OVHcloud intel

Incontro DevOps Italia

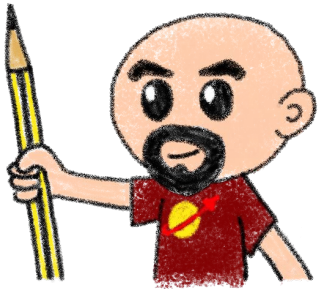
Kubernetes Operators:

Operating Cloud Native services at scale

Horacio Gonzalez 2021-02-05

Who are we?

Introducing myself and introducing ~~OVH~~ OVHcloud

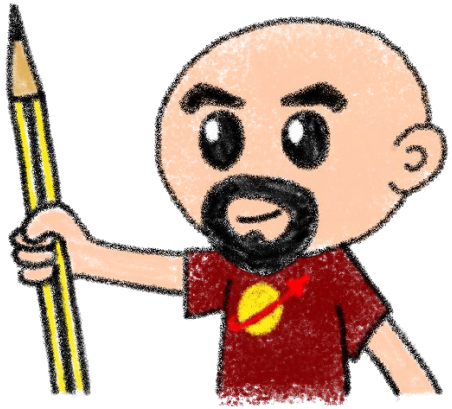


Horacio Gonzalez

Incontro DevOps Italia

@LostInBrittany

Spaniard lost in Brittany, developer, dreamer and all-around geek



DevFest du Bout du Monde

Finist Devs

OVHcloud

DevRel Leader

Google Developers Experts 2019

Web Technologies

GDE

Flutter

OVHcloud: A global leader

- **Web Cloud & Telcom**
- **Private Cloud**
- **Public Cloud**
- **Storage**
- **Network & Security**

- **30 Data Centers** in 12 locations
- **34 Points of Presence** on a 20 TBPS Bandwidth Network
- **2200 Employees** worldwide
- **115K Private Cloud** VMS running
- **300K Public Cloud** instances running
- **380K Physical Servers** running in our data centers

- **1 Million+ Servers** produced since 1999
- **1.5 Million Customers** across 132 countries
- **3.8 Million Websites** hosting
- **1.5 Billion Euros Invested** since 2016
- **P.U.E. 1.09** Energy efficiency indicator
- **20+ Years in Business** Disrupting since 1999

High performance at affordable prices

Infra-4

Processore: 2x Intel Xeon Silver 4214 - 12 c / 24 t - 2.2 GHz / 3.2 GHz
Memoria: A partire da 96GB
Storage: NVMe, SATA disponibile

Banda passante pubblica: A partire da 1 Gbps
Banda passante privata: A partire da 2 Gbps

Disponibile in 7 datacenter	Consegna a partire da 120 s

OVHcloud

HGR-SDS-1

Processore: Intel Xeon Gold 6242R - 20 c / 40 t - 3.1 GHz / 4.1 GHz
Memoria: A partire da 96GB
Storage: NVMe, SAS disponibile

Banda passante pubblica: A partire da 1 Gbps
Banda passante privata: A partire da 10 Gbps

Disponibile in 5 datacenter	Consegna a partire da 120 s

HGR-HCI-2

Processore: 2x Intel Xeon Gold 6242R - 20 c / 40 t - 3.1 GHz / 4.1 GHz
Memoria: A partire da 384GB
Storage: NVMe, SAS disponibile

Banda passante pubblica: A partire da 1 Gbps
Banda passante privata: A partire da 10 Gbps

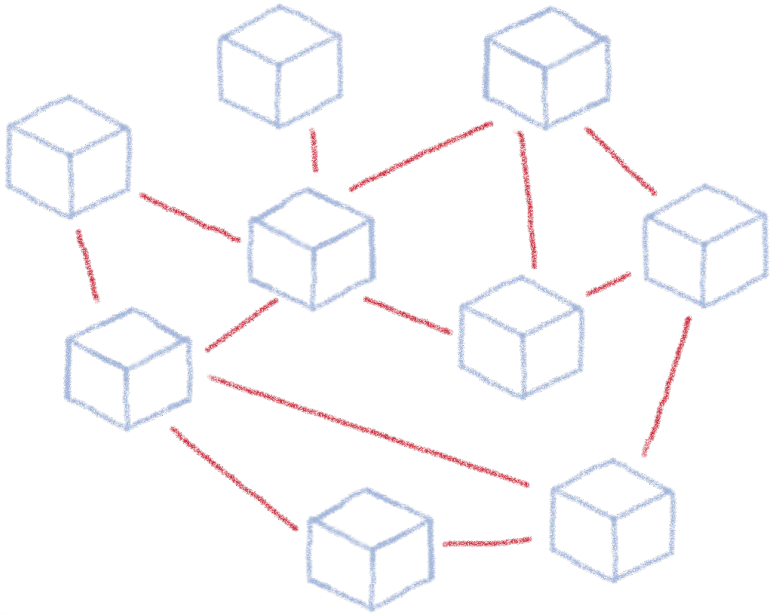
Disponibile in 5 datacenter	Consegna a partire da 10 g

From bare-metal servers to public or private cloud

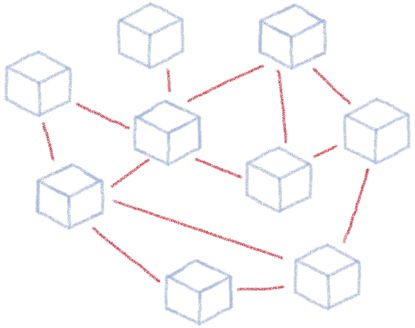
Kubernetes Operators

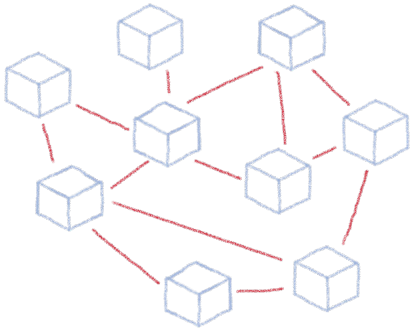
Helping to tame the complexity of K8s Ops

