DAWAN Paris 11,rue Antoine Bourdelle, 75015 PARIS
DAWAN Nantes 32, Bd Vincent Gâche, 5e étage - 44200 NANTES
DAWAN Lyon Bt Banque Rhône Alpes, 2ème étage – 235 cours Lafayette 69006 LYON

Formation Kubernetes

Plus d'info sur http://www.dawan.fr ou 0810.001.917

Formateur: Pierre Sablé

Présentation de la formation

Qui suis-je? « mon parcours » « mes compétences »

Exprimer votre besoin:

- Le contexte **IT**
- Votre environnement de travail
- Vos **objectifs** pour cette formation.

Y-a-t-il des personnes avec un handicap ?

DAWAN - Reproduction interdite

Objectifs

Comprendre la virtualisation et ses concepts

La philosophie **Kubernetes**

Installation et configuration de Kubernetes

Configurations avancées

Avec ces acquis vous serez en mesure de :

- Installer et configurer une plateforme K8s
- S’adapter facilement à des contextes différents

DAWAN - Reproduction interdite

Tests

Que vous évoque la virtualisation?

Qu’est ce qu’un conteneur?

DAWAN - Reproduction interdite

Introduction

DAWAN - Reproduction interdite

Enjeux de la Virtualisation

Gain énergétique

Gain QOS

Une flexibilité accrue

Cibler, au mieux, les besoins de votre parc informatique

DAWAN - Reproduction interdite

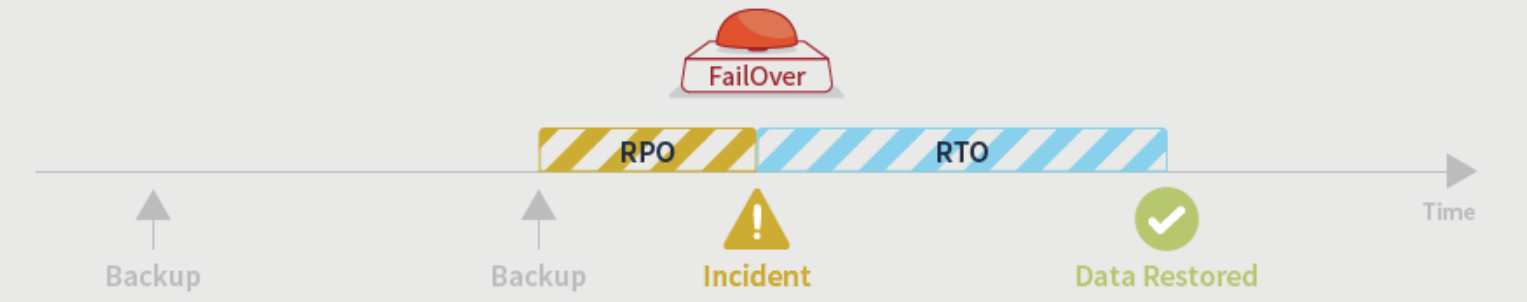
PRA - RTO/RPO

PRA: Plan de relance d’activité

RTO: Durée maximale d'interruption admissible

RPO: Durée maximum d'enregistrement des données qu'il est acceptable de perdre lors d'une panne.

Ex: sauvegarde toutes les 24H



Ces facteurs permettrons de cibler les technologies en adéquation avec les contraintes internes au projet.

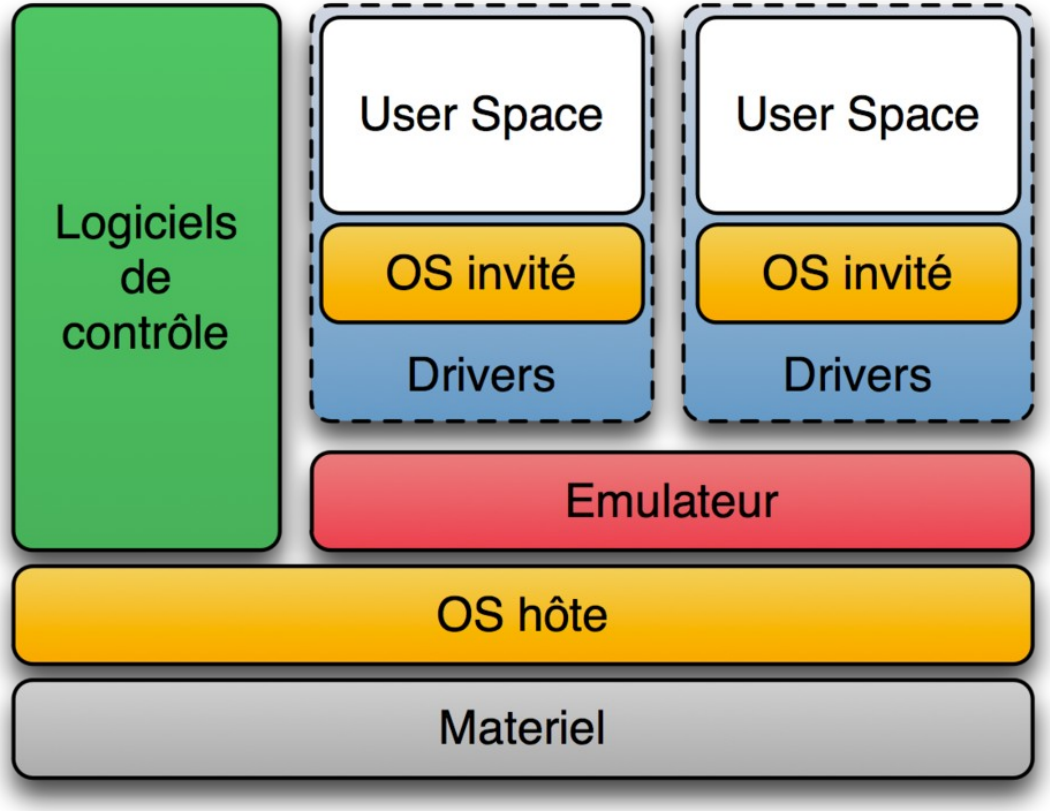
DAWAN - Reproduction interdite

Les différents types de virtualisation

DAWAN - Reproduction interdite

Hyperviseur niveau 2

Fonctionnement:



Avantage/Défaut:

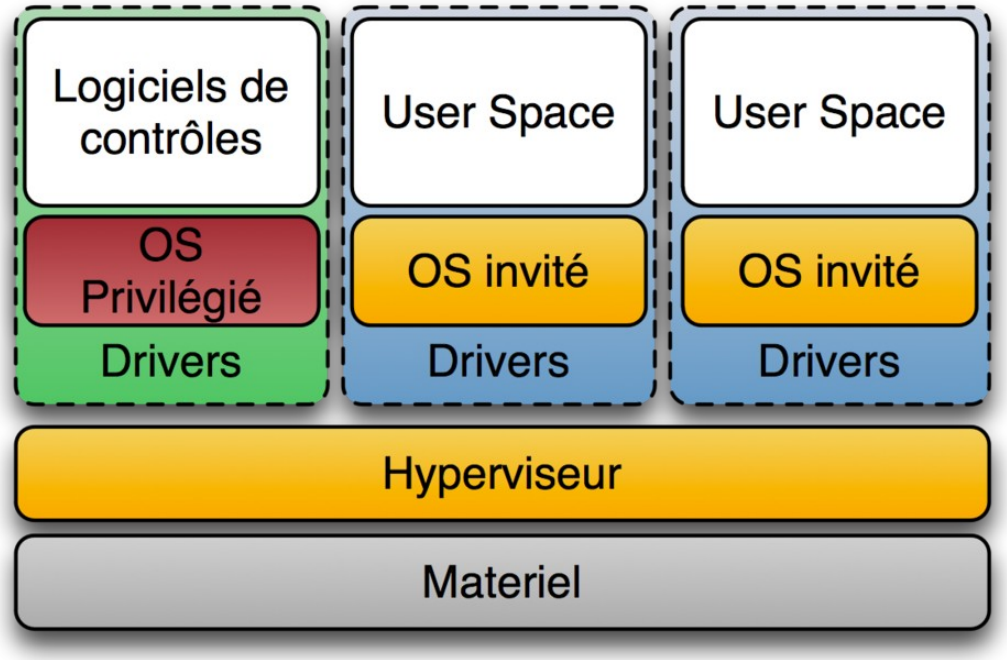
- Très peu performant
- Limité en terme d'infrastructure

\+ Facile d'installation

DAWAN - Reproduction interdite

Hyperviseur niveau 1

Fonctionnement:



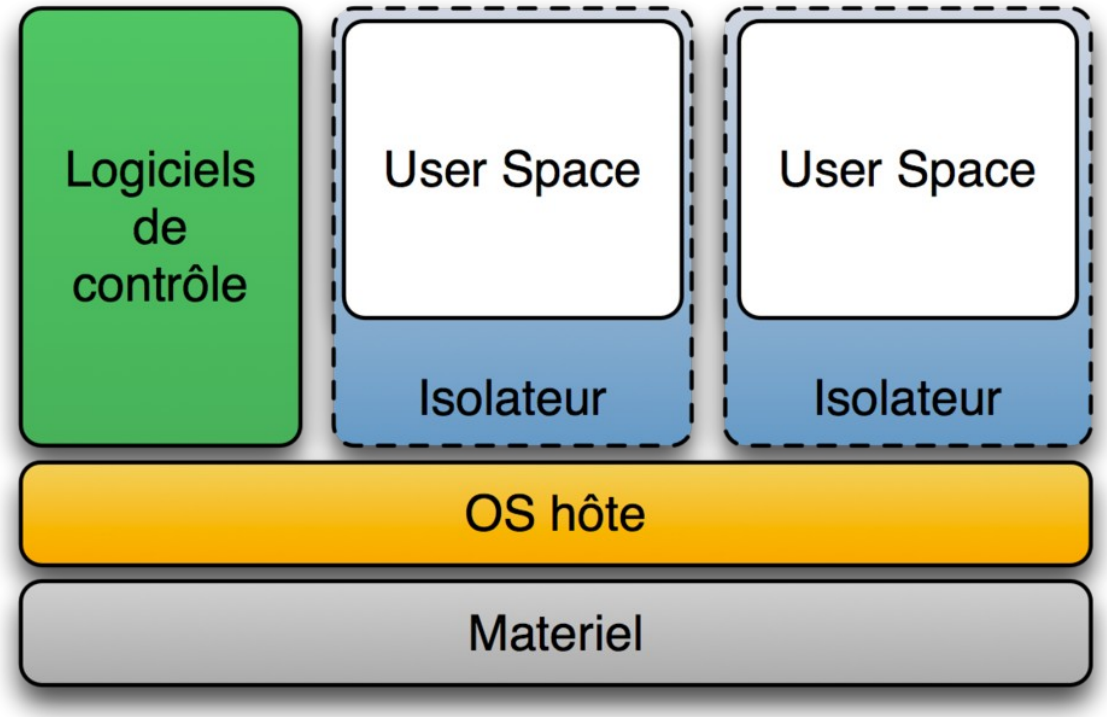
Avantage/Défaut:

- Plus complexe à mettre en œuvre
+ Performant
+ Modulable

DAWAN - Reproduction interdite

Isolateur

Fonctionnement:



Avantage/Défaut:

- Lié au système hôte

+ Performant (1~3% de Pénalité)

Exemple de solutions:

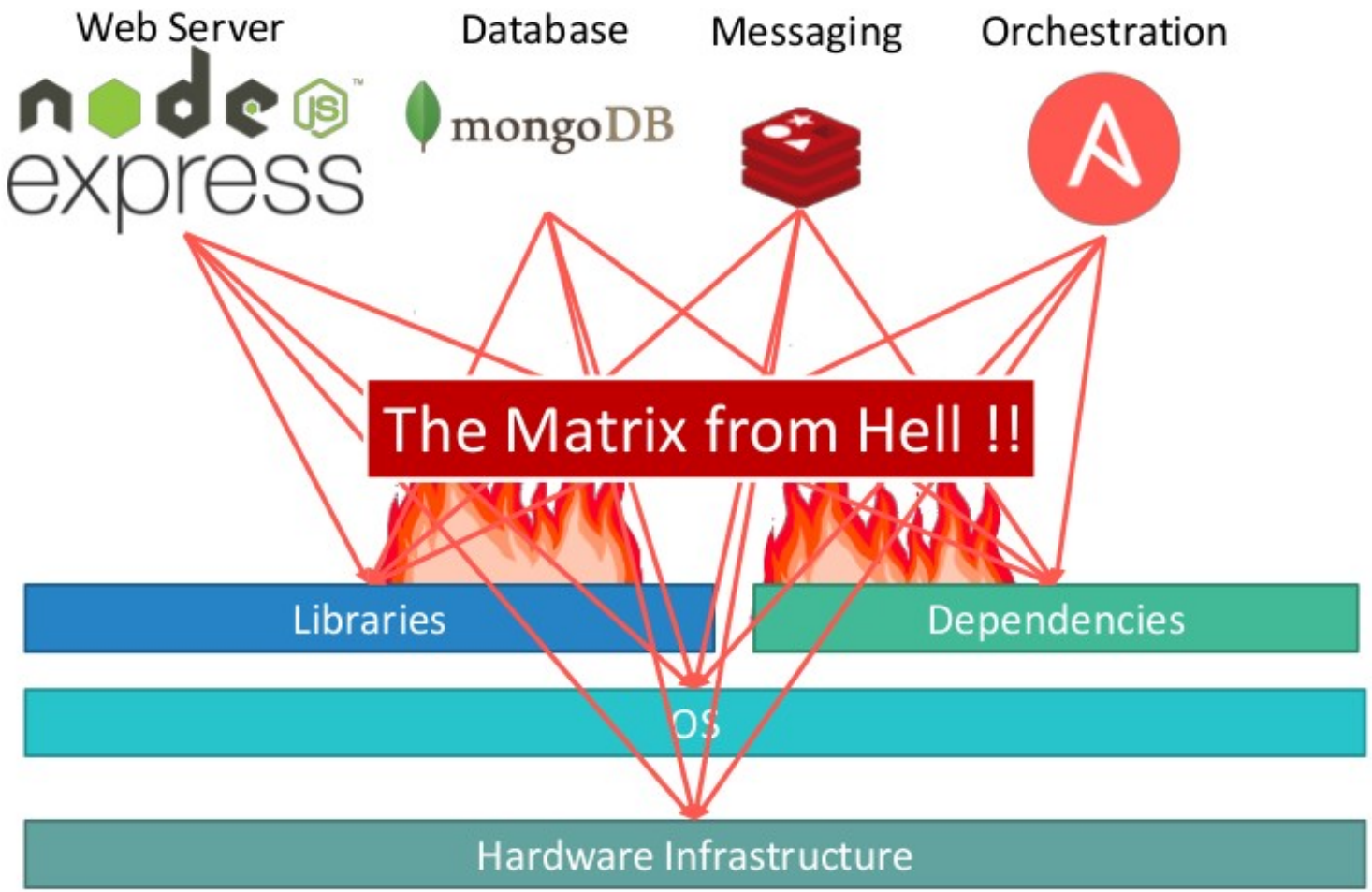
- OpenVZ (Isolateur d'OS)
- Chroot (Isolateur de racine)
- Linux VSEVER & BSD Jail (Isolateur d'espace utilisateur)

DAWAN - Reproduction interdite

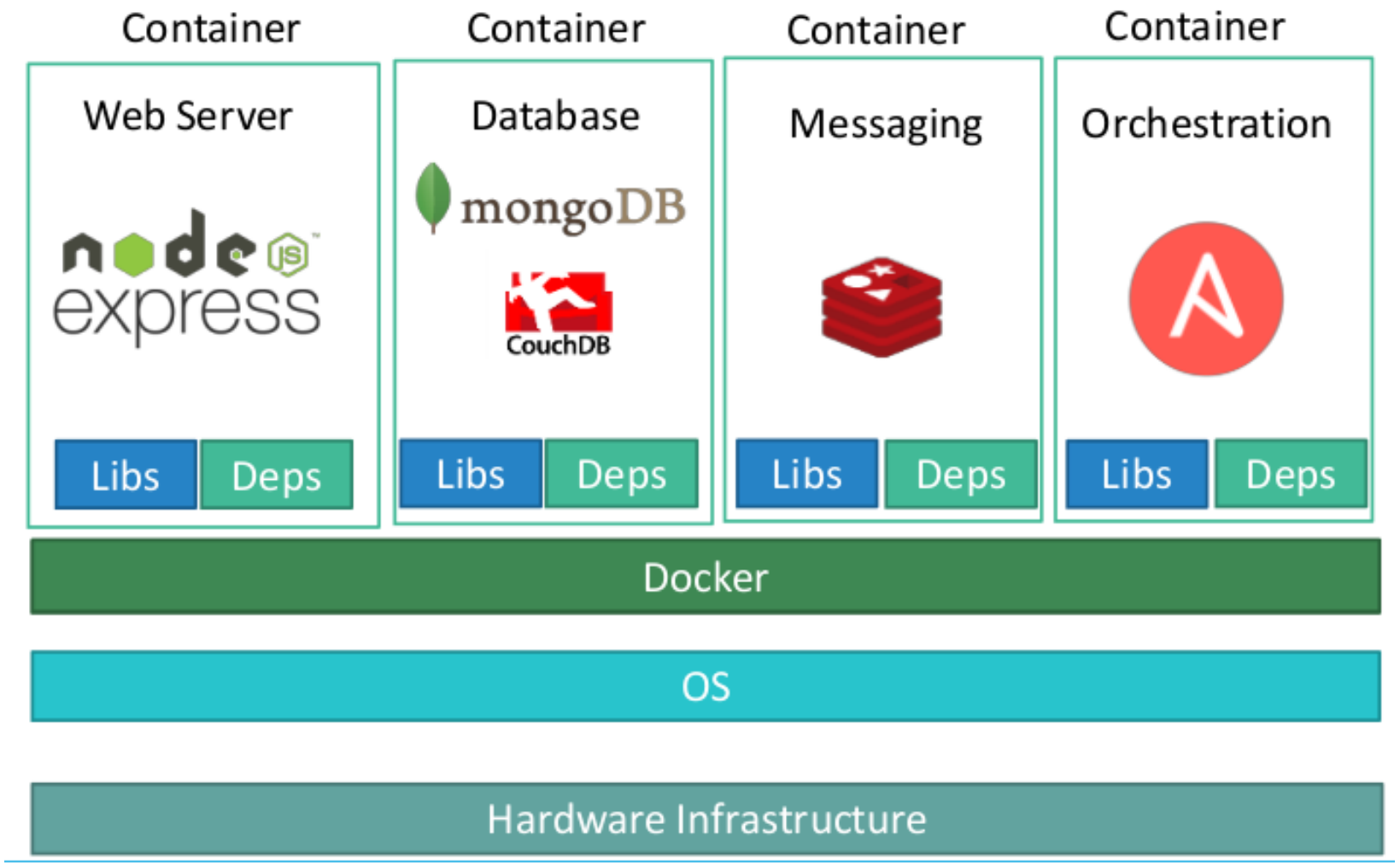
Concepts utiles

DAWAN - Reproduction interdite

Pourquoi basculer sur les conteneurs

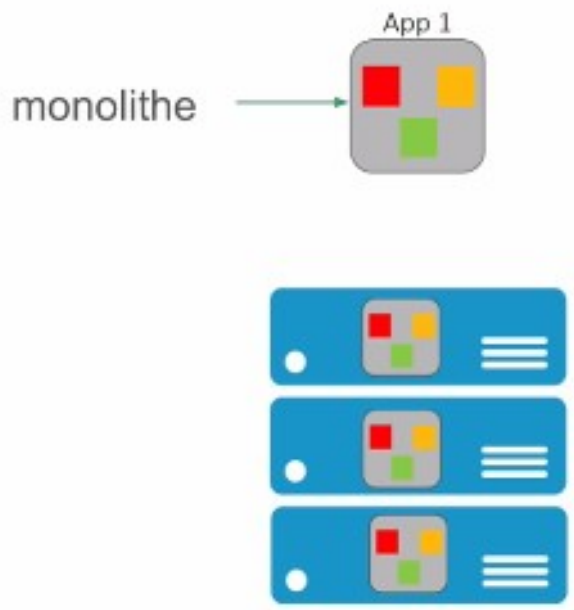


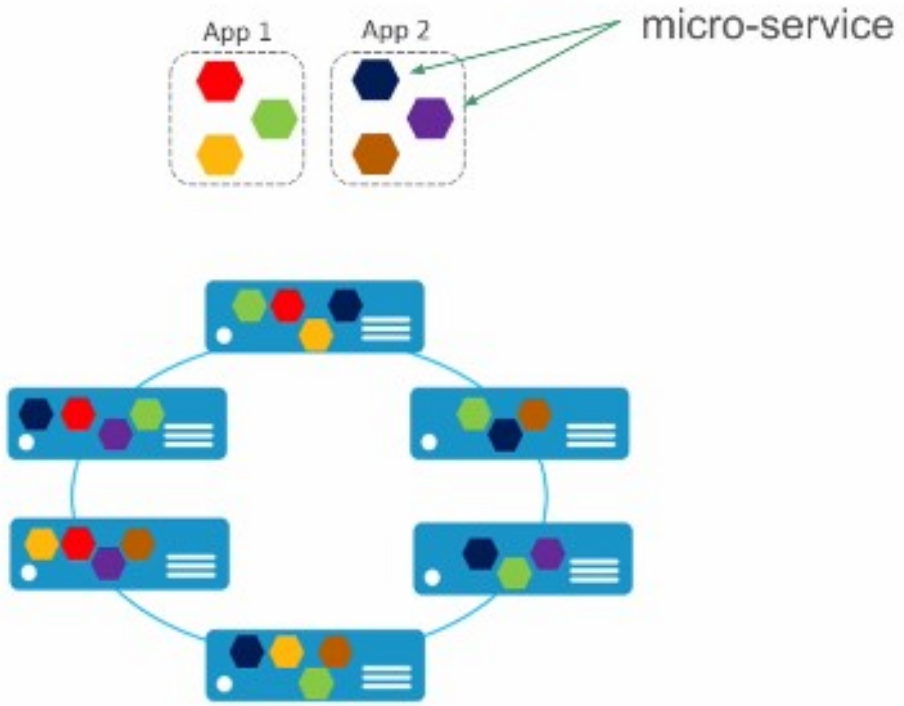
Application micro-services conterneurisées



DAWAN - Reproduction interdite

Architecture monolithique vs micro-services





https://docs.microsoft.com

DAWAN - Reproduction interdite

Architecture micro-services

- Découpage de l'application en multiples services
- Processus indépendant ayant sa propre responsabilité métier
- Plus grande liberté de choix dans le langage
- Équipe dédiée pour chaque service
- Un service peut être mis à jour indépendamment des autres services
- Containers très adaptés pour les micro-services
- Nécessite des interfaces bien définies
- Déplace la complexité dans l'orchestration de l'application globale

