zalando

Blue elephant on-demand:

PostgreSQL + Kubernetes

FOSDEM 2018, Brussels

Oleksii Kliukin, Jan Mußler

03-02-2018

SELECT title FROM agenda;

DBaaS at Zalando

UI and monitoring

PostgreSQL on Kubernetes

Kubernetes-native Patroni

Postgres operator

About us

Oleksii Kliukin oleksii.kliukin@zalando.de
Database Engineer @ Zalando
Twitter: @hintbits

Jan Mussler jan.mussler@zalando.de
Engineering Lead @ Zalando
Twitter: @JanMussler

ZALANDO AT A GLANCE

>300 databases

In data centers

> 150

Postgres clusters on AWS EC2

> 200

Postgres clusters on Kubernetes

as at May 2017

zalando

Running PostgreSQL in two data centers

Bare metal with LXC containers

Single Git repository with all configs

Database discovery service

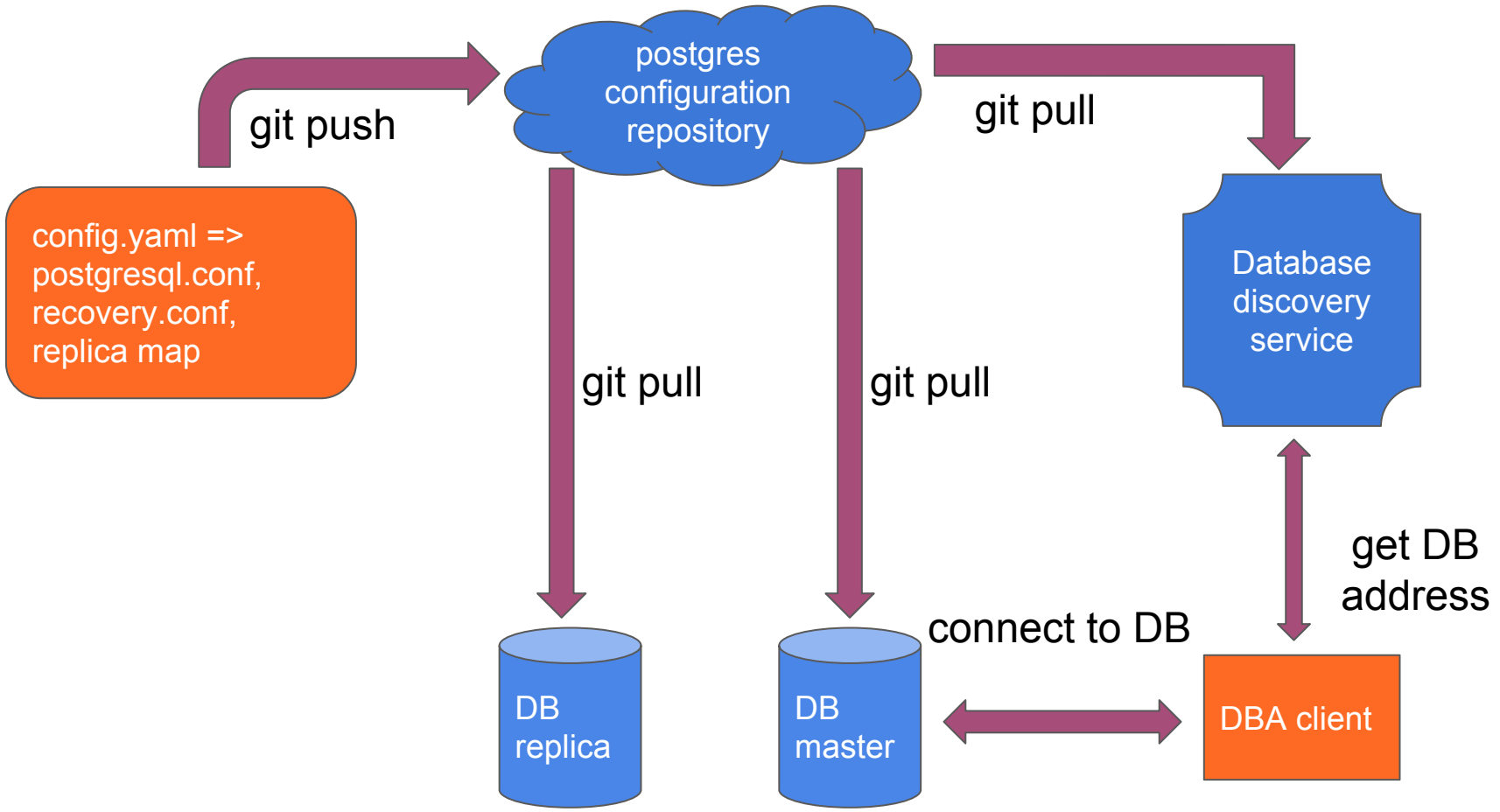
Script to initialize new nodes

Init from replicas to lower impact

Time delayed replicas in one data center

PostgreSQL versions: 9.3+

Git-driven workflow in data centers



zalando

PostgreSQL on Amazon AWS

Faster database provisioning

Flexible hardware configuration

CPU, Memory, Storage, Price

Docker is enforced at Zalando

Expected more node failures

Needs more automation

Patroni to the rescue

PostgreSQL management “daemon”

Adaptable to different platforms

Implemented in Python

Master election (using etcd, …)

Growing adoption and contributors

Zalando’s first open-source repo surpassing 1000 ☆

Why not AWS RDS or Aurora PostgreSQL

Not an easy answer :)

Full control

- Independent of cloud provider
- Real super user available
- Custom extensions, PAM
- Streaming/WAL replication in and out
- Local storage not supported on RDS (NVMe SSDs)

Costs? Cost of development? ...