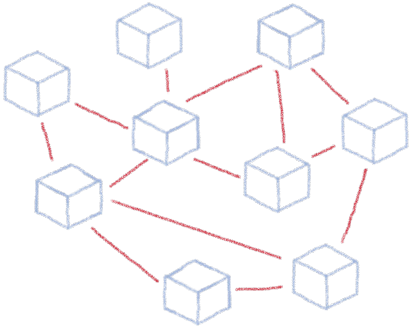
Taming microservices with Kubernetes



What about complex deployments







Ingress

Services

Deployments

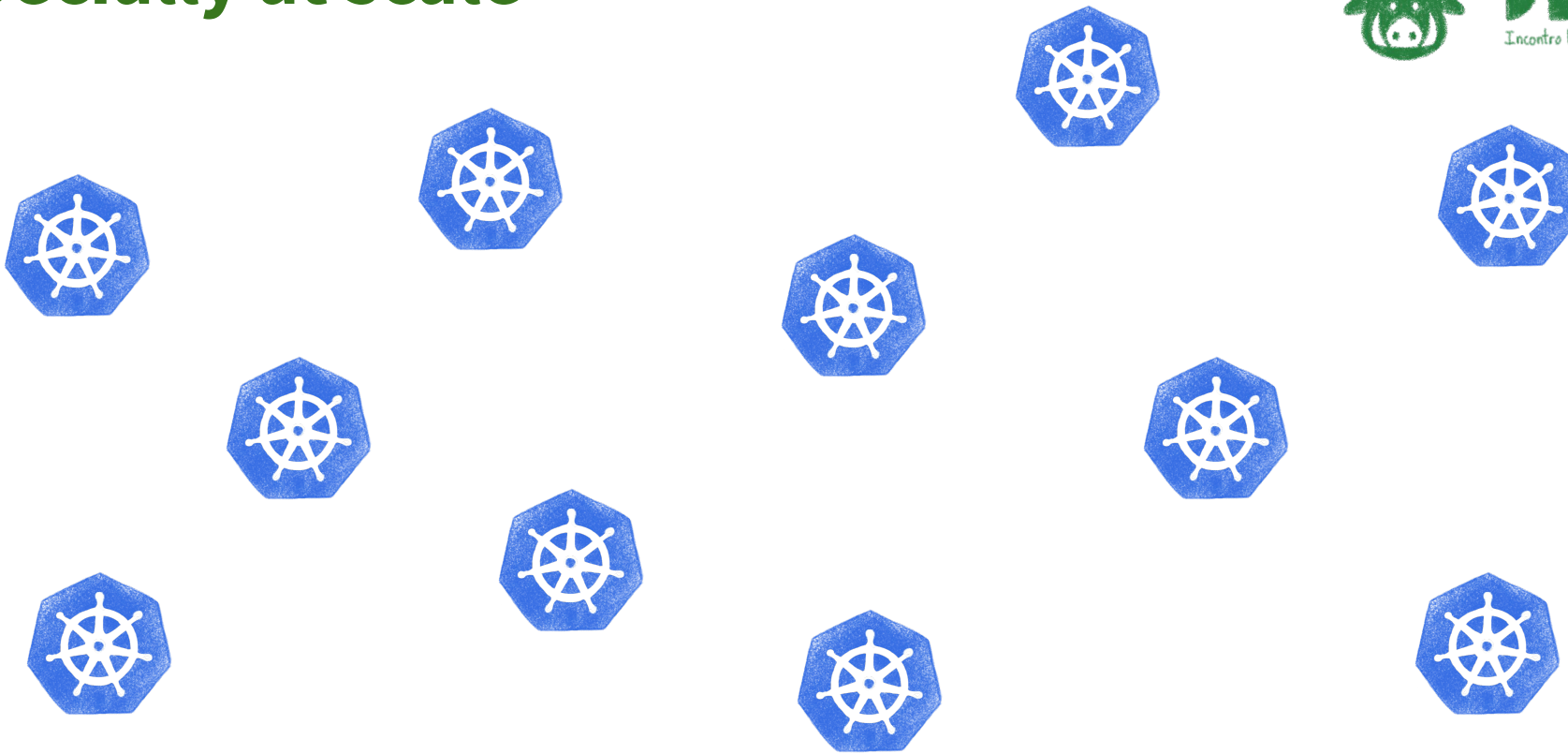
Pods

Sidecars

Replica Sets

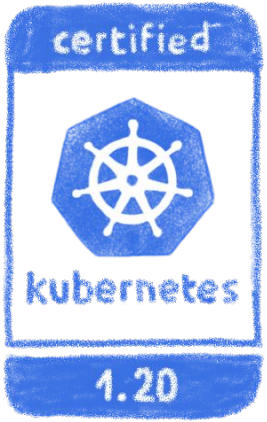
Statepul Sets

Specially at scale



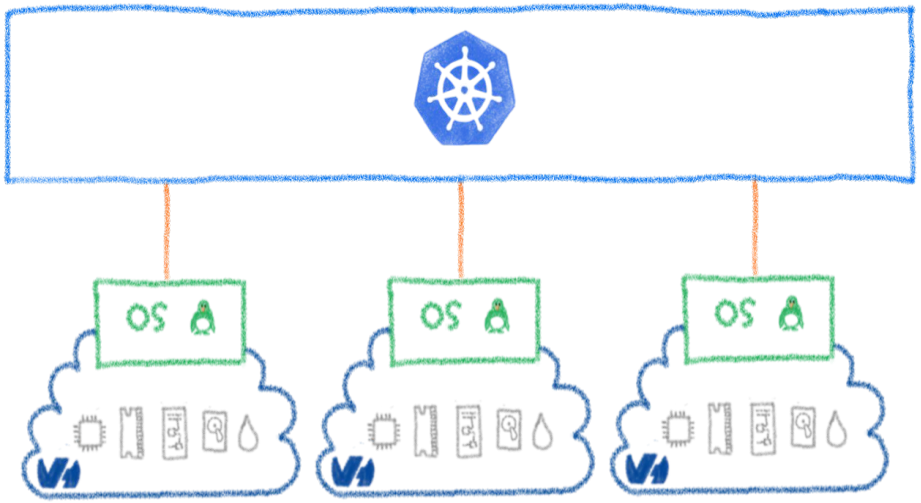
Lots of clusters with lots and lots of deployments

That's just our case



We both use Kubernetes and
operate a Managed Kubernetes platform

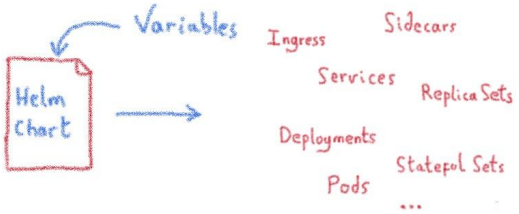
Built over our Openstack based Public Cloud



We need to tame the complexity

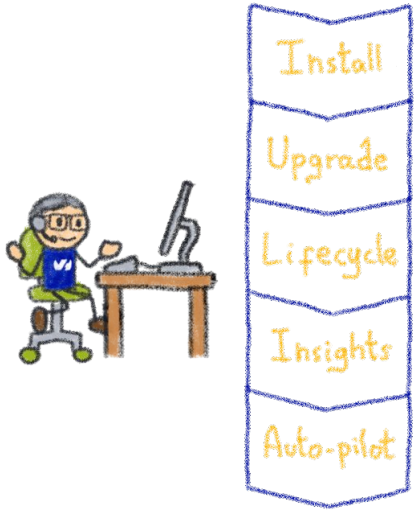
Taming the complexity

A package manager for Kubernetes



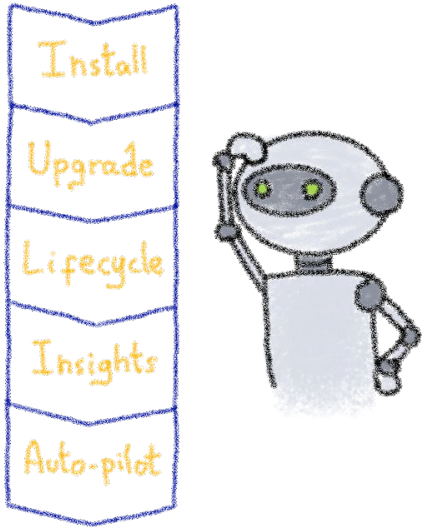
- Manage complexity
- Simple sharing
- Easy upgrades
- Easy rollbacks

Helm Charts are configuration



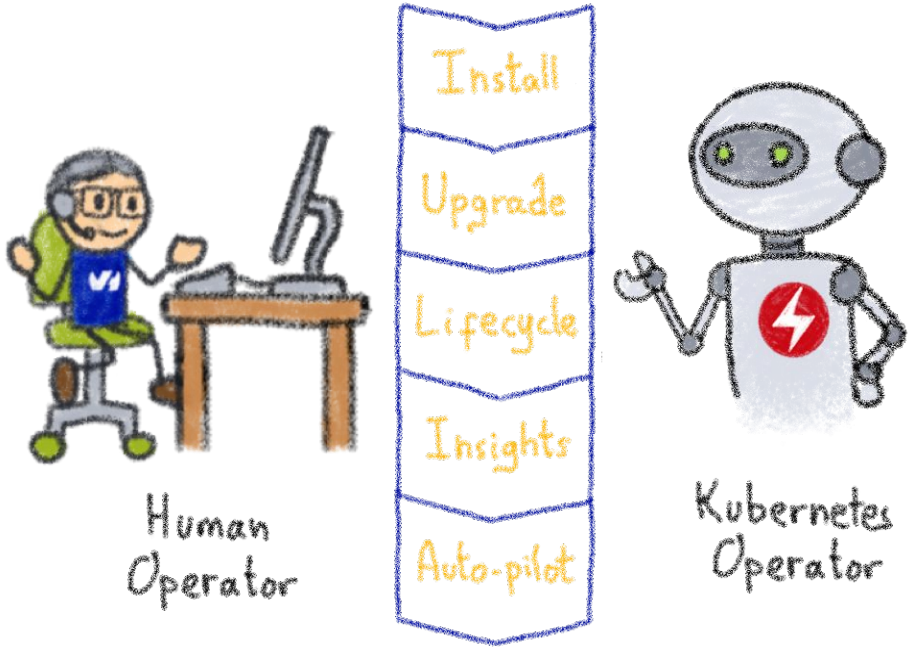
Operating is more than installs & upgrades

Kubernetes is about automation



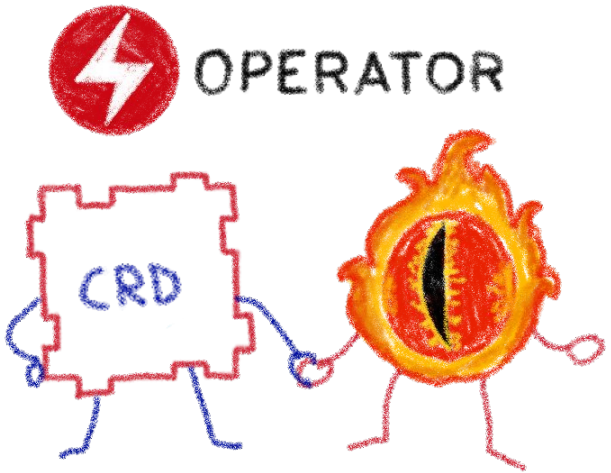
How about automating human operators?

