OVHcloud Kubernetes Initiation Tech Lab

Horacio Gonzalez

2023-06-05 - Madrid

@LostInBrittany

Who are we?

Introducing myself and introducing OVHcloud

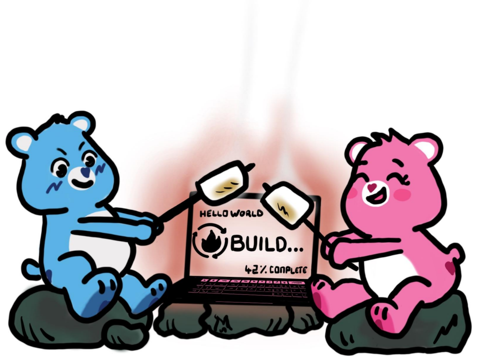
Horacio Gonzalez

@LostInBrittany

Spaniard Lost in Brittany

OVHcloud

DevRel Leader



DevFest du
Bout du Monde

Google Developers
Experts
2019

Web Technologies
GDE

Flutter

OVHcloud

- **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

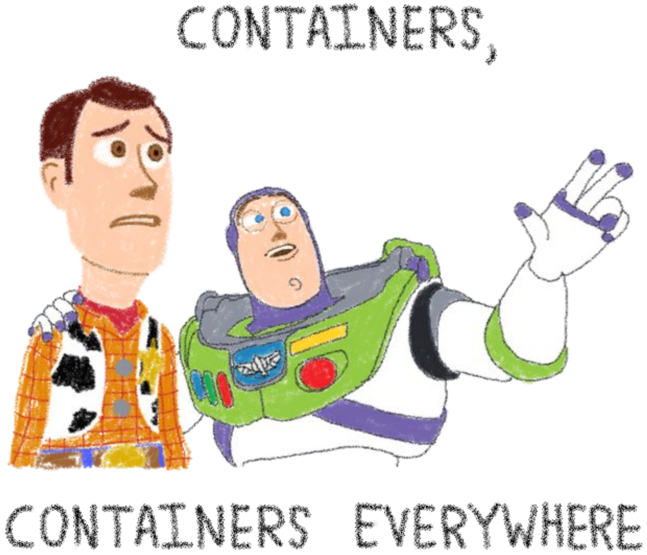
1.5 Billion Euros Invested
since 2016

P.U.E. 1.09
Energy efficiency indicator

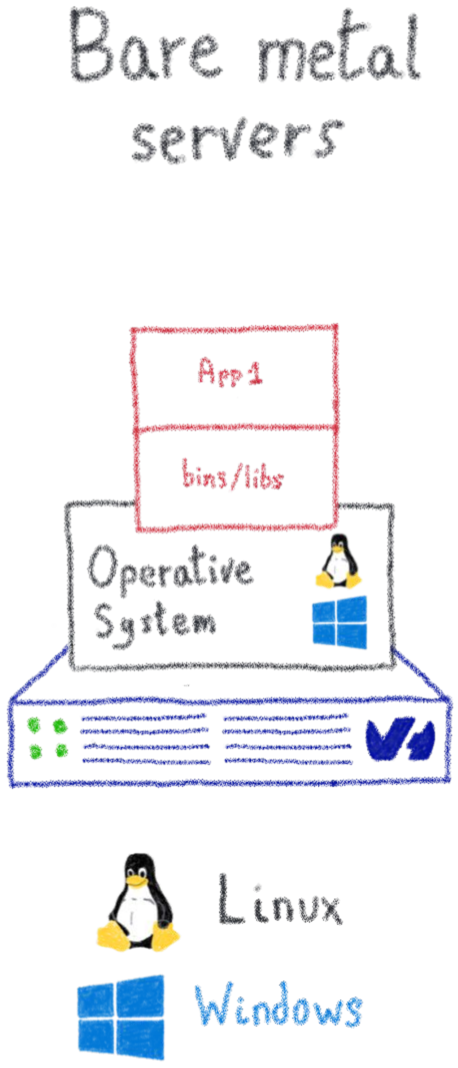
20+ Years in Business
Disrupting since 1999

Why do we need Kubernetes?

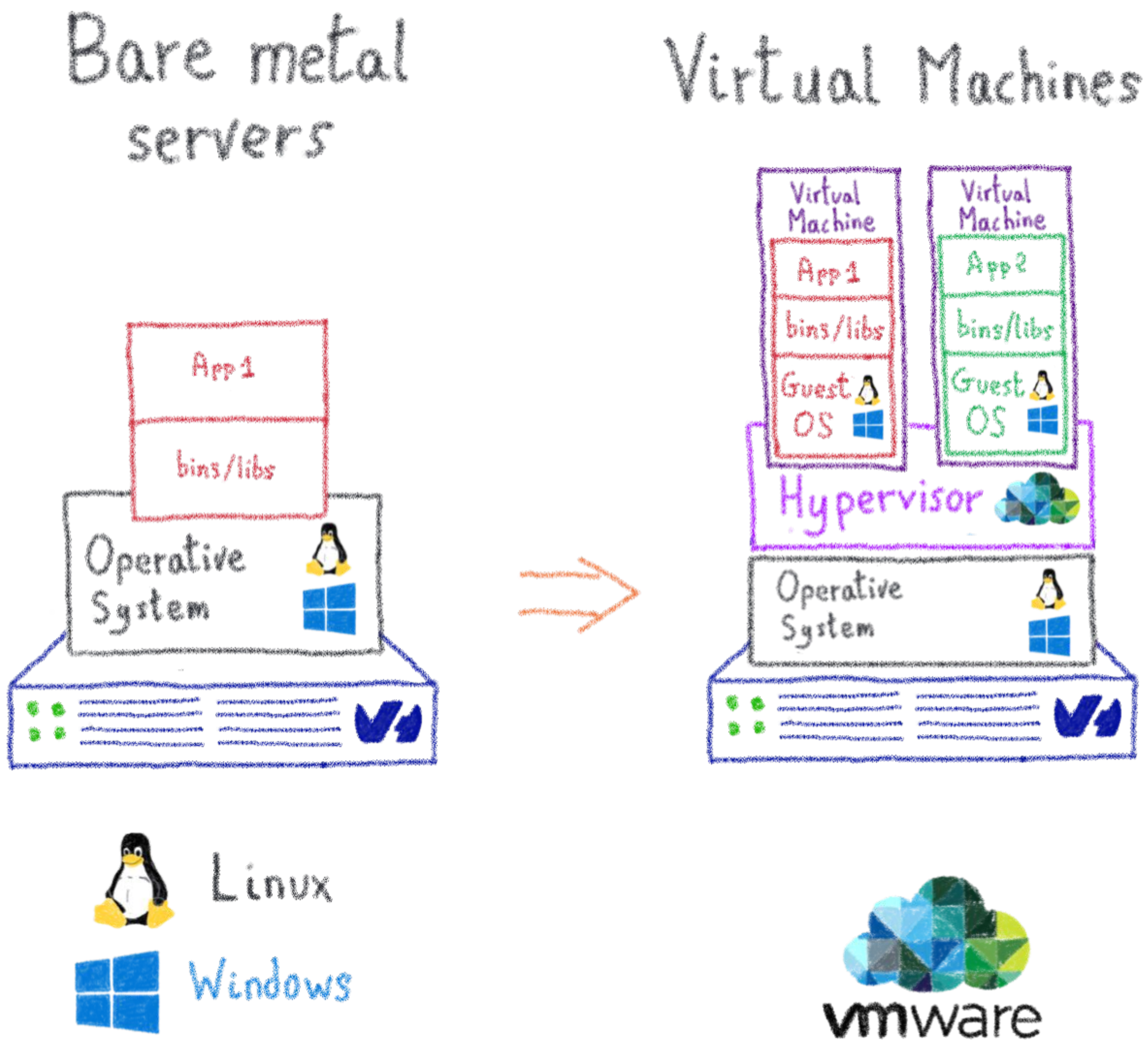
Taming the complexity of operating containers



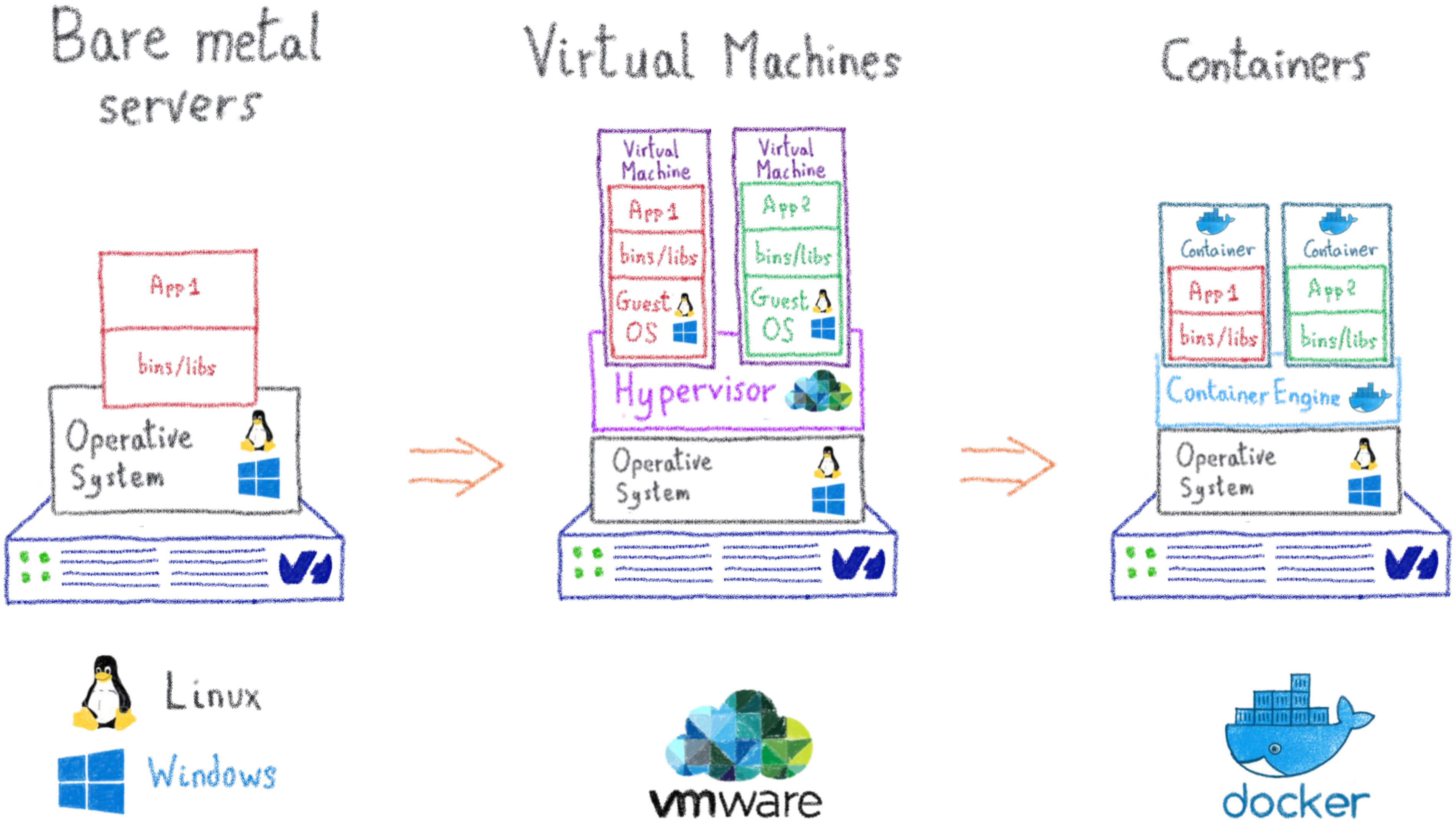
From bare metal to containers



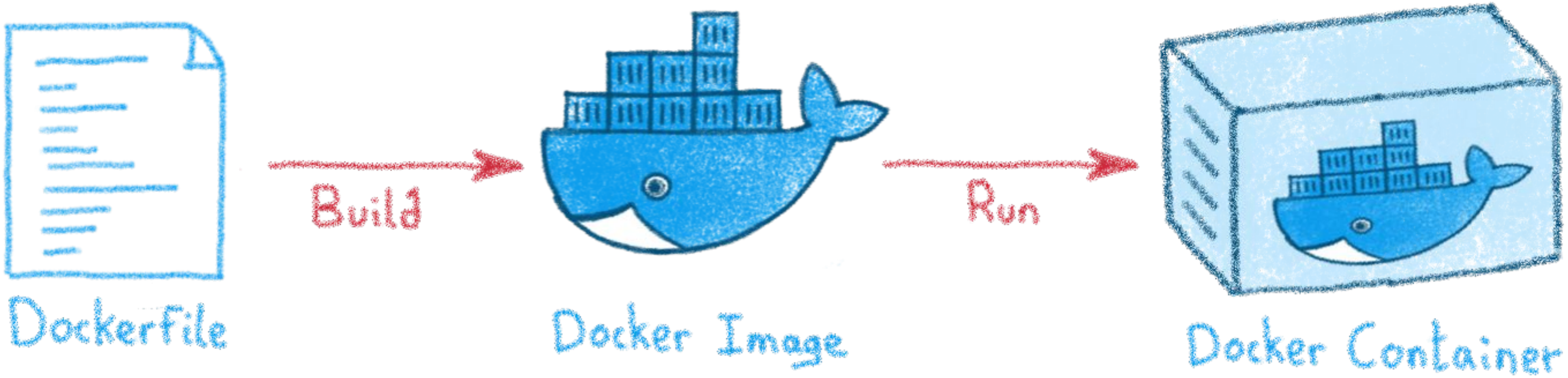
From bare metal to containers



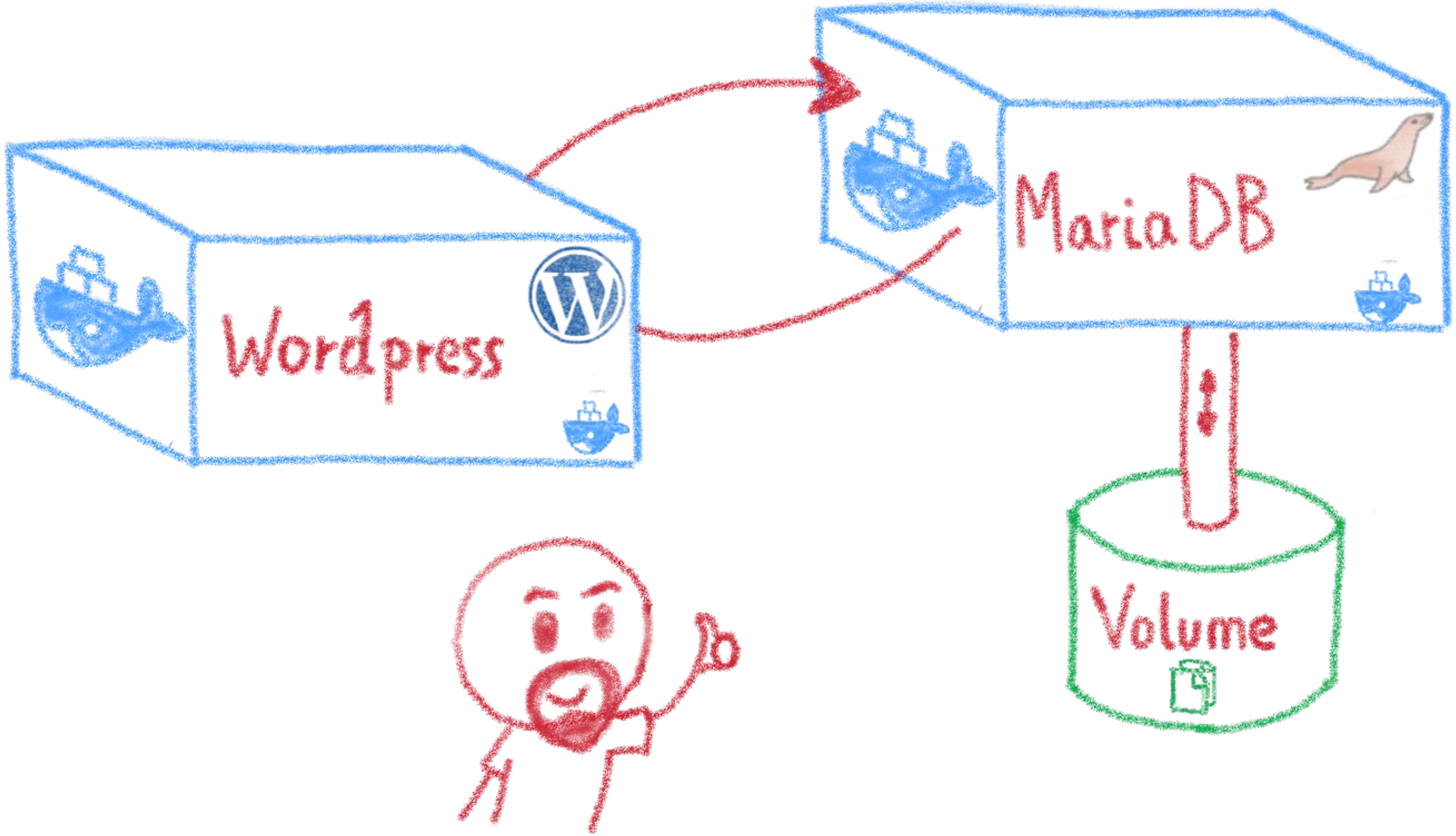
From bare metal to containers



Dockerfiles, images and containers

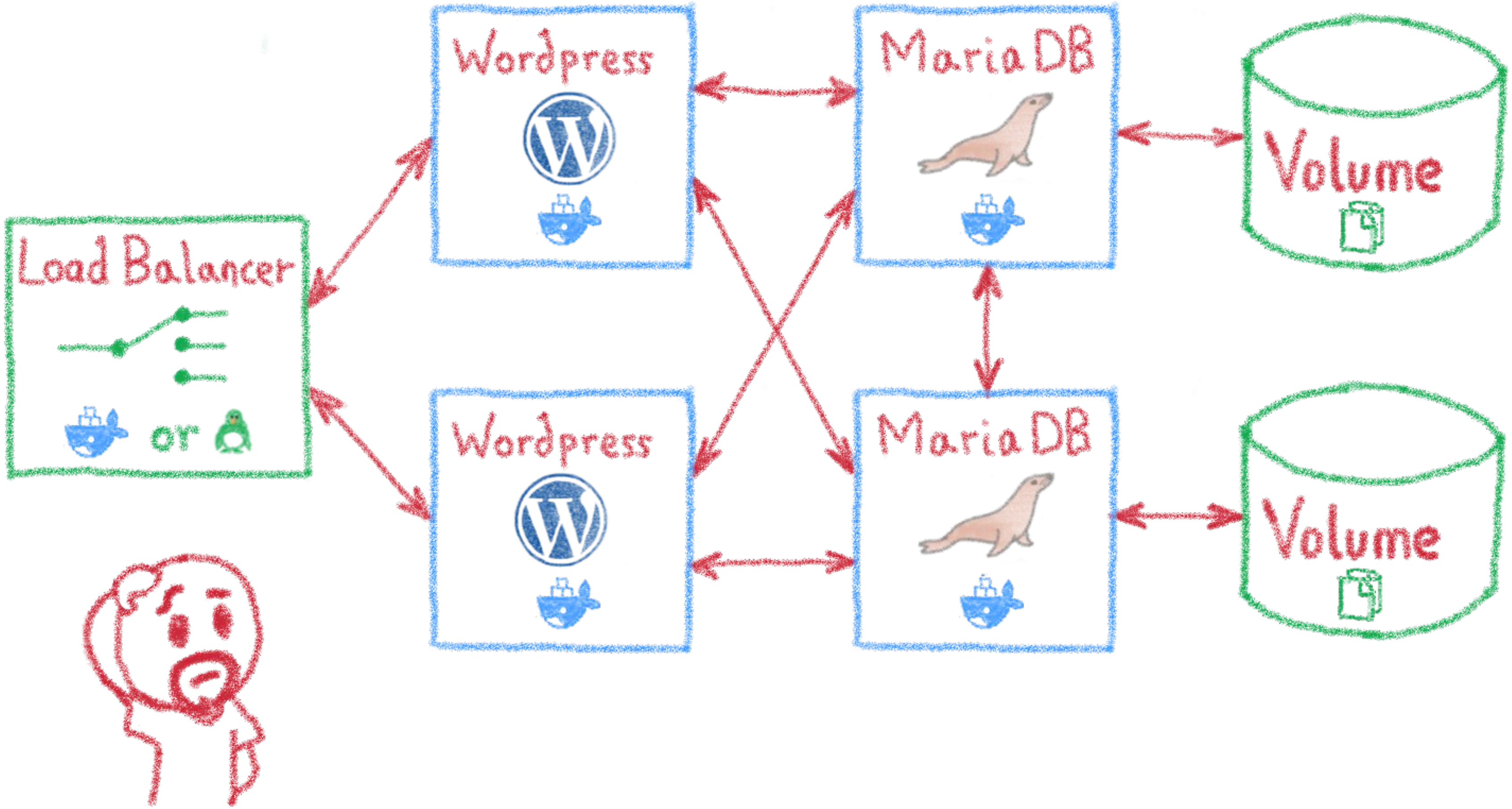


Containers are easy...



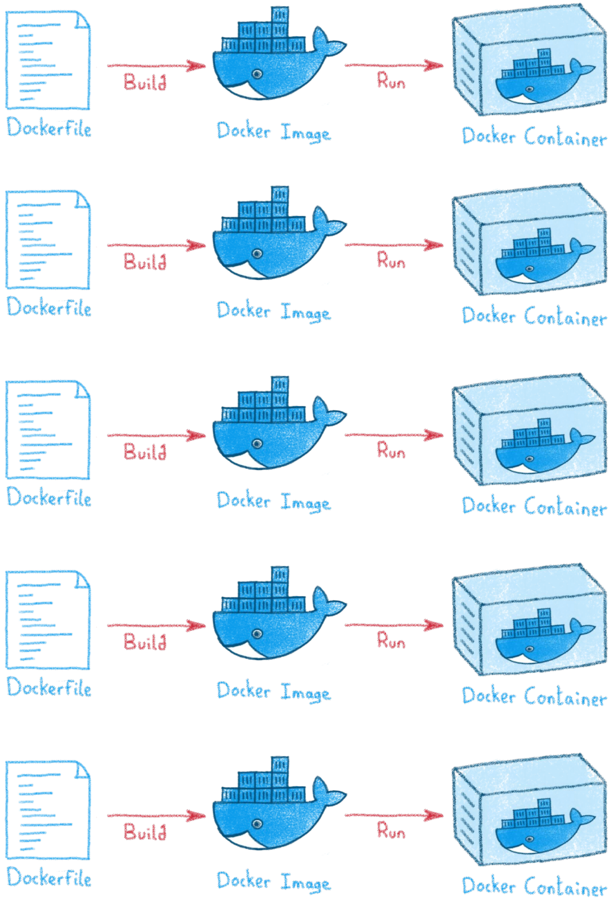
For developers

Less simple if you must operate them



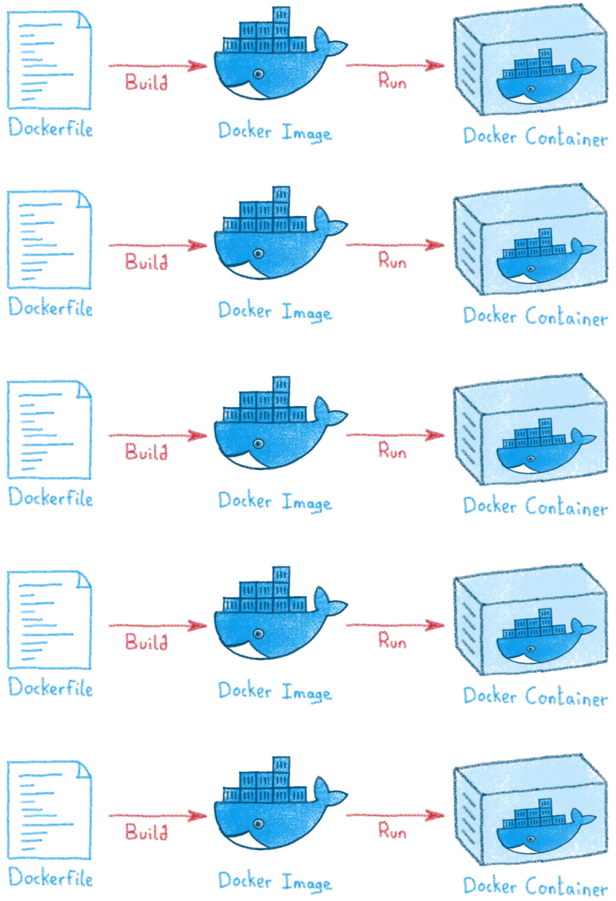
Like in a production context

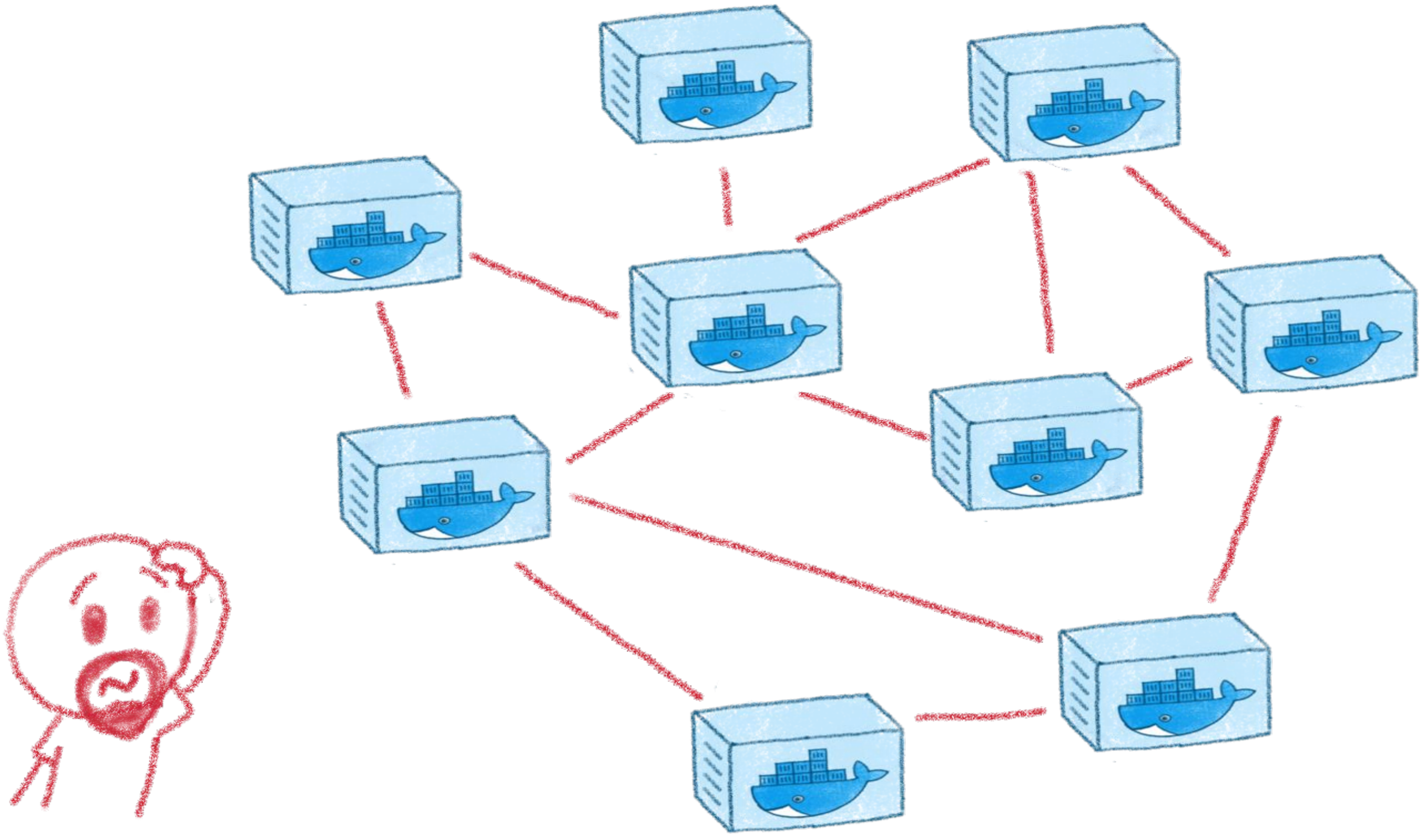
And what about microservices?