DAWAN - Reproduction interdite

Application Cloud Native

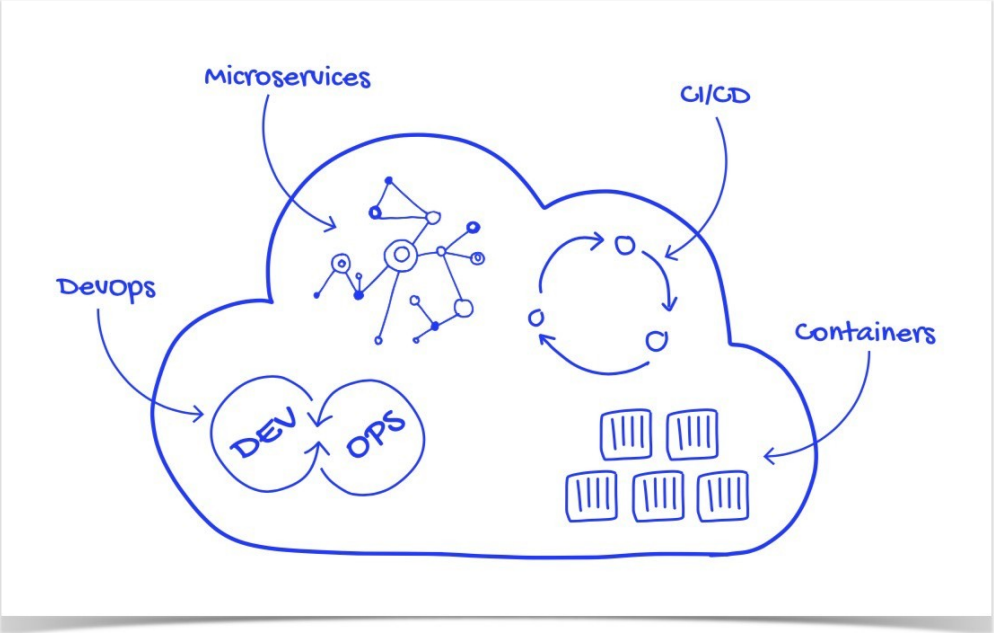
- Application orientée microservice

- Packagée dans des container

- Orchestration dynamique

- cncf.io

- Nombreux projets portés par la CNCF
  (Cloud Native Computing Foundation)
  Kubernetes
  Prometheus



DAWAN - Reproduction interdite

DevOps

- **Un objectif :** minimiser le temps de livraison d’une fonctionnalité

- **Déploiements réguliers**

- **Tests d’intégrations**

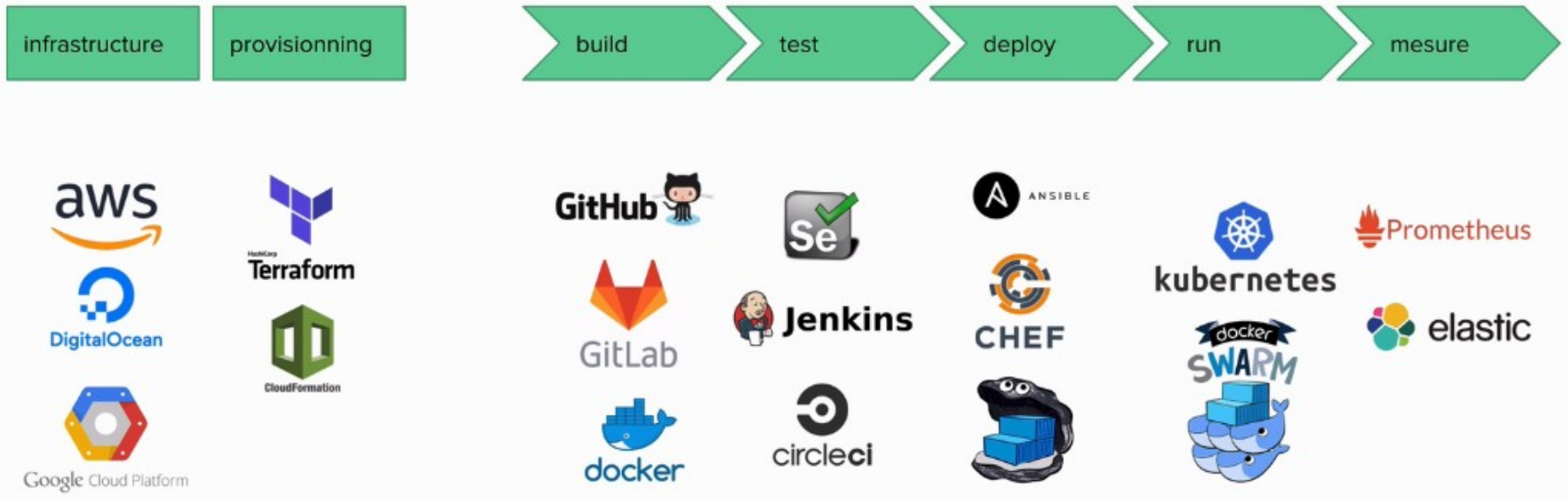
- **Automatisation des processus :** provisionning, IaC, CI/CD, monitoring

- **Amélioration continue :** cycles courts

- **Langage :** YAML

DAWAN - Reproduction interdite

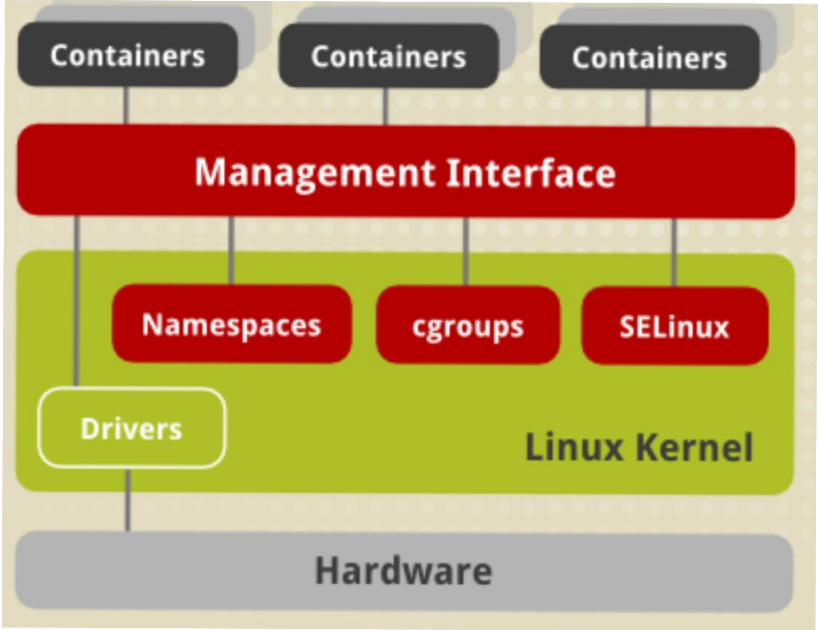
DevOps : quelques outils et produits



Les containers Linux :

Concept :

- **Un processus !**
- **Isolé des autres processus**
- **Avec sa propre vision du système sur lequel il tourne (Namespace)**
- **Limité dans les ressources qu'il peut utiliser (Control Groups)**
- **Partage le Kernel de la machine hôte avec les autres containers**



DAWAN - Reproduction interdite

La plateforme Docker

- Facilite l’utilisation des conteneurs

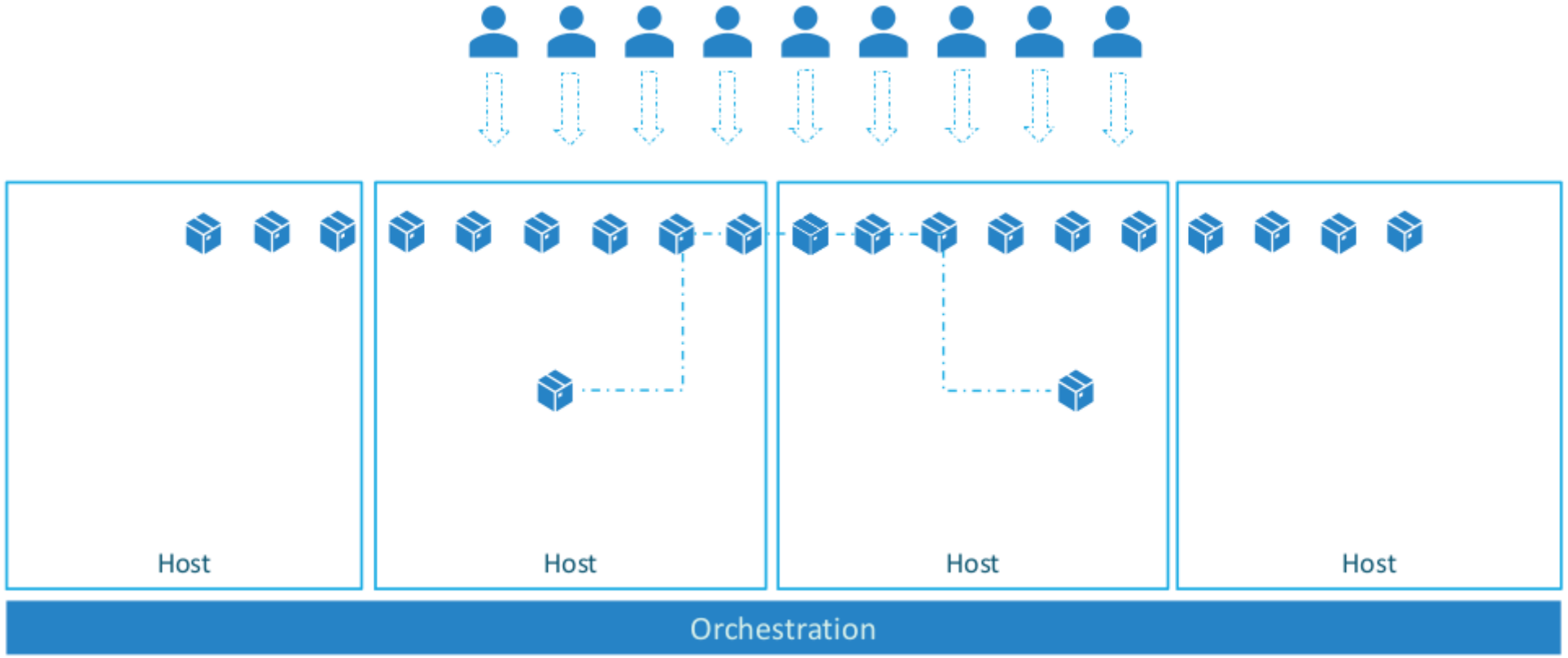
- Apporte le concept d’images
  . Packaging d’une application et de ses dépendances
  . Instanciée dans un conteneur
  . Déploiement sur une multitude d’environnements

- Plateforme docker hub

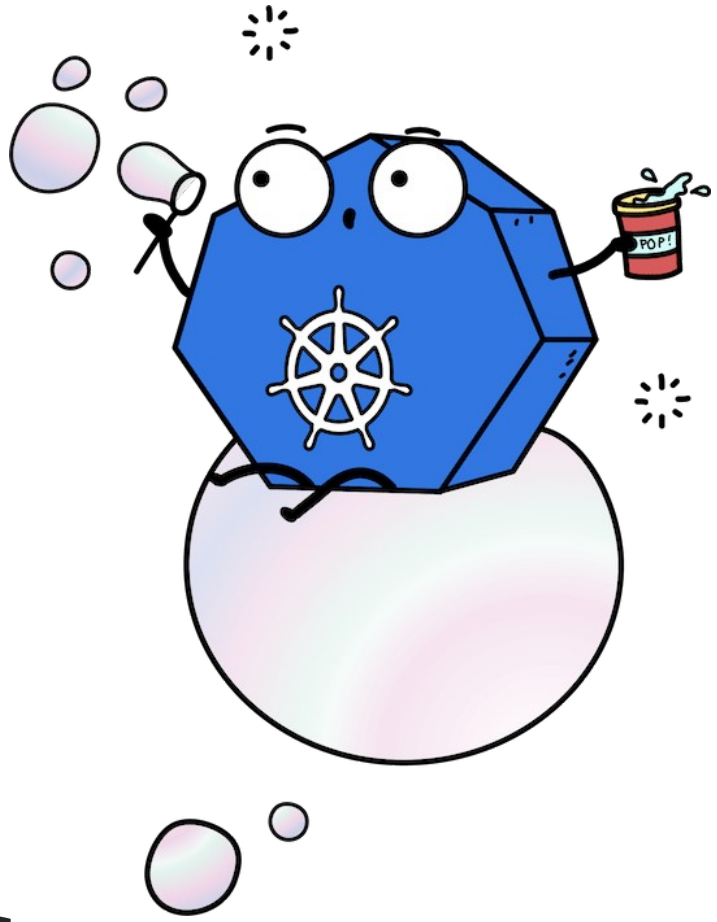
- Orchestration via Swarm

DAWAN - Reproduction interdite

Orchestration de conteneurs



DAWAN - Reproduction interdite



Kubernetes ou K8s

DAWAN - Reproduction interdite

Introduction

Kubernetes / K8s / Kube

En grec κυβερνήτης : « timonier » ou « pilote »

OpenSource : orchestration de conteneurs

Influence : système Borg de google

Versions :

- 1.0 : juillet 2015
- 1.21.2 : avril 2021

DAWAN - Reproduction interdite

K8s: Fonctionnalités

- **Plate-forme de conteneur**

- **Automatisation de déploiements**

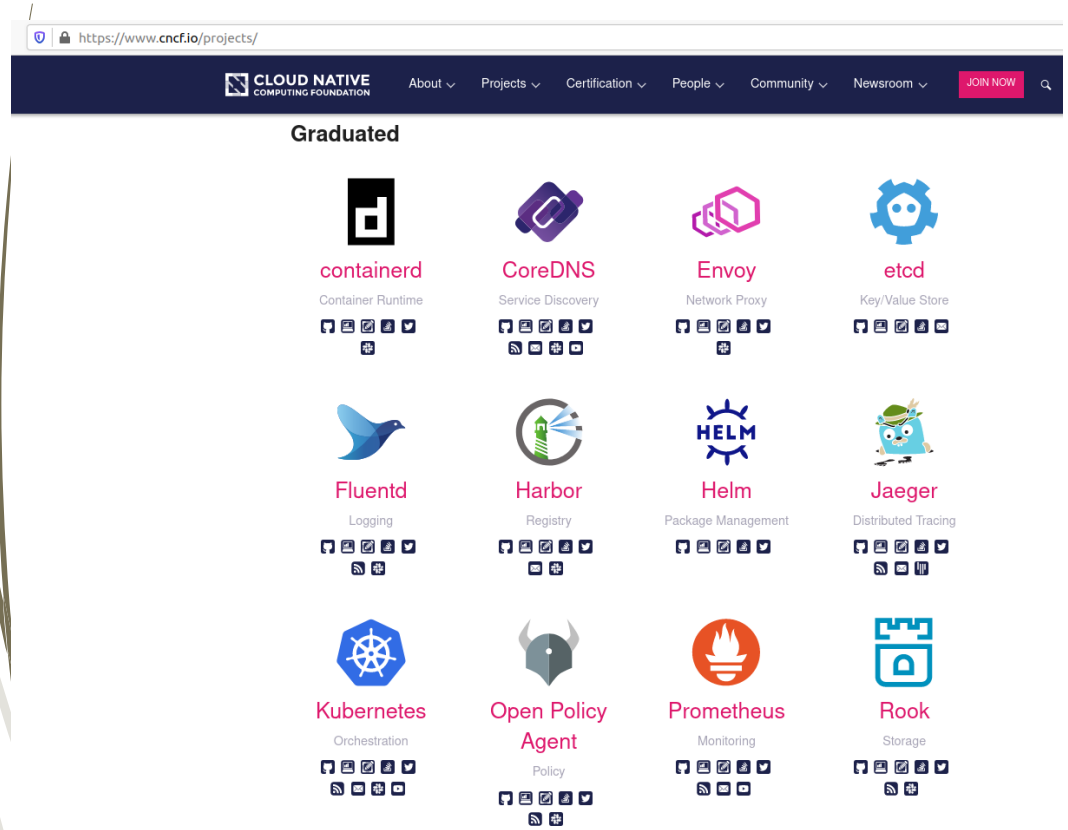
- **Mise à l’échelle (scaling)**

- **Gestions des états souhaités (Boucle de réconciliation, contrôleurs)**

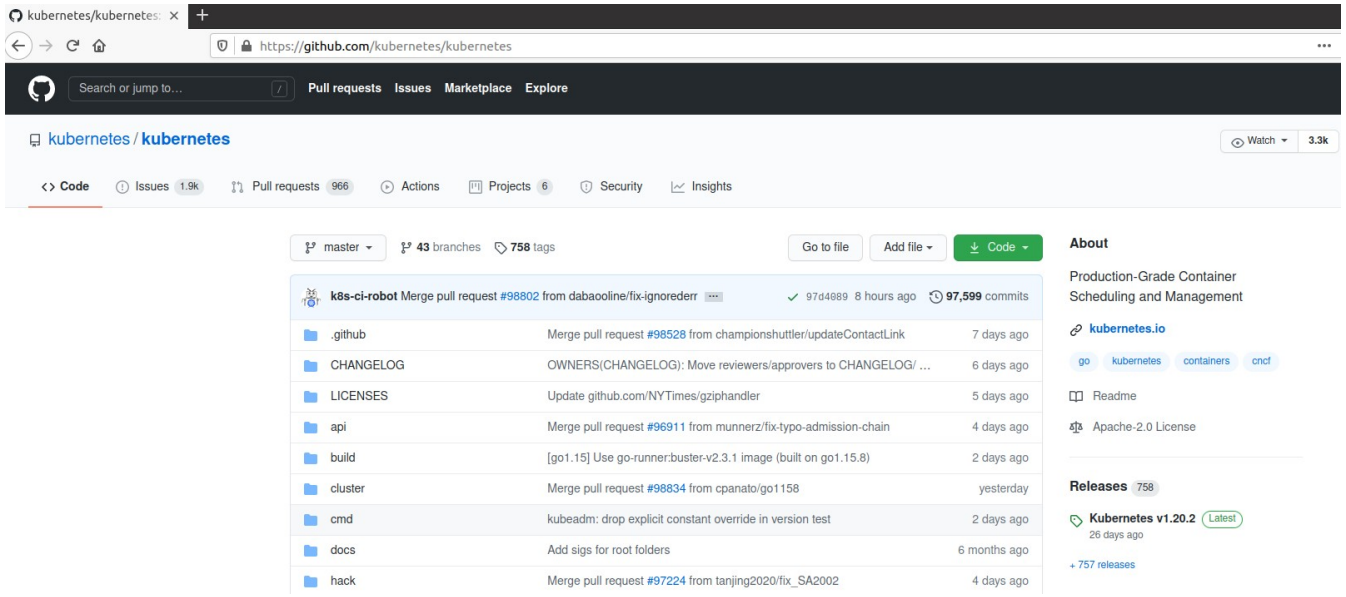
- **Applis Stateless et Stateful**

DAWAN - Reproduction interdite

K8s: Projet phare de l’écosystème



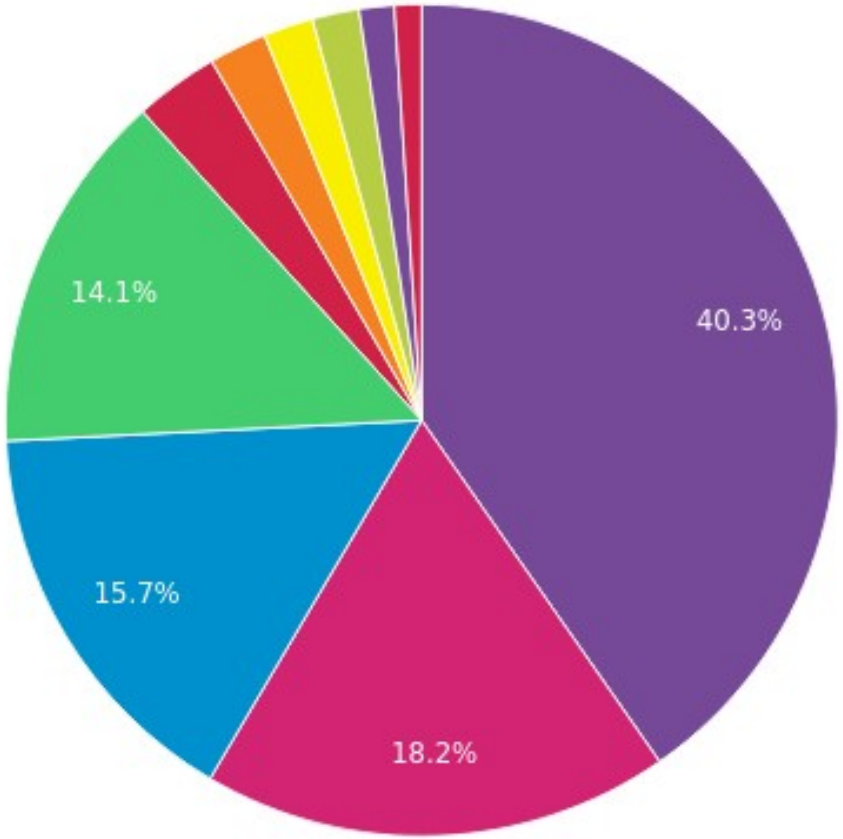
https://www.cncf.io/projects/



https://github.com/kubernetes/kubernetes

K8s: Les principaux contributeurs

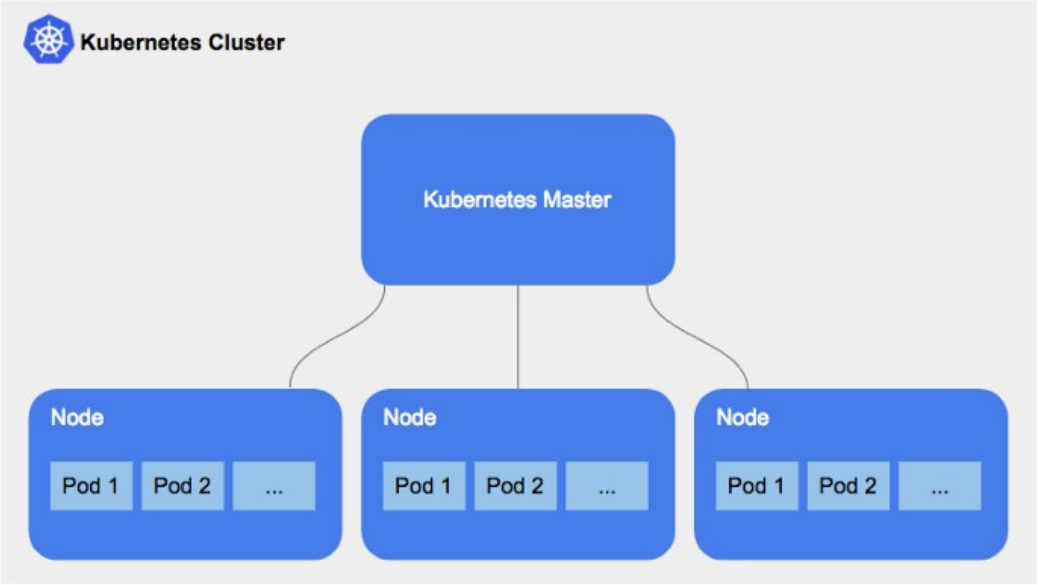
#	Company	Commits
1	Google	22542
	*independent	10162
2	Red Hat	8791
3	Huawei	1839
4	ZTE Corporation	1258
5	VMware	1083
6	Microsoft	1017
7	IBM	760
8	FathomDB	587
9	Intel	570



DAWAN - Reproduction interdite

K8s : Concepts

Cluster : Ensemble de nodes Linux ou Windows (VM / bare metal)