Kubernetes Operators



A Kubernetes version of the human operator

Building operators

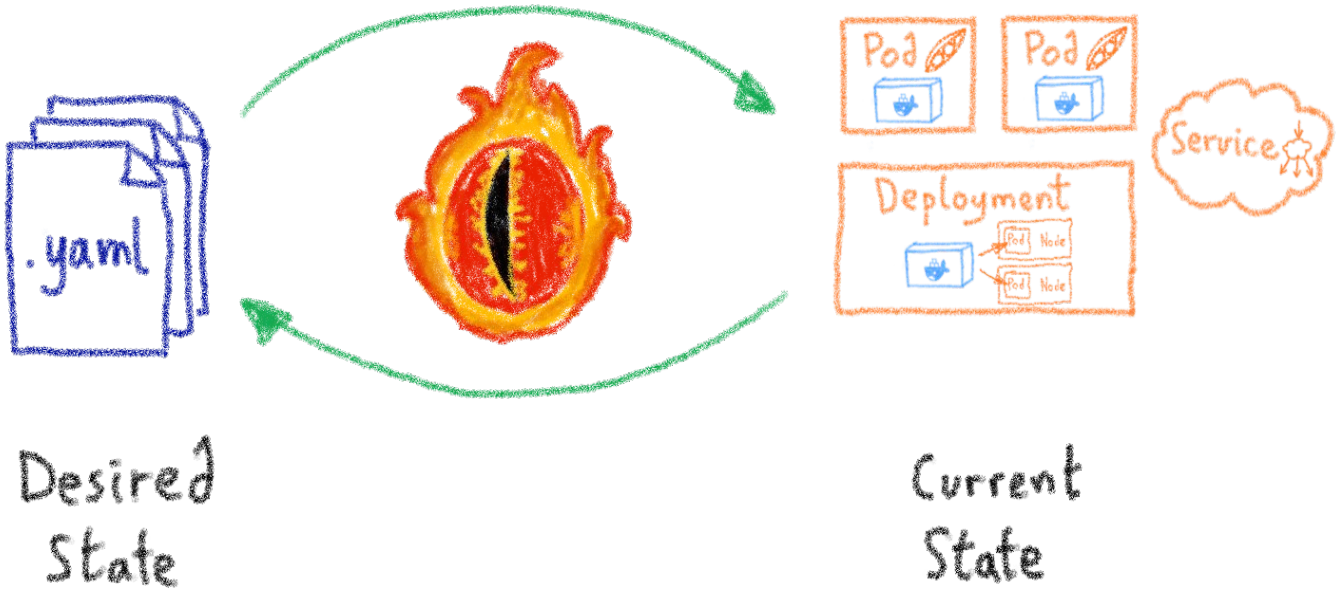


Basic K8s elements: Controllers and Custom Resources

Kubernetes Controllers

Keeping an eye on the resources

A control loop



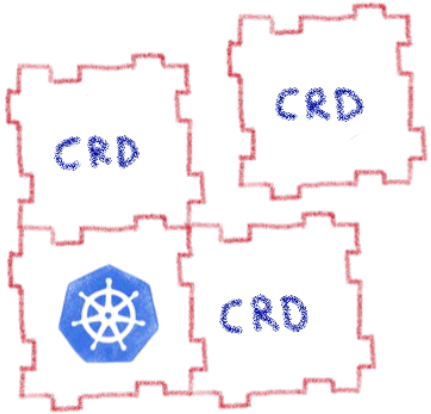
They watch the state of the cluster,
and make or request changes where needed

A reconcile loop

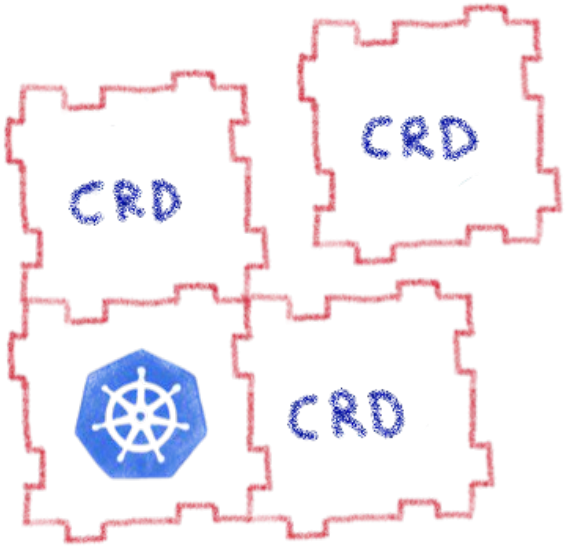
Strives to reconcile current state and desired state

Custom Resource Definitions

Extending Kubernetes API



Extending Kubernetes API

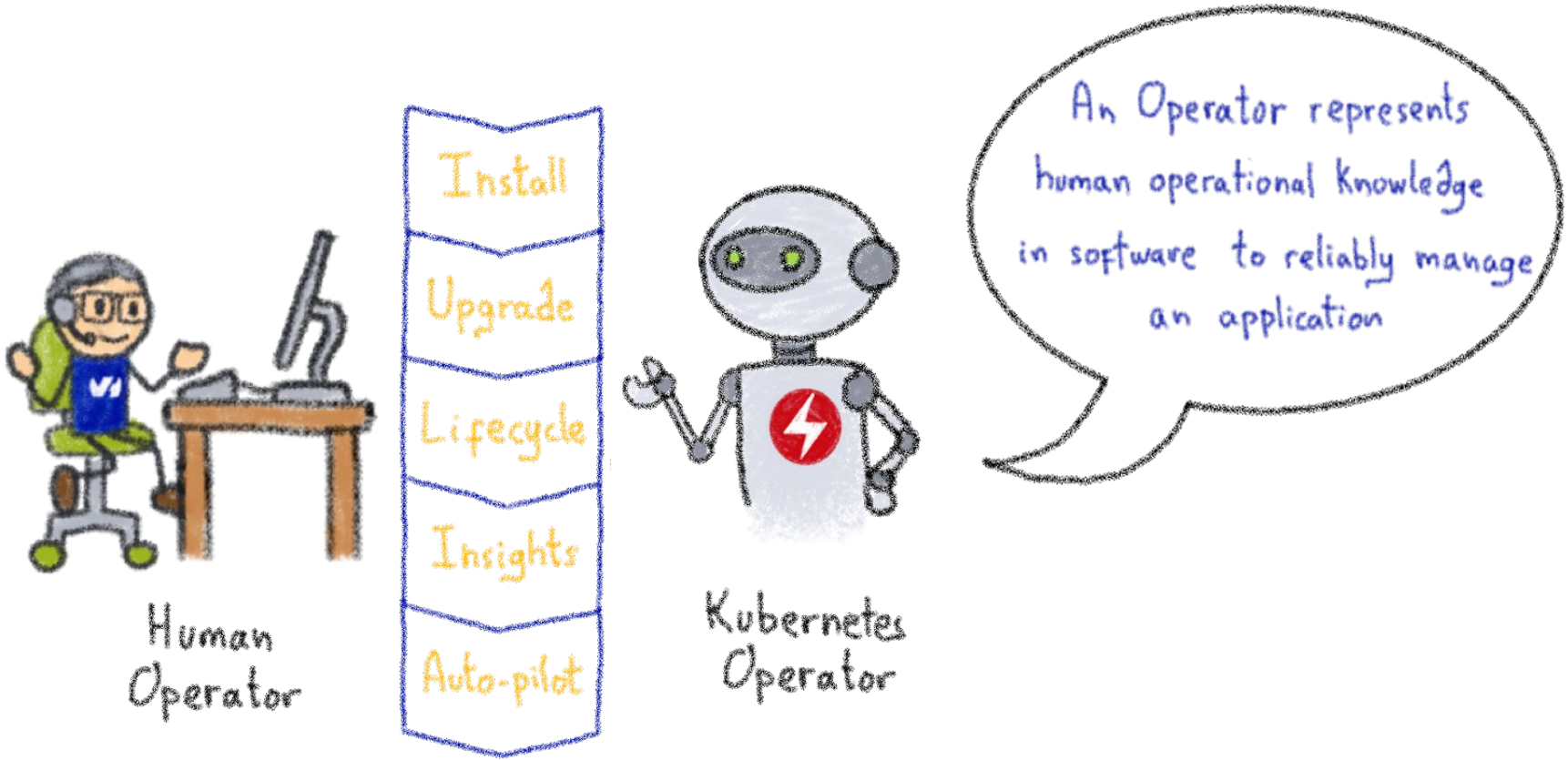


By defining new types of resources

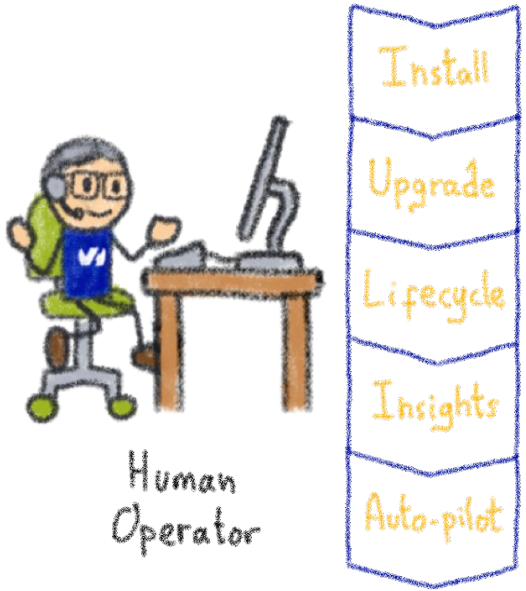
Kubernetes Operator

Automating operations

What's a Kubernetes Operator?



