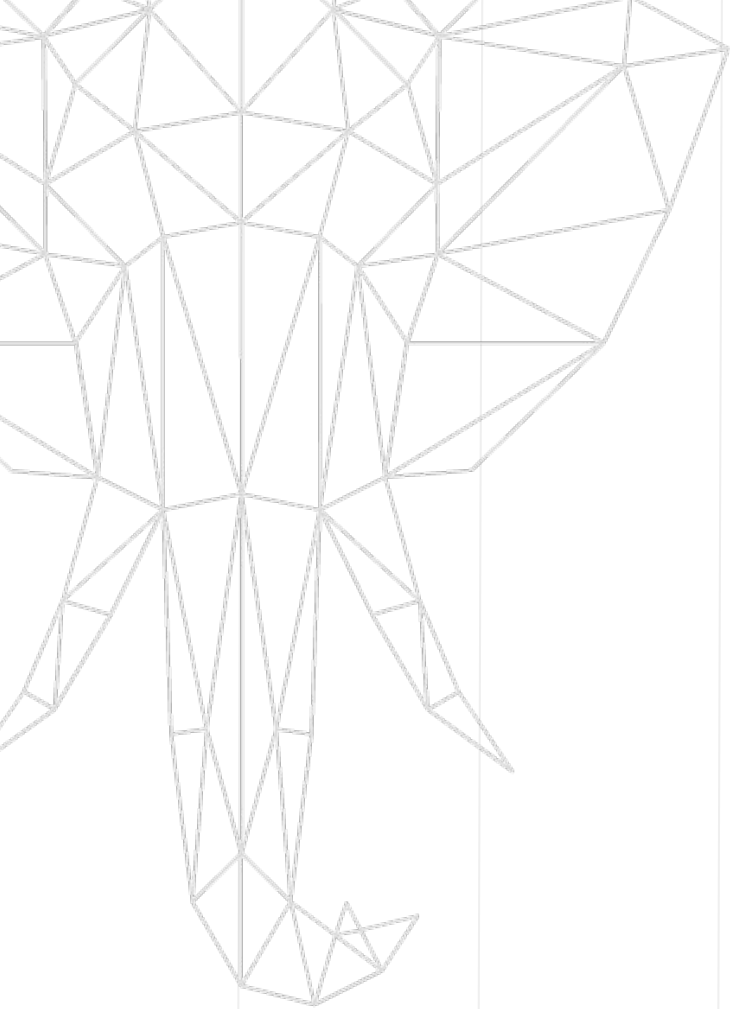


POSTGRESQL ON KUBERNETES

Álvaro Hernández

ONGRES

POSTGRESQL ON KUBERNETES

`whoami`

- Founder & CEO, OnGres
- 20+ years PostgreSQL user and DBA
- Mostly doing R&D to create new, innovative software on Postgres
- Frequent speaker at PostgreSQL, database conferences
- Principal Architect of ToroDB
- Founder and President of the NPO *Fundación PostgreSQL*
- AWS Data Hero

Álvaro Hernández

<aht@ongres.com>

@ahachete

ONGRES

POSTGRESQL ON KUBERNETES

WHAT IS KUBERNETES

(non-classical intro)

//WHAT IS KUBERNETES?

An HTTP Server

//WHAT IS KUBERNETES?

An object HTTP Server

//WHAT IS KUBERNETES?