PostgreSQL as a Service

Goals

Automation

- Self service for everyone
- Quick and easy way to get new cluster
- Enable users to modify cluster setup
- Restore and clone triggered by users

Integration

- Works with deployment pipeline
- Employee and application user provisioning
- Real time monitoring out of the box
- ZMON integration, entity discovery
- Zalando IAM integration

Create a new PostgreSQL cluster

Cluster YAML definition

```yaml
kind: "postgresql"
apiVersion: "acid.zalan.do/v1"

metadata:
  name: "acid-"
  namespace: "default"
  labels:
    team: acid

spec:
  teamId: "acid"
  postgresql:
    version: "10"
  numberOfInstances: 1
  volume:
    size: "10Gi"


  allowedSourceRanges:
  # IP ranges to access your cluster go here

  resources:
    requests:
      cpu: 100m
      memory: 1Gi
    limits:
      cpu: 1000m
      memory: 1Gi
```

Cluster configuration

Name	new-cluster		
Owning team	acid		
PostgerSQL version	10		
DNS name:	acid-.default		
Number of instances	1		
Replica load balancer	Enable replica ELB		
Volume size	10		Gi
Users			
Databases			
Resources	CPU	request	100 m
		limit	1000 m
	Memory	request	1 Gi
		limit	1 Gi

Copy definiton | Create cluster

Waiting for operator to create K8S objects

PostgreSQL cluster status **acid-fosdem-2018** edit

Cluster YAML definition

```yaml
apiVersion: acid.zalan.do/v1
kind: postgresql
metadata:
  clusterName: ''
  creationTimestamp: '2018-01-31T13:40:31Z'
  deletionGracePeriodSeconds: null
  deletionTimestamp: null
  generation: 0
  initializers: null
  labels:
    team: acid
  name: acid-fosdem-2018
  namespace: default
spec:
  allowedSourceRanges: null
  numberOfInstances: 2
  postgresql:
    version: '10'
  resources:
    limits:
      cpu: 1000m
      memory: 1Gi
    requests:
      cpu: 100m
      memory: 1Gi
  teamId: acid
  volume:
    size: 10Gi
status: Creating
```

Checking status of Cluster

StatefulSet pending

PostgreSQL 3rd party object created

Create request successful

zalando

Waiting for master to become available

PostgreSQL cluster status **acid-fosdem-2018** edit

Cluster YAML definition

```yaml
apiVersion: acid.zalan.do/v1
kind: postgresql
metadata:
  clusterName: ''
  creationTimestamp: '2018-01-31T13:40:31Z'
  deletionGracePeriodSeconds: null
  deletionTimestamp: null
  generation: 0
  initializers: null
  labels:
    team: acid
  name: acid-fosdem-2018
  namespace: default
spec:
  allowedSourceRanges: null
  numberOfInstances: 2
  postgresql:
    version: '10'
  resources:
    limits:
      cpu: 1000m
      memory: 1Gi
    requests:
      cpu: 100m
      memory: 1Gi
  teamId: acid
  volume:
    size: 10Gi
status: Creating
```

Checking status of Cluster

Waiting for master to become available

First PostgreSQL cluster container spawned

StatefulSet created

PostgreSQL 3rd party object created

Create request successful

Cluster create completed

PostgreSQL cluster status **acid-fosdem-2018** edit

Cluster YAML definition

```yaml
apiVersion: acid.zalan.do/v1
kind: postgresql
metadata:
  clusterName: ''
  creationTimestamp: '2018-01-31T13:40:31Z'
  deletionGracePeriodSeconds: null
  deletionTimestamp: null
  generation: 0
  initializers: null
  labels:
    team: acid
  name: acid-fosdem-2018
  namespace: default
spec:
  allowedSourceRanges: null
  numberOfInstances: 2
  postgresql:
    version: '10'
  resources:
    limits:
      cpu: 1000m
      memory: 1Gi
    requests:
      cpu: 100m
      memory: 1Gi
  teamId: acid
  volume:
    size: 10Gi
status: Running
```

Checking status of Cluster

PostgreSQL ready: **acid-fosdem-2018.default**

PostgreSQL master available, label is attached

First PostgreSQL cluster container spawned

StatefulSet created

PostgreSQL 3rd party object created

Create request successful

zalando

Automated role and database creation

```
users:
  application_owner:
  application_user:
databases:
  applicaiton_database: application_owner
```

```
# notice "_" -> "-" replacement

env:
  - name: FLYWAY_USER
    # OR just value: "app_owner"
    valueFrom:
      secretKeyRef:
        name: app-owner.acid-application1-db.credentials
        key: username

  - name: FLYWAY_PASSWORD
    valueFrom:
      secretKeyRef:
        name: app-owner.acid-application1-db.credentials
        key: password
```

“Hands free” deployment

K8S secrets for credentials

Encourage role split:

