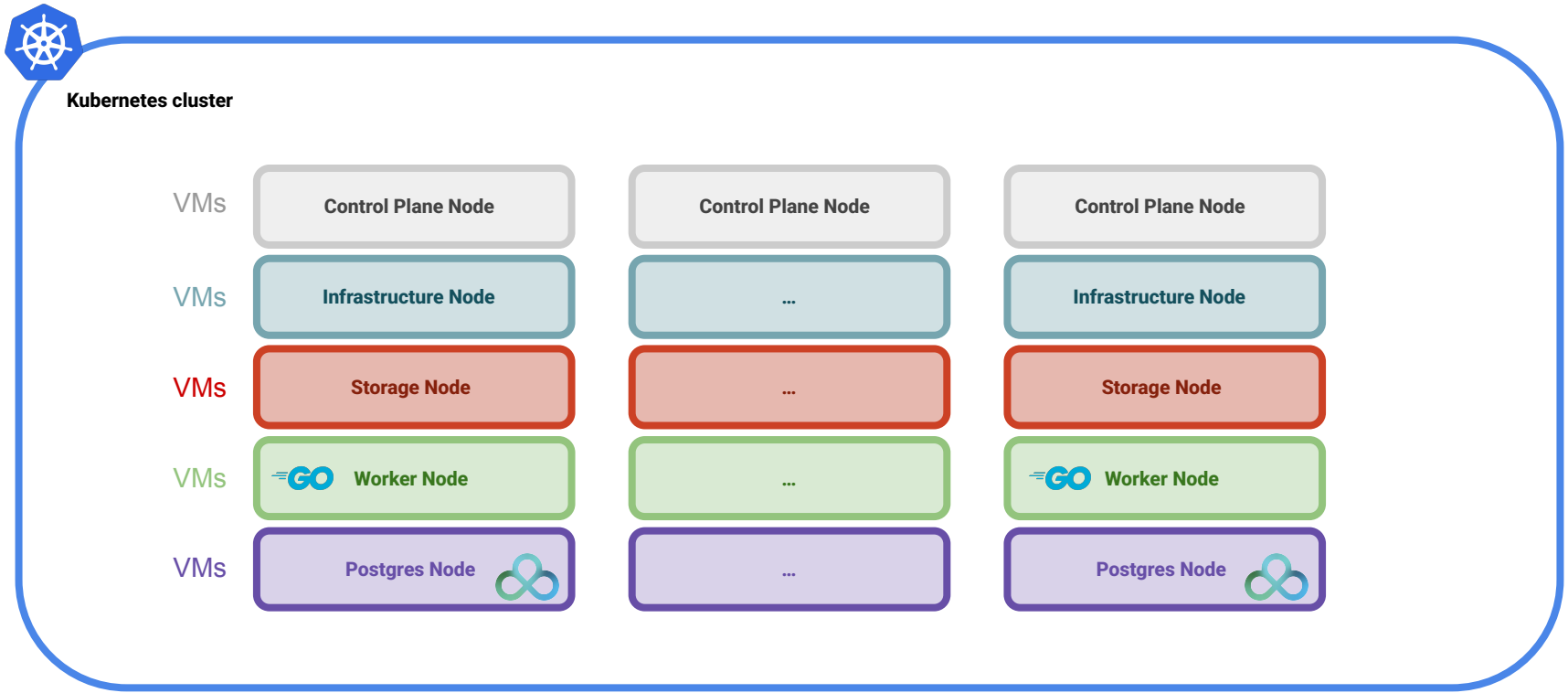
```
kubectl label ${node} \
    node-role.kubernetes.io/postgres="" \
    node-role.kubernetes.io/worker-

kubectl adm taint node ${node} \
    node-role.kubernetes.io/postgres=:NoSchedule
```

Reserving nodes for Postgres workloads

```
# <snip> "Cluster" resource
 affinity:
   enablePodAntiAffinity: true
   topologyKey: kubernetes.io/hostname
   podAntiAffinityType: required
   nodeSelector:
     node-role.kubernetes.io/postgres: ""
   tolerations:
   - key: node-role.kubernetes.io/postgres
     operator: Exists
     effect: NoSchedule
 # <snip>
```

Basic approach: use VMs for Postgres nodes (dedicated compute, shared storage)



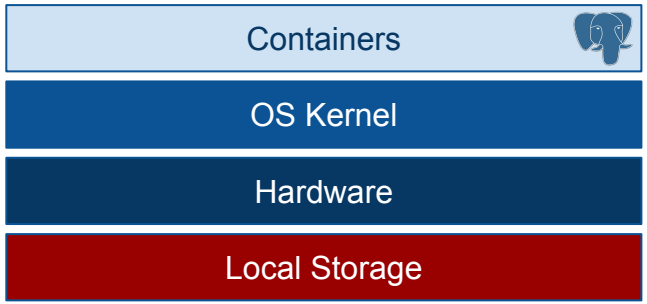
More resources, more VMs, more licenses

Does it mean that we can only run workloads on Kubernetes nodes installed in virtual machines?

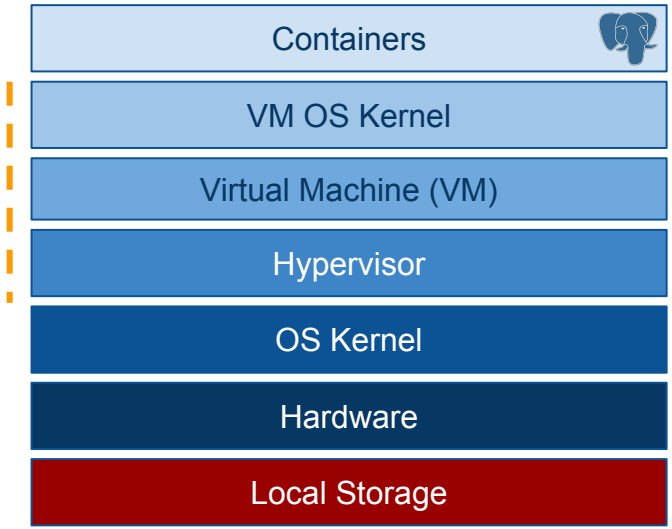
You can run Kubernetes on bare metal nodes

With locally attached and dedicated storage. Migrating “Postgres on VMs” to CloudNativePG on bare metal Kubernetes nodes.