An object HTTP Server

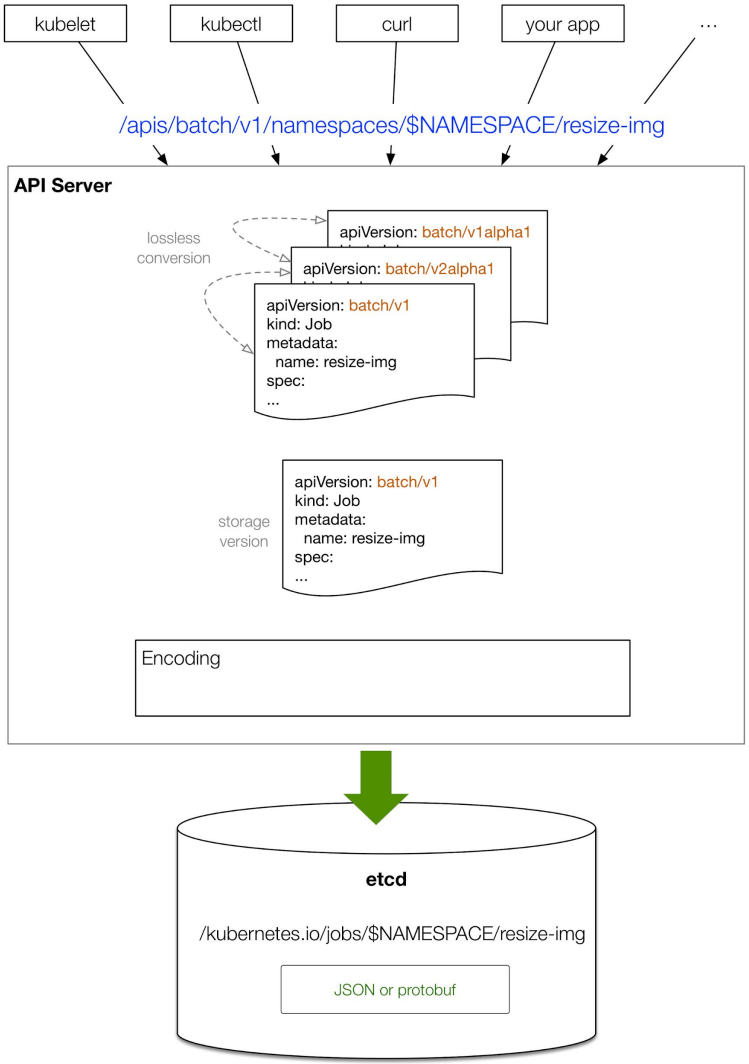
That executes actions on containers when create/modify/delete operations are run on the HTTP-exposed objects

//WHAT IS KUBERNETES?

An object HTTP Server

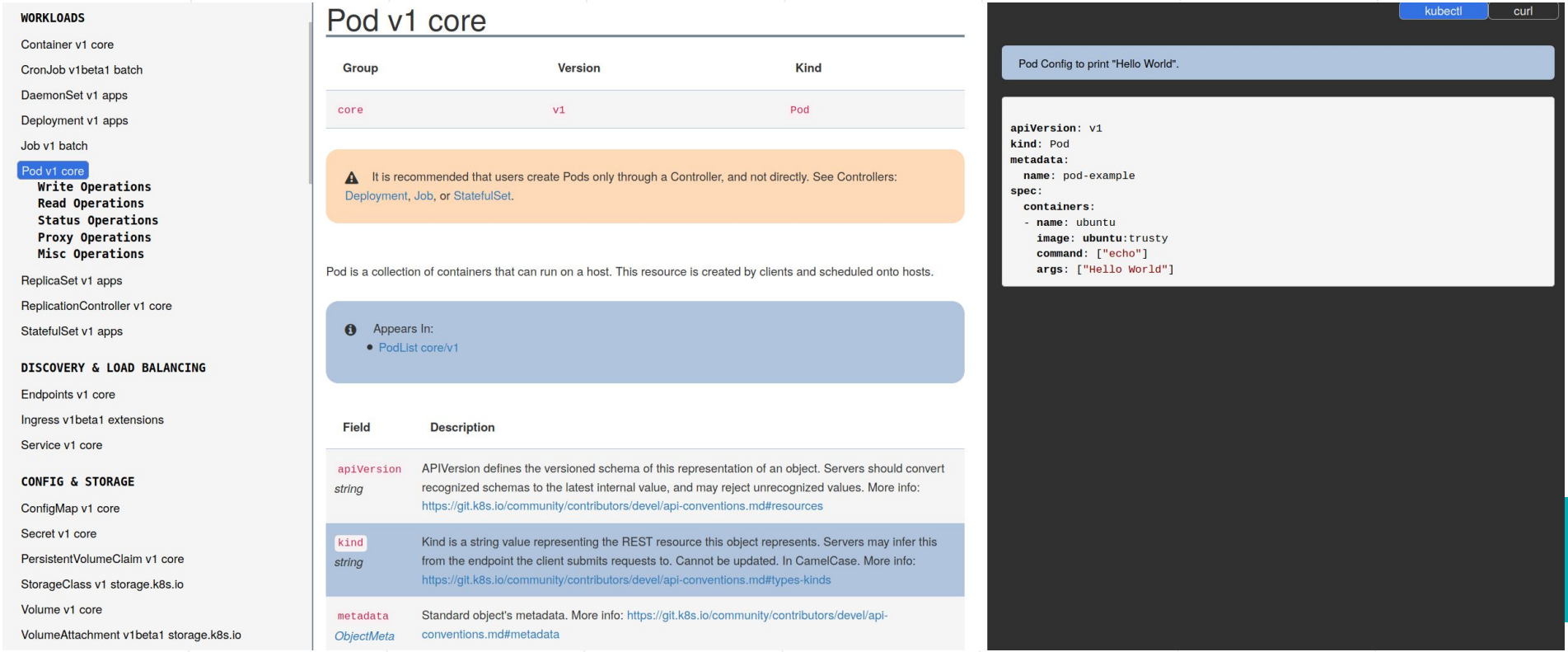
Where the URLs of the objects and their fields form an API façade that represents the actions to be taken on containers

