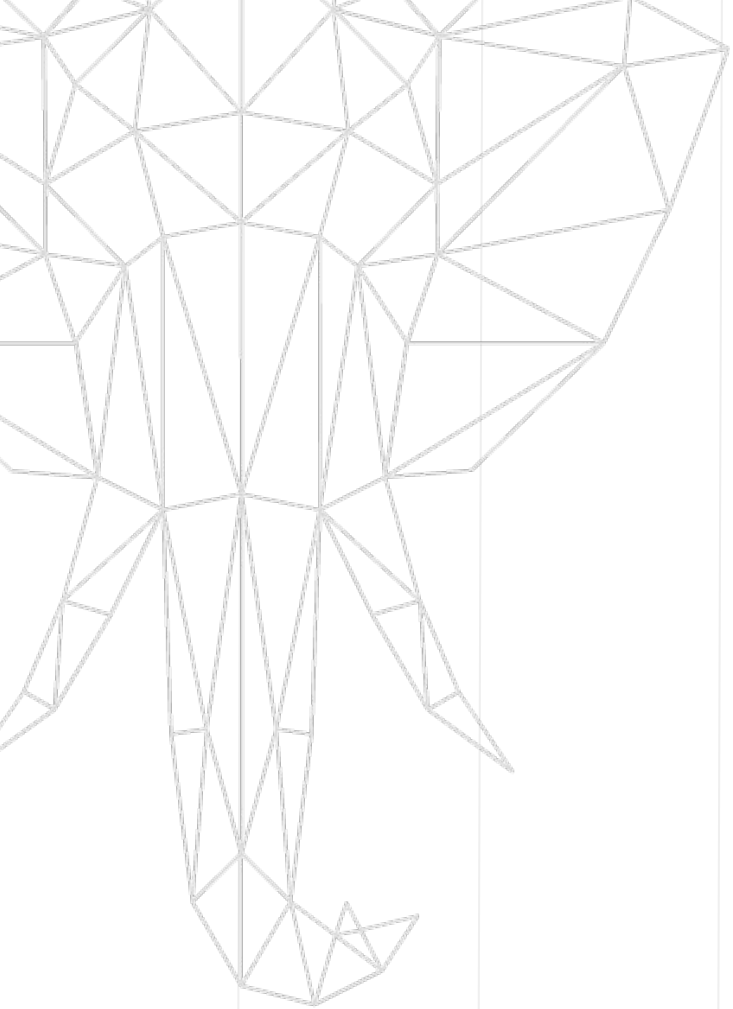


CLOUD NATIVE POSTGRESQL IN KUBERNETES

ÁLVARO HERNÁNDEZ

ONGRES

`whoami`

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

Álvaro Hernández
<aht@ongres.com>
@ahachete

PRE-DEMO

https://gitlab.com/ongresinc/stackgres-tutorial

THE “STACK” PROBLEM

//Postgres and Oracle Install Size

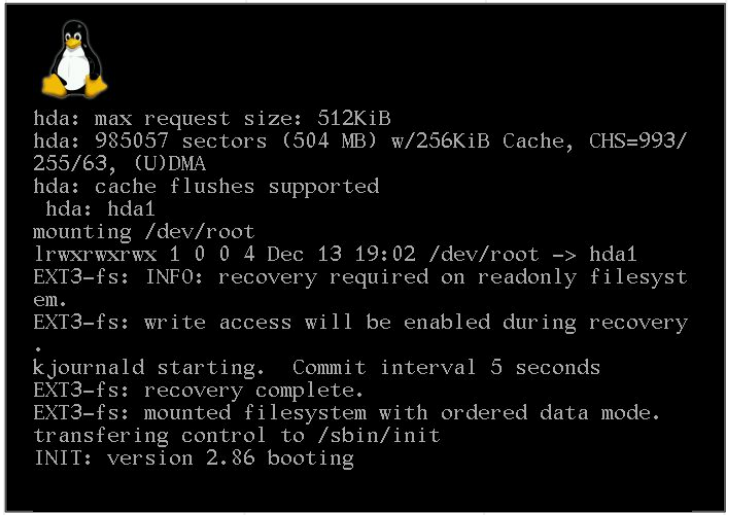
```
$ podman images --format "table {{.Repository}} {{.Tag}} {{.Size}}" \
docker.io/library/postgres
REPOSITORY                    TAG     SIZE
docker.io/library/postgres    alpine  76.9 MB
docker.io/library/postgres    12.0    356 MB
```

Index of /pub/source/v12.0/

Name↓	Last Modified:	Size:	Type:
../		-	Directory
postgresql-12.0.tar.bz2	2019-Sep-30 20:10:49	19.2M	application/x-bzip

```
$ podman images --format "table {{.Repository}} {{.Tag}} {{.Size}}" \
docker.io/store/oracle/database-enterprise
REPOSITORY                                    TAG         SIZE
docker.io/store/oracle/database-enterprise    12.2.0.1    3.46 GB
```

//Postgres Is “Just a Kernel”



Postgres is like the Linux kernel

Running Postgres in production requires “a RedHat” of Postgres. A curated set of open source components built, verified and packaged together.