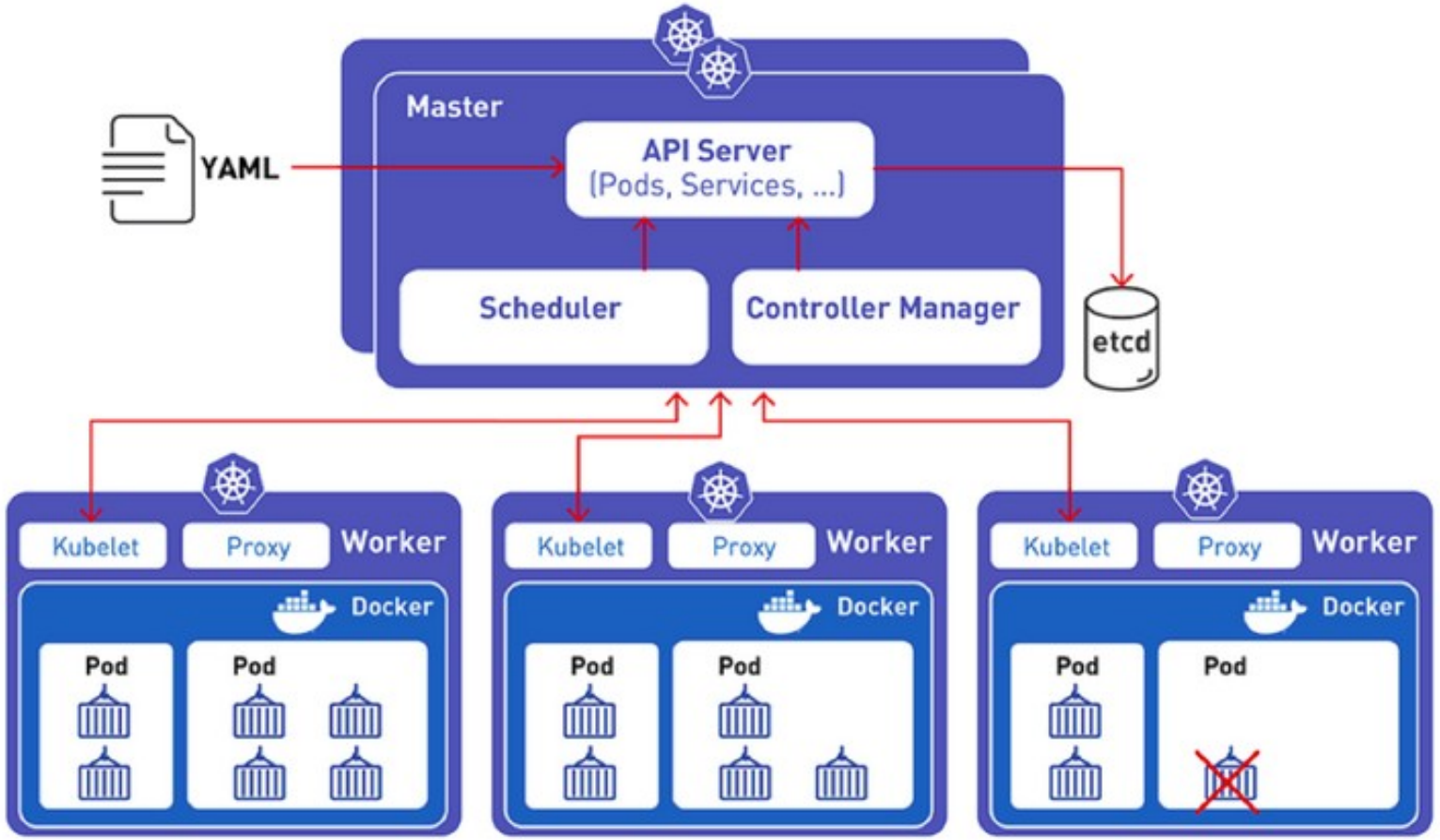


Master : responsable du maintien de l'état souhaité pour le cluster

Node : workers : machines qui exécutent les applications et workflows

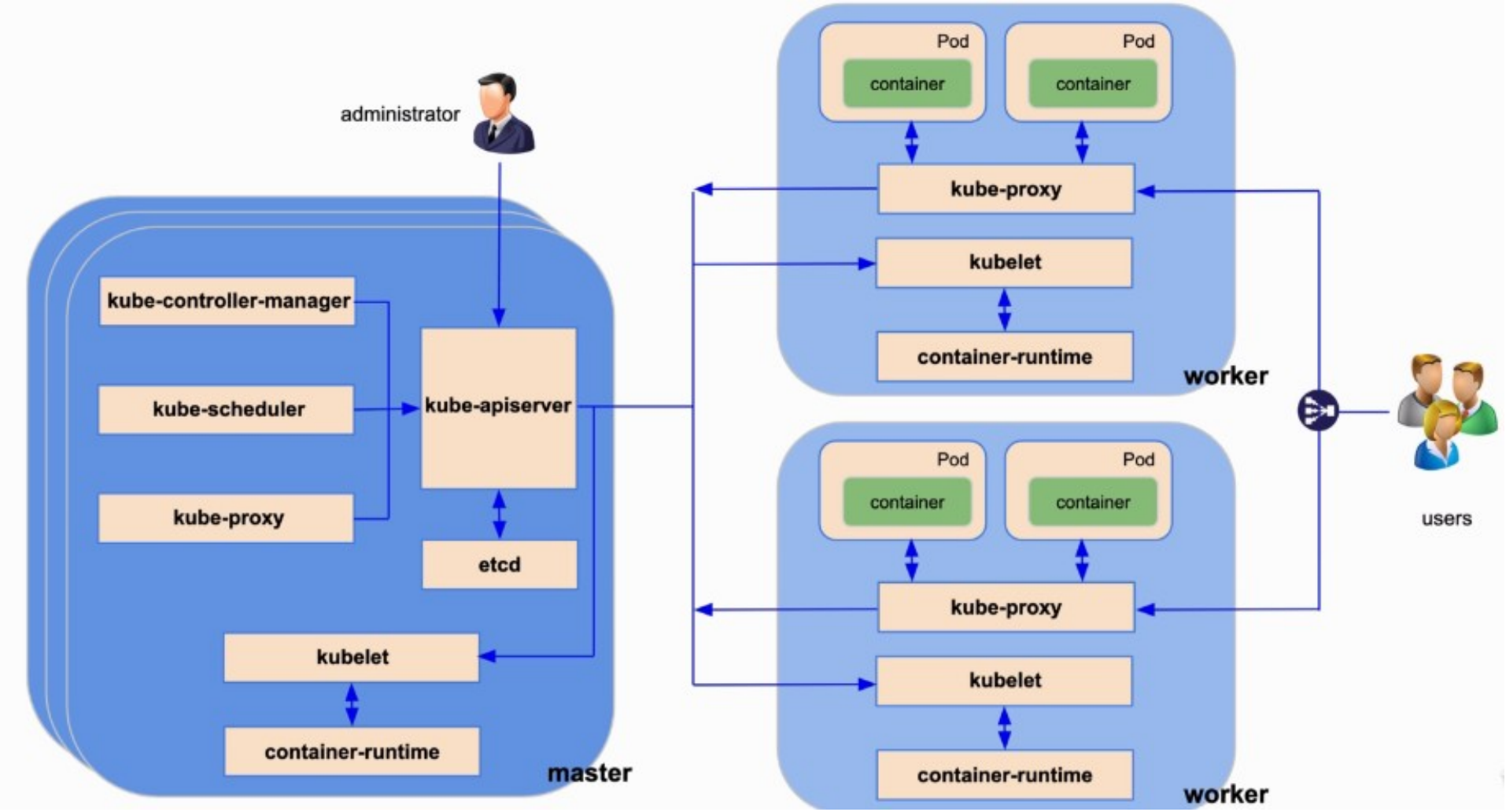
DAWAN - Reproduction interdite

Architecture K8s



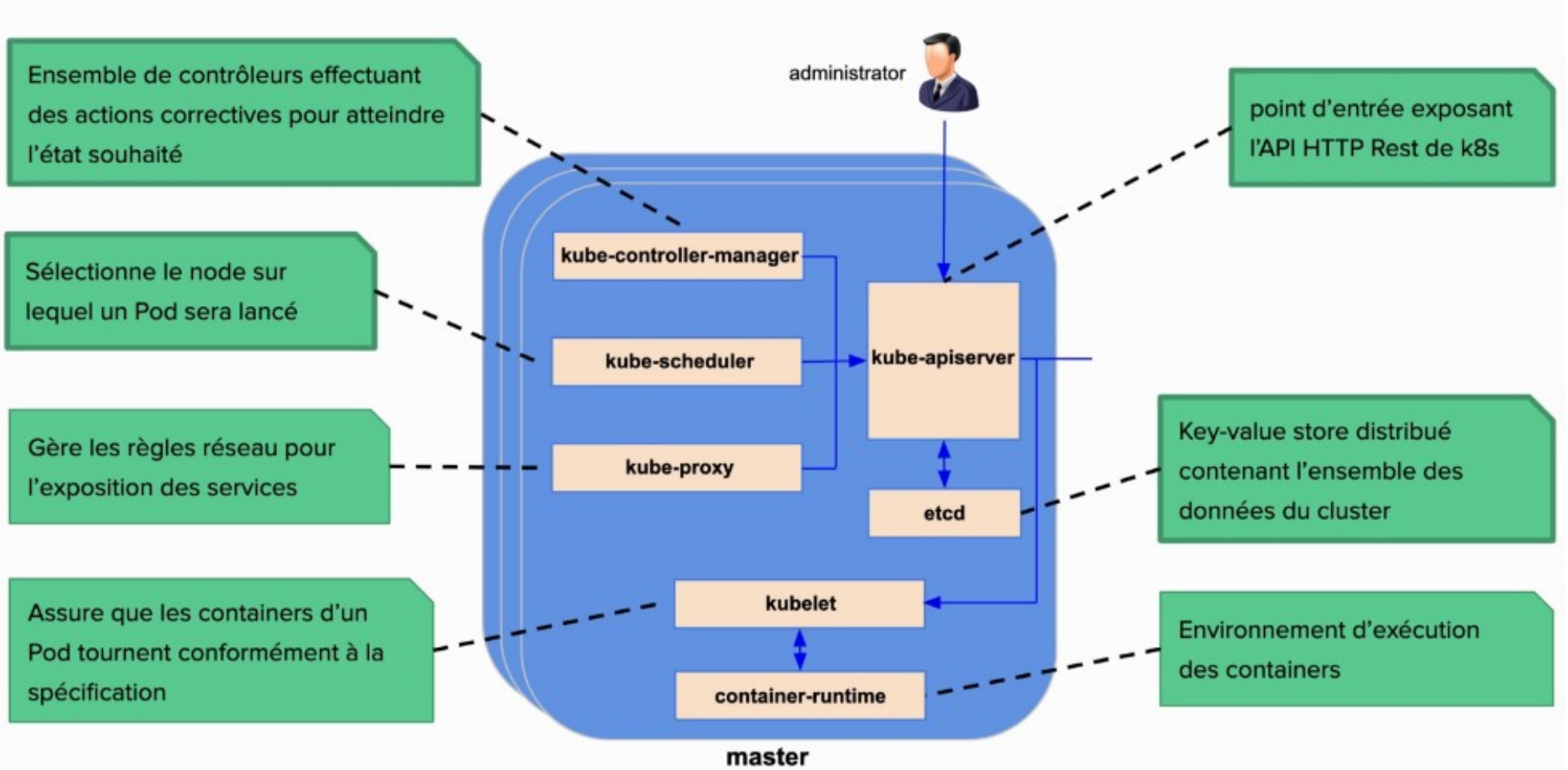
DAWAN - Reproduction interdite

Architecture K8s

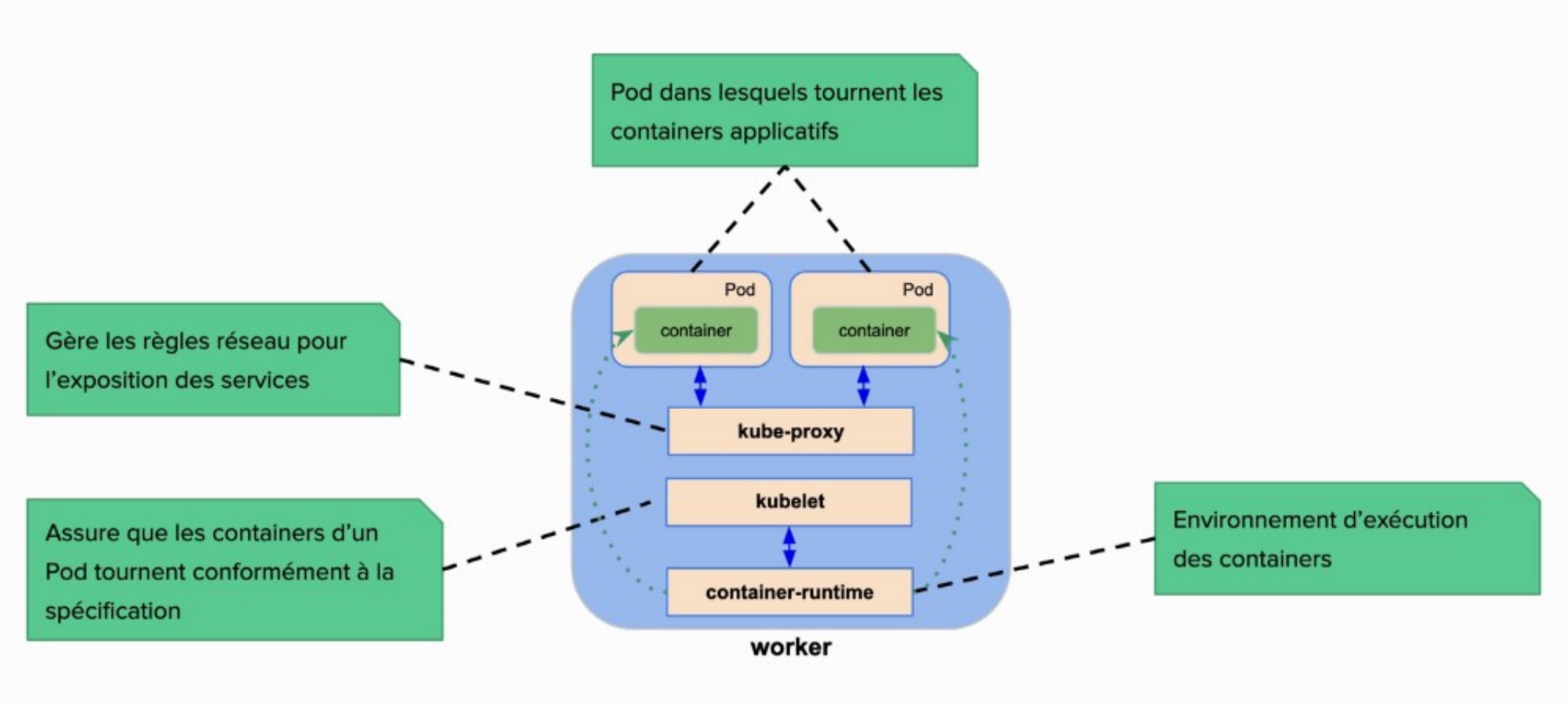


DAWAN - Reproduction interdite

Focus : Architecture master



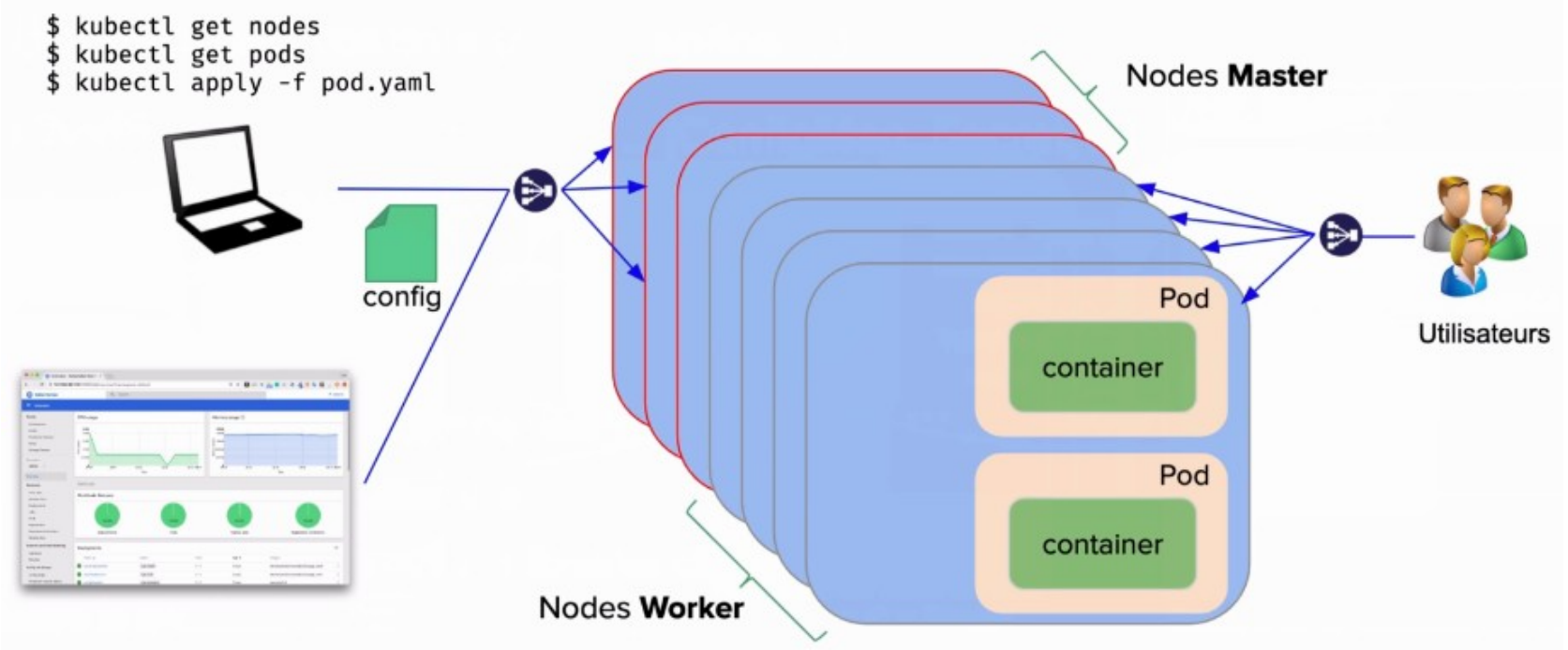
Focus : Architecture worker



DAWAN - Reproduction interdite

Communiquer avec le cluster

- Binaire kubectl



- Interface web

Les outils

- **Minikube :** Tous les composants Kubernetes dans une VM locale
  S'intègre avec différents hyperviseurs (Hyper-V, VirtualBox, VMWare)

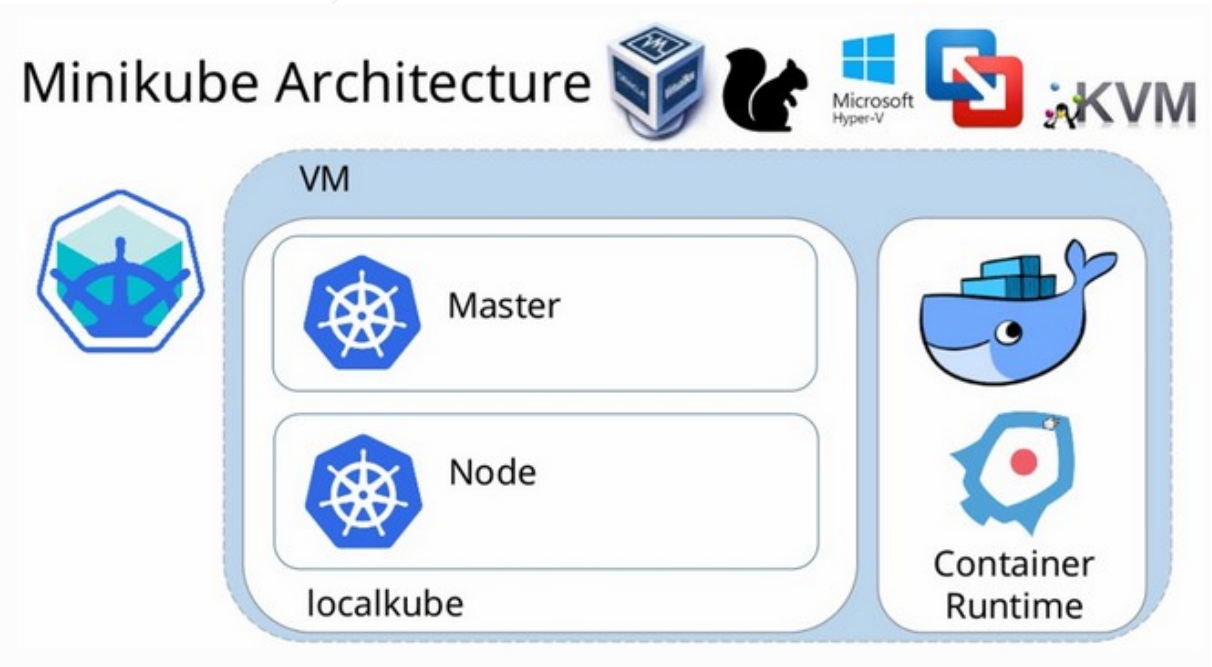
  https://kubernetes.io/docs/tasks/tools/
- **Docker Desktop :** DockerHub, choix orchestrateur (swarm ou K8S)

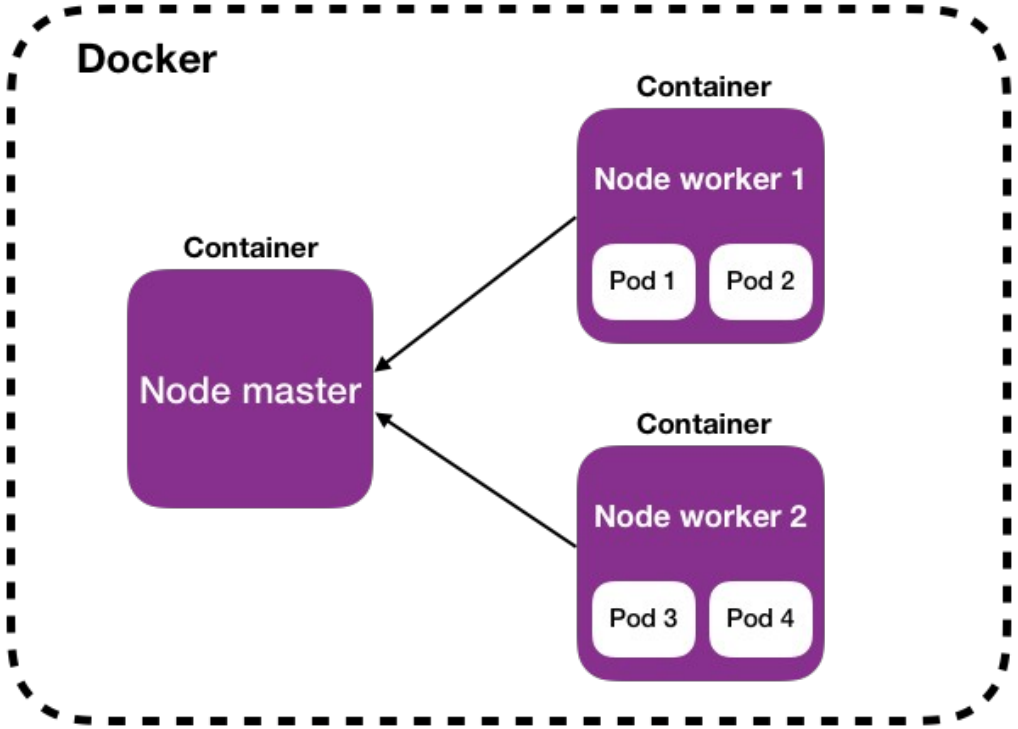
- **Kind** : Les nodes tournent dans des containers Docker
  https://kind.sigs.k8s.io/docs/user/quick-start/

- **MicroK8s**
- **K3S**

DAWAN - Reproduction interdite

Minikube


Minikube Architecture
Microsoft Hyper-V
KVM
VM
Master
Node
localkube
Container Runtime


Docker
Container
Node worker 1
Pod 1
Pod 2
Container
Node master
Container
Node worker 2
Pod 3
Pod 4

Cluster Kubernetes en Production

- **Managé :** Cluster administré par une entreprise tierce
  GKE : Google Kubernetes Engine
  AKS : Azure Container Service
  EKS : Amazon Elastic Container Serice
  OVH, DigitalOcean...

