



# Kubernetes from the database out



Alastair Turner

# Kubernetes from the database out

PGConf Europe  
October 2025

# Our demo environment



 DB Details

Basic Information

Upgrade

|           |             |
|-----------|-------------|
| Type      | PostgreSQL  |
| Name      | postgres-17 |
| Namespace | everest     |
| Version   | 17.2        |

Connection Details

|          |                         |
|----------|-------------------------|
| Host     | 34.39.52.136            |
| Port     | 5432                    |
| Username | postgres                |
| Password | 3C5aHoWvjJ4^6)Fh/=TbI3K |

Connection URL

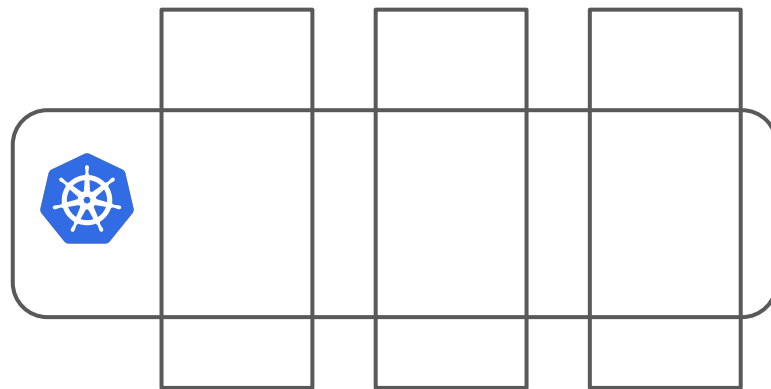
- Postgres cluster (3 nodes)
  - Postgres 17.2
  - Running on GKE
- Deployed using Percona Everest
  - Percona open source platform automate database:
    - Provisioning
    - Management
  - Get involved : [github.com/percona/everest](https://github.com/percona/everest)

♪ Got to get them disambiguated ♪

Deployment



Cluster



## From the database terminal

```
minion@thwimpad:~$ psql -h 34.142.109.174 -U postgres
Password for user postgres:
psql (17.6 (Ubuntu 17.6-2.pgdg24.04+1), server 17.4 - Percona Server for PostgreSQL 17.4.1)
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression: off, A
LPN: none)
Type "help" for help.

postgres=# SELECT pid, datid, username, application_name, client_addr
postgres=#   FROM pg_stat_activity
postgres=#   WHERE pid = pg_backend_pid();
  pid | datid | username | application_name | client_addr
-----+-----+-----+-----+-----
 35273 |     5 | postgres | psql              | 10.60.2.53
(1 row)

postgres=# SHOW listen_addresses;
listen_addresses
-----
*
(1 row)

postgres=#
```

## From the Kubernetes terminal

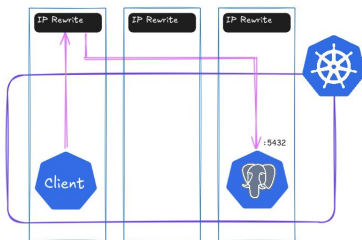
```
minion@thwimpad:~$ kubectl get services -n everest
```

| NAME                            | AGE | TYPE         | CLUSTER-IP     | EXTERNAL-IP    | PORT |
|---------------------------------|-----|--------------|----------------|----------------|------|
| percona-xtradb-cluster-operator | 11d | ClusterIP    | 34.118.231.82  | <none>         | 443  |
| pg-upgrade-me-ha                | 18h | ClusterIP    | 34.118.225.81  | <none>         | 543  |
| pg-upgrade-me-ha-config         | 18h | ClusterIP    | None           | <none>         | <no  |
| pg-upgrade-me-pgbouncer         | 18h | LoadBalancer | 34.118.228.10  | 34.142.109.174 | 543  |
| pg-upgrade-me-pods              | 18h | ClusterIP    | None           | <none>         | <no  |
| pg-upgrade-me-primary           | 18h | ClusterIP    | None           | <none>         | 543  |
| pg-upgrade-me-replicas          | 18h | ClusterIP    | 34.118.233.194 | <none>         | 543  |