Are you sure you want to operate them by hand?

And what about microservices?



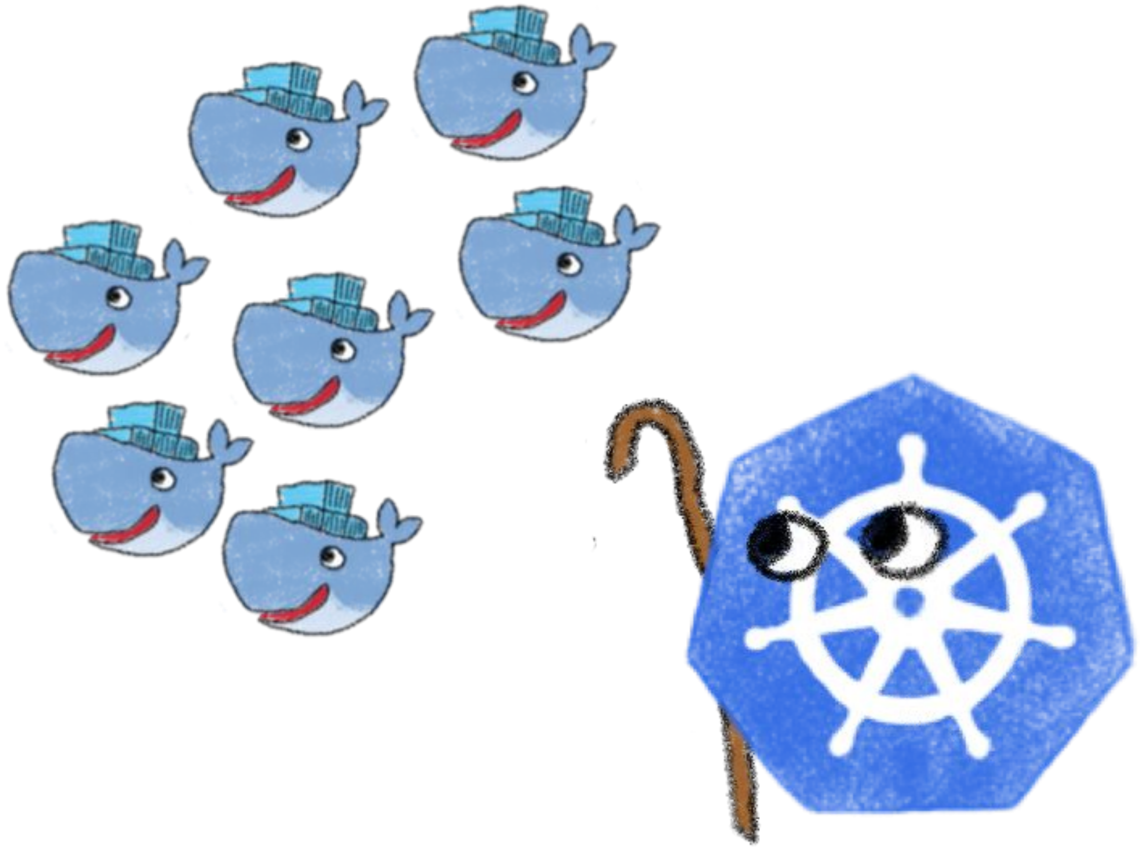


Are you sure you want to operate them by hand?

Kubernetes: a full orchestrator

Takes care of:

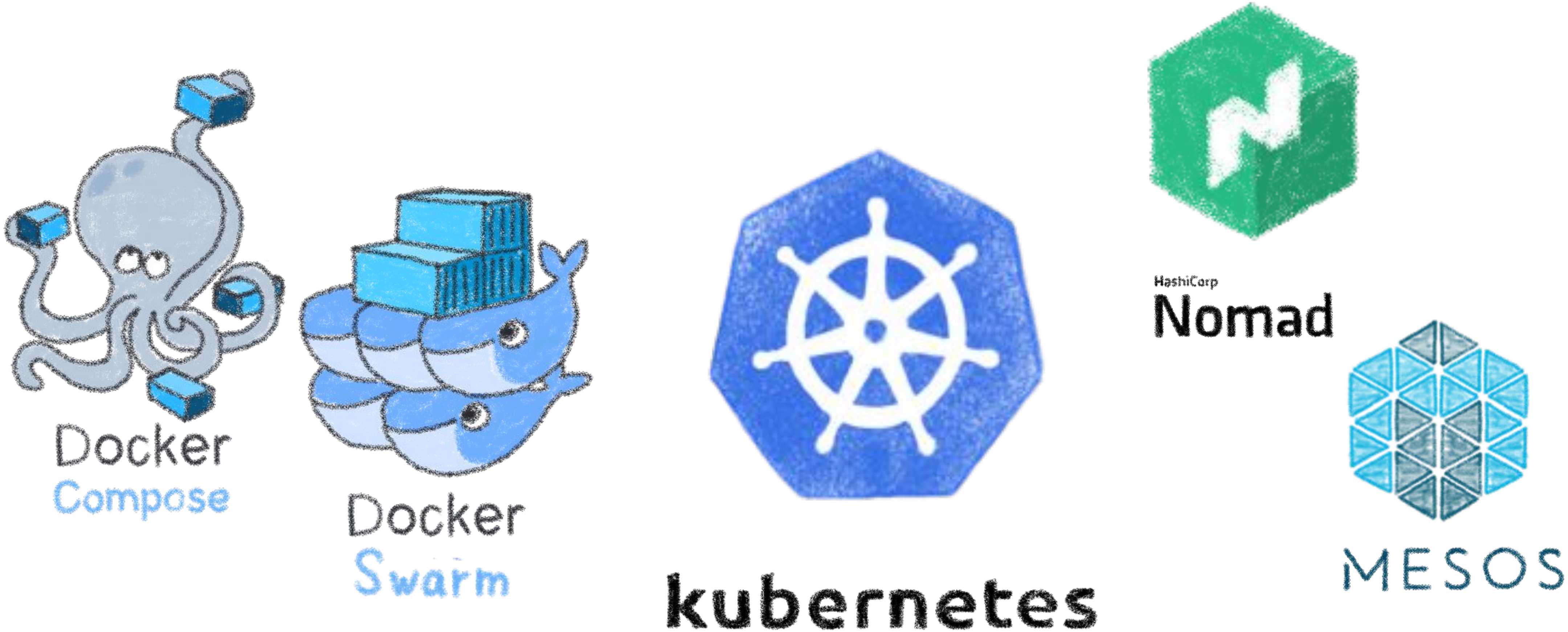
- Deployment
- Scaling
- Monitoring
- Repairing
- Securing
- ...



OVHcloud

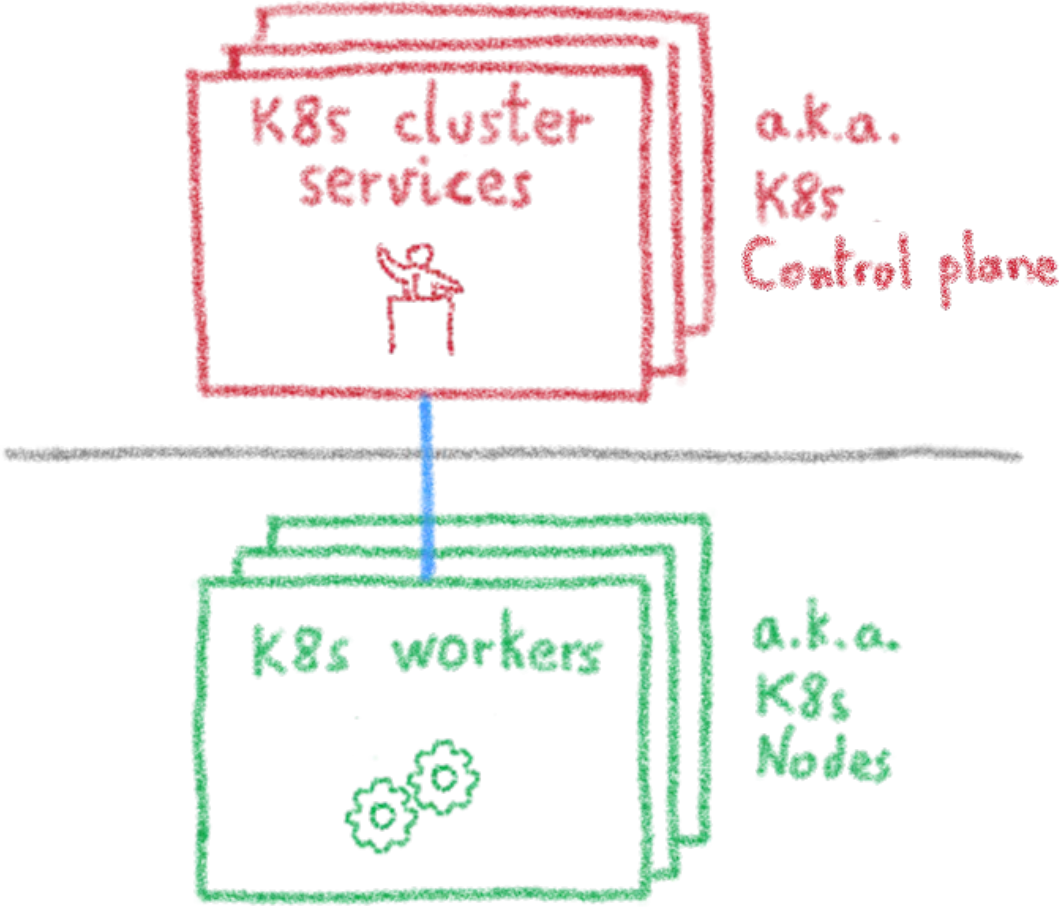
@LostInBrittany

Not the only orchestrator

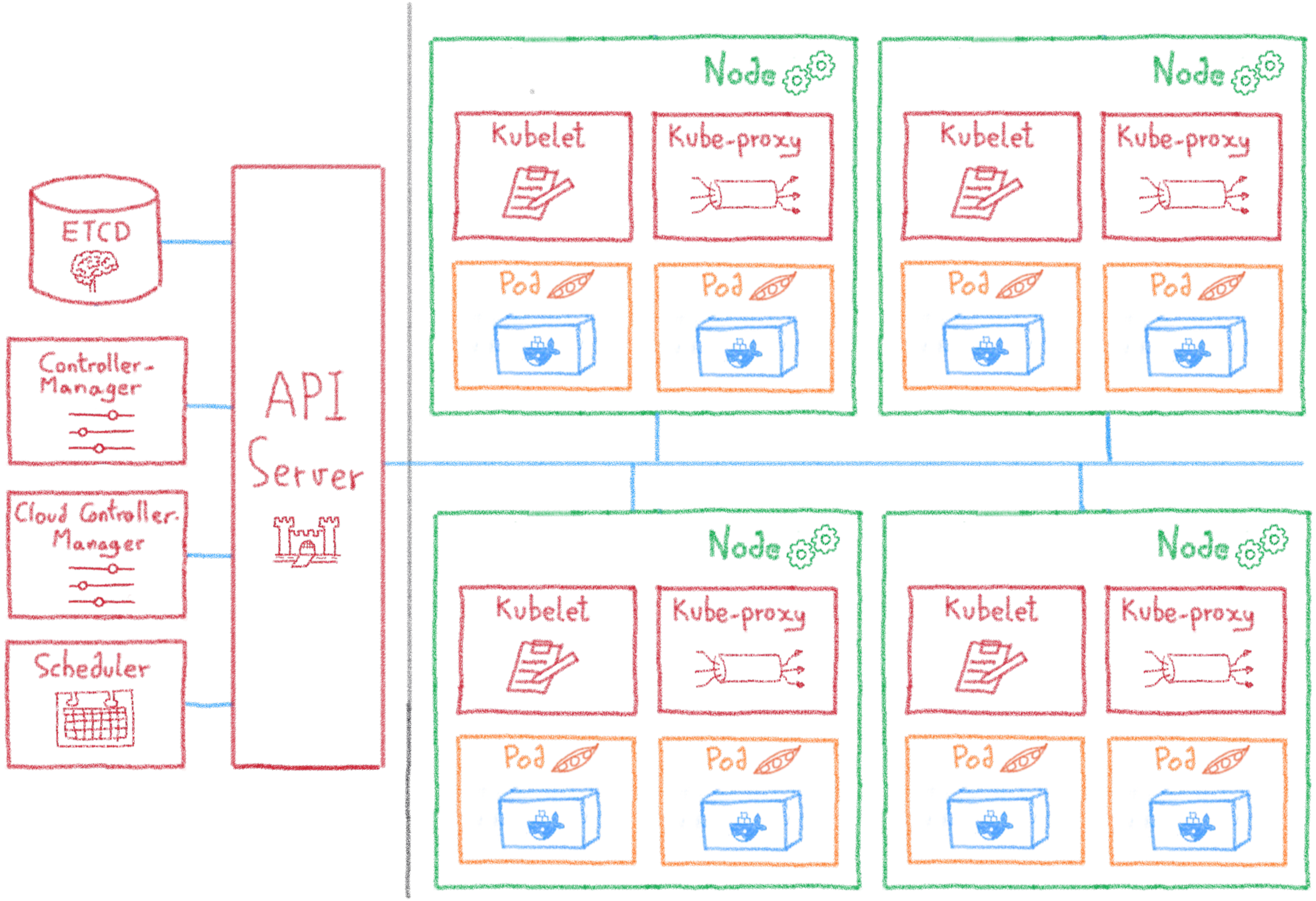


But the most popular one…

Kubernetes cluster: masters and nodes



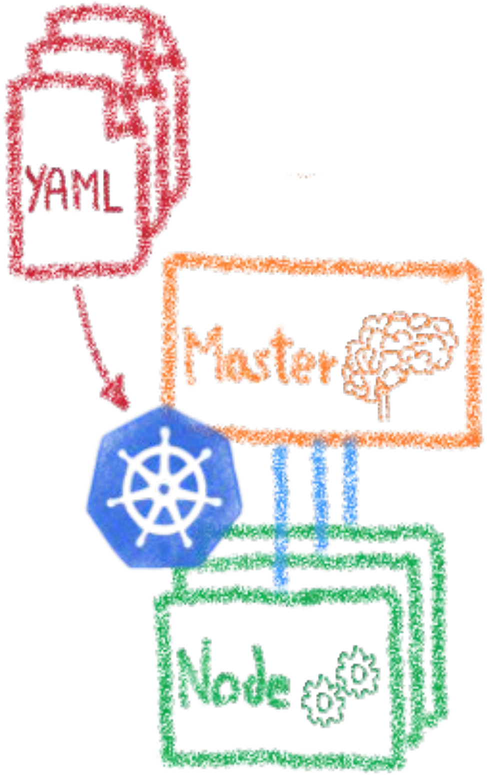
Kubernetes cluster: more details



OVHcloud

@LostInBrittany

Desired State Management



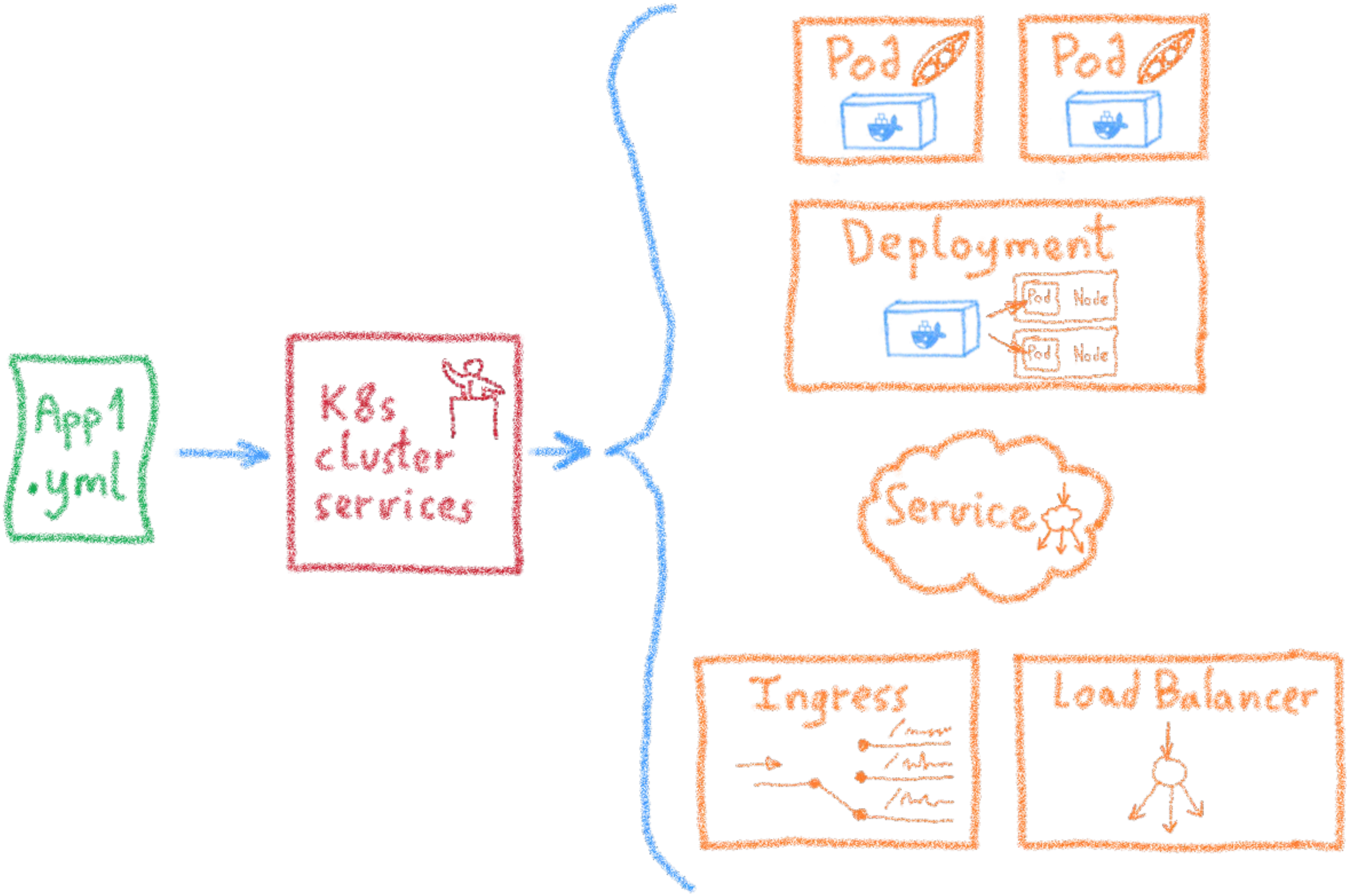
Manifest files:

Text files in YAML format

High-level description of the target architecture

Declarative infrastructure

Desired State Management



Ingress

Services

Deployments

Pods

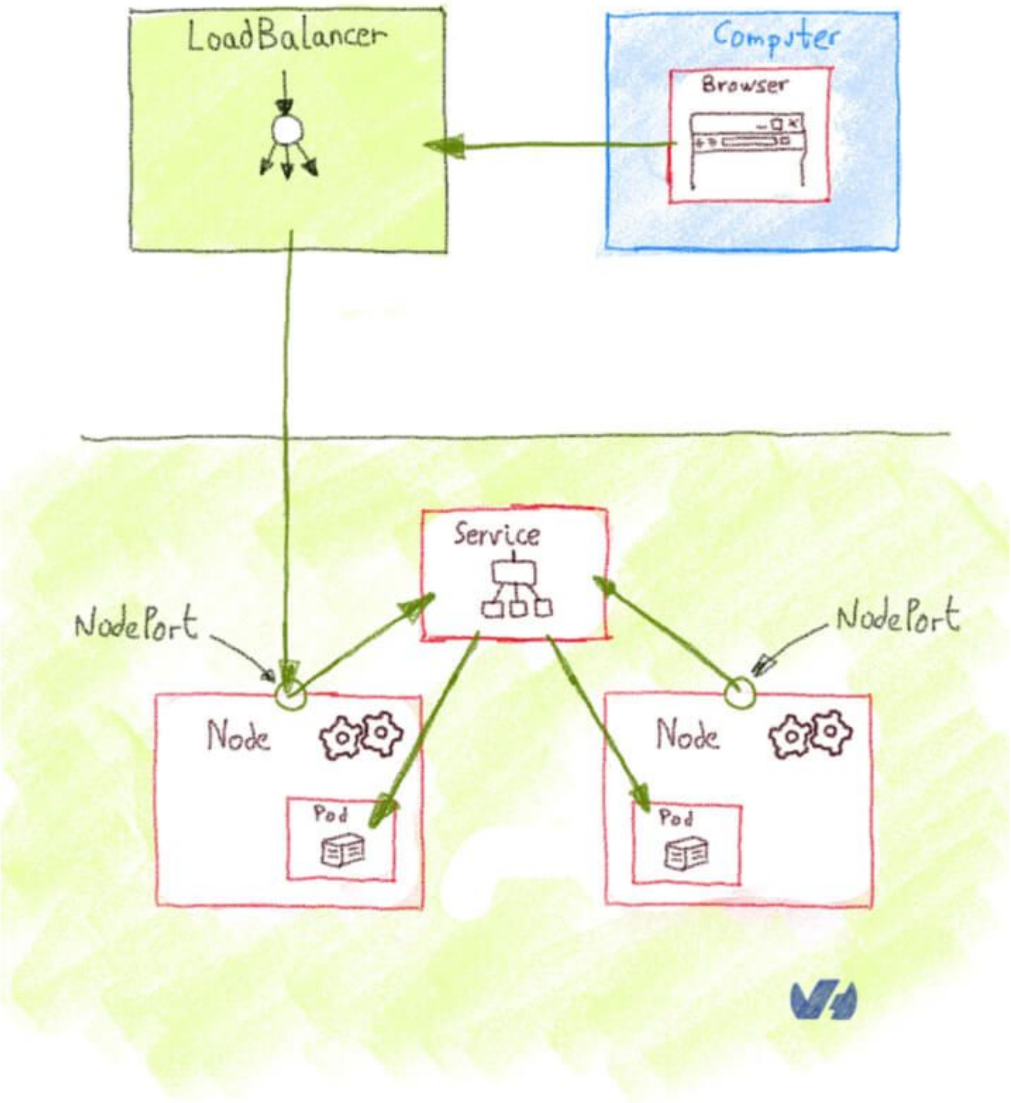
Sidecars

Replica Sets

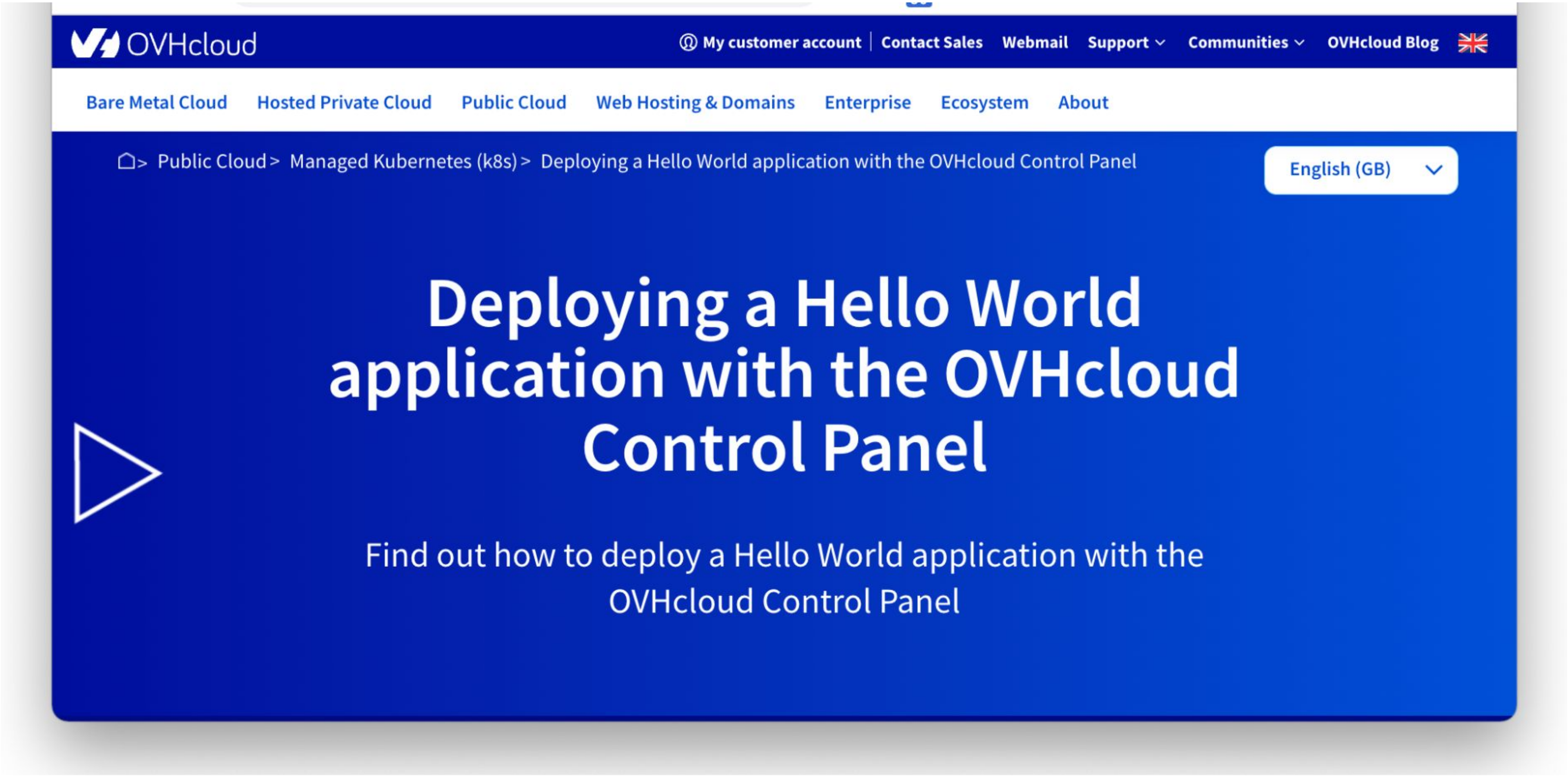
OVHcloud

@LostInBrittany

Let's deploy an application



Demo: Hello Kubernetes World



OVHcloud

My customer account | Contact Sales | Webmail | Support | Communities | OVHcloud Blog