ONGRES

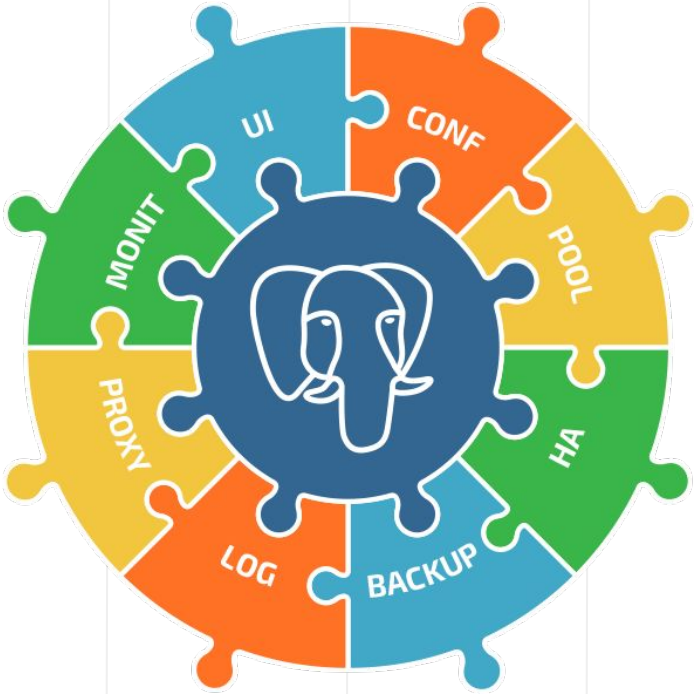
CLOUD NATIVE POSTGRESQL IN KUBERNETES

//The Postgres Ecosystem

ONGRES

CLOUD NATIVE POSTGRESQL IN KUBERNETES

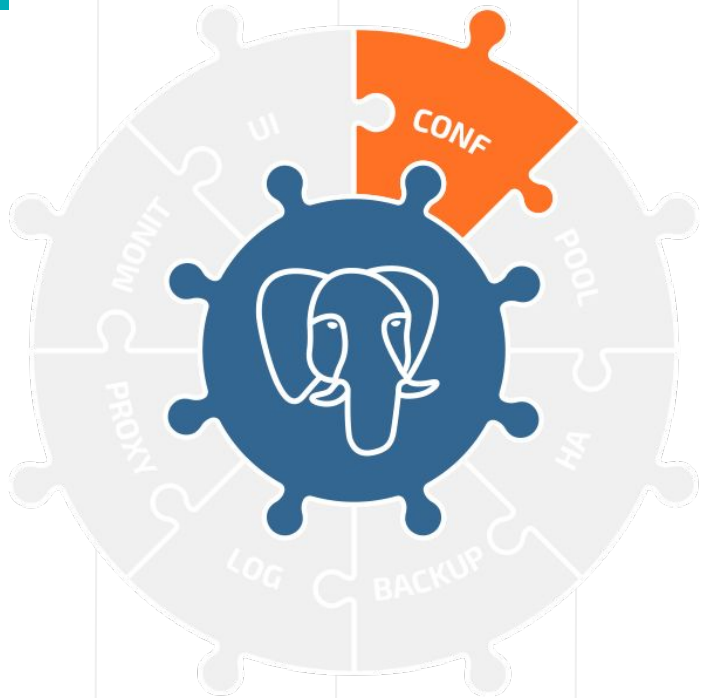
//An Enterprise-Grade Postgres Stack



ONGRES

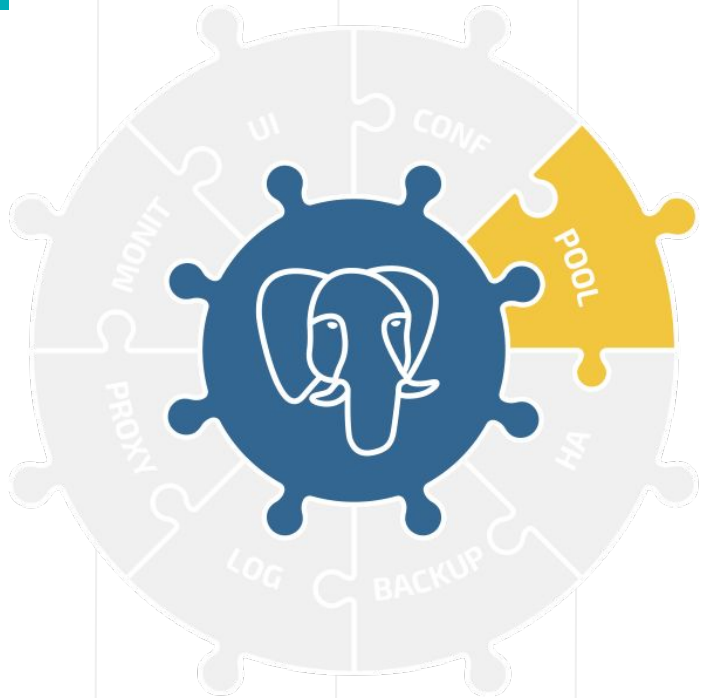
CLOUD NATIVE POSTGRESQL IN KUBERNETES

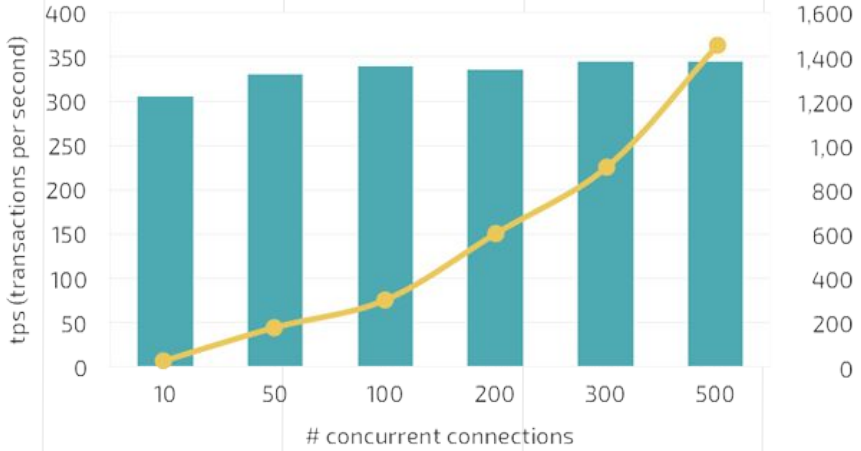
//Configuration



- OS, filesystem tuning
- PostgreSQL default configuration is very conservative.
- Resources:
  - https://postgresqlco.nf
  - PostgreSQL Configuration for *Humans*