- **Perso:** kubeadm
  kops
  kubespray
  rancher
  Terraform + Ansible
  Docker EE (deploy Swarm et Kubernetes)

DAWAN - Reproduction interdite

TP : Premiers pas : Installation de kubectl & minikube

https://kubernetes.io/fr/docs/tasks/tools/install-kubectl/

https://kubernetes.io/fr/docs/setup/learning-environment/minikube/

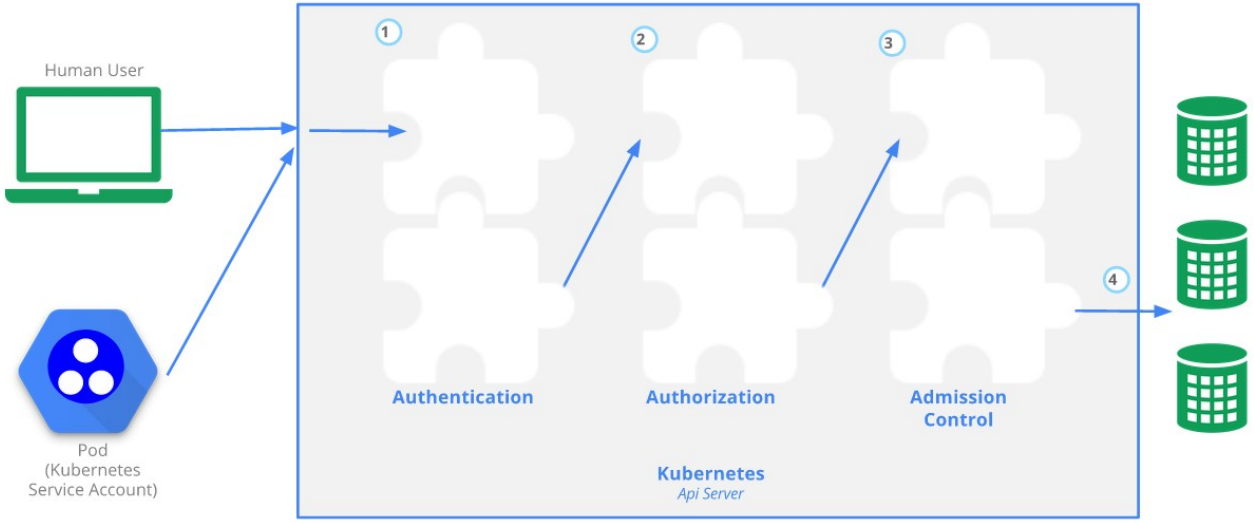
API Server

- **API REST :**

  https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.18/

- **Clients :** kubectl, interface web, application tournant dans le cluster

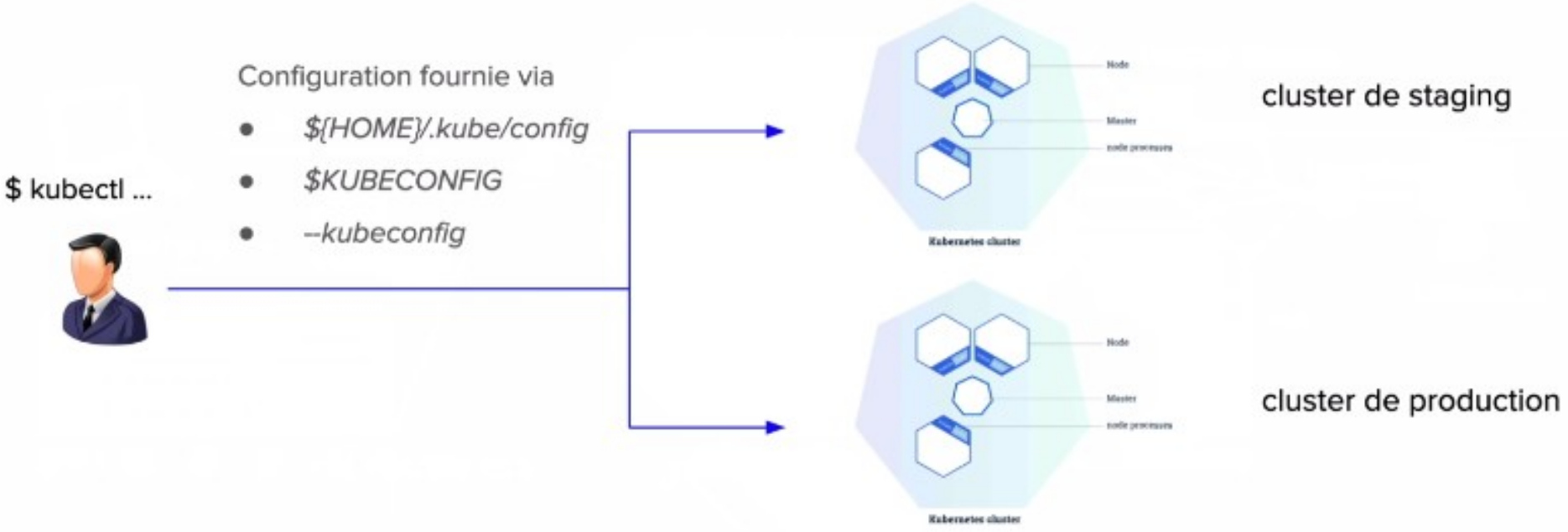
- **Pipeline des requêtes :**
  - authentification
  - autorisation
  - adminssion contrôleurs



DAWAN - Reproduction interdite

Contexte d’utilisation

- Définition du cluster auquel on s’adresse



Contexte d’utilisation

- Outils :

https://github.com/ahmetb/kubectx

```
USAGE:
  kubectx                    : list the contexts
  kubectx <NAME>             : switch to context <NAME>
  kubectx -                  : switch to the previous context
  kubectx -c, --current      : show the current context name
  kubectx <NEW_NAME>=<NAME> : rename context <NAME> to <NEW_NAME>
  kubectx <NEW_NAME>=.       : rename current-context to <NEW_NAME>
  kubectx -d <NAME>          : delete context <NAME> ('.' for current-context)
                               (this command won't delete the user/cluster entry
                               that is used by the context)
  kubectx -u, --unset        : unset the current context
```

DAWAN - Reproduction interdite

Kubectl - CLI

Binaire utilisé pour communiquer avec l'API server

Requêtes HTTP

Collecter des informations sur un dispositif

> kubectl get deployments | services | pods | nodes

Créer une nouvelle ressources

> Kubectl apply | create | run

Supprimer une ressources

> kubectl delete

Établir la connectivité

> Kubectl expose

Information sur les clusters à dispositions

> kubectl cluster-info

DAWAN - Reproduction interdite

Bonnes pratiques

$ kubectl api-resources

$ kubectl explain

> bash-completion : source <(kubectl completion bash)

> JSONPath

> custom-columns

--validate=true

--dry-run=client

Communiquer, communiquer … communiquer !

Les conventions : https://kubernetes.io/fr/docs/reference/kubectl/conventions/

DAWAN - Reproduction interdite

Processus : vision d’ensemble

```
$ kubectl get pod -n kube-system
NAME                                    READY   STATUS    RESTARTS   AGE
coredns-86c58d9df4-q84tg                1/1     Running   0          5m
coredns-86c58d9df4-qm6sx                1/1     Running   0          5m
etcd-k8s-node-1                         1/1     Running   0          4m
kube-apiserver-k8s-node-1               1/1     Running   0          4m
kube-controller-manager-k8s-node-1      1/1     Running   0          4m
kube-proxy-nc9cb                        1/1     Running   0          5m
kube-proxy-wll56                        1/1     Running   0          2m
kube-proxy-x77gd                        1/1     Running   0          2m
kube-scheduler-k8s-node-1               1/1     Running   0          4m
weave-net-29x5x                         2/2     Running   0          30s
weave-net-77ppt                         2/2     Running   0          30s
weave-net-q7fb5                         2/2     Running   0          30s
```

Processus tournant sur les masters

Processus tournant sur chacun des nodes (masters et workers)

Processus répliqués sur le cluster (ex: serveur DNS add-on)

DAWAN - Reproduction interdite

K8s : Concepts

Node : Un node ("nœud" en fr) est une machine de travail du cluster Kubernetes. (physiques, virtuelles, instances cloud)

Pod : Il s'agit de l'unité la plus petite de K8s, un pod encapsule le ou les conteneur(s) partageant ainsi la même stack réseau et le même stockage

Replicas : c'est le nombre d'instances d'un Pod ("réplique" en fr)

ReplicaSet : s'assure que les réplicas spécifiés sont actifs

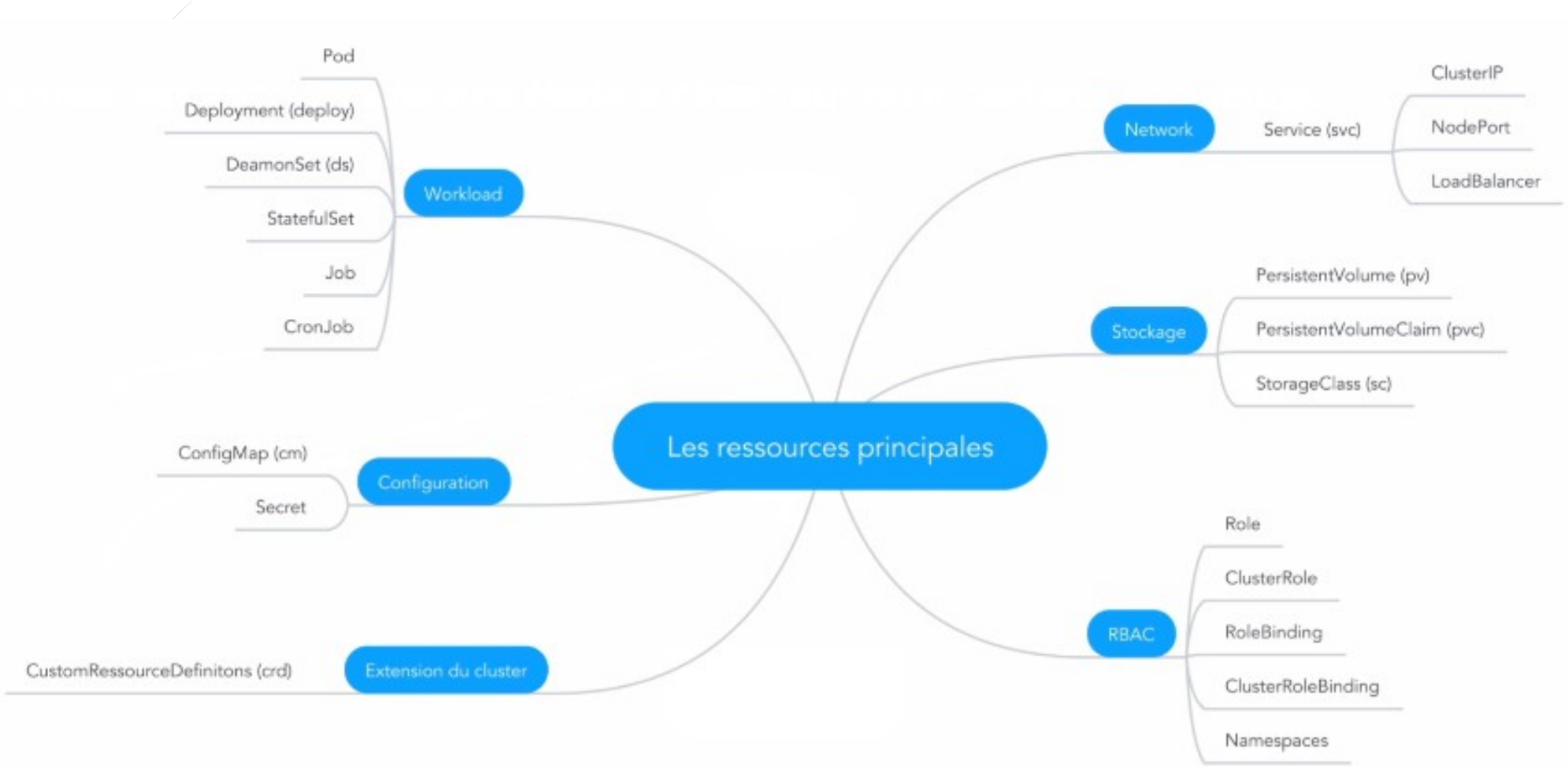
Deployment : défini l'état désiré et fournit des mises à jour déclaratives de vos Pods et ReplicaSets.

Service : service réseau. C'est un niveau d'abstraction au-dessus du pod, qui fournit un point d'accès unique pour un ensemble de pods.

Endpoint : Représente l'adresse IP et le port d'un service, il est automatiquement créé lors de la création d'un service avec les pods correspondants.

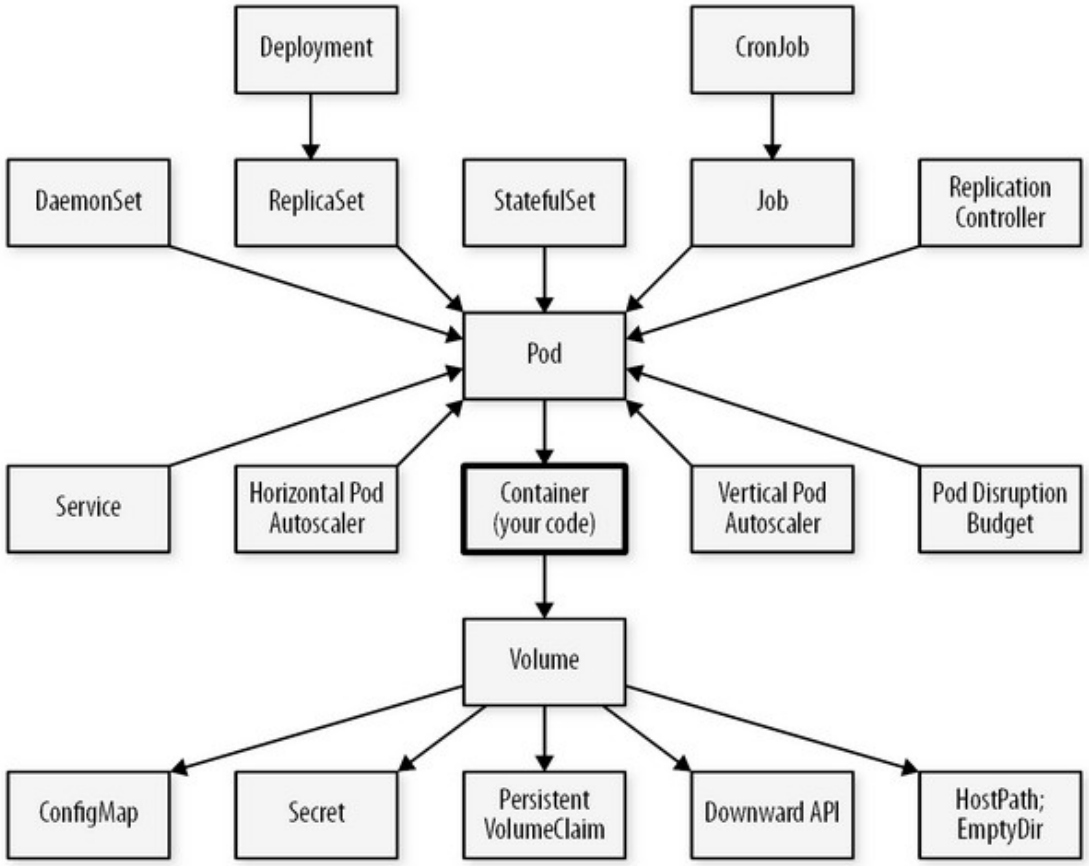
DAWAN - Reproduction interdite

K8s : les ressources – objets fondamentaux



DAWAN - Reproduction interdite

K8s : les ressources – objets fondamentaux



DAWAN - Reproduction interdite

K8s : les ressources

- **Gestion des applications :** Deployment, Pod, Job, DaemonSet, StatefulSet

- **Discovery et Load Balancing :** Service

- **Configuration des applications :** ConfigMap, Secrets...

- **Stockage :** PersistentVolume, PersistentVolumeClaim, Namespace...

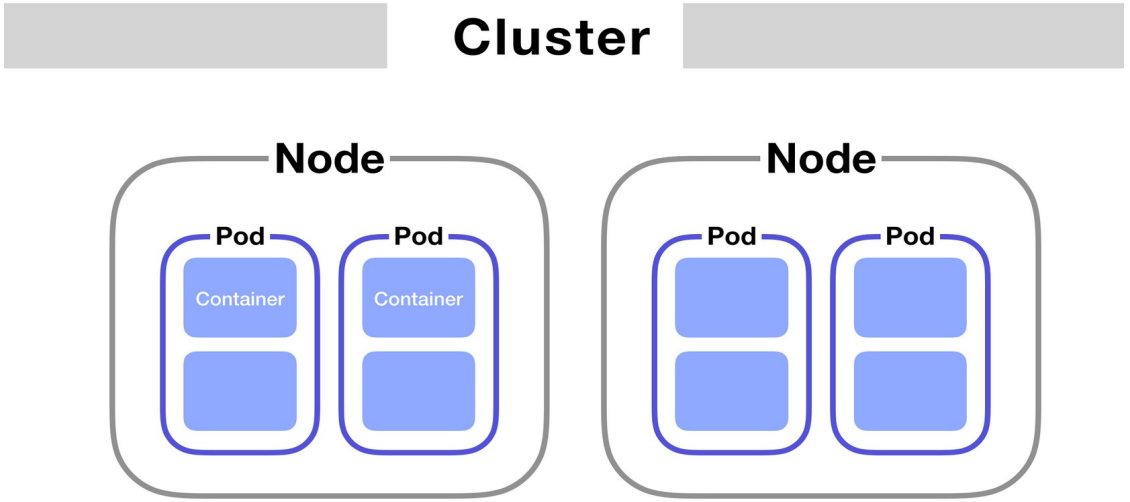
- **Configuration Cluster et metadata :** Role, ClusterRole

- **Objets supplémentaires pour ajout de fonctionnalités :** CRD

DAWAN - Reproduction interdite

K8s : Concepts fondamentaux

Pods: Unité d'exécution de base d'une application Kubernetes

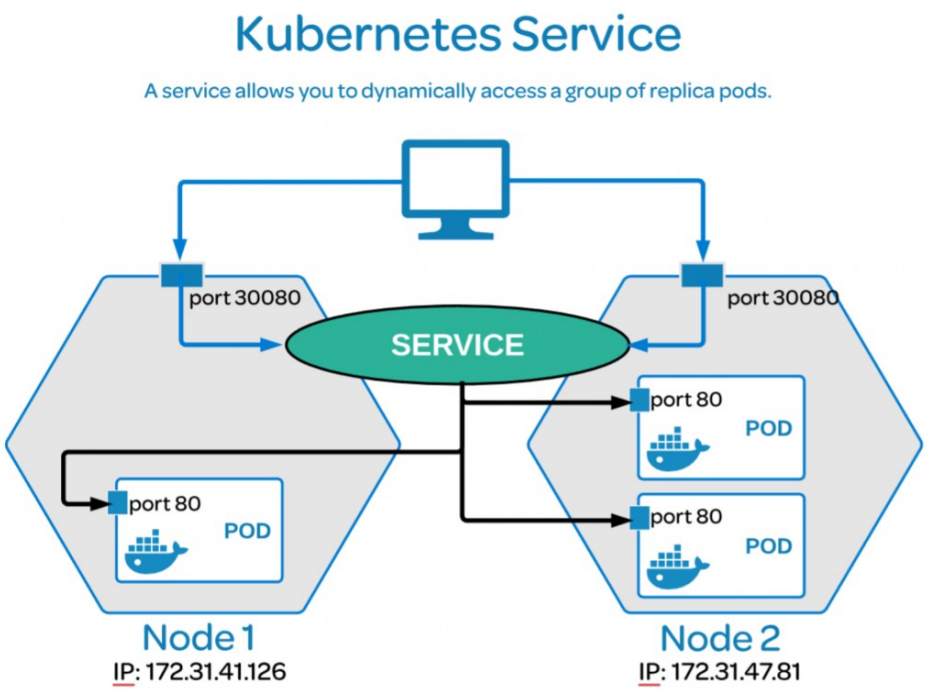


- Unité la plus petite et la plus simple dans le modèle d'objets de Kubernetes
- Modèle « un-conteneur-par-Pod » ou « groupe de conteneurs »

DAWAN - Reproduction interdite

K8s : Concepts

Service: Expose les applications des Pods similaires à l'intérieur ou à l'extérieur du cluster

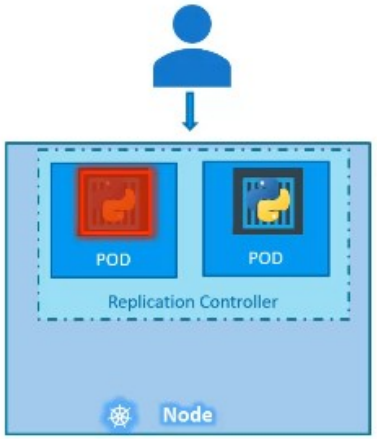


- Défini des règles réseaux pour l'accès aux pods et répartir la charge

DAWAN - Reproduction interdite

K8s : Concepts

ReplicatSets: Maintenir un ensemble stable de Pods

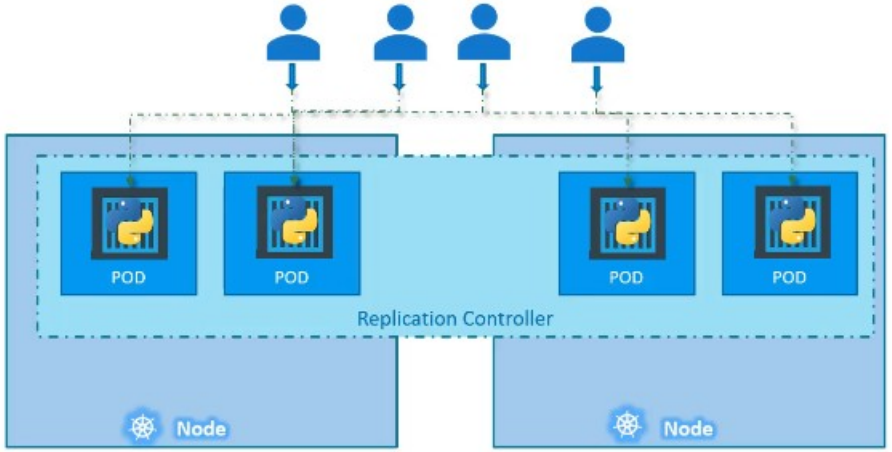


- Nous aide à exécuter plusieurs instances d'un seul POD dans le cluster kubernetes
- Même avec un seul node, création automatique d’un nouveau POD lorsque l'existant échoue
- Garantie que le nombre spécifié de POD fonctionne à tout moment. Même si c'est juste 1 ou 100.