Example: databases

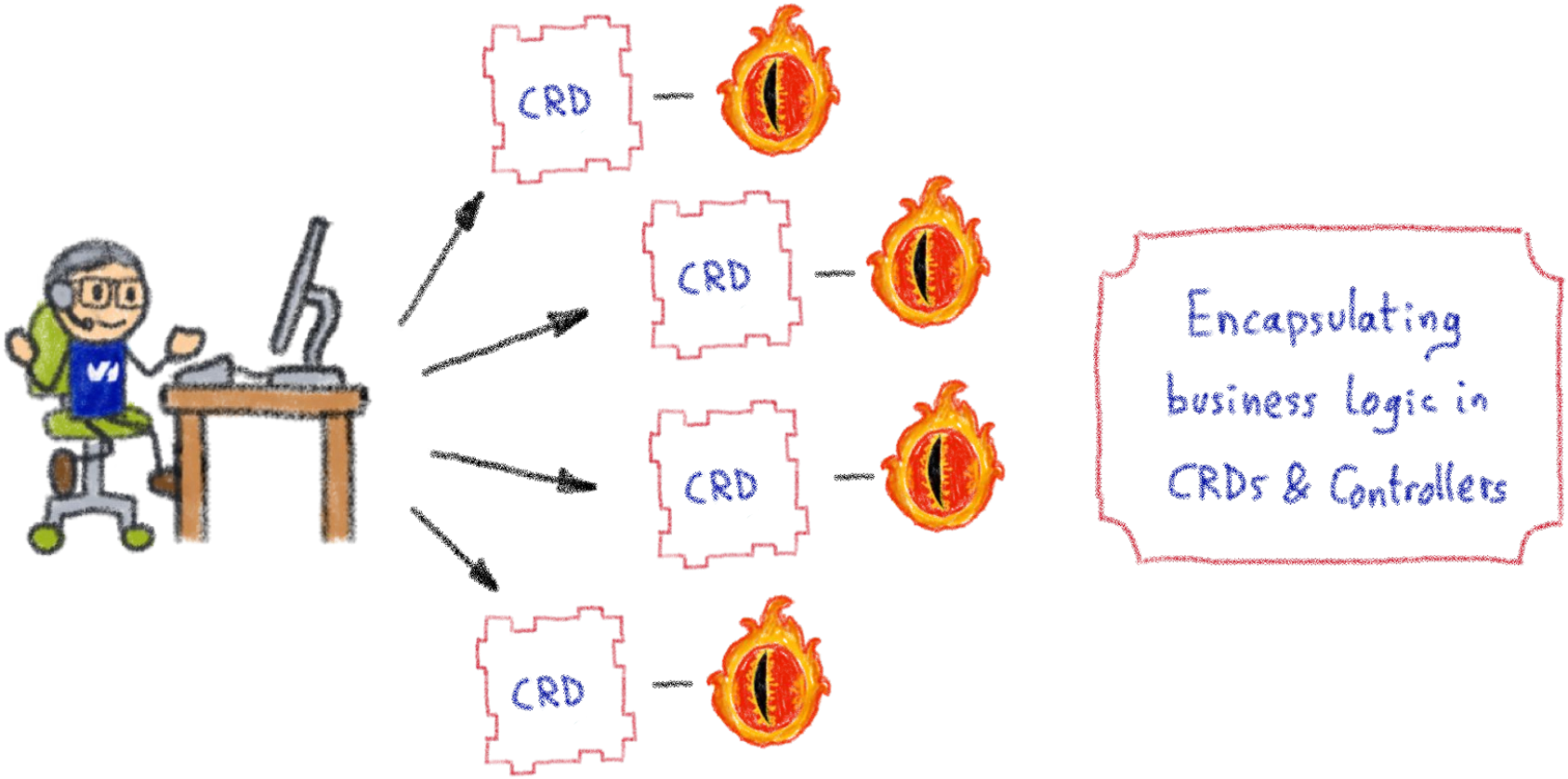


PostgreSQL

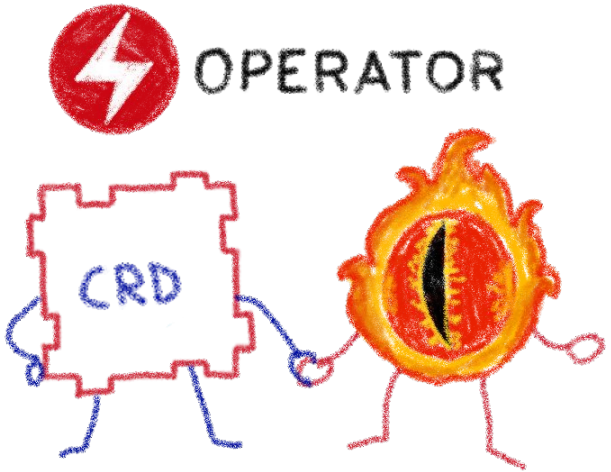
redis

Things like adding an instance to a pool,
doing a backup, sharding...

Knowledge encoded in CRDs and Controllers

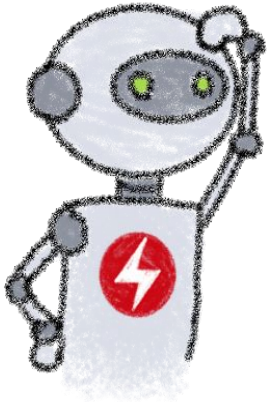


Custom Controllers for Custom Resources



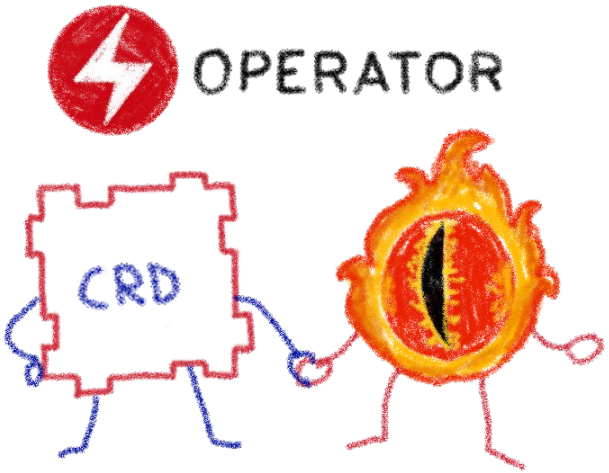
Operators implement and manage Custom Resources using custom reconciliation logic

Operator Capability Model



Gauging the operator maturity

How to write an Operator

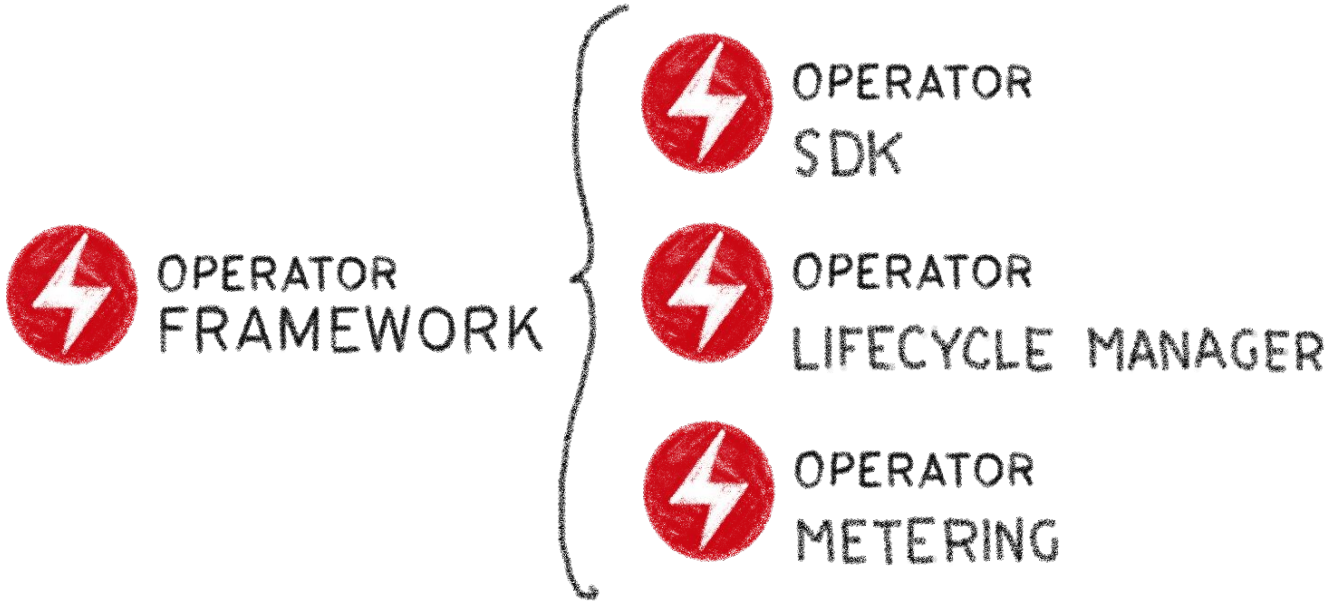


1- Create a new project
2- Write the CRDs to define new resource APIs
3- Specify resources to watch
4- Define the reconciliation logic in the Controllers
5- Build the Operator

Kubebuilder

SDK for building Kubernetes APIs using CRDs

The Operator Framework

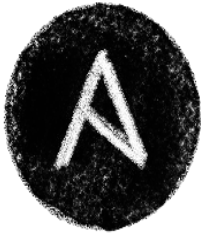


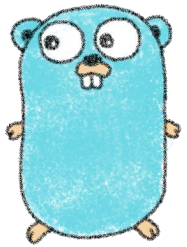
Open source framework to accelerate
the development of an Operator