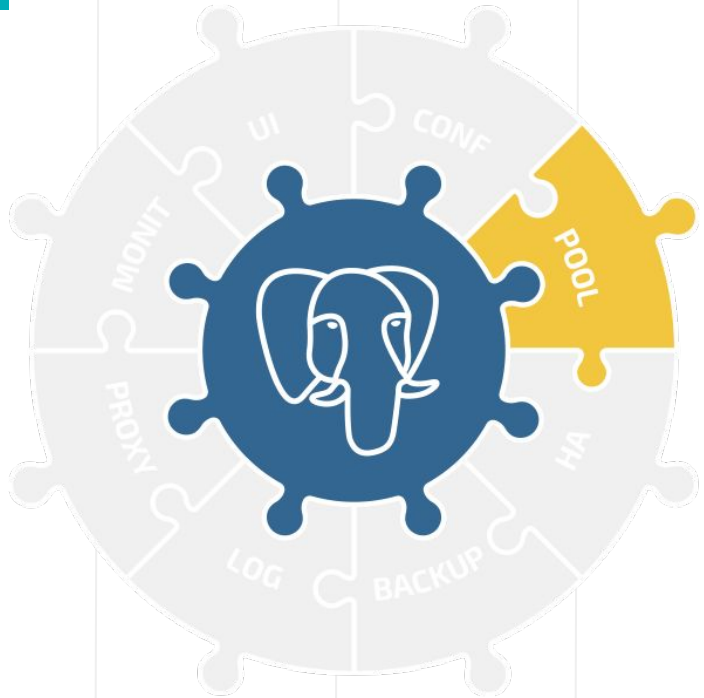
//Connection Pooling





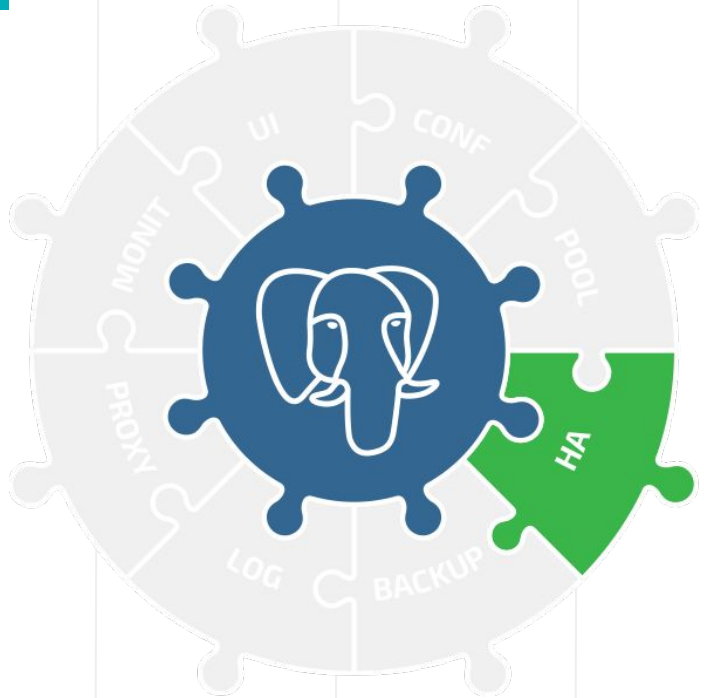
pg_bench, scale 2000, m4.large
(2 vCPU, 8GB RAM, 1k IOPS)

//Connection Pooling



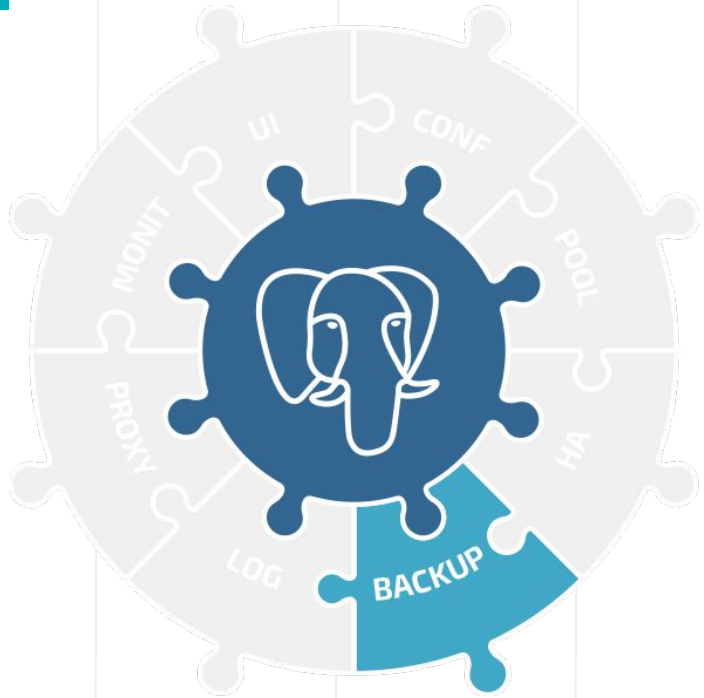
- PgPool?
- PgBouncer?
- Odyssey?
- Pgagroal?
- Where do we place the pool?
  - Client-side