One for application deployment / DDL

One for application runtime / DML

No objects owned by employee roles

No “psql” required

Infrastructure roles

Employees and IAM integration

Use postgres with PAM authentication

Custom PAM authentication verifying our JWT token

Token valid for 60 minutes

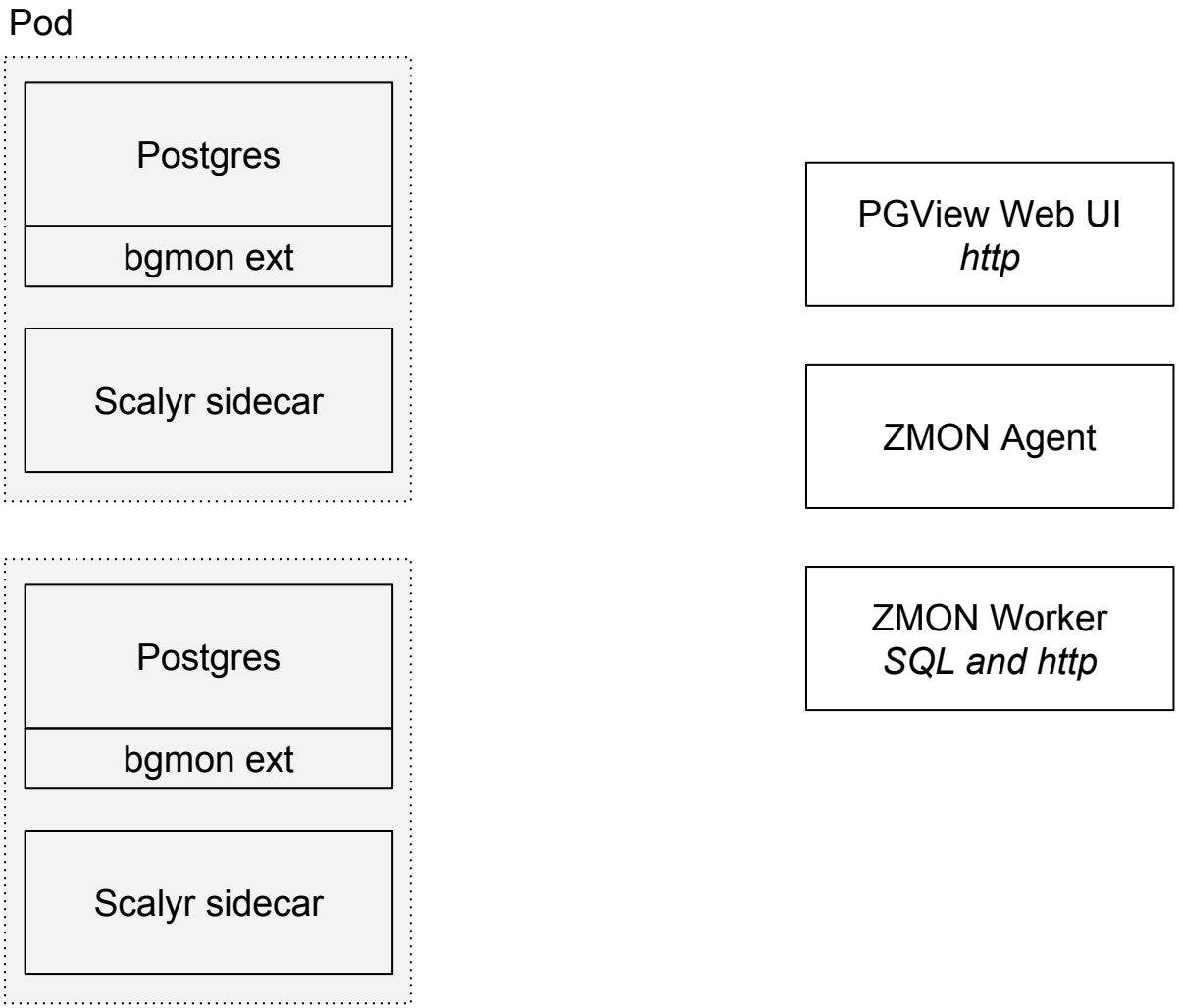
MFA for free via Google

No password sync, one less thing to remember for employees

```
export PGPASSWORD=$(ztoken)
export PGSSLMODE=require
psql -h cluster-name.team.domain -d postgres
```

Monitoring

Monitoring setup

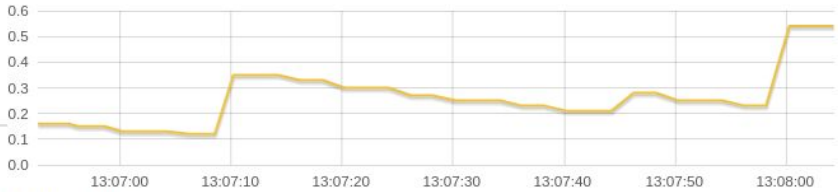


zalando

Monitoring with pgview.web

CPU (Cores: 4)

Idle %	System %	User %	IOWait %	Load 1	Load 5	Load 15
86.5	6.9	6.4	0	0.54	0.42	0.43



Memory

Total	Free	Buffers	Cached	Dirty	Commit Limit	Commited As
32.15GB	5.721GB	2.342GB	20.71GB	1.176MB	B	B

Partitions

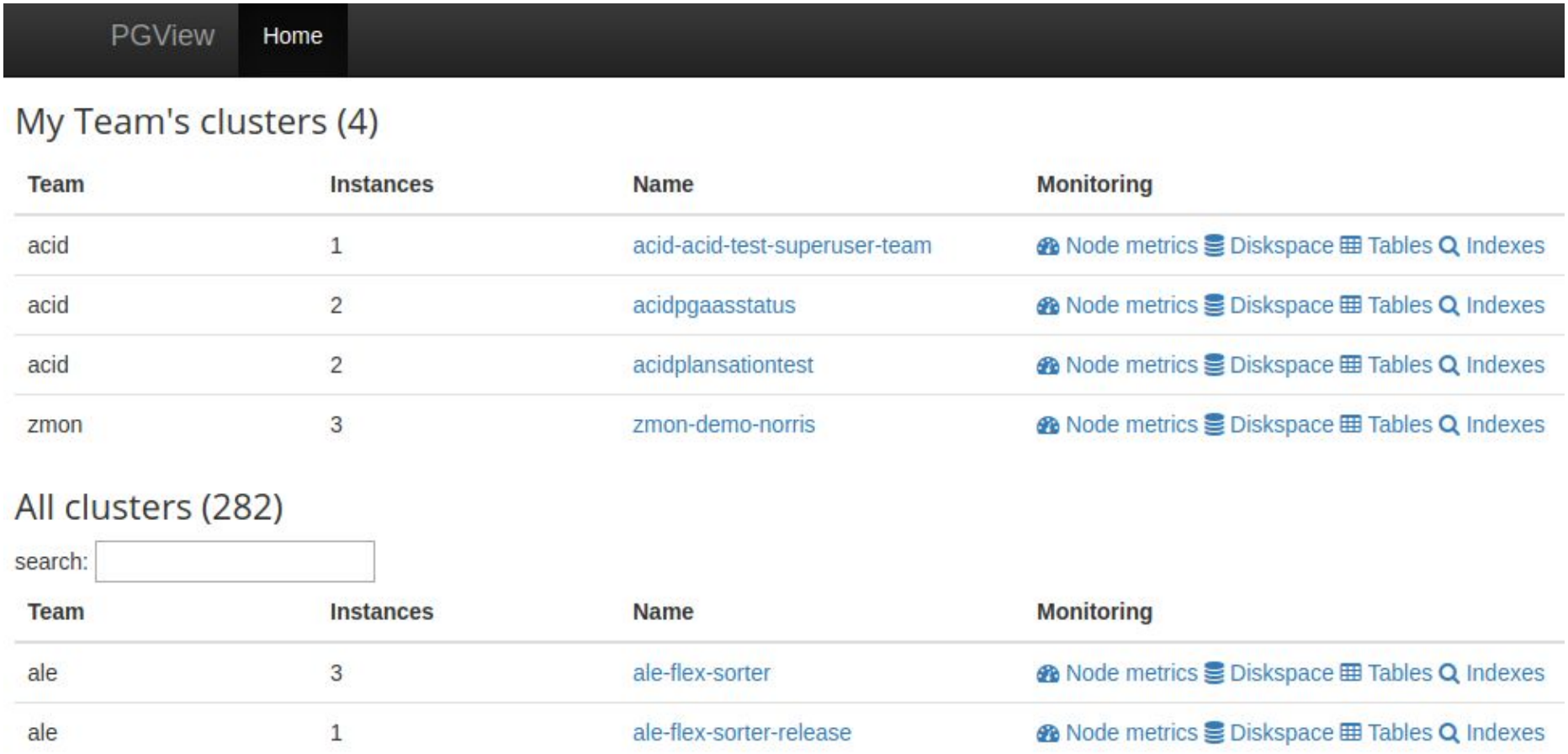
Device	await	read	write	Free	Total	Path	Size
data	0	0	0	49.67GB	52.71GB	/home/postgres/pgdata/pgroot/data	242.6MB
wal	0	0	0	49.67GB	52.71GB	/home/postgres/pgdata/pgroot/data/pg_wal	218.2MB