Bare Metal Cloud | Hosted Private Cloud | Public Cloud | Web Hosting & Domains | Enterprise | Ecosystem | About

> Public Cloud > Managed Kubernetes (k8s) > Deploying a Hello World application with the OVHcloud Control Panel

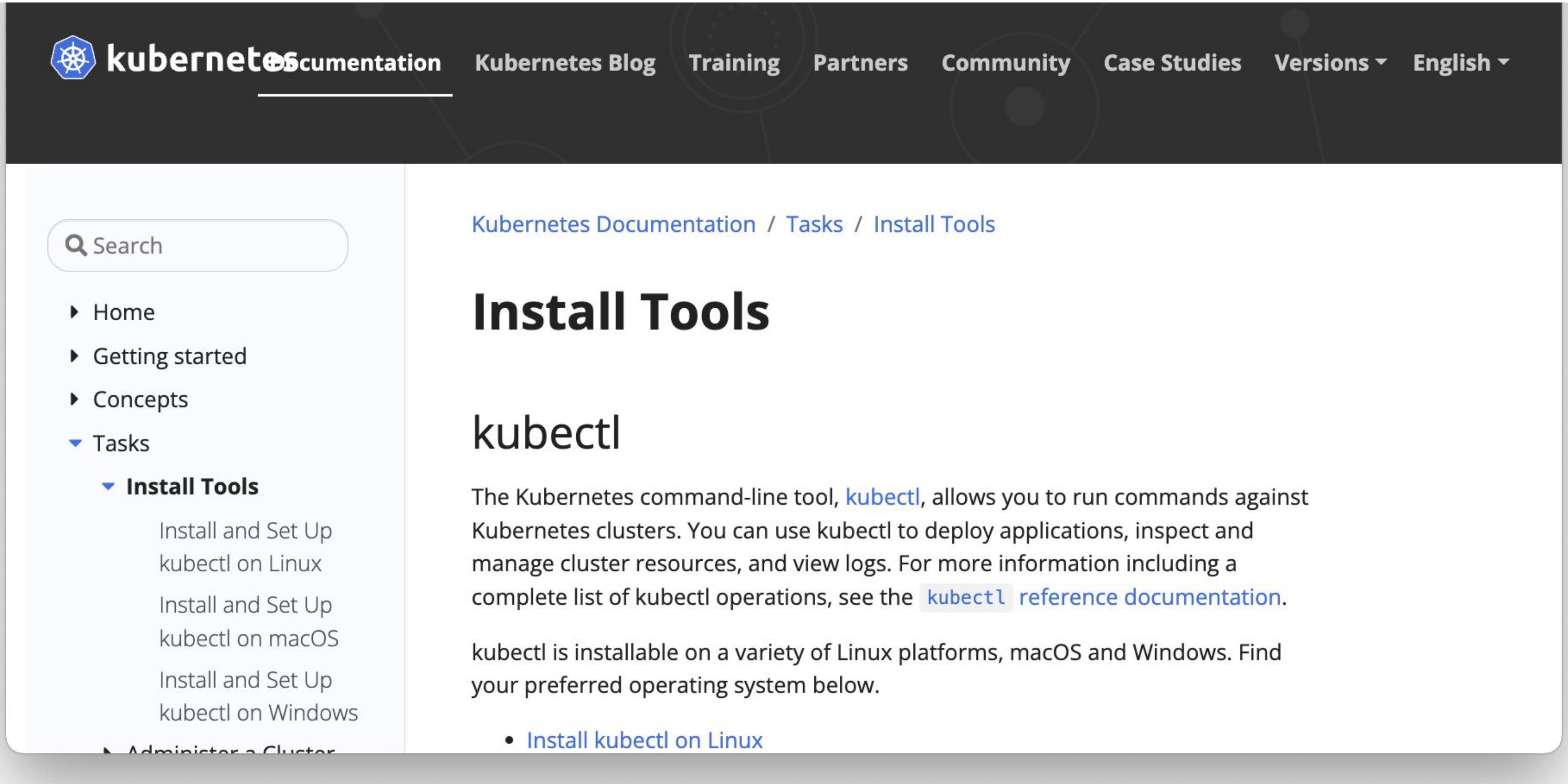
English (GB)

Deploying a Hello World application with the OVHcloud Control Panel

Find out how to deploy a Hello World application with the OVHcloud Control Panel

https://docs.ovh.com/gb/en/kubernetes/deploying-hello-world/

Needed tools: kubectl



Kubernetes Documentation / Tasks / Install Tools

Install Tools

kubectl

The Kubernetes command-line tool, kubectl, allows you to run commands against Kubernetes clusters. You can use kubectl to deploy applications, inspect and manage cluster resources, and view logs. For more information including a complete list of kubectl operations, see the `kubectl` reference documentation.

kubectl is installable on a variety of Linux platforms, macOS and Windows. Find your preferred operating system below.

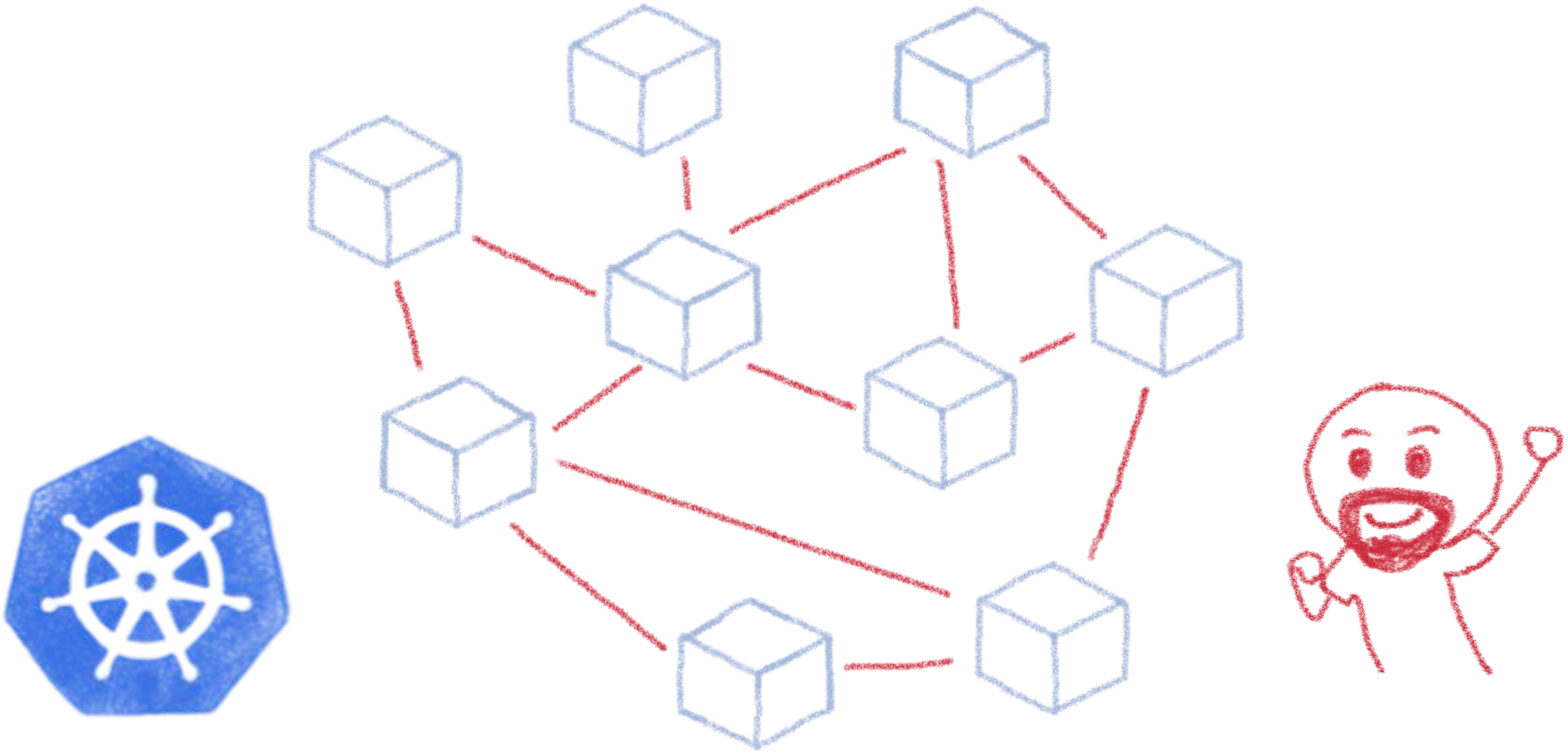
- Install kubectl on Linux

https://kubernetes.io/docs/tasks/tools/

Putting Kubernetes in production

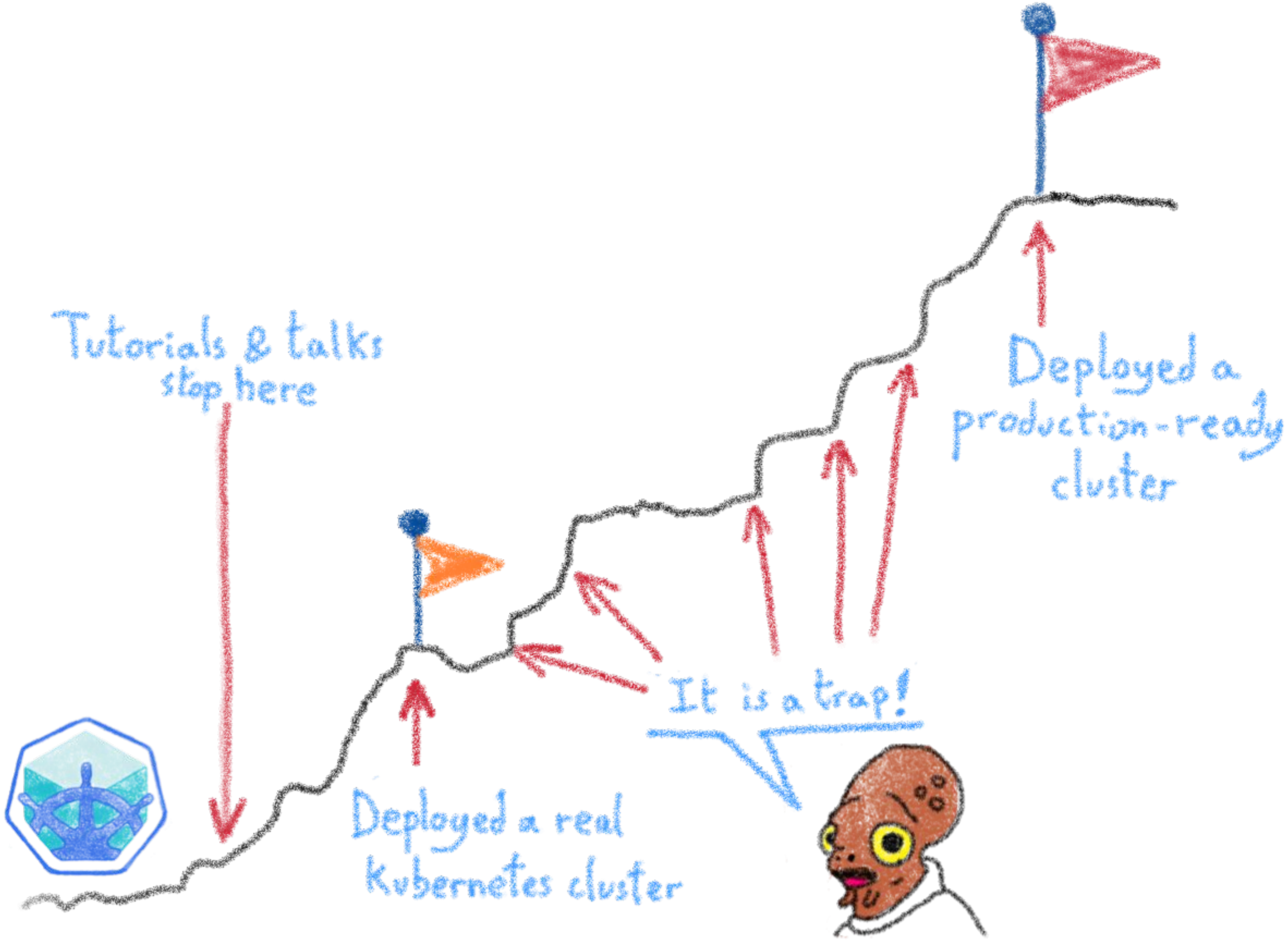
A journey not for the faint of heart

Kubernetes can be wonderful

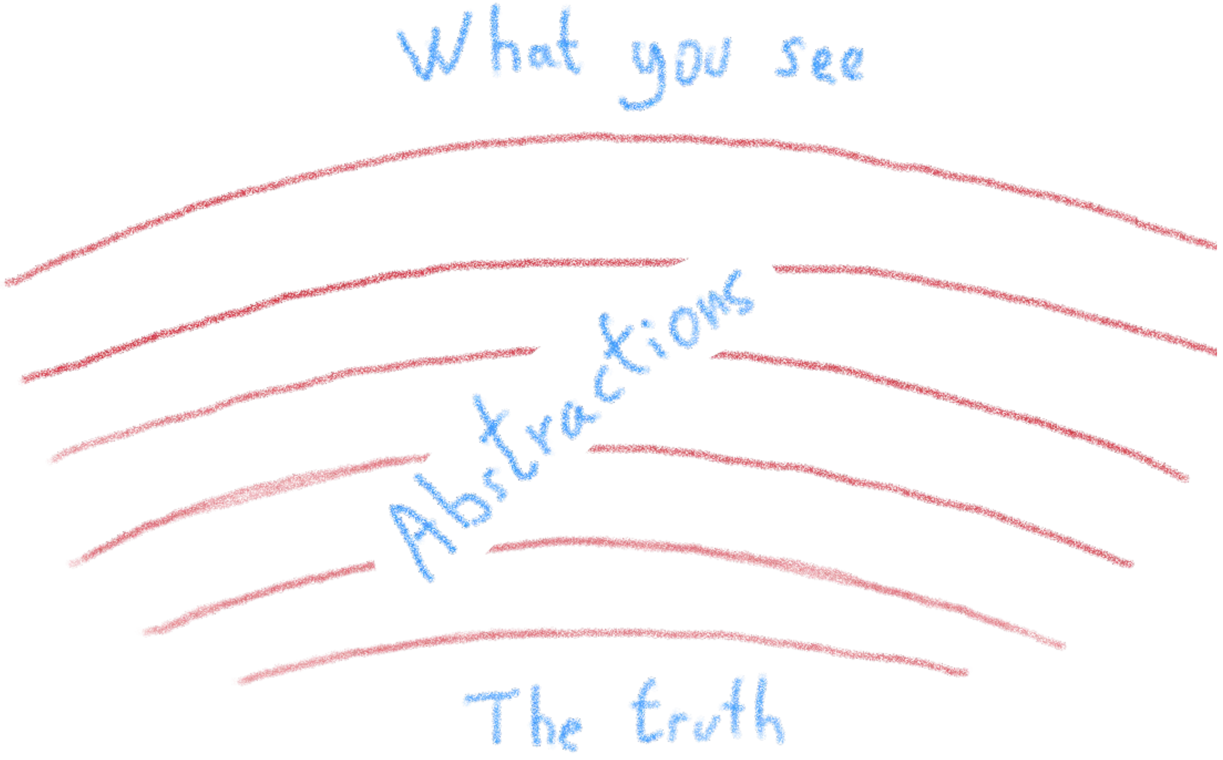


For both developers and devops

The journey from dev to production

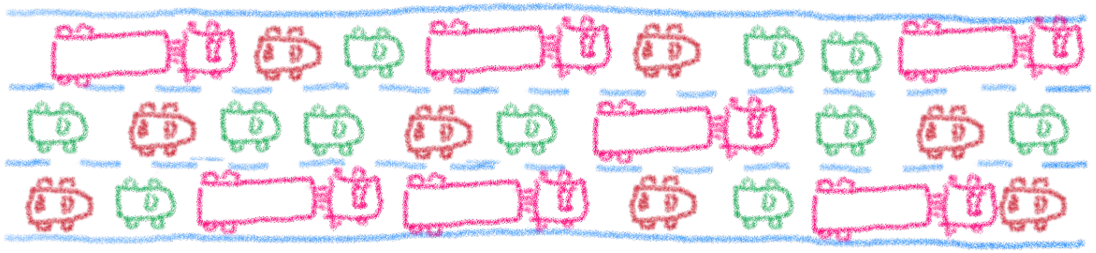

Tutorials & talks stop here
Deployed a real Kubernetes cluster
It is a trap!
Deployed a production-ready cluster

It's a complex technology



Lots of abstraction layers

Kubernetes networking is complex...



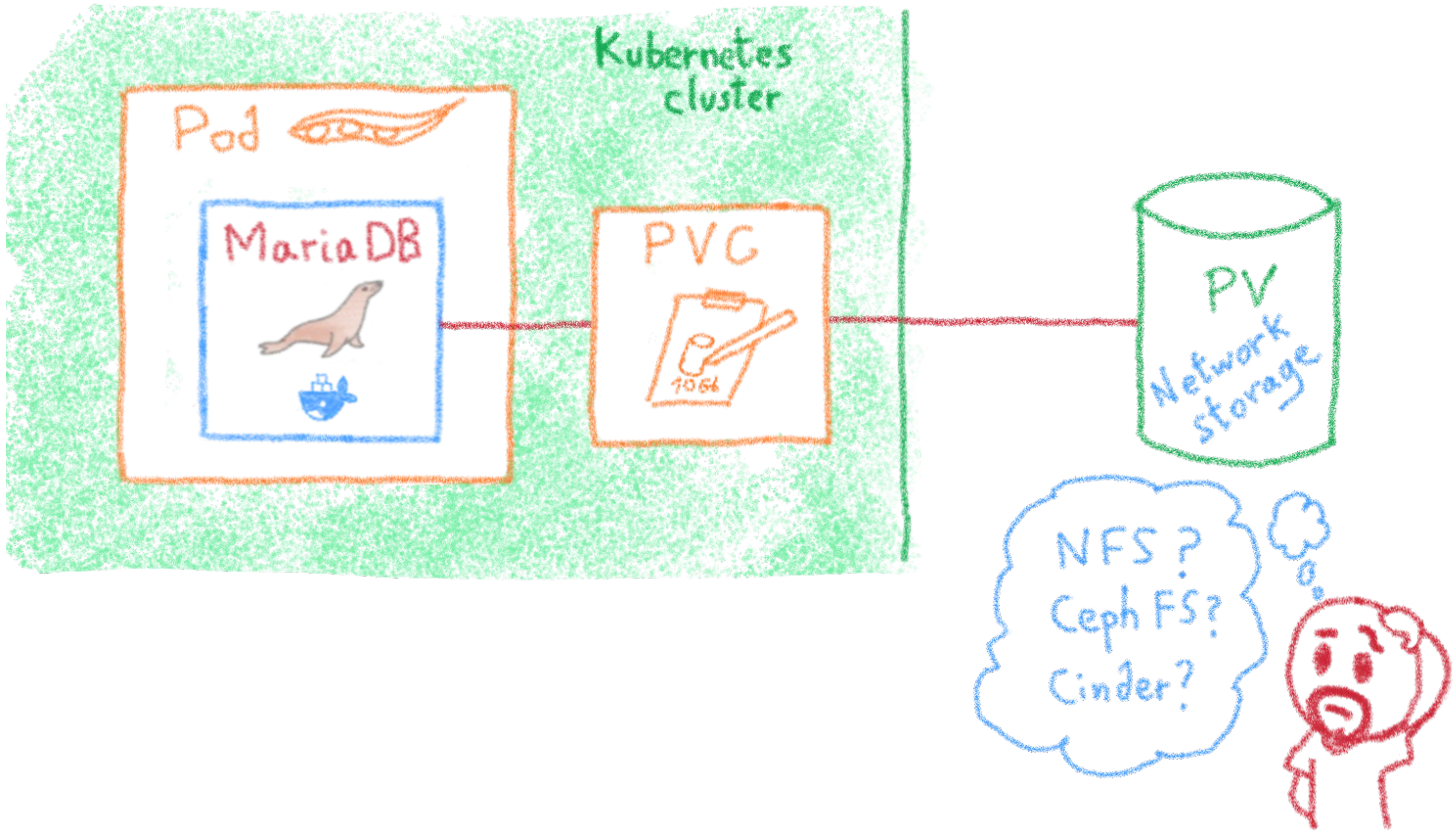
Network plugins (Flannel, Calico, Weave...)

- IPAM
- routing
- iptables
- crossnode networking

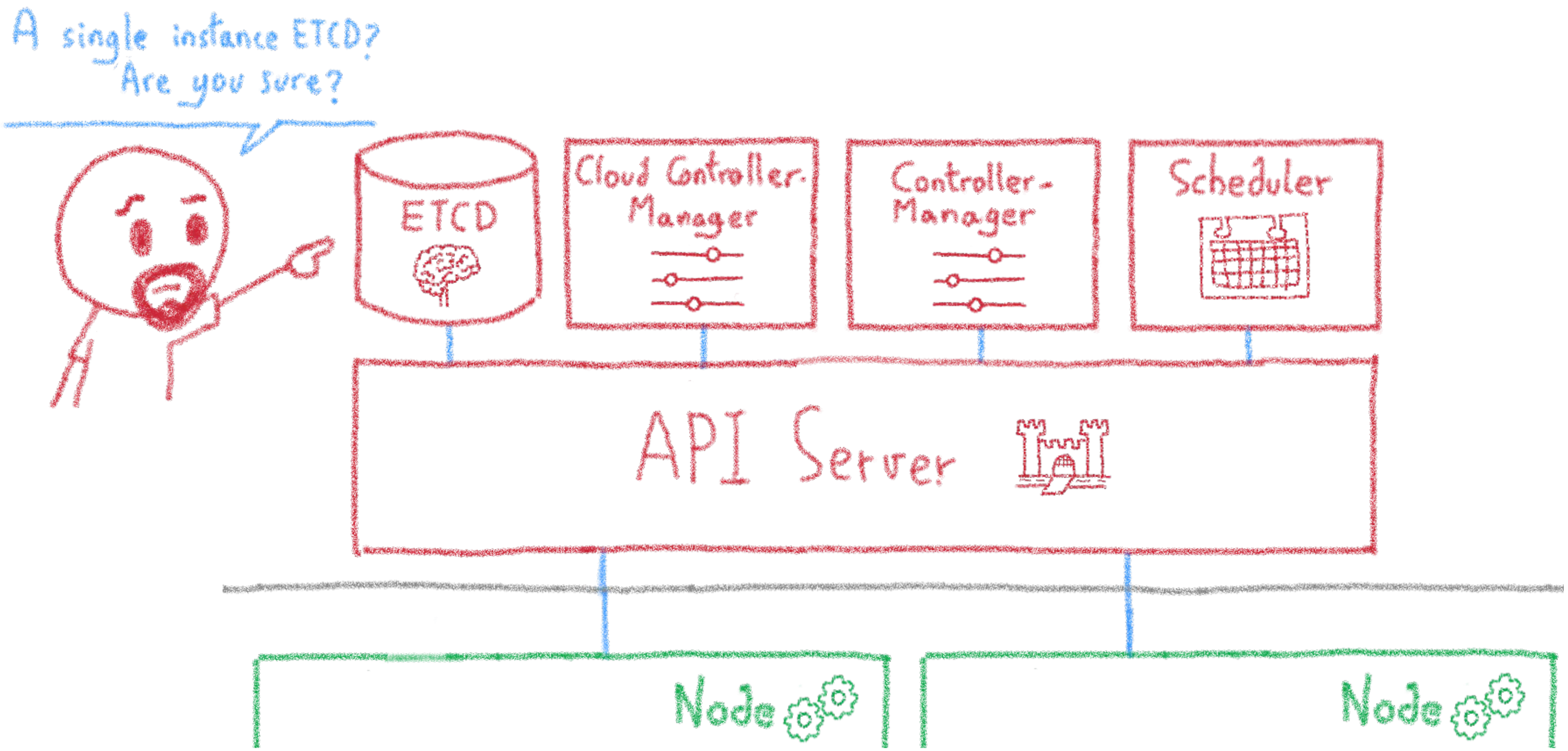
Cluster IP, NodePort, Ingress

Service Meshes, Istio

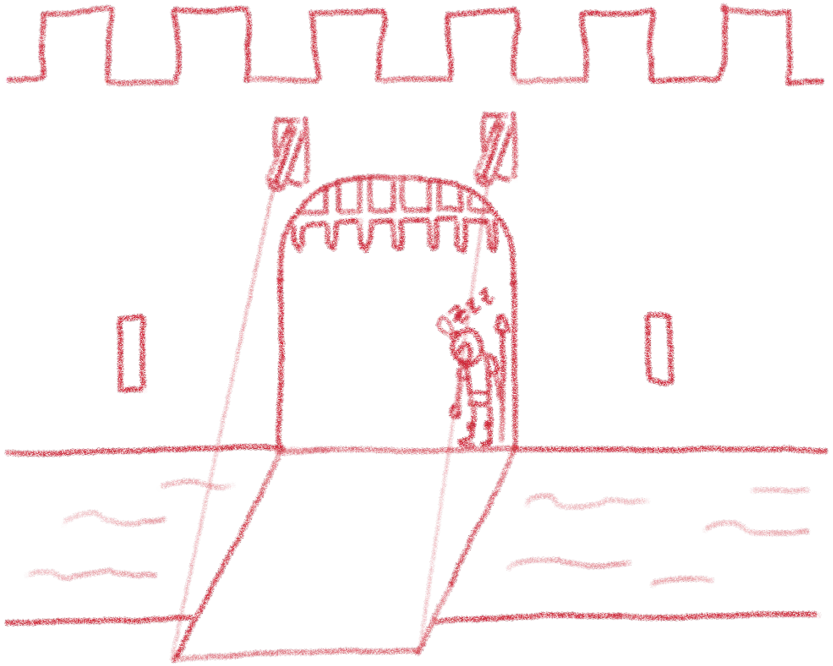
The storage dilemma



The ETCD vulnerability



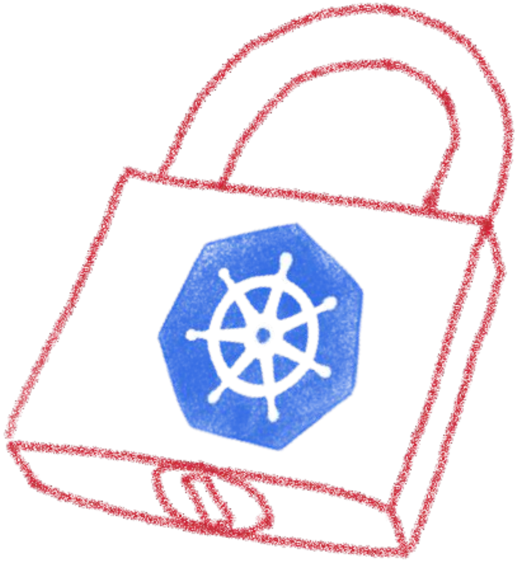
Kubernetes is insecure by design*



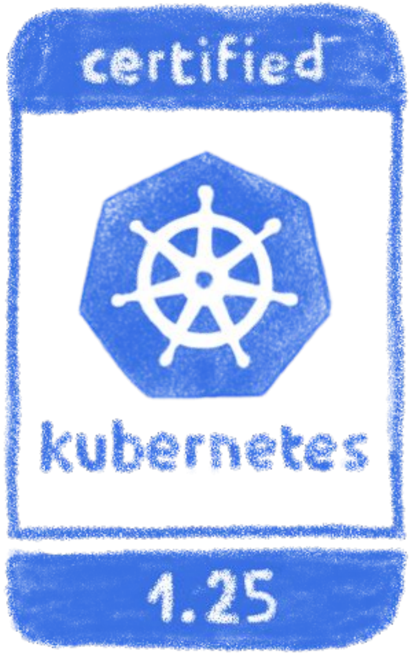
It's a feature, not a bug.
Up to K8s admin to secure it according to needs