  - Server-side
  - Middle-ware
  - Some or all of the above

//High Availability



- ~~Manual~~?
- PgPool?
- Repmgr?
- Patroni?
- pg_autofailover?
- PAF?
- Stolon?

//Backups and DR

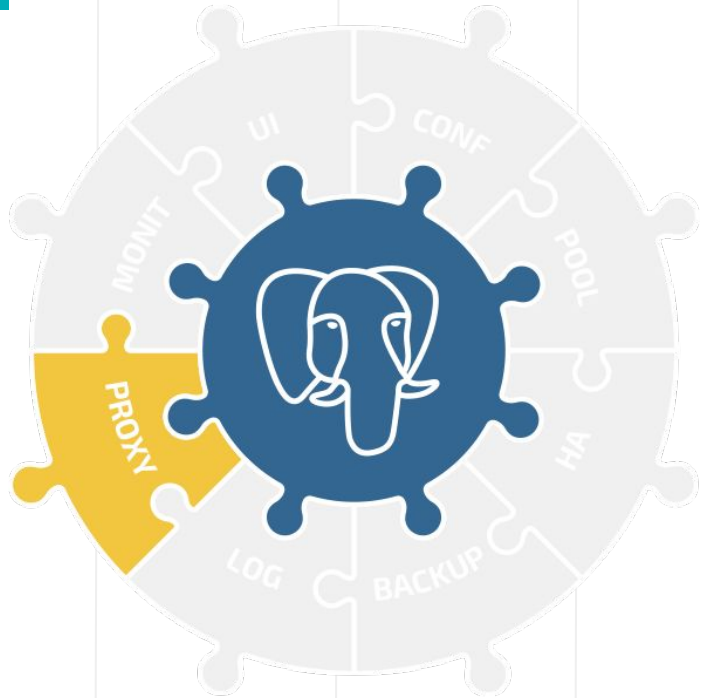


- ~~pg_dump~~?
- Barman?
- Pgbackrest?
- Wal-e / Wal-g?
- pg_probackup?
- To disk? To cloud storage?