Bare metal



VMs

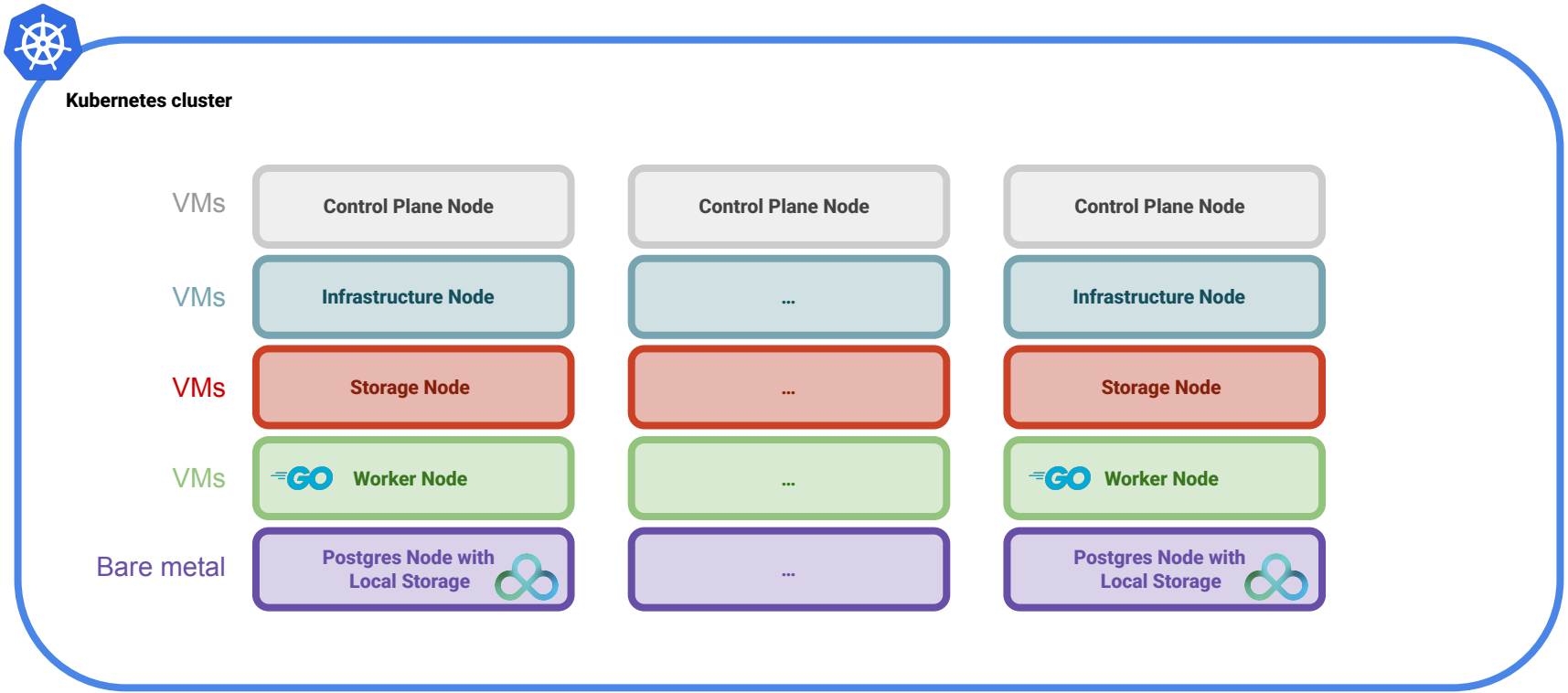


CloudNativePG leverages PostgreSQL's native physical replication to synchronise states across different locations, including cascading and synchronous replication at the transaction level, as well as Hot Standby.

File storage replication in PostgreSQL has gradually become obsolete, starting in 2005 with the introduction of Warm Standby.

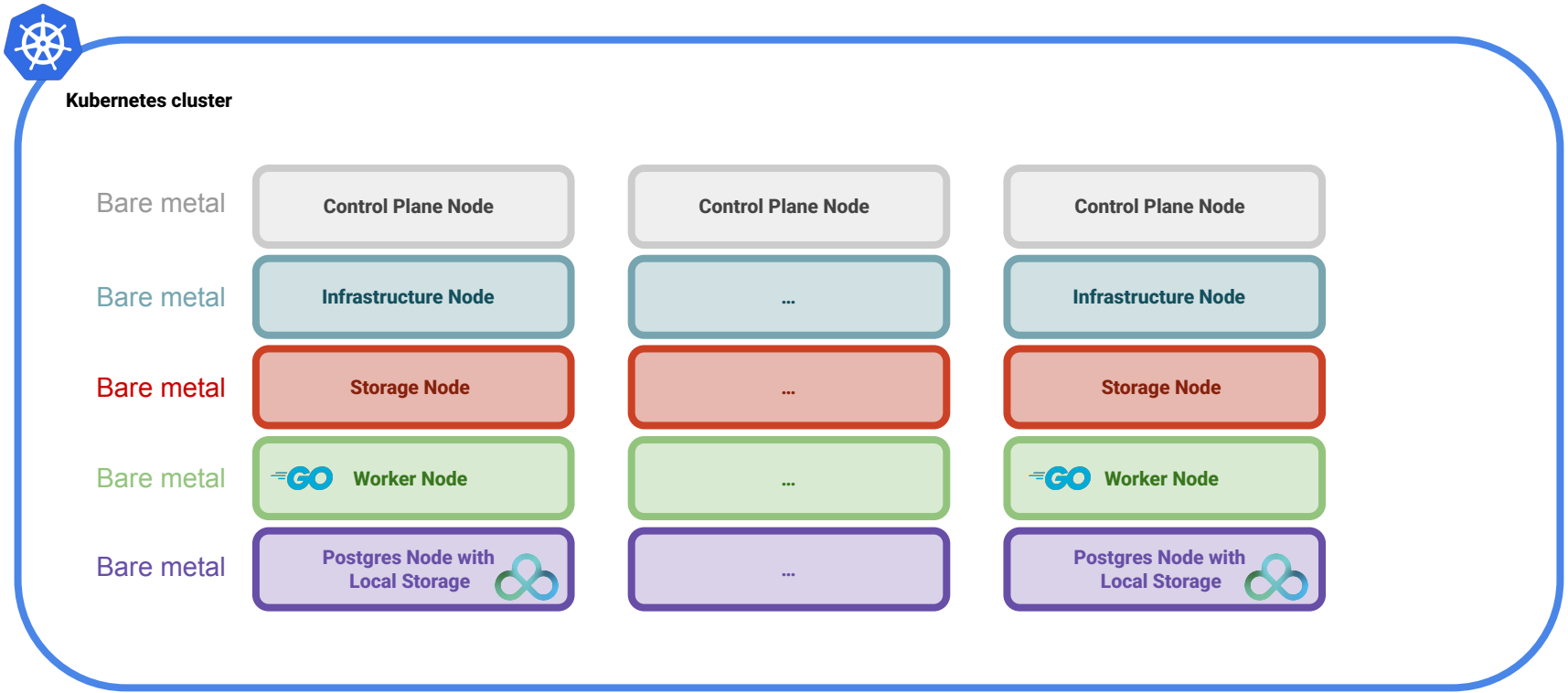
Although CloudNativePG is storage-agnostic, **relying on storage replication for PostgreSQL in Kubernetes is considered bad practice.** It also leads to “write-amplification” when used with Postgres replicas.

Recommended approach: shared-nothing architecture



Mindshift required, but better predictability, stability, durability, scalability

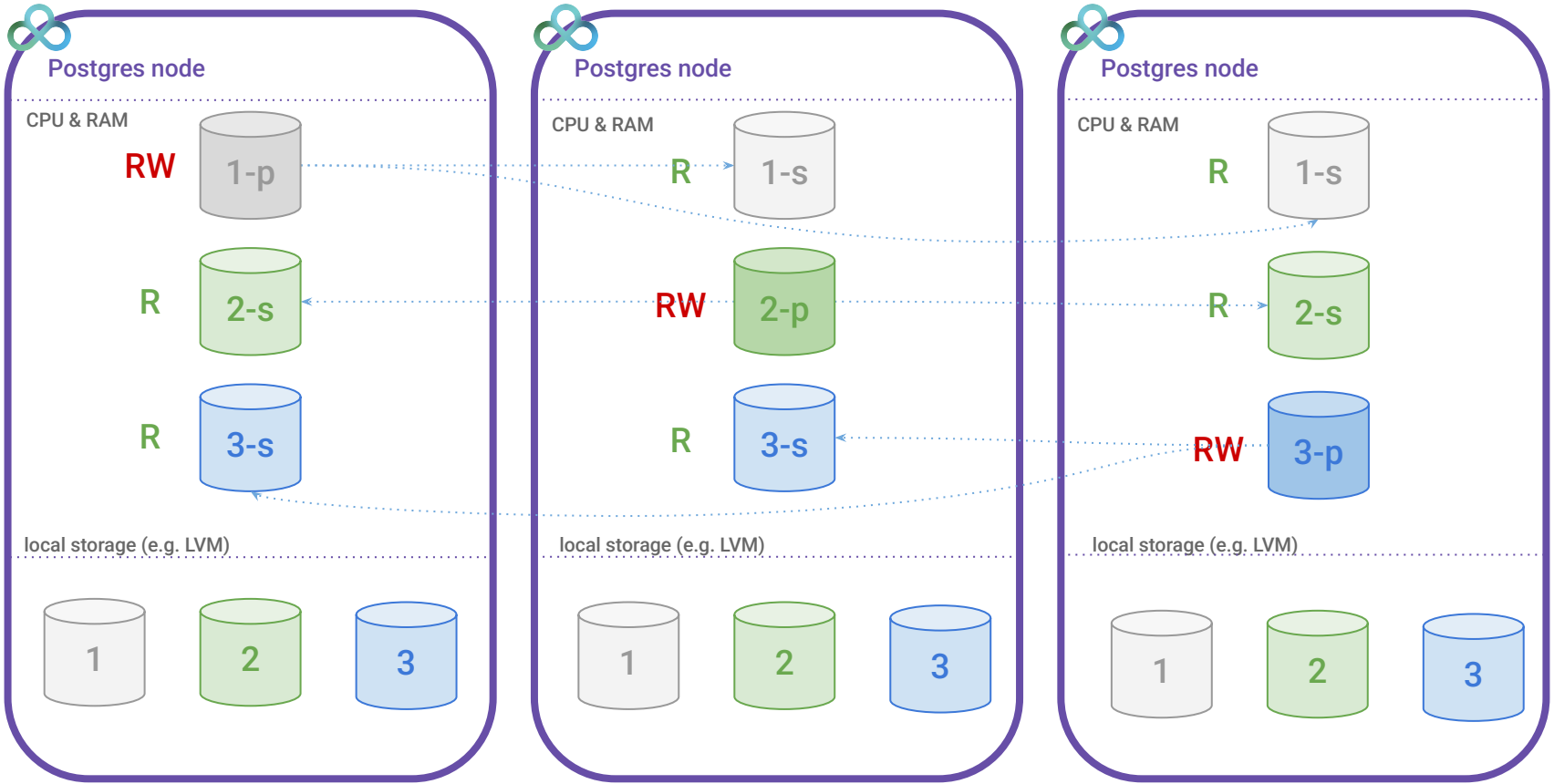
Recommended approach: full bare-metal with shared-nothing architecture for Postgres