Not everybody has the same security needs

Kubernetes allows to enforce security practices as needed



Always keep up to date



Both Kubernetes and plugins

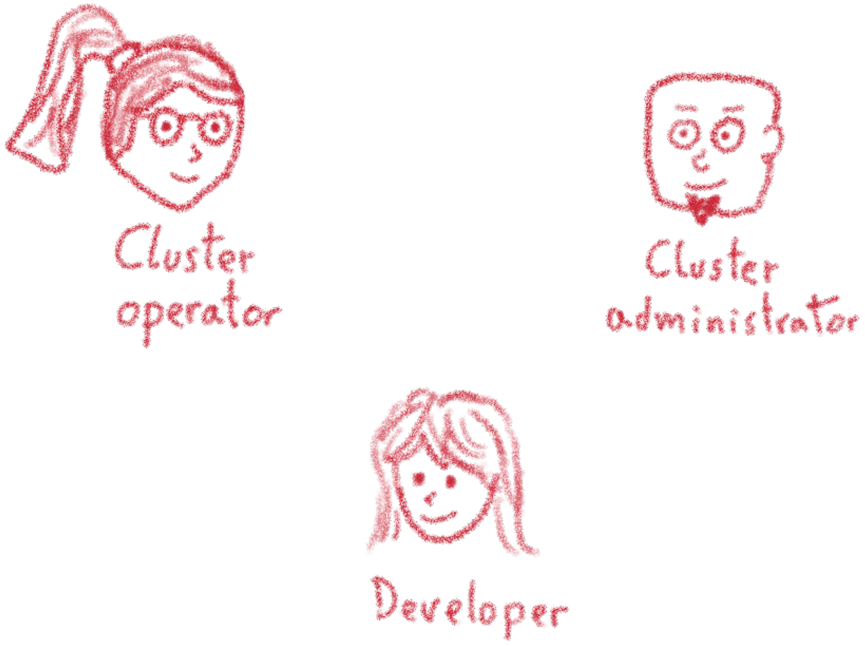
And remember,
even the best can get hacked

One of Tesla's cluster got hacked via an unprotected K8s API endpoint, and was used to mine cryptocurrency...

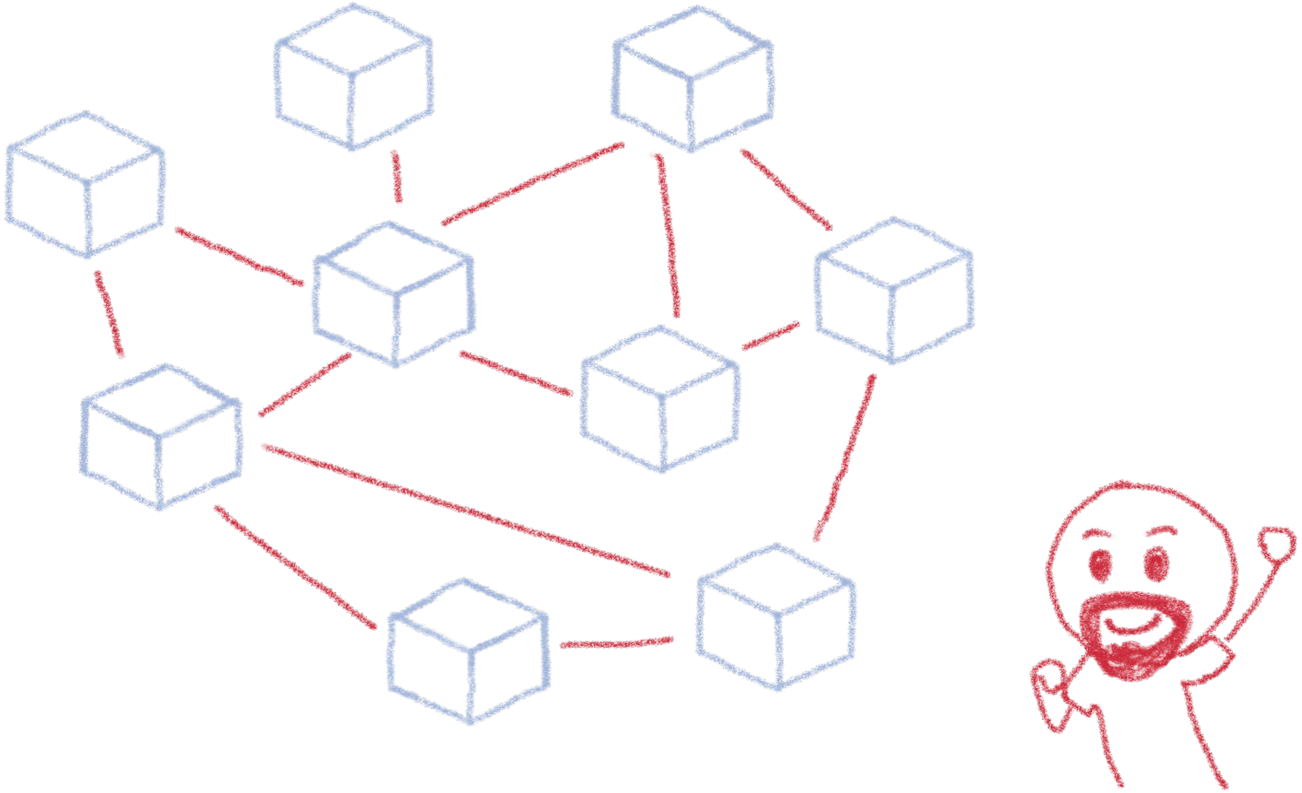
Remain attentive, don't get too confident

A managed Kubernetes

Because your company job is to use Kubernetes, not to operate it!

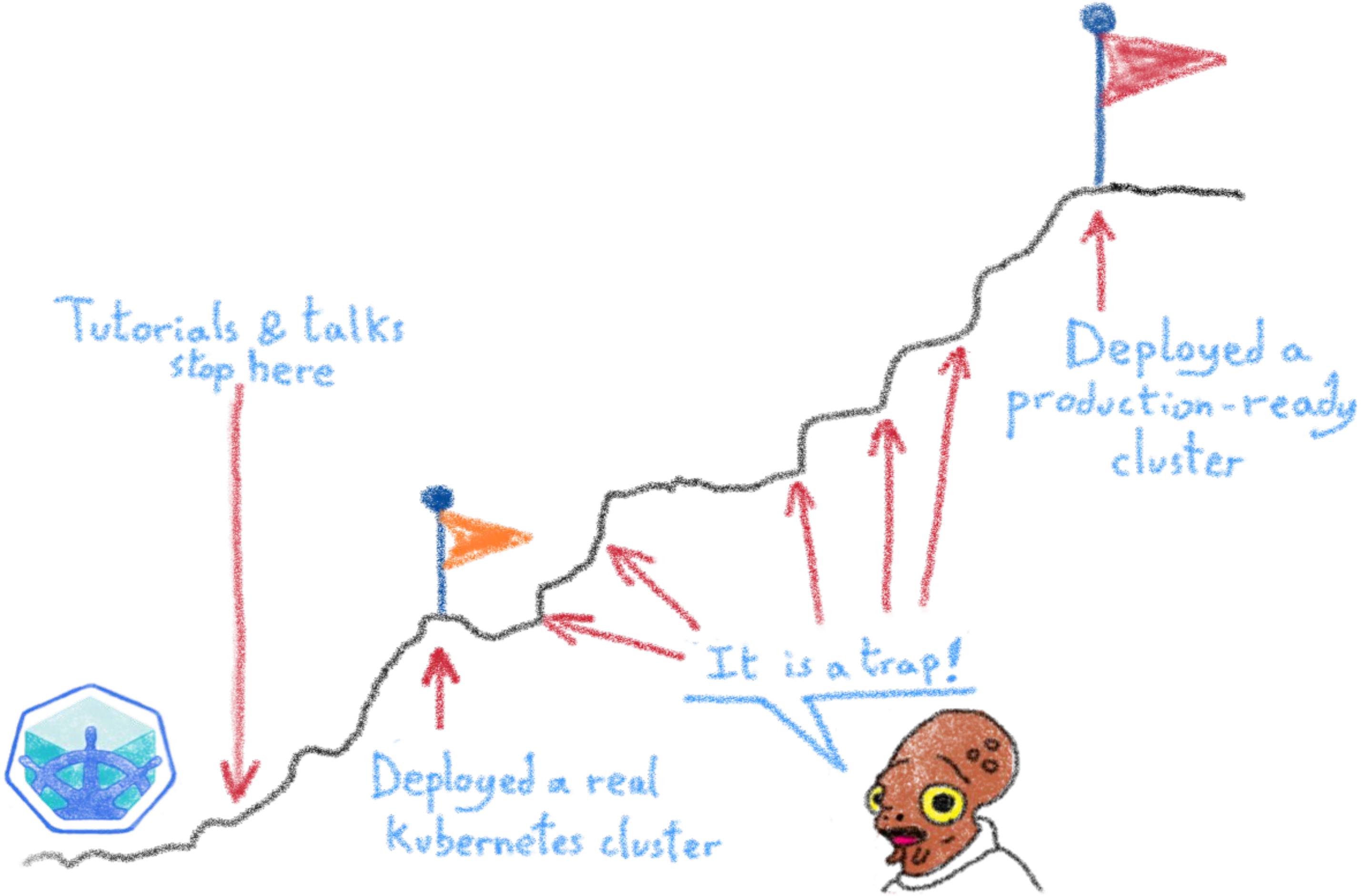


Kubernetes is powerful



It can make Developers' and DevOps' lives easier

But there is a price: operating it

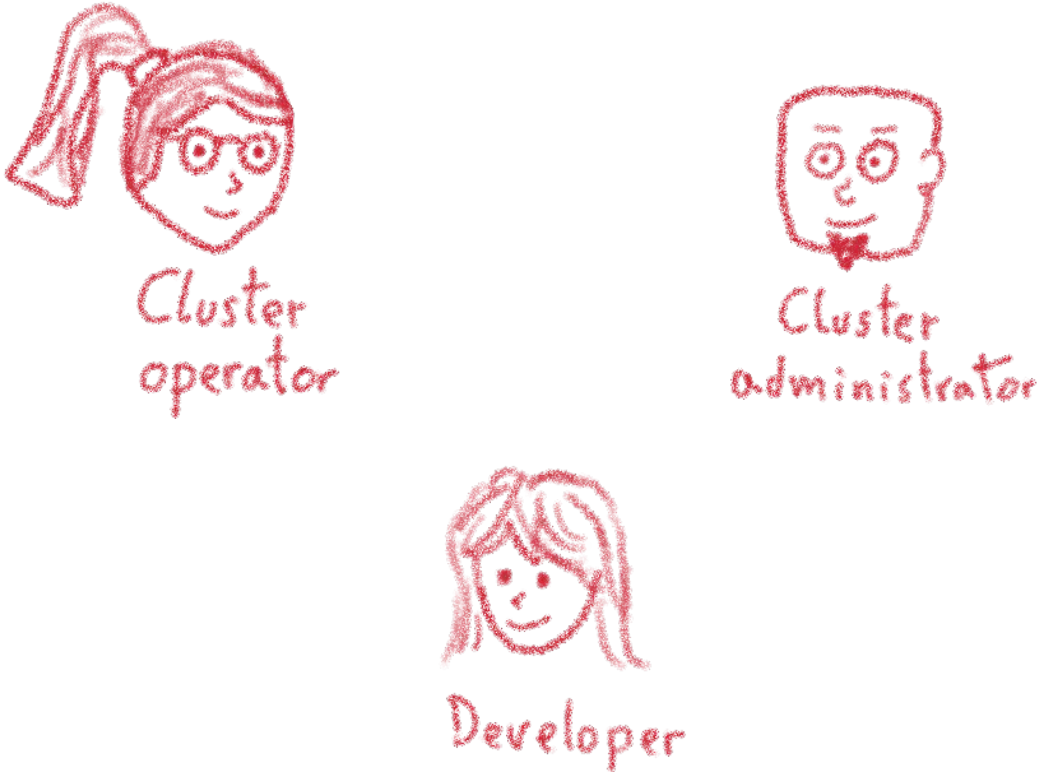

Tutorials & talks stop here
Deployed a real Kubernetes cluster
It is a trap!
Deployed a production-ready cluster

Lot of things to think about

We have seen some of them

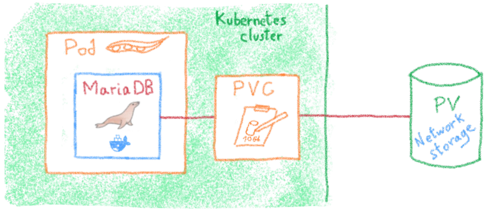
- Security
- Deployment
- Monitoring
- Backups

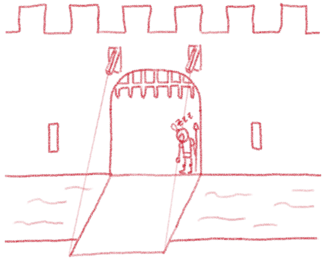
Different roles

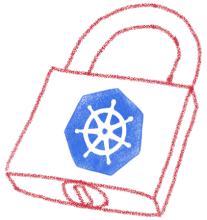


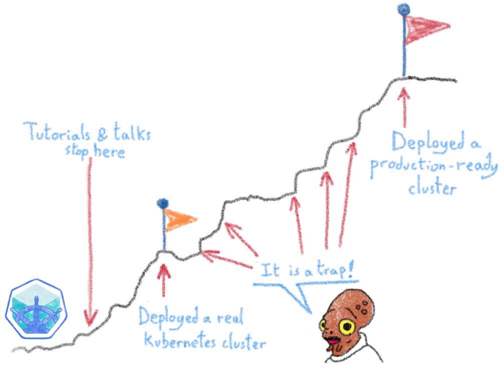
Each role asks for very different knowledge and skill sets

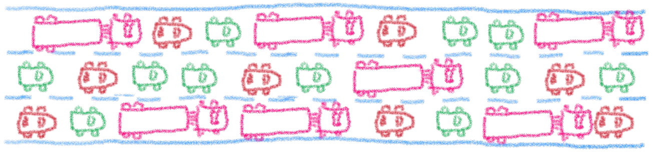
Operating a Kubernetes cluster is hard





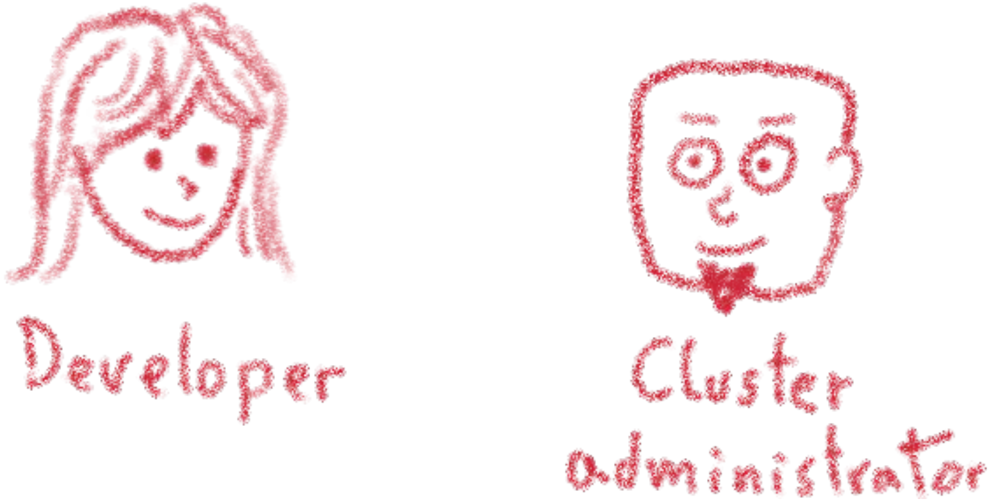



Tutorials & talks stop here
Deployed a production-ready cluster
It is a trap!
Deployed a real Kubernetes cluster



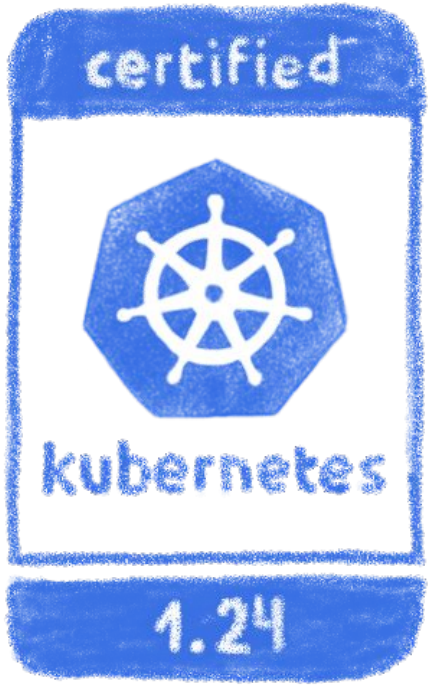
But we have a good news...

Most companies don't need to do it!



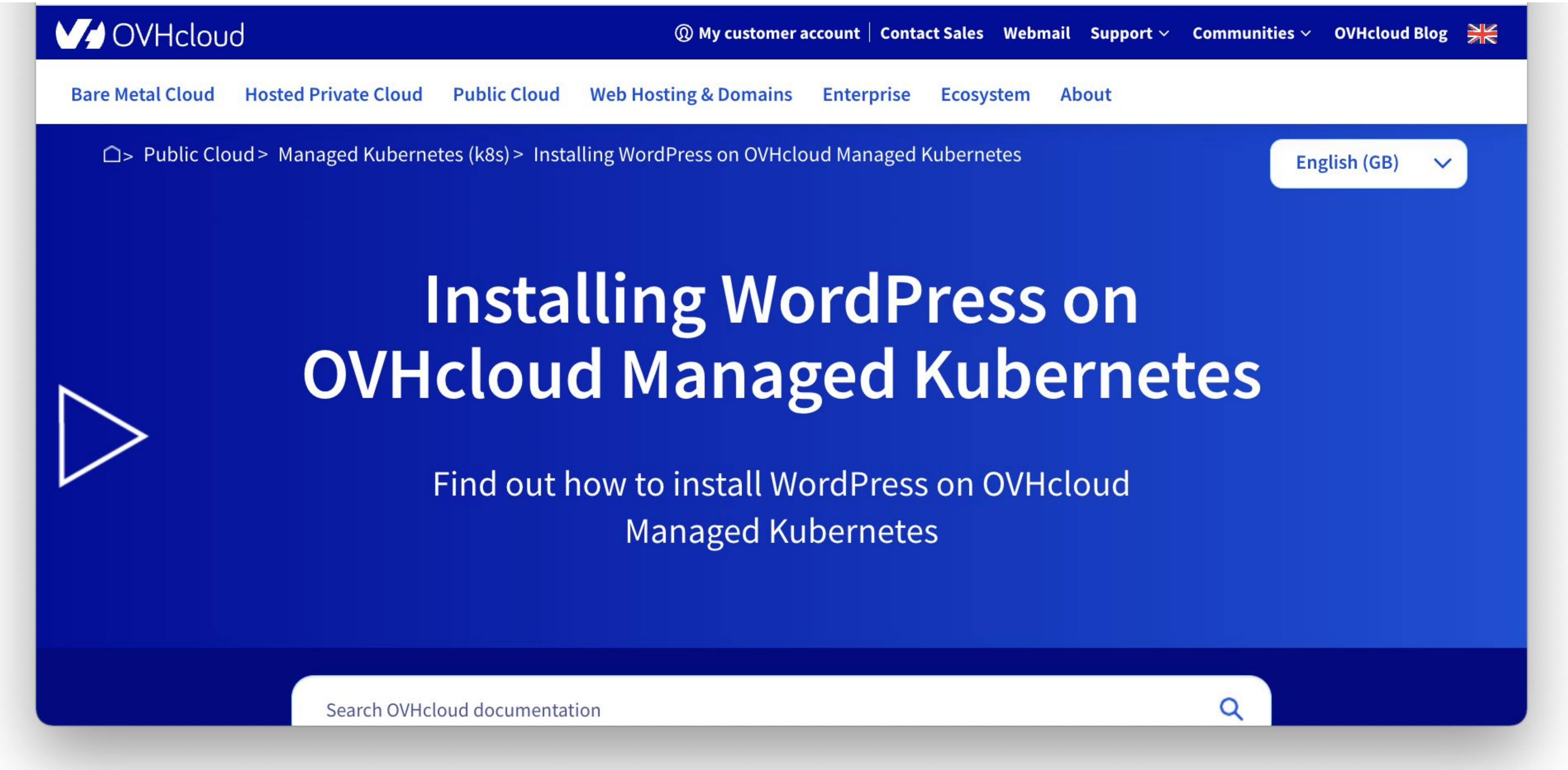
As they don't build and rack their own servers!