//WHAT IS KUBERNETES?



https://blog.openshift.com/kubernetes-deep-dive-api-server-part-2

//WHAT IS KUBERNETES?



//WHY KUBERNETES?

- K8s is “the JVM” of the architecture of distributed systems: **an abstraction layer & API to deploy and automate infrastructure**.
- K8s provides APIs for nodes and IPs discovery, secret management, network proxying and load balancing, storage allocation, etc
- A PostgreSQL deployment can be fully automated!

DEMO - POSTGRESQL on K8S

(“show the code”)

TURNING POSTGRESQL CLOUD NATIVE

ONGRES

POSTGRESQL ON KUBERNETES

//CLOUD NATIVE

Cloud native applications are:

- designed to be packaged in containers
- scale and can be orchestrated for high availability

And follow cloud-native **best practices** including:

- Single-process hierarchy per container
- Sidecar containers to separate concerns
- Design for mostly ephemeral containers

//CONTAINERS ARE NOT SLIM VMS

- A container is an abstraction over a process hierarchy, with its own network, process namespaces and virtualized storage.
- But it is just a process hierarchy. Not many processes!
- No kernel, kernel modules, device drivers, no init system, bare minimum OS.
- **Should be just the binary of your process and its dynamic libraries and support files it needs.**

//IS POSTGRESQL FOR CONTAINERS?

- Overhead is minimal (1-2%): it is just a wrapper over the processes!
- Containers are as ephemeral as the process hierarchy they wrap.
- Advantage: they can be restarted somewhere if they fail.
- It's easier with stateless apps. But storage can be easily decoupled from containers: there are many storage persistence technologies.
- The entrypoint problem is typically solved by the container orchestration layer.

//MINIMAL CONTAINER IMAGE

- It's not about disk space or I/O.
  It's about security and good design principles.
- PostgreSQL binaries are minimal: container image cannot be huge.
  Remove:
  - Non-essential PostgreSQL binaries
  - Docs, psql
  - OS non system tools --all but */bin*, */sbin*, */lib**
  - Init system if any!

//LEVERAGE THE SIDECAR PATTERN

If a container should only have a single process hierarchy, how can we add support daemons like monitoring or HA agents?

- In K8s a **pod** is a set of 1+ containers that share the same namespaces, and run side-by-side on the same host.
- Sidecar pattern: deploy side functionality (like agents) to side containers (sidecars) on the same pod as PostgreSQL's container.
- Sidecars have the same IP and port space; process space (can send kill signals to processes), see the same persistent volume mount.

//HIGH AVAILABILITY (HA)

- HA is a native concept of *cloud native*.
- K8s provides mechanisms for leader election and HA.
  But are not good for PostgreSQL!
- Leader election needs to be replication lag and topology aware.
- Also need to run operations after {fail,switch}over.
- Use PostgreSQL-specific HA mechanisms.
- Use K8s to automatically restart pods if they fail, and scale replicas.

//CENTRALIZED LOGGING

- A pattern that is not exclusive to containers, but reinforced in K8s.
- DBAs need not to “login” to every container to check logs.
- Centralized logs allow to:
  - Correlate events across multiple servers (leader / replicas).
  - Manage logs persistence once.
  - Run periodic reporting and alerting processes (like pgBadger).
  - Correlate with centralized monitoring (like Prometheus).

//K8S OPERATORS: AUTOMATE POSTGRESQL OPS!

