{RIVIERADEV}

Kubernetes, dépassionné et pour les ultra débutants

Sébastien Blanc, Sun Tan & Horacio Gonzalez

2023-07-10

aiven

Red Hat

OVHcloud

Who are we?

Sébastien Blanc

DevRel
Aiven
@sebi2706

Horacio Gonzalez

DevRel
OVHCloud
@LostInBrittany

Sun Tan

Senior Software Engineer
Red Hat
@__sunix_

Agenda

Introduction

- Why Kubernetes
- Containers
- What is Kubernetes?

1 - Diving into K8s building blocks

- Playing with kubectl
- YAML

2 - Being a good cloud native citizen

- Requests and limits
- Health probes
- ConfigMap and Secrets

3 - Advanced K8s

- Persistent Volumes
- Tolerance and taints
- Operators

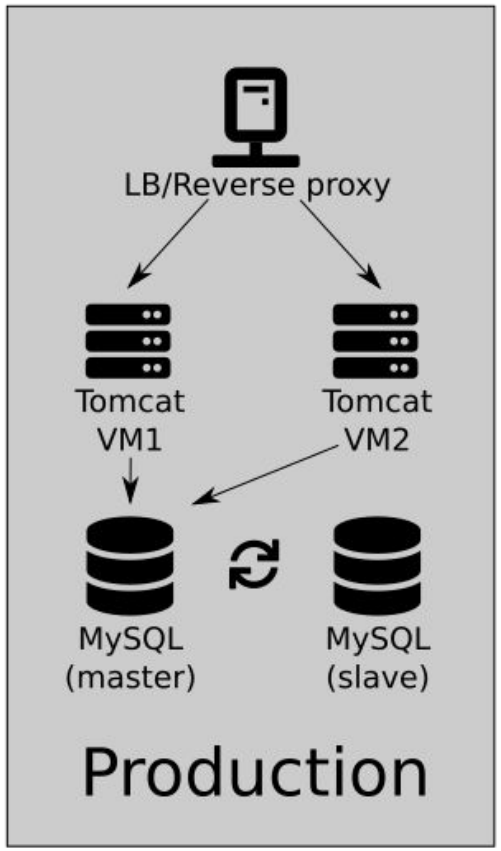
Introduction

Why Kubernetes?

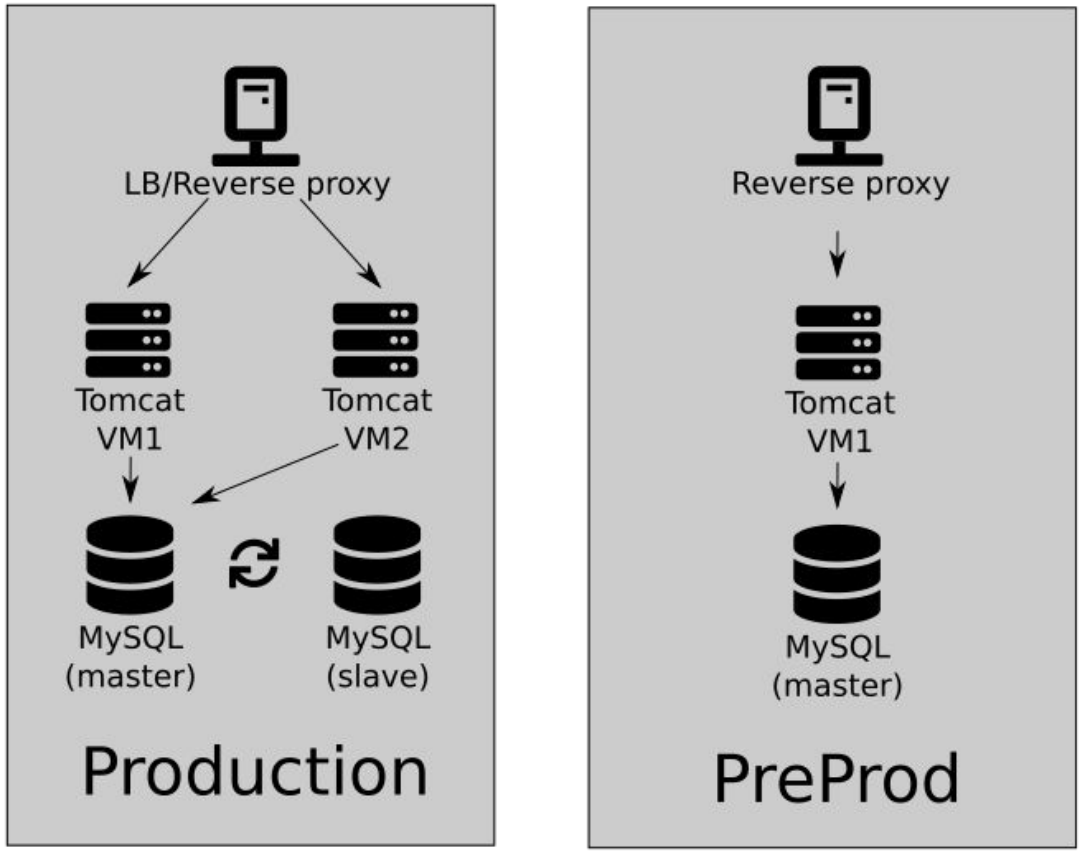
Containers

What is Kubernetes?

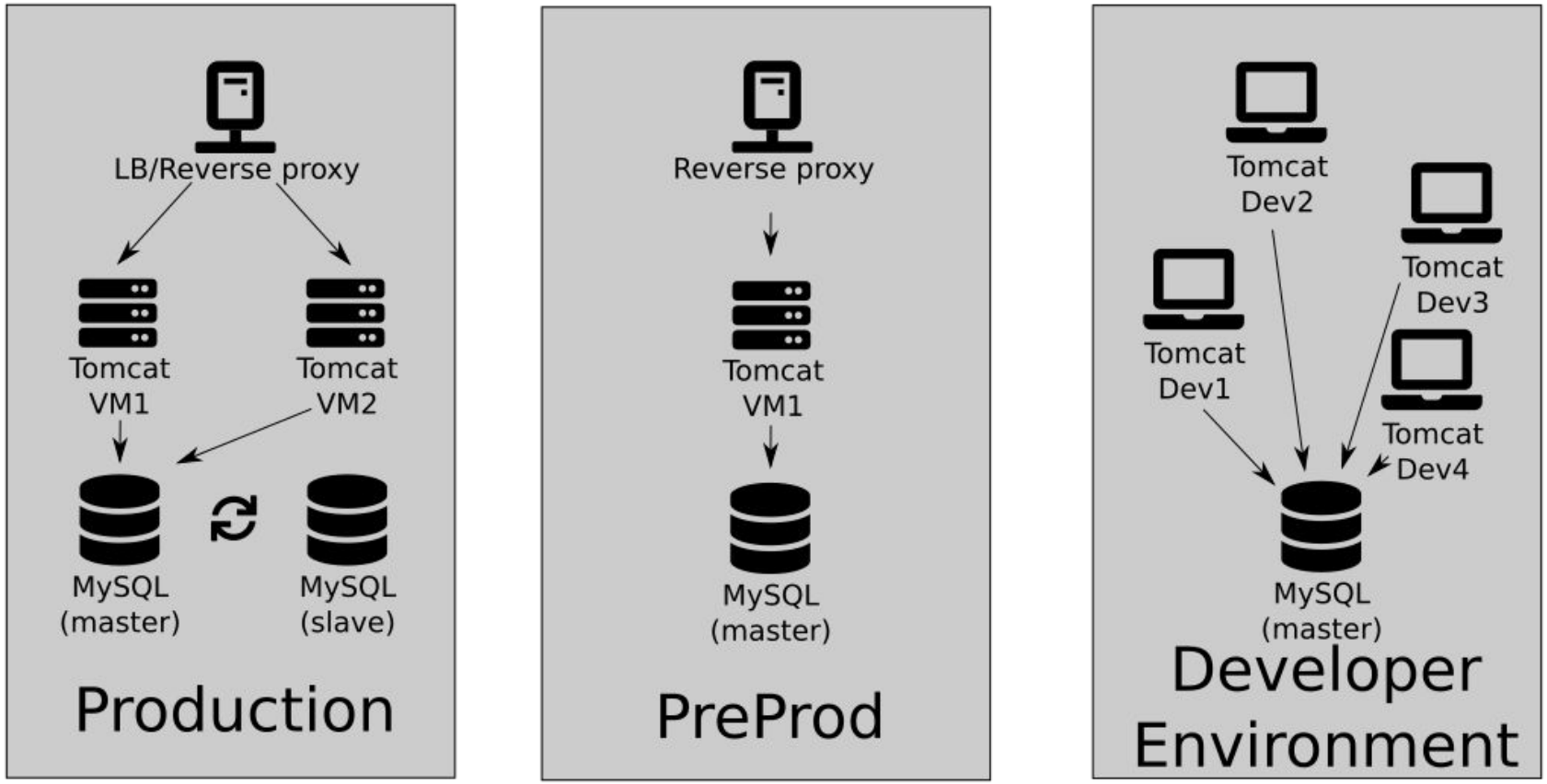
A typical Java application



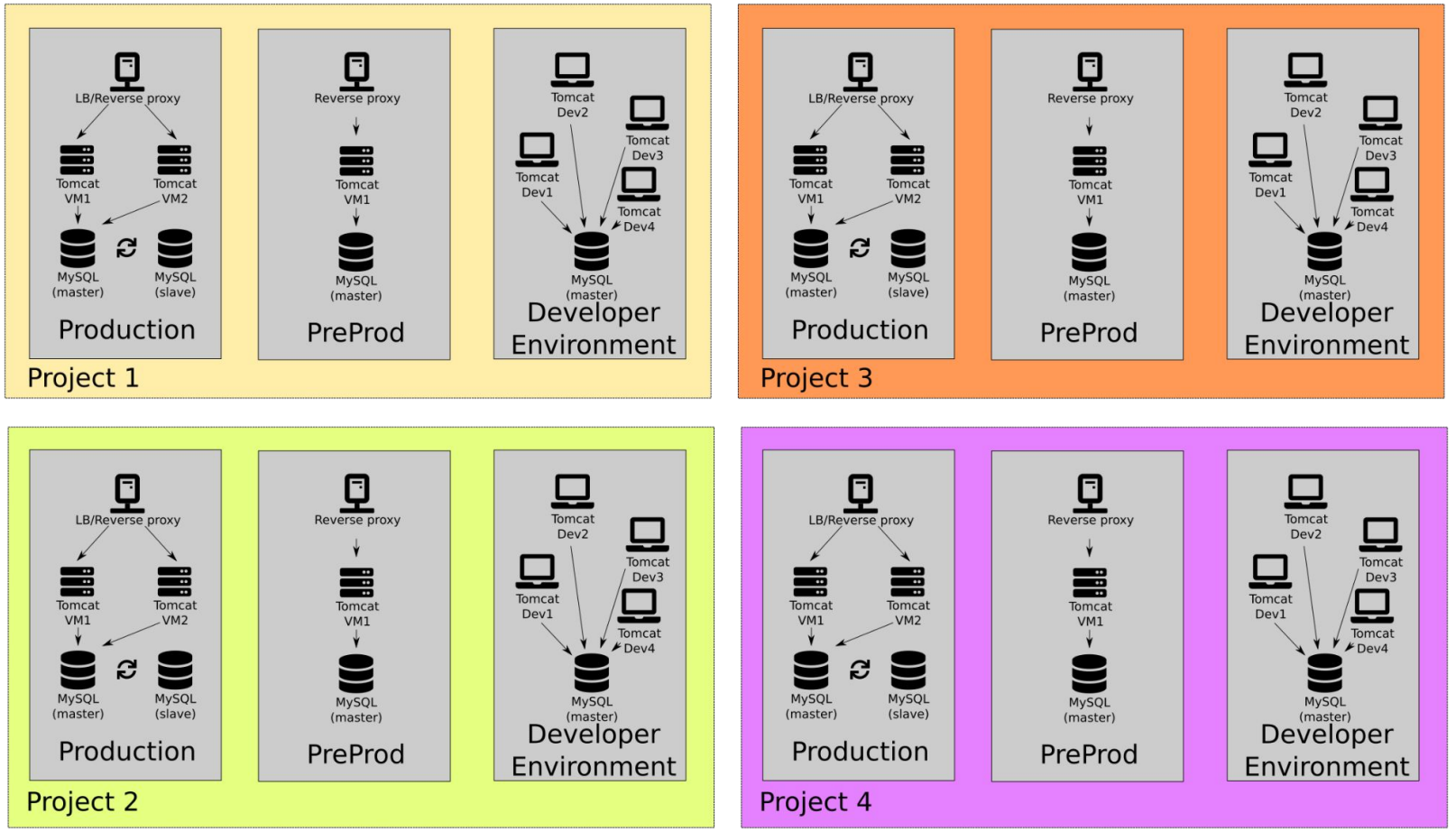
A typical Java application



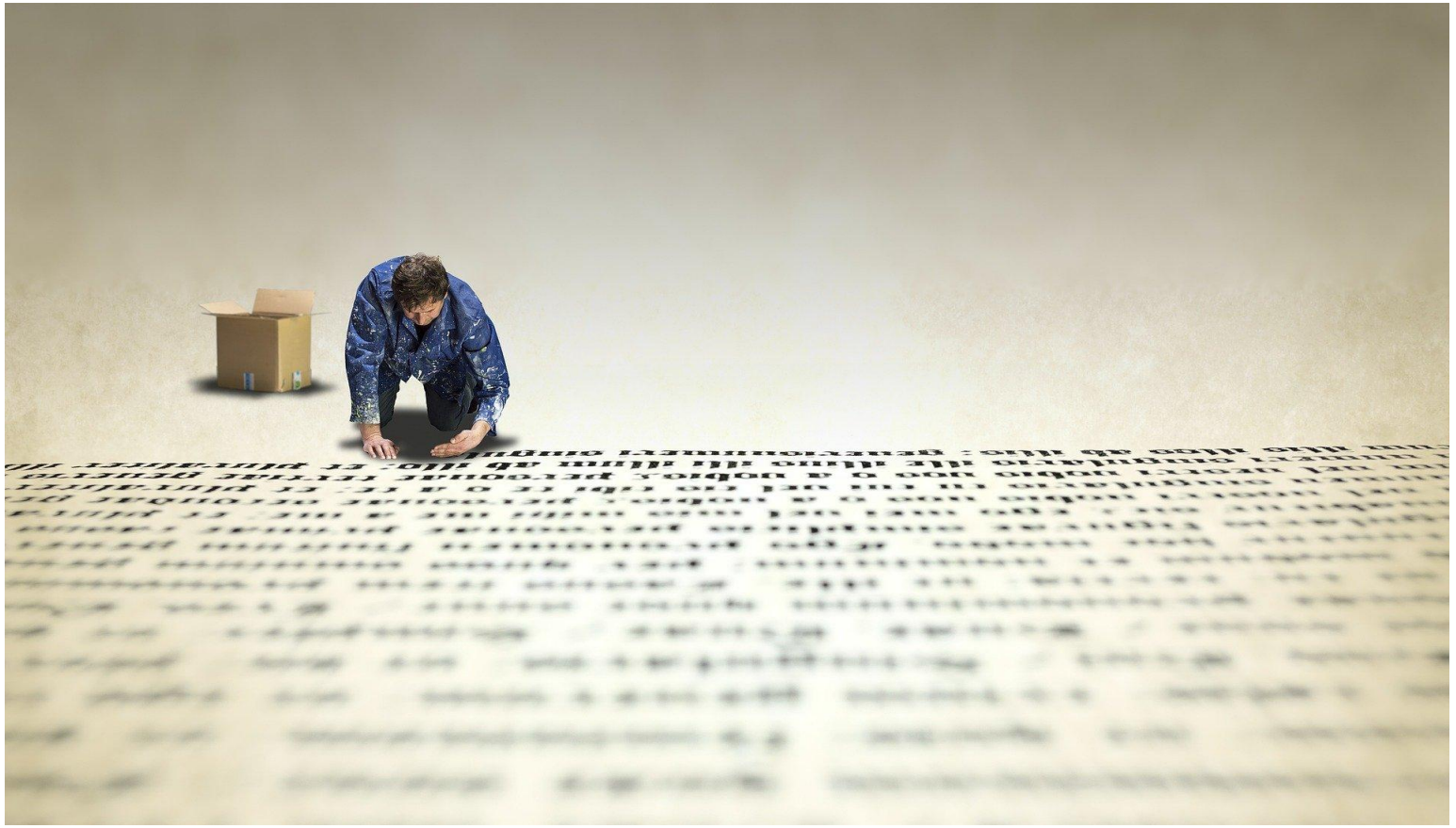
A typical Java application



A typical Java application



Pain point #1

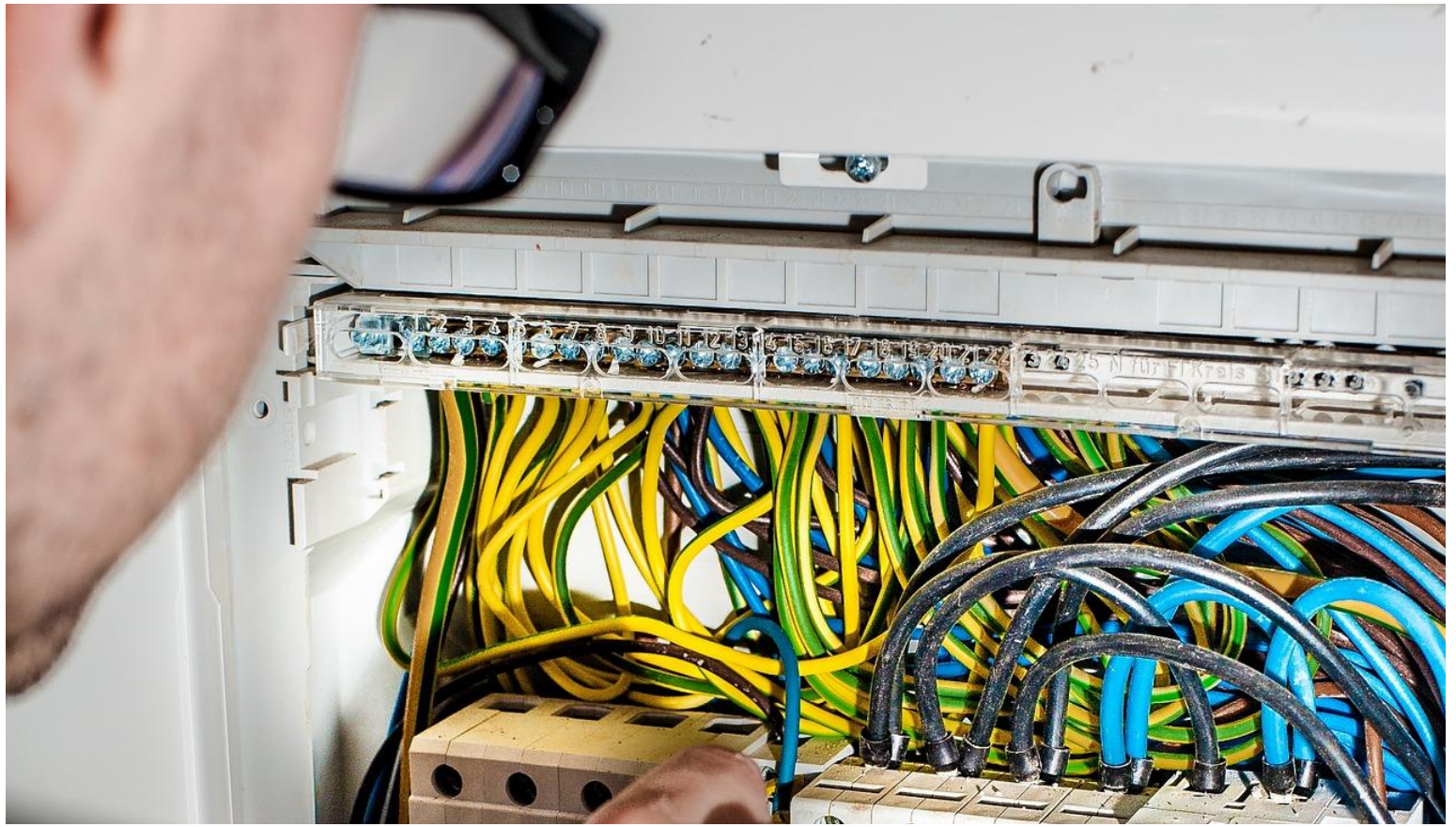


Manual deployments

Pain point #2

Scaling

Pain point #3



Developer Environment

Kubernetes

To the rescue!

Kubernetes seems too difficult

Think big, start small, learn fast

Think Big,

Start Small,

~~Scale~~ Learn Fast

--

Jim Carroll

Start small with Containers

Containers are used in Kubernetes

Containers can be used without Kubernetes

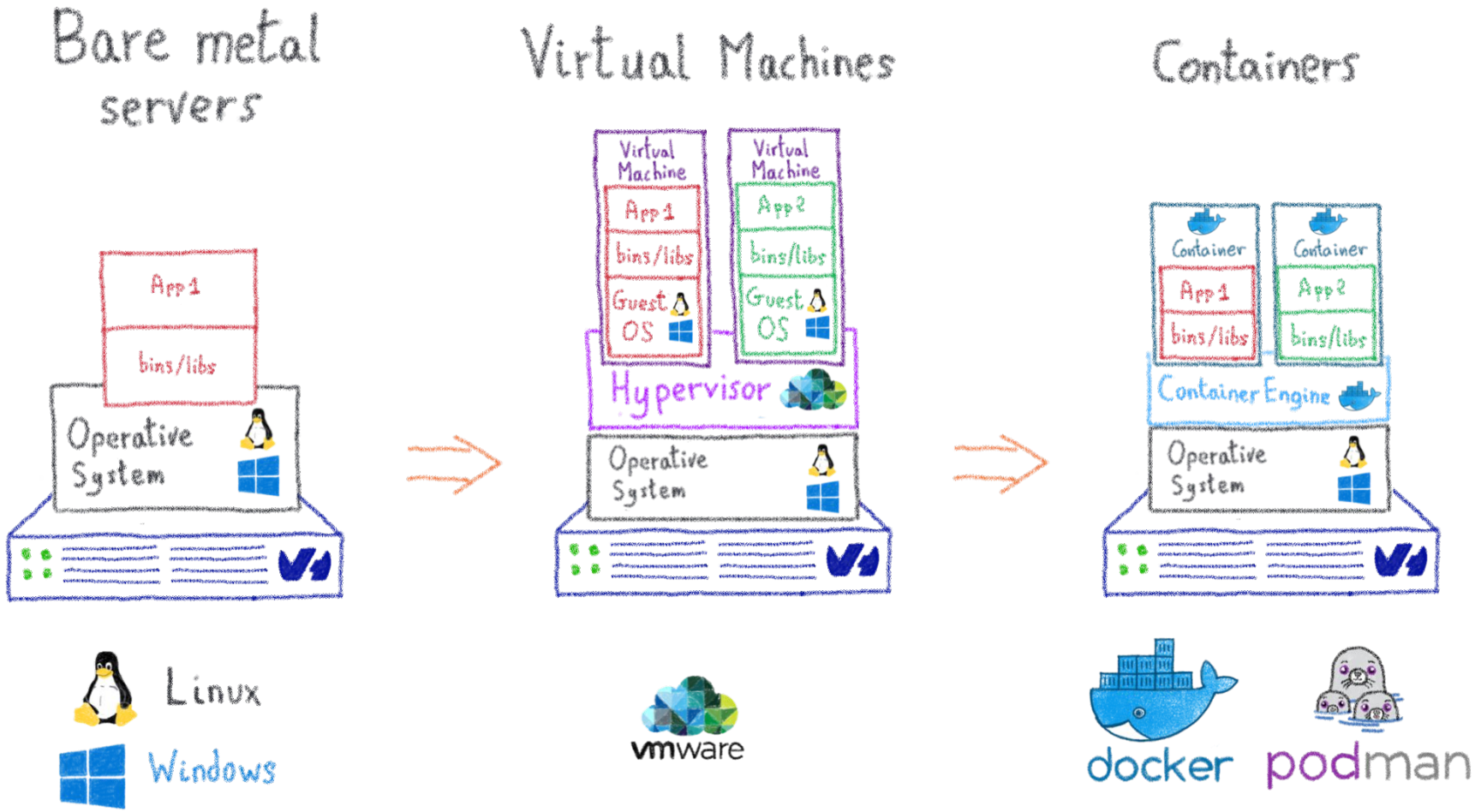
Introduction

Why Kubernetes?