DAWAN - Reproduction interdite

K8s : Concepts

ReplicatSets: Partager/Equilibrer la charge entre plusieurs chemins/nodes



- **Deux termes :** Replication Controler & ReplicatSet

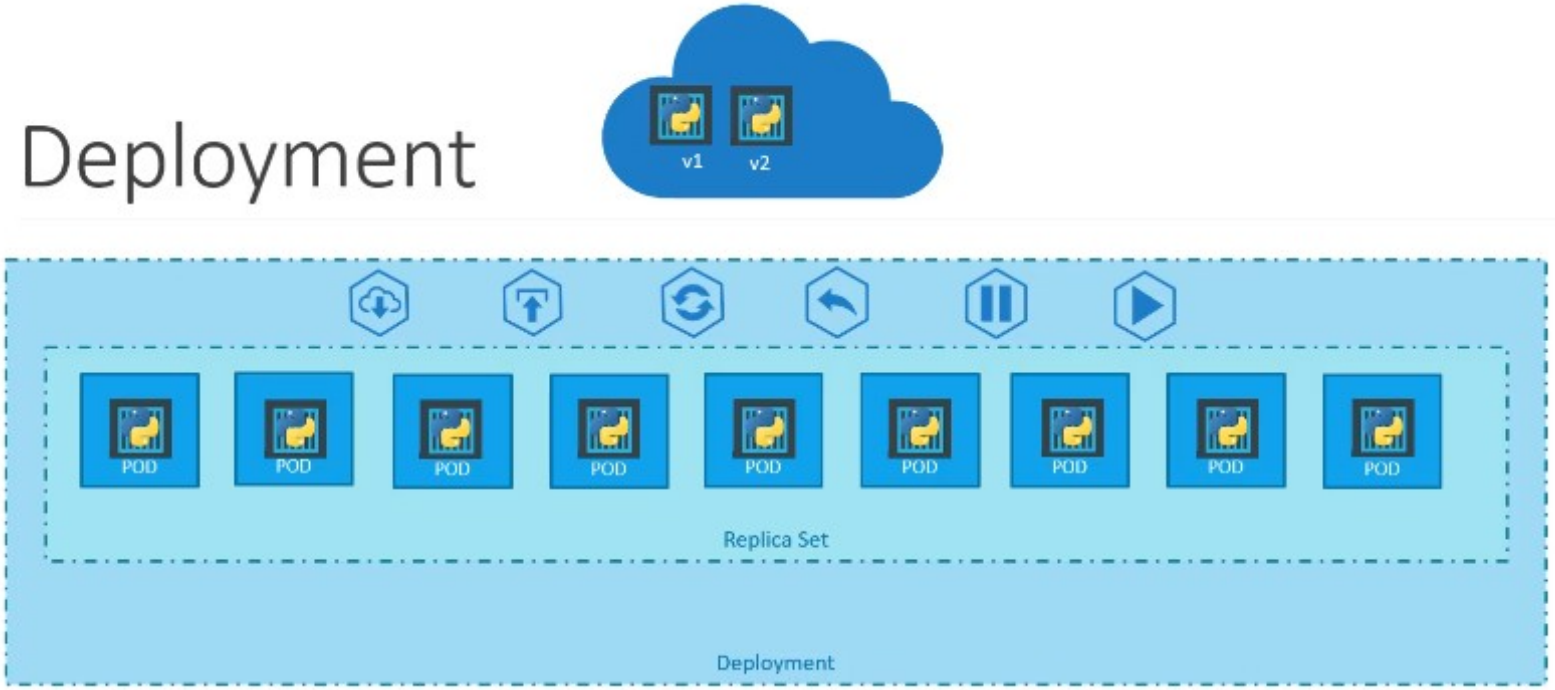
=> nouvelle méthode recommandée

- Permet de déployer des pods supplémentaires pour équilibrer la charge
  - sollicitation utilisateurs
  - manque de ressources sur un node

K8s : Concepts

Deployment : Permet de gérer un ensemble de règles, mises à jours déclaratives pour Pods et ReplicatSets

(réplicas / mise à jours (images) / rollback)



K8s : Labels et annotations

- Informations attachées aux ressources

- Format : key / value

- Labels

- utilisés pour la sélection d’objets
- récupération de collections

- Annotations :

- non internes à Kubernetes
- utilisées par des outils et librairies clientes

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: www
  annotations:
    ingress.kubernetes.io/rewrite-target: /
  labels:
    app: www
...
```

DAWAN - Reproduction interdite

K8s : Spécification des ressources

- Généralement en **YAML**

- **Structure commune :**
  - apiVersion
  - kind
  - metadata
  - spec

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: www
  labels:
    app: w3
spec:
  containers:
  - name: www
    image: nginx:1.16
```

Exemple de spécification d'un Pod

```yaml
apiVersion: v1
kind: Service
metadata:
  name: www
spec:
  selector:
    app: w3
  ports:
  - port: 80
    targetPort: 80
```

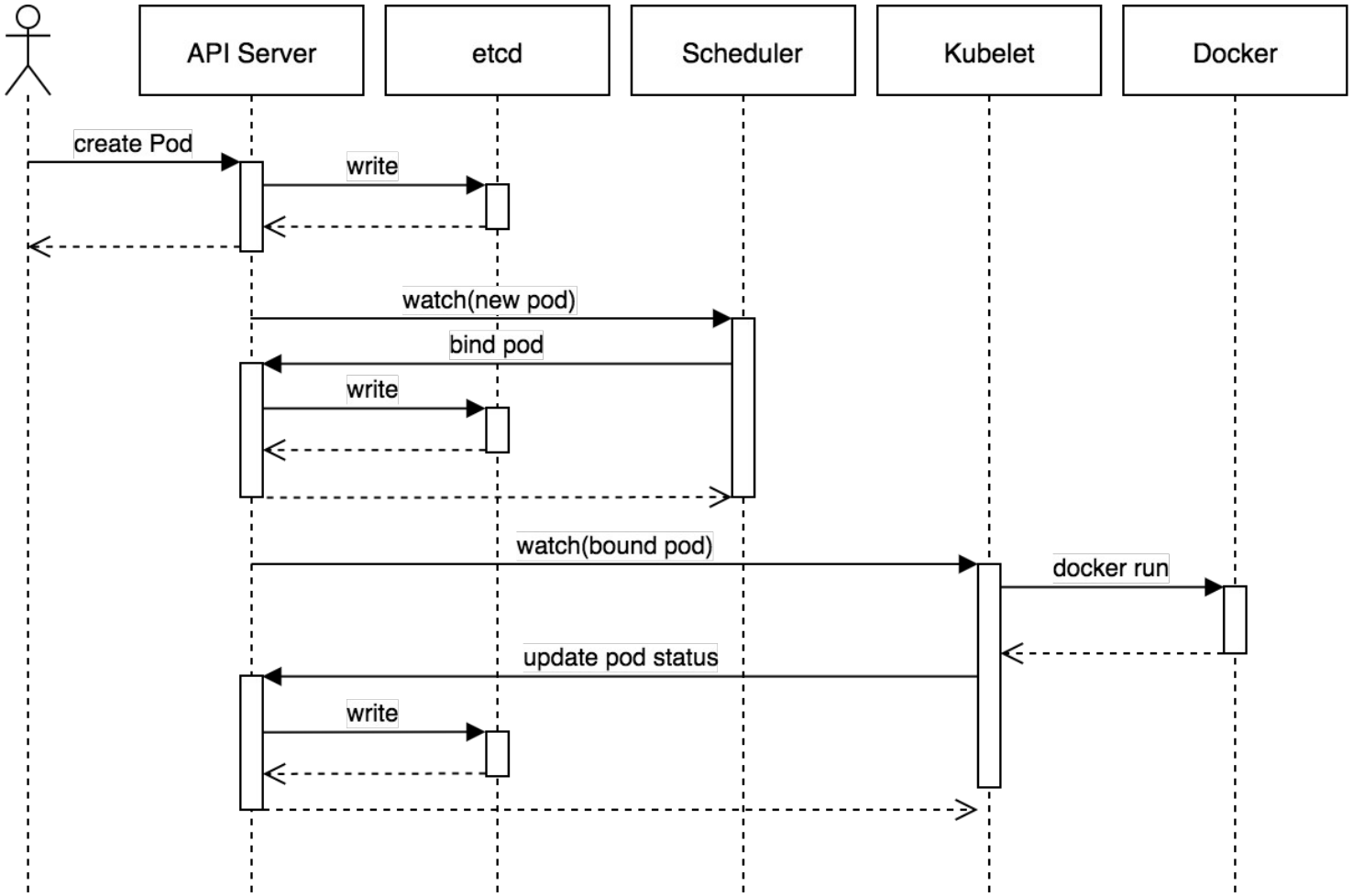
Exemple de spécification d'un Service

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: www
spec:
  replicas: 3
  selector:
    matchLabels:
      app: w3
  template:
    metadata:
      labels:
        app: w3
    spec:
      containers:
      - name: www
        image: nginx:1.16
```

Exemple de spécification d'un Deployment

DAWAN - Reproduction interdite

Création POD : workflow

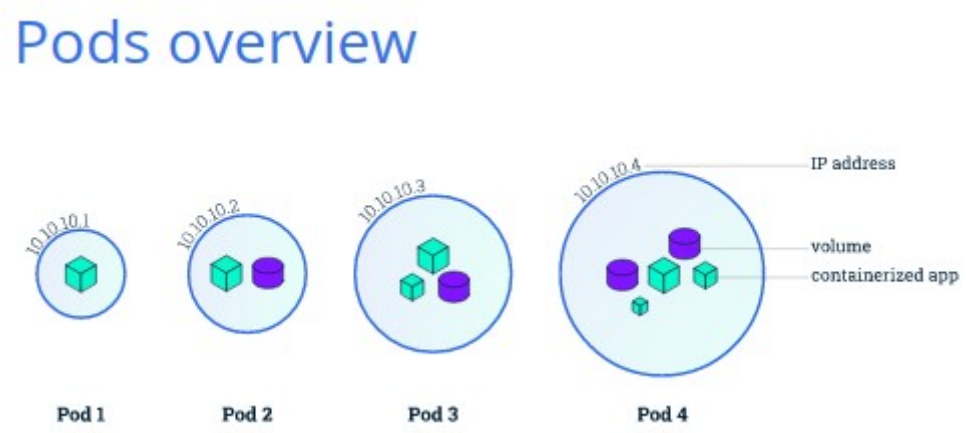


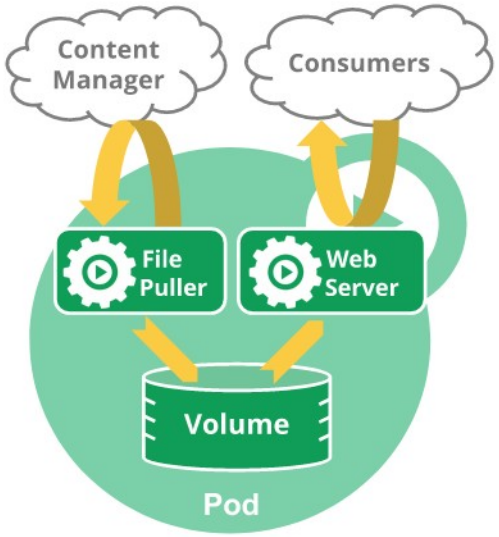
DAWAN - Reproduction interdite

Pods

- **Plus petite unité applicative dans K8s**
  - Groupe de containers : même contexte d’isolation (namespace)
  - Partage de stack réseau et stockage
  - Adresse IP dédiée

- **Mono container ou Multi containers**
- **Un pod modélise un "hôte logique" spécifique à une application**





DAWAN - Reproduction interdite

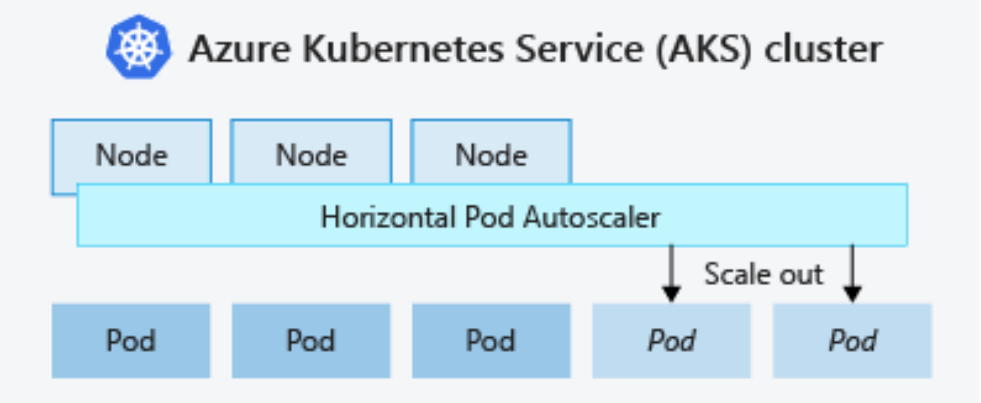
Pods

- Application découpée en plusieurs spécifications de Pods

- Chaque spécification correspond à un service metier (microservice)

- Permet le scaling horizontal via le nbre de réplicas d'un Pod

Création de nouveaux Pods basés sur la même spécification



DAWAN - Reproduction interdite

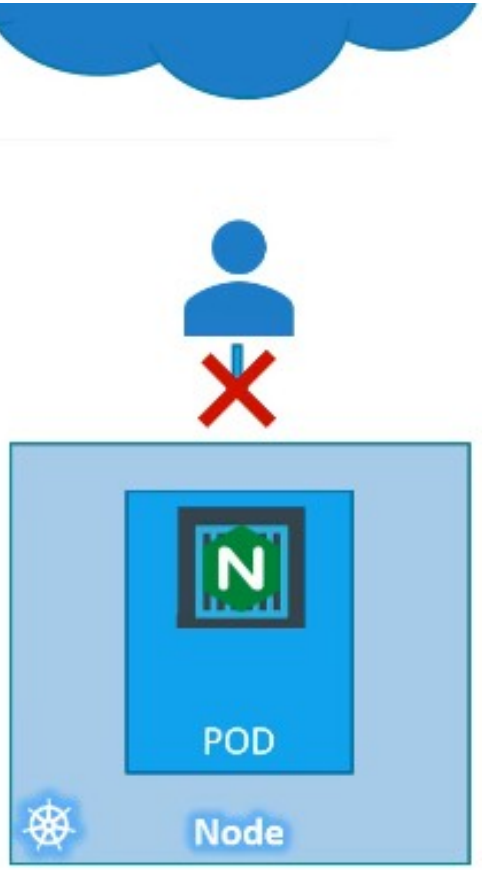
K8s : Test

kubectl

```
kubectl run nginx --image nginx
```

```
kubectl get pods
NAME    READY   STATUS              RESTARTS   AGE
nginx   0/1     ContainerCreating   0          6s
```

```
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          34s
```



DAWAN - Reproduction interdite

Pods – exemple de spécification

```yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
    - name: nginx
      image: nginx:1.19
```

DAWAN - Reproduction interdite

Pods – cycle de vie

- Lancement d’un Pod

$ kubectl create -f POD_SPEC.yml

- Liste des Pods

$ kubectl get pod (namespace default)

- Description d’un Pod

$ Kubectl describe *pod POD_NAME*

$ kubectl describe *po/POD_NAME*

- Logs d’un container d’un Pod

$ kubectl logs POD_NAME [-c CONTAINER_NAME]

- Lancement d’une commande dans un Pod existant

$ kubectl exec POD_NAME [-c CONTAINER_NAME] – COMMAND

- Suppression d’un Pod

$ kubectl delete pod POD_NAME

DAWAN - Reproduction interdite

Pods – Forward de port

- Commande utilisée pour le développement et le debugging

- Permet de publier le port d'un Pod sur la machine hôte

$ kubectl port-forward *POD_NAME HOST_PORT:CONTAINER_PORT*

```
pierre  ~  kubectl port-forward nginx 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
```

DAWAN - Reproduction interdite

Pods – multi containers

- Exemple Wordpress

- Deux containers

> moteur wordpress

> base de données MySQL

- Volume pour persistence de données de la bdd

> type : emptyDir : associé au cycle de vie du Pod

DAWAN - Reproduction interdite

TP : Pods – multi containers Wordpress

```yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: wp
spec:
  containers:
    - image: wordpress:4.9-apache
      name: wordpress
      env:
        - name: WORDPRESS_DB_PASSWORD
          value: mysqlpwd
        - name: WORDPRESS_DB_HOST
          value: 127.0.0.1
      resources:
        limits:
          memory: "128Mi"
          cpu: "500m"
    - image: mysql:5.7
      name: mysql
      env:
        - name: MYSQL_ROOT_PASSWORD
          value: mysqlpwd
      volumeMounts:
        - name: data
          mountPath: /var/lib/mysql
      resources:
        limits:
          memory: "256Mi"
          cpu: "500m"
  volumes:
    - name: data
      emptyDir: {}
```

DAWAN - Reproduction interdite

Pods – Scheduling

- **Sélection du node sur lequel un Pod sera déployé**

- Composant **kube-scheduler**

```
pierre  ~  kubectl run www --image=nginx:1.19 --restart=Never
pod/www created
```

```
pierre  ~  kubectl describe pod www
```

```
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  11s   default-scheduler  Successfully assigned default/www to minikube-m04
  Normal  Pulled     10s   kubelet            Container image "nginx:1.19" already present on machine
  Normal  Created    10s   kubelet            Created container www
  Normal  Started    10s   kubelet            Started container www
```

Pods – nodeSelector

- **Permet de scheduler un Pod sur un node ayant un label spécifique**

```
pierre  ~  kubectl get nodes
NAME           STATUS   ROLES                  AGE    VERSION
minikube       Ready    control-plane,master   4d5h   v1.20.2
minikube-m02   Ready    <none>                 4d5h   v1.20.2
minikube-m03   Ready    <none>                 4d5h   v1.20.2
minikube-m04   Ready    <none>                 4d5h   v1.20.2
pierre  ~  # Ajout d'un label sur le node minikue-m02
pierre  ~  kubectl label nodes minikube-m02 disktype=ssd
node/minikube-m02 labeled
pierre  ~  kubectl get node/minikube-m02 -o yaml
apiVersion: v1
kind: Node
metadata:
  annotations:
    kubeadm.alpha.kubernetes.io/cri-socket: /var/run/dockershim.sock
    node.alpha.kubernetes.io/ttl: "0"
    volumes.kubernetes.io/controller-managed-attach-detach: "true"
  creationTimestamp: "2021-02-04T08:43:42Z"
  labels:
    beta.kubernetes.io/arch: amd64
    beta.kubernetes.io/os: linux
    disktype: ssd
```

Génération de la spécification d’un Pod

--dry-run : simule la création d’une ressource

- version < 1.18 => s’utilise sans valeur
- version ≥ 1.18 => 2 valeurs possibles
  * client : la ressource n’est pas envoyée à l’API server
  * server : traitée par l’API Server mais non persistée

```
pierre  ~  kubectl run db --image mong:4.0 --dry-run=client -o yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: db
  name: db
spec:
  containers:
  - image: mong:4.0
    name: db
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
```

DAWAN - Reproduction interdite

Mise en réseau - Service

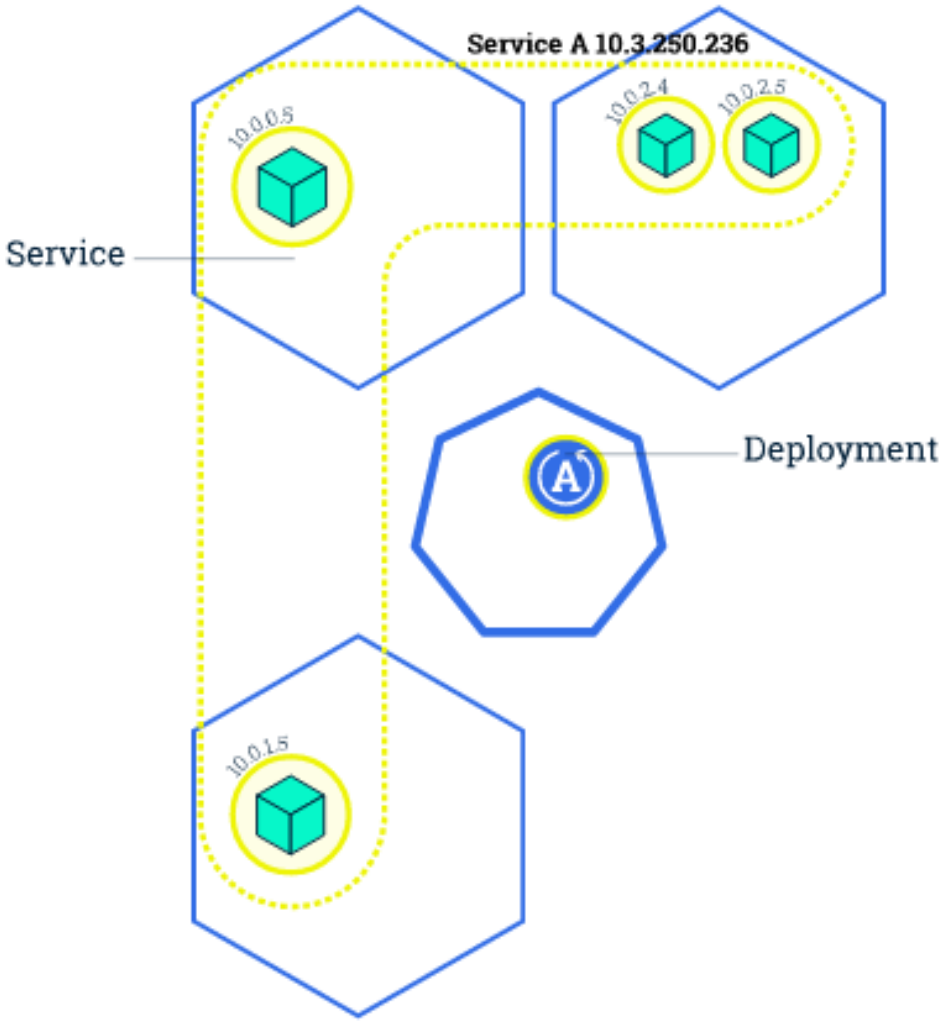
- **Expose les Pods d'une application via des règles réseau**
  * collection de Pods exposée à travers un **endpoint**
- **Utilise des labels pour grouper les Pods**
- **Adresse IP persistante (VIP : Virtual IP Adress)**

- **kube-proxy : en charge du load-balancing sur les Pods**
  * **userspaces / iptables / IPVS**

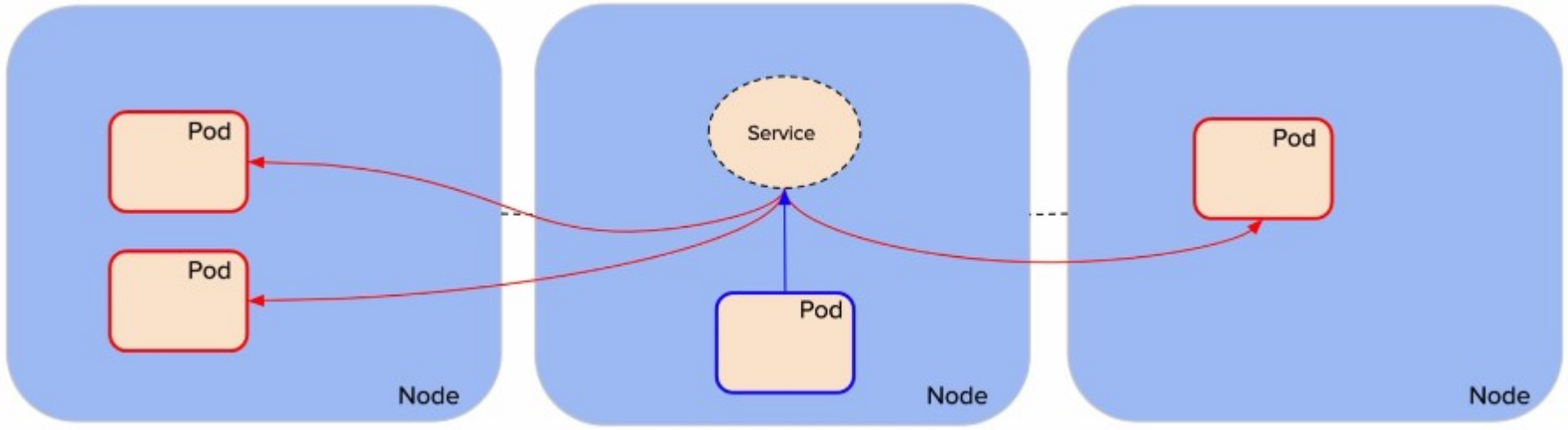
- **Différents types :**
  * **ClusterIP :** [défaut] exposition à l'intérieur du cluster
  * **NodePort :** exposition vers l'extérieur du cluster (NAT)
  * **LoadBalancer :** utilise l'équilibeur de charge des fournisseurs cloud
  * **ExternalName :** associe le service à un nom DNS (redirection CNAME)

DAWAN - Reproduction interdite

Mise en réseau - Service



Service - ClusterIP



DAWAN - Reproduction interdite

Service - ClusterIP

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: vote
  labels:
    app: vote
spec:
  containers:
  - name: vote
    image: instavote/vote
    ports:
    - containerPort: 80
```