Operator SDK

OPERATOR
SDK

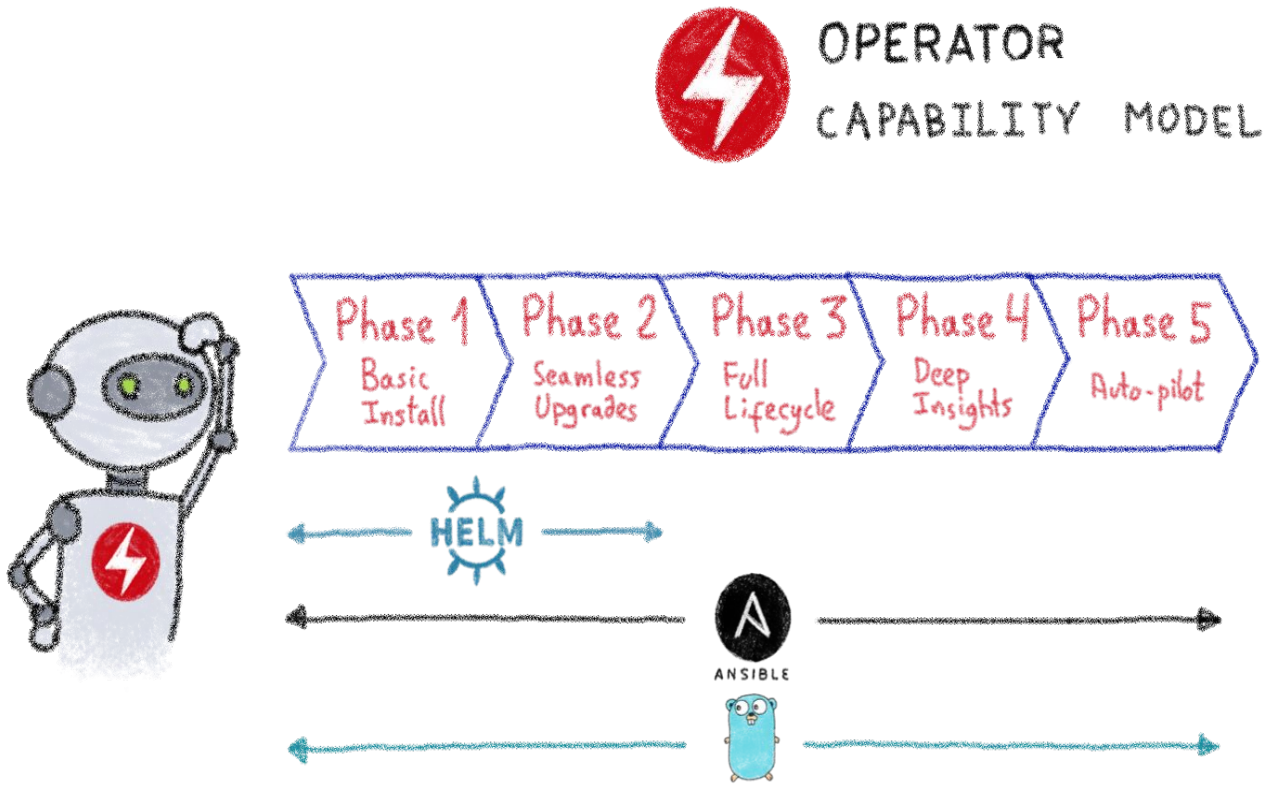
BUILD
TEST
ITERATE





Three different ways to build an Operator

Operator SDK and Capability Model

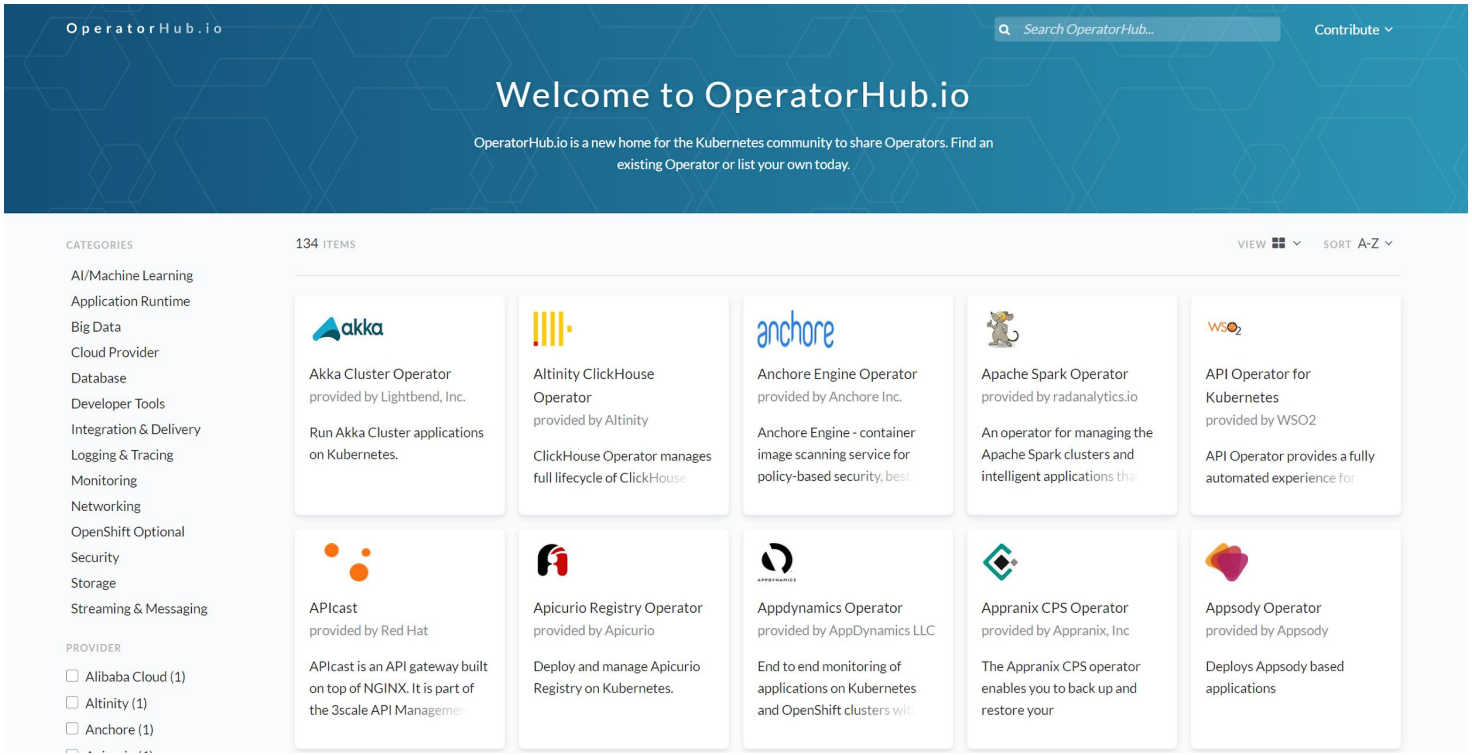


Operator Lifecycle Manager

OPERATOR
LIFECYCLE MANAGER

INSTALL
MANAGE
UPDATE

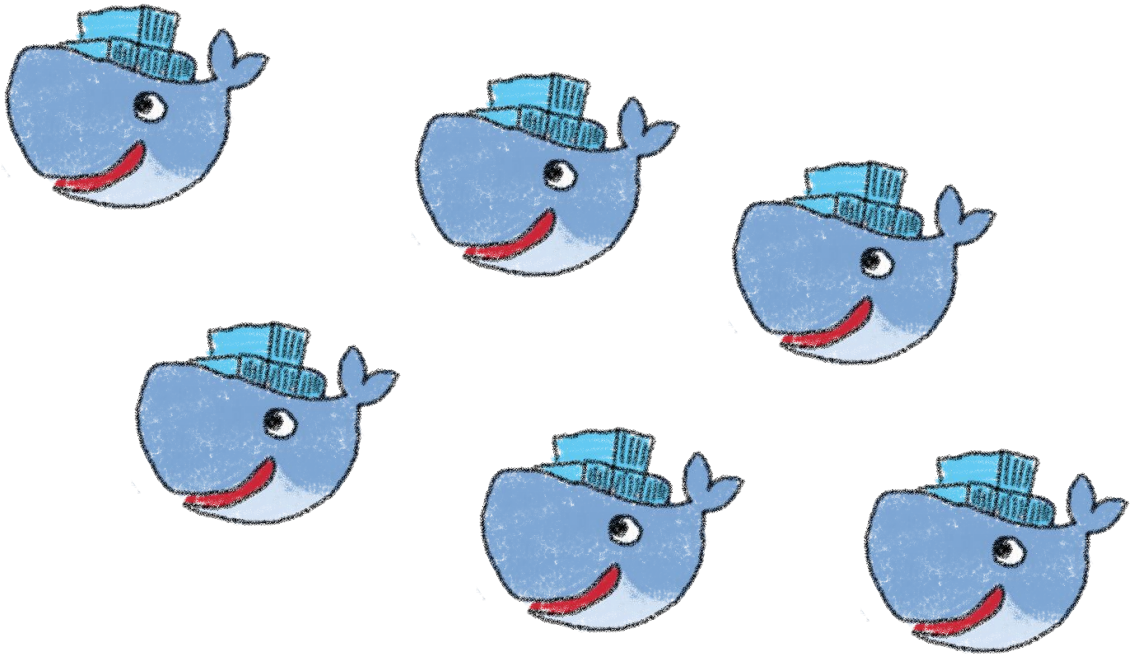
OperatorHub.io



Harbor Operator

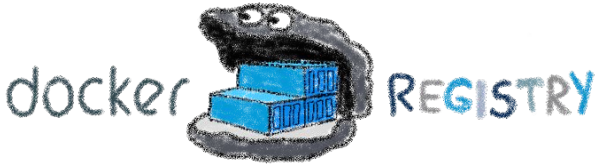
Managing private registries at scale

We wanted to build a new product



OVHcloud Managed Private Registry

Looking at the Open Source world



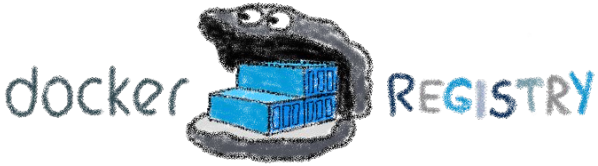
Two main alternatives around Docker Registry

Harbor has more community traction

Two main alternatives

Harbor has lots of components

NGINX



notary

But it has a Helm Chart

It should be easy to install, isn't it?

```
$ helm install harbor
```

What about configuration?