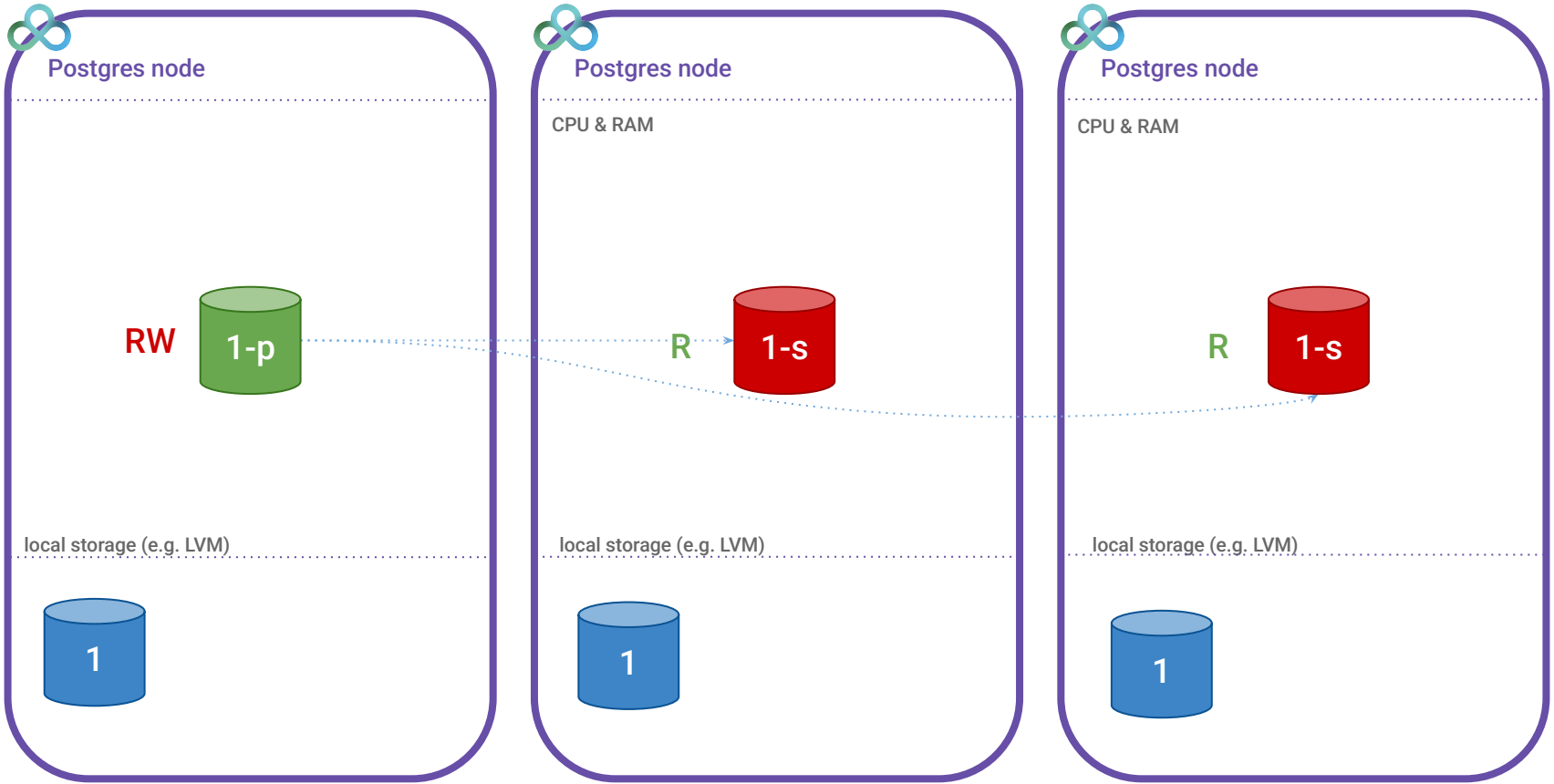
Use Kubernetes to consolidate your workloads, without an underlying hypervisor

The actual scheduling order might change

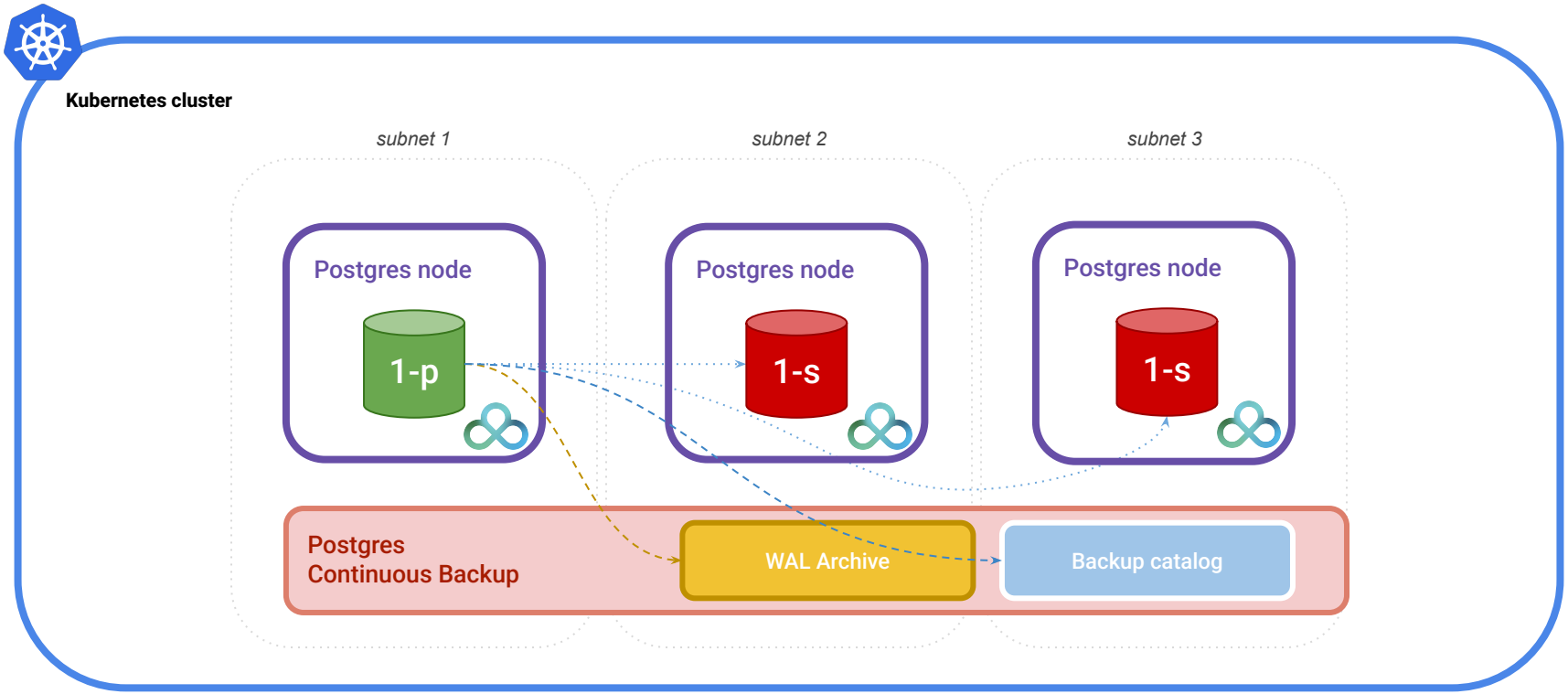
How CloudNativePG uses Postgres worker nodes