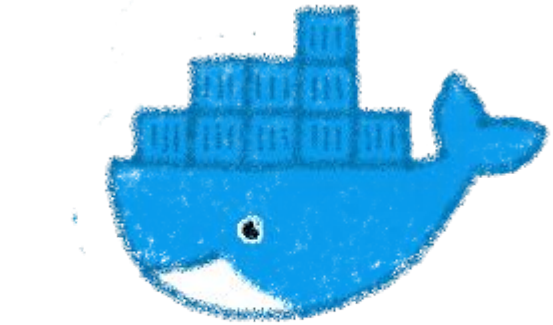
Containers

What is Kubernetes?

Container evolution



Container tools



The most popular

Daemon less

Pods/containers

Containers were there for a while

1979: Unix V7 (Chroot)

2000: FreeBSD Jails

2001: Linux VServer

2004: Solaris Containers

2005: Open VZ (Open Virtuzzo)

2006: Process Containers (cgroups)

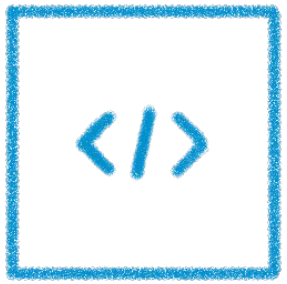
2008: LXC

2011: Warden

2013: LMCTFY

2013: Docker

Container image



Source code

Basically a Dockerfile

Build

Using `Docker or Podman

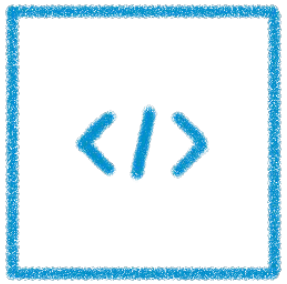
Push/Pull

Optionally to a container image registry like dockerhub or quay.io

Run anywhere

Any linux host that support container technology should be able to run it.

vs a Java application



Source code

Basically Java files

Build

Using Maven or Gradle

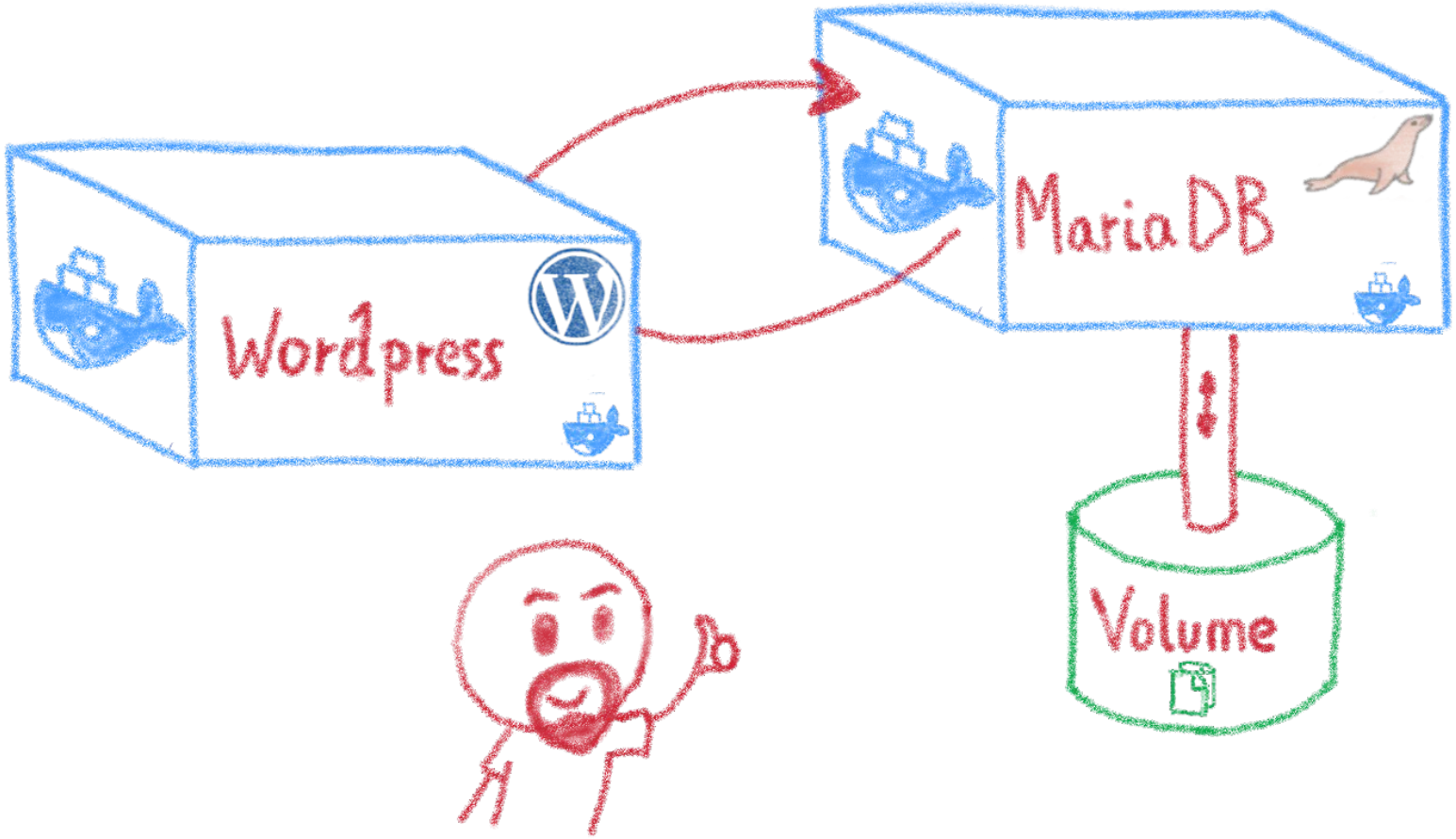
Push/Pull

Optionally to a Maven repo like Nexus or Artifactory

Run anywhere

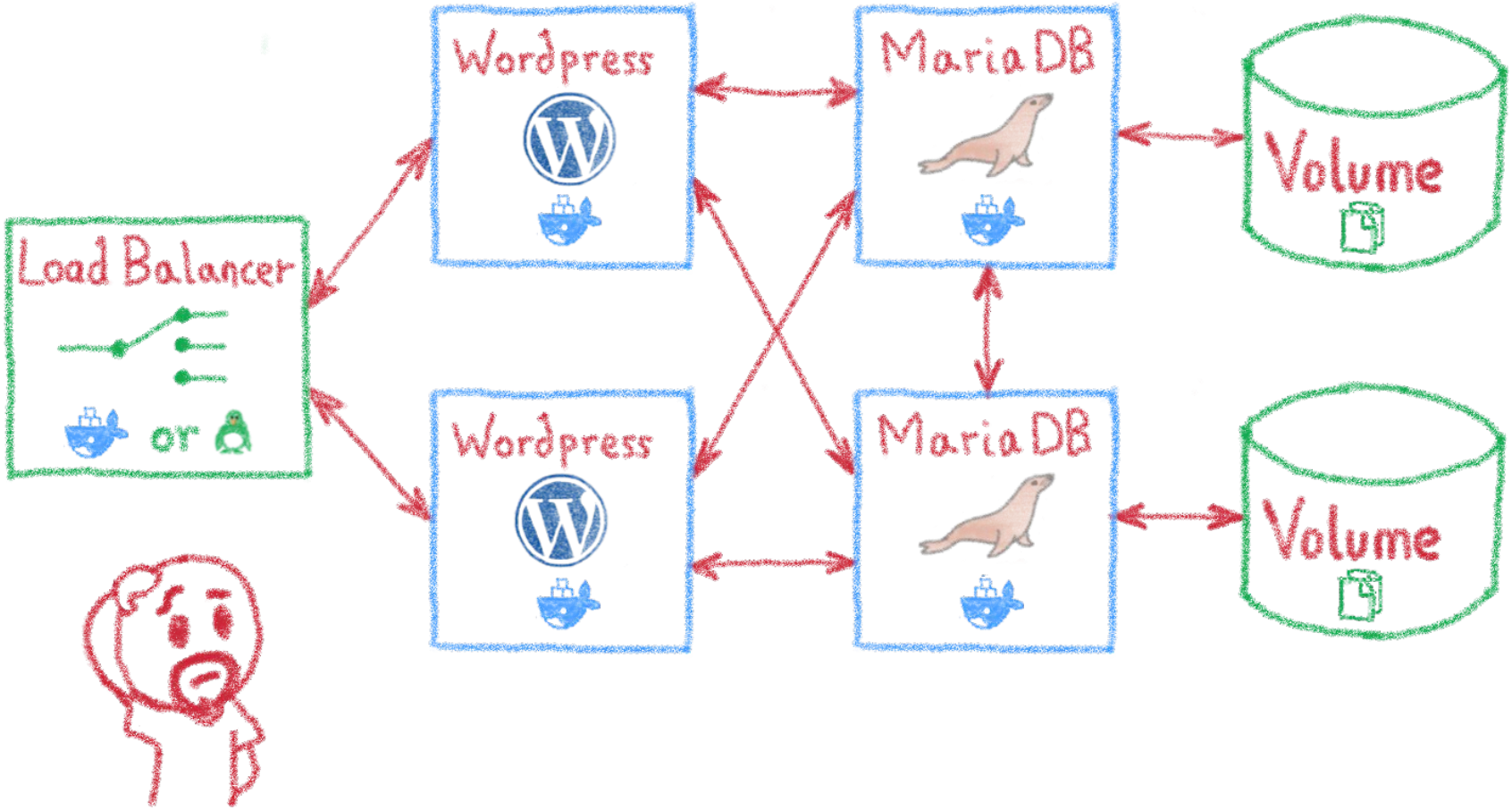
Any OS host that support JVM technology should be able to run it.

Containers are easy…



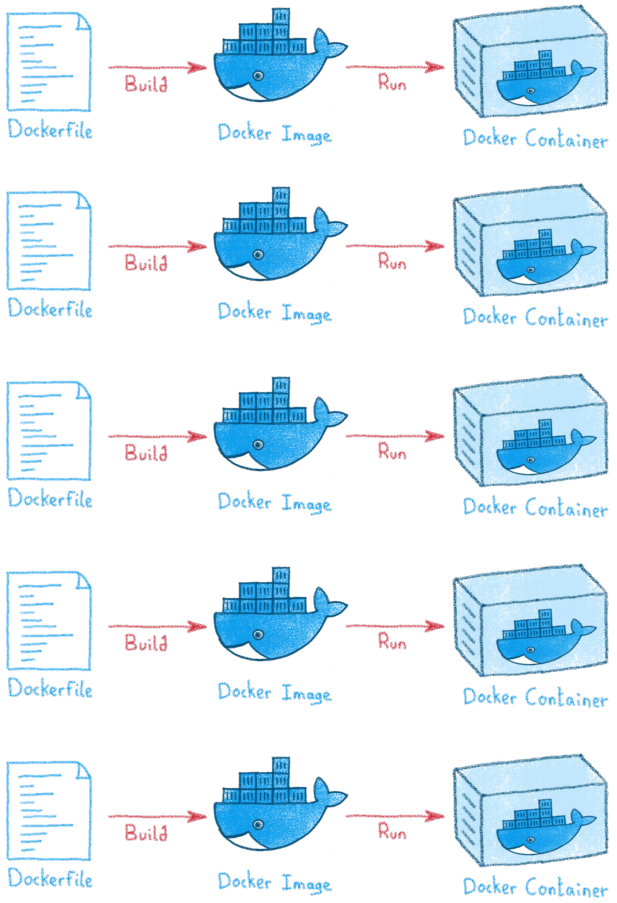
For developers

Less simple if you must operate them



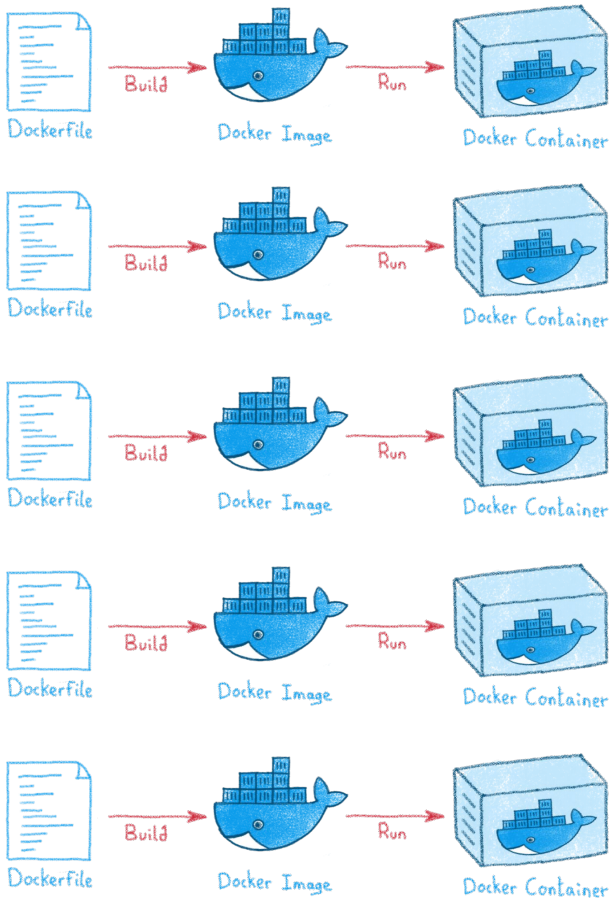
Like in a production context

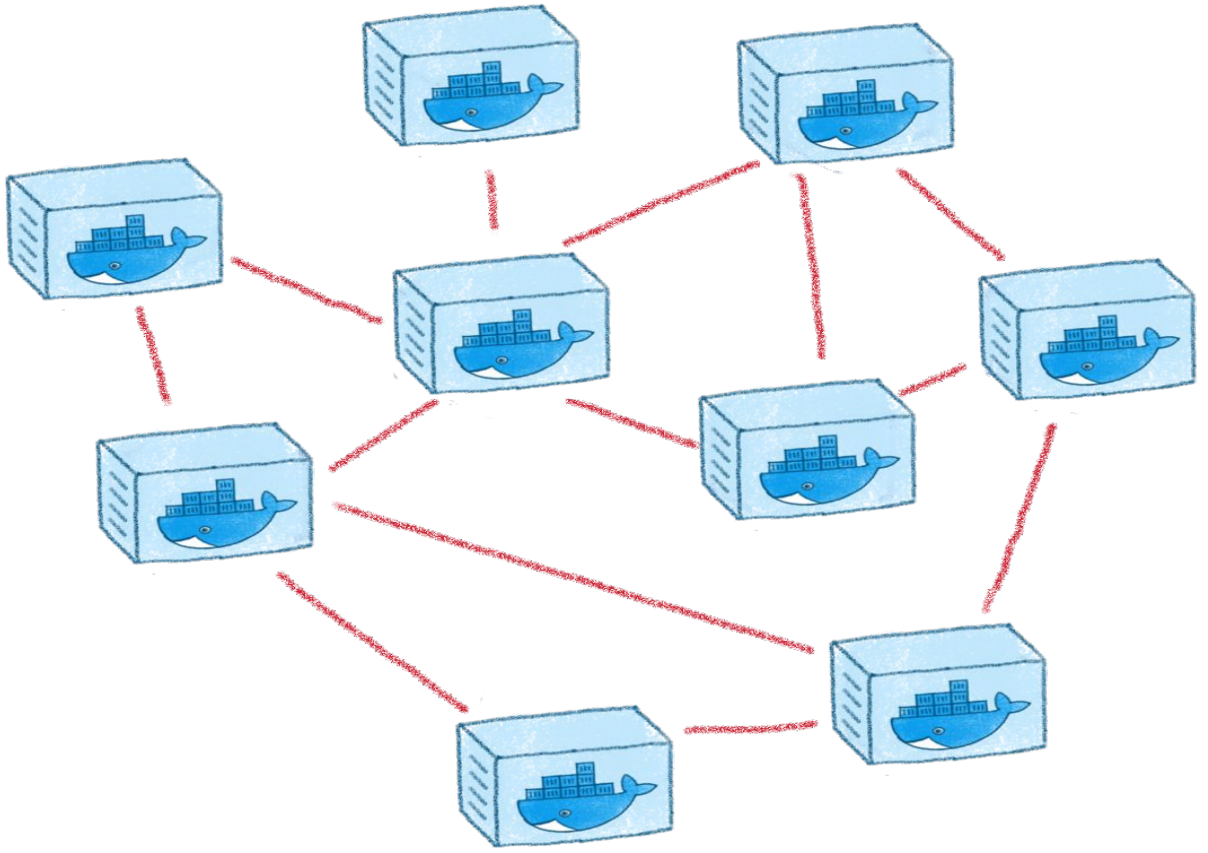
And what about microservices?



Are you sure you want to operate them by hand?

And what about microservices?



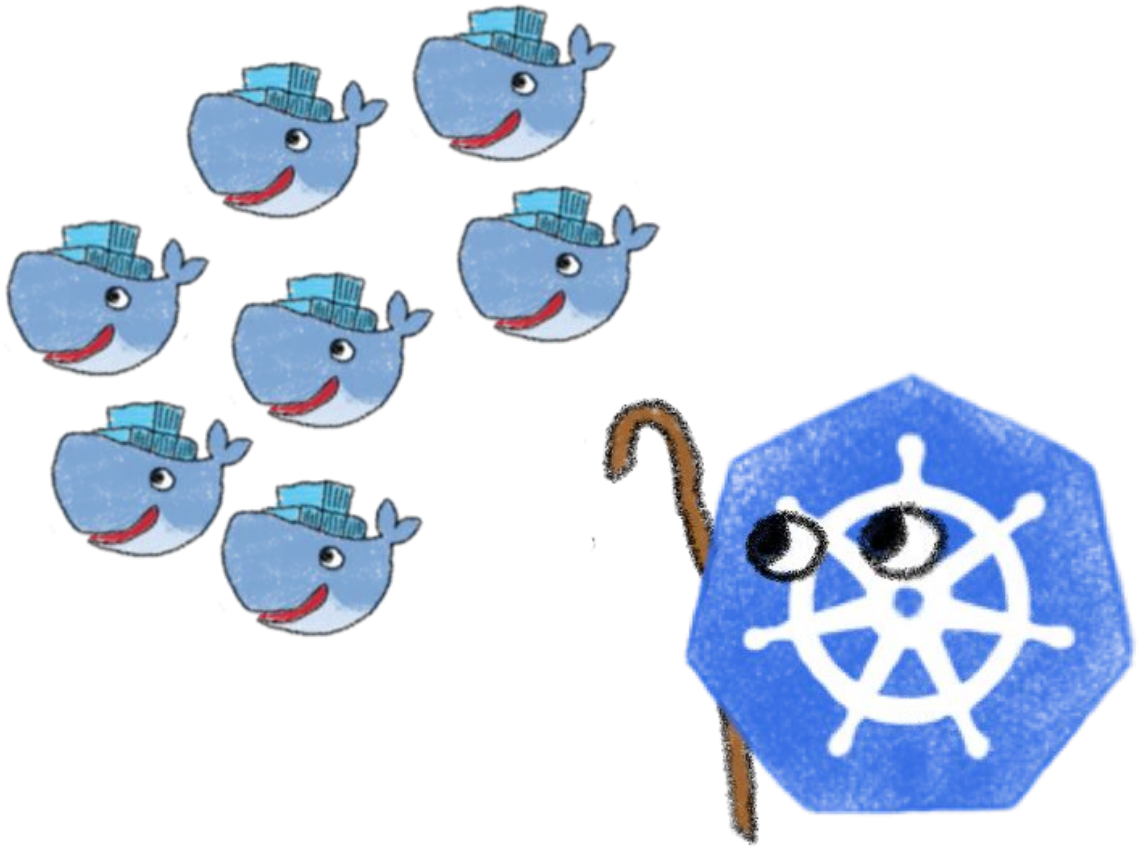


Are you sure you want to operate them by hand?

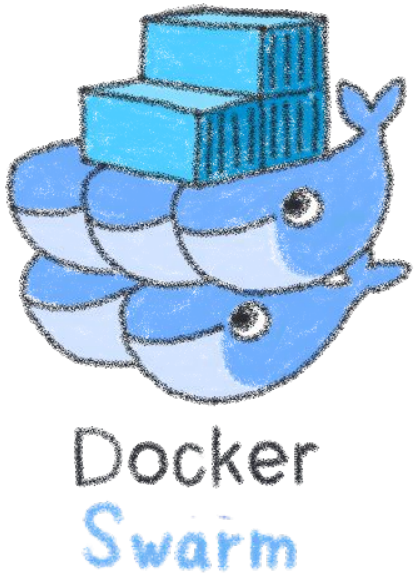
Kubernetes: a full orchestrator

Takes care of:

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



Not the only orchestrator



kubernetes

HashiCorp
Nomad

MESOS

But the most popular one…

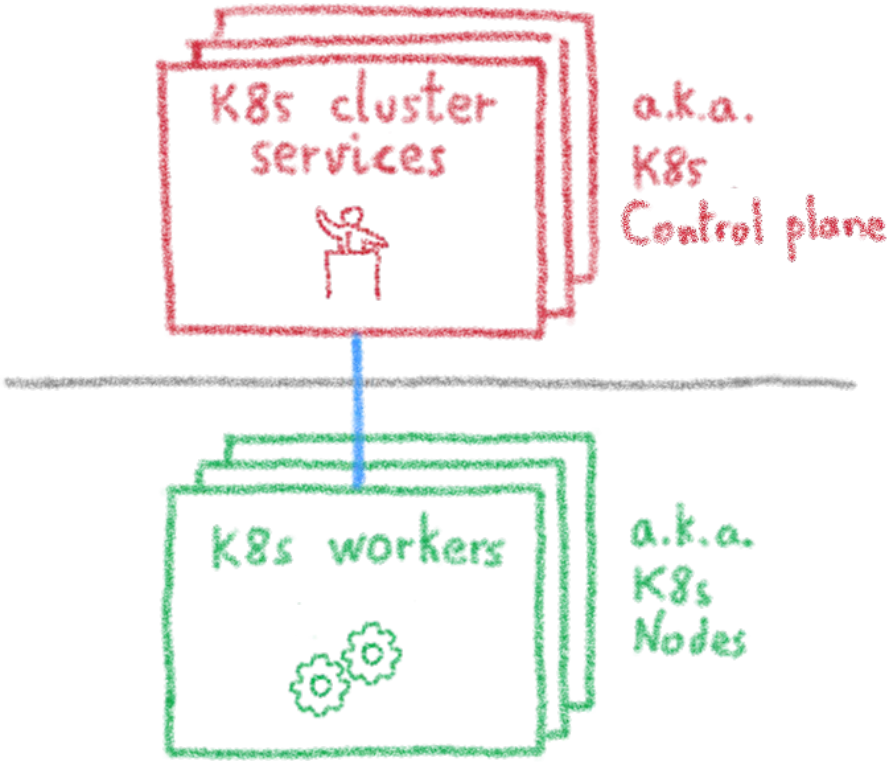
Introduction

Why Kubernetes?