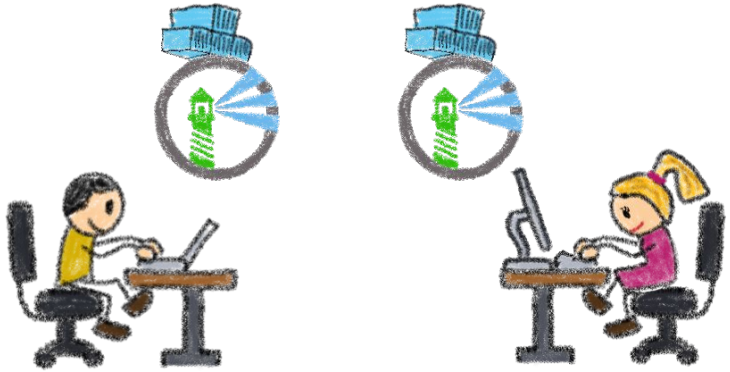
Installing a 200 GB K8s volume?

Nginx pods for routing requests?

One DB instance per customer?

Managing pods all around the cluster?

We wanted a Managed Private Registry



One Harbor instance per customer

One-click deployment, API

Shared tooling, isolated data

Ingress controller

redis

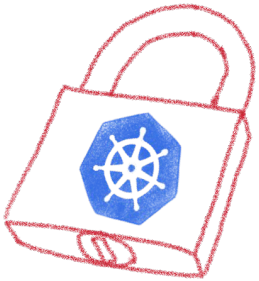
PostgreSQL

Object Storage

} as a Service

Reusing existing services

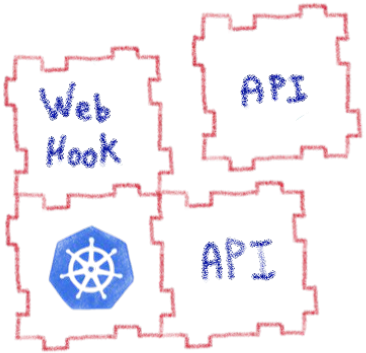
Using the platform



RBAC

Security policies

API inputs validation



Modularity & Extensibility

APIception

Web hooks

Kubernetes tooling to the rescue

Let's automate it

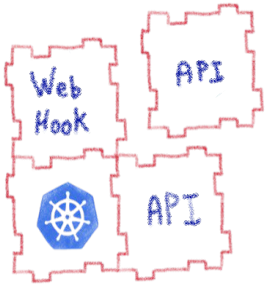
docker REGISTRY

notary

HELM

CHARTMUSEUM

PostgreSQL



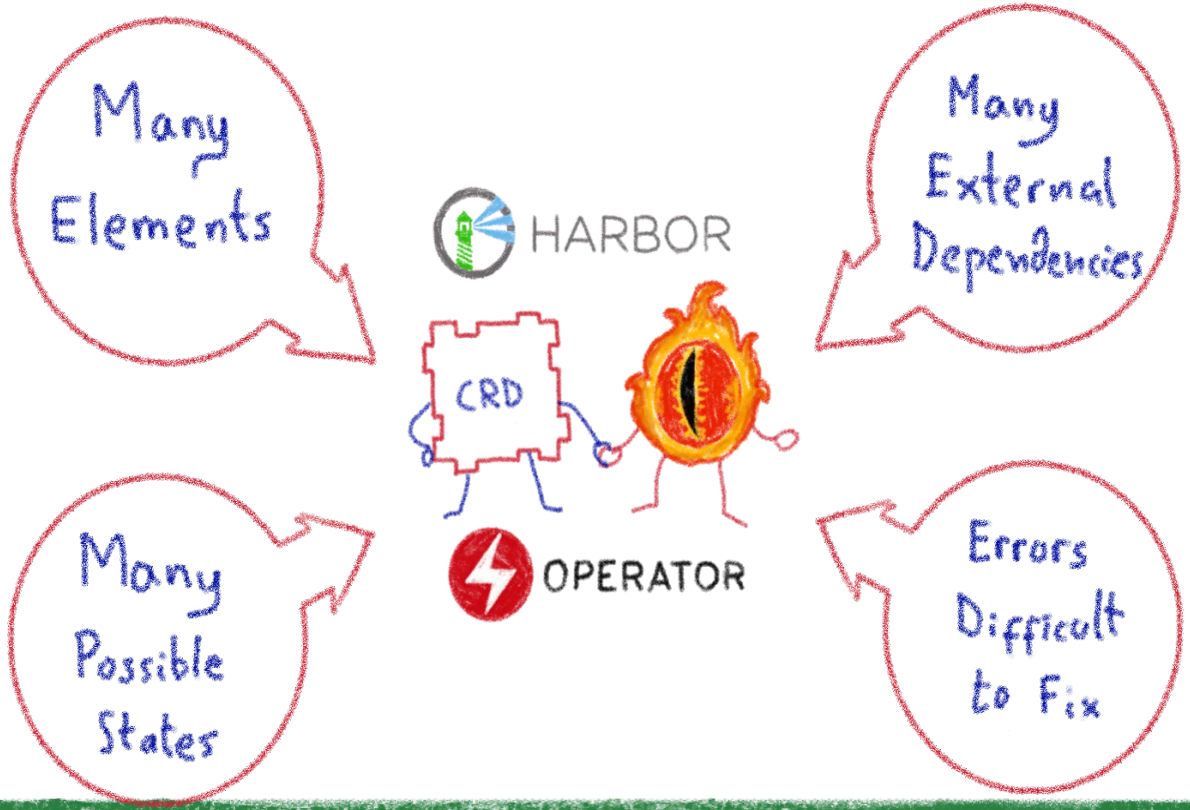
redis

We needed an operator… and there wasn't any

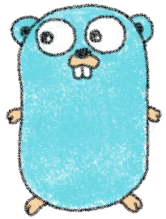
Working with the community

Harbor community also needed the operator

The challenge: reconciliation loop



The Harbor Operator



Written in Go

7 Components
- Config Map
- Secrets
- Ingress
- Certificates
- Deployments
- Services

1 CRD & 1 Controller

kubebuilder

OPENTRACING

Uses other operators for specific tasks (e.g. Cert Manager)

It's Open Source

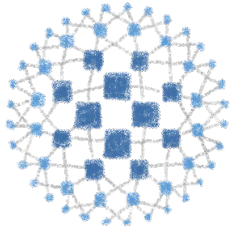
Donated by OVHcloud to the

CLOUD NATIVE COMPUTING FOUNDATION

https://github.com/goharbor/harbor-operator

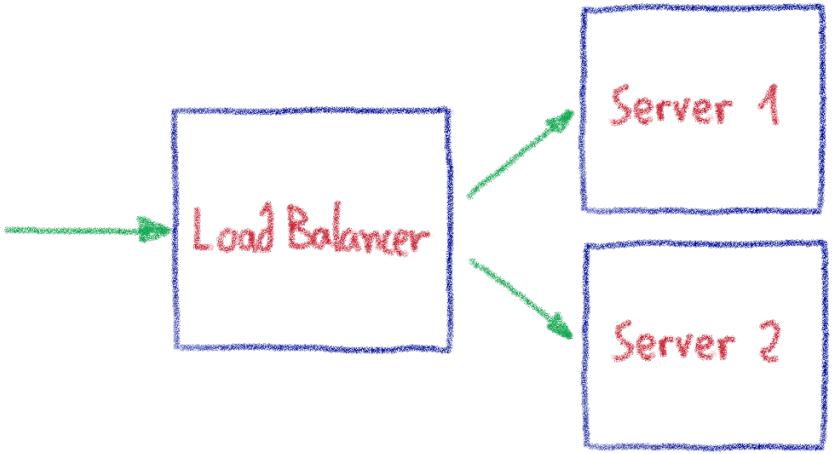
LoadBalancer Operator

A managed LoadBalancer at scale



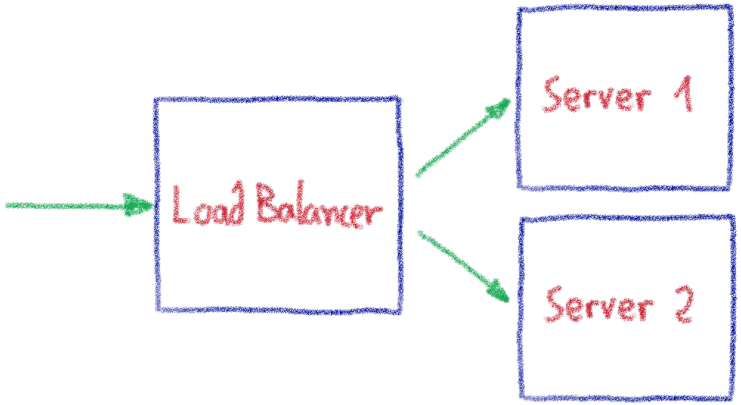
HAPROXY

Load Balancer: a critical cog



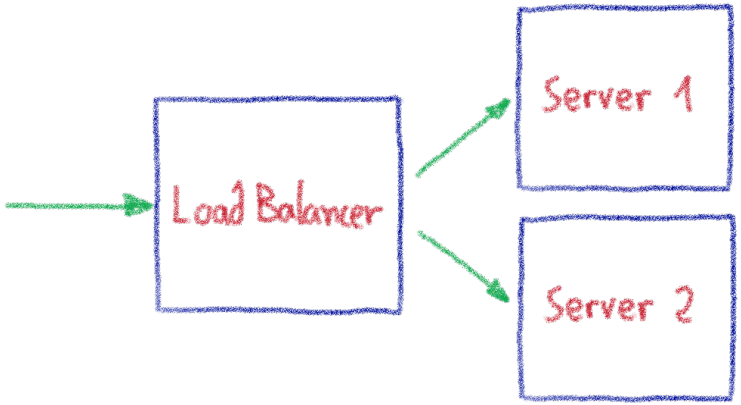
Cornerstone of any Cloud Provider's infrastructure