Let’s examine in more detail a single Postgres cluster lifecycle and architecture

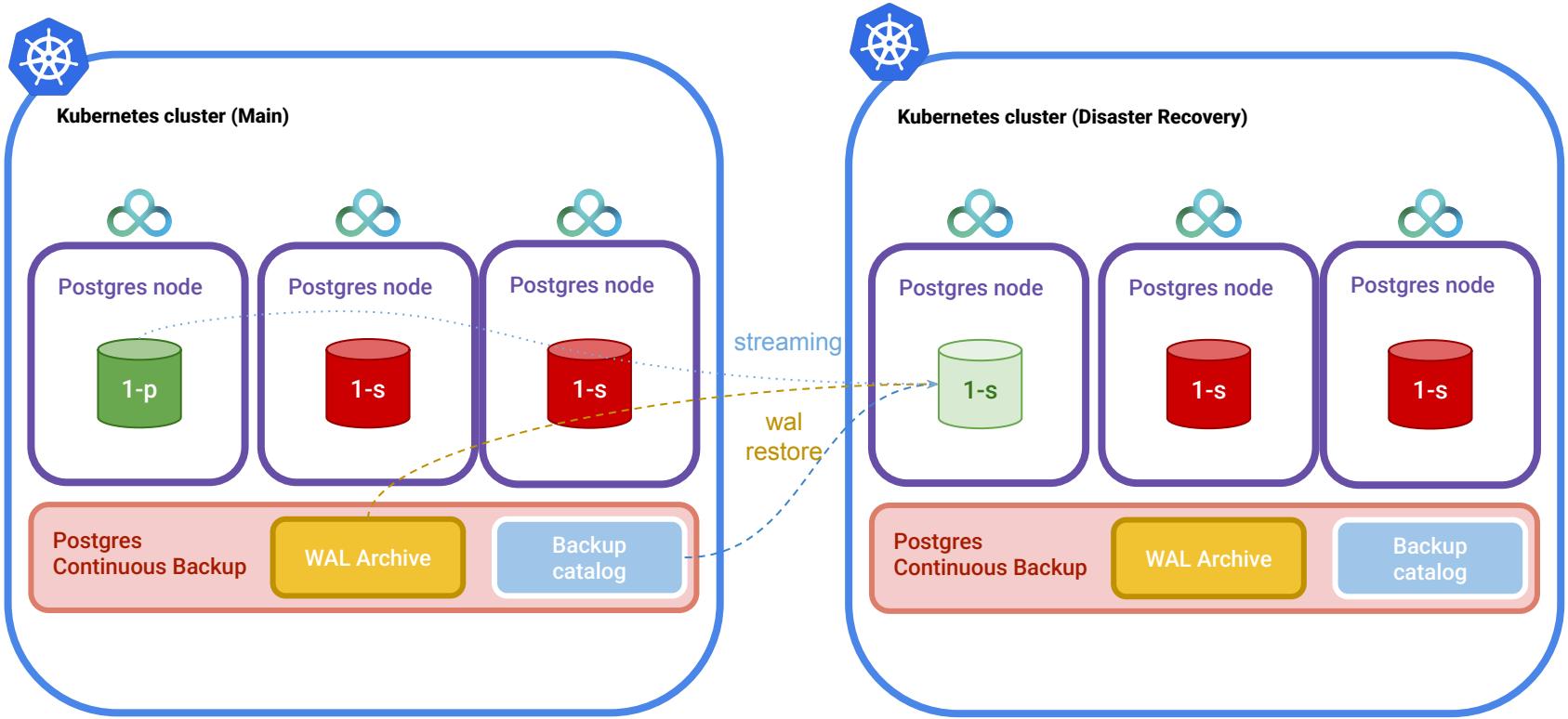


Let’s examine in more detail a single Postgres cluster lifecycle and architecture

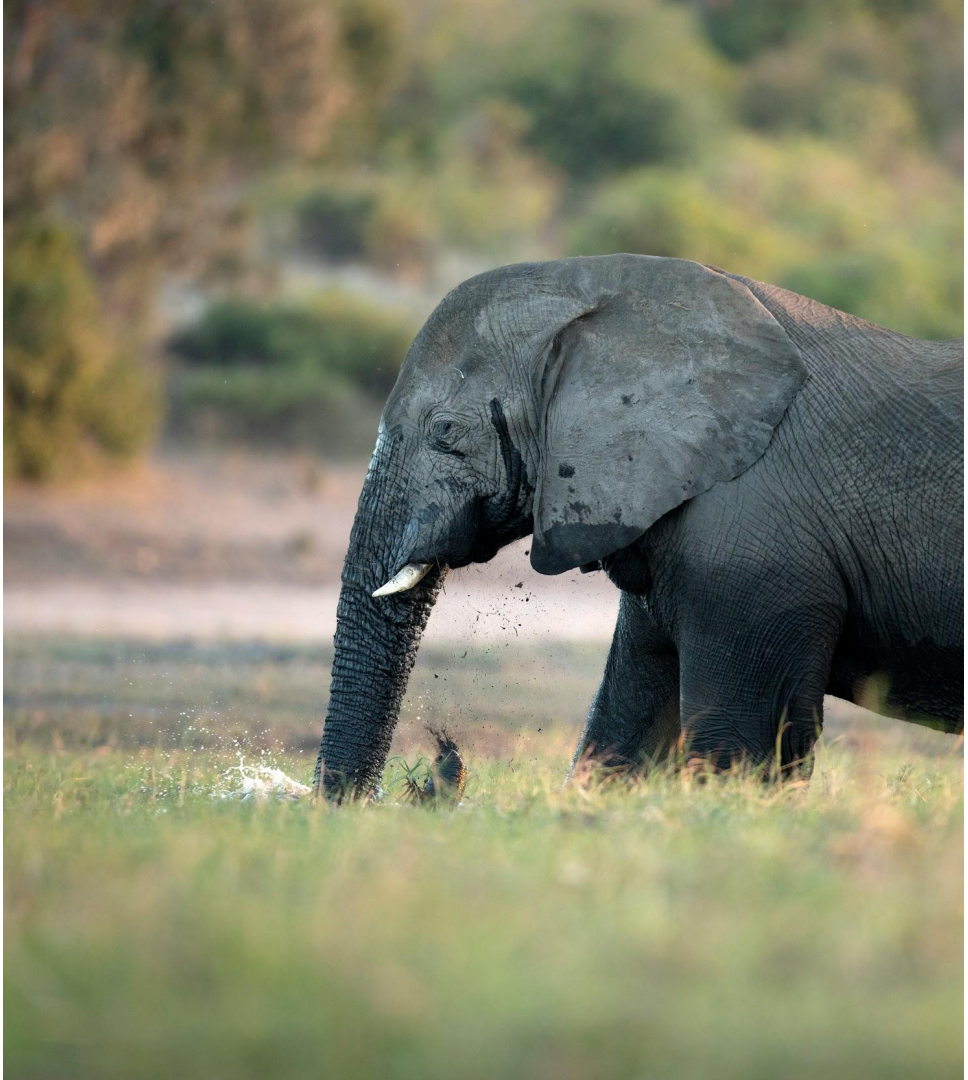


Full HA and DR within a Kubernetes Cluster

How Disaster Recovery across two Red Hat OpenShift clusters is addressed



Declarative demotion, Declarative promotion (no automated failover)