If you don't need to build it, choose a certified managed solution



You get the cluster, the operator get the problems

Demo: A complete app - Wordpress



OVHcloud

My customer account | Contact Sales Webmail Support Communities OVHcloud Blog

Bare Metal Cloud Hosted Private Cloud Public Cloud Web Hosting & Domains Enterprise Ecosystem About

> Public Cloud > Managed Kubernetes (k8s) > Installing WordPress on OVHcloud Managed Kubernetes

English (GB)

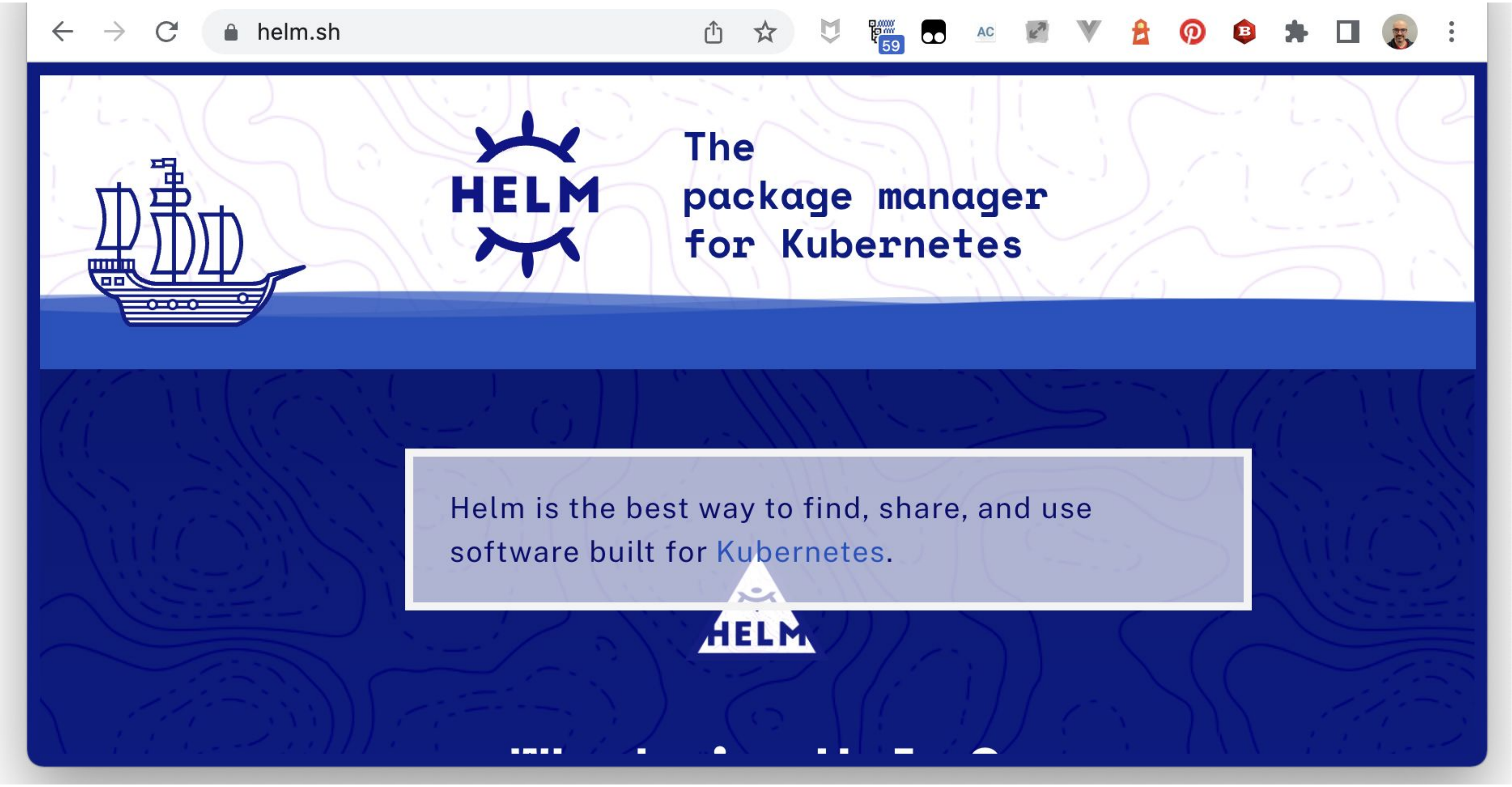
Installing WordPress on OVHcloud Managed Kubernetes

Find out how to install WordPress on OVHcloud Managed Kubernetes

Search OVHcloud documentation

https://docs.ovh.com/gb/en/kubernetes/installing-wordpress/

Needed tools: helm



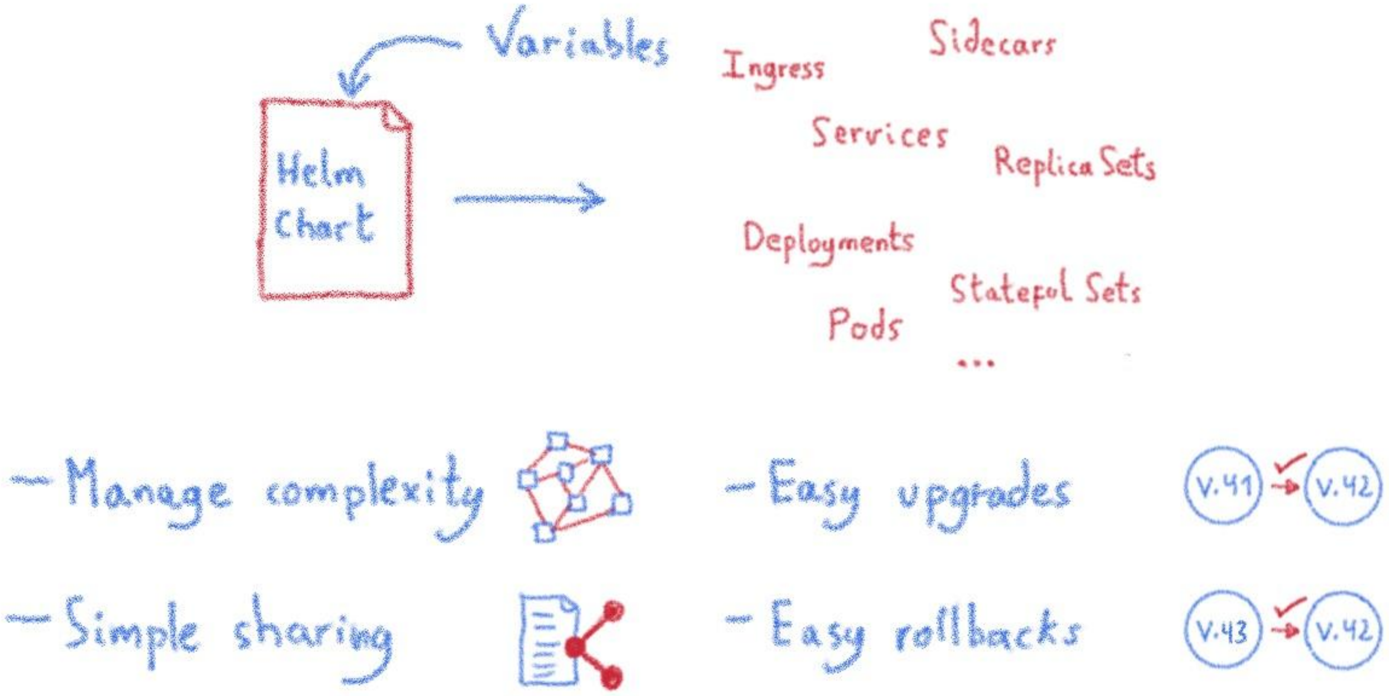
helm.sh

The package manager for Kubernetes

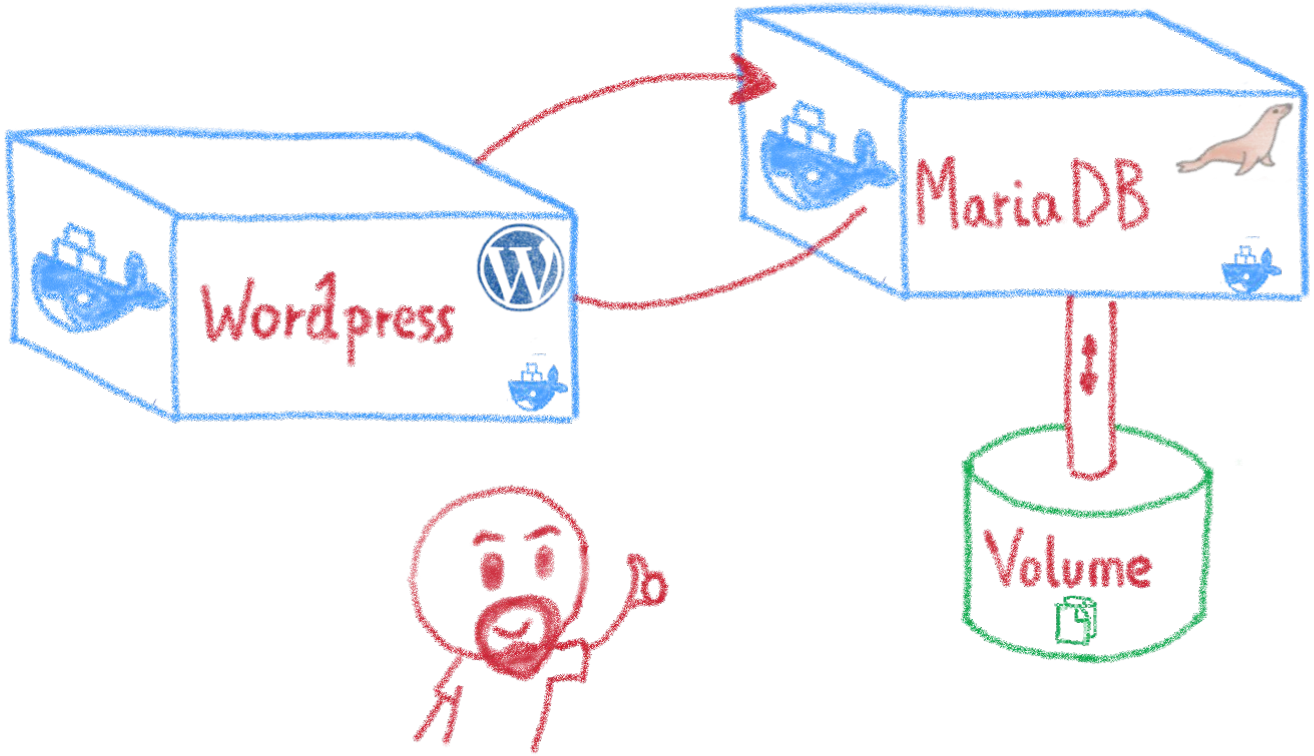
Helm is the best way to find, share, and use software built for Kubernetes.

https://helm.sh/

Helm: a package manager for K8s

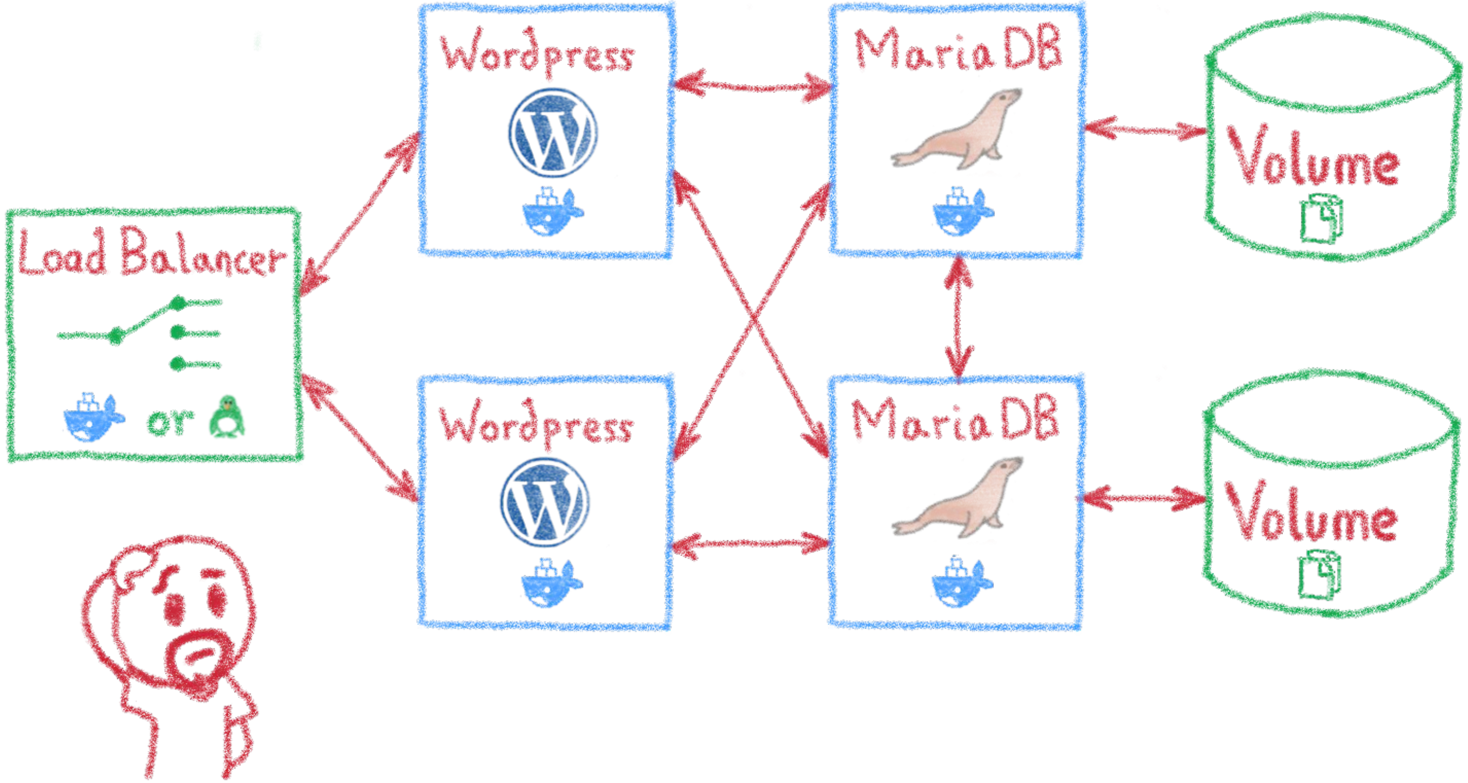


Wordpress is easy...



Two pods and a persistent volume

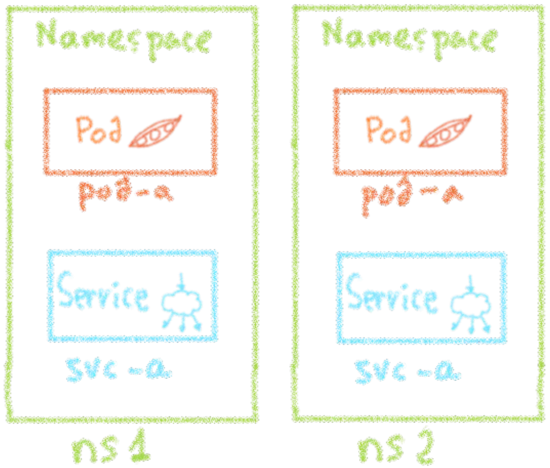
Yet is a complete app



Specially when deployed in production context

Namespaces

Logical isolation



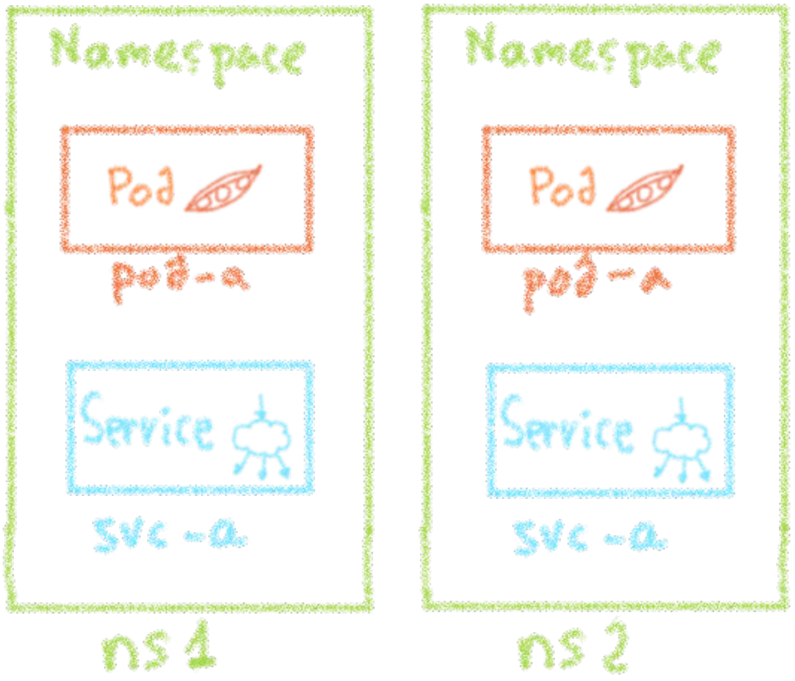
Namespaces

Namespaces : logical isolation

You give then a meaning

Isolation by project?

By team? By environment?



Inside a namespace

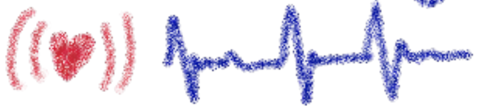
resource names are unique

Initial namespaces

- default – namespace by default

- kube-system – objects created by K8s system
- kube-public – readable by all clients, without auth
- kube-node-lease – lease objects for each node

lease objects allows the kubelet to send heartbeat messages to the control plane



Working with namespaces

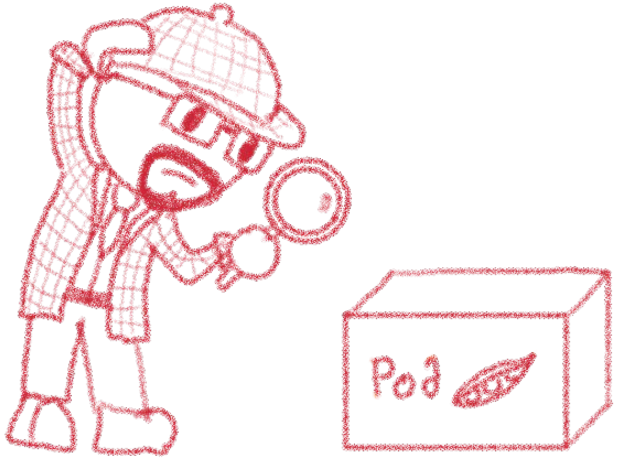
```
$ kubectl create namespace my-namespace
namespace/my-namespace created

$ kubectl get namespaces
NAME               STATUS   AGE
default            Active   45d
kube-node-lease    Active   45d
kube-public        Active   45d
kube-system        Active   45d
my-namespace       Active   7s

$ kubectl get pods --all-namespaces
NAMESPACE     NAME                                        READY   STATUS    RESTARTS   AGE
kube-system   calico-kube-controllers-6b5885747b-m79ng   1/1     Running   0          6m58s
kube-system   canal-22dj9                                2/2     Running   0          7m
kube-system   canal-4l4mv                                2/2     Running   0          6m39s
kube-system   canal-6rdxv                                2/2     Running   0          7m19s
kube-system   coredns-9f744c589-64spf                    1/1     Running   0          42s
kube-system   coredns-9f744c589-tl26z                    1/1     Running   0          6m25s
[...]
```

Working with namespaces

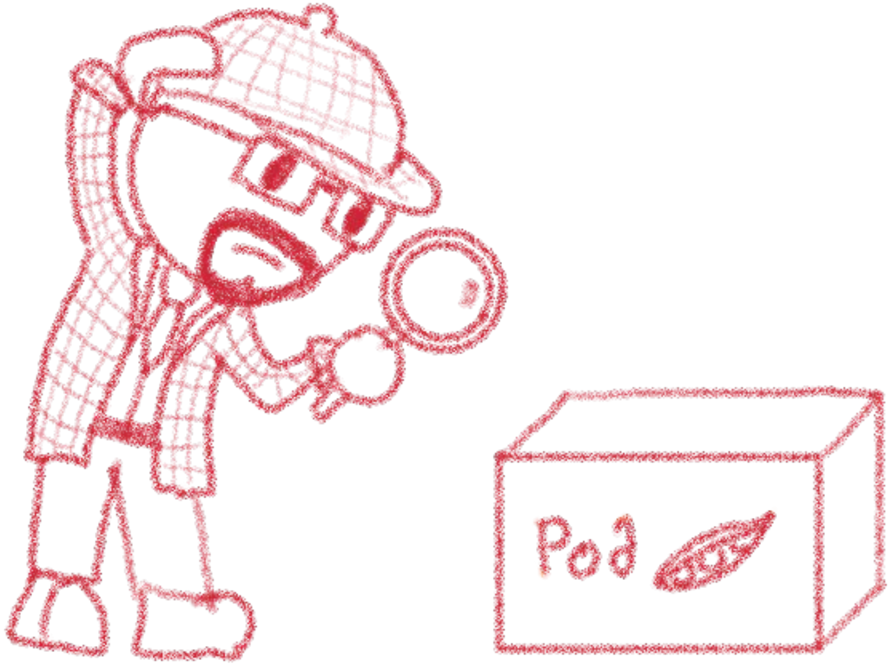
```
$ kubectl apply -f hello.yml -n my-namespace
service/hello-world-service created
deployment.apps/hello-world-deployment created

$ kubectl get pods --all-namespaces
NAMESPACE     NAME                                       READY   STATUS    RESTARTS   AGE
kube-system   calico-kube-controllers-6b5885747b-m79ng   1/1     Running   0          6m58s
kube-system   canal-22dj9                                2/2     Running   0          7m
kube-system   canal-4l4mv                                2/2     Running   0          6m39s
kube-system   canal-6rdxv                                2/2     Running   0          7m19s
kube-system   coredns-9f744c589-64spf                    1/1     Running   0          42s
kube-system   coredns-9f744c589-tl26z                    1/1     Running   0          6m25s
[...]
kube-system    wormhole-vx6sn                             1/1     Running   0          9m53s
my-namespace   hello-world-deployment-bc4fd6b9-5mtk4      1/1     Running   0          37s

$ kubectl delete namespace my-namespace
namespace "my-namespace" deleted
```

Executing commands

kubectl exec



Pods are black boxes



How can we debug them?

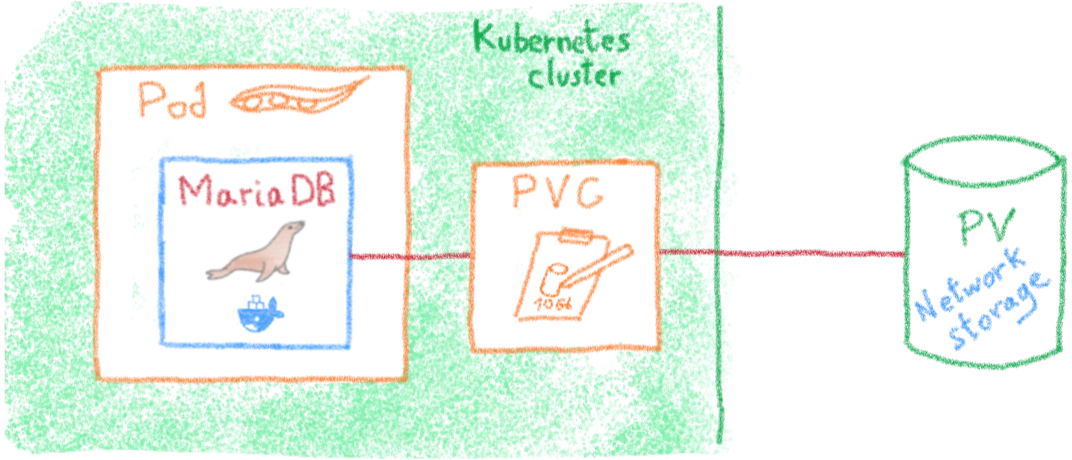
Interactively execute commands

```
$  kubectl exec hello-world-deployment-bc4fd6b9-5sgls -c hello-world -it -- sh
/ # ls
bin      dev      etc      home     lib   mnt      proc     root     run      sbin     srv      sys      tmp      usr      var
/ #
```

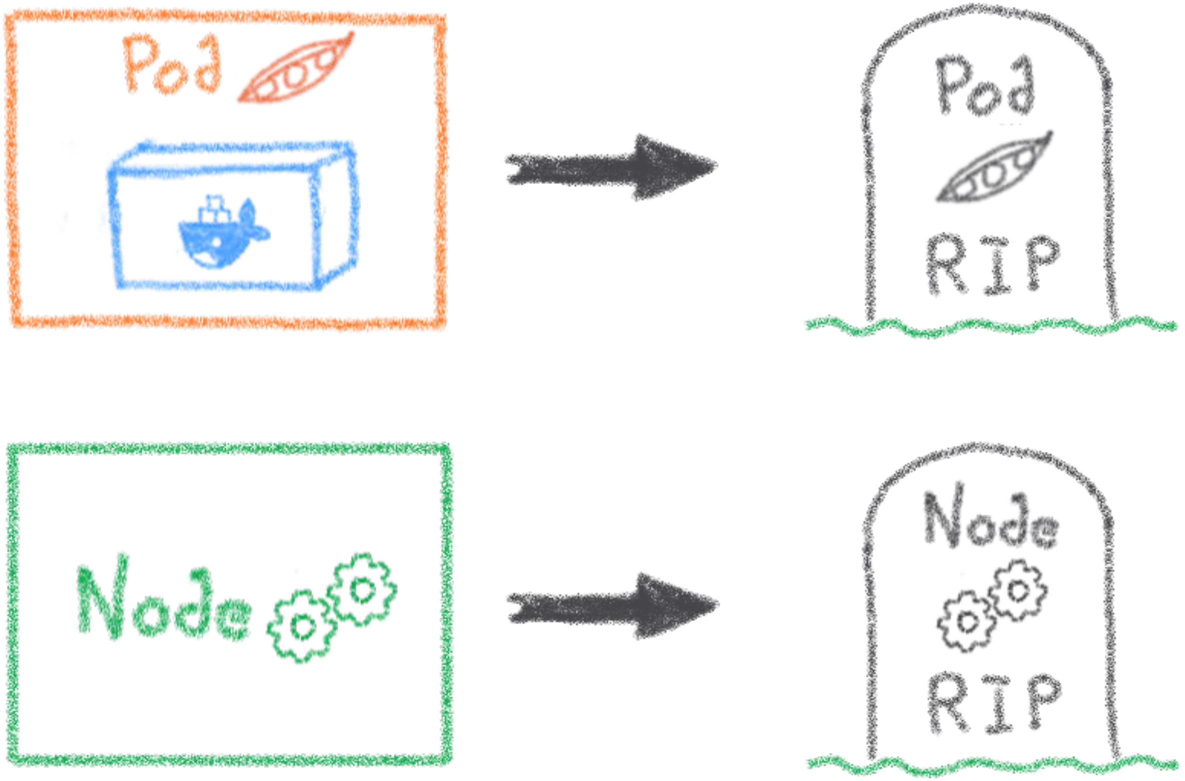
Execute commands in a container inside a pod