Our legacy Load Balancer stack



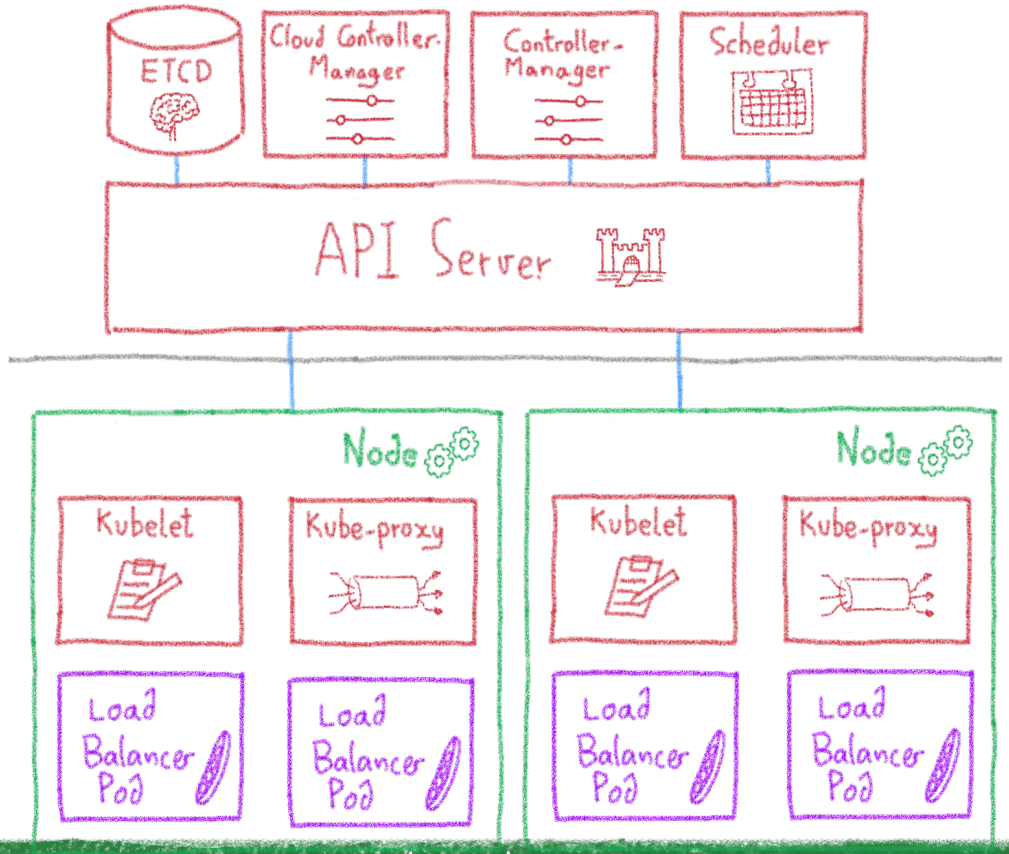
- Excellent performances
  - Built on bare metal servers + BGP
  - Custom made servers tuned for network traffic
- Carry the TLS termination
  - SSL / LetsEncrypt
- Not cloud ready
  - Piloted by configuration files
  - Long configuration loading time
- Custom made hardware
  - Slower to build
  - Needs to be deployed on 30 datacenters

Our needs for a new Load Balancer

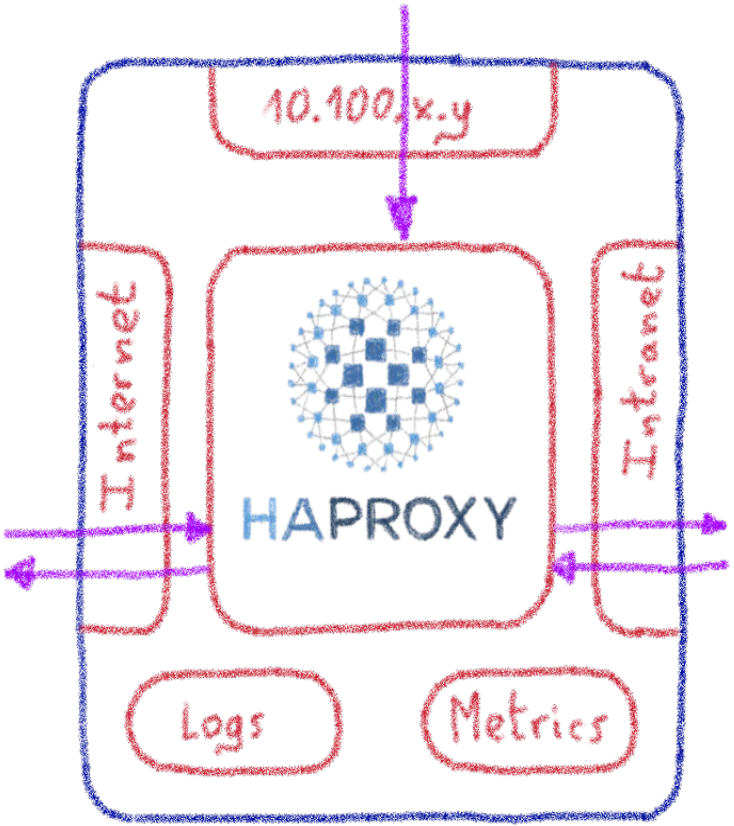


- Supporting mass update
- Quickly reconfigurable
- Available anywhere quickly
- Easily operable
- Integrated into our Public Cloud