EDB
Postgres® for the AI Generation

How the database changes

CloudNativePG and its API

Database workloads

- **Kubernetes' standard resources are not sufficient for databases**
- A PostgreSQL database is a complex application:
  - Deployment and configuration
  - Failure detection and Failover
  - Updates and switchovers
  - Backup and Recovery
  - …
- **Running PostgreSQL in Kubernetes requires an Operator**

Extending the Kubernetes API

- Kubernetes API is extensible (similarly to PostgreSQL)
  - You can define new data types with custom resources
  - You can define a custom controller for a custom resource (or more)
    - You declare the state of the custom resource
    - The controller ensures that the **current** state matches the **declared** one
      - Foundation of self-healing
- The operator pattern is a development pattern
  - Designed to manage complex applications
  - Custom resources and custom controllers
  - Ensures a declarative API

Imperative vs Declarative with an example

Imperative

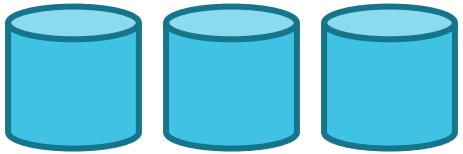
1. Create a PostgreSQL 17 instance
2. Configure for replication
3. Clone a second one
4. Set it as a replica
5. Clone a third one
6. Set it as a replica

Declarative (Kubernetes)

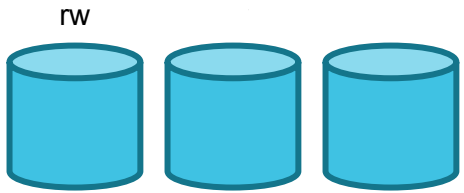
There's a PostgreSQL 17 cluster with 3 instances (i.e. one primary and two replicas). **At any time.**

A PostgreSQL 17 cluster resource with 3 instances

spec (Desired State)

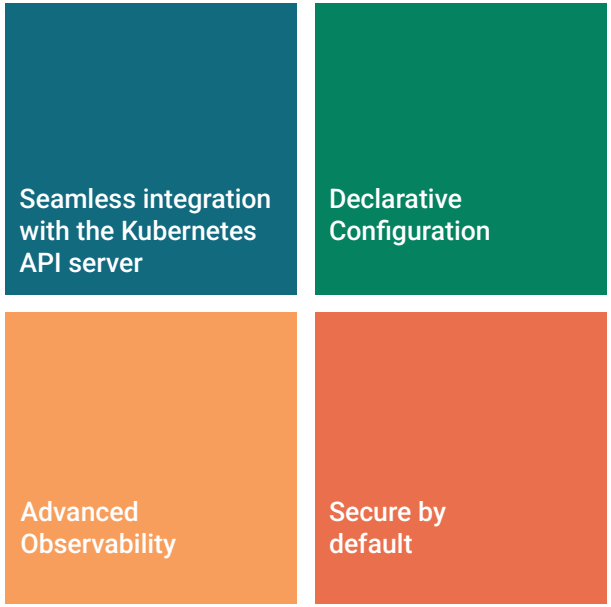


status (Observed State)



(simplified view)

The **4 pillars** of CloudNativePG that make it a **Kubernetes native database**



CloudNativePG

github.com/cloudnative-pg

The main features of CloudNativePG

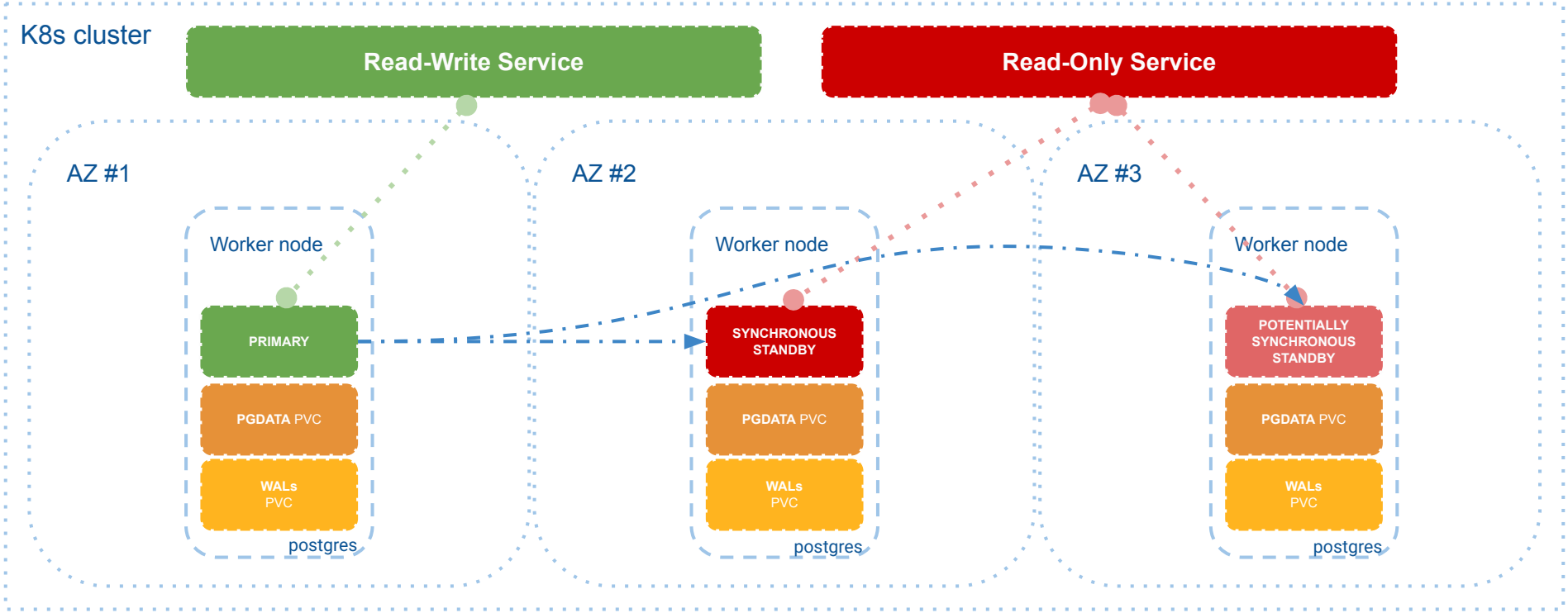
Overview of CloudNativePG's main features covering through Day 2 operations

- High Availability and Self-Healing
- Support for local PVCs
- Managed services for rw and ro workloads
- Continuous backup (including snapshots)
- Point In Time Recovery (incl. snapshots)
- Scale up/down of read-only replicas
- "Security by default", including mTLS
- Native Prometheus exporter
- Logging to stdout in JSON format
- Rolling updates, incl. minor Postgres releases
- Synchronous replication
- Major upgrades of Postgres
- Online import of Postgres databases
- Separate volume for WALs
- Postgres tablespaces, including temporary
- Replica clusters and distributed topologies
- Declarative roles and database
- Declarative hibernation and fencing
- Connection pooling
- Postgres extensions (pgvector, PostGIS, …)