Processes 0 / 3 of maximum 100

Pause | Non Backends

PID	Lock	Type	utime	stime	read	write	age	DB	User	Query
48		logger	0	0	0	0				
243		checkpointer	0	0	0	0				
244		writer	0	0	0	0				
245		stats collector	0	0	0	0				
857		wal writer	0	0	0	0				
858		autovacuum launcher	0	0	0	0				
859		archiver	0	0	0	0				last was 000000080000000000000008E
861		bgworker: logical replication launcher	0	0	0	0				
893		wal sender	0	0	0	0				standby 10.2.1.111(46864) streaming 0/8F000000
1027		wal sender	0	0	0	0				standby 10.2.17.26(52268) streaming 0/8F000000

Integration with ZMON



zalando

EC2 Instance Metrics via ZMON

zalando

Cloud-native Postgres infrastructure

zalando

Kubernetes introduction

- Container management
- Cluster-wide application scheduling and autoscaling
- Application deployments automation
- Abstracts bare metal and most cloud providers (google, aws, azure, etc)
- Declarative description of resources and deployments
- Rich metadata (versions, labels, annotations)
- Supported by open-source community

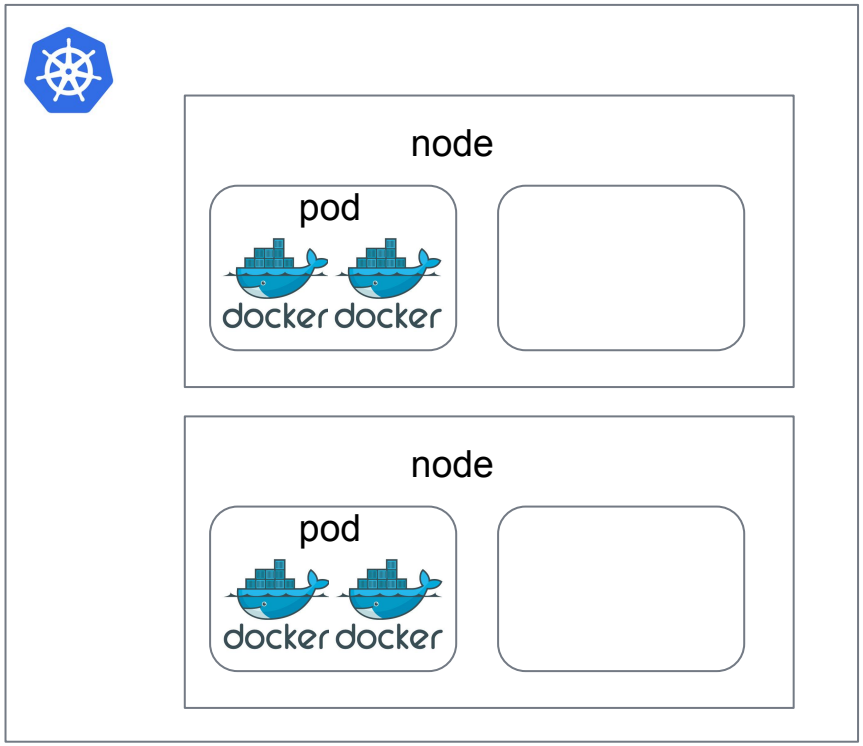
Labels

- Labels can be attached to almost any Kubernetes objects
- Each object can have multiple labels (name = value)
- Labels can be used to query groups of objects (all replicas belonging to a PostgreSQL cluster test):

```
$ kubectl get pods -l cluster-name=test
```