Containers

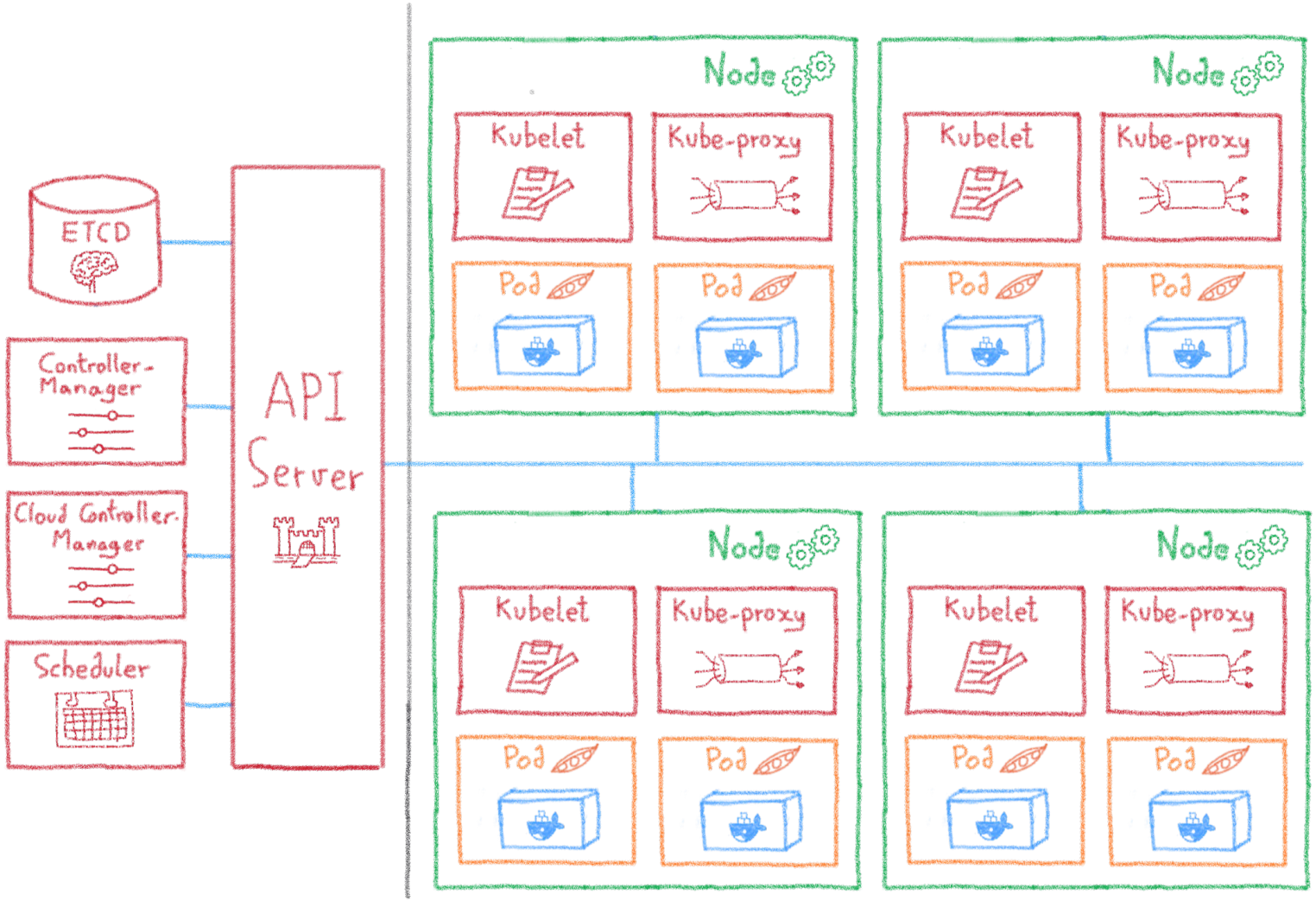
What is Kubernetes?

An open-source container orchestration system

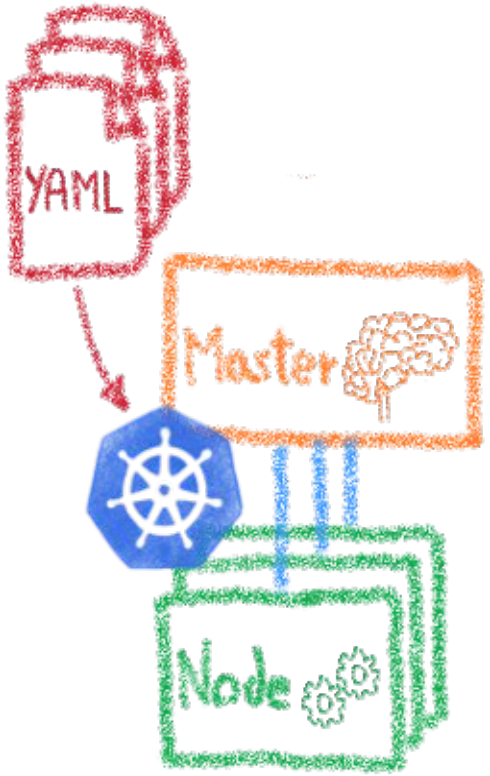


A cluster of instances

Kubernetes cluster: more details



Desired State Management



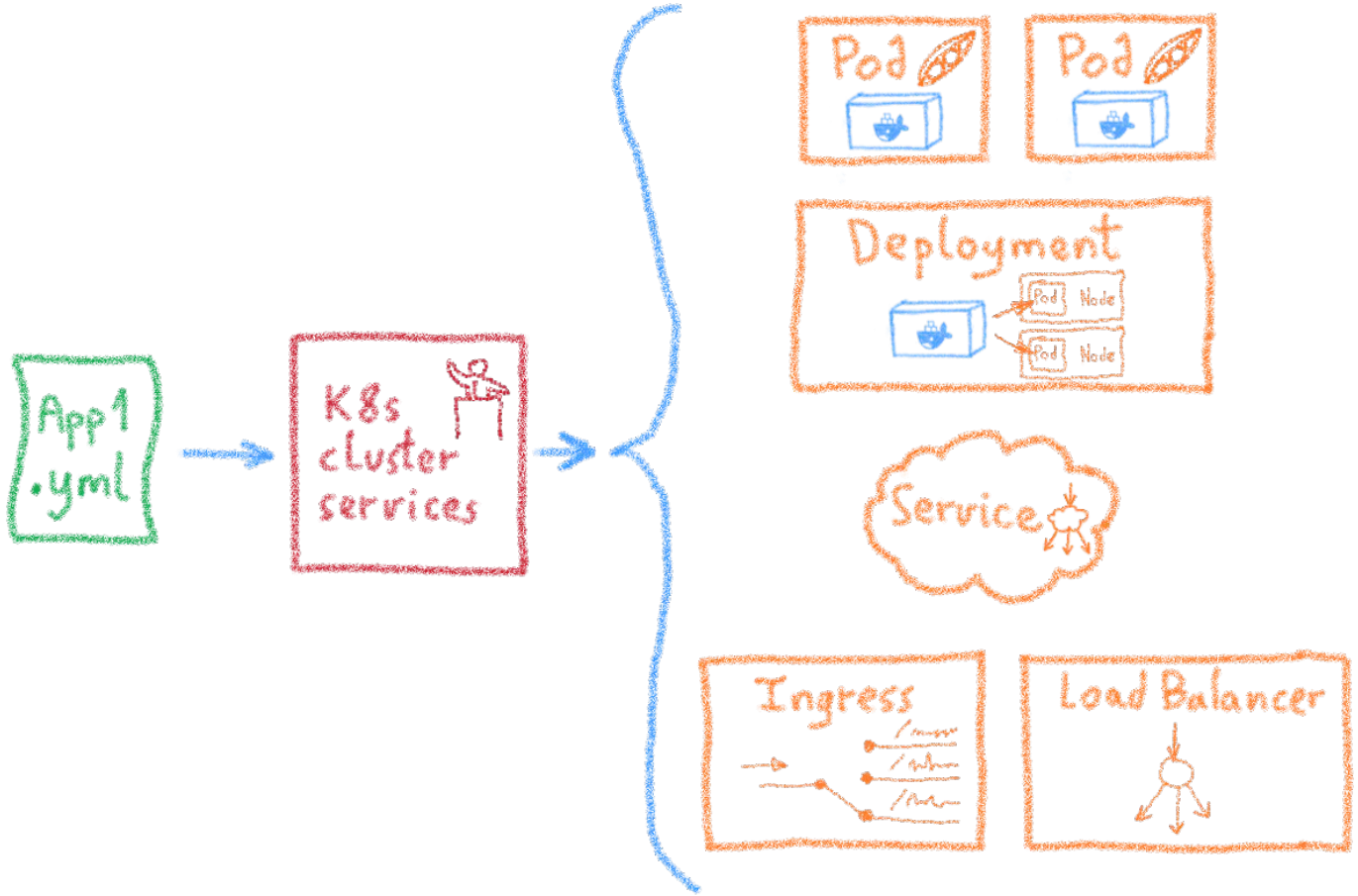
Manifest files:

Text files in YAML format

High-level description of the target architecture

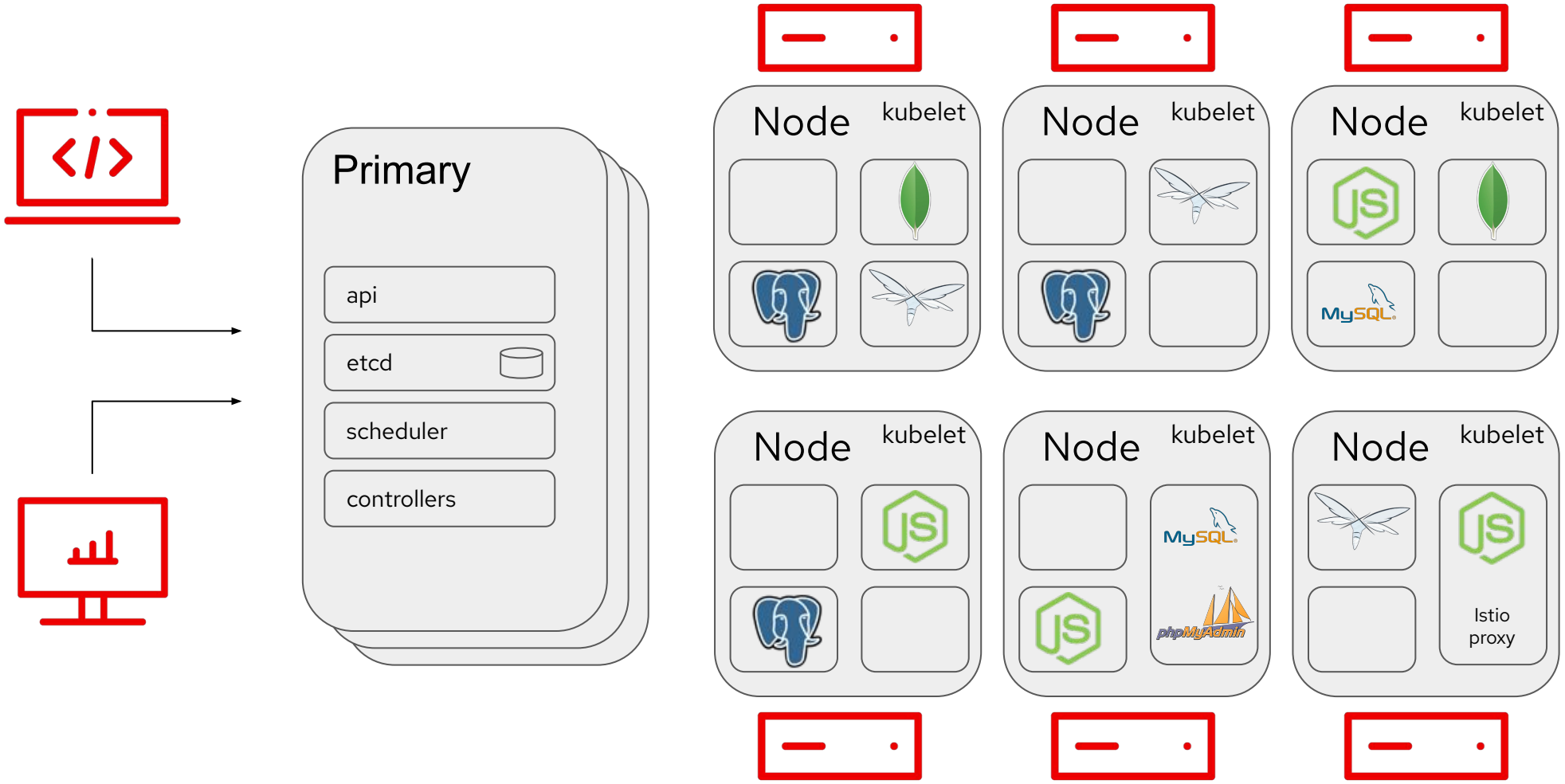
Declarative infrastructure

Desired State Management

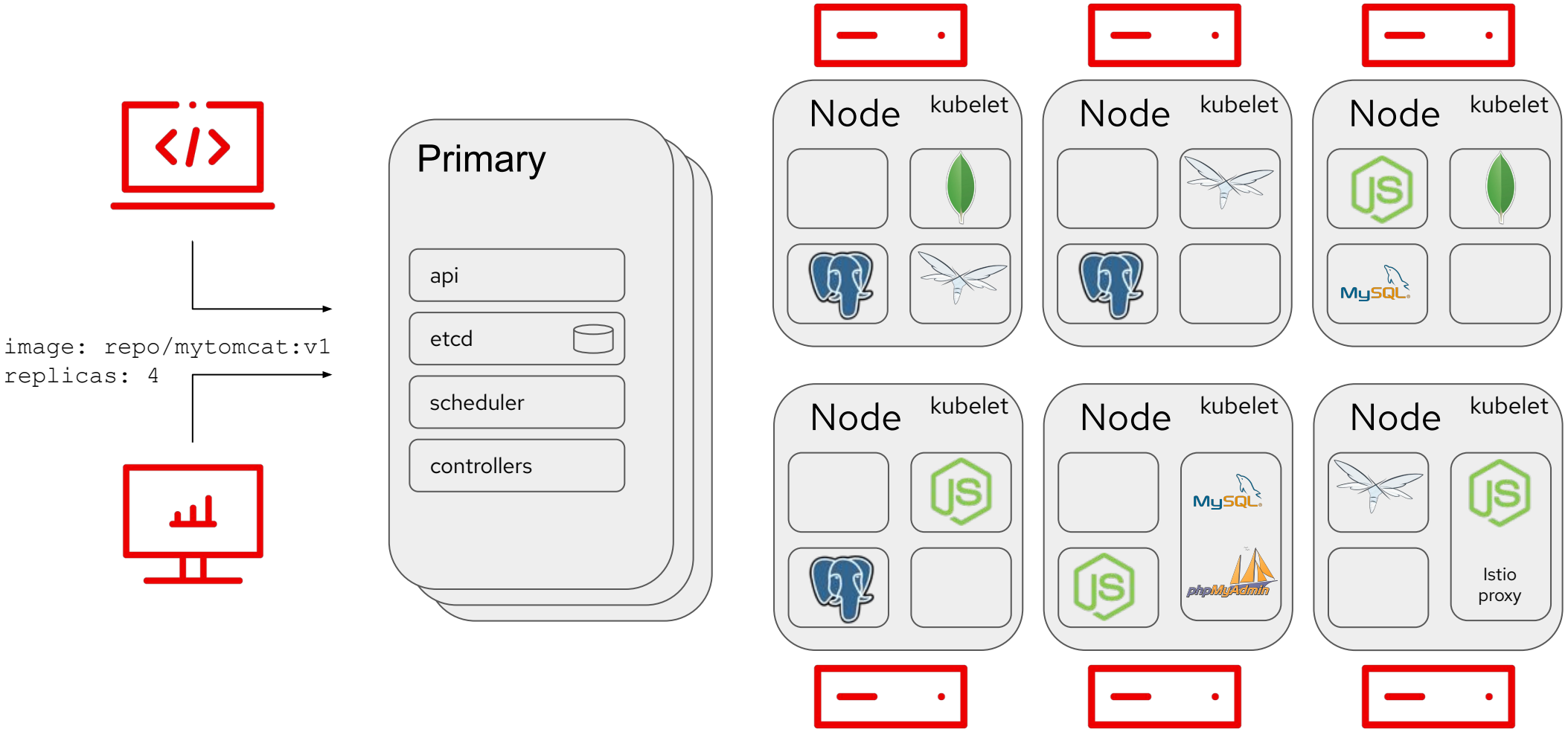


Let's begin with 5 objects

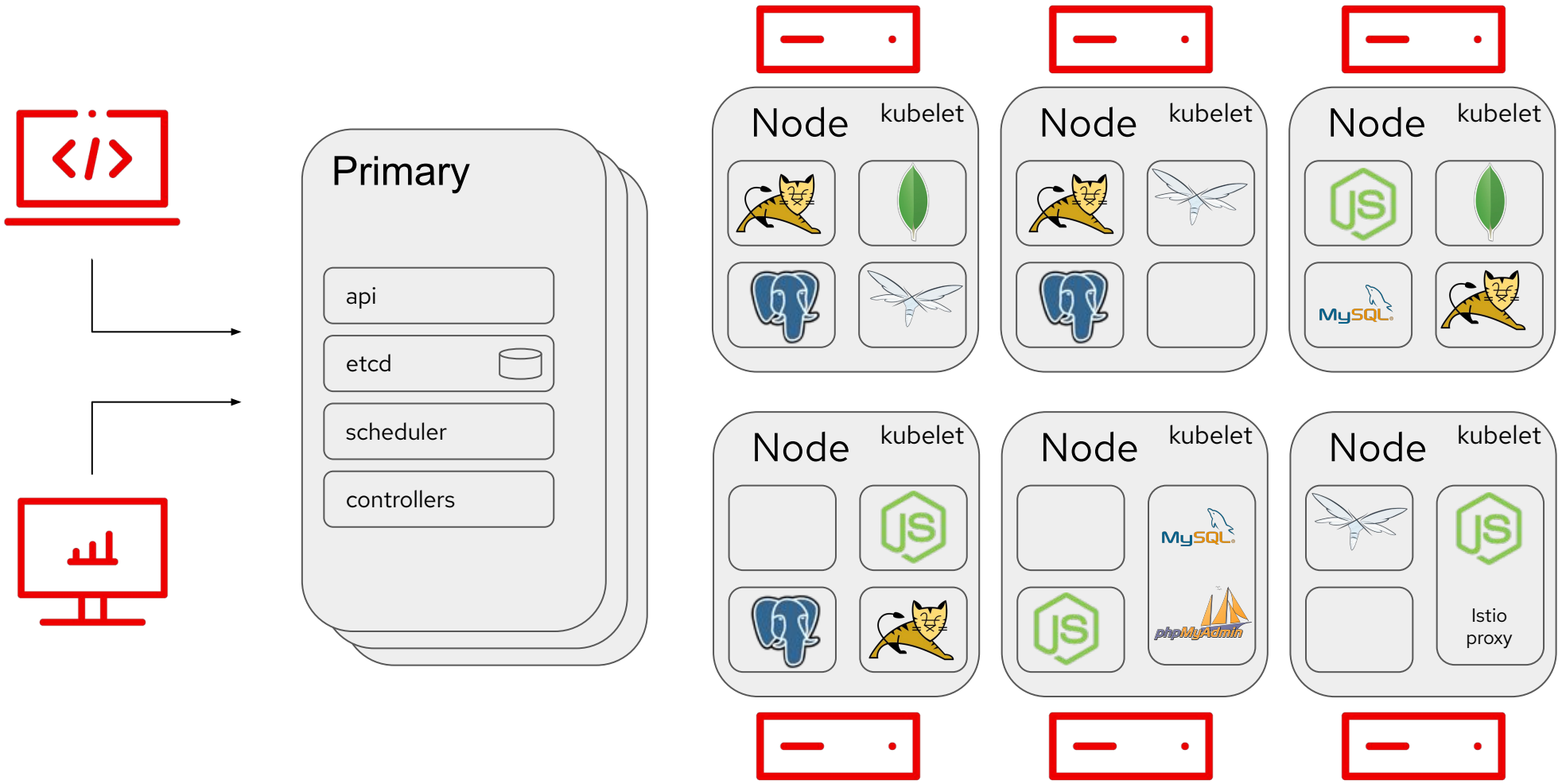
Kubernetes Cluster - Nodes



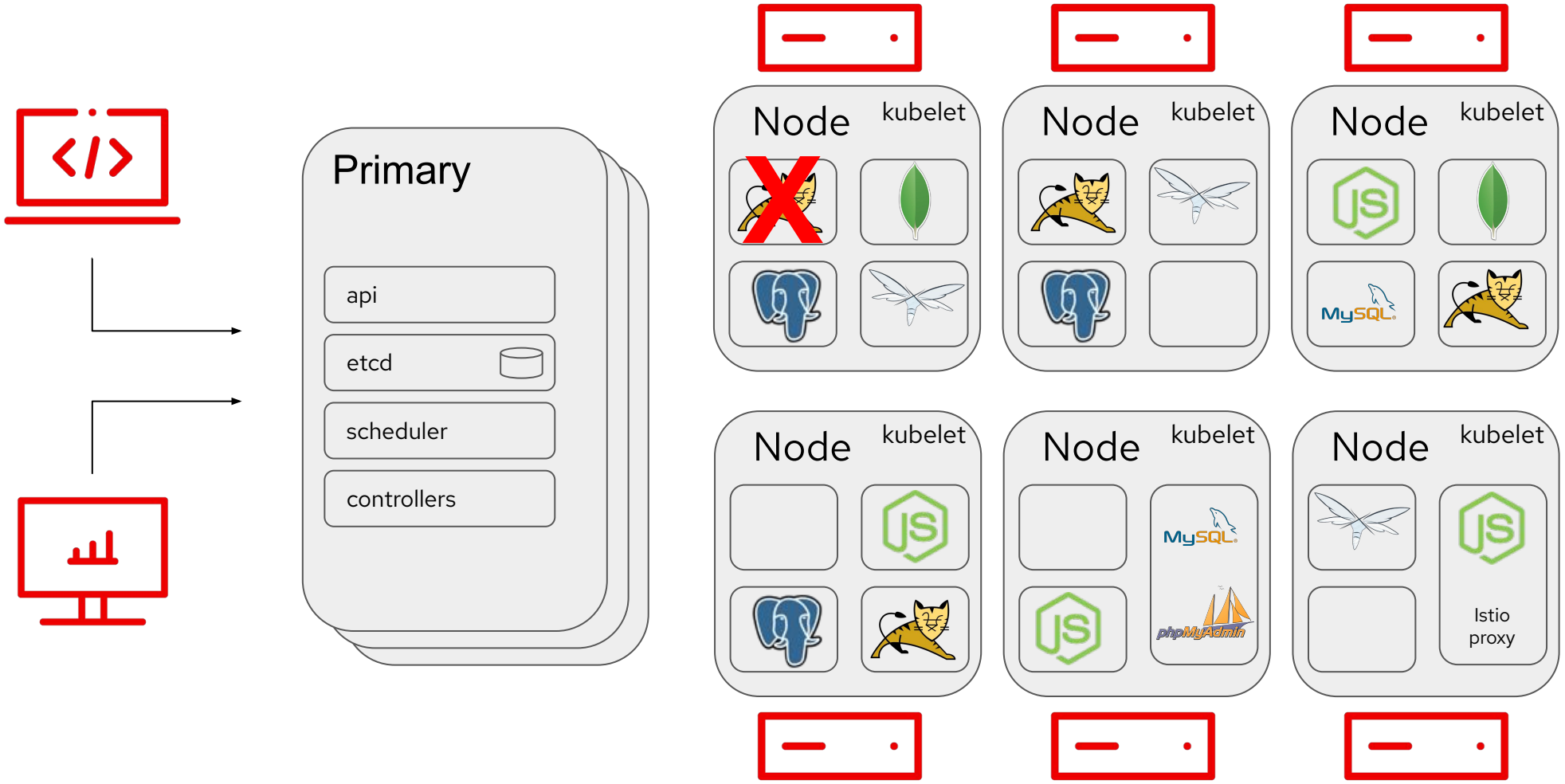
Kubernetes Cluster - Declarative API



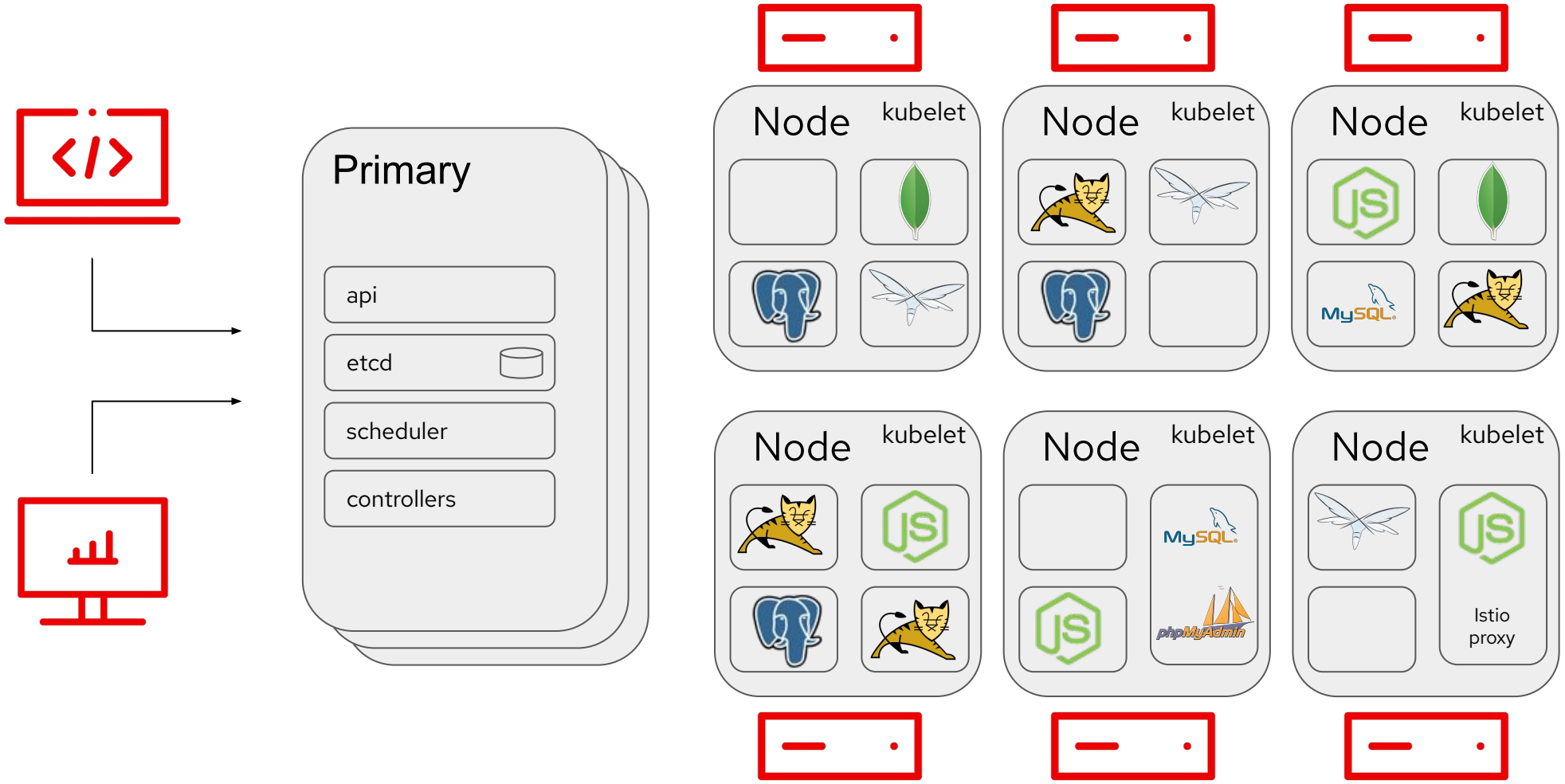
Kubernetes Cluster - 4 Tomcat Instances



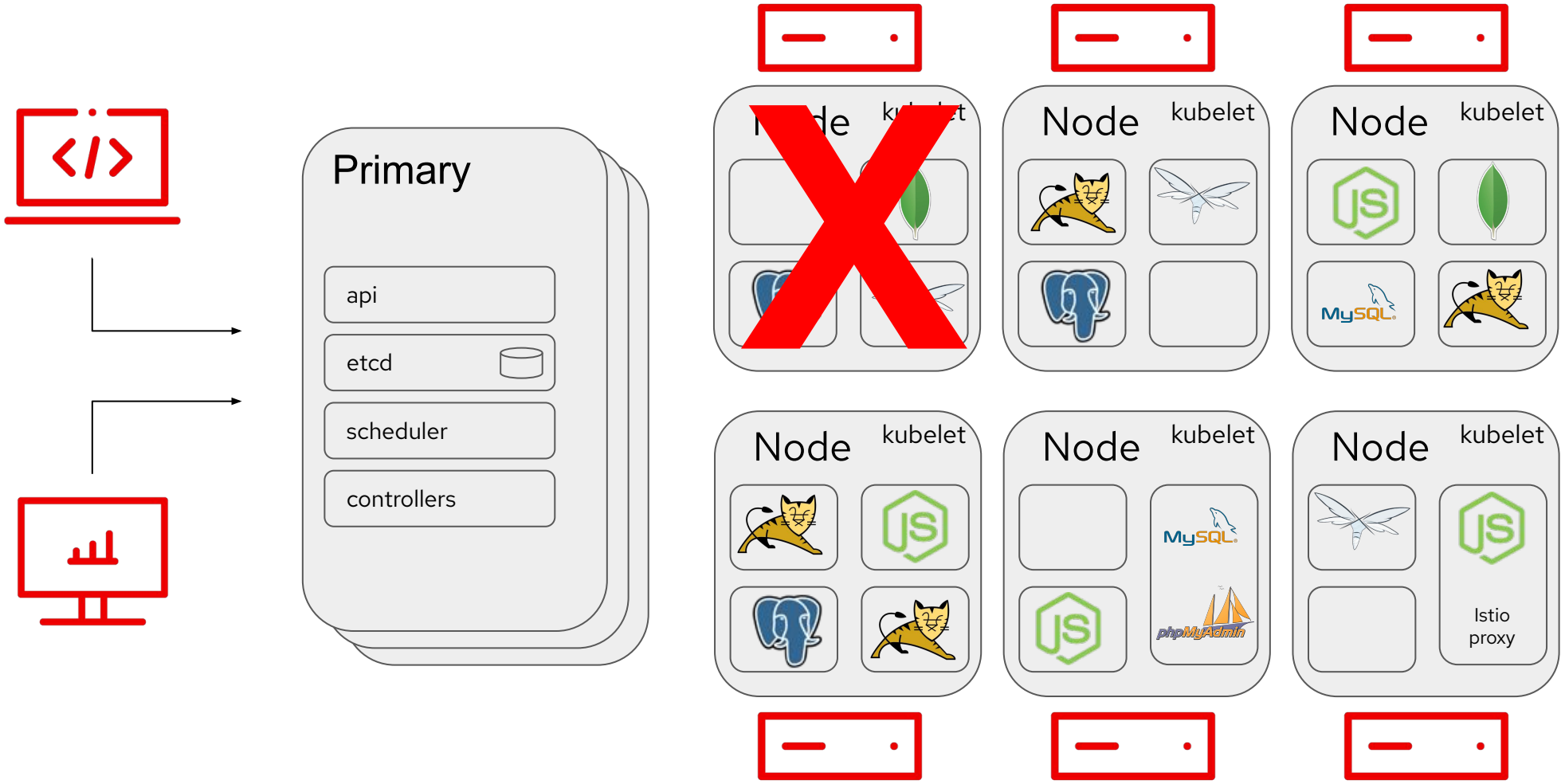
Kubernetes Cluster - Pod Failure



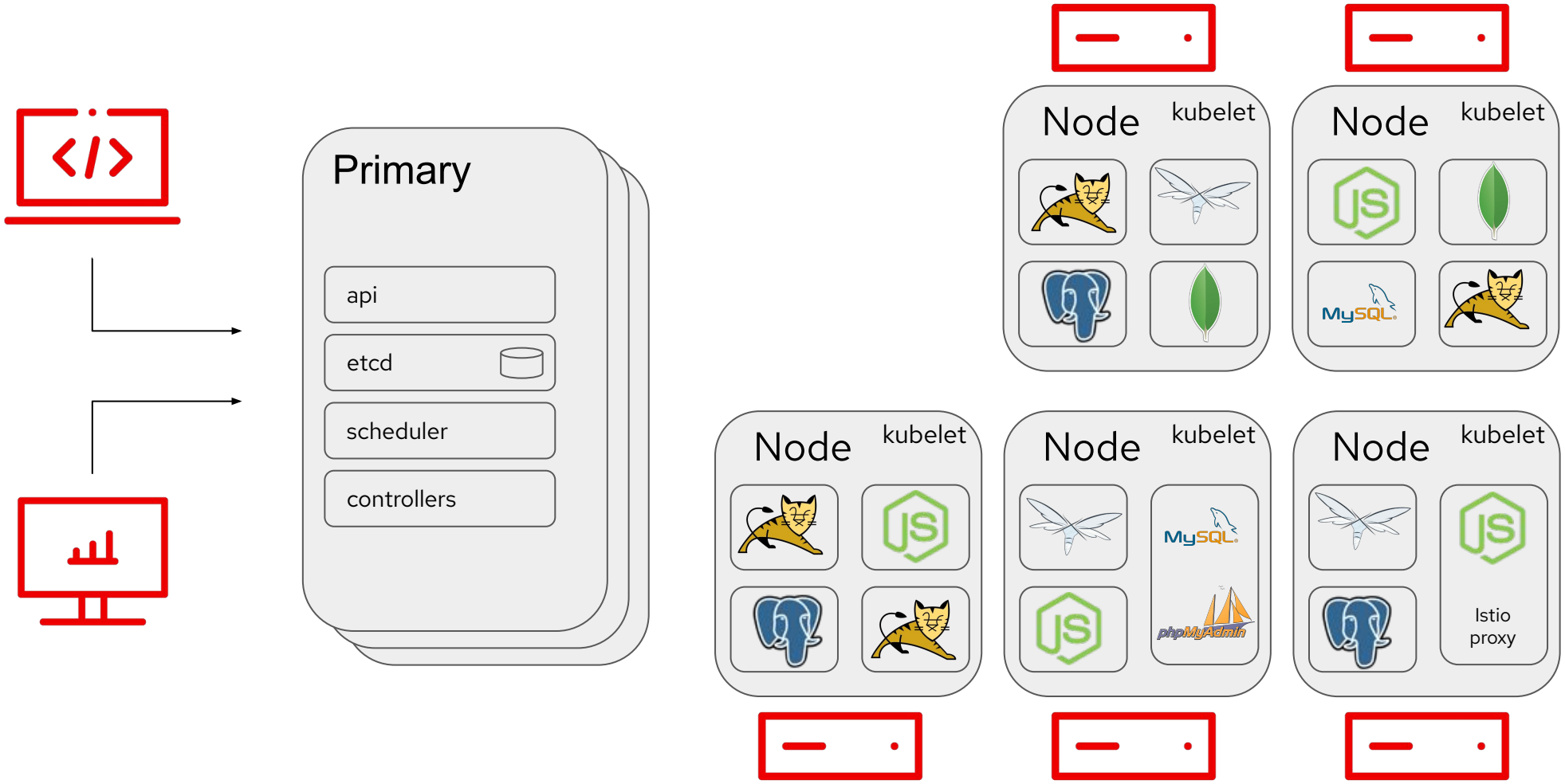
Kubernetes Cluster - Recovery



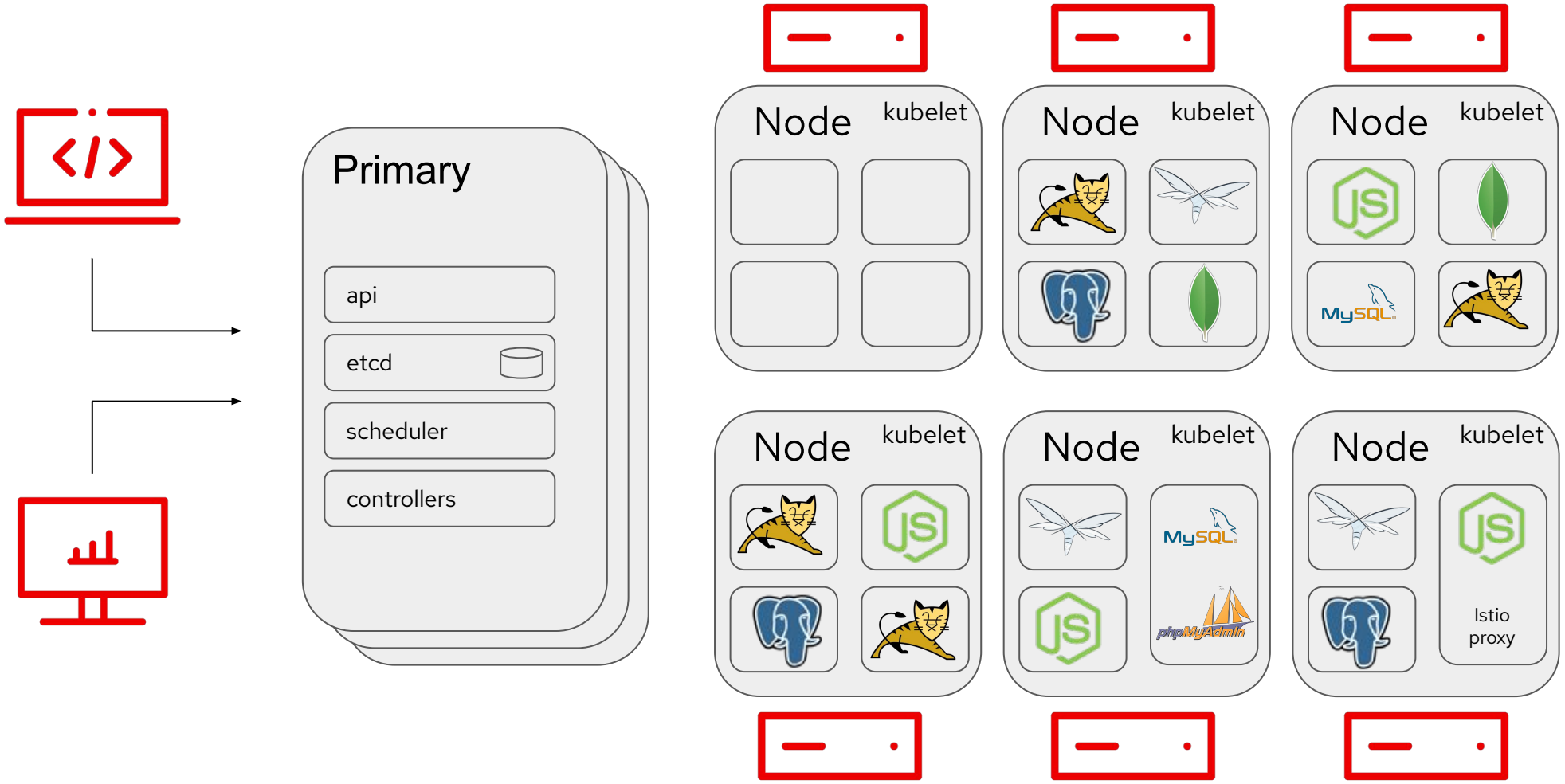
Kubernetes Cluster - Node failure



Kubernetes Cluster - Pods replaced



Kubernetes Cluster - New node

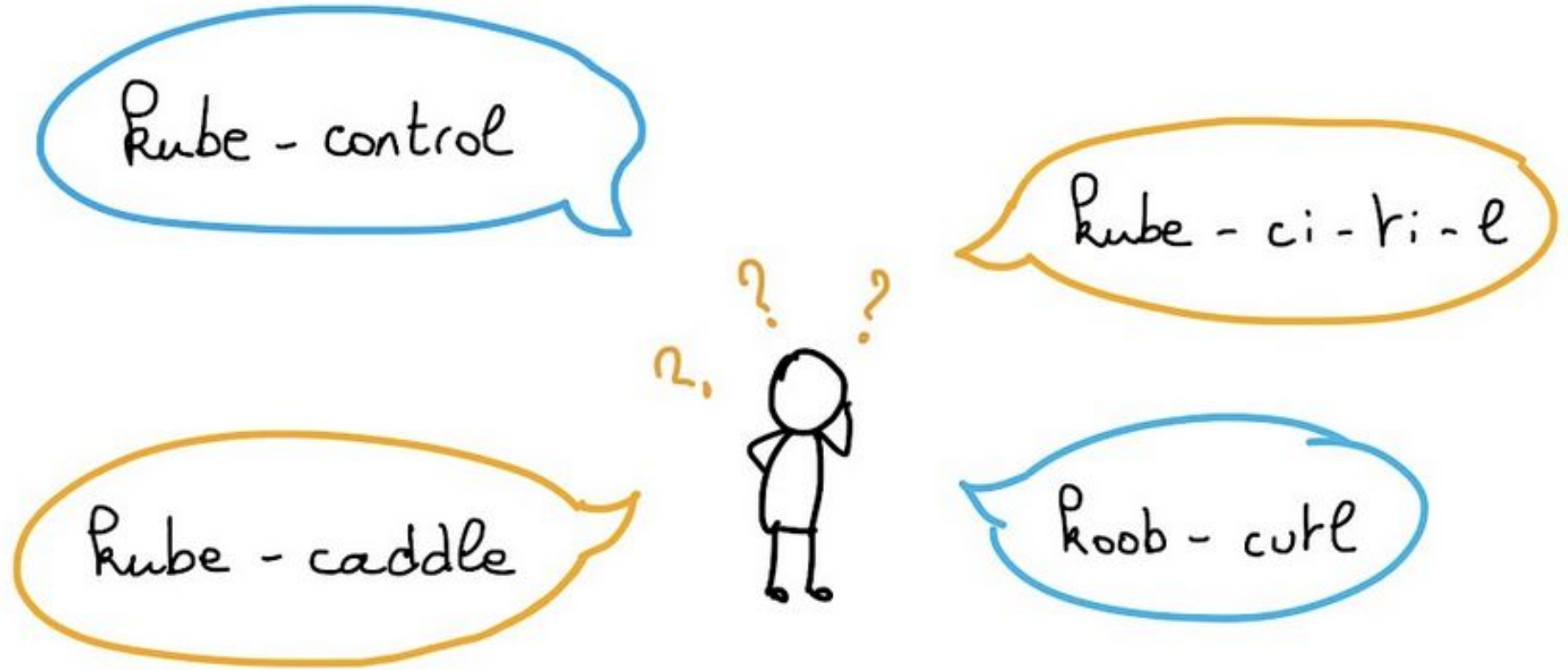


1- Dive into K8s building blocks

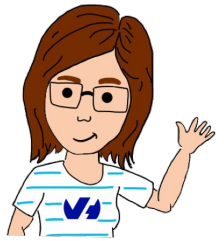
Playing with Kubectl

YAML

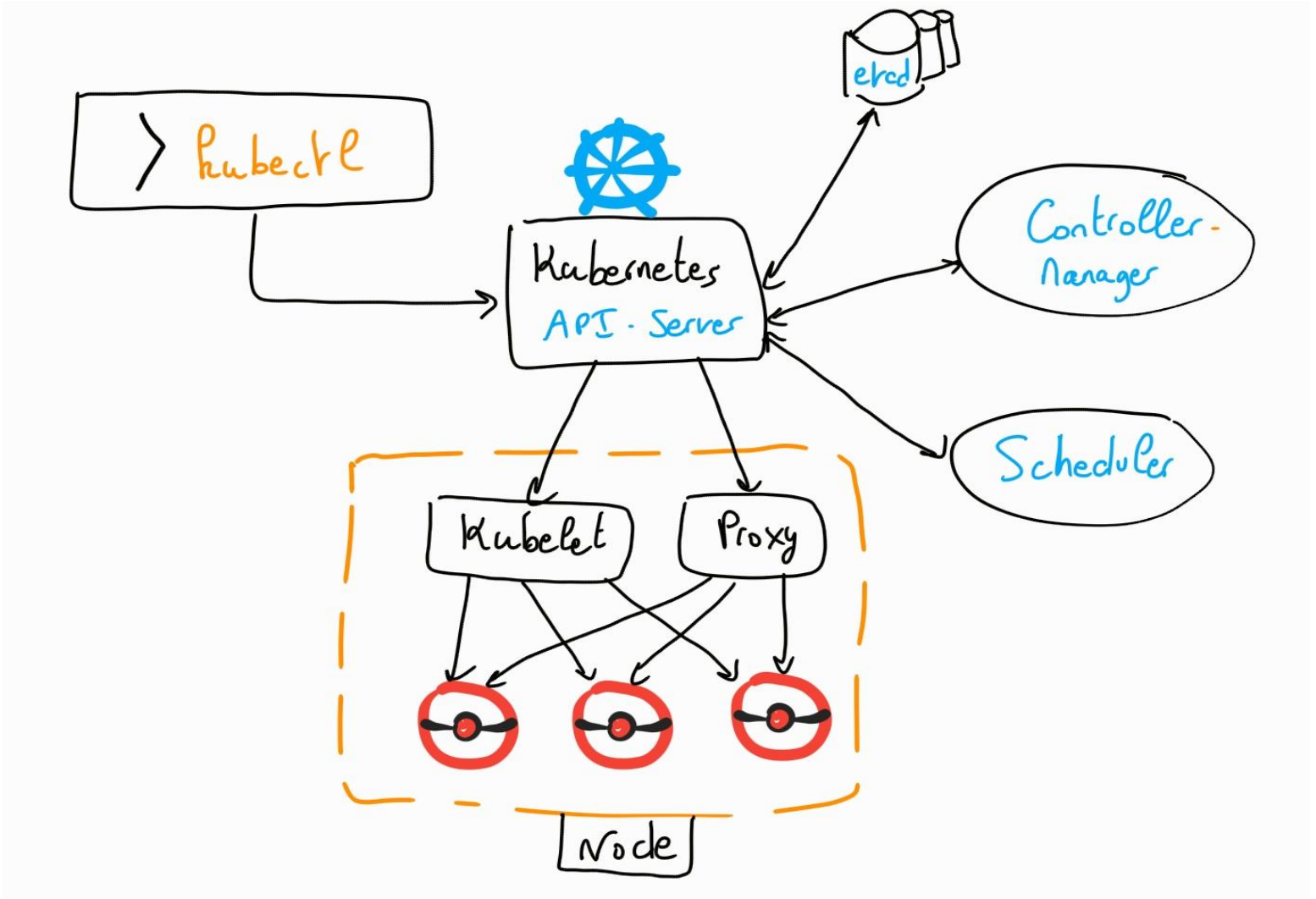
Kubectl > pronunciation fight

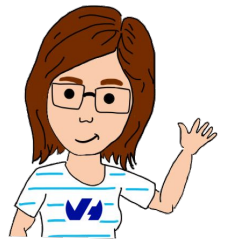


Pronounce `kubectl` as you want 😅



Kubectl > kubernetes tool/cli





Démo Kubectl

1- Dive into K8s building blocks

Playing with Kubectl

YAML

Kubernetes

Kubernetes is a distributed and structured YAML database

CRUD, structured and typed objects: Resources

Resources live in Namespaces

https://asciinema.org/a/lfxttSBoSoVH9hkS4lOxzuGdk

Create a Resource object

speaker.yaml

```yaml
apiVersion: "stable.world.com/v1"
kind: Speaker
metadata:
 name: horacio
spec:
 name: "Horacio"
 title: "DevRel at OVH Cloud"
 action: "speak"
```