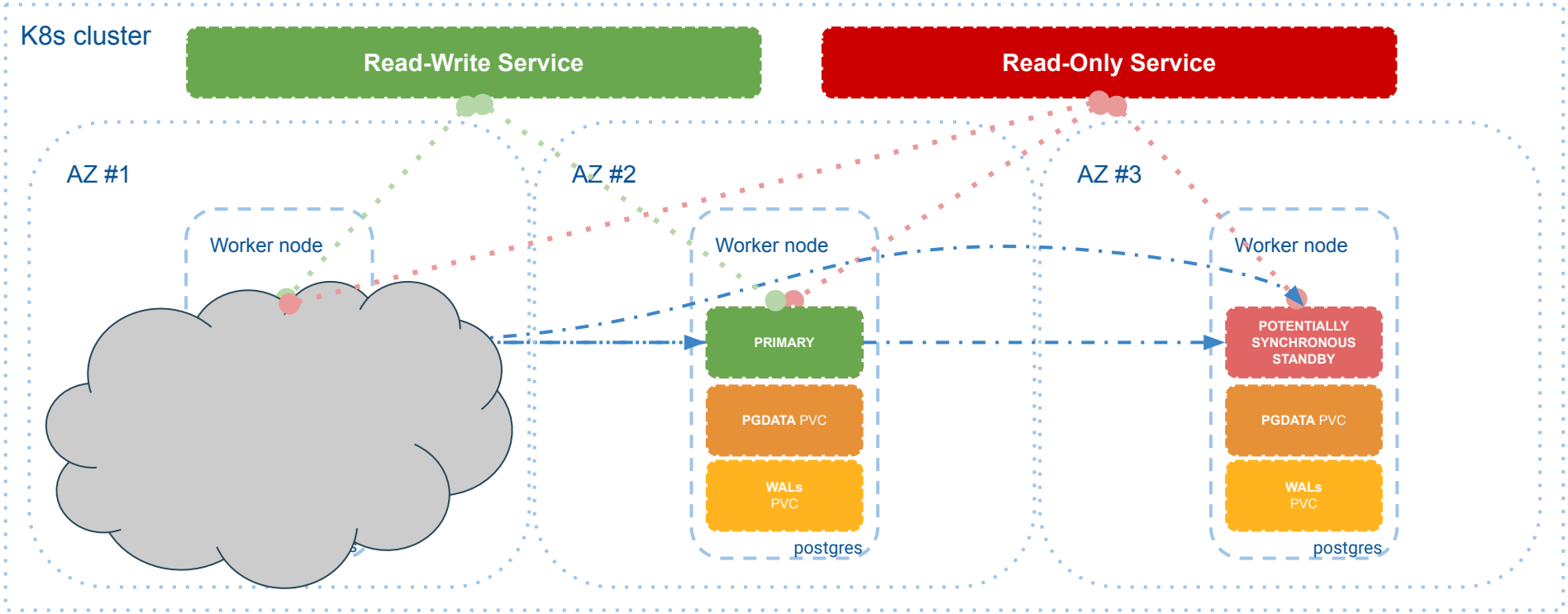
The PostgreSQL `Cluster` resource

```
apiVersion: postgresql.cnpg.io/v1
kind: Cluster
metadata:
  name: clapton
spec:
  instances: 3
  affinity:
    nodeSelector:
      node-role.kubernetes.io/postgres: ""
  postgresql:
    synchronous:
      method: any
      number: 1
  storage:
    size: 40Gi
  walStorage:
    size: 10Gi
```

Highly Available PostgreSQL Cluster



Automated failover (HA with very low RTO)

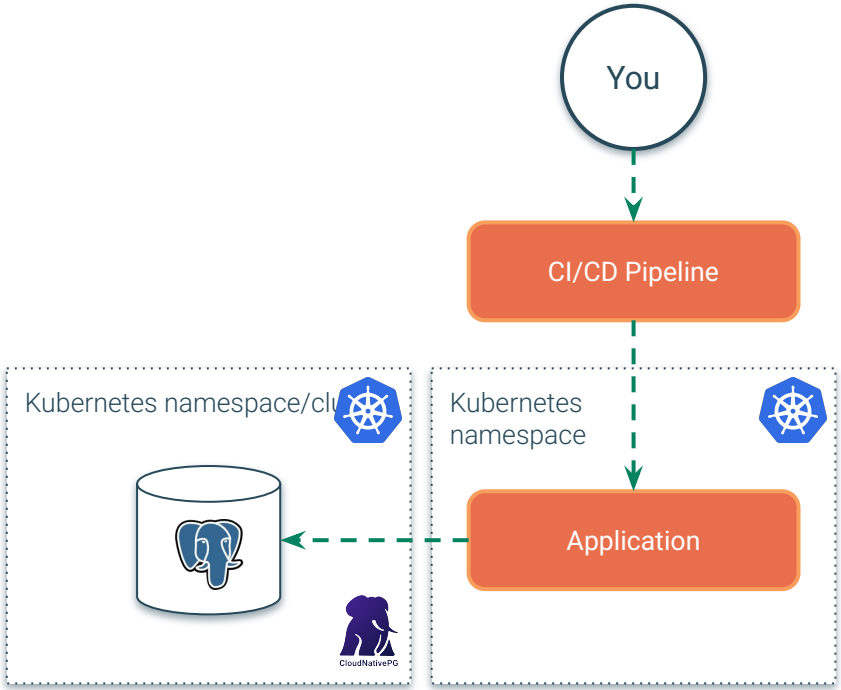


Major differences with traditional Postgres deployments

How cloud native integration is forcing DBAs to change

- DBAs will own “Cluster” resources
  - Shifting from “Postgres instances” to “Postgres HA clusters”
- Everything must be declarative
  - In most cases, no superuser access is needed (no ALTER SYSTEM)
  - Changes need to go through YAML configuration, including roles and databases
- Logs will not be stored in the containers
  - Sent to the stdout channel of the container in JSON format and integrated with infrastructure logs
- Metrics are exported for Prometheus
  - Custom metrics can be defined using YAML (config maps)
  - Alerts can be defined with Alert Manager
- Although CNPG takes care of most day 1 and 2 operations, humans are needed

Database outside your Kubernetes namespace
(CloudNativePG internal DBaaS approach)