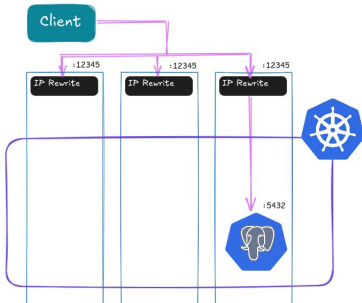
```
minion@thwimpad:~$
```

# Connecting

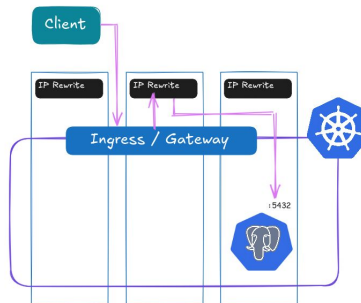
clusterIP



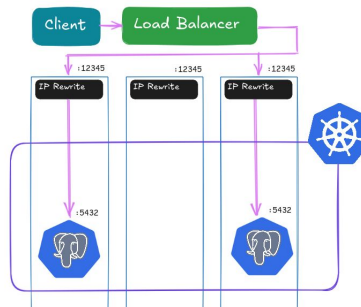
nodePort



Gateway / Ingress



loadBalancer



Kubernetes services provide a stable address to connect to

**KubeProxy** configures kernel networking

Types build on each other:  
**ClusterIP** – Address in cluster's internal address space  
**NodePort** – Port exposed on each node in the cluster  
**LoadBalancer** – Network load balancer across NodePorts

Gateway and Ingress build on ClusterIP in a different direction, not currently used for Postgres on K8s

## From the Kubernetes terminal

```
app.kubernetes.io/instance=pg-upgrade-me
app.kubernetes.io/managed-by=percona-postgresql-operator
app.kubernetes.io/name=percona-postgresql
app.kubernetes.io/part-of=percona-postgresql
pgv2.percona.com/version=2.6.0
postgres-operator.crunchydata.com/cluster=pg-upgrade-me
postgres-operator.crunchydata.com/role=pgbouncer
Annotations:      cloud.google.com/neg: {"ingress":true}
Selector:          postgres-operator.crunchydata.com/cluster=pg-upgrade-me,postgres-operator.crunchydata.com/role=pgbouncer
Type:              LoadBalancer
IP Family Policy:  SingleStack
IP Families:       IPv4
IP:                34.118.228.10
IPs:               34.118.228.10
LoadBalancer Ingress: 34.142.109.174 (VIP)
Port:              pgbouncer 5432/TCP
TargetPort:        pgbouncer/TCP
NodePort:          pgbouncer 31102/TCP
Endpoints:         10.60.12.42:5432,10.60.2.53:5432,10.60.0.14:5432
Session Affinity:  None
External Traffic Policy: Cluster
Internal Traffic Policy: Cluster
Events:            <none>
minion@thwimpad:~$
```

## From the database terminal

```
minion@thwimpad:~$ psql -h 34.142.109.174 -U postgres
Password for user postgres:
psql (17.6 (Ubuntu 17.6-2.pgdg24.04+1), server 17.4 - Percona Server for PostgreSQL 17.4.1)
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression: off, ALPN: none)
Type "help" for help.

postgres=# SELECT pid, usesysid, application_name, client_addr FROM pg_stat_replication;
 pid | usesysid | application_name | client_addr
-----+-----+-----+-----
  186 |    16384 | pg-upgrade-me-instance1-gbxv-0 | 10.60.2.52
  299 |    16384 | pg-upgrade-me-instance1-5xft-0 | 10.60.12.43
(2 rows)

postgres=#
```

# Querying

Data is stored on Persistent Volumes

The database backend gets access to the volume through a Claim (**PVC**)

Persistent Volumes are not usually created manually, but by making a claim

Container Storage Interface (**CSI**) provides the link between Kubernetes operations and the storage infrastructure



Vegas

Miami

New York

AWS

Azure

VMware

...

## From the database terminal

```
.4.1)
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression: off, A
LPN: none)
Type "help" for help.

postgres=# SHOW data_directory;
 data_directory
-----
 /pgdata/pg17
(1 row)

postgres=# SHOW temp_tablespace;
 temp_tablespace
-----

(1 row)

postgres=# SELECT spcname, pg_tablespace_location(oid) FROM pg_tablespace;
 spcname | pg_tablespace_location
-----+-----
 pg_default |
 pg_global |
(2 rows)

postgres=#
```

## From the Kubernetes terminal

```

Volumes:
  cert-volume:
    Type:          Projected (a volume that contains injected data from multiple sources
    )
    SecretName:    pg-upgrade-me-cluster-cert
    Optional:      false
    SecretName:    pg-upgrade-me-replication-cert
    Optional:      false
  postgres-data:
    Type:          PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the s
ame namespace)
    ClaimName:     pg-upgrade-me-instance1-5xft-pgdata
    ReadOnly:      false
  database-containerinfo:
    Type:          DownwardAPI (a volume populated by information about the pod)
    Items:
      limits.cpu -> cpu_limit
      requests.cpu -> cpu_request
      limits.memory -> mem_limit
      requests.memory -> mem_request
      metadata.labels -> labels
      metadata.annotations -> annotations
  pgbackrest-server:
    Type:          Projected (a volume that contains injected data from multiple sources
  )