Execute

```
$ kubectl apply -f speaker.yaml
$ kubectl get Speaker
```

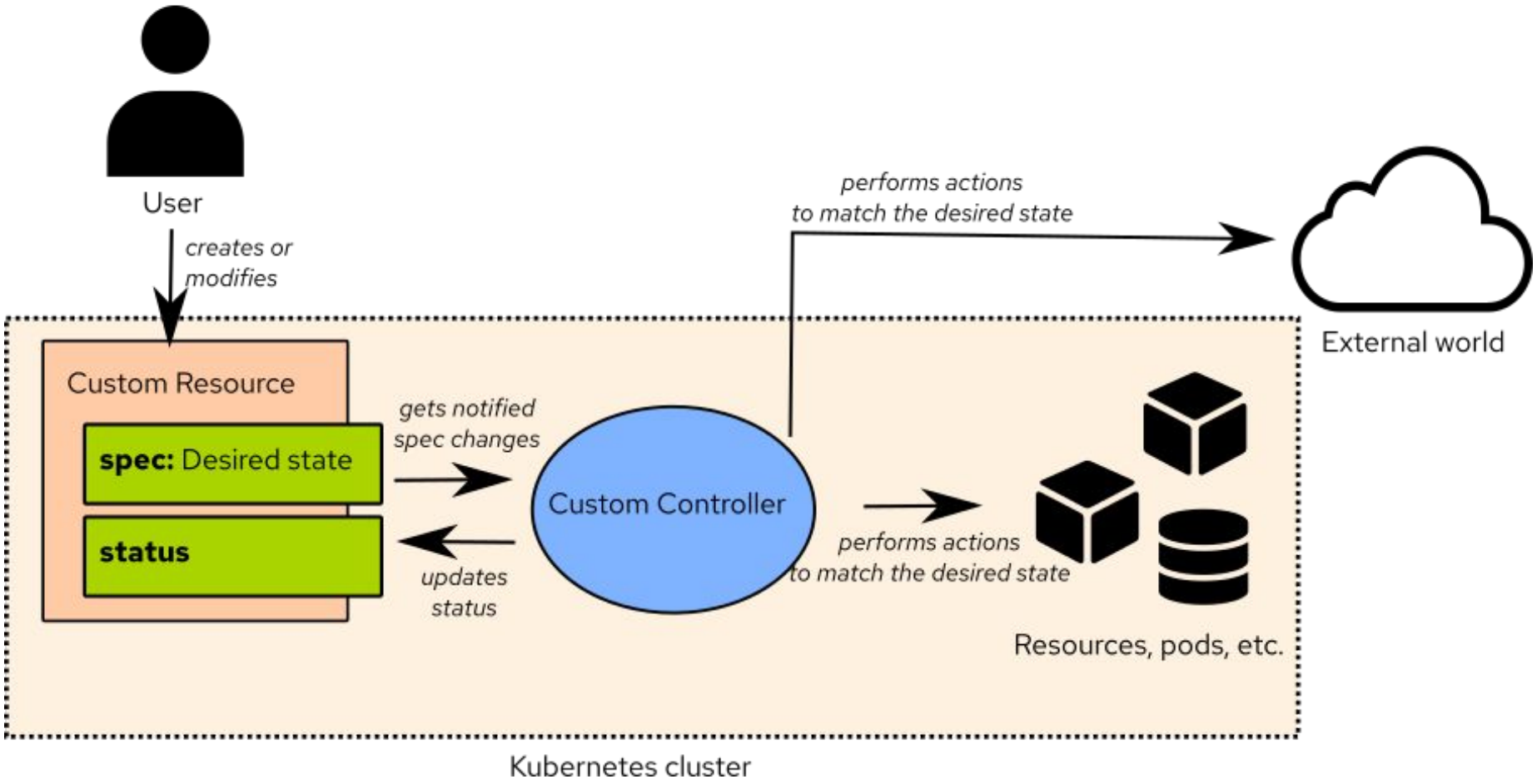
Kubernetes

Kubernetes is a distributed and structured YAML database

Controllers that do the job

- Listening to Resources Create/Update/Delete events: the user requirements
- Perform to match the user requirements

Kubernetes controller



https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

Kubernetes

Kubernetes is a distributed and structured YAML database

- By default, a set of Resources and Controllers to manage a cluster of machines

Pod

Container(s) sharing network addressing/volumes, etc.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  containers:
  - name: hellotomcat-container
    image: quay.io/sunix/hello-tomcat
    imagePullPolicy: IfNotPresent
    ports:
    - containerPort: 8080
```

https://asciinema.org/a/EeeNkoQ2eJ76Twx2S0sCybTzz

Deployment

Deploy and manage identical pods

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
name: hellotomcat
labels:
  app: hellotomcat
spec:
replicas: 2
selector:
  matchLabels:
    app: hellotomcat
template:
  metadata:
    labels:
      app: hellotomcat
  spec:
    containers:
    - name: hellotomcat-container
      image: quay.io/sunix/hello-tomcat
      ports:
      - containerPort: 8080
      imagePullPolicy: IfNotPresent
```

https://asciinema.org/a/EsaRue6eDKWyvRCHmRKxIfydI

Service

Let the pods communicates in the cluster or outside

```yaml
apiVersion: v1
kind: Service
metadata:
  name: hellotomcat-service
spec:
  type: NodePort
  selector:
    app: hellotomcat
  ports:
  - protocol: TCP
    port: 8080
    targetPort: 8080
```

Ingress

Manage the paths and domain name redirectio

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: hellotomcat-ingress
  labels:
    app: hellotomcat
spec:
  rules:
  - host: 192.168.49.2.nip.io
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: hellotomcat-service
            port:
              number: 8080
```

https://asciinema.org/a/PpW6P3EftEUWb13UOvoBK6wOW

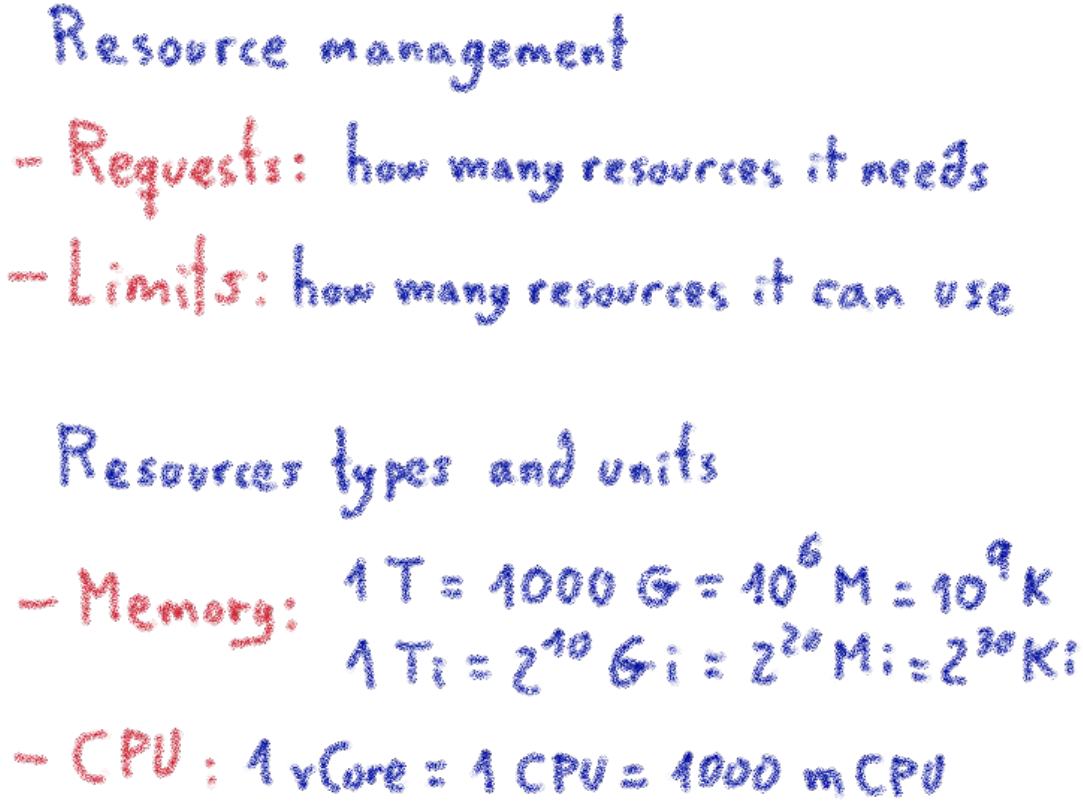
2 - Being a good cloud native citizen

Requests and limits

Health probes

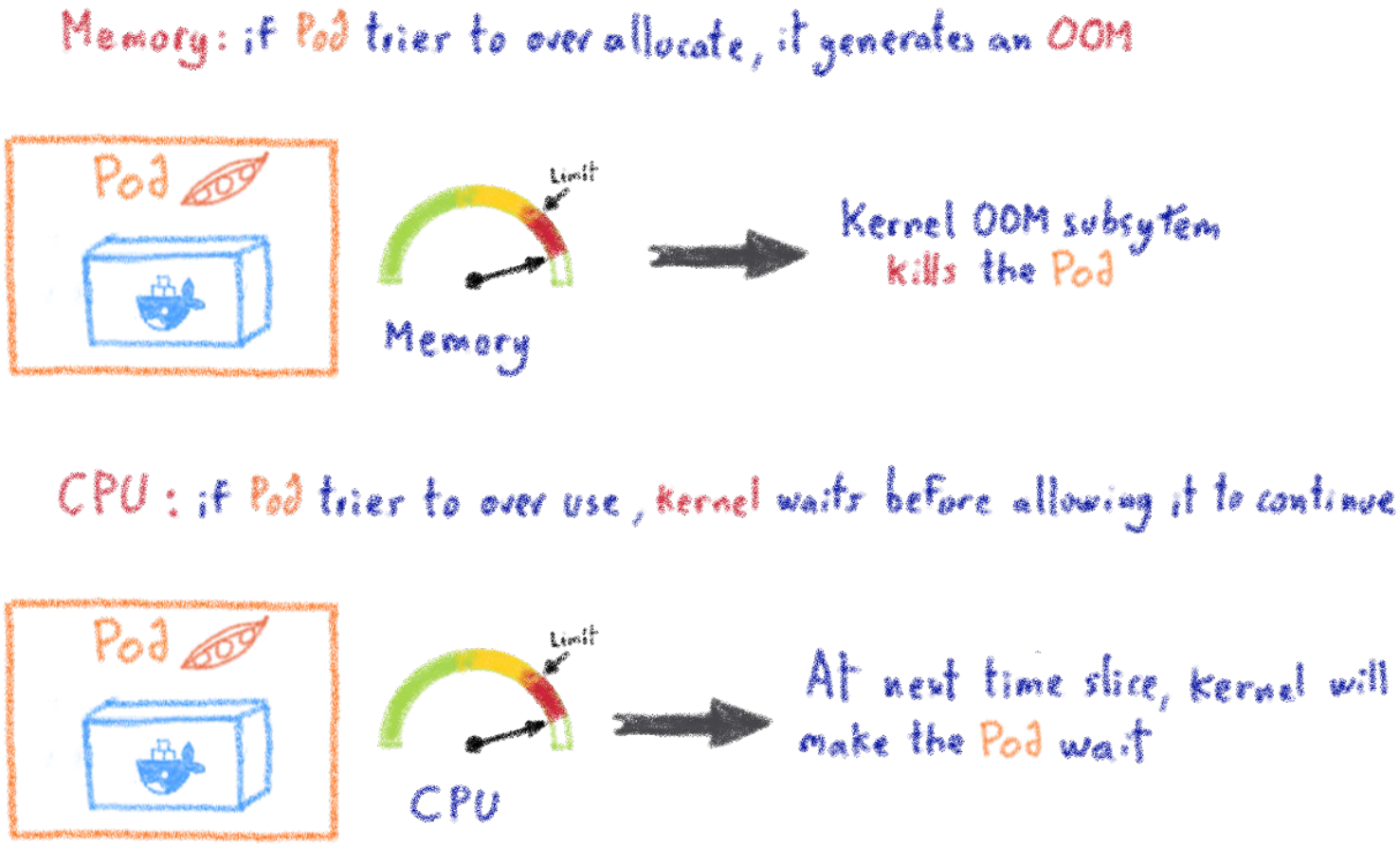
ConfigMap and Secrets

Resource management



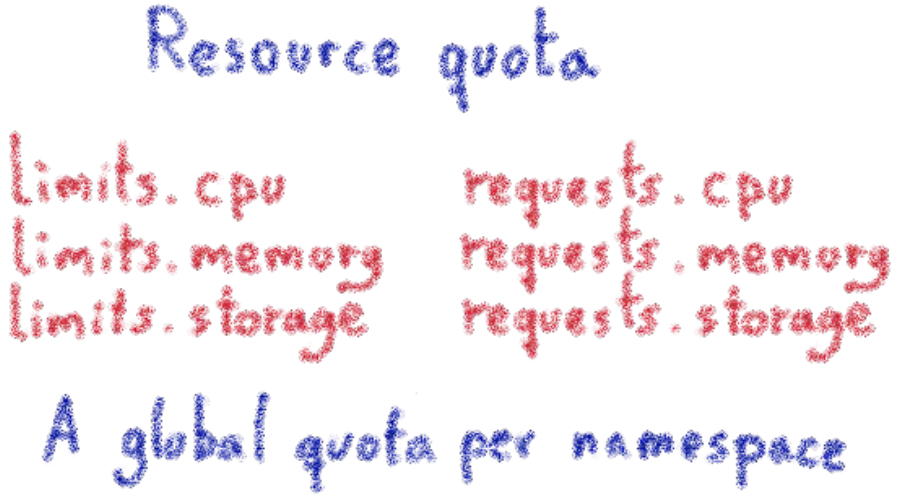

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: frontend
spec:
 containers:
 - name: app
   image: images.my-company.example/app
   resources:
     requests:
       memory: "64Mi"
       cpu: "250m"
     limits:
       memory: "128Mi"
       cpu: "500m"
```

What if a pod uses too many resources?



CPU is compressible, memory is incompressible

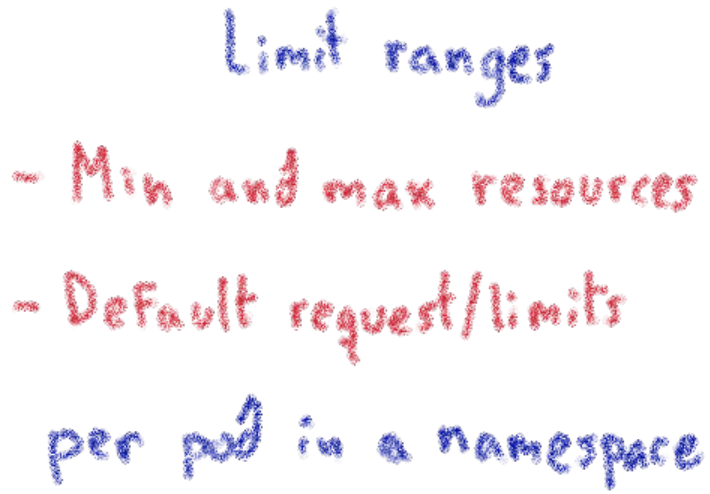
Resource quota




```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




```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%
```

Démo Requests & Limits

2 - Being a good cloud native citizen

Requests and limits

Health probes

ConfigMap and Secrets

Readiness probe

Tell people you're ready

Readiness probe

```yaml
apiVersion: v1
kind: Pod
metadata:
 labels:
   test: readiness
 name: readiness-exec
spec:
 containers:
 - name: readiness
   image: organisation/readiness
   readinessProbe:
     httpGet:
       path: /ready
       port: 80
     initialDelaySeconds: 10
     periodSeconds: 3
```

Readiness probe – Telling the cluster that the pod is ready

Liveness probe

Tell people you're alive

Liveness probe

```yaml
apiVersion: v1
kind: Pod
metadata:
 labels:
   test: liveness
 name: liveness-exec
spec:
 containers:
 - name: liveness
   image: organisation/liveness
   livenessProbe:
     exec:
       command:
       - cat
       - /tmp/alive
     initialDelaySeconds: 10
     periodSeconds: 5
     timeoutSeconds: 2
```

Liveness probe - Telling the cluster that the pod is alive

Startup probe

Tell people you have started

Startup probe

```yaml
apiVersion: v1
kind: Pod
metadata:
 labels:
   test: starting
 name: starting-exec
spec:
 containers:
 - name: starting
   image: organisation/starting
   startupProbe:
     httpGet:
       path: /ready
       port: 80
     periodSeconds: 3
     failureThreshold: 24
```

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

Démo Requests & Limits

2 - Being a good cloud native citizen

Requests and limits

Health probes

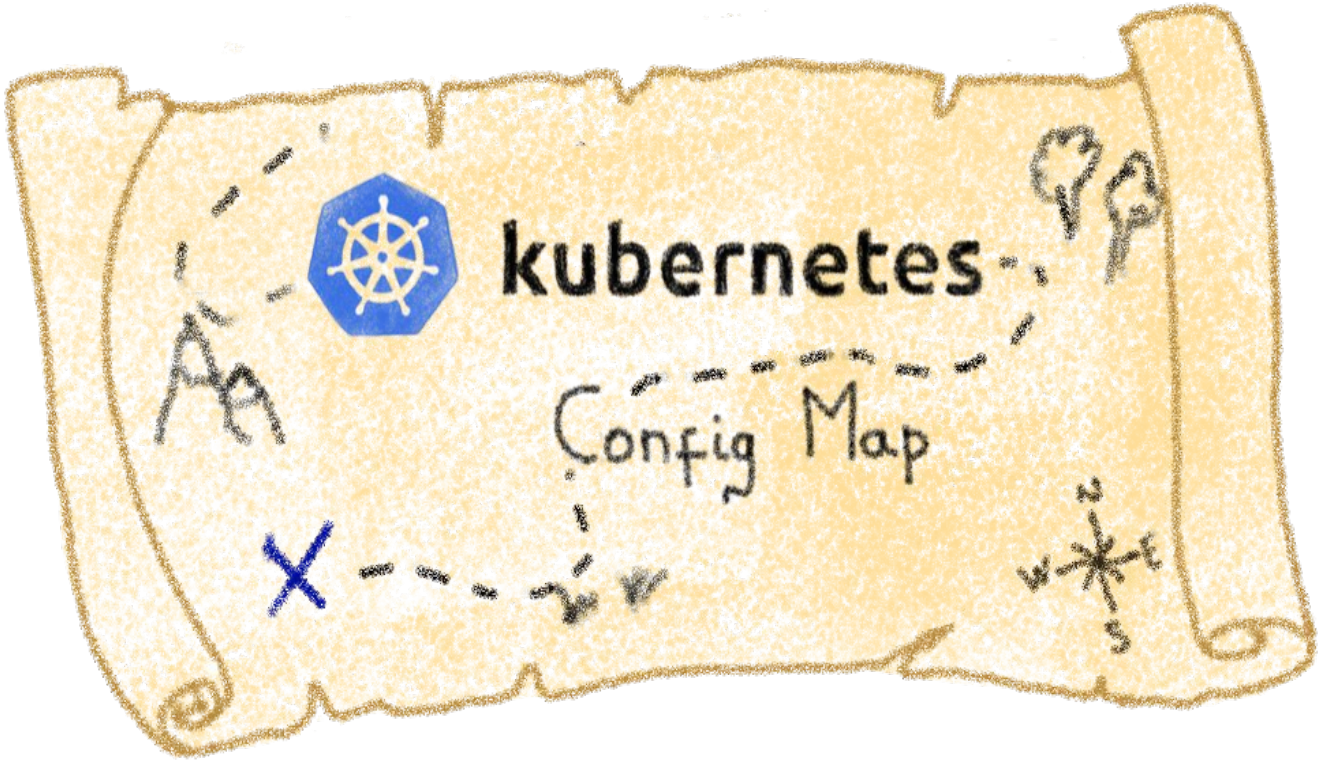
ConfigMap and Secrets

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

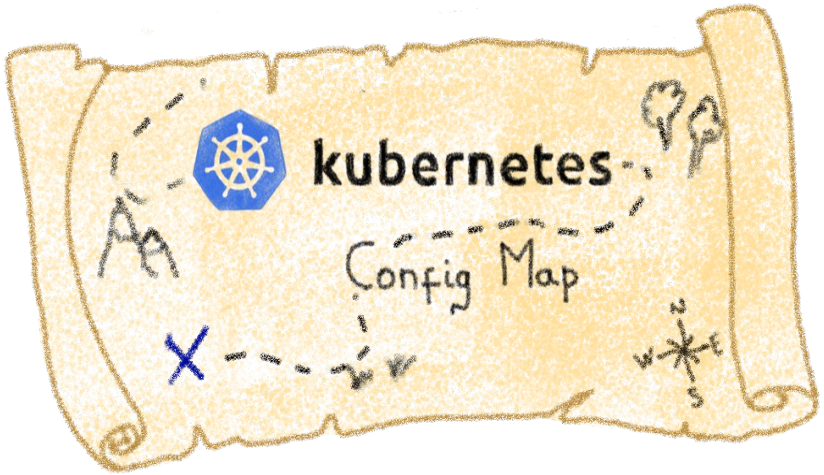


Storing configuration for other objects to use

Describing a Config Map

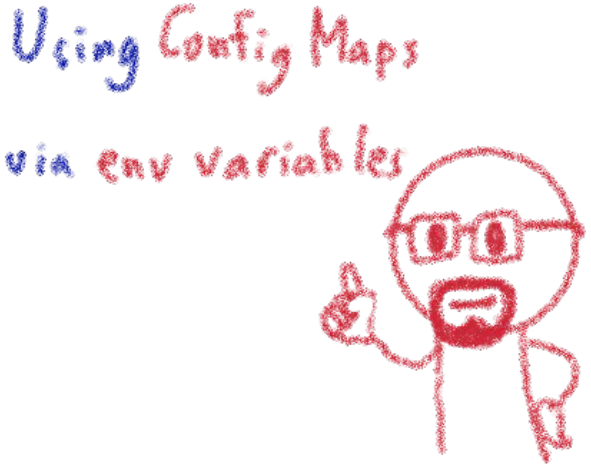
```yaml
apiVersion: v1
kind: ConfigMap
metadata:
 name: game-demo
data:
 # property-like keys; each key maps to a simple value
 player_initial_lives: "3"
 ui_properties_file_name: "user-interface.properties"

 # 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
```



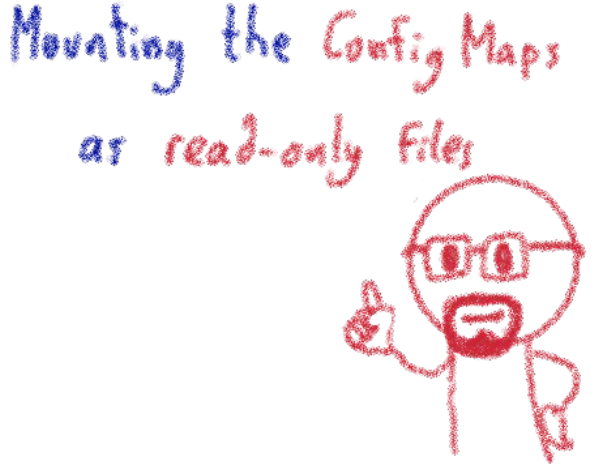
Using a Config Map in a Pod