//Centralized Logging

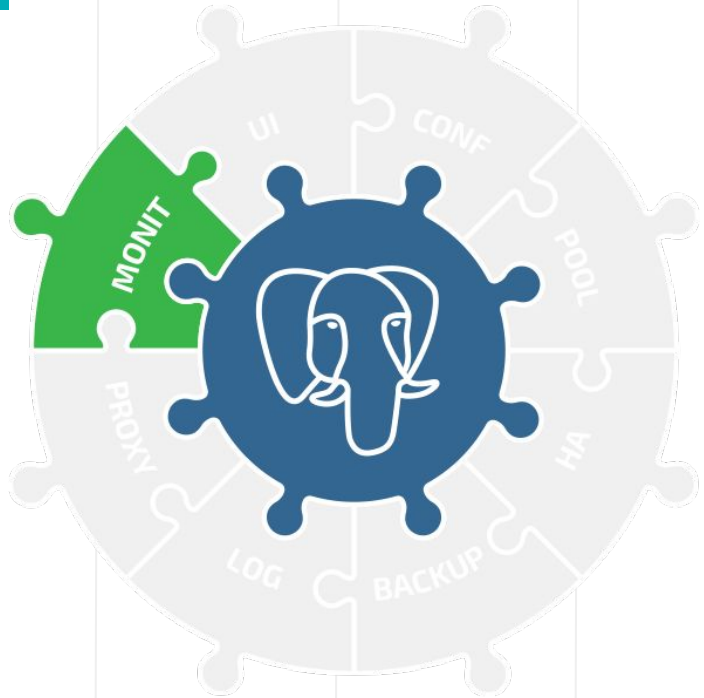
- ~~Logs on every server~~
- There is not a good solution for this
- Cloud-native solutions like fluentd or Loki may work
- Store the logs on Timescale

//Network Proxy. Entrypoint Problem



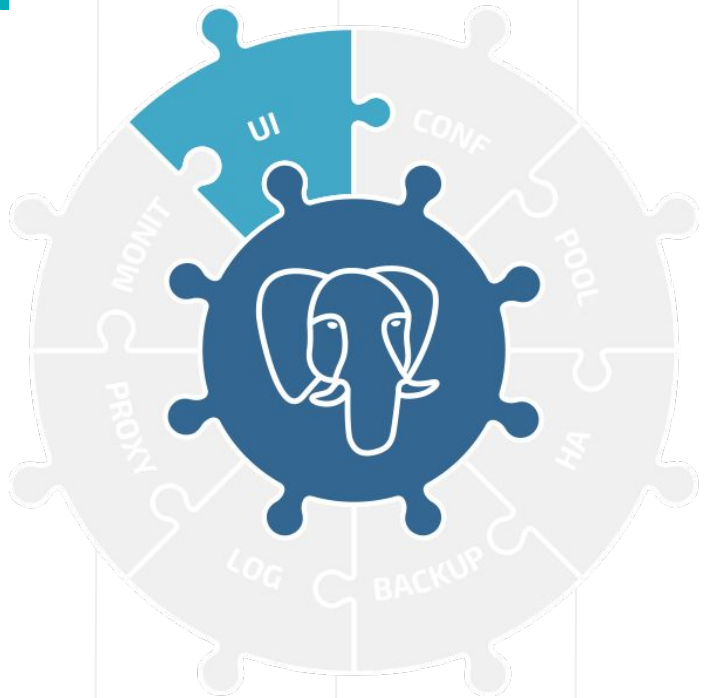
- **Entrypoint**: how do I locate the master, if it might be changing?
- How do I obtain traffic metrics?
- Is it possible to manage traffic: duplicate, A/B to test clusters, or even inspect it?
- Offload TLS?