EDB
Postgres® for the AI Generation

How the application changes

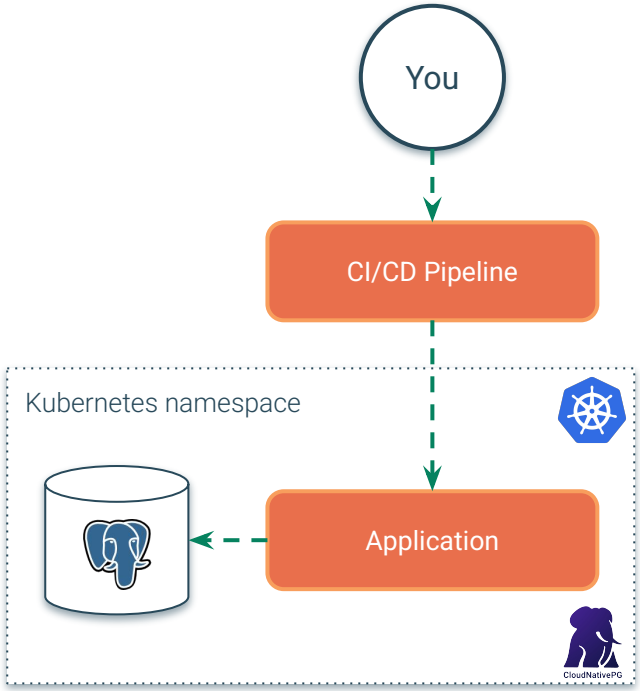
The microservice database

The microservice database

Application teams should own their Postgres databases

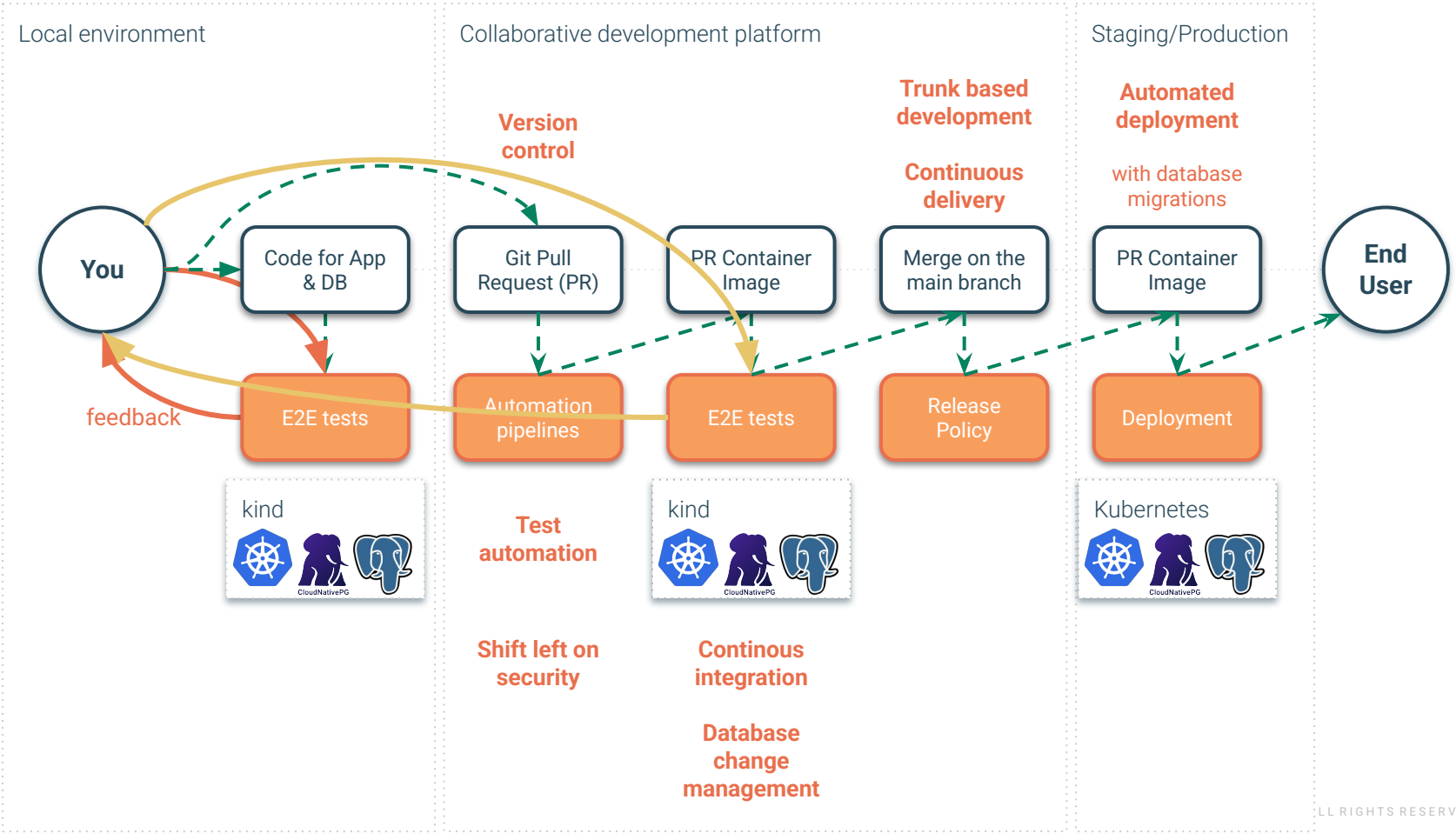
- DBAs work with application teams
- The “Cluster” resource is part of the application
  - Source code
  - GitOps processes
    - Database change management part of automated pipelines
- One Postgres Cluster hosts:
  - a single “microservice” database (e.g. “app”)
  - owned by a regular user (e.g. “app”)
- Multi-tenancy is achieved at Kubernetes namespace layer

Database inside your Kubernetes namespace
(CloudNativePG microservice database)



Stream aligned team

DevOps capabilities, CloudNativePG databases and automated pipelines

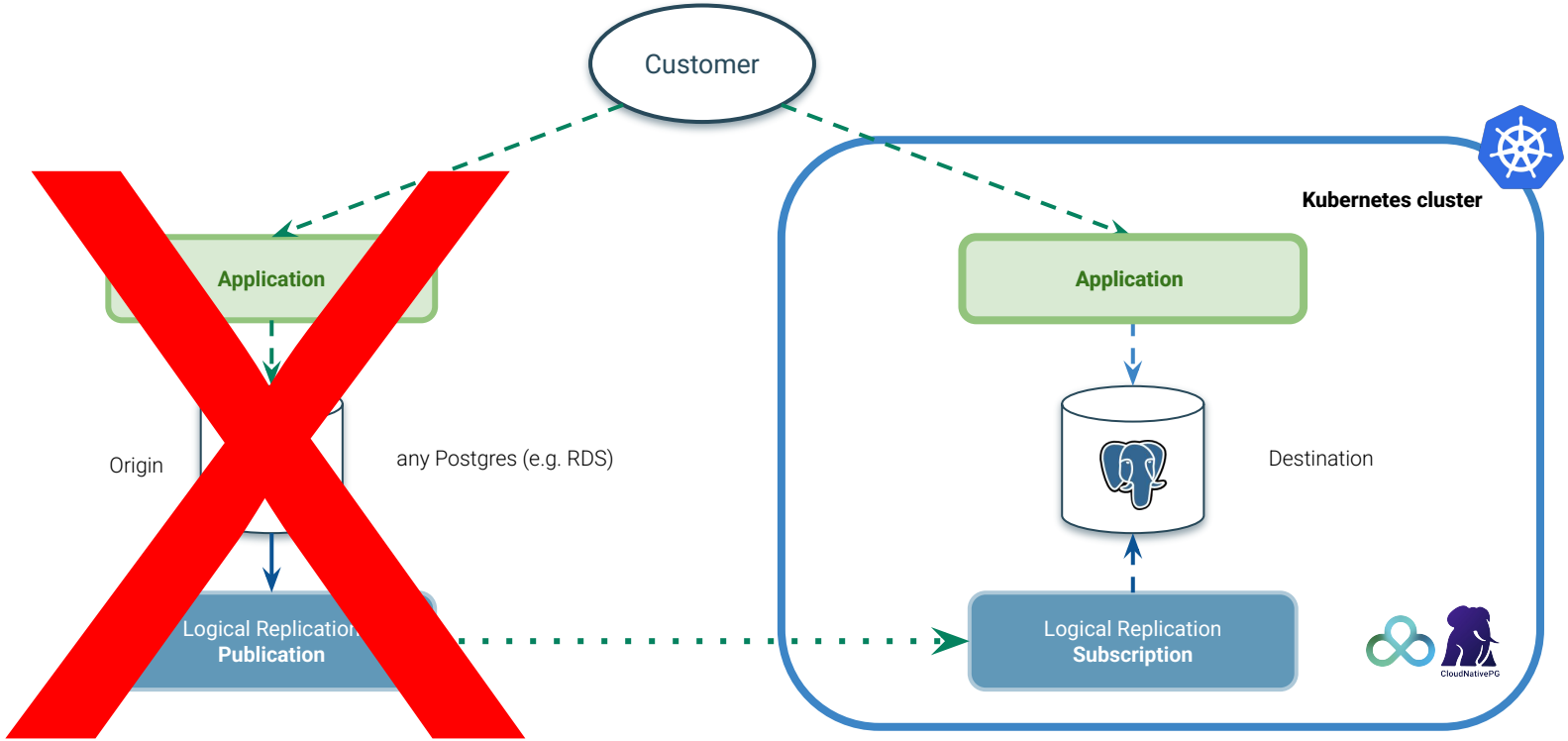


EDB
Postgres® for the AI Generation

Conclusions

~Zero cutover migrations from any Postgres anywhere

Use Postgres native logical replication and CloudNativePG declarative subscriptions for blue/green migrations with ~0 downtime