Nodes and pods

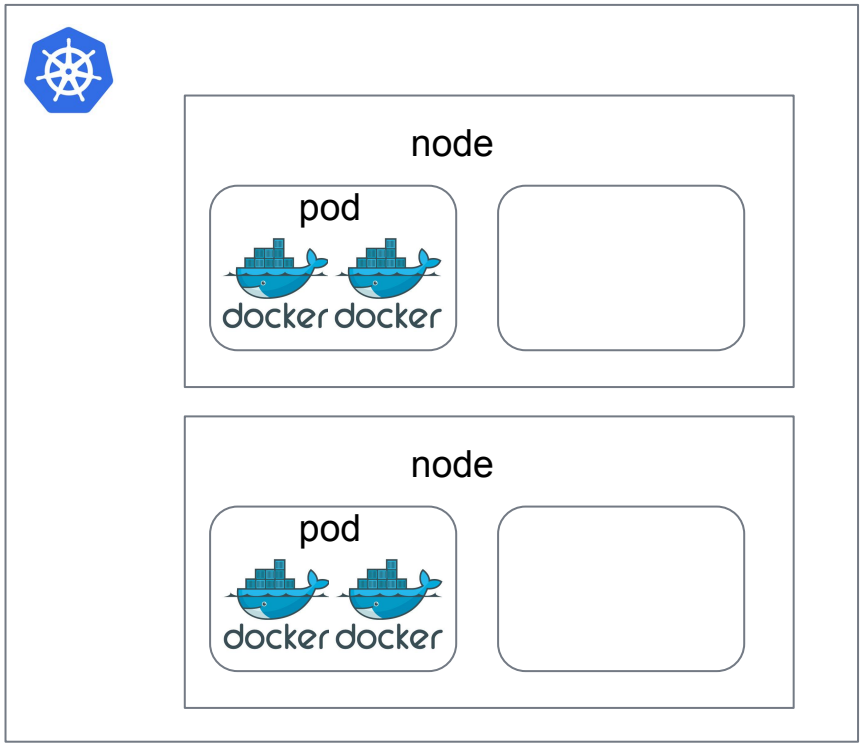
- Nodes are equivalents of physical servers
- Pods correspond to applications
- One pod may have many containers
- Pods are scheduled on nodes
- Scheduling is controlled by resource requests and limits.



Nodes and pods

Example:

- Amazon EC2 instance is a node
- On a node Postgres pod is running
- Postgres pod consists of 2 containers: Postgres container and a database log shipping container



Nodes and pods

- Pods are scheduled on nodes
- Scheduling is automatic and is controlled by resource requests and limits on pods

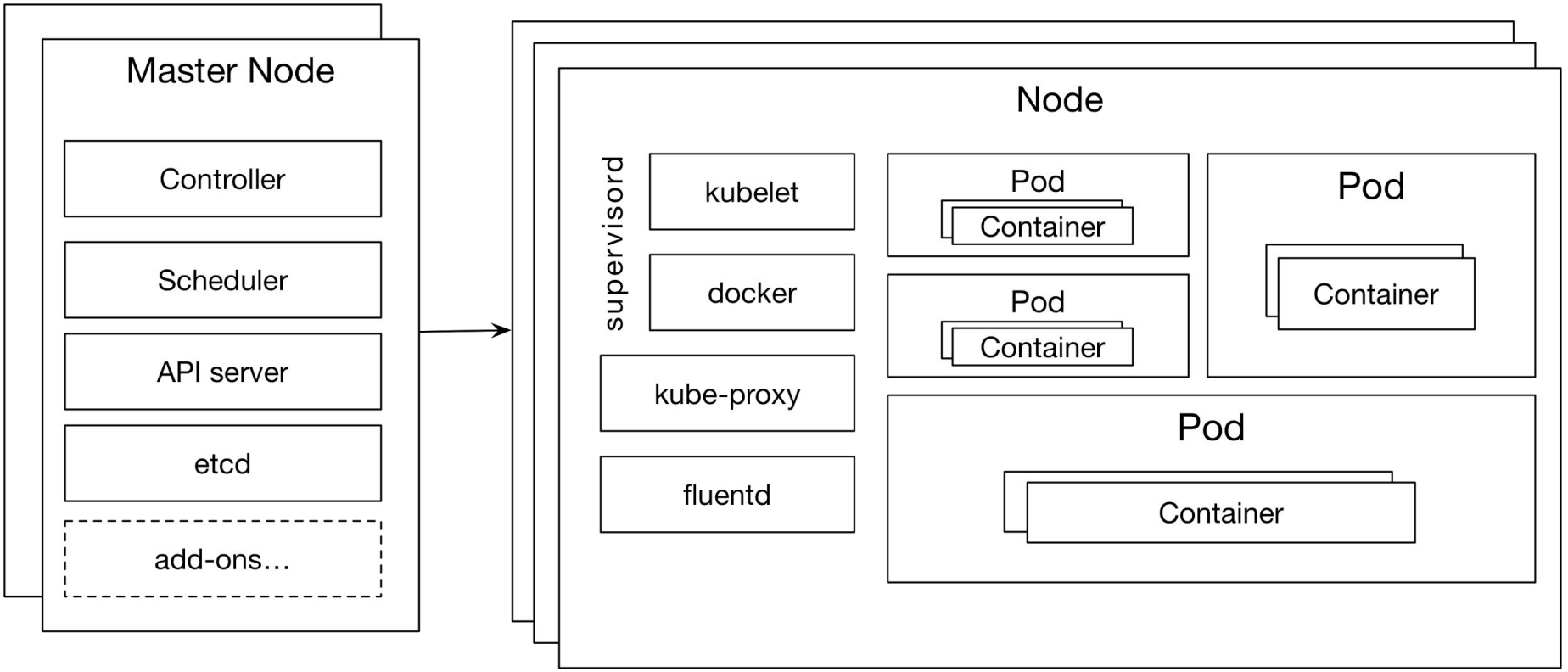
```
resources:
    limits:
      cpu: "3"
      memory: 1Gi
    requests:
      cpu: 100m
      memory: 100Mi
```

Nodes and pods

Example:

Multiple staging PostgreSQL pods can be scheduled on one node, saving resources and keeping database users isolated at the same time.