//Monitoring



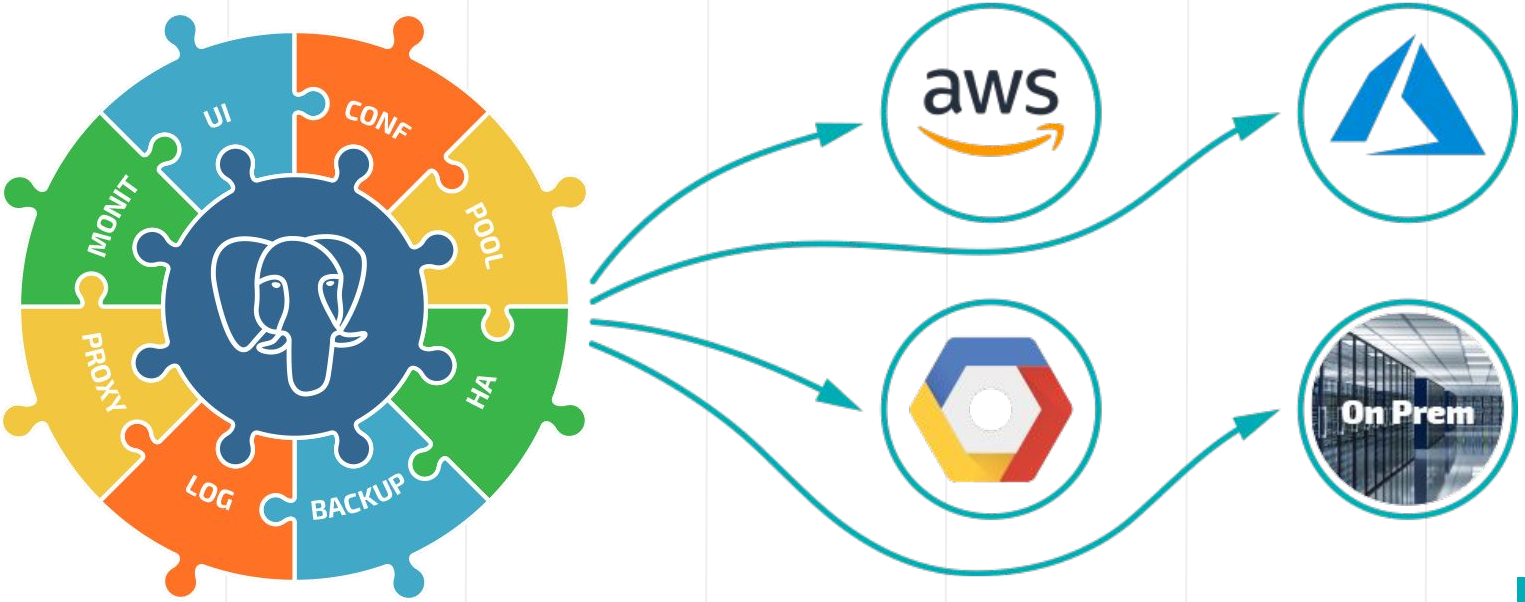
- Zabbix?
- Okmeter?
- Pganalyze?
- Pgwatch2?
- PoWA?
- New Relic?
- DataDog?
- Prometheus?

//Management Interface



- There are no tools like OEM…
- UI oriented towards cluster management
- ClusterControl?
- Elephant Shed?

//Where Do We Deploy The Stack?



ONGRES

CLOUD NATIVE POSTGRESQL IN KUBERNETES

DEPLOYING THE POSTGRES STACK ON KUBERNETES

ONGRES

CLOUD NATIVE POSTGRESQL IN KUBERNETES

//Why Kubernetes?

<Really, really short introduction to 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!

//K8s Operators: Automate Postgres Ops!

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

//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.**

TURNING POSTGRESQL CLOUD NATIVE

ONGRES

CLOUD NATIVE POSTGRESQL IN KUBERNETES

//Is Postgres 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 Postgres!
- 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

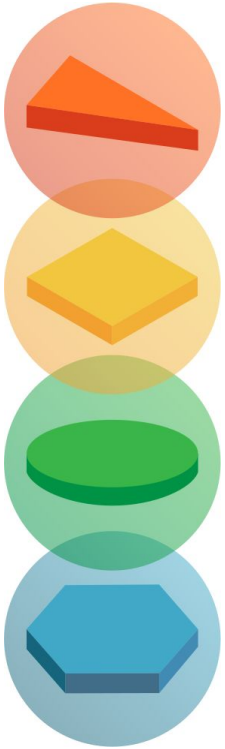
- A pattern that is not exclusive to containers, but reinforced in K8s.
- DBAs need not to “login” to every container to check logs.
- Centralized logs allow to:
  - Correlate events across multiple servers (leader / replicas).
  - Manage logs persistence once.
  - Run periodic reporting and alerting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