```

## From the Kubernetes terminal

```
minion@thwimpad:~$ kubectl get pvc pg-upgrade-me-instance1-5xft-pgdata -n everest -o json | jq '.spec'
```

```
{
  "accessModes": [
    "ReadWriteOnce"
  ],
  "resources": {
    "requests": {
      "storage": "25Gi"
    }
  },
  "storageClassName": "standard-rwo",
  "volumeMode": "Filesystem",
  "volumeName": "pvc-13d2c5e9-e624-416c-8230-d6304ea89581"
}
```

```
minion@thwimpad:~$ kubectl get storageclasses -n everest
```

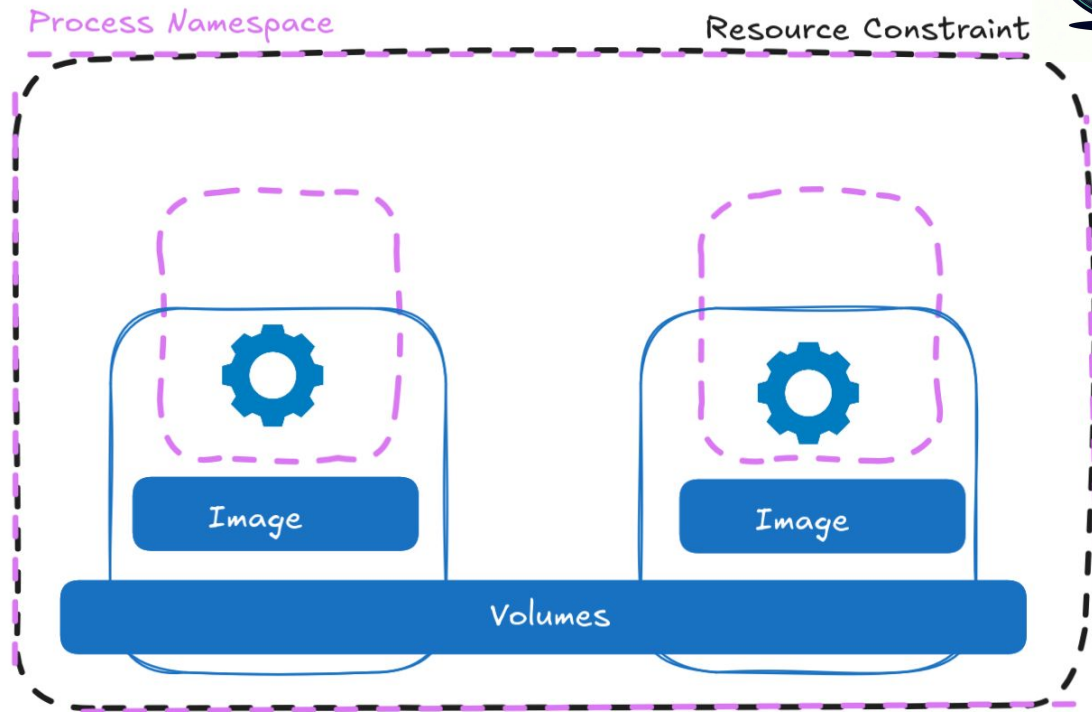
| NAME                   | PROVISIONER           | RECLAIMPOLICY | VOLUMEBINDINGMODE    |
|------------------------|-----------------------|---------------|----------------------|
| ALLOWVOLUMEEXPANSION   | AGE                   |               |                      |
| premium-rwo            | pd.csi.storage.gke.io | Delete        | WaitForFirstConsumer |
| true                   | 11d                   |               |                      |
| standard               | kubernetes.io/gce-pd  | Delete        | Immediate            |
| true                   | 11d                   |               |                      |
| standard-rwo (default) | pd.csi.storage.gke.io | Delete        | WaitForFirstConsumer |
| true                   | 11d                   |               |                      |

```
minion@thwimpad:~$
```

# A digression concerning pods



# Peas in a ...



A **pod** is a group of containers managed together

Most pods have only one container

Resource constraints are applied across everything in the pod

Each container has a mount namespace with its image and the pod volumes

Process namespaces may be shared

## From the Kubernetes terminal

```
minion@thwimpad:~$ kubectl get pods -l app.kubernetes.io/component=pg -n everest
NAME                                READY   STATUS    RESTARTS   AGE
pg-upgrade-me-instance1-5xft-0     4/4     Running   0           19h
pg-upgrade-me-instance1-gbxv-0     4/4     Running   0           19h
pg-upgrade-me-instance1-psdw-0     4/4     Running   0           19h
minion@thwimpad:~$ kubectl get pod pg-upgrade-me-instance1-5xft-0 -n everest \
> -o json | jq '.spec.containers[].name'
"database"
"replication-cert-copy"
"pgbackrest"
"pgbackrest-config"
minion@thwimpad:~$
```