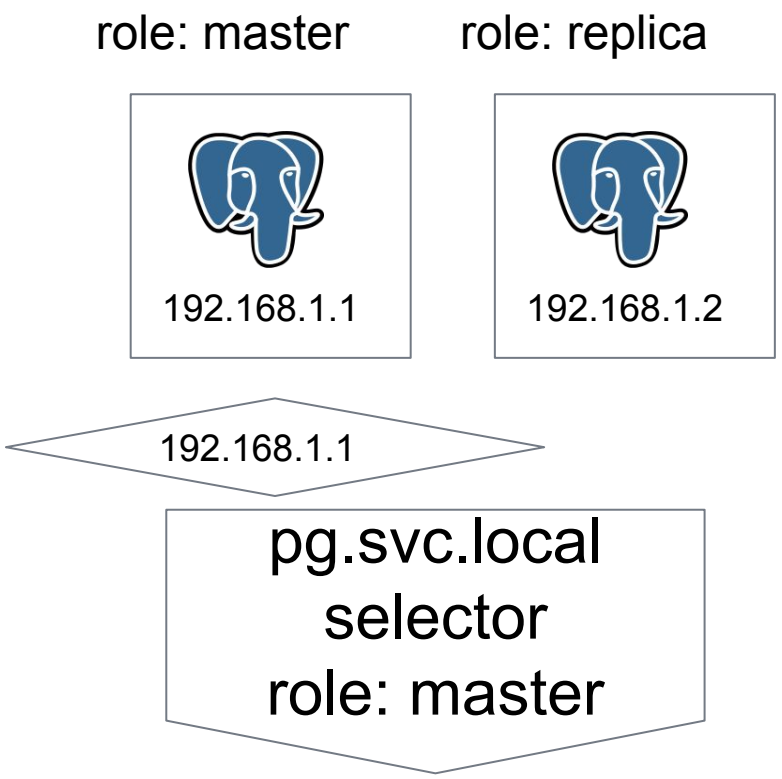
System and worker nodes



zalando

Services and endpoints

- Services connect clients to pods
- Endpoints contain actual pod addresses
- Endpoints can be managed by services or independently
- Services may define selectors to point to pod labels



StatefulSets

- Persistent Volume (PV):
  i.e. NAS, EBS

- Persistent Volume Claim (PVC):
  request to find a persistent volume with at last as much disk space as the claim

- StatefulSet
  - joins pods and persistent volume claims
  - when a pod terminates - it gets respawned and the same volume is reattached
  - ip address of the node is preserved between terminations

StatefulSets

Example:

- A StatefulSet defines 3 pods
- Each pod claims 100GB Persistent Volume.
- Each pod mounts the volume as /home/postgres/pgroot.
- When postgres container starts - it does initdb at /home/postgres/pgroot/pgdata when it is container 0, otherwise, tries to pg_basebackup from container 0.
- When postgres container dies it gets restarted and /home/postgres/pgroot/pgdata is not empty. In that case, it just tries to start PostgreSQL.

Running a PostgreSQL cluster on Kubernetes

- Bare StatefulSet with PVC-backed instances
- Helm
- Bot-pattern based solution with automatic failover
- Use a managed (by someone else) service

Advantages of StatefulSets with PVC

- Statefulset re-creates pods and re-attaches persistent volumes.
- Compatible with stock PostgreSQL image.
- Easy to implement (write a single manifest, change number of replicas and a cluster name).
- Easy to configure services (master is always pod 0).

Disadvantages of StatefulSets with PVC

- Downtime in minutes when master pod goes down
- Bugs in StatefulSets, attachments of PVCs, etc.
- When the volume is corrupted - you need to manually rebuild pod 0 and do at least 2 failovers.
- Manual cluster-wide changes to PostgreSQL configuration with a downtime
- Hard to monitor (what if the pod is up but PostgreSQL is not running)?

Helm

- Abstracts actual manifests from the deployer.
- Use solutions well-tested by others.
- Someone needs to write a Helm chart.