STACKGRES

ONGRES

CLOUD NATIVE POSTGRESQL IN KUBERNETES

//StackGres: Cloud Native Postgres



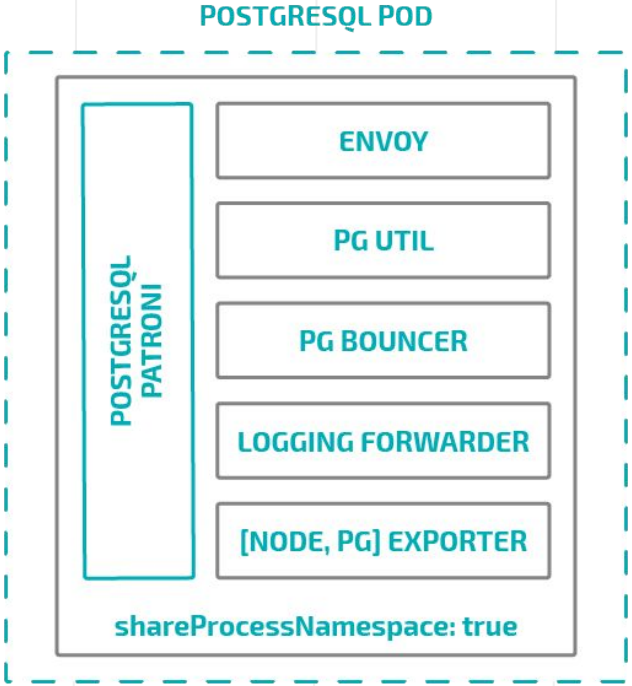
Running on Kubernetes. Embracing multi-cloud and on-premise.

Enterprise-grade, highly opinionated Postgres stack.

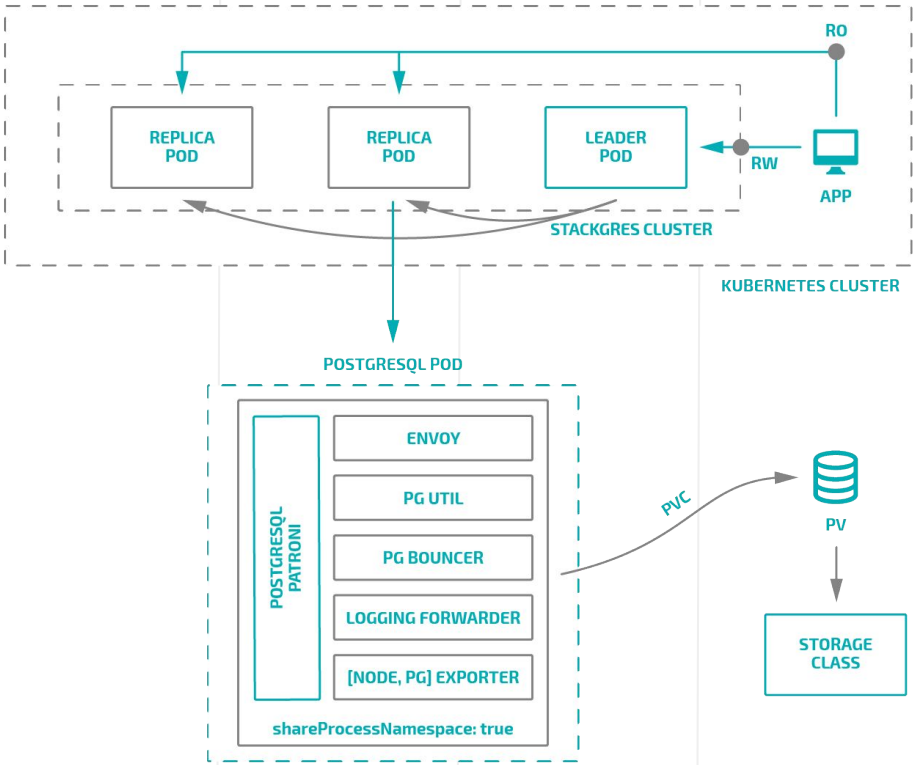
DB-as-a-Service without vendor lock-in. Root access.

Open source!