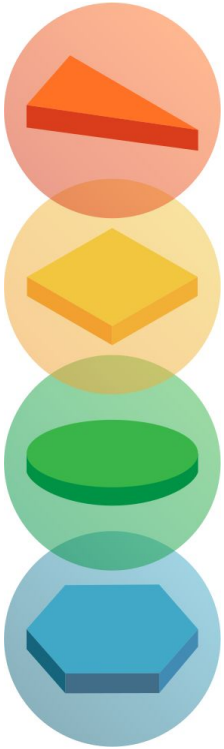
- Operators are just applications, developed for K8s
- **Understand PostgreSQL operations**
- Call K8s APIs to execute the operations
- Automate:
  - Minor version upgrades (rolling strategy)
  - Explicit vacuums
  - Repacks / reindex
  - Health checks

STACKGRES

ONGRES

POSTGRESQL ON KUBERNETES

//STACKGRES: CLOUD NATIVE POSTGRESQL



Running on Kubernetes. Embracing multi-cloud and on-premise.

Carefully selected and tuned a set of surrounding PostgreSQL components.

DB-as-a-Service without vendor lock-in. Root access.

Enterprise-grade PostgreSQL stack.

ONGRES

POSTGRESQL ON KUBERNETES

//A POSTGRES DISTRIBUTION (I)

Postgres installation is 25MB. Would you run this as-is on production?

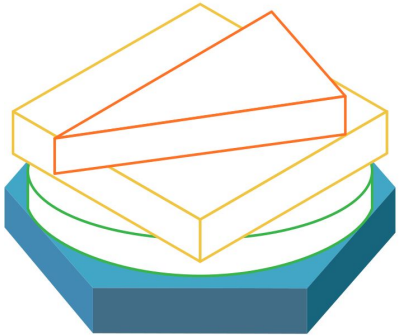
OS, filesystem, Postgres tuning	Backups and DR
Connection pooling	High Availability
Centralized logging	Log parsing / alerting
Monitoring	Health checks
Repack	Logical replication software

//A POSTGRES DISTRIBUTION (II)

- StackGres = PostgreSQL + agg(all necessary PG ecosystem tools)
- Packaged as a single, easy deployable unit.
- Strongly opinionated stack (we picked the best tools for you).
- Based on Red Hat's UBI 8 (Universal Base Image).
- Basic PostgreSQL container is just 100MB.

The kernel is to Linux, what PostgreSQL is to StackGres

//THE STACKGRES STACK (I)



UBI 8 minimal image

Vanilla PostgreSQL v11+

Persistent storage via StorageClass

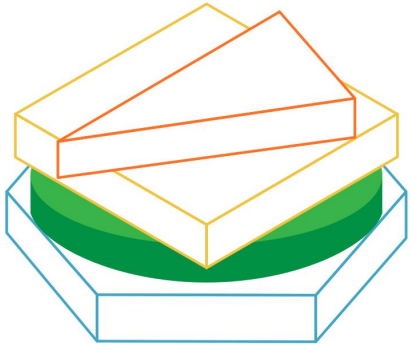
Tuned by default, user configurable

Util container

ONGRES

POSTGRESQL ON KUBERNETES

//THE STACKGRES STACK (II)



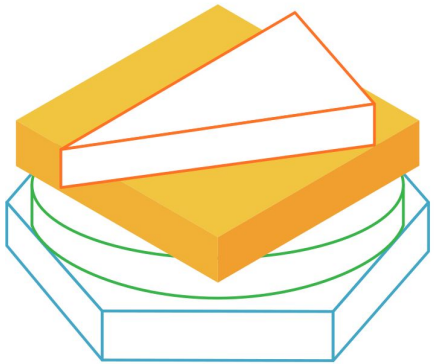
Connection pooling

Automatic Failover + HA: Patroni

Scale to any number of nodes

RW + RO stable entry points

//THE STACKGRES STACK (III)

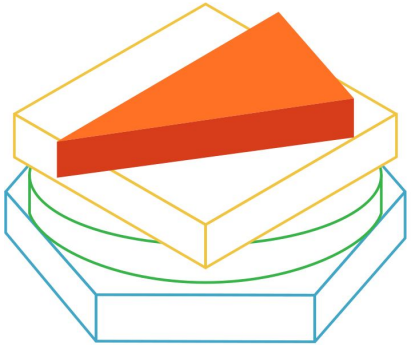


Centralized log management

Monitoring w/ Prometheus
(built-in or external)