pod.yaml

kubectl apply -f pod.yaml

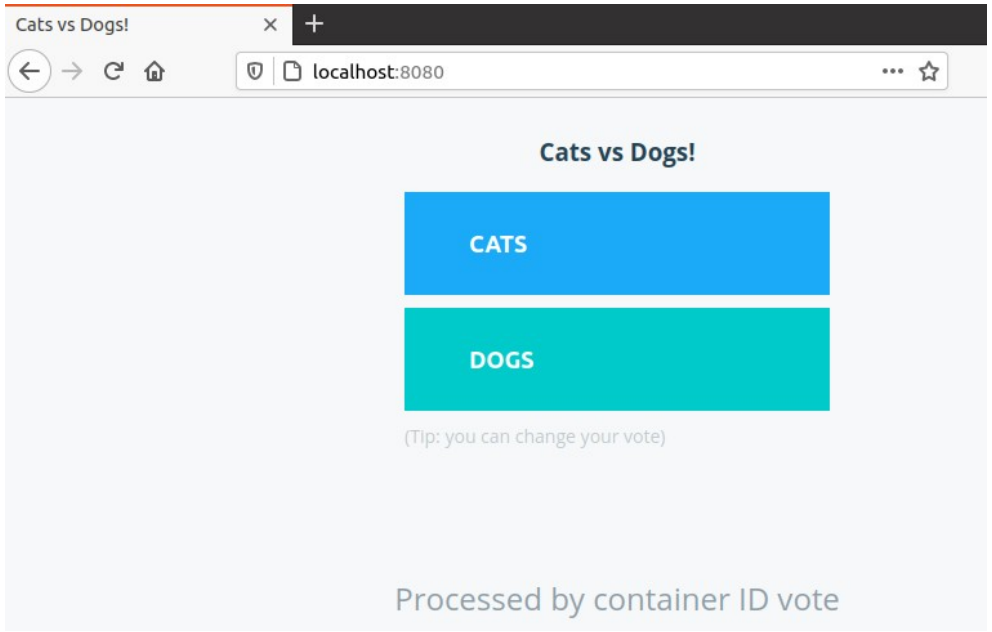
```yaml
apiVersion: v1
kind: Service
metadata:
  name: vote
spec:
  selector:
    app: vote
  type: ClusterIP
  ports:
  - port: 80
    targetPort: 80
```

service.yaml

kubectl apply -f service.yaml

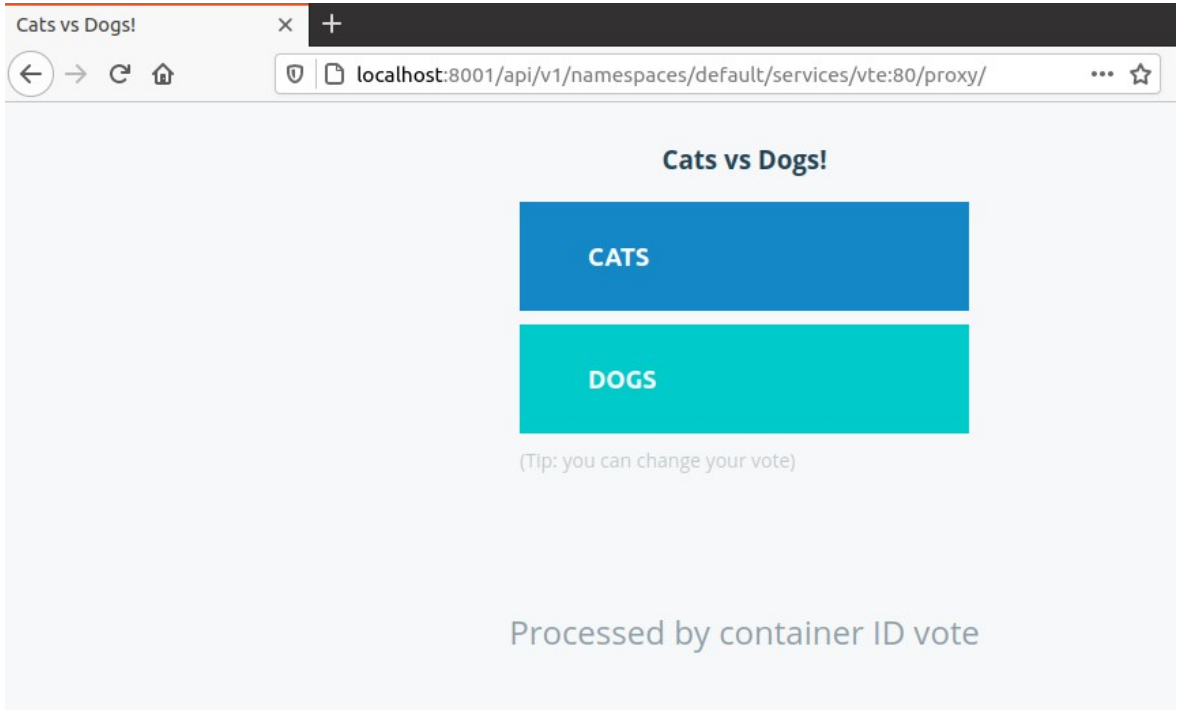
Service – ClusterIP + portforward

```
pierre  ~  # Accès au service depuis l'extérieur
pierre  ~  kubectl port-forward svc/vote 8080:80
```



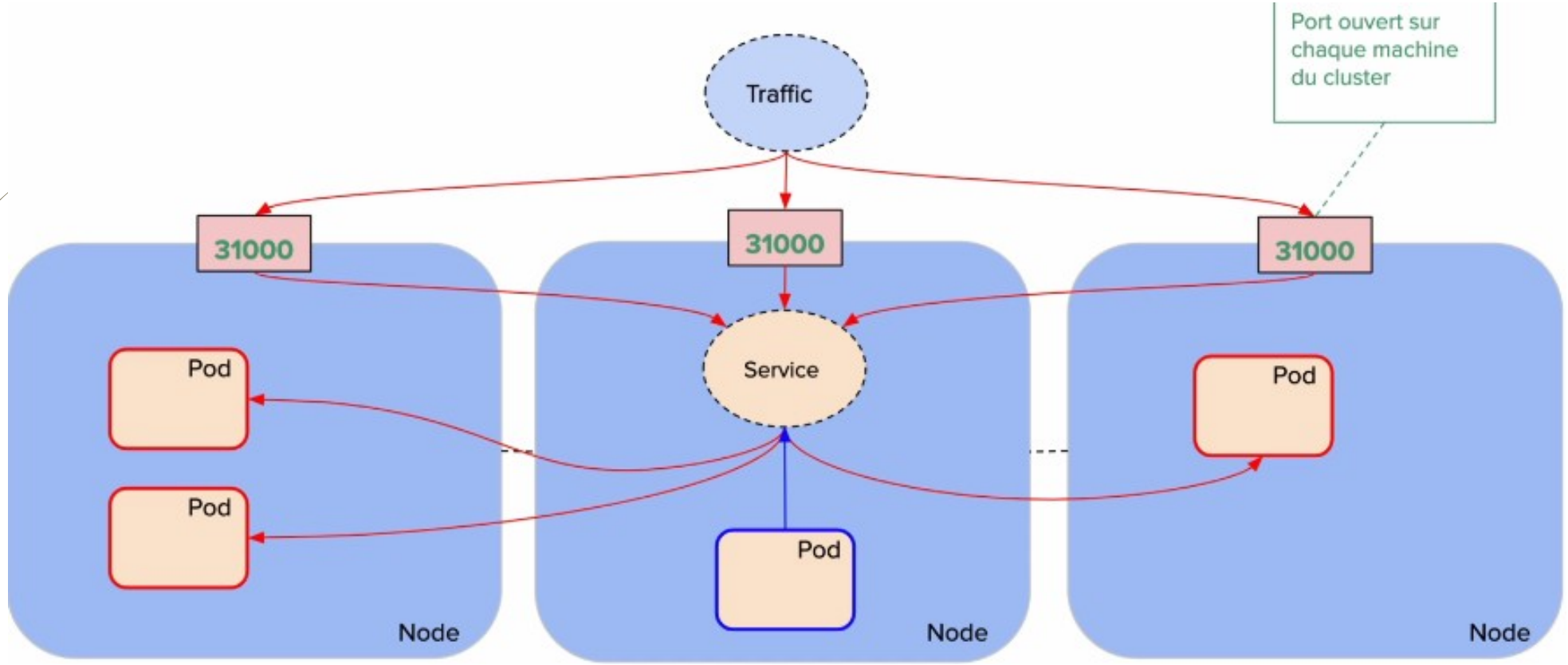
Service – ClusterIP + proxy

```
pierre  ~  kubectl proxy
Starting to serve on 127.0.0.1:8001
```



DAWAN - Reproduction interdite

Service – NodePort



Service – NodePort

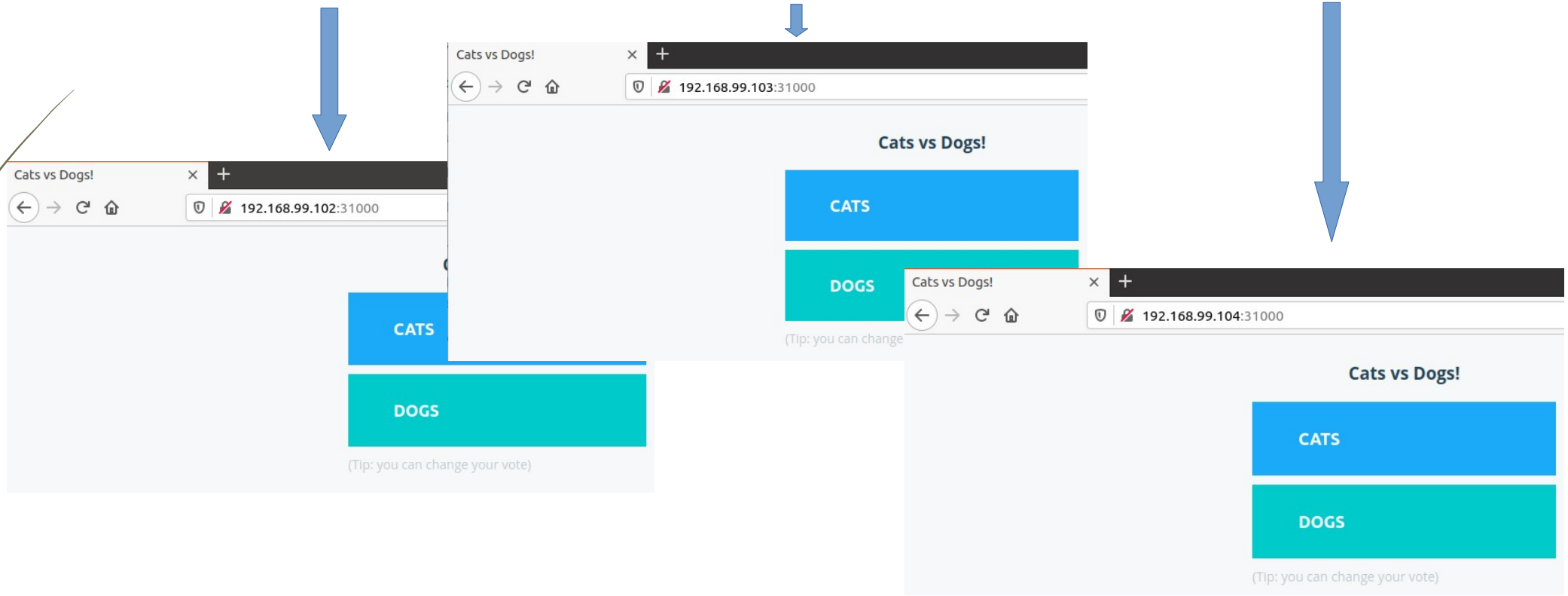
```yaml
apiVersion: v1
kind: Service
metadata:
  name: vote-np
spec:
  selector:
    app: vote
  type: NodePort
  ports:
  - port: 80
    targetPort: 80
    nodePort: 31000
```

service-np.yaml

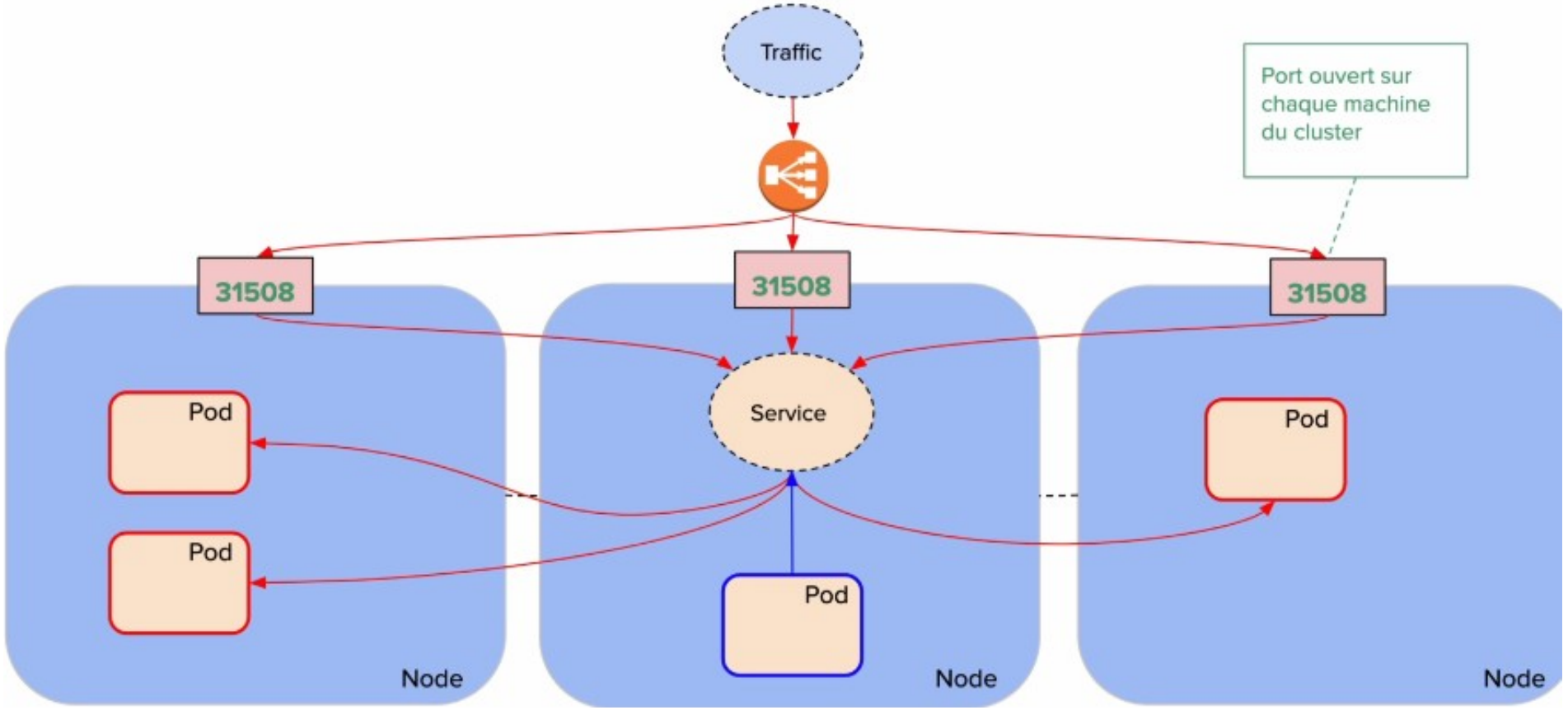
```
$ kubectl apply -f service-np.yaml
```

Service – NodePort

```
pierre  ~  kubectl get nodes -o wide
NAME           STATUS   ROLES                  AGE    VERSION   INTERNAL-IP      EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION   CONTAINER-RUNTIME
minikube       Ready    control-plane,master   4d7h   v1.20.2   192.168.99.101   <none>        Buildroot 2020.02.8   4.19.157         docker://20.10.2
minikube-m02   Ready    <none>                 4d7h   v1.20.2   192.168.99.102   <none>        Buildroot 2020.02.8   4.19.157         docker://20.10.2
minikube-m03   Ready    <none>                 4d7h   v1.20.2   192.168.99.103   <none>        Buildroot 2020.02.8   4.19.157         docker://20.10.2
minikube-m04   Ready    <none>                 4d7h   v1.20.2   192.168.99.104   <none>        Buildroot 2020.02.8   4.19.157         docker://20.10.2
```



Service – LoadBalancer



DAWAN - Reproduction interdite

Service – approche impérative

```
pierre  ~  kubectl expose pod whoami \
    --type=NodePort \
    --port=8080 \
    --target-port 80 \
    --dry-run=client \
    -o yaml
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: null
  labels:
    run: whoami
  name: whoami
spec:
  ports:
  - port: 8080
    protocol: TCP
    targetPort: 80
  selector:
    run: whoami
  type: NodePort
```

DAWAN - Reproduction interdite

Service – approche impérative

```
pierre  ~  kubectl create service \
    nodeport \
    whoami \
    --tcp 8080:80 \
    --dry-run=client \
    -o yaml
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: null
  labels:
    app: whoami
  name: whoami
spec:
  ports:
  - name: 8080-80
    port: 8080
    protocol: TCP
    targetPort: 80
  selector:
    app: whoami
  type: NodePort
```

DAWAN - Reproduction interdite

Service – approche impérative

```
pierre  ~  kubectl run db --image=mongo:4.2 --port=27017 --expose --dry-run=client -o yaml
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: null
  name: db
spec:
  ports:
  - port: 27017
    protocol: TCP
    targetPort: 27017
  selector:
    run: db
status:
  loadBalancer: {}
---
---
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: db
  name: db
spec:
  containers:
  - image: mongo:4.2
    name: db
    ports:
    - containerPort: 27017
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
```

DAWAN - Reproduction interdite

Service – approche impérative

```
pierre  ~  kubectl create service \
    nodeport \
    whoami \
    --tcp 8080:80 \
    --dry-run=client \
    -o yaml
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: null
  labels:
    app: whoami
  name: whoami
spec:
  ports:
  - name: 8080-80
    port: 8080
    protocol: TCP
    targetPort: 80
  selector:
    app: whoami
  type: NodePort
```

DAWAN - Reproduction interdite

Service – commandes de base

- Création d’un service

$ kubectl apply -f *PATH_SPEC_YAML*

$ kubectl expose …

$ kubectl create service …

- Lister l’ensemble des services

$ kubectl get service (svc)

- Principales infos concernant un service

$ Kubectl get svc/SERVICE_NAME

- Infos détaillées d’un service

$ kubect describe svc SERVICE_NAME

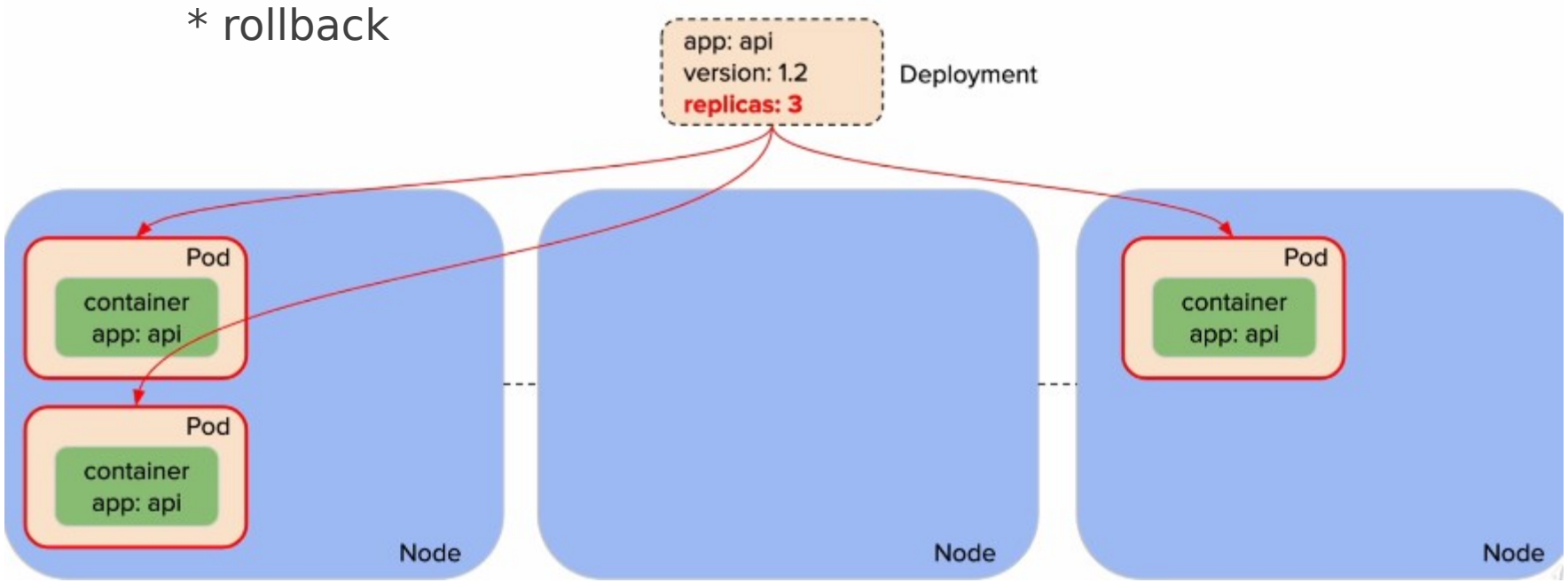
- Suppression d’un service

$ kubectl delete svc/SERVICE_NAME

DAWAN - Reproduction interdite

Deployment

- **Permet de gérer un ensemble de Pods identiques**
  - * Création / suppression de Pods
  - * Scaling
  - * mise à jour
  - * rollback