Persistent Volumes

How to store persistent data in K8s

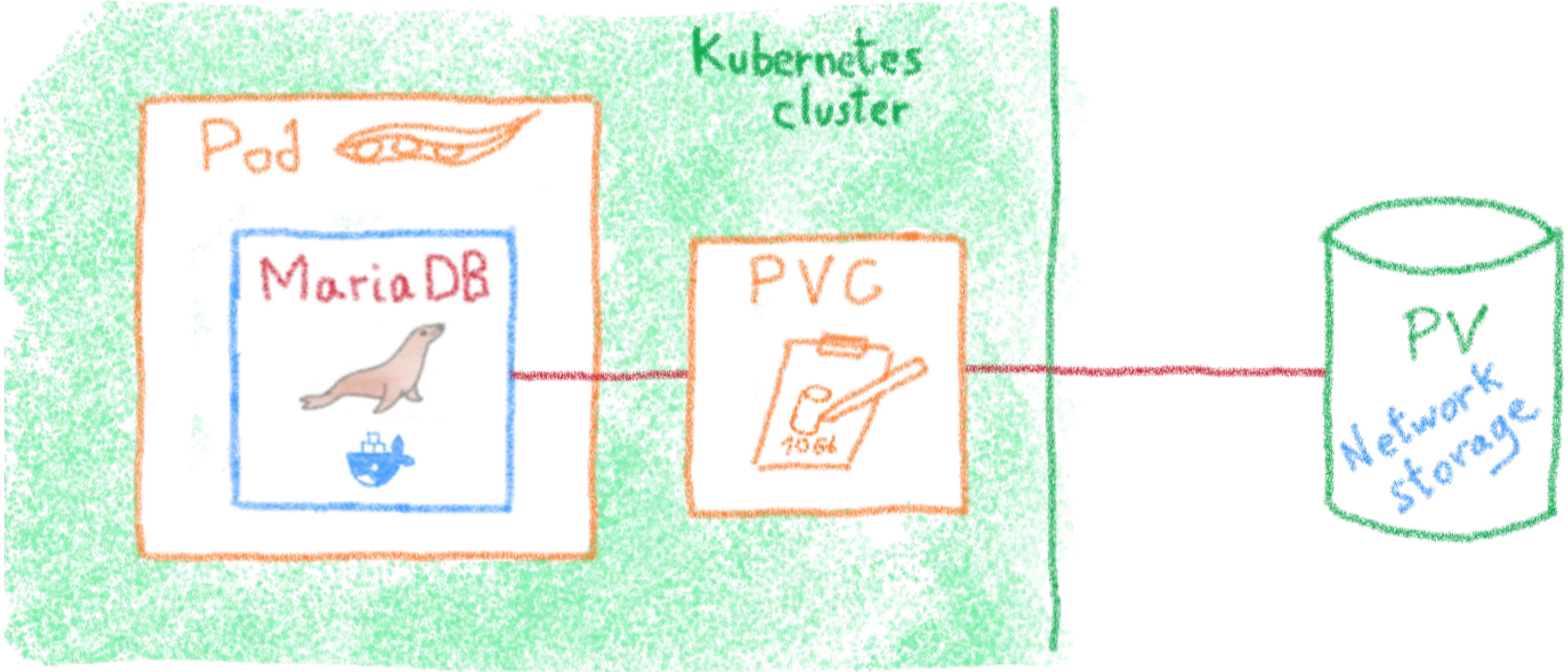


Local storage is a bad idea

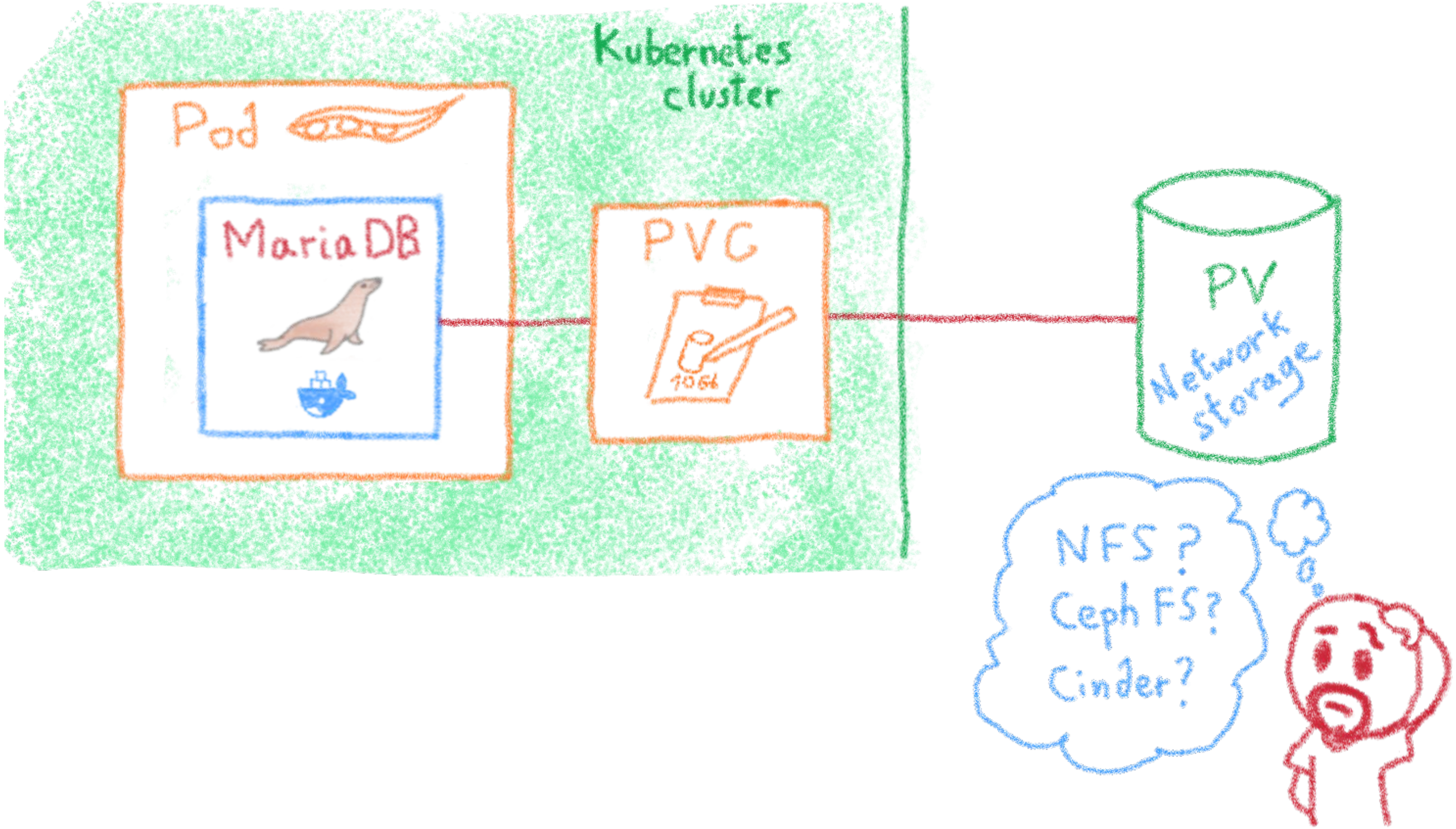


Pods & Nodes are transient, they can and will die

Persistent Volumes



The storage dilemma



Resource management

Request and limits

Pod
Limit
Memory

Pod
Limit
CPU

Resource management

Resource management

- Requests: how many resources it needs
- Limits: how many resources it can use

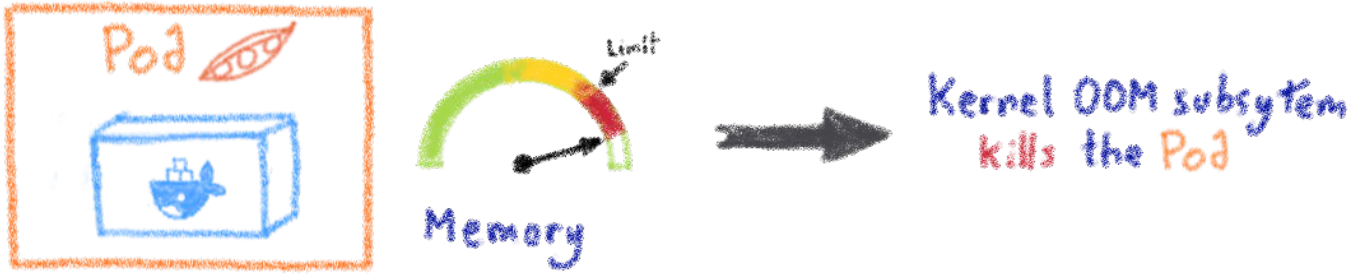
Resources types and units

- Memory: $1\,T = 1000\,G = 10^6\,M = 10^9\,K$
  $1\,Ti = 2^{10}\,Gi = 2^{20}\,Mi = 2^{30}\,Ki$
- CPU : 1 vCore = 1 CPU = 1000 mCPU

```
apiVersion: v1
kind: Pod
metadata:
 name: frontend
spec:
 containers:
 - name: app
   image: images.my-company.example/app
   resources:
     requests:
       memory: "64Mi"
       cpu: "250m"
     limits:
       memory: "128Mi"
       cpu: "500m"
```

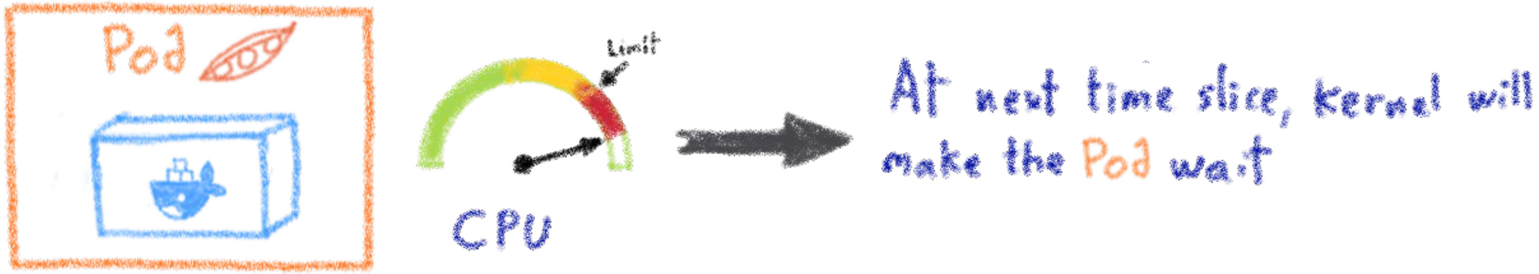
What if a pod uses too many resources?

Memory: if Pod tries to over allocate, it generates an OOM



Kernel OOM subsytem kills the Pod

CPU: if Pod tries to over use, kernel waits before allowing it to continue



At next time slice, kernel will make the Pod wait

CPU is compressible, memory is incompressible

Resource quota

Resource quota

limits.cpu requests.cpu
limits.memory requests.memory
limits.storage requests.storage

A global quota per namespace

```yaml
kind: ResourceQuota
metadata:
 name: compute-resources
spec:
 hard:
   requests.cpu: "1"
   requests.memory: 1Gi
   limits.cpu: "2"
   limits.memory: 2Gi
   requests.nvidia.com/gpu: 4
```

Limit the total sum of compute resources that can be requested in a given namespace

Limit range

Limit ranges

- Min and max resources
- Default request/limits

per pod in a namespace

```yaml
apiVersion: v1
kind: LimitRange
metadata:
name: cpu-resource-constraint
spec:
limits:
- default: # this section defines default limits
    cpu: 500m
  defaultRequest: # this section defines default requests
    cpu: 500m
  max: # max and min define the limit range
    cpu: "1"
  min:
    cpu: 100m
  type: Container
```

Default, minimum and maximum resources usage
per pod in a namespace

Verifying resource usage

```
% kubectl top pods
NAME                                  CPU(cores)   MEMORY(bytes)
hello-world-deployment-bc4fd6b9-dgspd   3m           2Mi
hello-world-deployment-bc4fd6b9-f85mf   3m           2Mi
hello-world-deployment-bc4fd6b9-hh7xs   4m           2Mi
hello-world-deployment-bc4fd6b9-lz494   5m           2Mi

% kubectl top pods --containers
POD                                   NAME          CPU(cores)   MEMORY(bytes)
hello-world-deployment-bc4fd6b9-dgspd   hello-world   0m           2Mi
hello-world-deployment-bc4fd6b9-f85mf   hello-world   1m           2Mi
hello-world-deployment-bc4fd6b9-hh7xs   hello-world   1m           2Mi
hello-world-deployment-bc4fd6b9-lz494   hello-world   0m           2Mi

% kubectl top nodes
NAME                                      CPU(cores)   CPU%   MEMORY(bytes)
MEMORY%
nodepool-ce18c6cd-1291-4a6e-83-node-5c283f   110m         5%     1214Mi          23%
nodepool-ce18c6cd-1291-4a6e-83-node-85b011   104m         5%     1576Mi          30%
nodepool-ce18c6cd-1291-4a6e-83-node-c3cfcf   121m         6%     1142Mi          22%
```

Health probes

Telling Kubernetes that the pod is alive and healthy

Liveness probe

```yaml
apiVersion: v1
kind: Pod
metadata:
 labels:
   test: liveness
 name: liveness-exec
spec:
 containers:
 - name: liveness
   image: registry.k8s.io/busybox
   args:
   - /bin/sh
   - -c
   - touch /tmp/healthy; sleep 30; rm -f /tmp/healthy; sleep 600
   livenessProbe:
     exec:
       command:
       - cat
       - /tmp/healthy
     initialDelaySeconds: 5
     periodSeconds: 5
```

Liveness probe – Telling the cluster that the pod is alive

Readiness probe

```yaml
apiVersion: v1
kind: Pod
metadata:
 labels:
   test: liveness
 name: liveness-exec
spec:
 containers:
 - name: liveness
   image: registry.k8s.io/busybox
   args:
   - /bin/sh
   - -c
   - touch /tmp/healthy; sleep 30; rm -f /tmp/healthy; sleep 600
   readinessProbe:
     exec:
       command:
       - cat
       - /tmp/healthy
     initialDelaySeconds: 5
     periodSeconds: 5
```

Readiness probe – Telling the cluster that the pod is ready

Startup probe

```yaml
apiVersion: v1
kind: Pod
metadata:
 labels:
   test: liveness
 name: liveness-exec
spec:
 containers:
 - name: liveness
   image: registry.k8s.io/busybox
   livenessProbe:
     exec:
       command:
       - cat
       - /tmp/healthy
     initialDelaySeconds: 5
     periodSeconds: 5
   startupProbe:
     exec:
       command:
       - cat
       - /tmp/healthy
     periodSeconds: 5
     failureThreshold: 24
```

Startup probe – Hold off other probes until the pod has started

Defining configuration

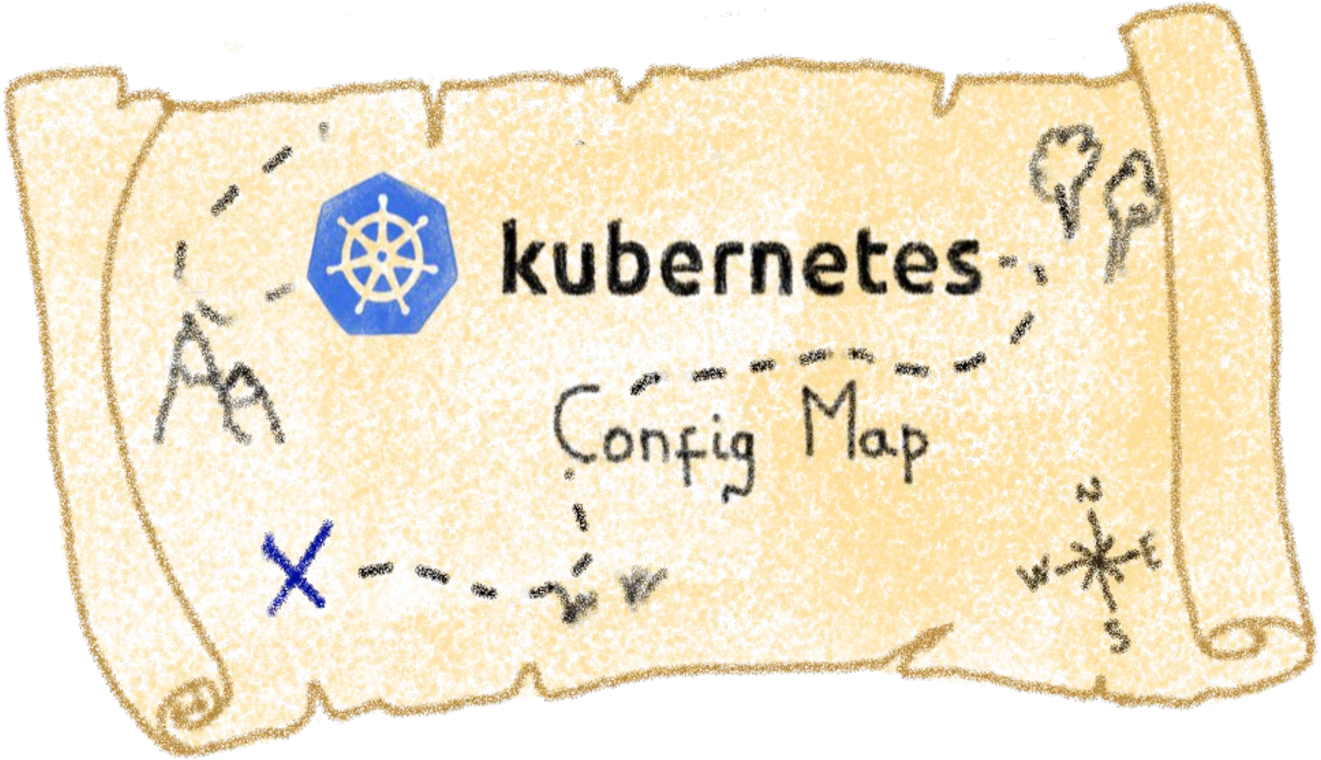
Config maps & secrets

Config files are a bad practice

Configuration shoud be decoupled from container images to make apps portable

But how I give the env specific configuration to the app without config files?

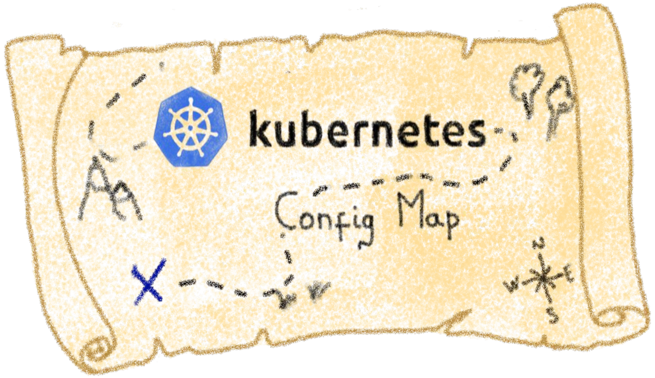
Config maps



Storing configuration for other objects to use

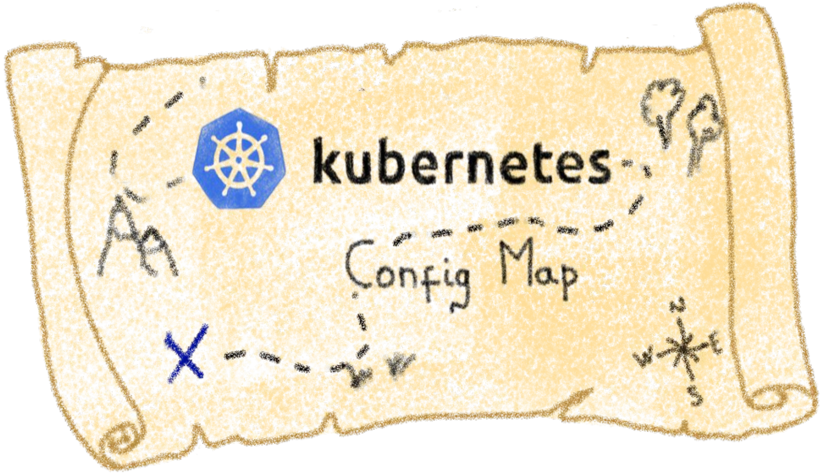
Creating a Config Map

```
# Create a new configmap named my-config with keys for each file in folder bar
$ kubectl create configmap my-config-1 --from-file=./config/bar
configmap/my-config created

# Create a new configmap named my-config with specified keys instead of names on disk
$ kubectl create configmap my-config-2 --from-file=ssh-privatekey=~/.ssh/id_rsa
--from-file=ssh-publickey=~/.ssh/id_rsa.pub
configmap/my-config created

# Create a new configMap named my-config with key1=config1 and key2=config2
$ kubectl create configmap my-config-3 --from-literal=key1=config1 --from-literal=key2=config2
configmap/my-config created
```



Describing a Config Map

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
 name: game-demo
data:
 # property-like keys; each key maps to a simple value
 player_initial_lives: "3"
 ui_properties_file_name: "user-interface.properties"

 # file-like keys
 game.properties: |
   enemy.types=aliens,monsters
   player.maximum-lives=5
 user-interface.properties: |
   color.good=purple
   color.bad=yellow
   allow.textmode=true
```



Using a Config Map in a Pod

1- Inside a container command and args

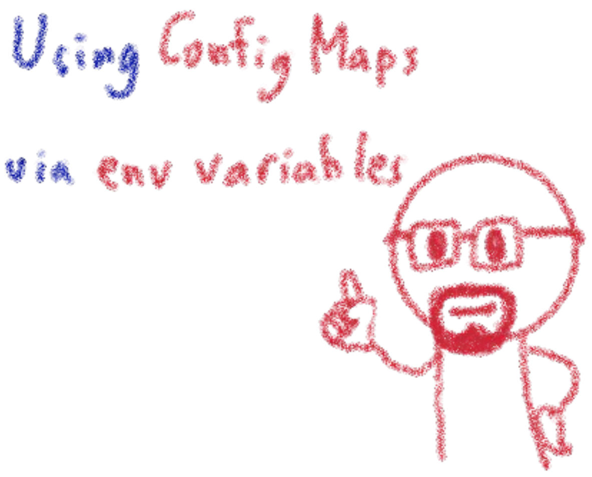
2- Container env variables

3- Mounting the ConfigMap as a read-only file

4- Using the K8s API from the container

Using a Config Map in a Pod

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: configmap-demo-pod
spec:
 containers:
   - name: demo
     image: alpine
     command: ["sleep", "3600"]
     env:
       # Define the environment variable
       - name: PLAYER_INITIAL_LIVES # Notice that the case is different here
                                    # from the key name in the ConfigMap.
         valueFrom:
           configMapKeyRef:
             name: game-demo           # The ConfigMap this value comes from.
             key: player_initial_lives # The key to fetch.
       - name: UI_PROPERTIES_FILE_NAME
         valueFrom:
           configMapKeyRef:
             name: game-demo
             key: ui_properties_file_name
```



Using a Config Map in a Pod

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: configmap-demo-pod
spec:
 containers:
   - name: demo
     image: alpine
     command: ["sleep", "3600"]
     volumeMounts:
     - name: config
       mountPath: "/config"
       readOnly: true
 volumes:
 # You set volumes at the Pod level, then mount them into containers inside that Pod
 - name: config
   configMap:
     # Provide the name of the ConfigMap you want to mount.
     name: game-demo
     # An array of keys from the ConfigMap to create as files
     items:
     - key: "game.properties"
       path: "game.properties"
     - key: "user-interface.properties"
       path: "user-interface.properties"
```

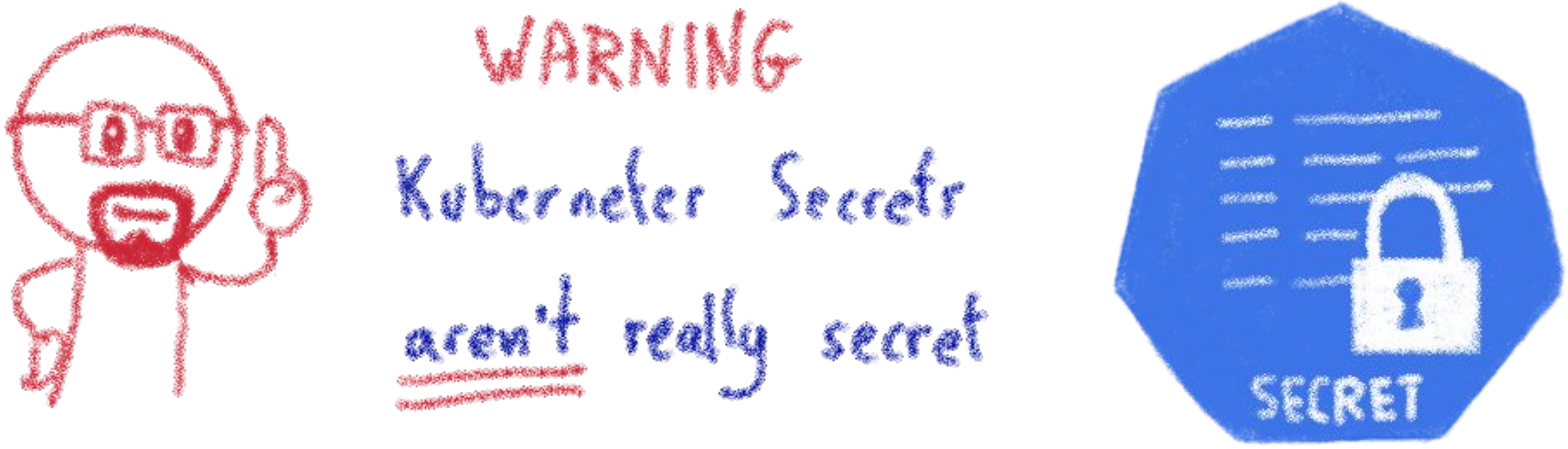
Mounting the Config Maps as read-only Files

Kubernetes secrets

Storing sensitive information inside the cluster

Encoded in Base64, decoded when attached to a pod

A warning on Kubernetes Secrets



No full encryption

All YAMLs and base64

Creating a Secret

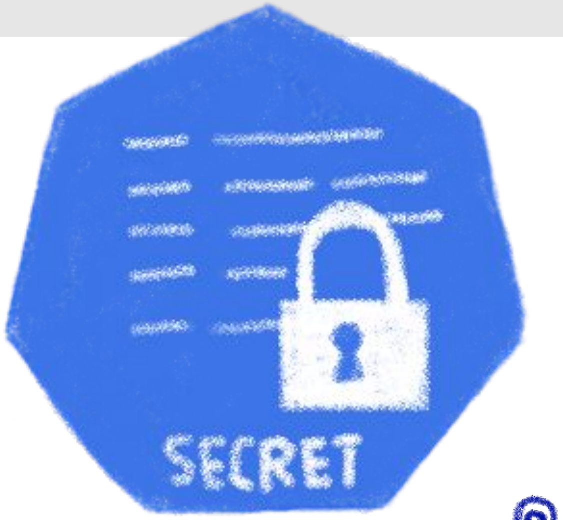
```
# Create a new Secret named db-user-pass with username=admin and password='S!B\*d$zDsb='
$ kubectl create secret generic db-user-pass \
    --from-literal=username=admin \
    --from-literal=password='S!B\*d$zDsb='

# Or store the credentials in files:
$ echo -n 'admin' > ./username.txt
$ echo -n 'S!B\*d$zDsb=' > ./password.txt

# And pass the file paths in the kubectl command:
$ kubectl create secret generic db-user-pass \
    --from-file=username=./username.txt \
    --from-file=password=./password.txt
```

Verifying a Secret

```
# Verify the Secret
$ kubectl get secrets
NAME            TYPE      DATA    AGE
db-user-pass    Opaque    2       3m34s

$ kubectl describe secret db-user-pass
Name:           db-user-pass
Namespace:      default
Labels:         <none>
Annotations:    <none>

Type:   Opaque

Data
====
password:   12 bytes
username:   5 bytes
```

Decoding a Secret

```
# View the contents of the Secret you created:
 $ kubectl get secret db-user-pass -o jsonpath='{.data}'
{"password":"UyFCXCpkJHpEc2I9","username":"YWRtaW4="}

# Decode the password data:
$ echo 'UyFCXCpkJHpEc2I9' | base64 --decode
S!B\*d$zDsb=

# In one step:
$ kubectl get secret db-user-pass -o jsonpath='{.data.password}' | base64 --decode
S!B\*d$zDsb=
```



Using a Secret in a Pod

1- Container env variables

2- Mounting the ConfigMap as a read-only file

3- By the kubelet when pulling the image

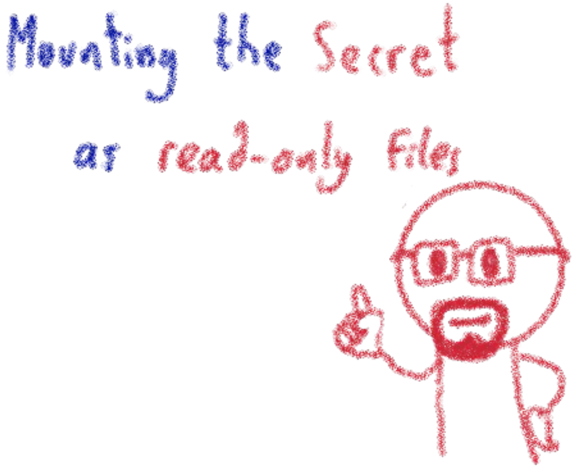
Using a Secret in a Pod

```
apiVersion: v1
kind: Pod
metadata:
 name: mypod
spec:
 containers:
 - name: mypod
   image: redis
   volumeMounts:
   - name: foo
     mountPath: "/etc/foo"
     readOnly: true
 volumes:
 - name: foo
   secret:
     secretName: mysecret
     optional: true
```

Mounting the Secret
as env variables

Using a Secret in a Pod