Backup to PV for DR, PITR

//THE STACKGRES STACK (IV)



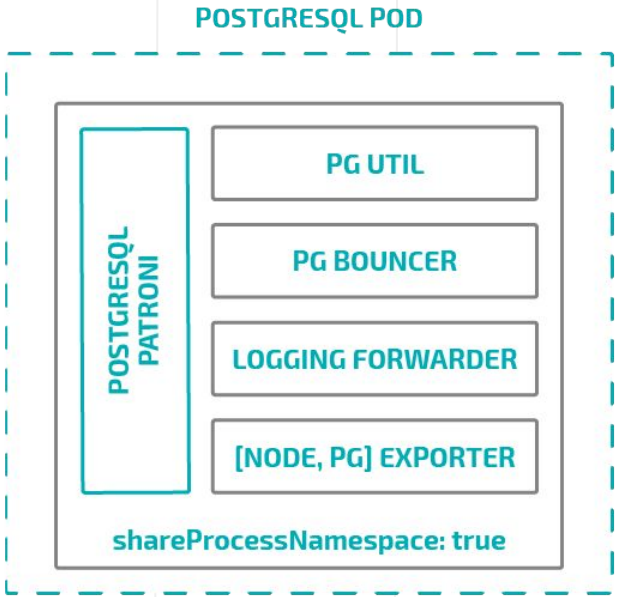
CLI & API cluster management

Web UI management interface

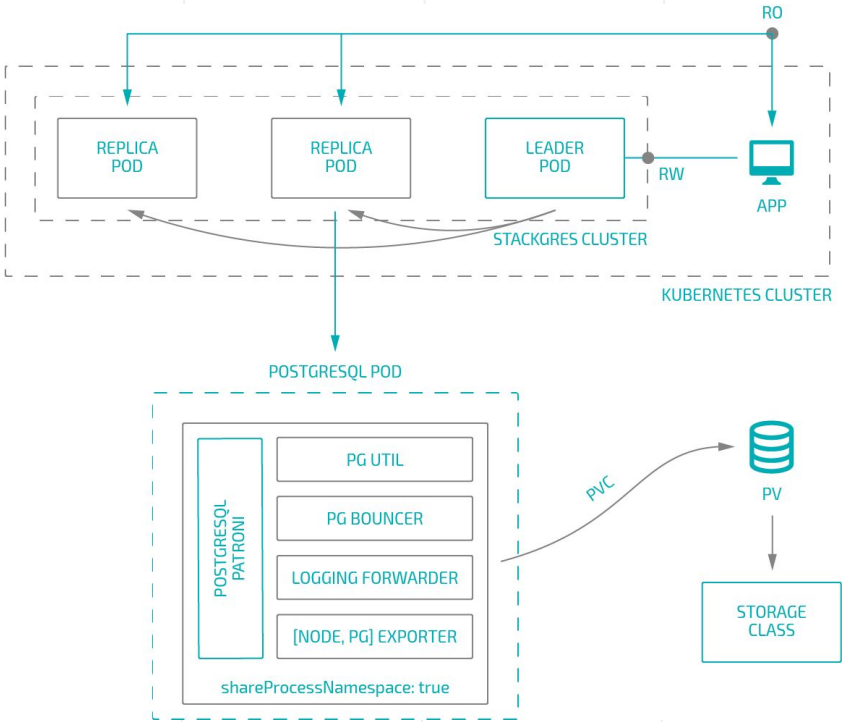
Automatic, minor version rolling upgrades