```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"
```



Kubernetes secrets

Cambridge Dictionary

Secret

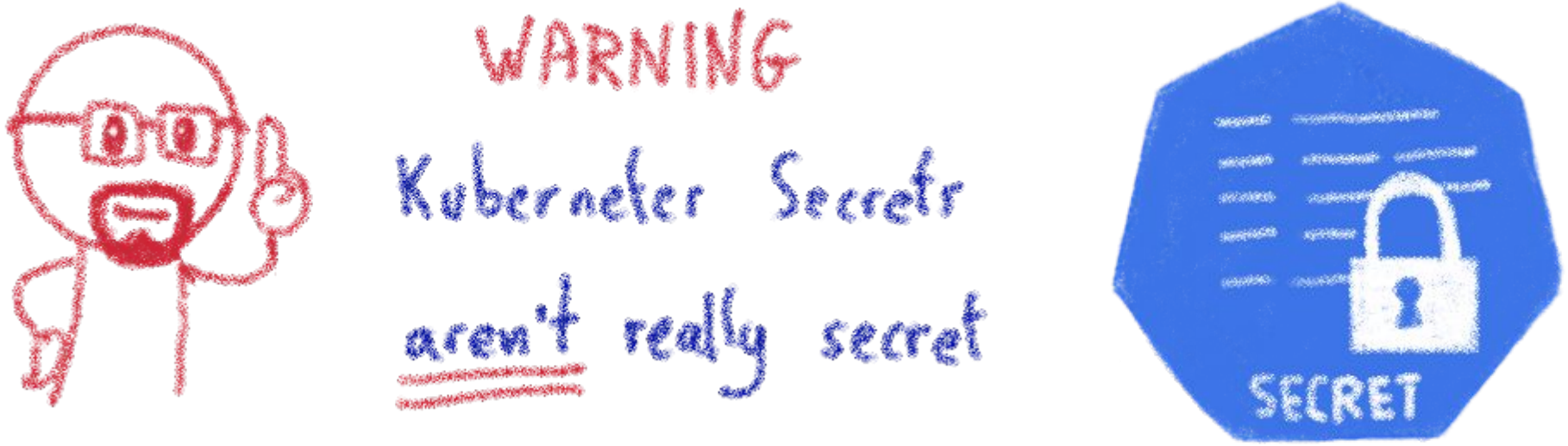
A piece of information that is only known by one person or a few people and should not be told to others.

Kubernetes secrets

Object that contains a small amount of sensitive data.

Injected as volume or environment variable.

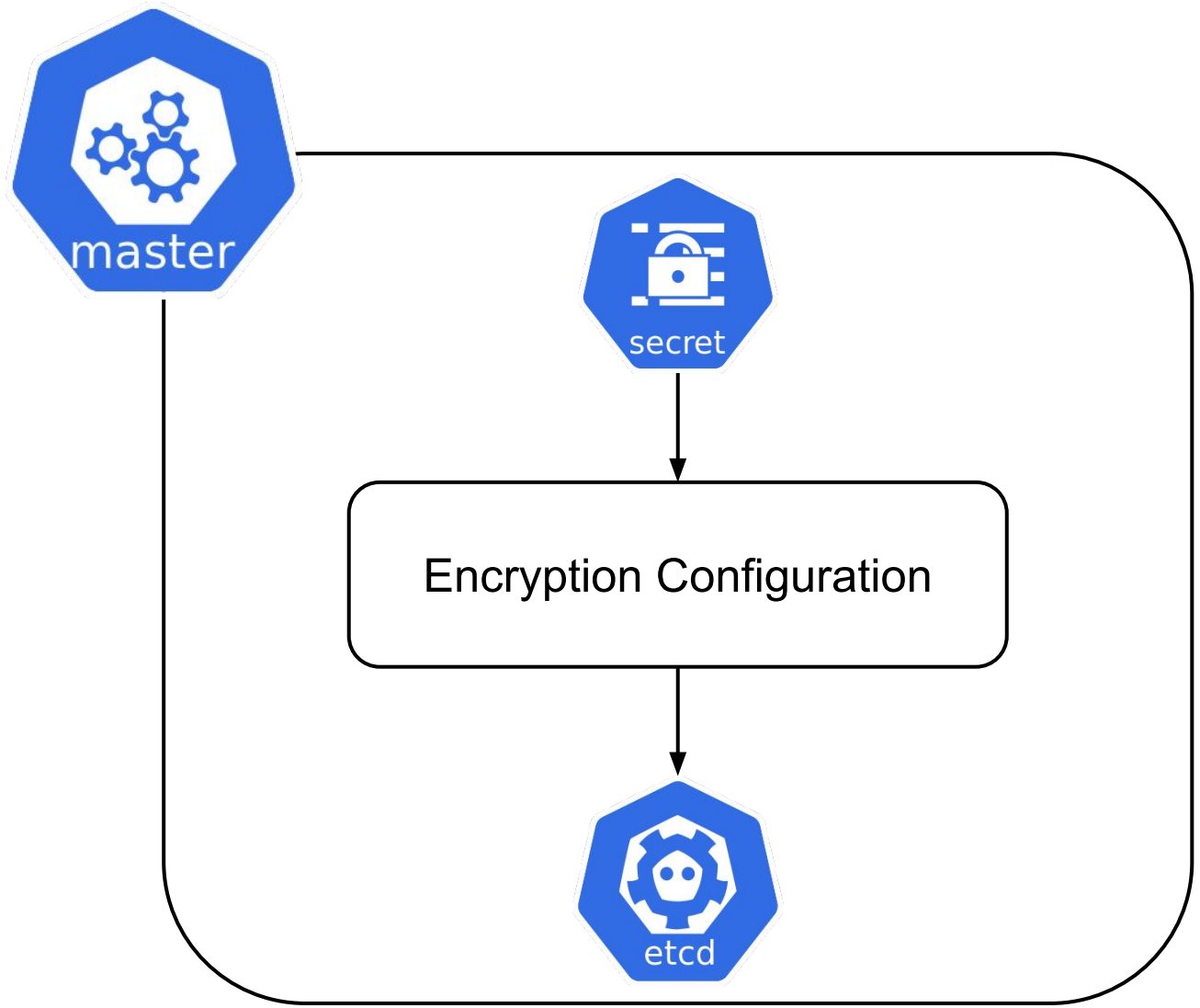
A warning on Kubernetes Secrets



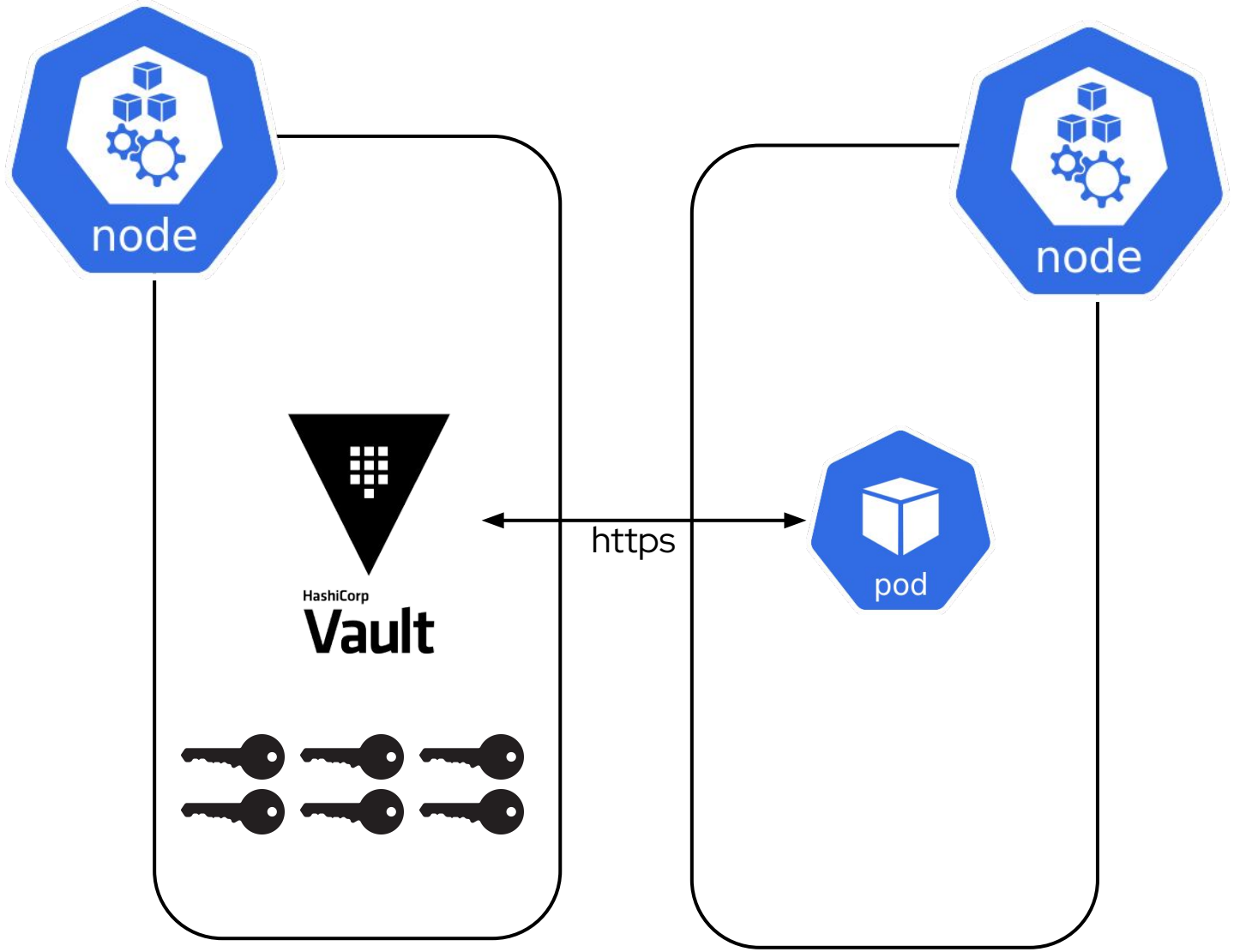
No full encryption

All YAMLs and base64

Kubernetes secrets



Vaults provide full encryption



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

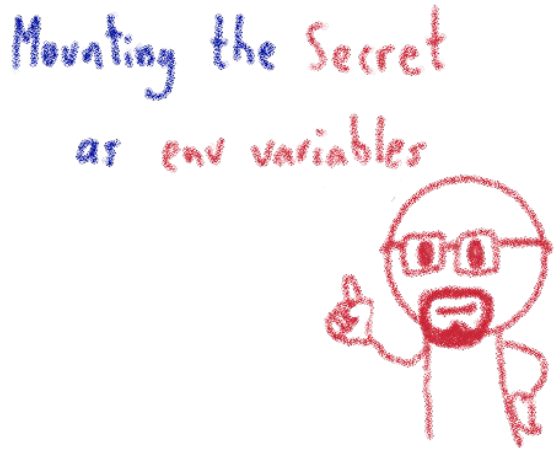
```
# View the contents of the Secret you created:
 $ kubectl get secret db-user-pass -o jsonpath='{.data}'
{"password":"UyFCXCpkJHpEc2I9","username":"YWRtaW4="}

# 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=
```

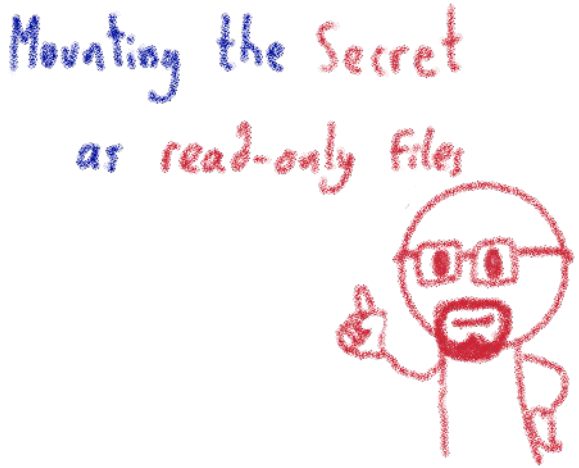
Using a Secret in a Pod

```yaml
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
```



Using a Secret in a Pod

```yaml
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.
```



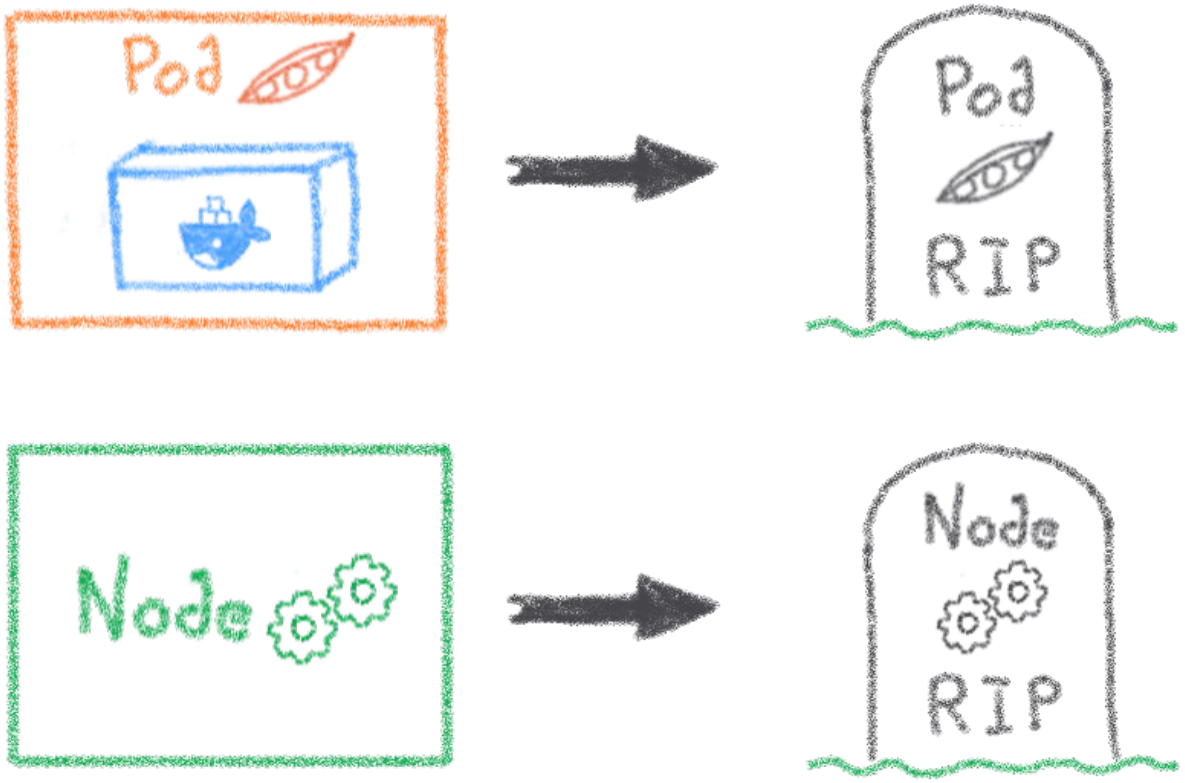
3 - Advanced K8s

Persistent Volumes

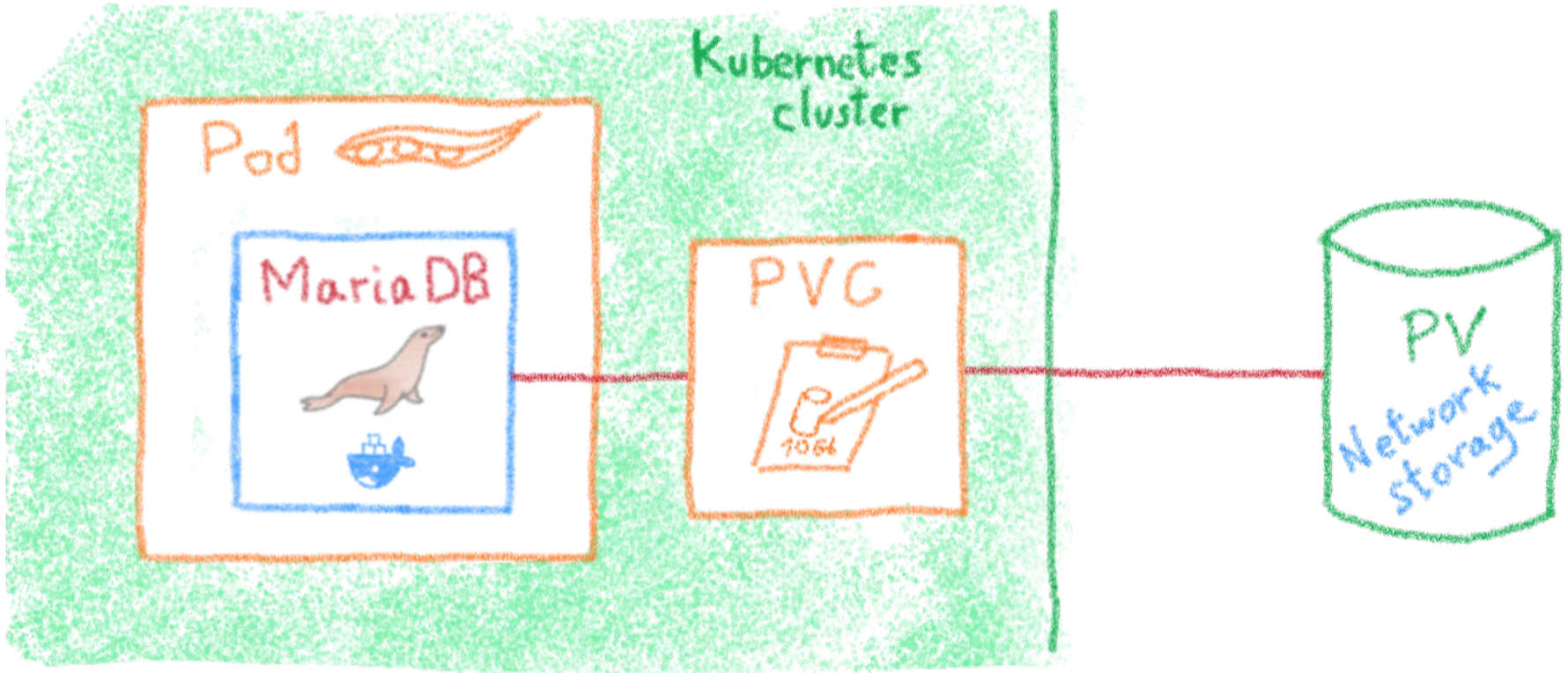
Tolerance and taints

Operators

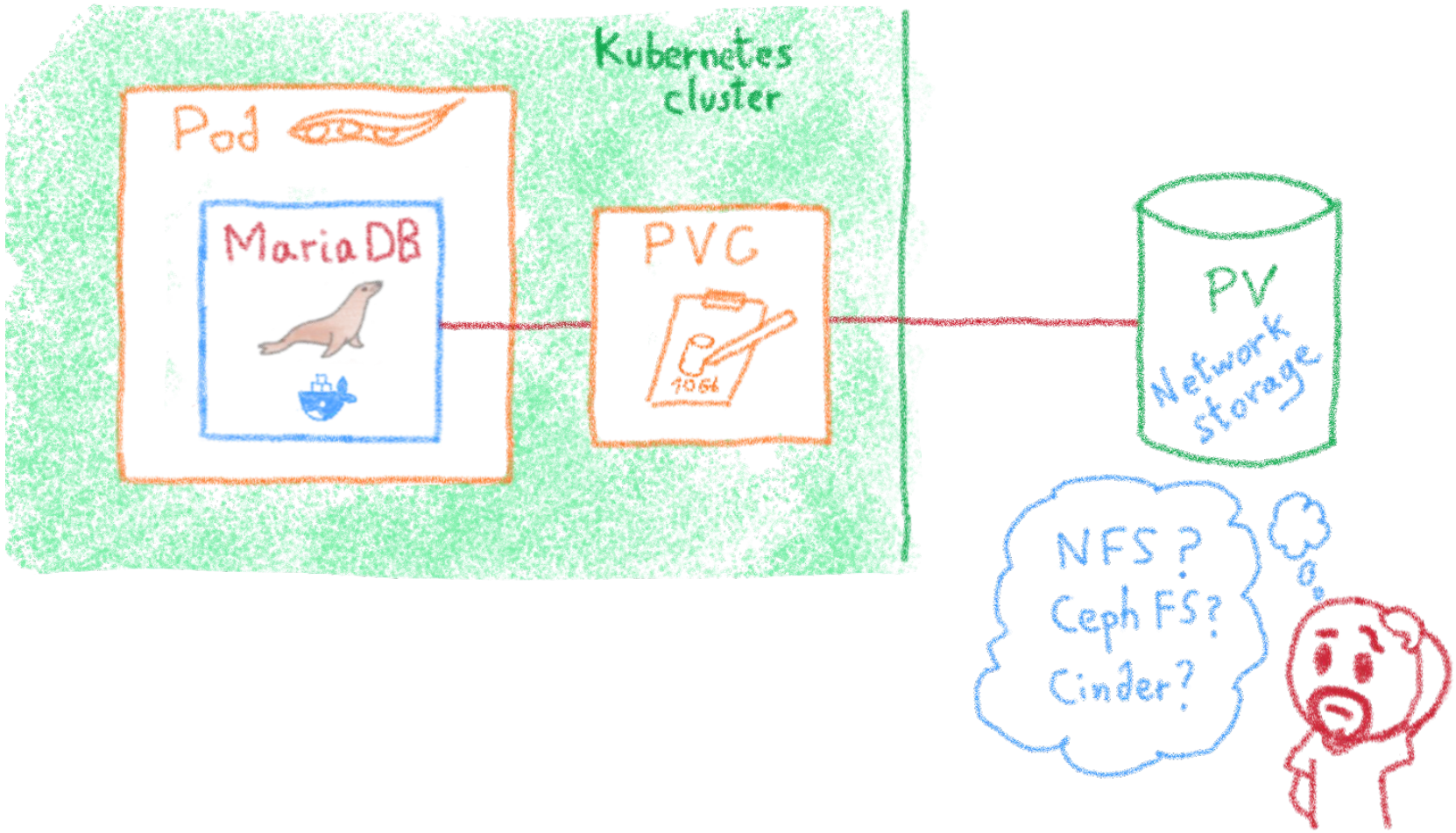
Local storage is a bad idea



Persistent Volumes



The storage dilemma



Demo MySQL

3 - Advanced K8s

Persistent Volumes

Tolerance and taints

Operators

Taints & Tolerations

Taint

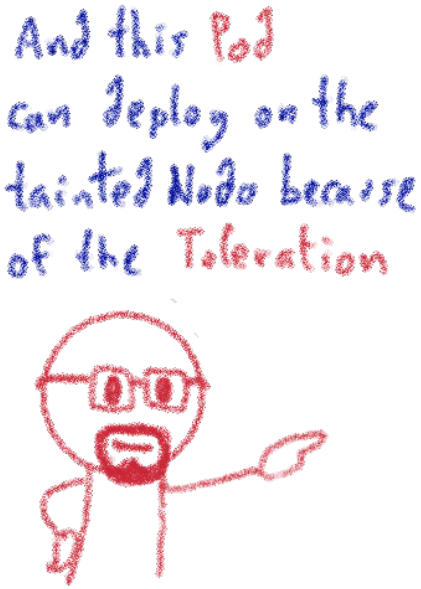
applied to a Kubernetes Node that signals the scheduler to avoid or not schedule certain Pods

Toleration

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

Using Taints & Tolerations

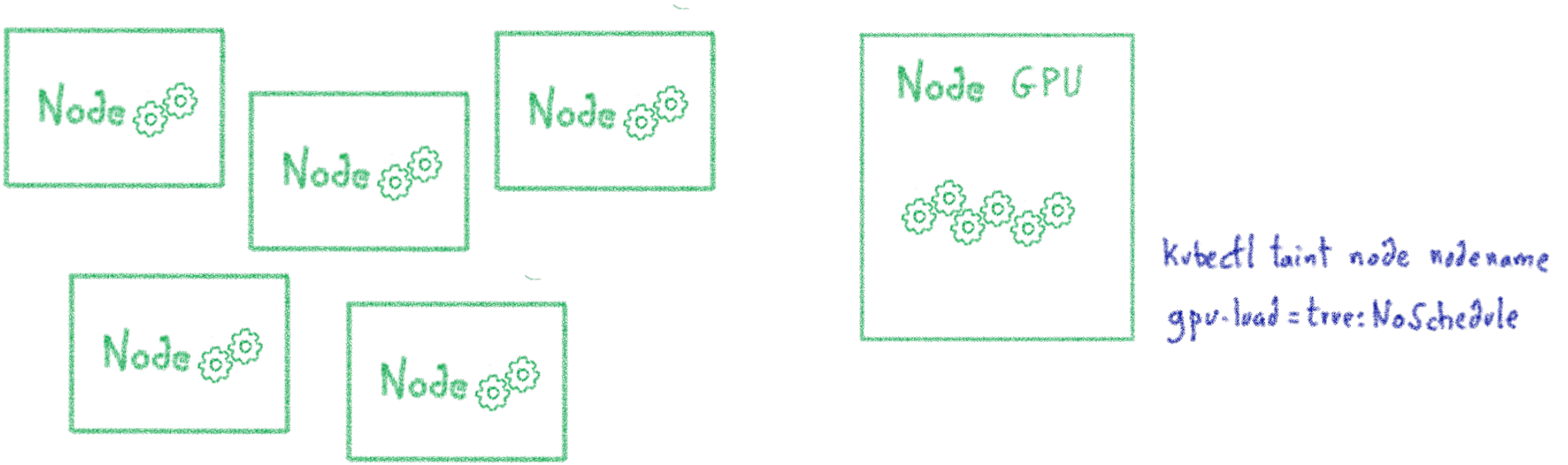

```
# No pod will be able to schedule onto node-5c283f unless it has a matching toleration.
$ kubectl taint nodes node-5c283f  type=high-cpu:NoSchedule
node/node-5c283f tainted
```




```yaml
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



Dedicated nodes

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.)