```
apiVersion: v1
kind: Pod
metadata:
 name: secret-demo-pod
spec:
 containers:
   - name: demo
     image: alpine
     command: ["sleep", "3600"]
     env:
       # Define the environment variable
       - name: PASSWORD
         valueFrom:
           SecretKeyRef:
             name: game-secret         # The Secret this value comes from.
             key: game-password        # The key to fetch.
```



Taints & Tolerations

And Affinity & Anti-affinity

Taints & Tolerations

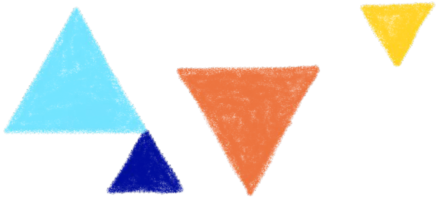
Taint

applied to a Kubernetes Node that signals the scheduler to avoid or not schedule certain Pods

Toleration

applied to a Pod definition and provides an exception to the taint

Using Taints & Tolerations



We define the Taint

```
# No pod will be able to schedule onto node-5c283f unless it has a matching toleration.
$ kubectl taint nodes node-5c283f  type=high-cpu:NoSchedule
node/node-5c283f tainted
```

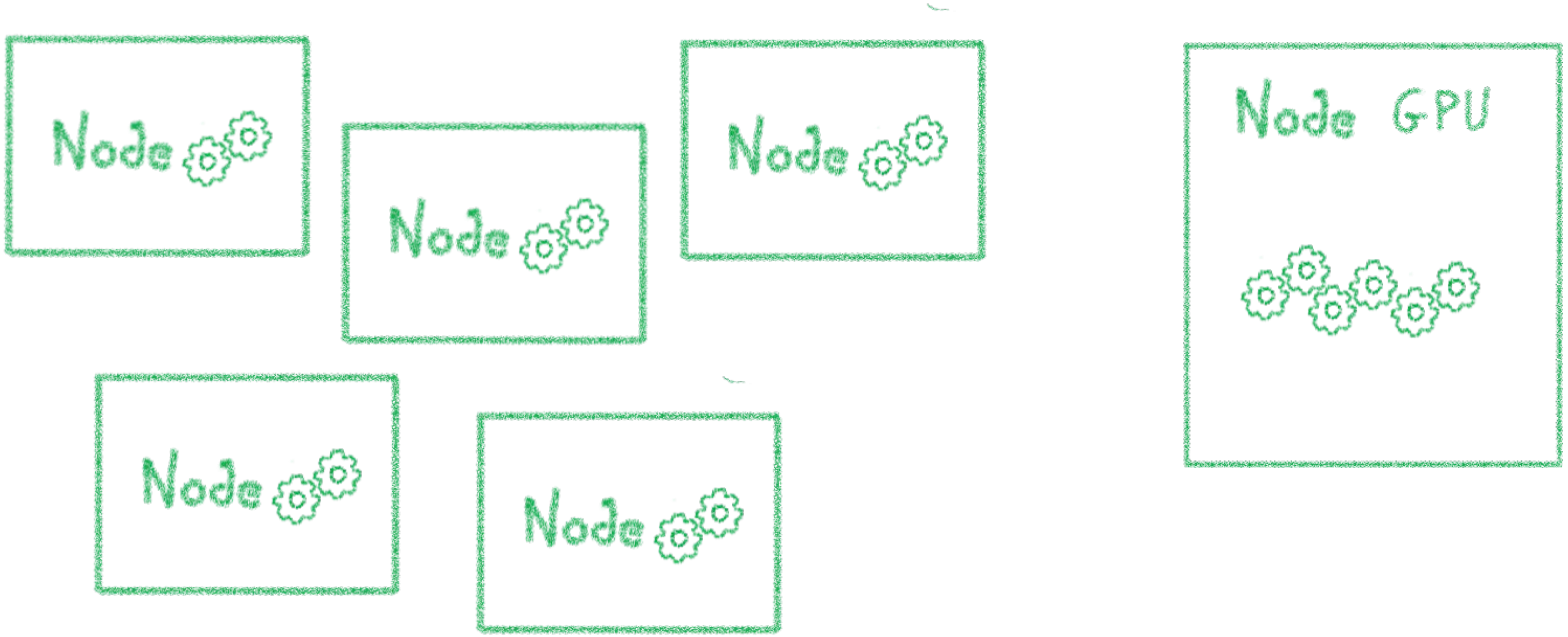
And this Pod can deploy on the tainted Node because of the Toleration

```
apiVersion: v1
kind: Pod
metadata:
 name: nginx
 labels:
   env: test
spec:
 containers:
 - name: nginx
   image: nginx
   imagePullPolicy: IfNotPresent
 tolerations:
 - key: "high-cpu"
   operator: "Exists"
   effect: "NoSchedule"
```

A Toleration matches a Taint if the keys and effects are the same and

- the operator is Exist
- the operator is Equal and value is the same

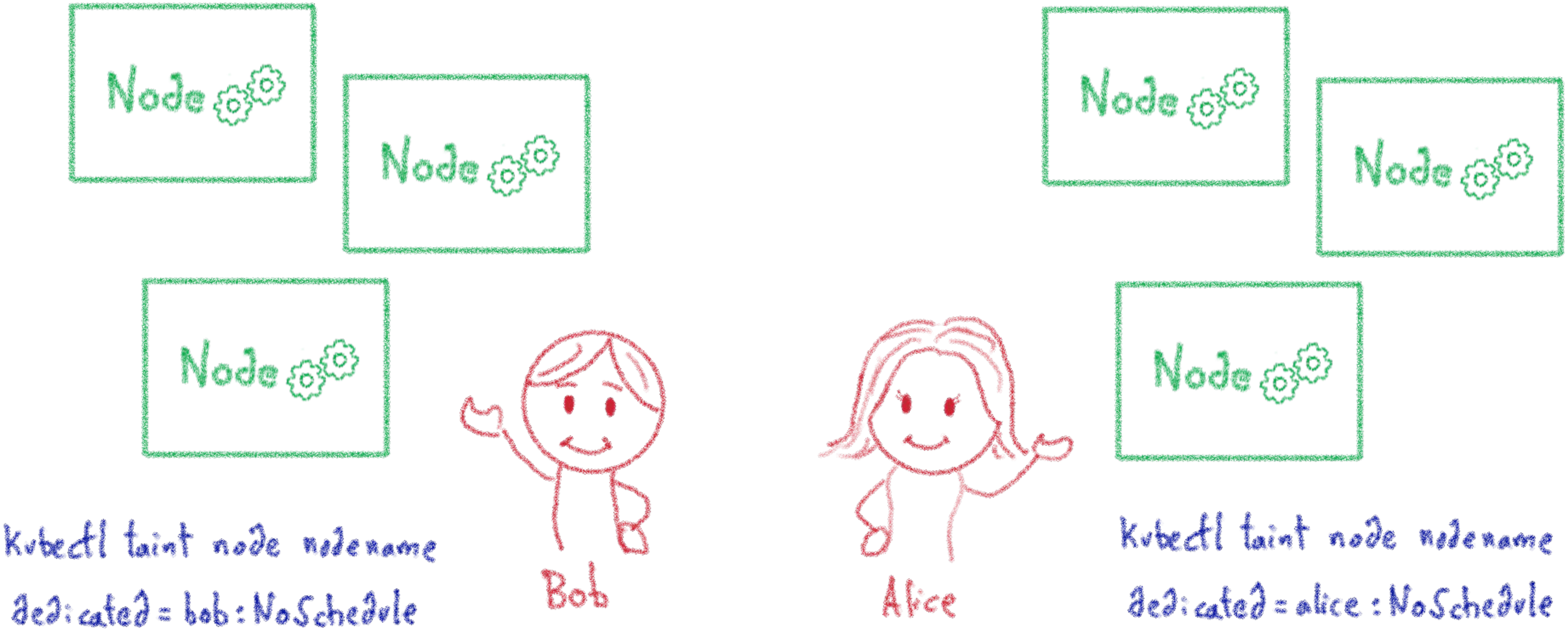
Example use cases for Taints



```
Kubectl taint node nodename
gpu-load=true:NoSchedule
```

Dedicated nodes

Example use cases for Taints