Managed database services

zalando

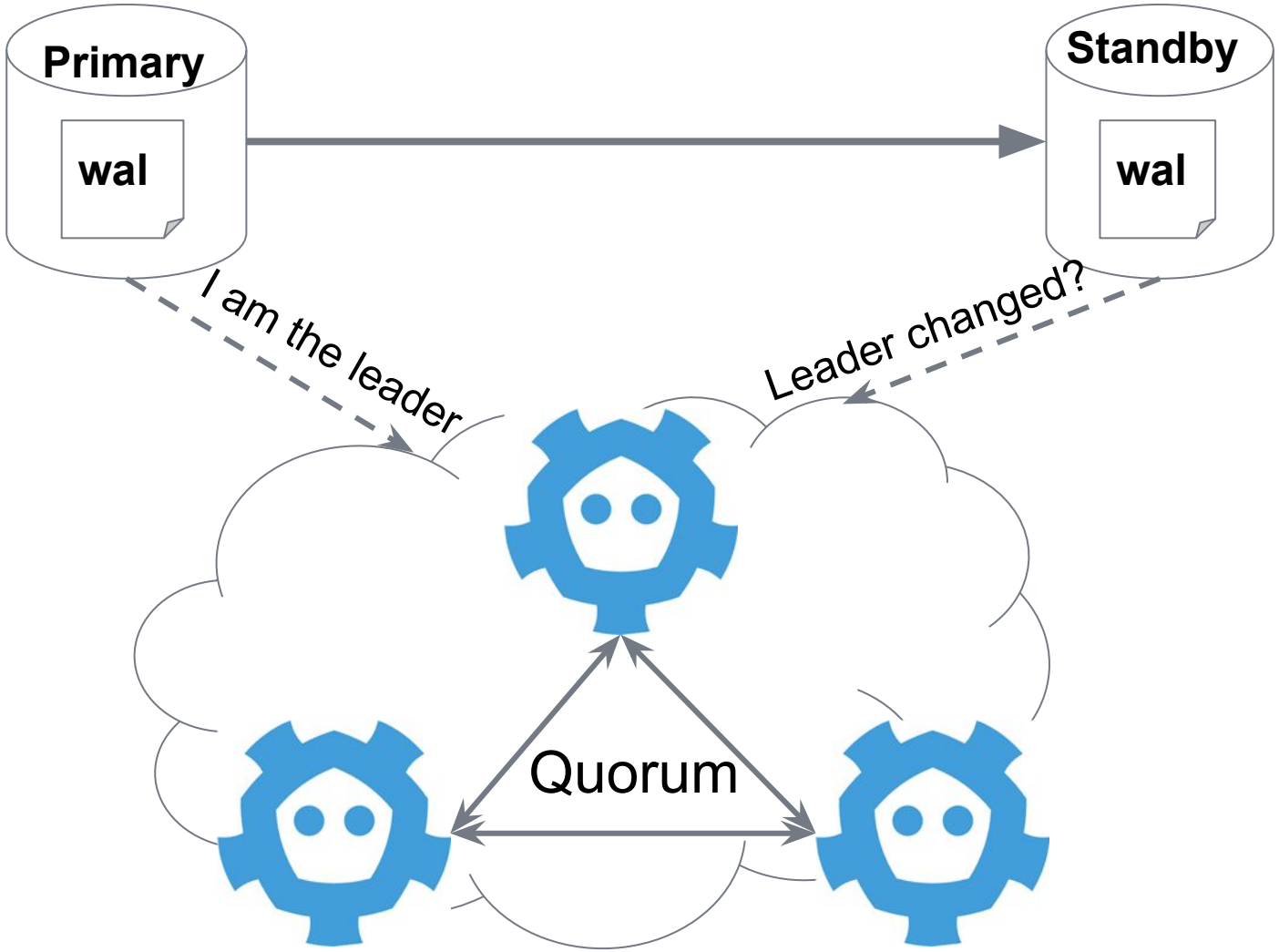
Managed services

- Vendor-lock
- Limited set of extensions
- Usually no superuser access
- Cannot use alphas/betas/compile your own
- Need to wait for vendor to apply patches
- Expensive

Automatic failover: the right way

- Quorum to decide who is the leader
- Think what happens if the network is partitioned
- Kill old client connections
- STONITH (shoot the other node in the head)
- Configure the Watchdog

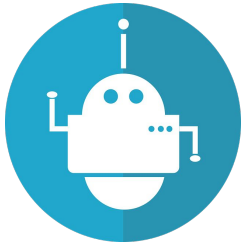
Automatic failover done right



zalando

Bot pattern

- PostgreSQL cannot talk to Kubernetes directly
- Let’s employ a bot to manage PostgreSQL
- A bot should run alongside PostgreSQL
- A bot will talk to Etcd (or other DCS)
- A bot decides on promotion/demotion



Patroni

- Patroni implements bot pattern in Python.
- Official successor of Compose Governor.
- Developed in the open by Zalando and volunteers all over the world.
- Can be configured with environment variables.
- Supports Etcd, Consul, Zookeeper, Exhibitor and Kubernetes.

https://github.com/zalando/patroni

Using Kubernetes as a consistency store

- Developed by Alex Kukushkin
- Use annotations on:
  - Pods for cluster members
  - Dedicated Endpoint for cluster configuration.
  - Service-related Endpoint for leader information.
- Reliability: always use EndPoints.
- Compatibility mode: use ConfigMaps, not Endpoints.

http://patroni.readthedocs.io/en/latest/kubernetes.html

Spilo

- Packages Patroni and Postgres in a Docker image
- Kubernetes-native configuration with Environment variables
- Multiple PostgreSQL versions (9.3, 9.4, 9.5, 9.6, 10)
- Small size
- Installs a number of useful extensions
- PAM authentication for Postgres

https://github.com/zalando/spilo

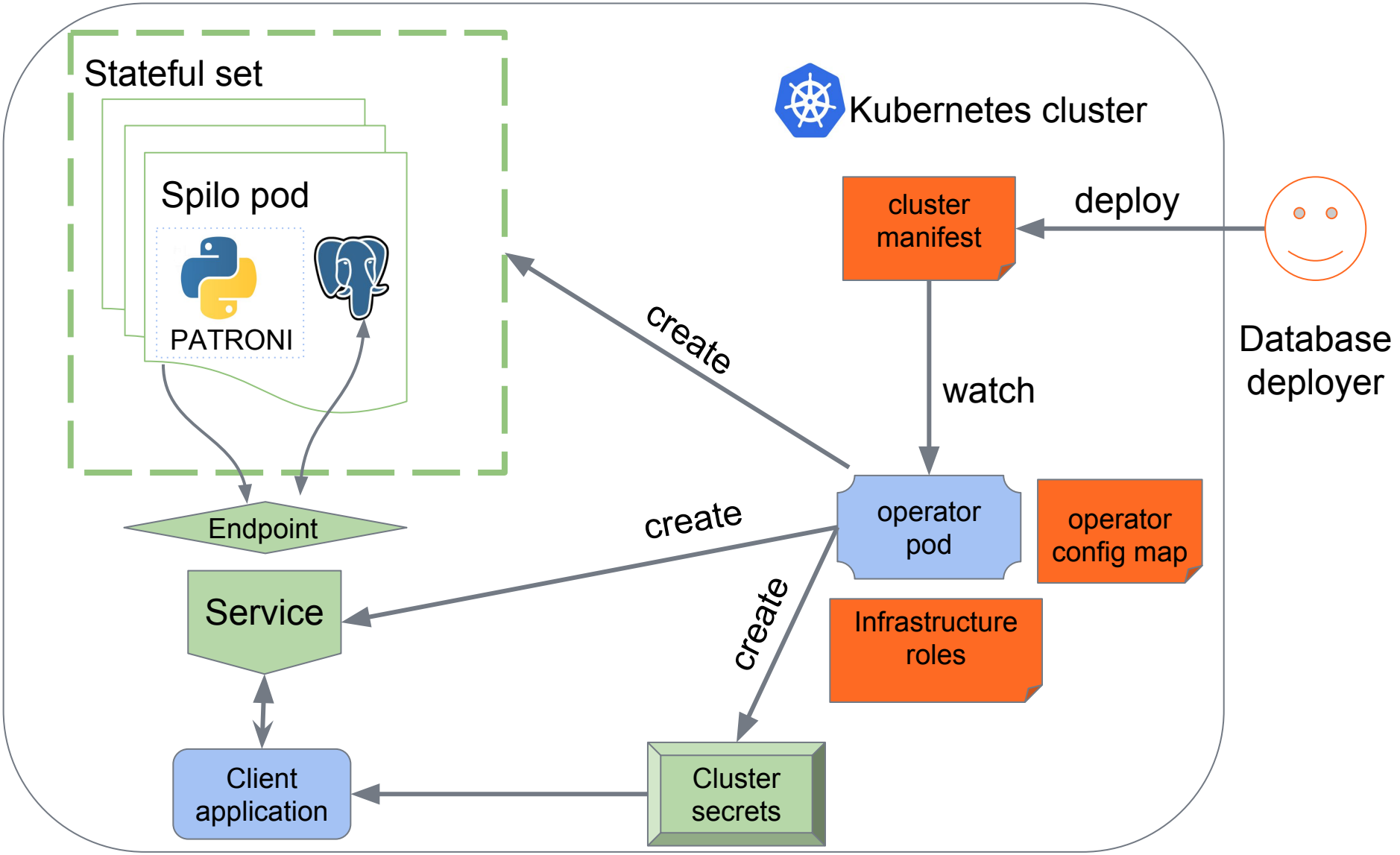
Running PostgreSQL on Kubernetes at Scale