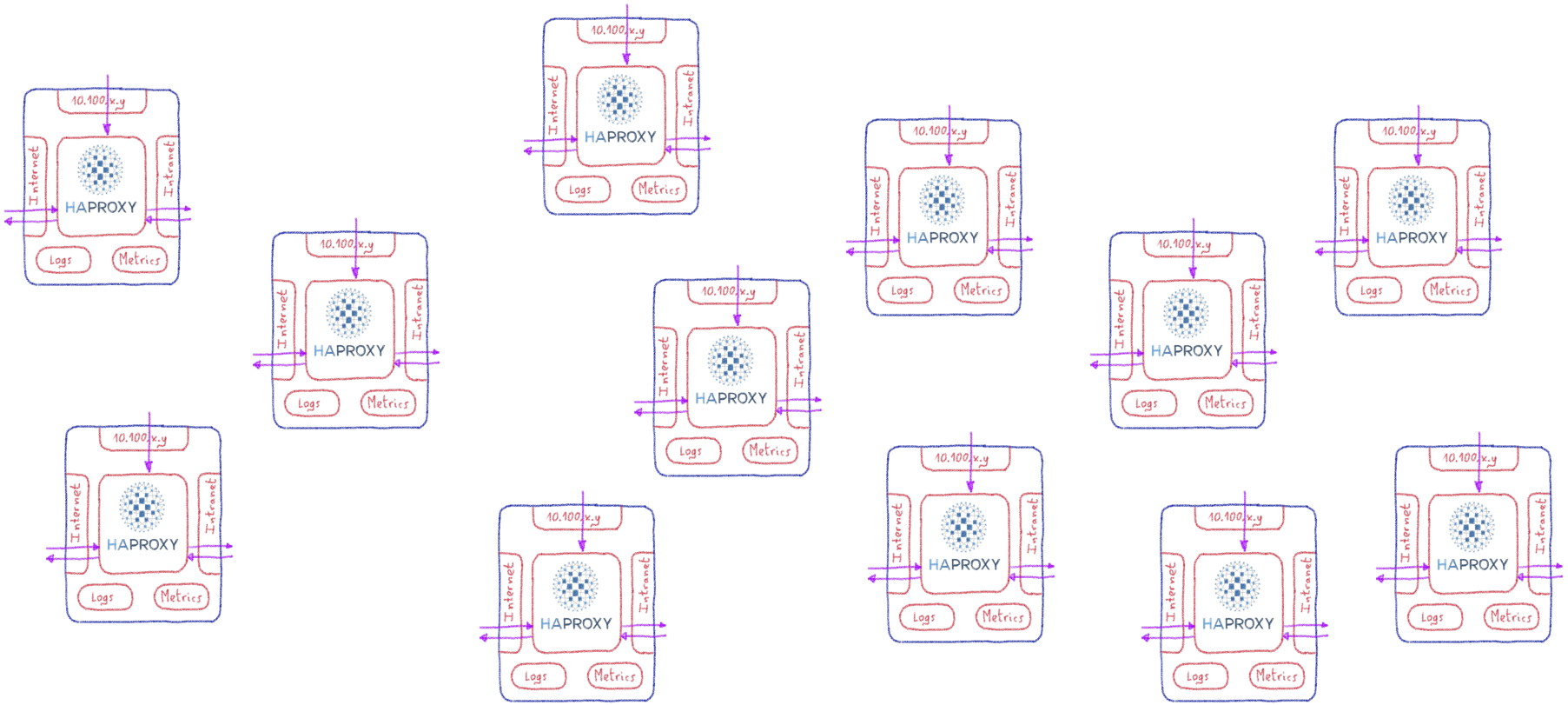
Building it on Kubernetes



A Load Balancer in a pod

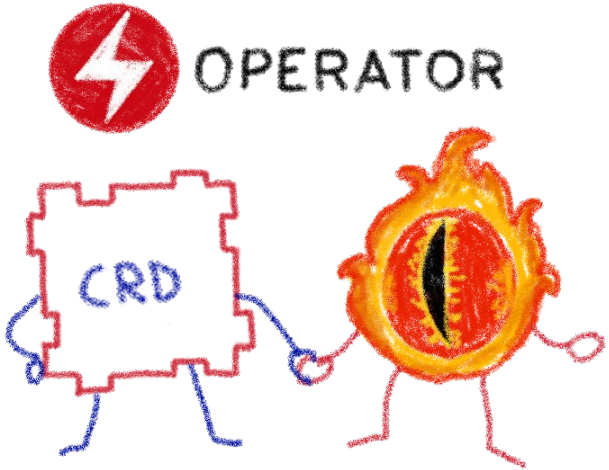


Orchestrating one million LBs...



`kubectl apply -f lb` is not an option!

We needed an Operator

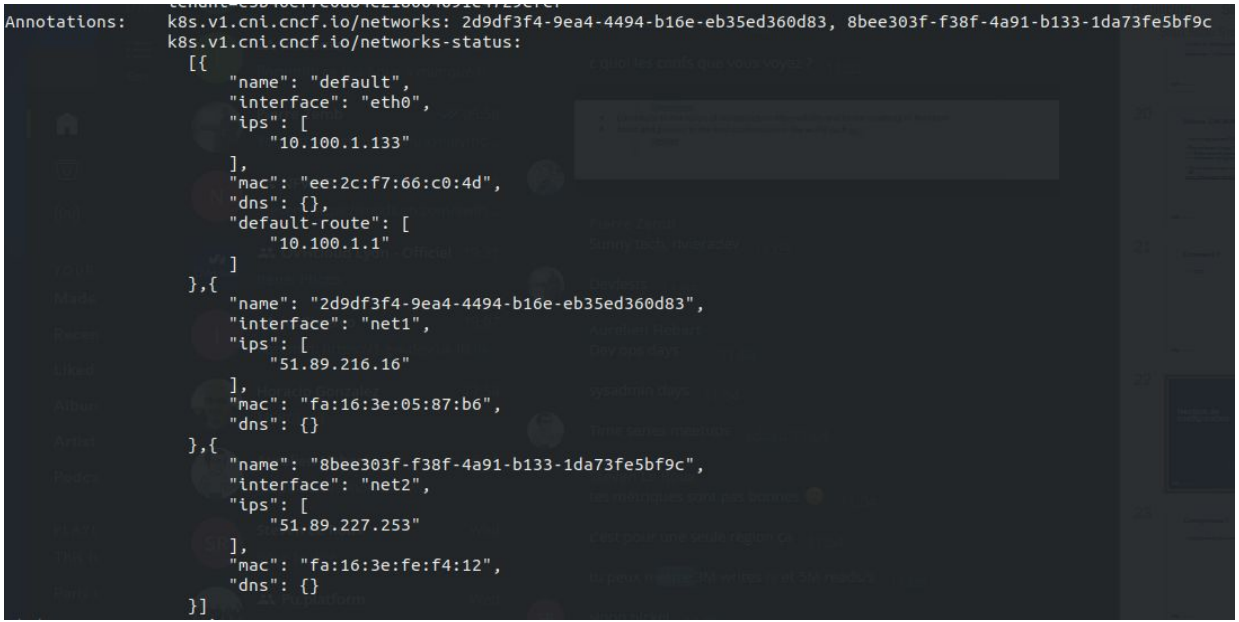


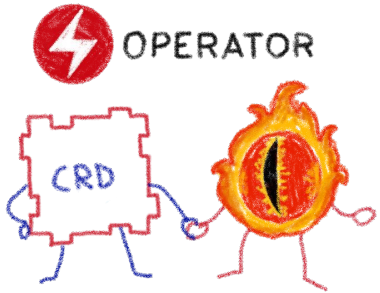
Network: multus-cni

Attaching multiple network interfaces to pods:

Bridge + Host-local

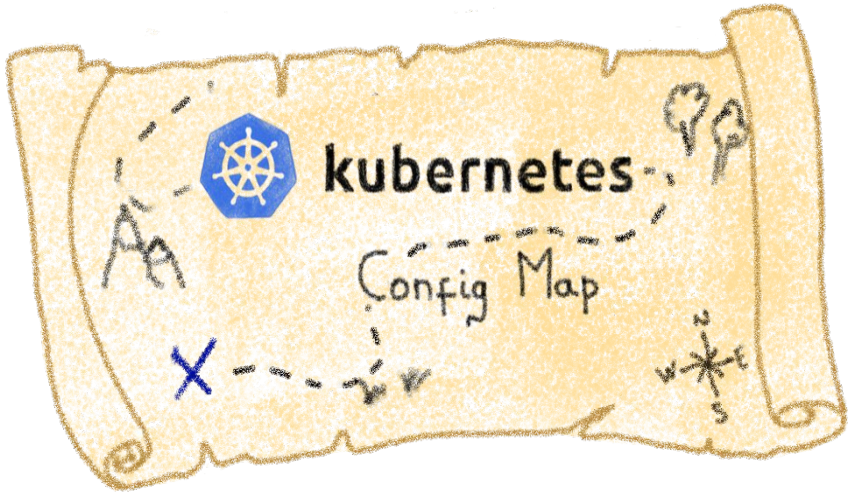
Adding network interfaces on the fly





Using annotations to add interfaces to pod

Config management



Using Config Map

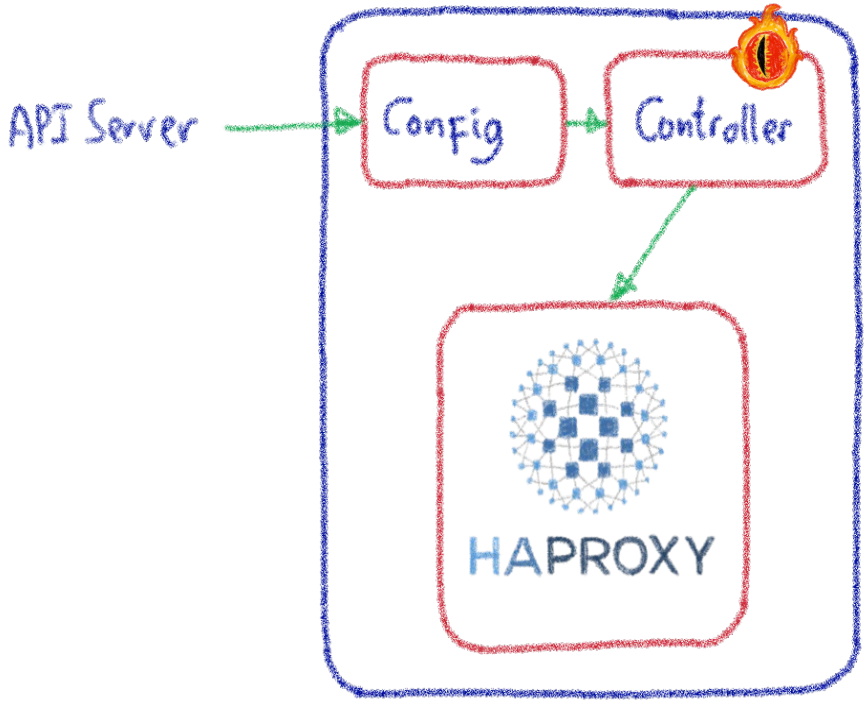
How to detect a change on Config Map files?

Watch + Trigger?

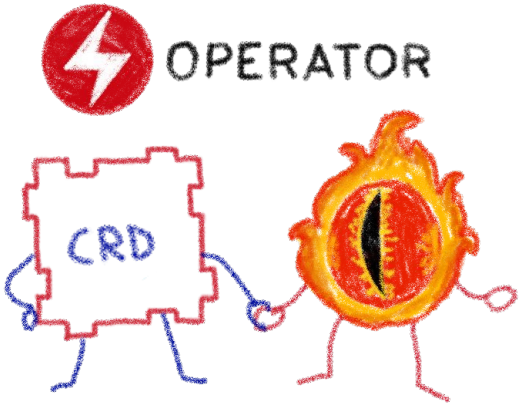
More information on Config Map working

`martensson.io/go-fsnotify-and-kubernetes-configmaps`

A Controller to watch and trigger



Observability



Tried Prometheus Operator, limited to one container per pod
Switched to Warp 10 with Beamium Operator

That's all, folks!

Thank you all!