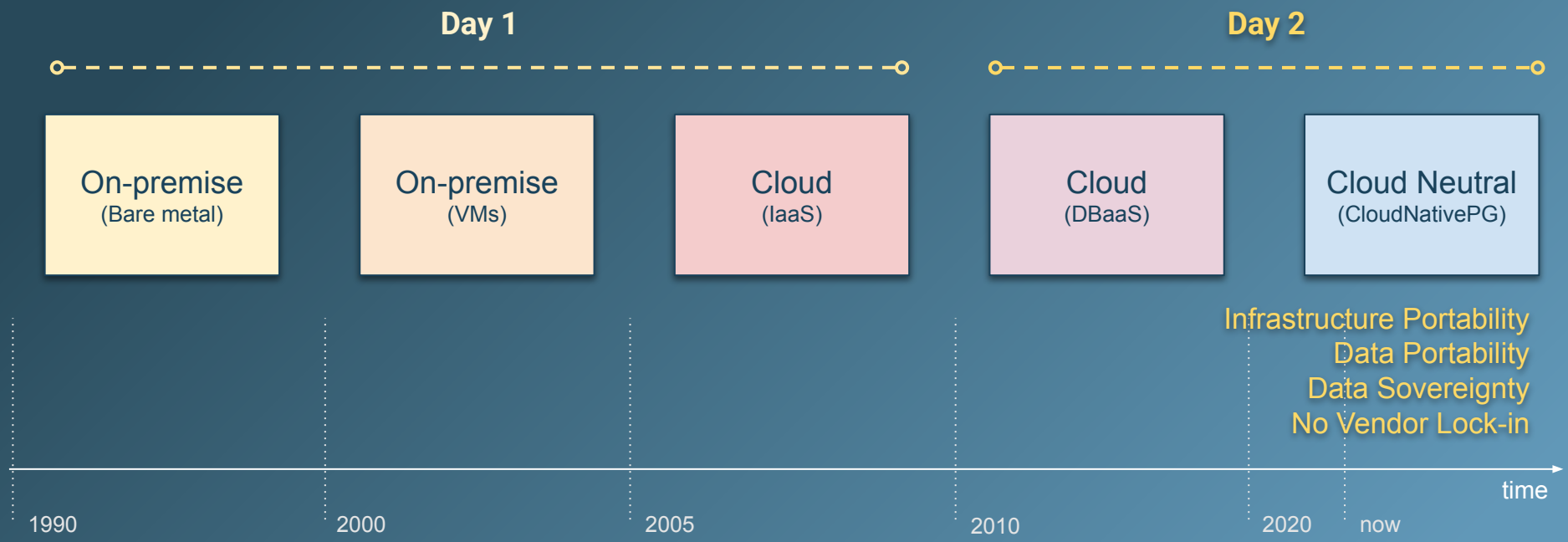
Moving Postgres to Kubernetes

It is not a lift-and-shift transition, it requires a mindshift

- **Break the silos**
  - Multi-disciplinary team with dev, infrastructure and DBA SMEs
  - You need Kubernetes and Postgres skills in your broader team
    - **That's where professional support can accelerate your process and reduce risks**
- **Application and Postgres database are in the same Kubernetes cluster**
  - Possibly in the same namespace
  - Separate Kubernetes clusters for databases are common (sigh!)
    - Silo culture "devs vs ops" no security gains, more operational complexity
- **Start with a pilot project**
  - Start with "cattle" approach, if the "elephant" is not possible
- **Based on success of the project, start planning for the "elephant" approach**
  - Isolate Postgres nodes, consider bare metal and local storage

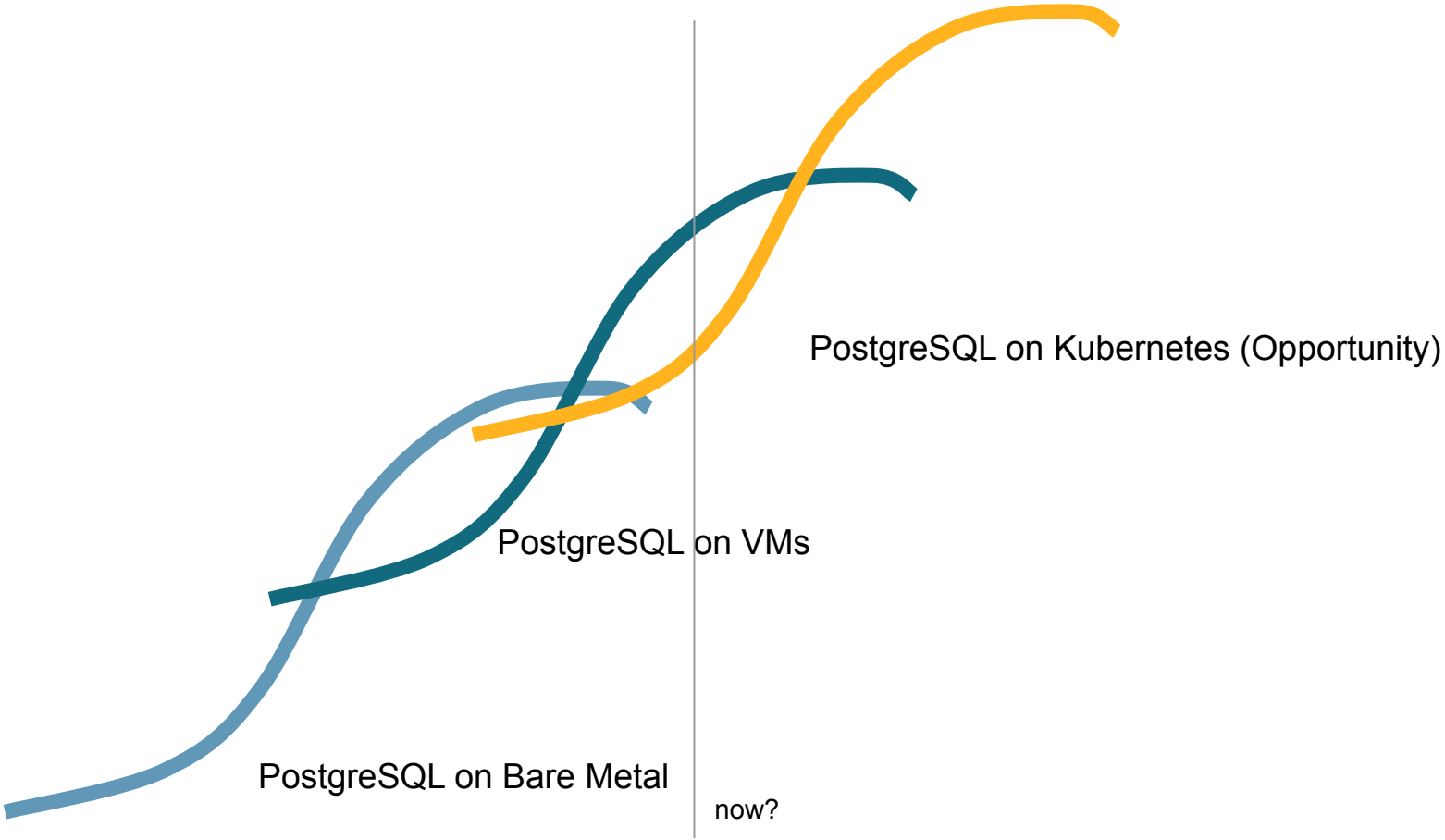
The evolution of Postgres use cases

From Handcrafted PostgreSQL to Cloud-Neutral Automation with GitOps & K8s



Suggested reading from the CNCF blog

Cloud Neutral Postgres Databases with Kubernetes and CloudNativePG



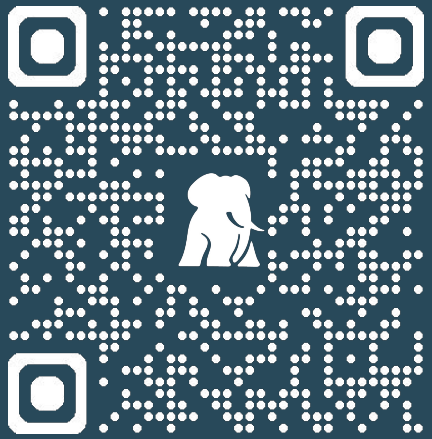
EDB

Key take aways

Your opportunity to run Postgres databases in Kubernetes starts today

- From “Kubernetes is stateless” to “Databases are #1 workloads”
- Start learning “enough” Kubernetes and become a CKA!
- Start talking about database architectures with your Kubernetes folks!
  - Keep pushing for “elephant herds” or you’ll get “cattle”!
  - Evaluate bare metal worker nodes for Postgres with local storage
- Learn CloudNativePG, you’ll love it!
  - Join our community
- Start small, aim big. Evaluate internal DBaaS through CloudNativePG.
  - Be aware of the microservice database approach
- Cloud Neutral PostgreSQL is an opportunity you cannot miss!

Questions?



gabrielebartolini.it

@_GBartolini_

Creators of  **CloudNativePG**

enterprisedb.com

cloudnative-pg.io

postgresql.org

dok.community

cncf.io