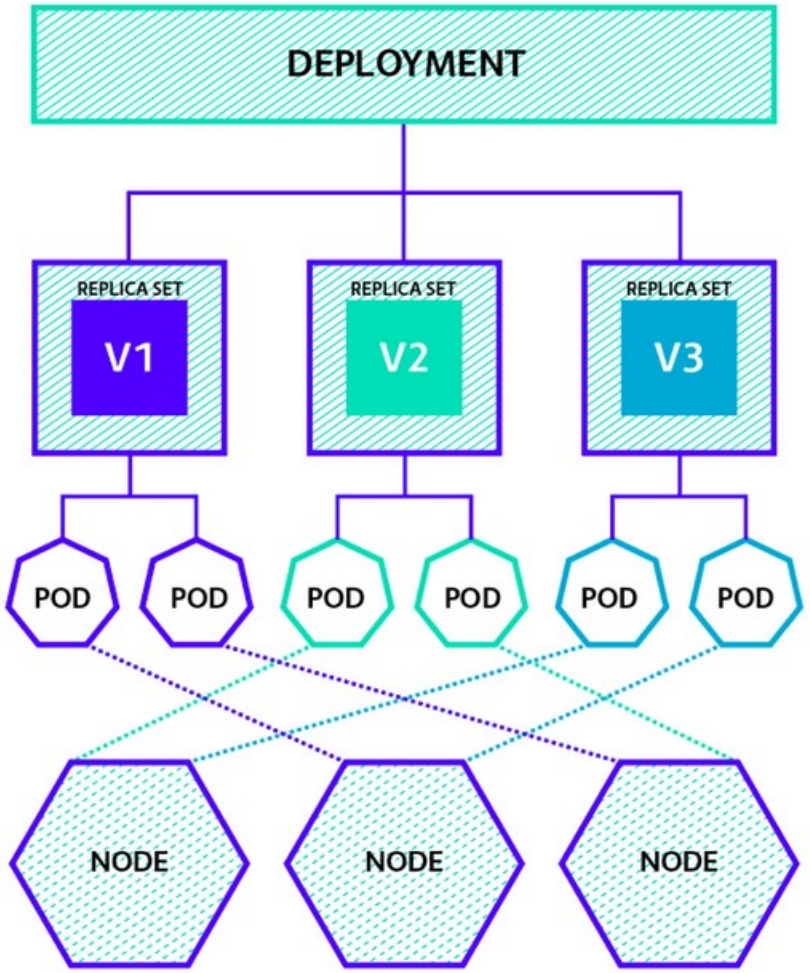
Deployment - utilisation

- Gestion du cycle de vie des Pods

* Création / Suppression
* Scaling
* Rollout / Rollback

- Différents niveaux d'abstraction

* Deployment
* ReplicaSet
* Pod



DAWAN - Reproduction interdite

Deployment - spécification

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: vote
spec:
  replicas: 3
  selector:
    matchLabels:
      app: vote
  template:
    metadata:
      labels:
        app: vote
    spec:
      containers:
      - name: vote
        image: instavote/vote
        ports:
        - containerPort: 80
```

Deployment

ReplicaSet

Pod

Deployment – approche impérative

```
pierre  ~  kubectl create deploy vote --image instavote/vote
deployment.apps/vote created
```

- Limitations :

* impossible de spécifier le nbre de réplicas (1 par défaut)
* impossible de spécifier plusieurs containers

- Pratique mais ...

* beaucoup moins flexible qu’une spécification yaml

Deployment – approche impérative

```
pierre  ~  kubectl create deploy vote --image instavote/vote --dry-run=client -o yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: vote
  name: vote
spec:
  replicas: 1
  selector:
    matchLabels:
      app: vote
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: vote
    spec:
      containers:
      - image: instavote/vote
        name: vote
        resources: {}
status: {}
```

DAWAN - Reproduction interdite

Deployment – Mise à jour

DEPLOYMENT STRATEGIES

When it comes to production, a ramped or blue/green deployment is usually a good fit, but proper testing of the new platform is necessary.

Blue/green and shadow strategies have more impact on the budget as it requires double resource capacity. If the application lacks in tests or if there is little confidence about the impact/stability of the software, then a canary, a/b testing or shadow release can be used.

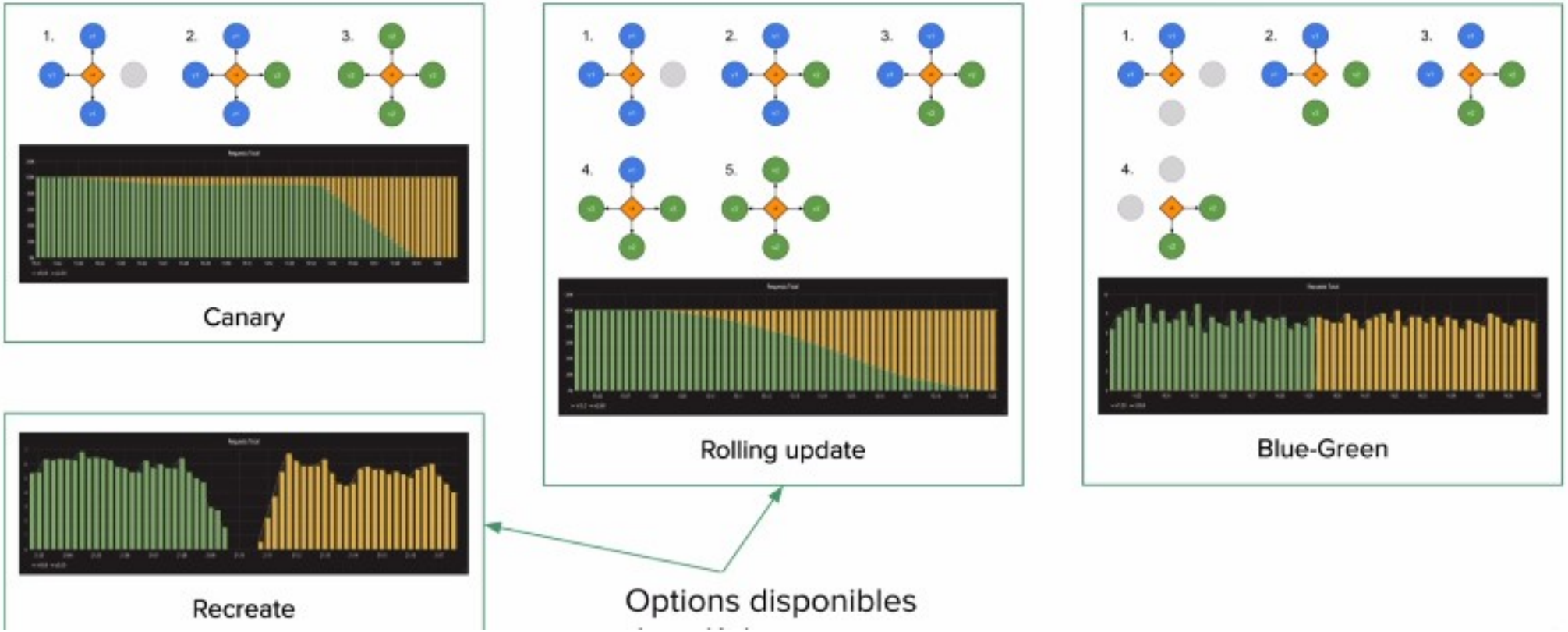
If your business requires testing of a new feature amongst a specific pool of users that can be filtered depending on some parameters like geolocation, language, operating system or browser features, then you may want to use the a/b testing technique.

Container Solutions

Strategy	ZERO DOWNTIME	REAL TRAFFIC TESTING	TARGETED USERS	CLOUD COST	ROLLBACK DURATION	NEGATIVE IMPACT ON USER	COMPLEXITY OF SETUP
RECREATE version A is terminated then version B is rolled out	✗	✗	✗	■□□	■■■	■■■	□□□
RAMPED version B is slowly rolled out and replacing version A	✓	✗	✗	■□□	■■■	■□□	■□□
BLUE/GREEN version B is released alongside version A, then the traffic is switched to version B	✓	✗	✗	■■■	□□□	■■□	■■□
CANARY version B is released to a subset of users, then proceed to a full rollout	✓	✓	✗	■□□	■□□	■□□	■■□
A/B TESTING version B is released to a subset of users under specific condition	✓	✓	✓	■□□	■□□	■□□	■■■
SHADOW version B receives real world traffic alongside version A and doesn't impact the response	✓	✓	✗	■■■	□□□	□□□	■■■

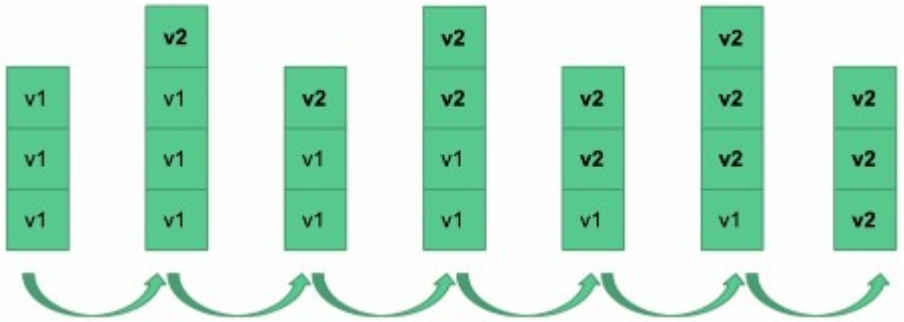
https://blog.container-solutions.com/deployment-strategies

Deployment – Mise à jour



Le « rolling update »

- Montée de version graduelle de l’ensemble de Pods
- Garantie de disponibilité
- Possiblité de rollback
- Historique des updates
- Paramètres pour contrôler l’update
    * maxUnavailable
    * maxSurge




```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: www
spec:
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
  ...
```

DAWAN - Reproduction interdite

Le « rolling update »

- Mise à jour à partir de la spécification

$ kubectl apply -f deploy.yml

- Mise à jour avec l'approche impérative

$ kubectl set image deploy/vote vote=instavote/vote:movies –record

** --record enregistre la commande dans une annotation de la ressource*

- Historique des mises à jour

$ kubectl rollout history deploy/vote

- Rollback : retour vers une révision antérieure

$ kubectl rollout undo deploy/vote

$ kubectl rollout undo deploy/vote –to-revision=X

- Forcer une recréation/mise à jour

$ kubectl rollout restart deploy/vote

DAWAN - Reproduction interdite

Le « scaling » horizontal

- Modification du nombre de Pods gérés par un Deployment

- 1 – Création d’un Deployment avec approche impérative

$ kubectl create deploy www –image=nginx:1.18

- 2 – Augmentation du nombre de réplicas

$ kubectl scale deploy/www --replicas=5

DAWAN - Reproduction interdite

HorizontalPodAutoscaler

- Modification du nombre de Pods en fonction :

*** de l’utilisation CPU**

*** de l’utilisation RAM**

*** de « custom metrics » (métier)**

Prérequis : Metrics server

$ kubectl top nodes

DAWAN - Reproduction interdite

Volume

- Découple les données du cycle de vie d’un container

- Permet aux containers d’un même Pod de partager des données

- intérieur de pods = non persistent

- extérieur de pods = persistent

- Nombreux types de volumes dispo :

* local
* nfs
* iscsi
* glusterfs
* cephfs
* vsphereVolume

Types de Volumes

Kubernetes supporte plusieurs types de Volumes:

- awsElasticBlockStore
- azureDisk
- azureFile
- cephfs
- cinder
- configMap
- csi
- downwardAPI
- emptyDir
- fc (fibre channel)
- flexVolume
- flocker
- gcePersistentDisk
- gitRepo (deprecated)
- glusterfs
- hostPath
- iscsi
- local
- nfs
- persistentVolumeClaim
- projected
- portworxVolume
- quobyte
- rbd
- scaleIO
- secret
- storageos
- vsphereVolume

DAWAN - Reproduction interdite

Volume – données persistées

- Différents types :

* data applicatives : fichiers plats, bdd
* logs
* fichiers de conf (on va privilégier les confi maps)
* fichiers partagés / sauvegardés

- Docker : 1 ou 2 types de volumes en général (bind, volumes)

- Kubernets : 4 types :

* emptyDir : pas de persistence mais partage entre pods/containers
* hostPath : répertoire partagé avec le host qui héberge le Pod
* persistent volume claim : specifique K8s
* externe : NFS, volume claim, glusterFS, vsphere...

DAWAN - Reproduction interdite

Volume

- **EmptyDir**

  * Lié au cycle de vie d’un Pod
  * Vide à la création
  * Supprimé à la destruction du Pod
  * Espace de travail et partage de données entre les containers du Pod

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: redis
spec:
  containers:
  - name: redis
    image: redis
    volumeMounts:
    - name: redis-storage
      mountPath: /data/redis
  volumes:
  - name: redis-storage
    emptyDir: {}
```

DAWAN - Reproduction interdite

Volume

- hostPath

* Ressource de la machine hôte dans un Pod (répertoire, fichier)
* Pas de mouvement possible entre nodes => prévoir un stockage distribué
* Ex :
  - donner accès à la socket Docker
  - présenter de la configuration

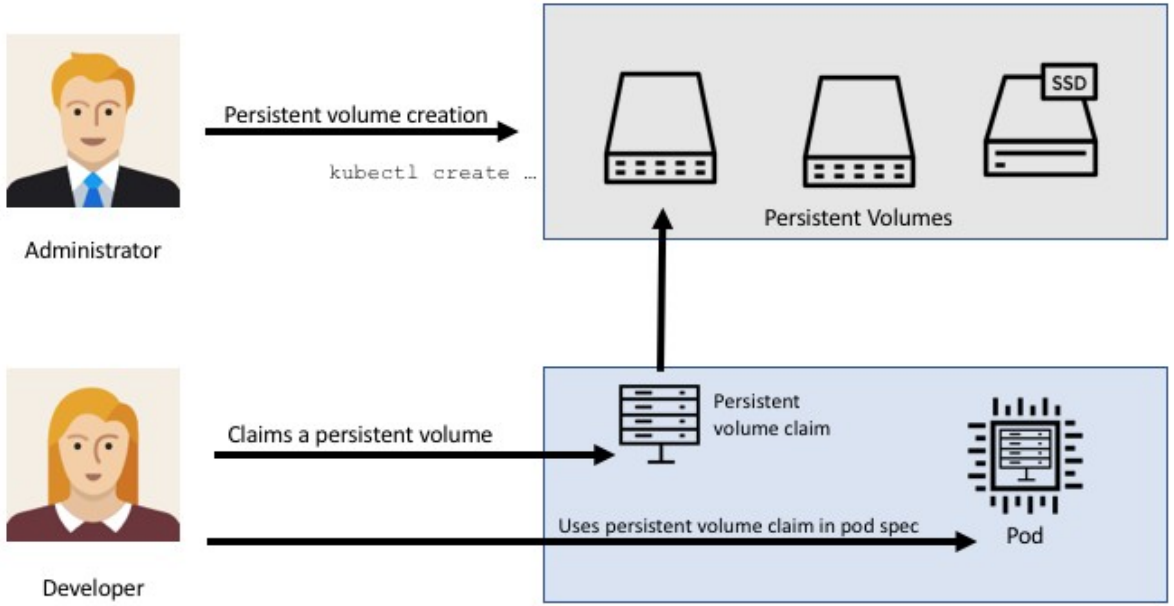
mongo-hostpath.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: mongo
spec:
  containers:
    - image: mongo:4.0
      name: mongo
      volumeMounts:
        - mountPath: /data/db
          name: data
  volumes:
    - name: data
      hostPath:
        path: /data-db
```

Volume

- **PersistentVolume & PersistentVolumeClaim**
  * provisioning
  * séparation entre provisioning et consommation

- **Imbrication :**
  * PV > PVC > Pods



Volume

- PersistentVolume

* ReadWriteOnce : monté sur un simple pod
* ReadOnlyMany : montés sur plusieurs pods en lecture
* ReadWriteMany : lecture écriture sur plusieurs pods

Path présent sur tous les nodes

persistent_volume.yaml

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: monpv
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/pvdata"
```

DAWAN - Reproduction interdite

Volume

- PersistentVolumeClaim

*Création d’une instance de consommation

persistent_volume_claim.yaml

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: monpvc
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

- Utilisation dans Pod

pod_pvc.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: monpodvol
spec:
  volumes:
    - name: monstorage
      persistentVolumeClaim:
        claimName: monpvc
  containers:
    - name: monnginx
      image: nginx
      ports:
        - containerPort: 80
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: monstorage
```

DAWAN - Reproduction interdite

Gestion des applications stateful

- Stateful : le serveur conserve des données de sessions

- Stateless : on ne conserve aucunes données de sessions

- Ressources en jeu :

* volume : partage de données entre les containers d’un Pod
* PersistentVolume (PV) : stockage provisionné par un administrateur
* PersistentVolumeClaim (PVC) : demande de stockage
* StorageClass : provisionnement dynamique du stockage
* StatefulSet

DAWAN - Reproduction interdite

StatefulSet

- Utilisé pour la gestion d’application Stateful

- Création ordonnée

- Création de service headless pour dns interne

- Chaque Pod à :

* un nom constant
* un identifiant réseau persistant
* son propre stockage

DAWAN - Reproduction interdite

Les « namespaces »

- Permet d’isoler un ensemble de ressources

* scope pour les Pods, Services, Deployments…

- Partage d’un cluster

* équipes, projets, clients
* clusters virtuels

- 3 namespaces par défaut :

```
pierre  ~  kubectl get namespace
NAME              STATUS   AGE
default           Active   3m20s
kube-node-lease   Active   3m22s
kube-public       Active   3m22s
kube-system       Active   3m22s
```

- Dans/Pas dans un namespace :

$ kubectl api-resources --namespaced=**true/false**

DAWAN - Reproduction interdite

Les « namespaces »

- Création 1

$ kubectl create namespace development

- Création 2

$ kubectl create -f namespace.yml

namespace.yml

```yml
apiVersion: v1
kind: Namespace
metadata:
  name: development
  labels:
    name: development
```

- Suppression

$ kubectl delete namespace/development

DAWAN - Reproduction interdite

Les « namespaces »

- Utilisation

$ kubectl create -f nginx-pod-dev.yml

- Liste (namespace default)

$ kubectl get po

- Liste namespace development

$ kubectl get po --namespace=development

- Liste all namespace

$ kubectl get po --all-namespace

nginx-pod-dev.yml

```yaml
---
apiVersion: v1
kind: pod
metadata:
  name: nginx
  namespace: development
spec:
  containers:
    - name: www
      image: nginx:1.19
```

DAWAN - Reproduction interdite

Les « namespaces »

- Possibilité d’ajouter des quotas

* Limiter les ressources pouvant être utilisées dans un namespace

quota.yml

```yaml
apiVersion: v1
kind: ResourceQuota
metadata:
  name: quota
spec:
  hard:
    requests.cpu: "1"
    requests.memory: 1Gi
    limits.cpu: "2"
    limits.memory: 2Gi
```

$ kubectl apply -f quota.yml --namespace=test

DAWAN - Reproduction interdite

ConfigMap

- Découplage d’une application et de sa configuration

* pas de configuration dans le code applicatif !

- Assure la portabilité, centralise la configuration, permet le partage entre Pods

- Simplifie la gestion par rapport à l’utilisation de variables d’environnement

- Créée à partir d’un fichier, d’un répertoire ou de valeurs littérales

- Contient une ou plusieurs paires de clé / valeur

DAWAN - Reproduction interdite

Secrets

- Protection des données sensibles

* clé privée, mot de passe...

- Evite de définir ces infos dans les images ou spécifications

- Meilleur contrôle de leur utilisation

- Créés par un utilisateur ou par le système (API)

- Utilisé dans un Pod

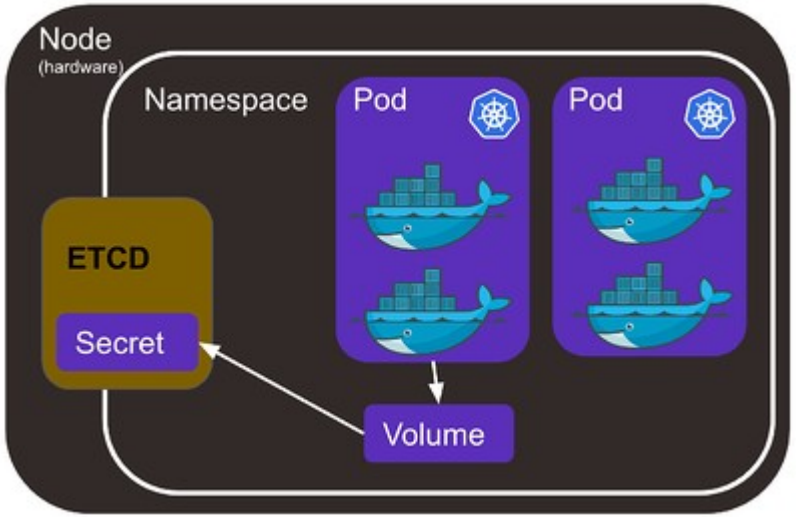
- Stocké dans etcd

DAWAN - Reproduction interdite

Secrets

- Différents types :

* generic : secret créé à partir d’un fichier, répertoire ou valeur littérale
* docker-registry : secret utilisé pour authentification à une registry
* TLS : secret utilisé pour la gestion des clés (PKI)



DaemonSet

- Haute disponibilité

- Objectif :

* Constituer des copies de Pods sur tous les nodes du cluster
* Si un nouveau node est ajouté au cluser, le Pod est déployé sur celui-ci

- Cas d’usages :

* collecte de logs
* monitoring
* stockage distribué

DAWAN - Reproduction interdite

Job / Cronjob

- Utilisé pour lancer des « batch-jobs

- Différentes possibilités :

* lancer un seul Pod
* lancer plusieurs Pods en parallèle ou en séquentiel
* géré par .spec.completions et .spec.parallelism

DAWAN - Reproduction interdite

Job

job-hello.yml

```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: hello
spec:
  completions: 3
  parallelism: 1
  template:
    spec:
      restartPolicy: OnFailure
      containers:
        - name: hello
          image: alpine
          args:
            - /bin/sh
            - -c
            - sleep 10; echo Hello from Kube; date
```

```
pierre  master  ~  Formations  2021  Kubernetes  kubectl apply -f job-hello.yml
job.batch/hello created
pierre  master  ~  Formations  2021  Kubernetes  kubectl get job -w
NAME    COMPLETIONS   DURATION   AGE
hello   0/3           10s        10s
hello   1/3           16s        16s
hello   2/3           31s        31s
hello   3/3           44s        44s
```

DAWAN - Reproduction interdite

Cronjob

- Lancement de tâches automatiques

- Similaire à la crontab Linux

- Cas d’usage :

* backup
* rotation logs
* envoi emails

DAWAN - Reproduction interdite

Cronjob

cronjob-helloyml