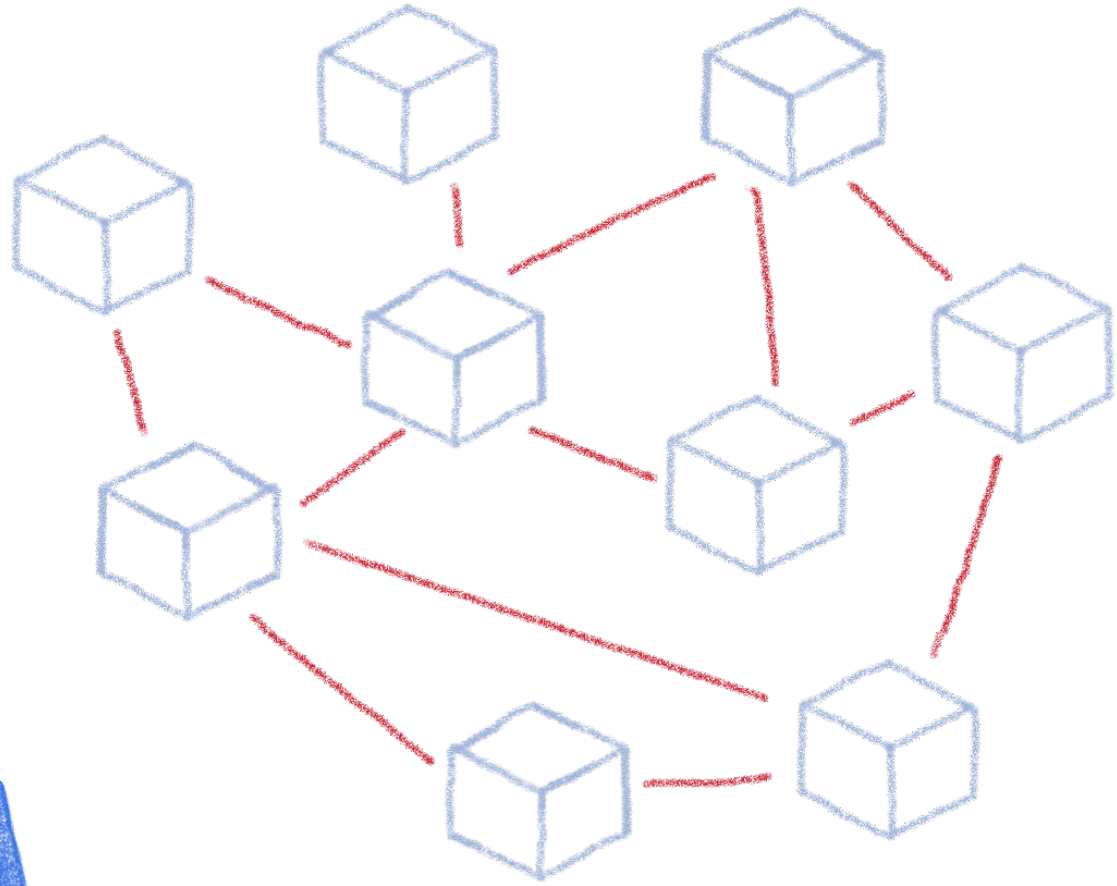
3 - Advanced K8s

Persistent Volumes

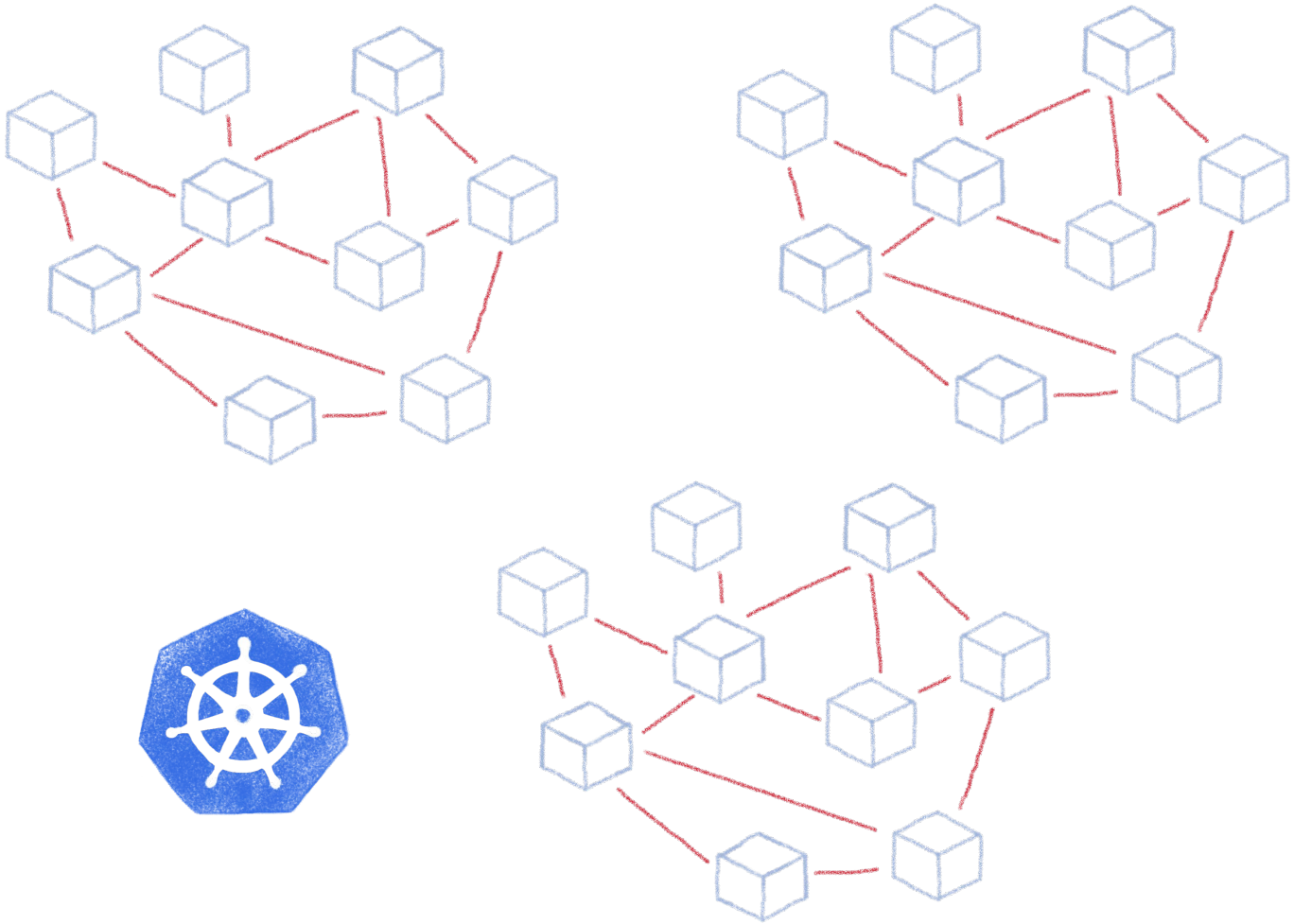
Tolerance and taints

Operators

Taming microservices with Kubernetes



What about complex deployments



Ingress

Services

Deployments

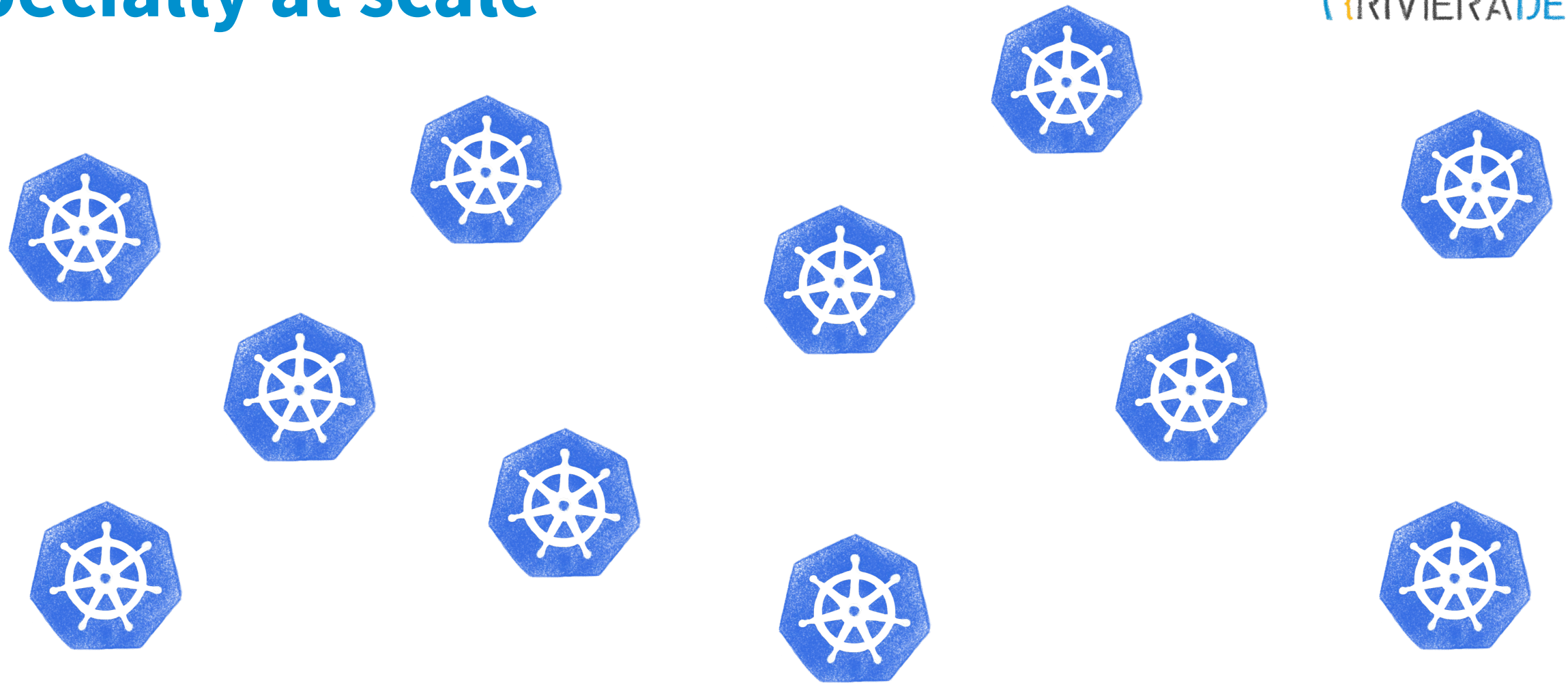
Pods

Sidecars

Replica Sets

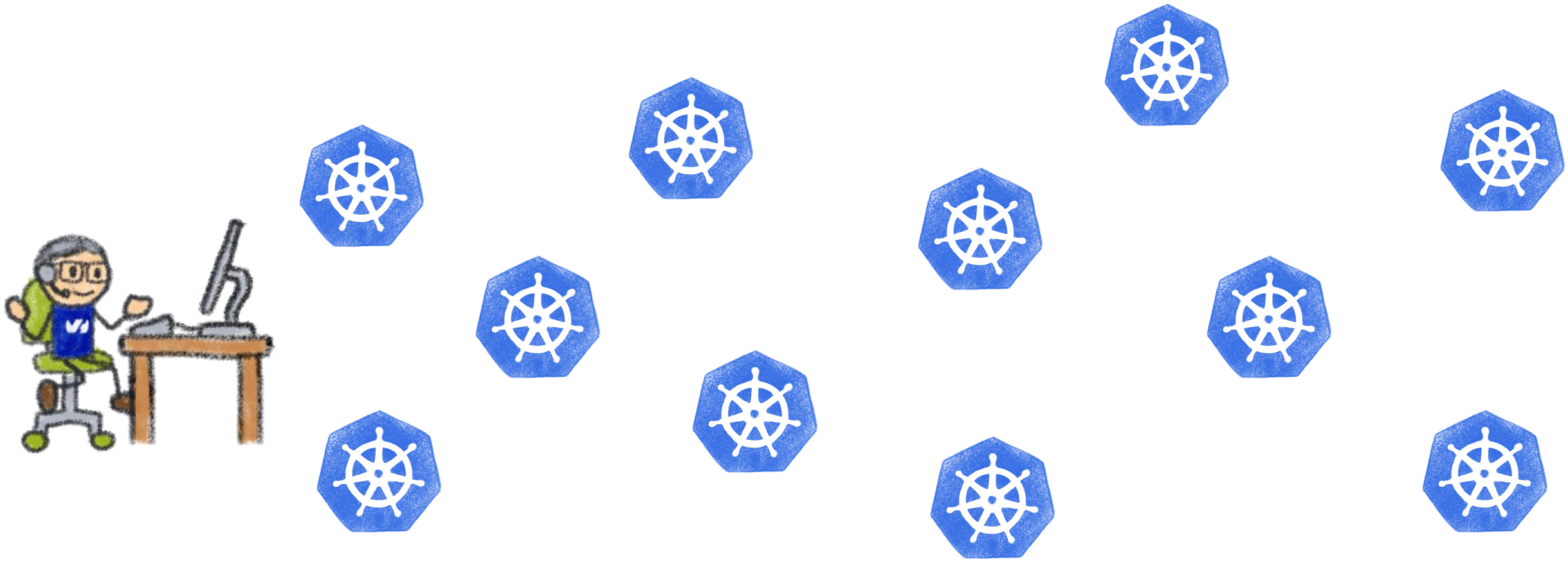
Statepul Sets

Specially at scale



Lots of clusters with lots and lots of deployments

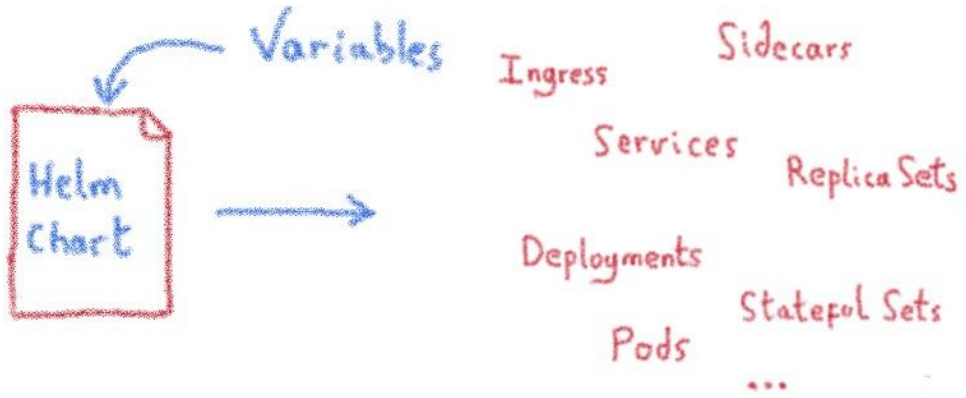
We need to tame the complexity



Making it easier to operate

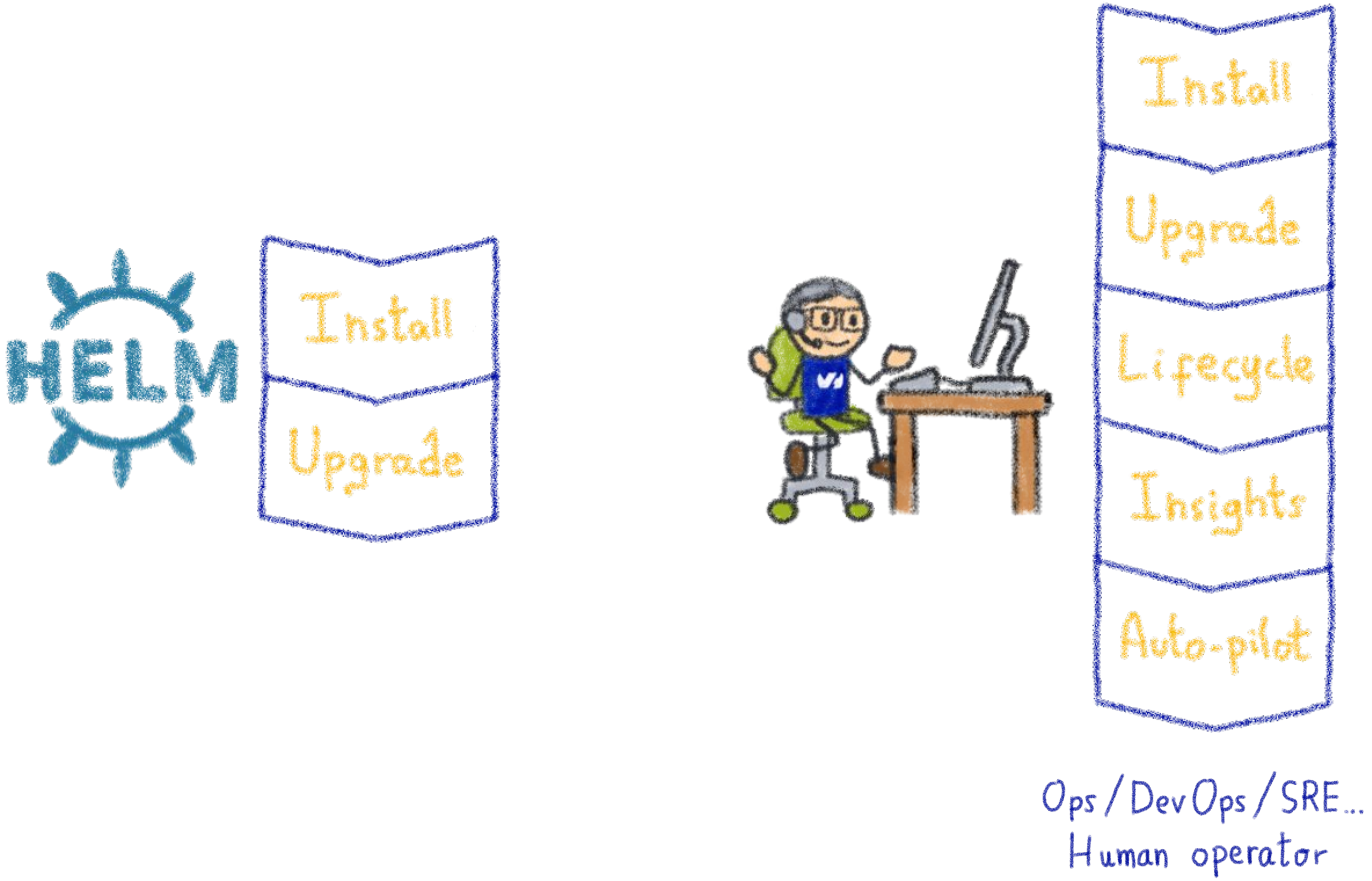
Taming the complexity

A package manager for Kubernetes



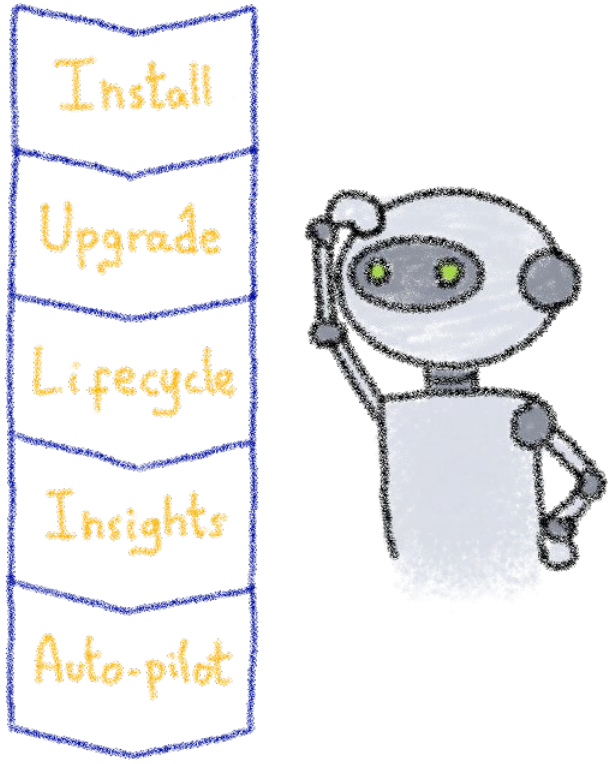
- Manage complexity
- Simple sharing
- Easy upgrades
- Easy rollbacks

Helm Charts are configuration



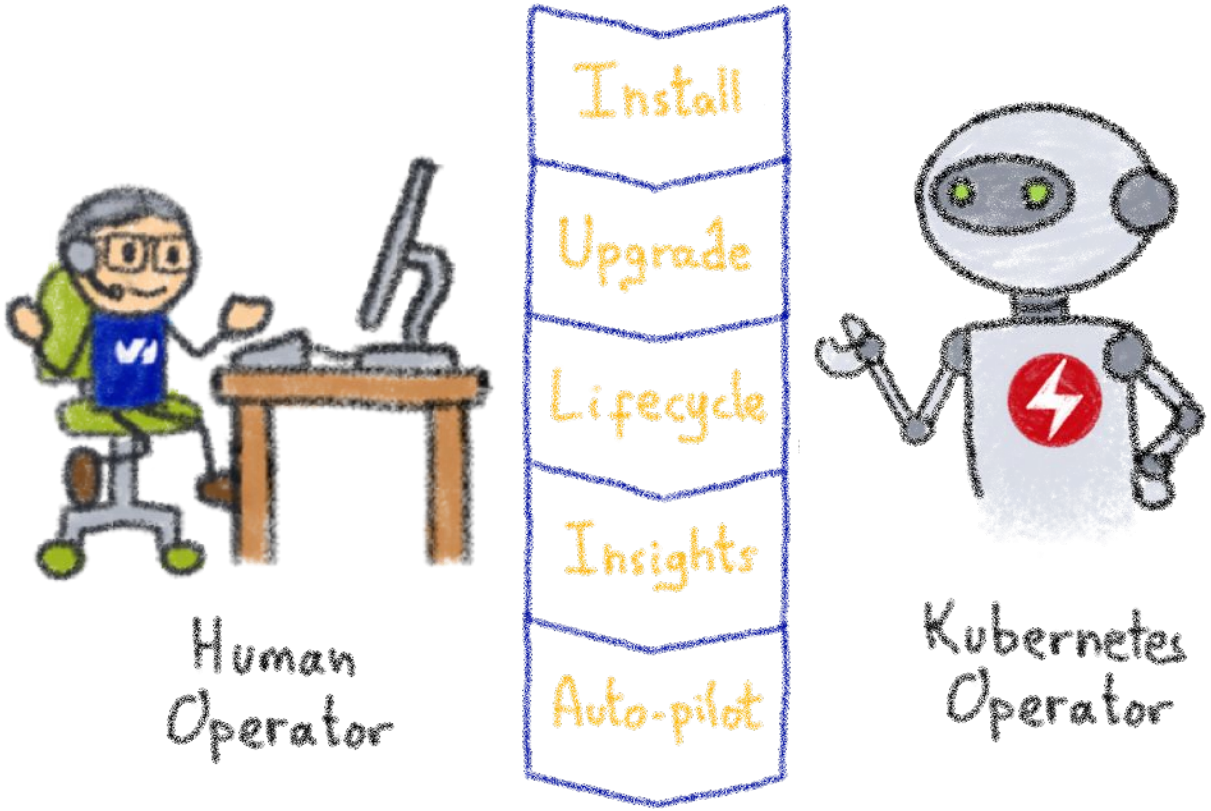
Operating is more than installs & upgrades

Kubernetes is about automation



How about automating human operators?