//StackGres Architecture



//StackGres Architecture

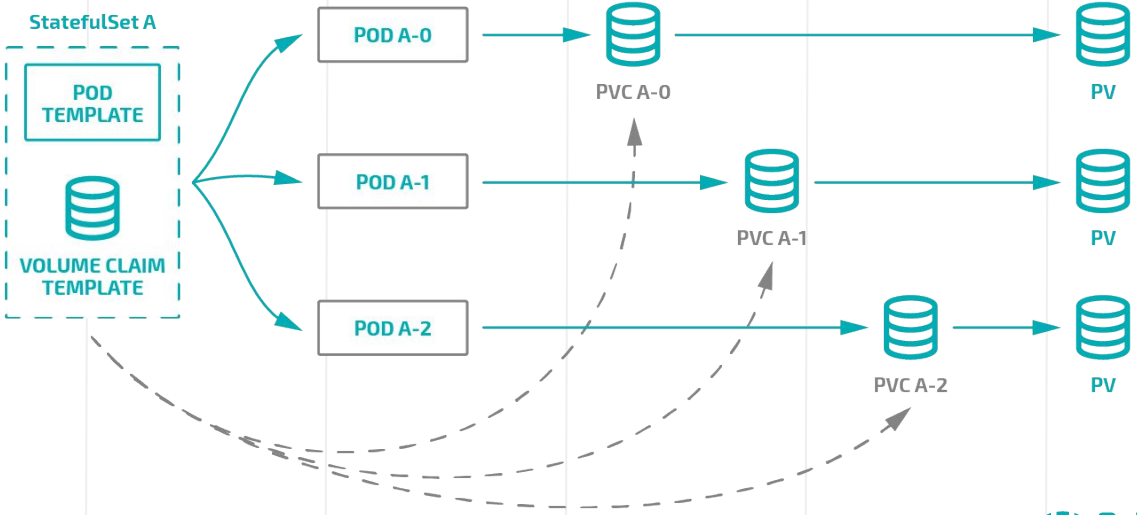


ONGRES

CLOUD NATIVE POSTGRESQL IN KUBERNETES

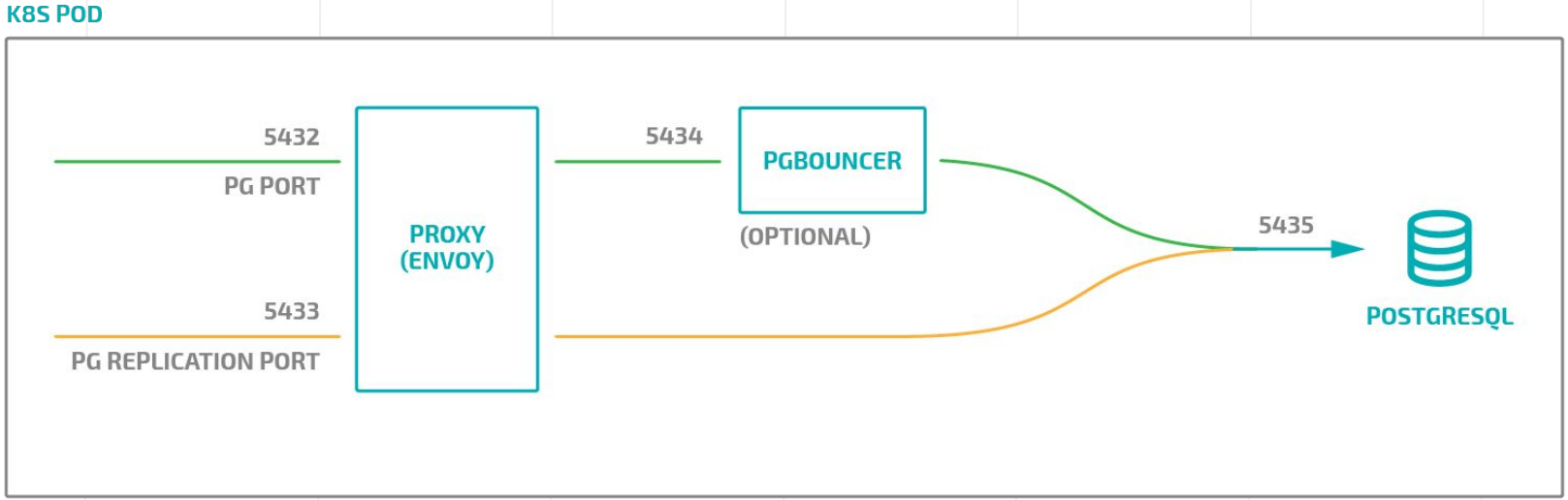
//StackGres Architecture

- Storage Class behavior:



//StackGres Architecture

- Networking



DEPLOY A POSTGRESQL-aaS WITH STACKGRES

ONGRES

CLOUD NATIVE POSTGRESQL IN KUBERNETES

//CRDs: StackGres “API”

- CRDs are Kubernetes custom objects (Custom Resource Definition).
- StackGres creates the CRDs and uses them extensively. An instance of a CRD is a “CR”.
- They define high-level concepts, such as a Postgres Cluster.
- **No need to install any separate tool or CLI: CRDs are our API, use *kubectl* to communicate with StackGres**.
- CRs are bi-directional: you specify in the *spec* part what you want; StackGres will report in the *status* field extra information.
- Some CRs may be created by StackGres, like automatic backups

DEMO

https://gitlab.com/ongresinc/stackgres-tutorial

//STACKGRES.IO

STACKGRES.io

https://stackgres.io

https://gitlab.com/ongresinc/stackgres