Integration with OLM

//STACKGRES ARCHITECTURE



//STACKGRES ARCHITECTURE

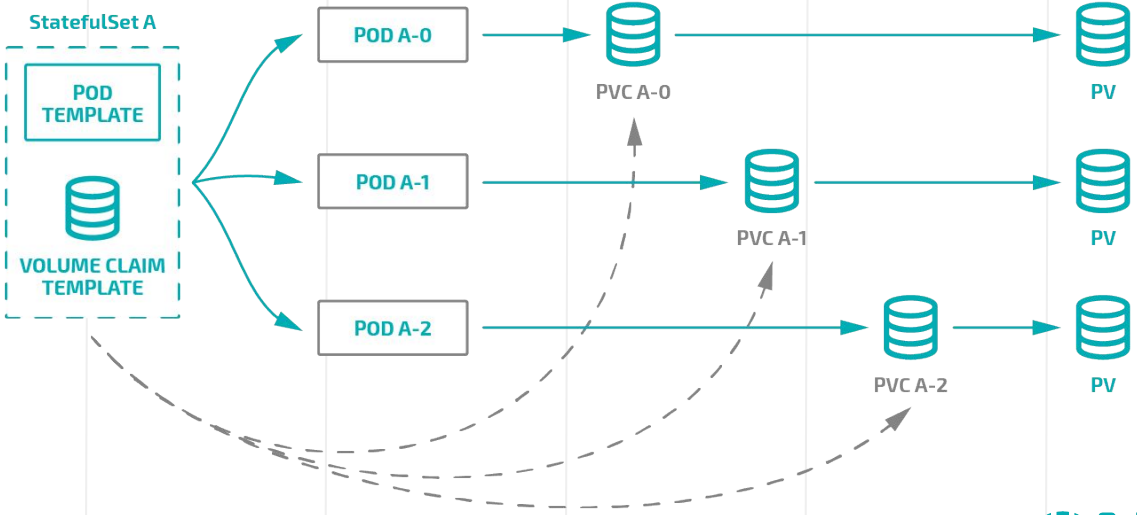


ONGRES

POSTGRESQL ON KUBERNETES

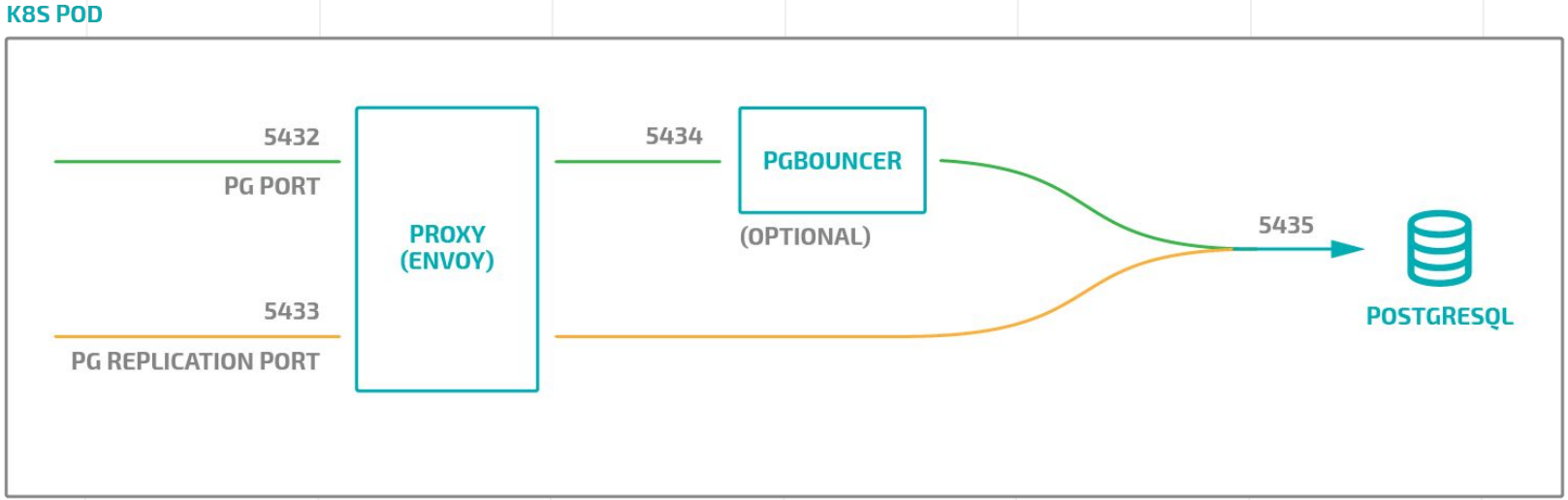
//STACKGRES ARCHITECTURE

- Storage Class behavior:



//STACKGRES ARCHITECTURE

- Networking (via Envoy)



DEMO

//THANK YOU

QUESTIONS?

Álvaro Hernández
<aht@ongres.com>
@ahachete / www.ongres.com