- Implement the operator pattern on top of Spilo
  - Create, update, sync, delete clusters
  - Simple YAML manifests to create clusters
  - UI tools to generate YAMLs for you
- Postgres Operator from Zalando
- Evolved from a hackweek project of Murat Kabilov
- Used for staging and production clusters by Zalando

https://github.com/zalando-incubator/postgres-operator

Layer by layer

- Operator starts pods with Spilo docker image
- Operator provides environment variables to Spilo
- Operator makes sure all Kubernetes objects are in sync
- Spilo generates Patroni configuration
- Patroni creates roles and configures PostgreSQL
- Patroni makes sure there is only one master
- Patroni uses Kubernetes for cluster state and leader lock
- Patroni creates roles and applies configuration
- Patroni changes service endpoints on failover


Stateful set
Spilo pod
PATRONI
Kubernetes cluster
cluster manifest
deploy
Database deployer
create
watch
Endpoint
create
operator pod
operator config map
Service
create
Infrastructure roles
Client application
Cluster secrets

Minimal cluster manifest

```yaml
apiVersion: "acid.zalan.do/v1"
kind: postgresql
metadata:
 name: acid-minimal-cluster
spec:
 teamId: "ACID"
 volume:
   size: 1Gi
 numberOfInstances: 2
 users:
   # database owner
   zalando:
   - superuser
   - createdb

   # role for application foo
   foo_user:

 #databases: name->owner
 databases:
   foo: zalando
 postgresql:
   version: "10"
```

Operator configuration ConfigMap

```
data:
 service_account_name: operator
 docker_image: registry.opensource.zalan.do/acid/demospilo-10:1.3-p3
 etcd_host: ""
 enable_teams_api: "false"
 infrastructure_roles_secret_name: postgresql-infrastructure-roles
 super_username: postgres
 replication_username: standby
 resync_period: 5m   # how often the clusters are synced
 workers: "4"
 api_port: "8080"
 pod_terminate_grace_period: 5m
 pdb_name_format: "postgres-{cluster}-pdb"
 node_readiness_label: "ready:true"
```

Initial cluster roles

- **Cluster manifest**
  Roles specific to the Postgres cluster

- **Infrastructure roles**
  Defined in the infrastructure roles secret, same for all Postgres clusters managed by the operator (i.e. monitoring roles)

- **Teams API**
  Human users, names automatically fetched from an external API based on the team defined in the cluster configuration.

Teams API

Operator configuration:

```
enable_teams_api: "true"
teams_api_url: http://fake-teams-api.default.svc.cluster.local
enable_team_superuser: "false" # grant superuser
team_admin_role: "admin" # team roles are members of this role
oauth_token_secret_name: "postgres-operator"
```

Cluster configuration:

```
spec:
 teamId: "ACID"
```

Test implementation: https://github.com/ikitiki/fake-teams-api

OAUTH2 PAM authentication

- PAM module written in C by Alex Kukushkin
- Open-source: https://github.com/CyberDem0n/pam-oauth2
- Equivalent of arbitrary-long automatically generated, auto-expiring passwords.
- Can supply arbitrary key=value pairs to check in the OAuth response (i.e. `realm=/employees`)

OAUTH2 PAM authentication

Operator configuration:

```
    pam_configuration:
https://info.example.com/oauth2/tokeninfo?access_token= uid
realm=/employees
    pam_role_name: users
```

Operator sets PAM_OAUTH2 Spilo environment variable

ands a line to pg_hba.conf

```
hostssl all +users all pam
```

Spilo writes /etc/pam.d/postgresql using PAM_OAUTH2 value.

Resizing volumes (EBS)

- Dynamically enlarge disk space (when running on AWS)
- Call EBS resize API
- Resize the filesystem by callng resize2fs inside the pod

Kubernetes cluster upgrades

- Kubernetes evolves very rapidly.
- Cluster upgrades require rotating nodes.
- A postgres master pod previously running on an old node can potentially land on another old node after the first node is terminated, **resulting in multiple failovers**.

Avoiding multiple failovers

- Operator defines a node label to consider a node as not ready

```
node_readiness_label: "lifecycle-status:ready
```

- A node that is not ready and drained triggers the operator to migrate master pods for all clusters running there

- Postgres pod can only be scheduled on a ready node using **NodeAffinity** feature in the StatefulSet definition

- The operator defines a **pod disruption budget** to avoid any master running on the nodes to be killed prematurely when the node is drained.

That’s it, thank you! Questions?

- Patroni: https://github.com/zalando/patroni
- Patroni Documentation: https://patroni.readthedocs.io
- Spilo: https://github.com/zalando/spilo
- Postgres operator: https://github.com/zalando-incubator/postgres-operator
- Postgres top as a background worker: https://github.com/CyberDem0n/bg_mon
- PAM Oauth2 module: https://github.com/CyberDem0n/pam-oauth2