# Oops, now what?

Resilience through automated recovery

Containers within a pod restarted

Pods recreated

Pods in a **StatefulSet** recreated with same address and persistent storage

StatefulSet has no information about pod's role in database



## Deploying, scaling, ...

- Operators add functions on top of composing a database service from core Kubernetes features
  - Describe the database service in DB terms, not infra terms
  - Application/DB specific monitoring and reconciliation in a control loop
- Interface to this extensibility through Custom Resources and Custom Resource Definitions (**CRDs**)
  - Custom Resources also exposed for backups
- Clearest benefits of Kubernetes and Operators come from automated and repeatable operational tasks
  - Scaling
  - Rolling upgrades
- Operators may take control of recreating pods, replacing StatefulSet functionality

# Your logs on Kubernetes

- Logs written to stdout / stderr
- Stored on each node
- No built-in cluster-wide aggregation
- Viewed through Kubernetes command line tooling
- Plugins to help with formatting, highlighting, ...

## From the database terminal

```
minion@thwimpad:~$ psql -h 34.142.109.174 -U postgres
Password for user postgres:
psql (17.6 (Ubuntu 17.6-2.pgdg24.04+1), server 17.4 - Percona Server for PostgreSQL 17.4.1)
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression: off, A
LPN: none)
Type "help" for help.

postgres=# SHOW log_destination;
 log_destination
-----
 stderr
(1 row)

postgres=# SHOW log_rotation_size;
 log_rotation_size
-----
 0
(1 row)

postgres=#
```

## From the Kubernetes terminal

```
minion@thwimpad:~$ kubectl logs pg-upgrade-me-instance1-5xft-0 -n everest | tail -n 10
Defaulted container "database" out of: database, replication-cert-copy, pgbackrest, pg
backrest-config, postgres-startup (init), nss-wrapper-init (init)
2025-09-30 15:56:16,172 INFO: no action. I am (pg-upgrade-me-instance1-5xft-0), a seco
ndary, and following a leader (pg-upgrade-me-instance1-psdw-0)
2025-09-30 15:56:26,175 INFO: no action. I am (pg-upgrade-me-instance1-5xft-0), a seco
ndary, and following a leader (pg-upgrade-me-instance1-psdw-0)
2025-09-30 15:56:36,174 INFO: no action. I am (pg-upgrade-me-instance1-5xft-0), a seco
ndary, and following a leader (pg-upgrade-me-instance1-psdw-0)
2025-09-30 15:56:46,174 INFO: no action. I am (pg-upgrade-me-instance1-5xft-0), a seco
ndary, and following a leader (pg-upgrade-me-instance1-psdw-0)
2025-09-30 15:56:56,172 INFO: no action. I am (pg-upgrade-me-instance1-5xft-0), a seco
ndary, and following a leader (pg-upgrade-me-instance1-psdw-0)
2025-09-30 15:57:06,169 INFO: no action. I am (pg-upgrade-me-instance1-5xft-0), a seco
ndary, and following a leader (pg-upgrade-me-instance1-psdw-0)
2025-09-30 15:57:16,173 INFO: no action. I am (pg-upgrade-me-instance1-5xft-0), a seco
ndary, and following a leader (pg-upgrade-me-instance1-psdw-0)
2025-09-30 15:57:26,173 INFO: no action. I am (pg-upgrade-me-instance1-5xft-0), a seco
ndary, and following a leader (pg-upgrade-me-instance1-psdw-0)
2025-09-30 15:57:36,168 INFO: no action. I am (pg-upgrade-me-instance1-5xft-0), a seco
ndary, and following a leader (pg-upgrade-me-instance1-psdw-0)
2025-09-30 15:57:46,175 INFO: no action. I am (pg-upgrade-me-instance1-5xft-0), a seco
ndary, and following a leader (pg-upgrade-me-instance1-psdw-0)
minion@thwimpad:~$
```

## In summary



- A database on Kubernetes package consists of
  - Container images for the database engine and supporting software
  - An **Operator** which automates operations
- Cluster controller and Operator drive actions to reach target config
  - At initial deployment and in case of component failure
  - Desired state described in a Custom Resource Definition (**CRD**)

## In further summary

- Database processes run in **Pods**
  - of containers which are managed together
- Data is stored on Persistent Volumes
  - Mounted into each pod by a Claim on the Persistent Volume (**PVC**)
- In case of failure
  - Containers are restarted
  - Pods are recreated
- Logs are collected on each Kubernetes node



# Questions?



How did I do?





# Thank you!