```
kubectl taint node nodename dedicated=bob:NoSchedule
```

```
kubectl taint node nodename dedicated=alice:NoSchedule
```

Nodes with Special Hardware

Affinity & Anti-affinity

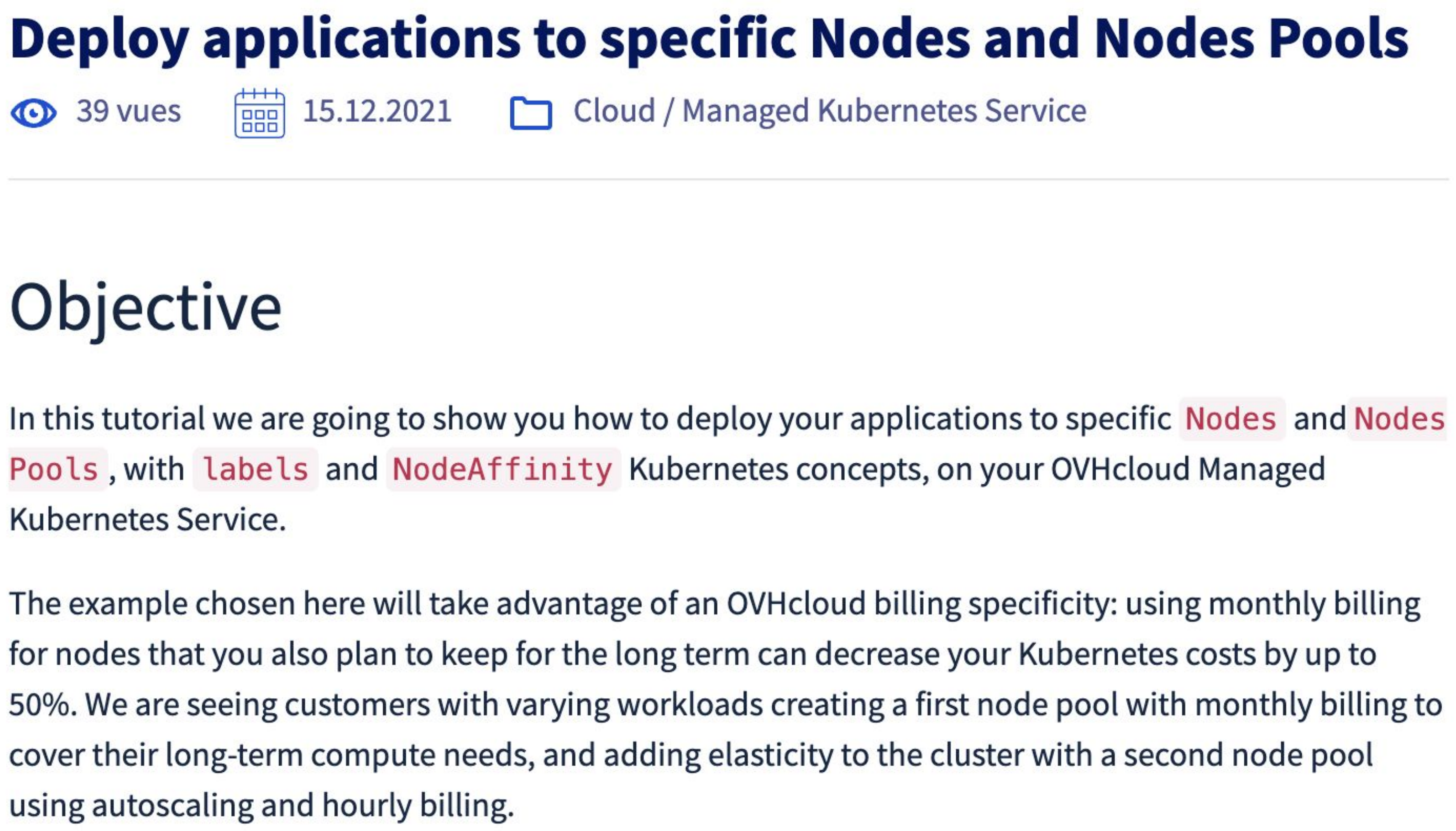
Node Affinity

rules that force the pod to be deployed, either exclusively or in priority, in certains nodes

Pod Affinity

indicate that a group of pods should always be deployed together on the same node (because of network communication, shared storage, etc.)

Deploy applications to specific Nodes



Deploy applications to specific Nodes and Nodes Pools

39 vues 15.12.2021 Cloud / Managed Kubernetes Service

Objective

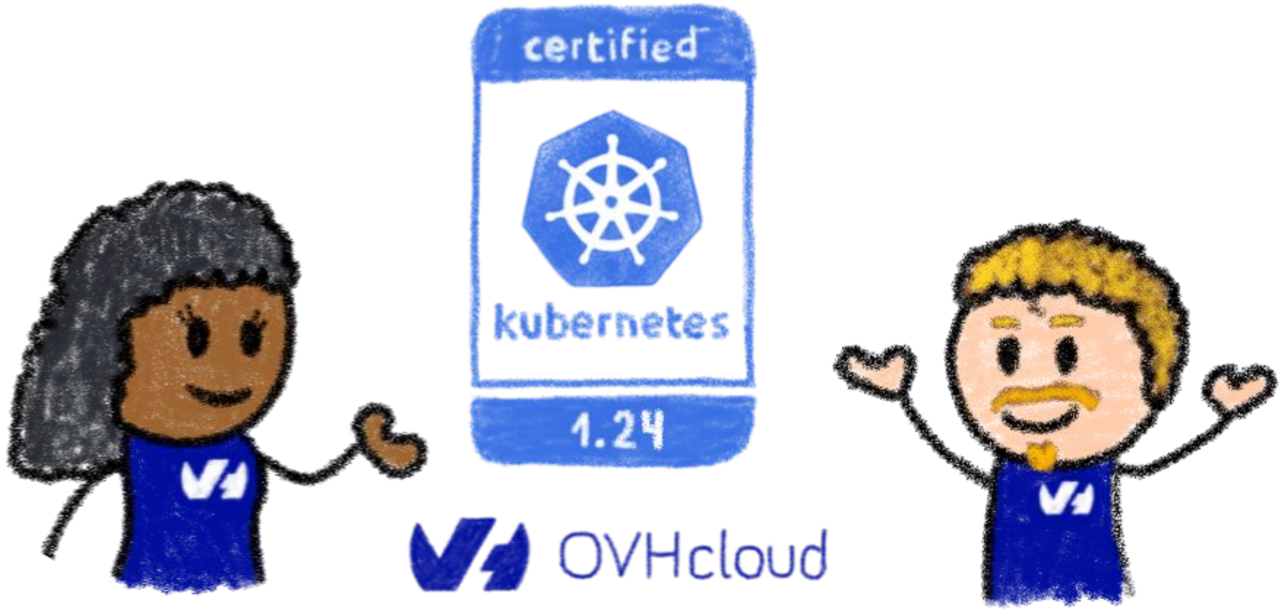
In this tutorial we are going to show you how to deploy your applications to specific `Nodes` and `Nodes Pools`, with `labels` and `NodeAffinity` Kubernetes concepts, on your OVHcloud Managed Kubernetes Service.

The example chosen here will take advantage of an OVHcloud billing specificity: using monthly billing for nodes that you also plan to keep for the long term can decrease your Kubernetes costs by up to 50%. We are seeing customers with varying workloads creating a first node pool with monthly billing to cover their long-term compute needs, and adding elasticity to the cluster with a second node pool using autoscaling and hourly billing.

https://help.ovhcloud.com/csm/fr-public-cloud-kubernetes-label-nodeaffinity-node-pools

OVHcloud Managed Kubernetes

Why would you choose ours?



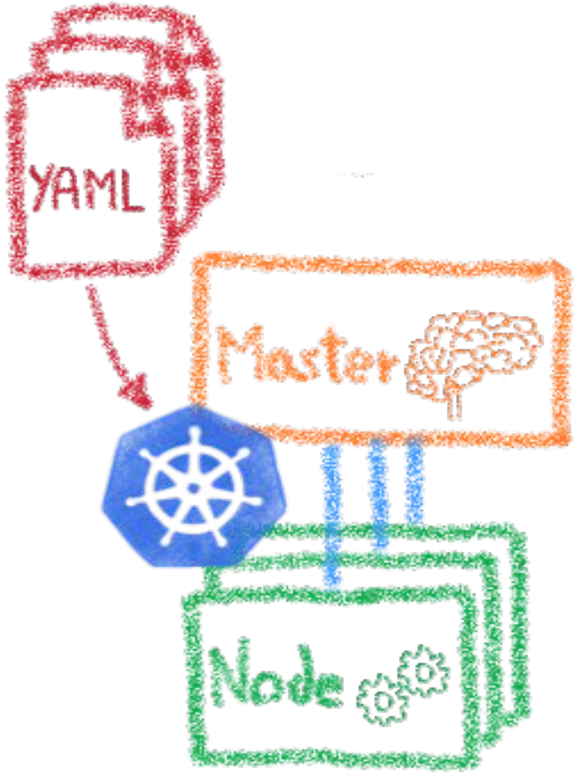
Certified Kubernetes platform

OVHcloud

Managed Kubernetes
certified Kubernetes 1.24

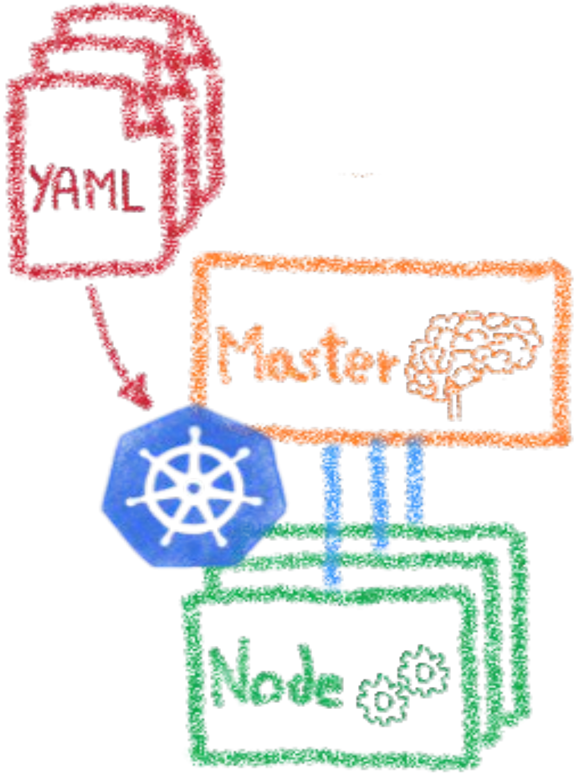
OVHcloud Managed Private Registry

Node Pools



Users can define node pools controlled from inside Kubernetes

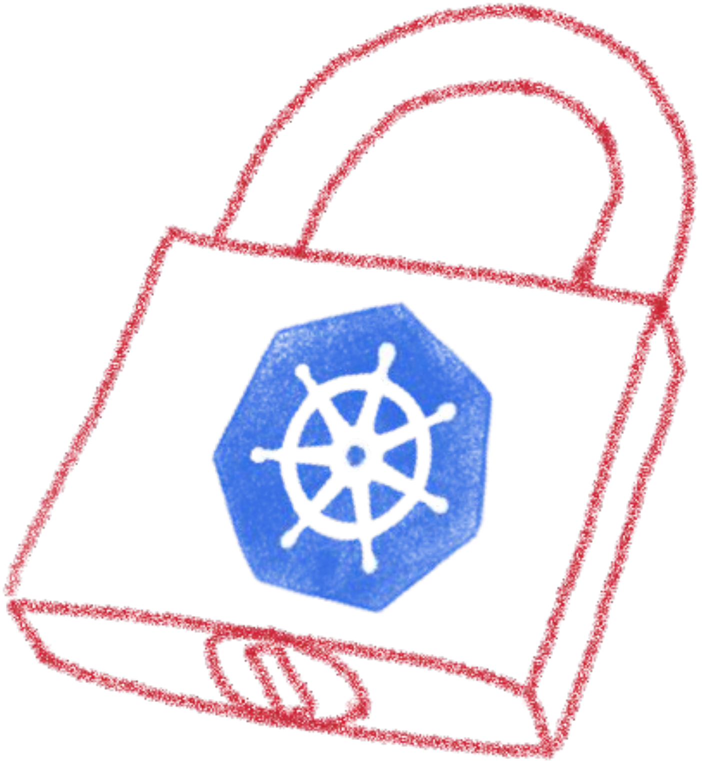
Autoscaling



Based on node pools

New instances are spawned or released based on load

Kubernetes in a private network



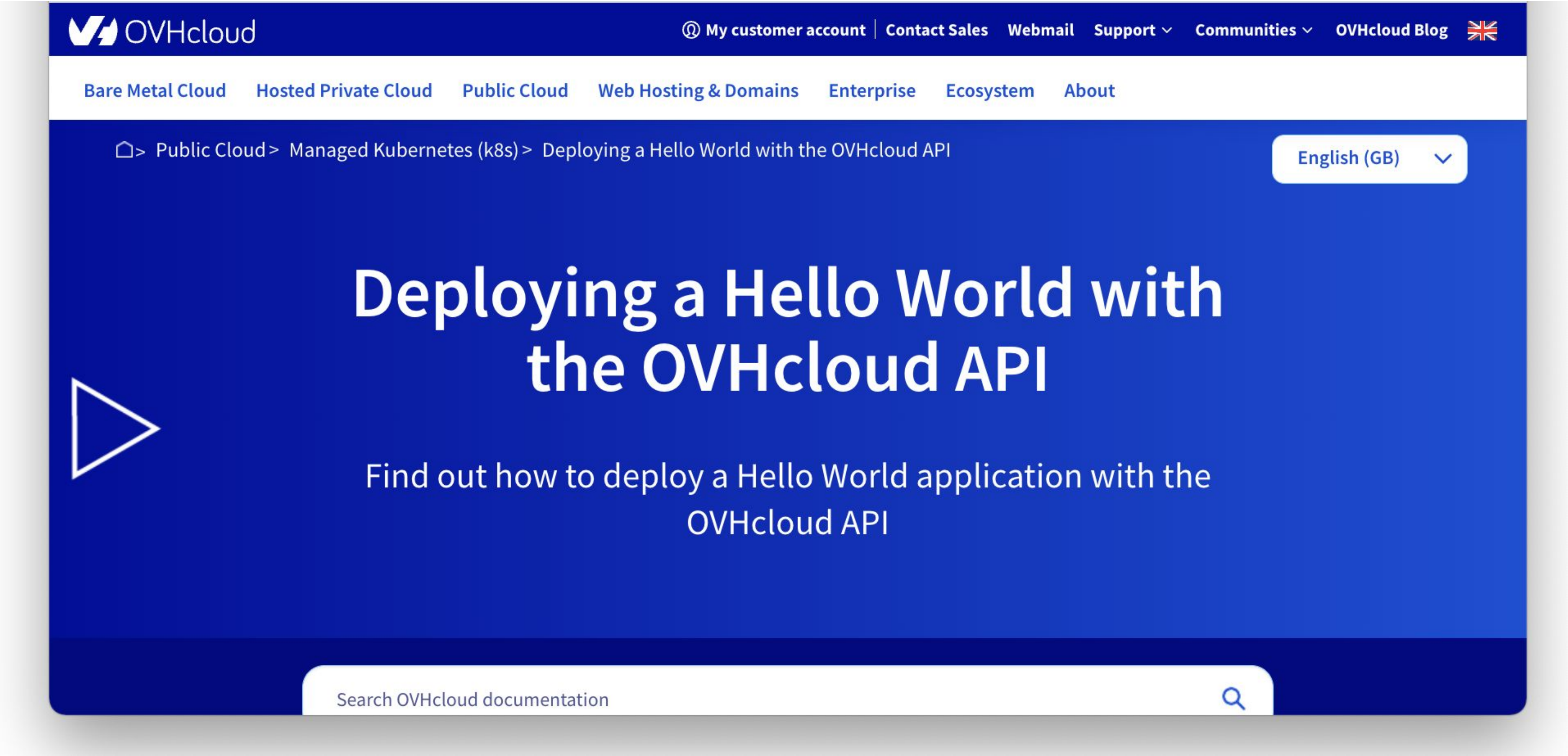
Other features

- Healthcare HDS 1 conformity
- ISO 27001/27701/27017/27018 conformity
- Terraform provider
- Control plane audit logs
- API server IP restrictions
- …

https://github.com/ovh/public-cloud-roadmap/projects/1

https://discord.com/invite/ovhcloud

Demo: Working with OVHcloud API



OVHcloud

My customer account | Contact Sales Webmail Support Communities OVHcloud Blog

Bare Metal Cloud Hosted Private Cloud Public Cloud Web Hosting & Domains Enterprise Ecosystem About

> Public Cloud > Managed Kubernetes (k8s) > Deploying a Hello World with the OVHcloud API

English (GB)

Deploying a Hello World with the OVHcloud API

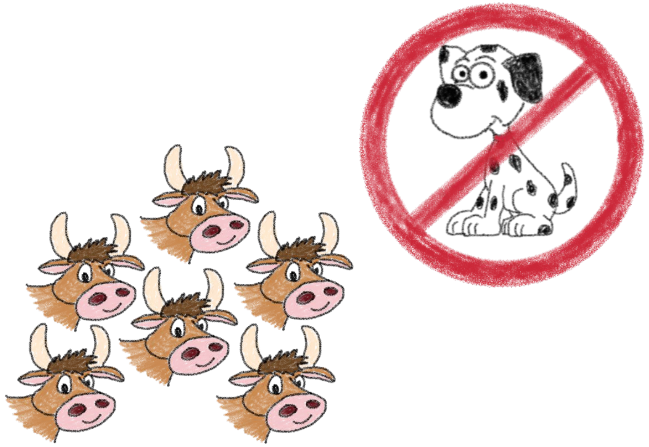
Find out how to deploy a Hello World application with the OVHcloud API

Search OVHcloud documentation

https://docs.ovh.com/gb/en/kubernetes/deploying-hello-world-ovh-api/

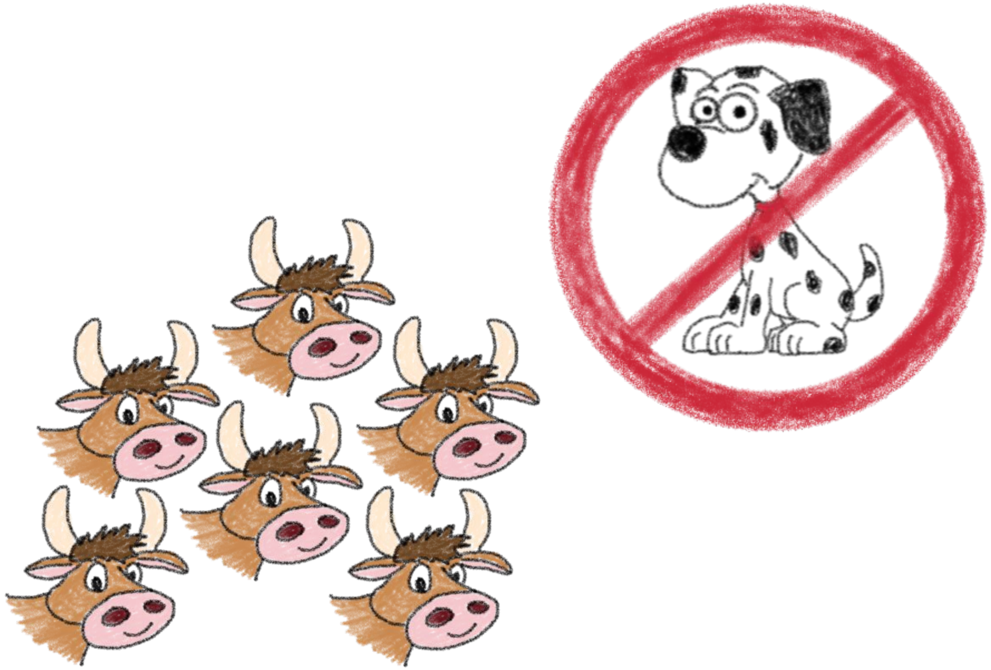
Infrastructure as Code

The perfect companion to a cloud



HashiCorp
Terraform

Infrastructure as Code (IaC)



Imperative - Instructions to follow step by step

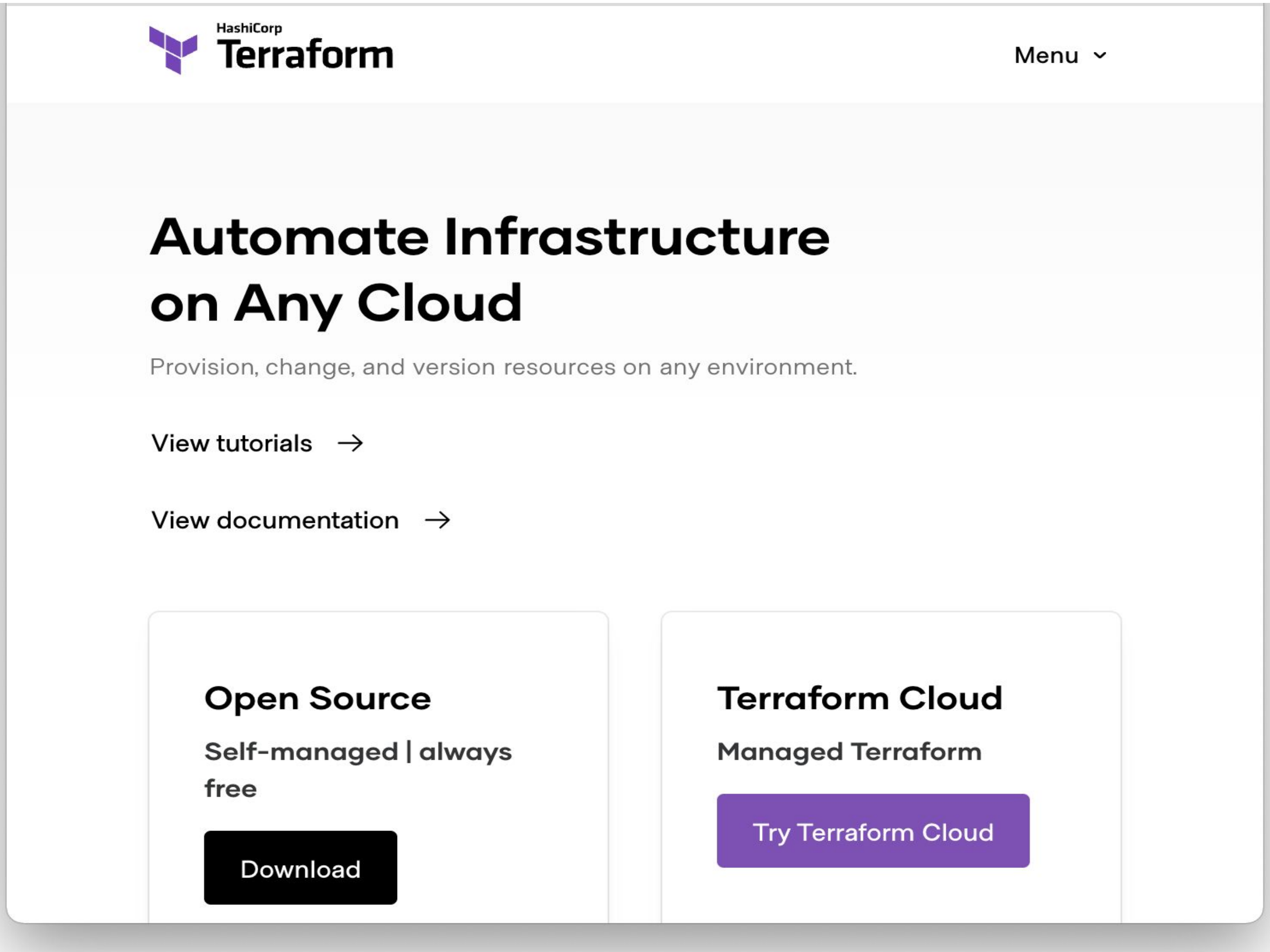
Declarative - Desired state description

Environment Aware - Intelligent desired state management

IaC tools

ANSIBLE
CHEF
HashiCorp
Terraform
puppet

HashiCorp Terraform



HashiCorp Terraform

Automate Infrastructure on Any Cloud

Provision, change, and version resources on any environment.

View tutorials →

View documentation →

Open Source

Self-managed | always free

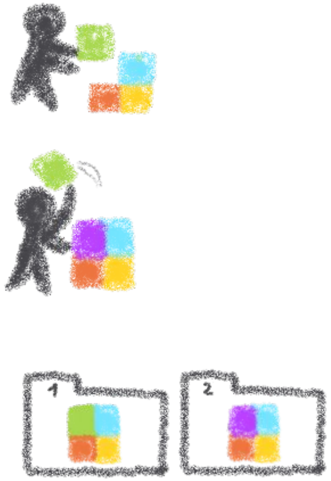
Terraform Cloud

Managed Terraform

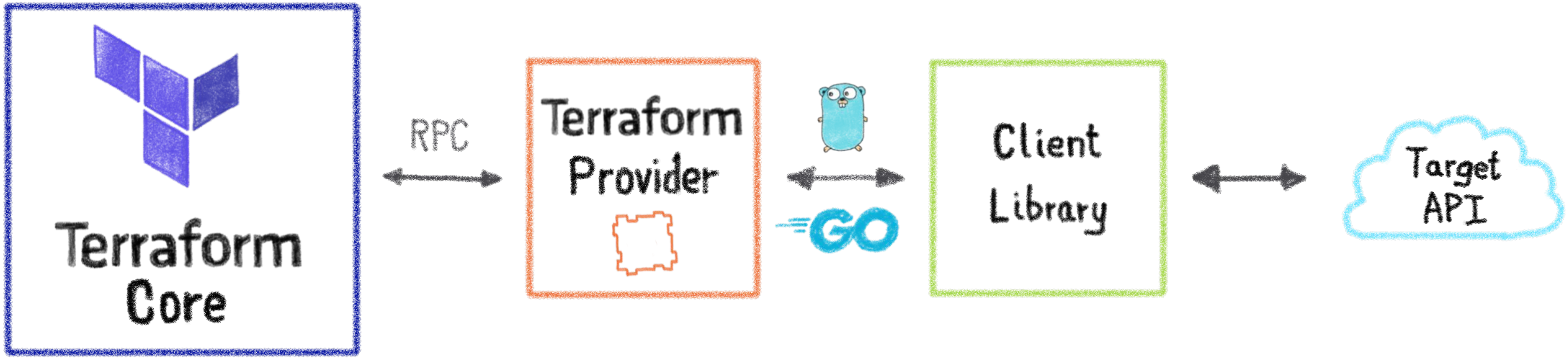
Terraform

- Build
- Modify
- Version

your infrastucture



Modular architecture: providers



Configuration packages: modules

Modules :

Collection of configuration files

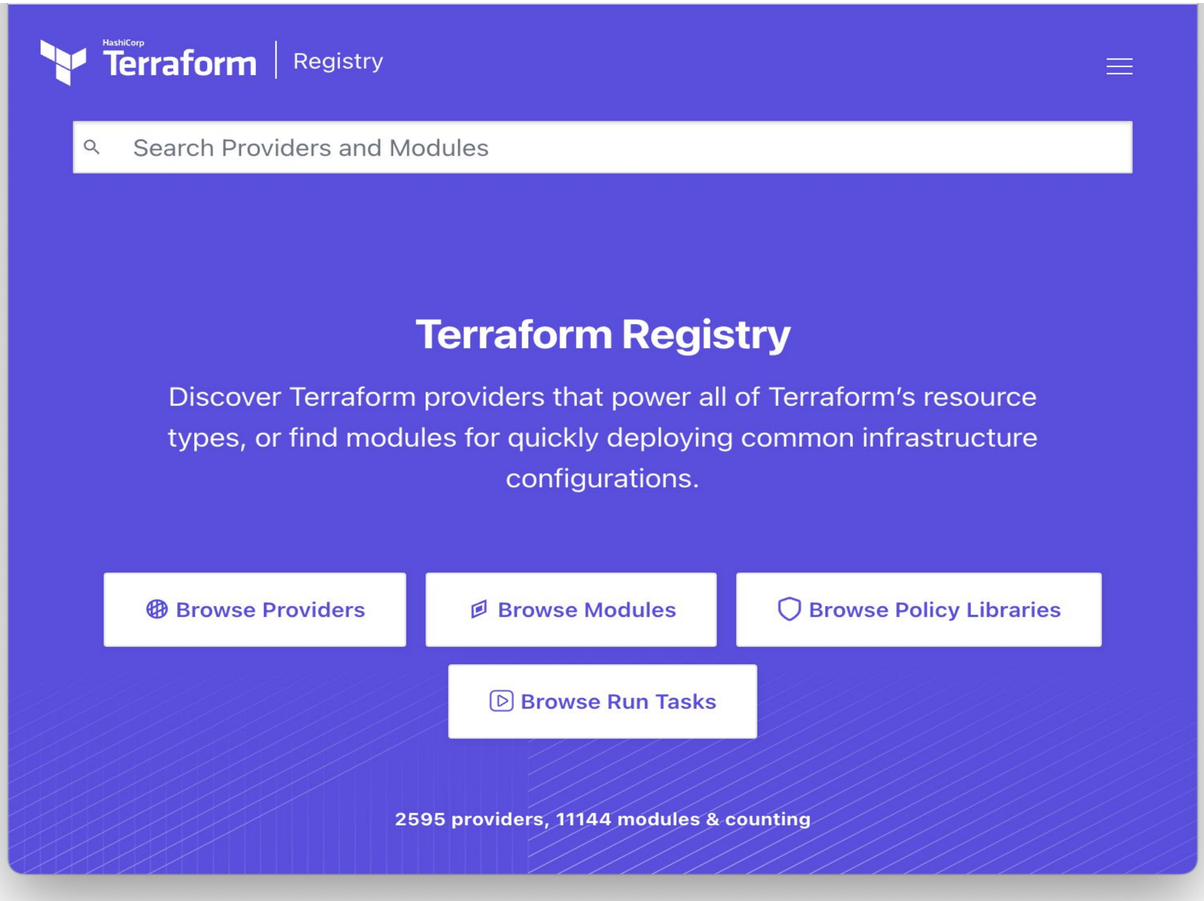
Terraform registry

Terraform Registry

Providers

&

Modules

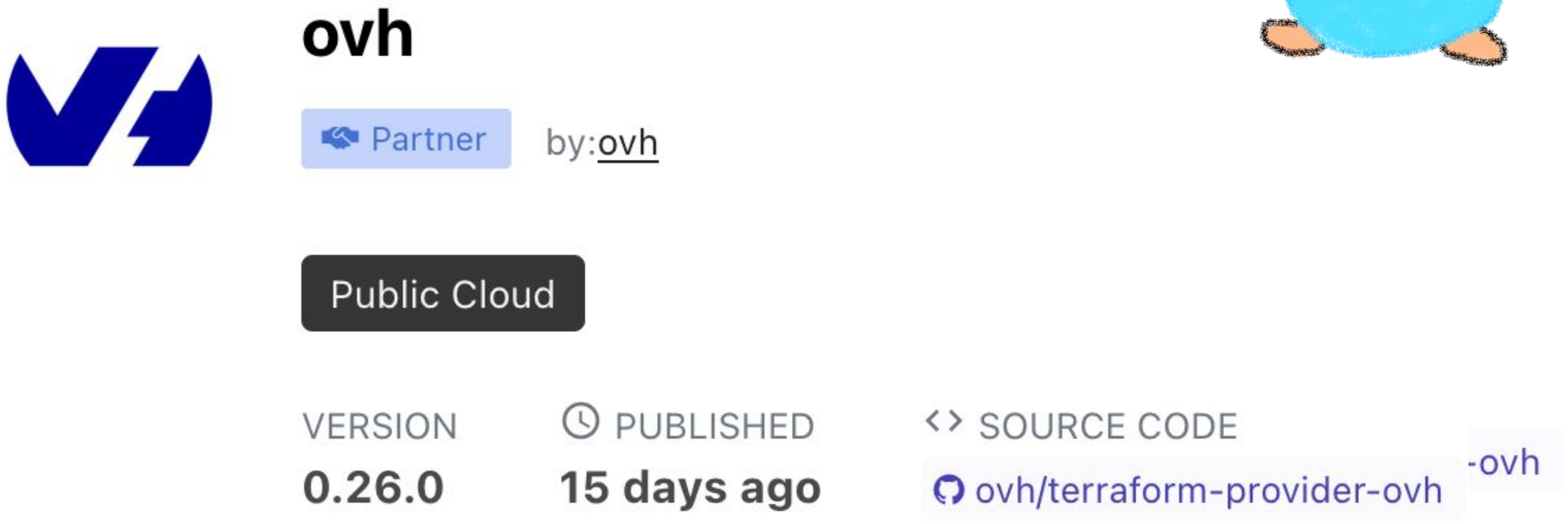


Terraform Registry

Discover Terraform providers that power all of Terraform's resource types, or find modules for quickly deploying common infrastructure configurations.

2595 providers, 11144 modules & counting

OVHcloud Terraform Provider



ovh

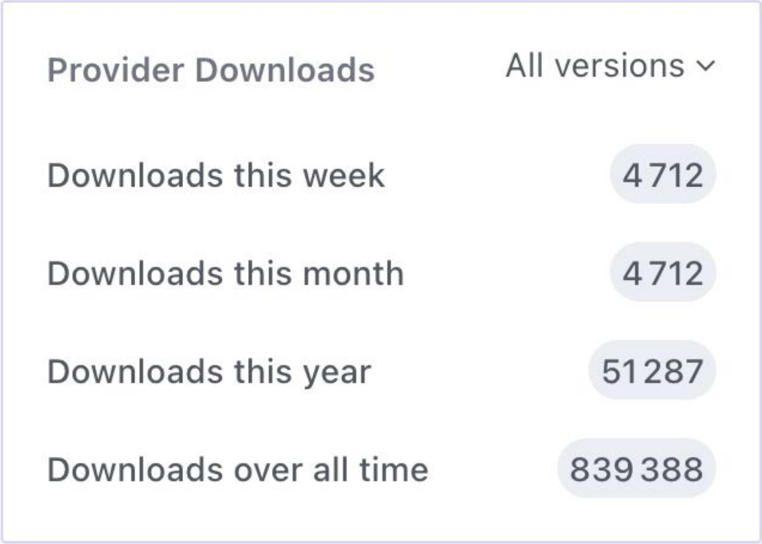
Partner by:ovh

Public Cloud

VERSION
0.26.0

PUBLISHED
15 days ago

SOURCE CODE
ovh/terraform-provider-ovh -ovh



Provider Downloads All versions

Downloads this week 4 712

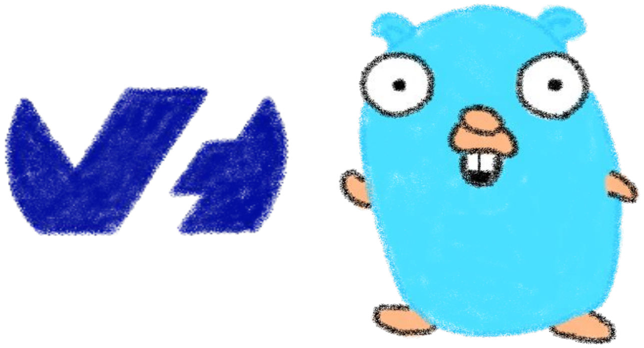
Downloads this month 4 712

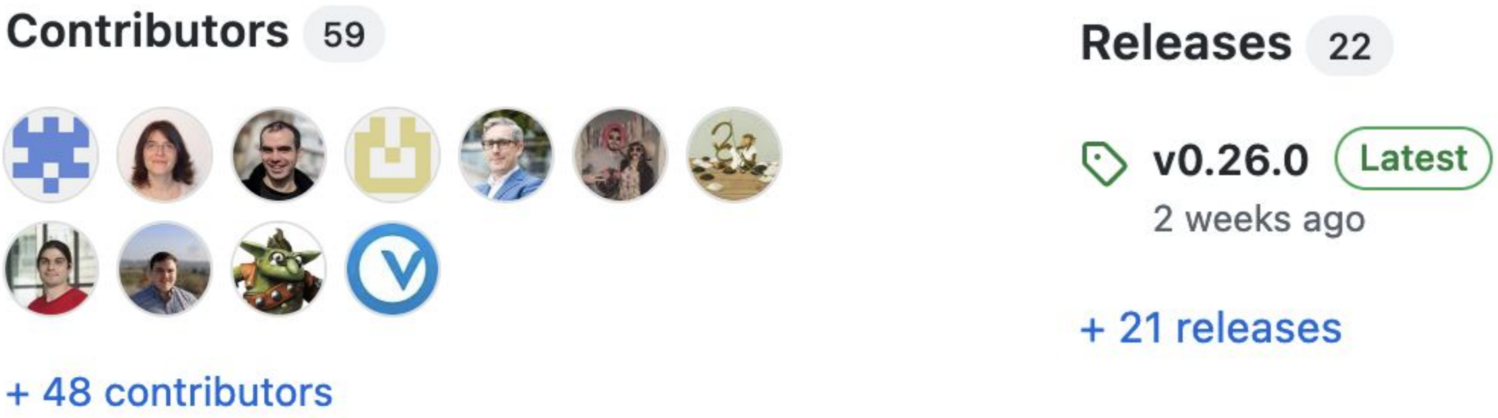
Downloads this year 51 287

Downloads over all time 839 388

https://registry.terraform.io/providers/ovh/ovh/latest/docs

OVHcloud Terraform Provider





Contributors 59

+ 48 contributors

Releases 22

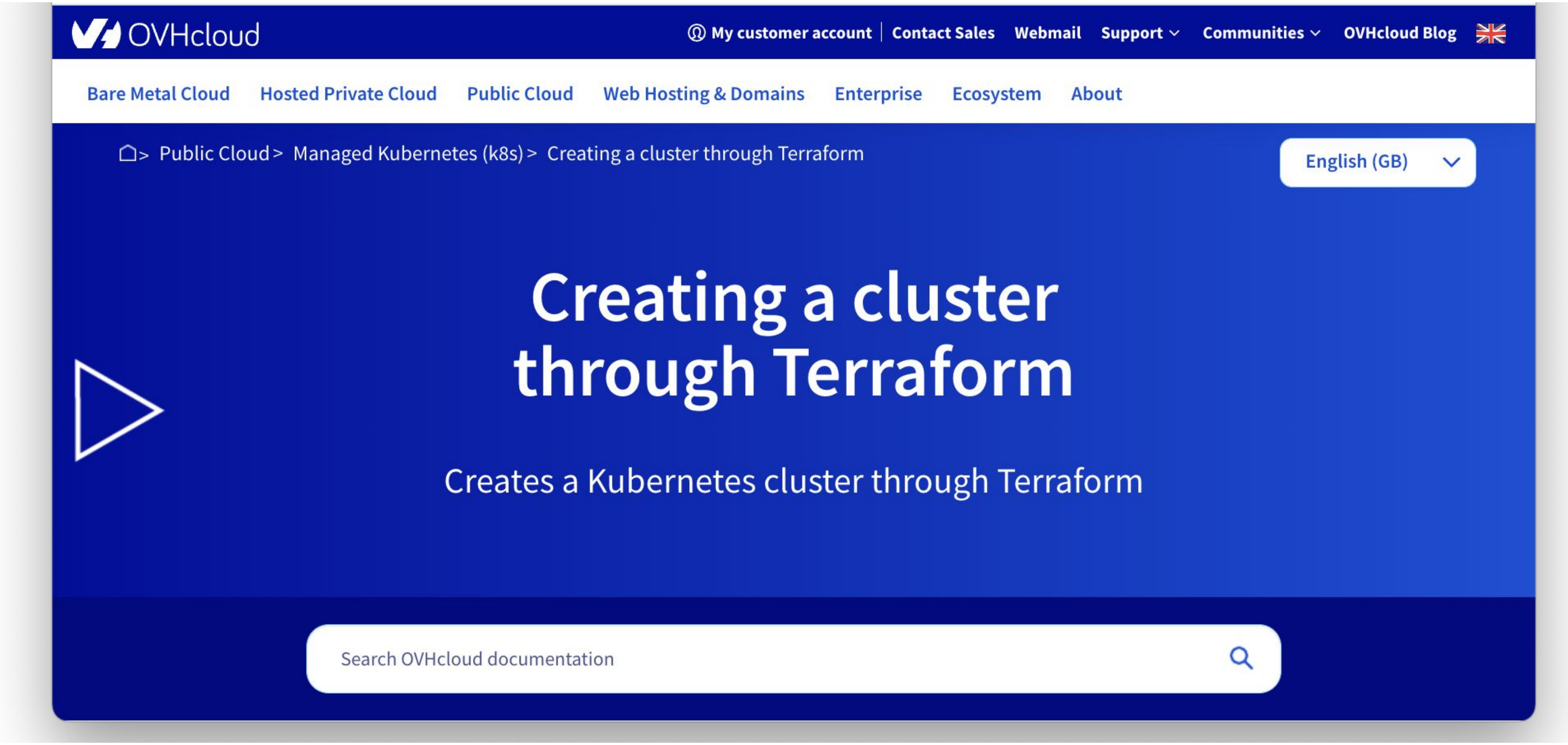
v0.26.0 Latest

2 weeks ago

+ 21 releases

https://github.com/ovh/terraform-provider-ovh

Demo: Using Terraform



OVHcloud My customer account | Contact Sales Webmail Support Communities OVHcloud Blog

Bare Metal Cloud Hosted Private Cloud Public Cloud Web Hosting & Domains Enterprise Ecosystem About

> Public Cloud > Managed Kubernetes (k8s) > Creating a cluster through Terraform

English (GB)

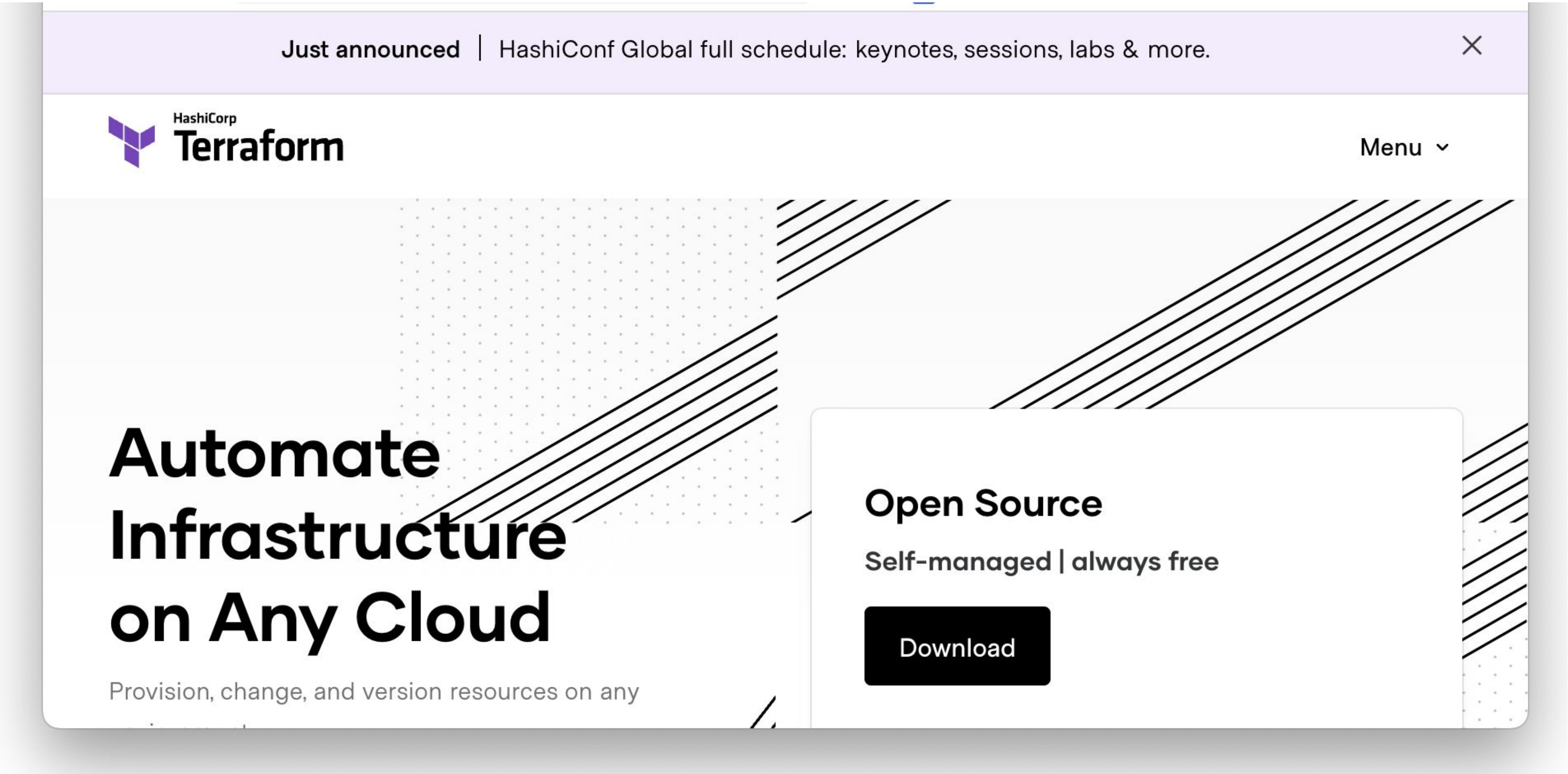
Creating a cluster through Terraform

Creates a Kubernetes cluster through Terraform

Search OVHcloud documentation

https://docs.ovh.com/gb/en/kubernetes/creating-a-cluster-through-terraform/

Needed tools: terraform



Just announced | HashiConf Global full schedule: keynotes, sessions, labs & more.

Automate Infrastructure on Any Cloud

Provision, change, and version resources on any

Open Source

Self-managed | always free

https://www.terraform.io/

That's all, folks!

Thank you all!