Kubernetes Operators

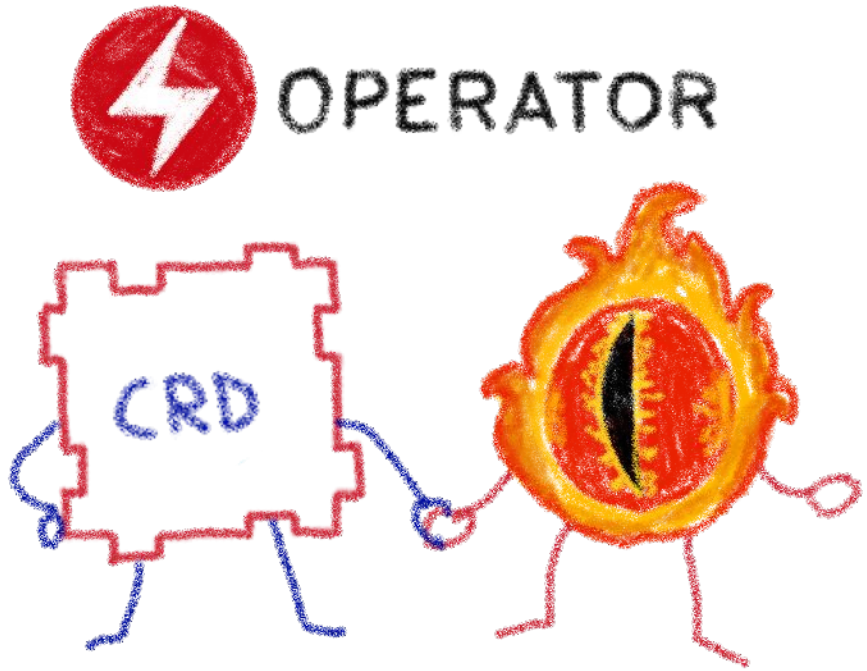


A Kubernetes version of the human operator

Kubernetes Controllers

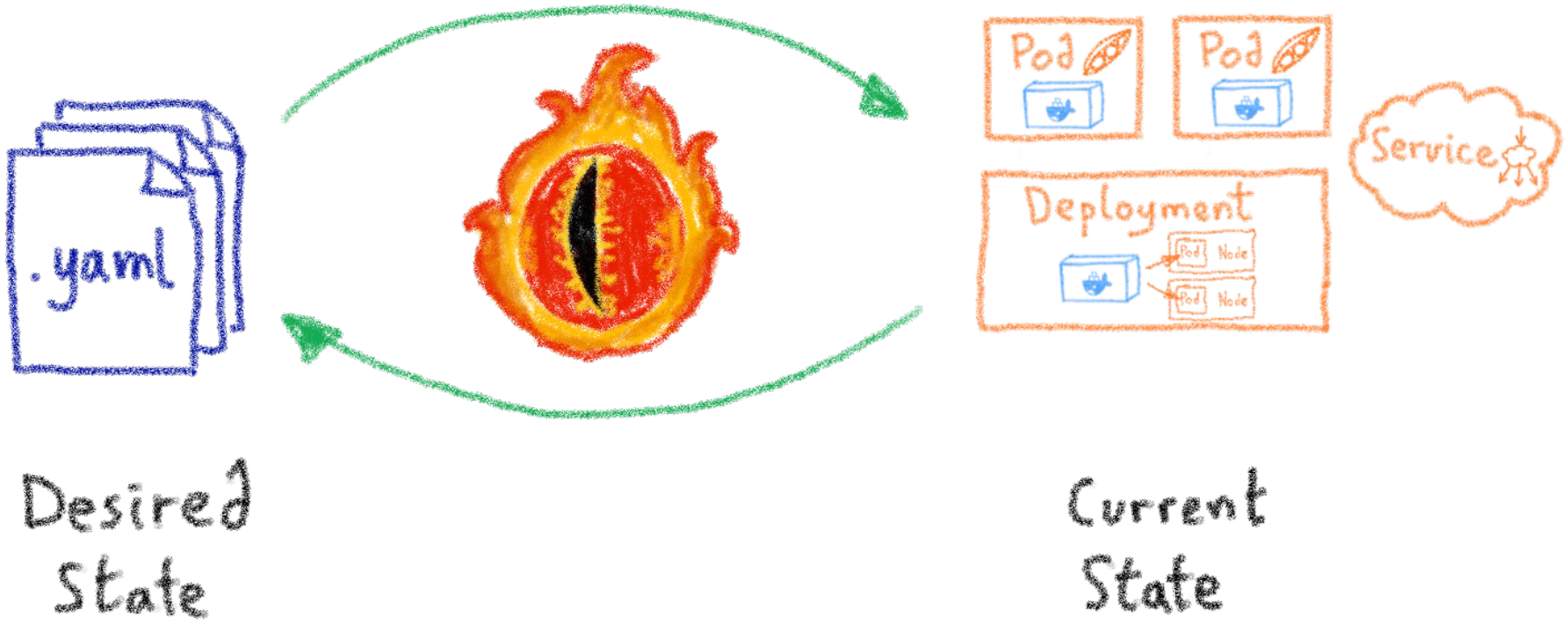
Keeping an eye on the resources

Building operators



Basic K8s elements: Controllers and Custom Resources

Kubernetes Controllers: control loops

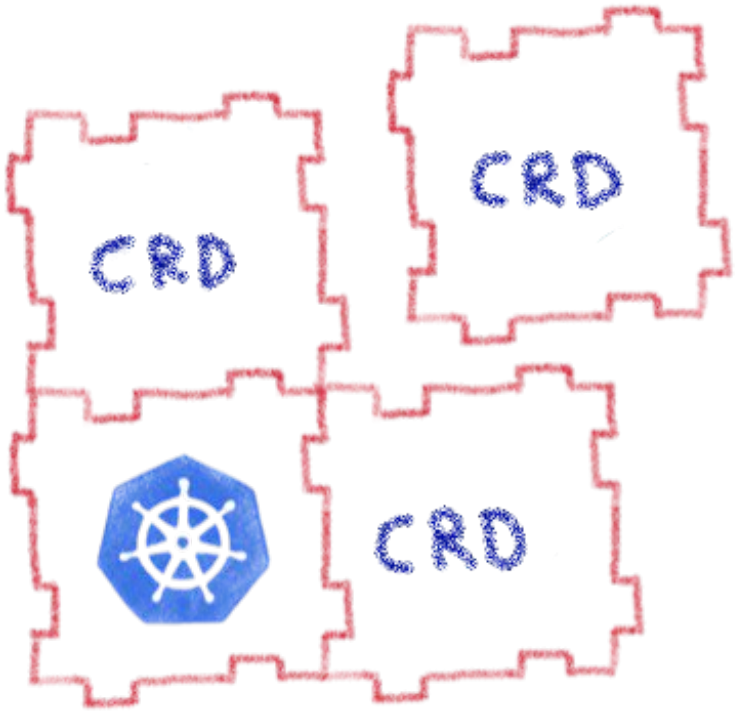


They watch the state of the cluster,
and make or request changes where needed

A reconcile loop

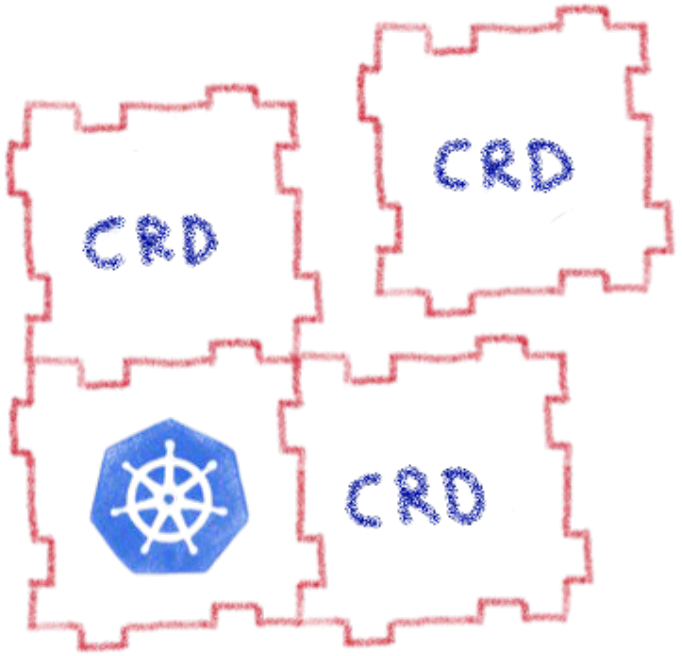
Strives to reconcile current state and desired state

Custom Resource Definitions



Extending Kubernetes API

Extending Kubernetes API

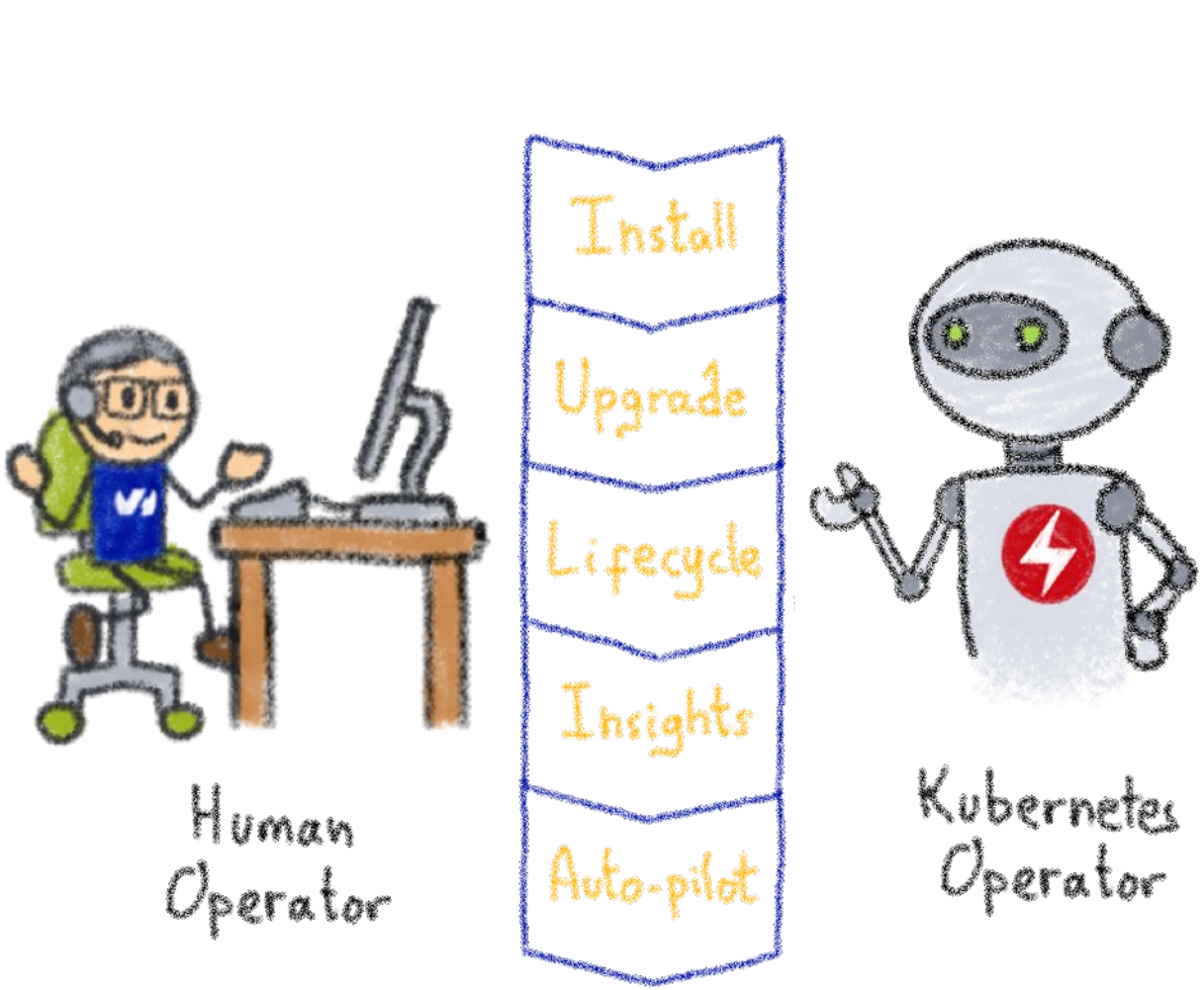


By defining new types of resources

Kubernetes Operator

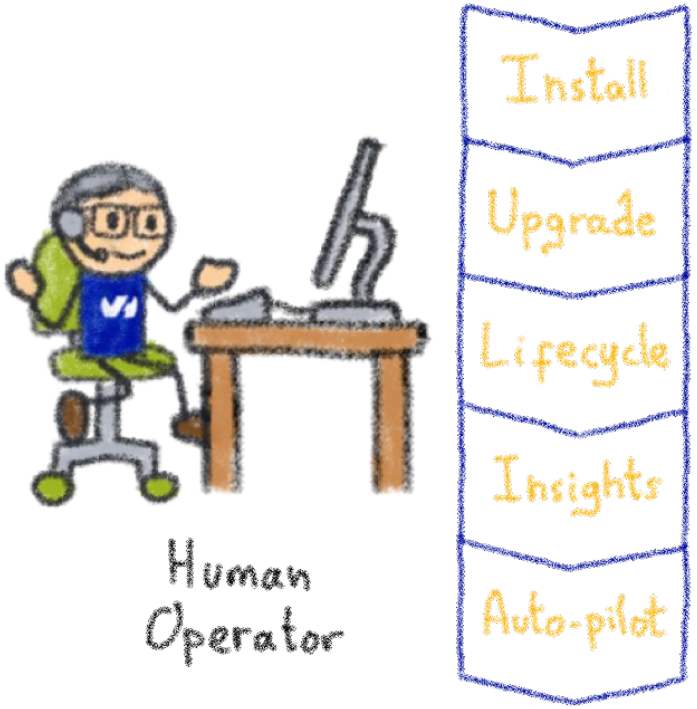
Automating operations

What's a Kubernetes Operator?



An Operator represents human operational knowledge in software to reliably manage an application

Example: databases

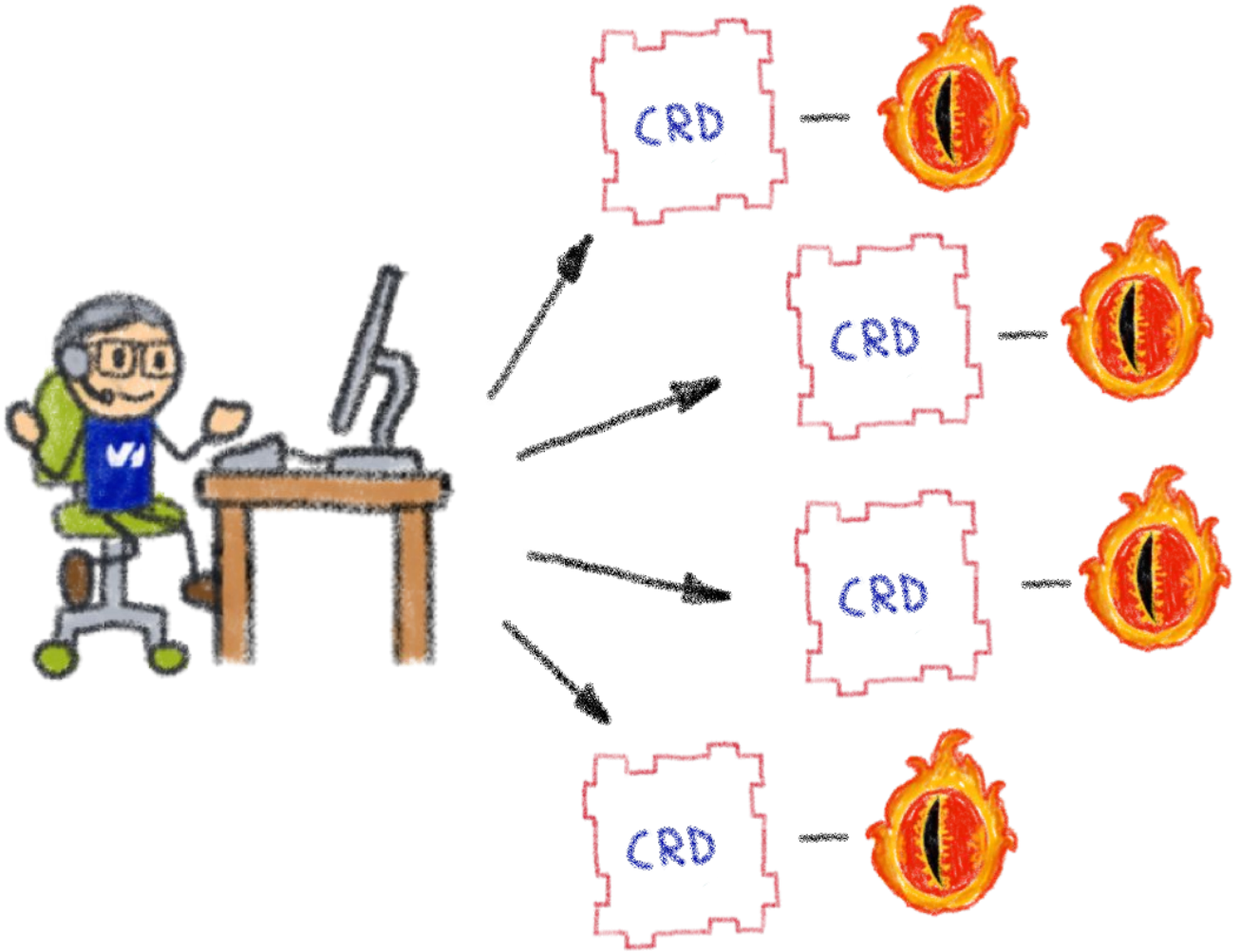


PostgreSQL

redis

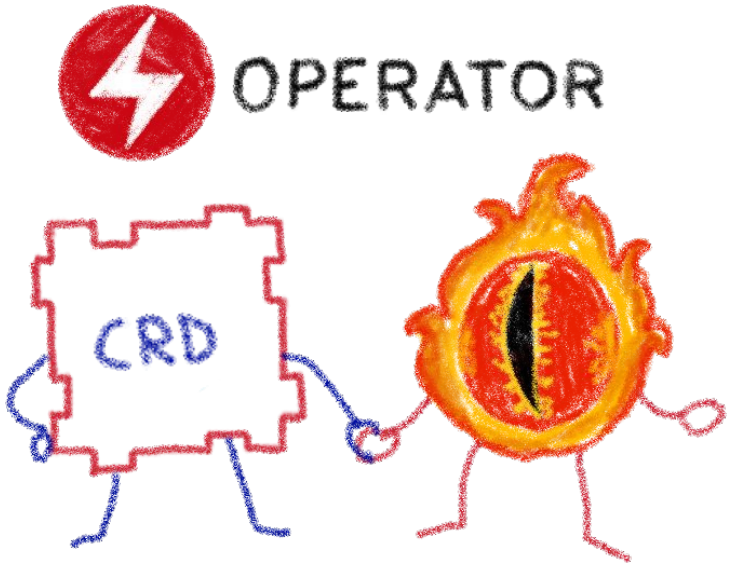
Things like adding an instance to a pool, doing a backup, sharding...

Knowledge encoded in CRDs and Controllers



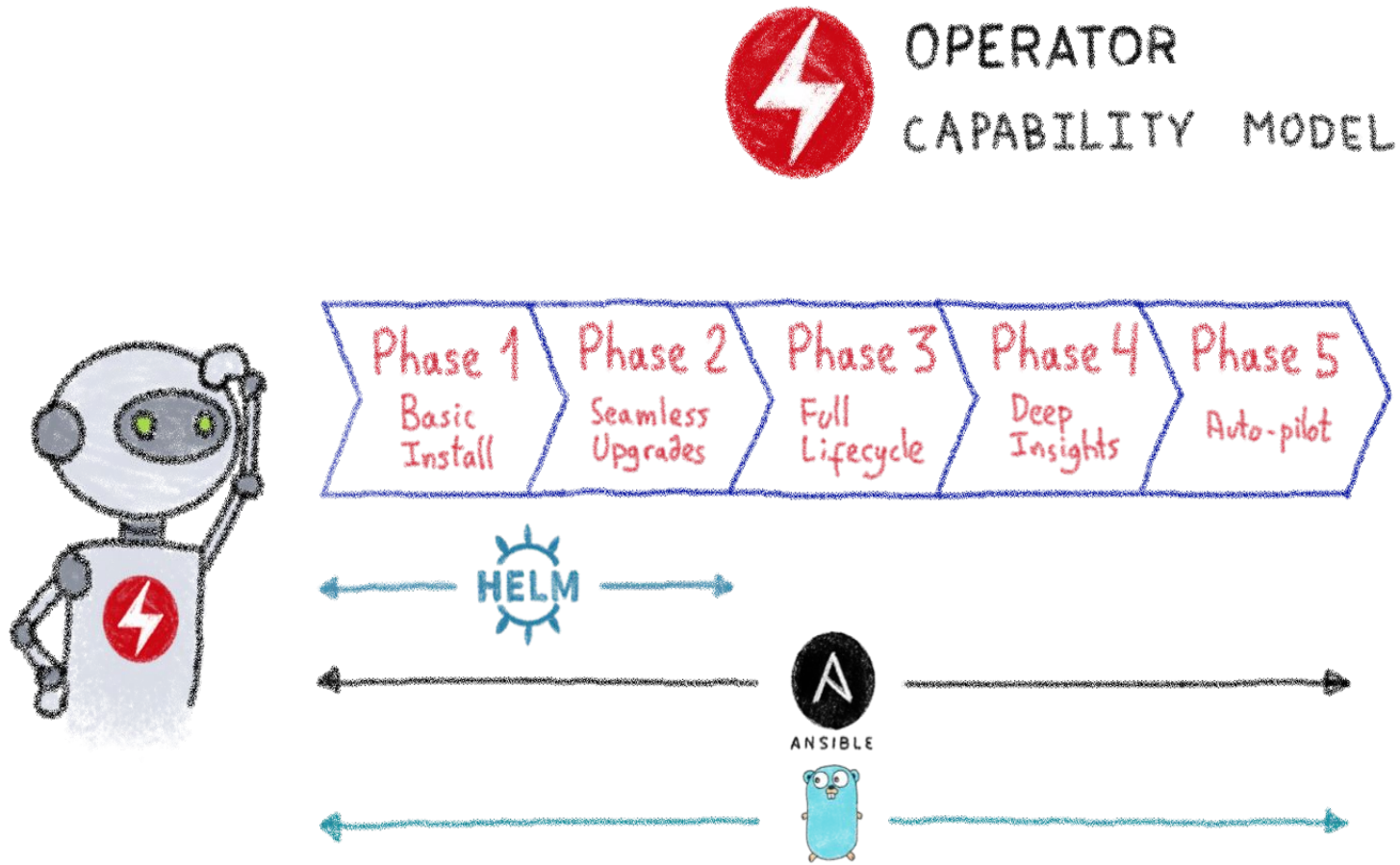
Encapsulating business logic in CRDs & Controllers

Custom Controllers for Custom Resources



Operators implement and manage Custom Resources using custom reconciliation logic

Operator Capability Model

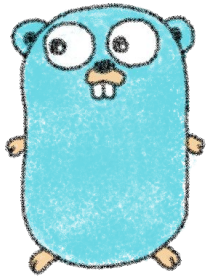


Gauging the operator maturity

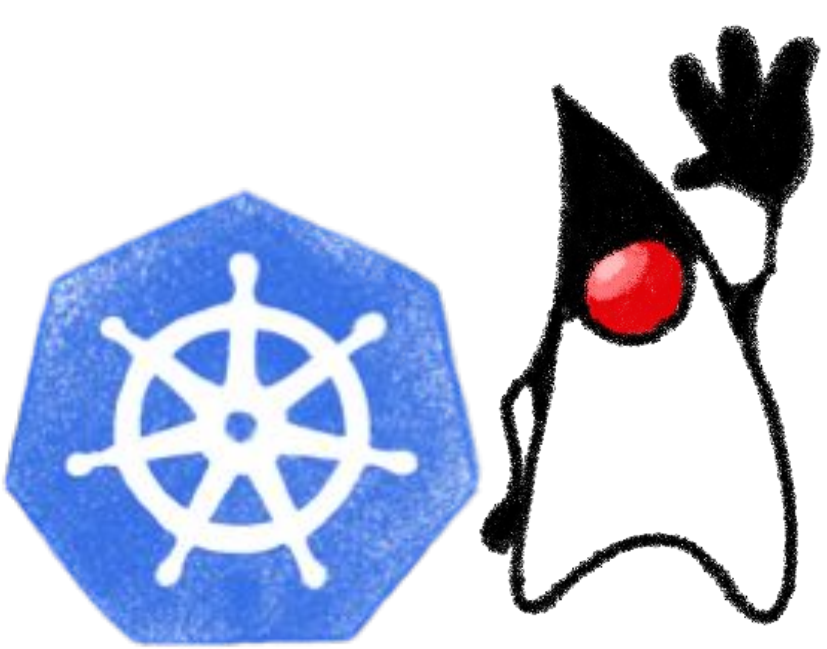
Operator SDK

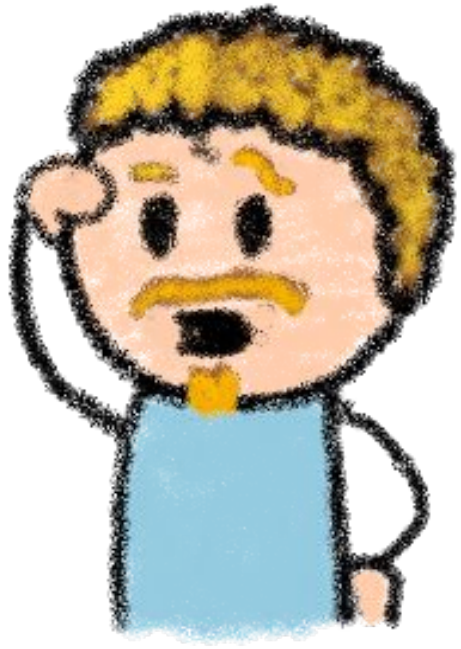
OPERATOR
SDK

BUILD
TEST
ITERATE



But I'm a Java developer!





Can I code Kubernetes Operators in Java?

Easily?

Operators in Java

Démo Opérator

Quizz Kahoot

Le livre d'Aurélie

That's all, folks!

Thank you all!