```yaml
apiVersion: batch/v1
kind: CronJob
metadata:
  name: hello
spec:
  schedule: "*/2 * * * *"
  jobTemplate:
    spec:
      template:
        spec:
          restartPolicy: OnFailure
          containers:
            - name: hello
              image: alpine
              args:
                - /bin/sh
                - -c
                - sleep 10; echo Hello from Kube as cronjob; date
```

Valider l’état de santé des applications

- Les « probes »

- Certaines applications ont besoin d'être restartées

- Prévenir des images mal construites non fonctionnelles

- Eviter les coupures lors de déploiements (zero downtime)

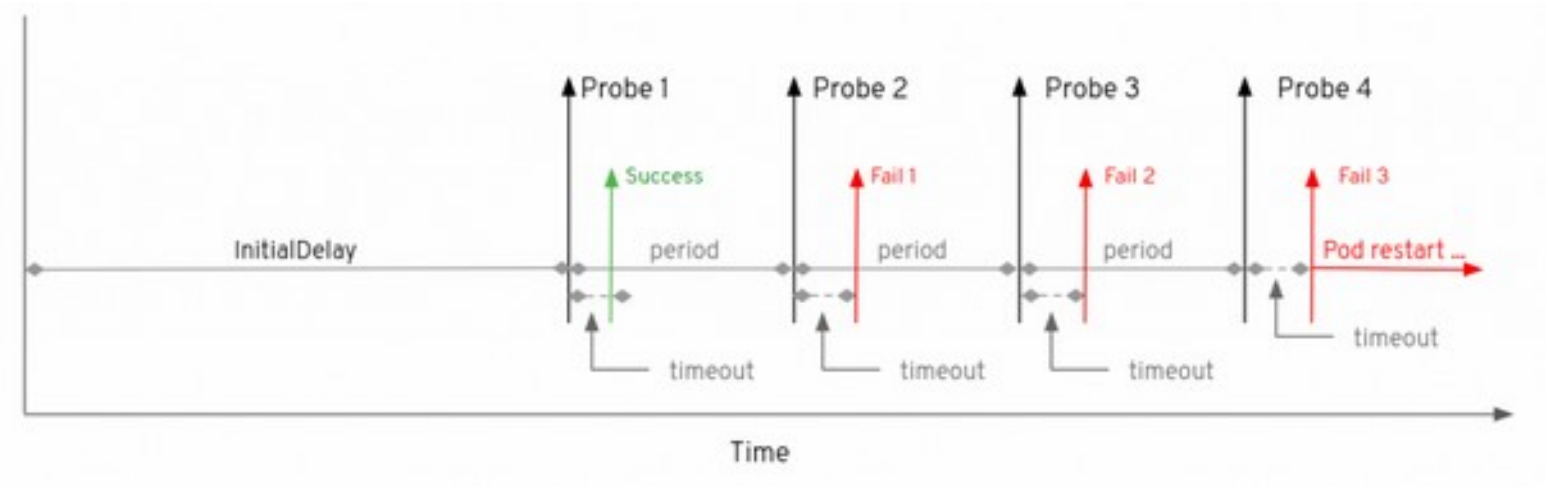
- Différentes concepts :

* Readiness probes : Affecte la mise en réseau des pods
* Liveness probes : Vérifie la disponibilité d’un service
* les deux sont complémentaires

DAWAN - Reproduction interdite

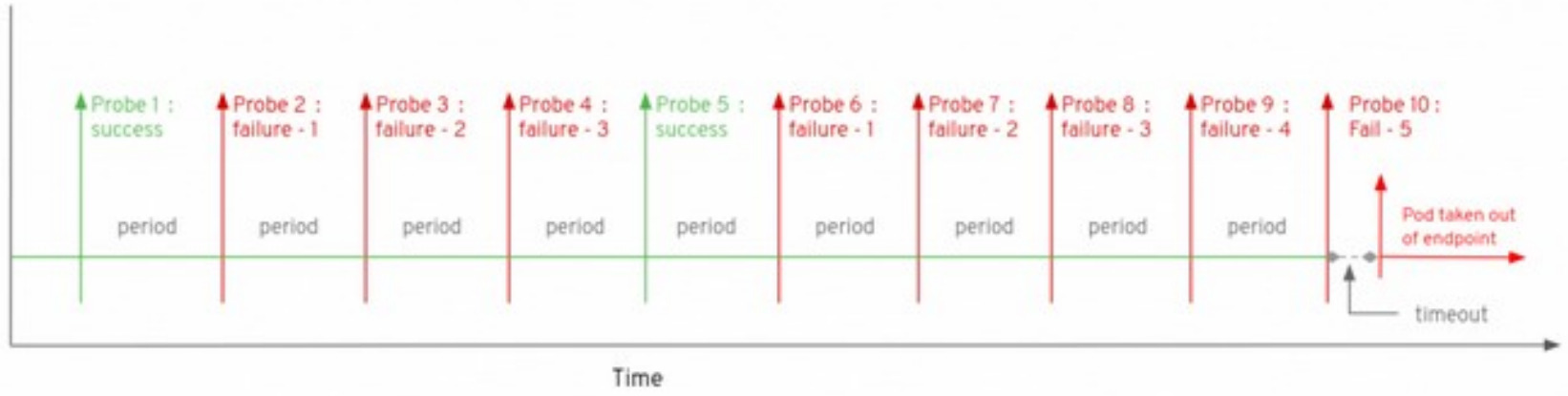
La probe liveness

- Si un Pod **ne répond pas** correctement à la requête, il est considéré comme corrompu et est **instantanément redémarré**



La probe readiness

- Si le pod **répond mal**, il est alors **détaché temporairement** du service (il est sorti du loadbalancer) jusqu'à ce que celui-ci soit de nouveau opérationnel. Cela permet d'**éviter que le pod soit redémarré** et qu'il perde ainsi les traitements en cours d'exécution.-



DAWAN - Reproduction interdite

Valider l’état de santé des applications

- Méthodes :

* command > **exec**
* http > **httpGet**
* tcp > **tcpSocket**

Pod Health checks

	Liveliness	Readiness
On failure	Kill container	Stop sending traffic to pod
Check types	Http , exec , tcpSocket	Http , exec , tcpSocket
Declaration example (Pod.yaml)	livenessProbe: failureThreshold: 3 httpGet: path: /healthz port: 8080	readinessProbe: httpGet: path: /status port: 8080

- Variables :

* **initialDelaySeconds :** à partir de quand commence le check ?
* **periodSeconds :** à quelle fréquence ?
* **timeoutSeconds :** à partir de combien on timeout ?
* **successThreshold :** au bout de combien de positifs on réactive ?
* **failureThreshold :** au bout de combien de négatifs on sort ?

DAWAN - Reproduction interdite

Ingress

- Ensemble de règles pour la connection aux service du cluster depuis ext

- Déploiement d’un « Ingress Controller » nécessaire

* container docker (nginx, traefik, HAProxy
* load balancer managé par K8s

- Cas d’usage :

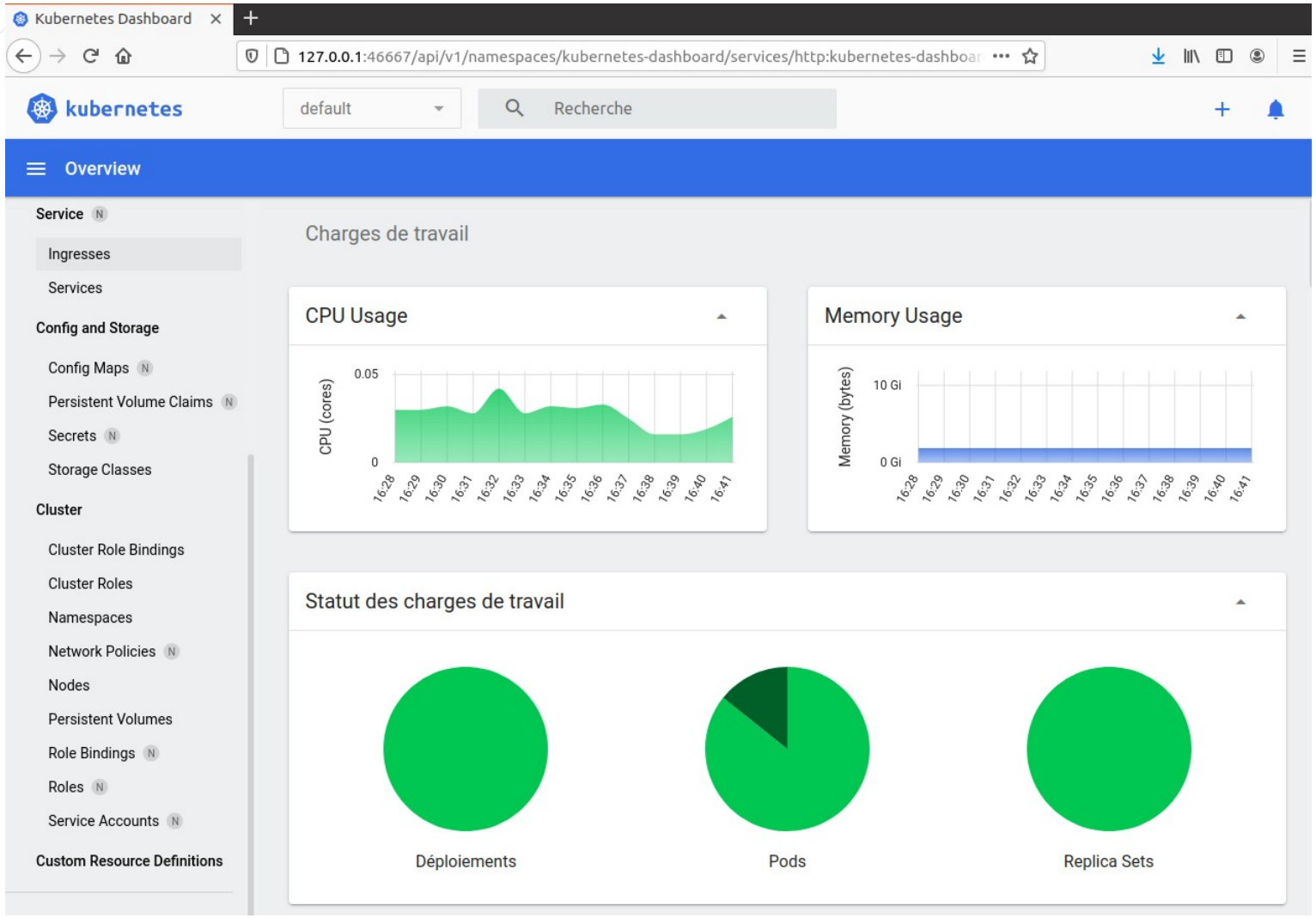
* routage HTTP
* load balancing

- Addons Minikube !

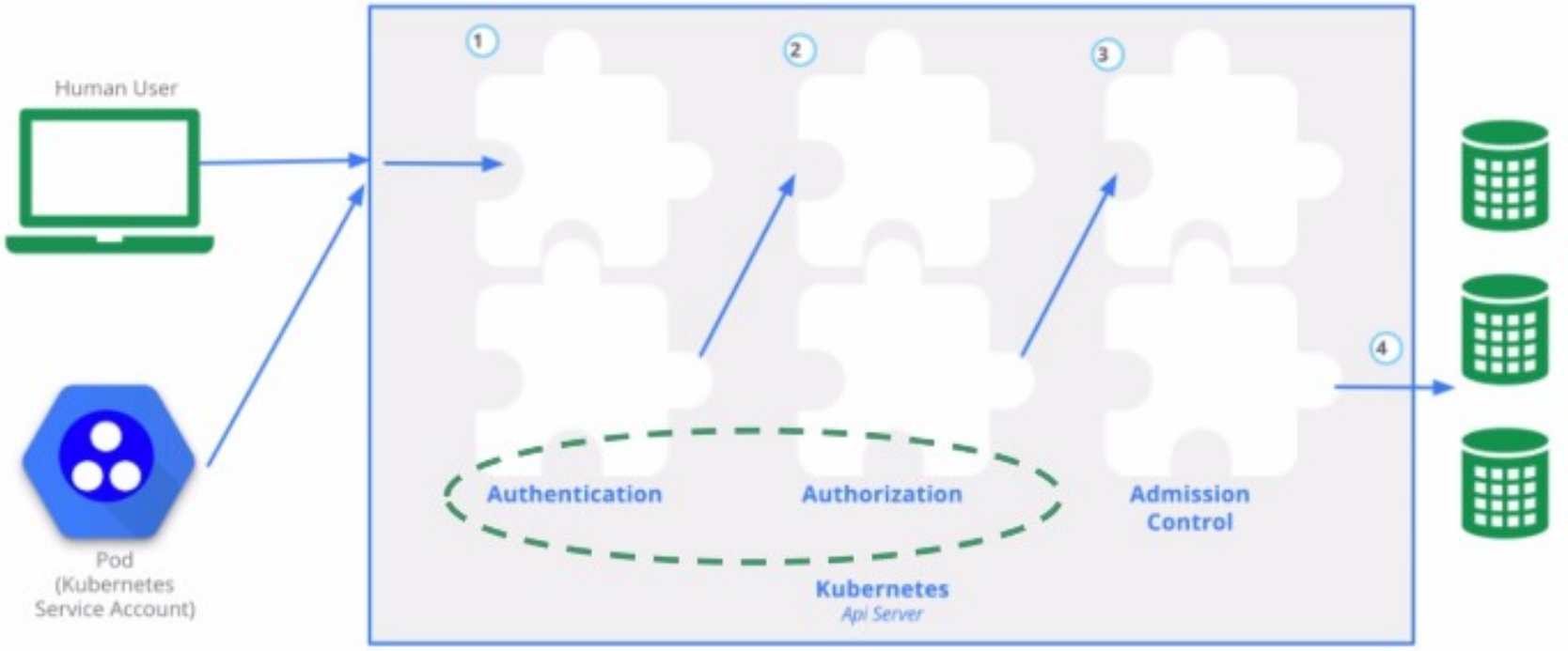
$ minikube addons enable ingress

DAWAN - Reproduction interdite

Kubernets Dashboard



Utilisateurs et droits d’accès - RBAC



https://kubernetes.io/docs/admin/accessing-the-api/

Authentification

- Utilisateur :

* Certificat client
* Token
* HTTP basic auth

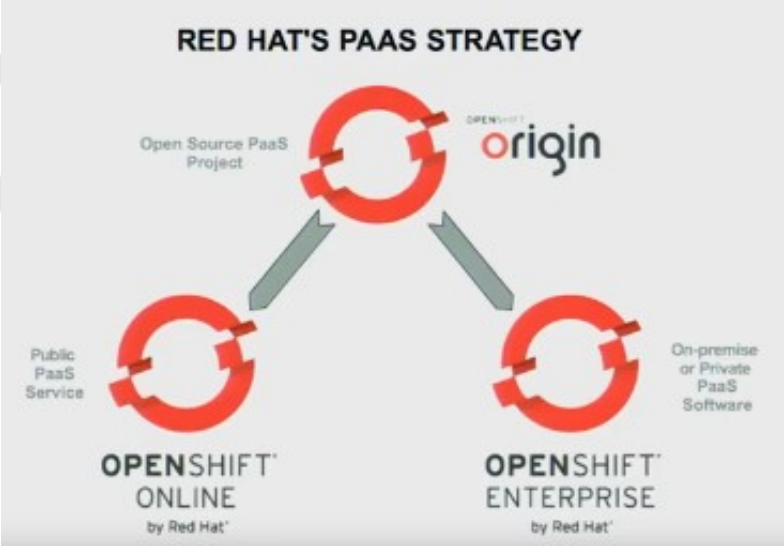
- Service Account :

* donne des droits aux containers tournant dans un Pod
* JWT token disponible via un secret

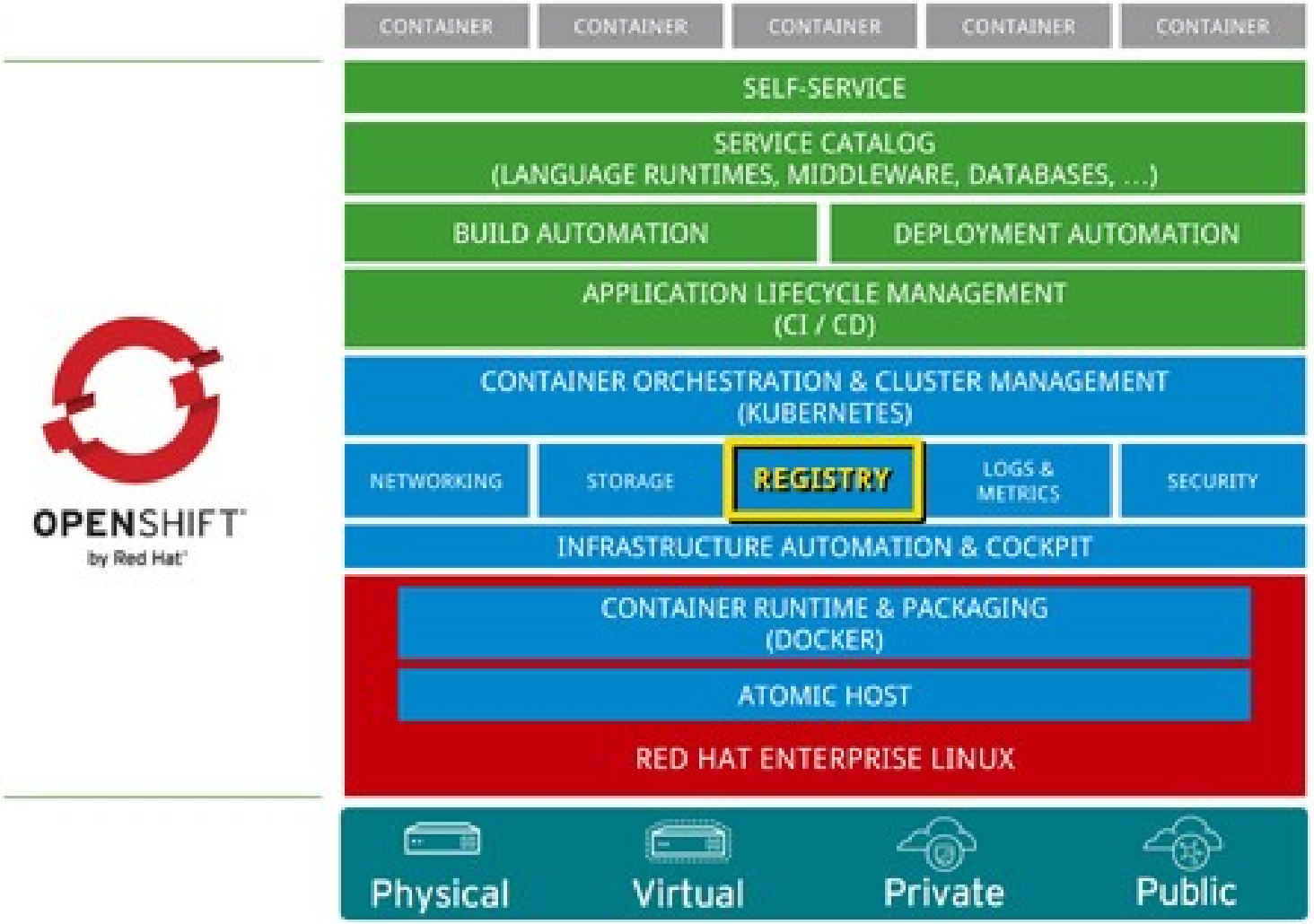
DAWAN - Reproduction interdite

- **OpenShift:**
  * RedHat
  * Déploiement de projets, application dans des conteneurs
  * Docker + Kubernetes

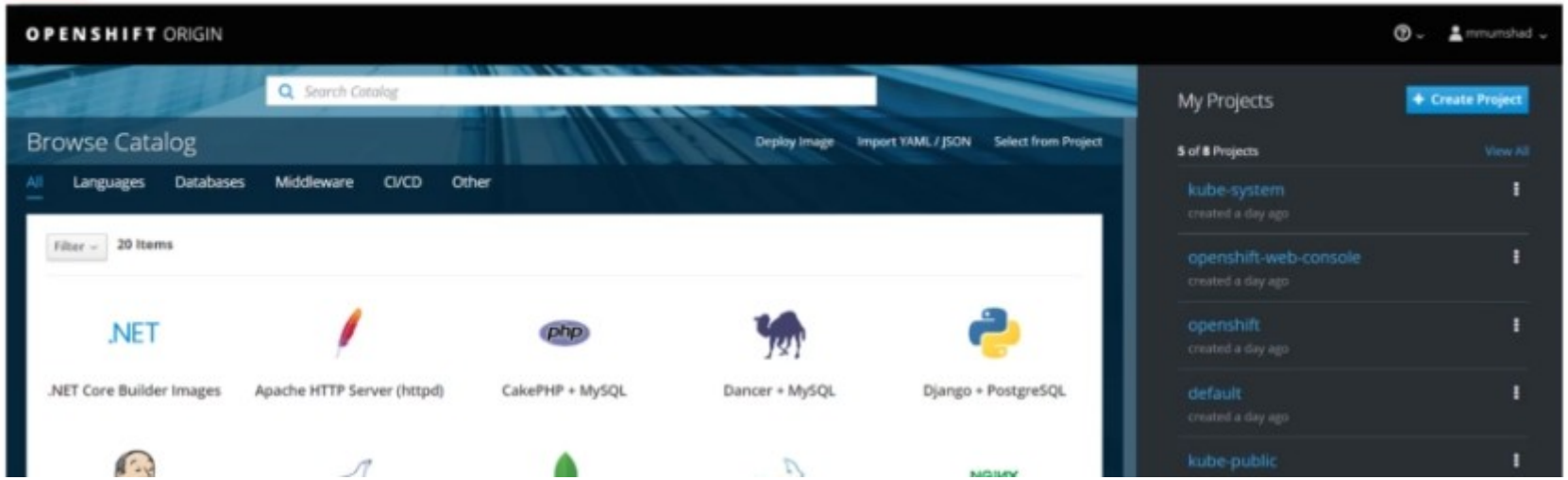
- **Plusieurs licences**
  * ODK
  * Red Hat OpenShift Container Platform
  * Cloud computing



	OpenShift	Kubernetes
Système d'exploitation	Linux, Fedora, CentOS	Tout système d'exploitation
Sécurité	Routine de sécurité élevée	La sécurité dépend de l'utilisateur
Cycle de mises à jour	Environ 3 par an	4 par an
CI/CD	Font partie intégrante du logiciel	Possibles avec Jenkins
Convivialité	Pensé pour une utilisation simple	Utilisation moins pratique
Interface utilisateur	Interface simple	Un tableau de bord peut être installé ultérieurement
Cible	OpenShift vise les entreprises, mais peut être adapté	Kubernetes peut être utilisé pour toute taille de projet
Modèles	Moins conviviaux	Flexibilité élevée grâce à Helm de Kubernetes
Mise en réseau	Open vSwitch offre des possibilités de mise en réseau	Des plugins de fournisseurs tiers permettent la mise en réseau


OPENSHIFT
by Red Hat
CONTAINER
CONTAINER
CONTAINER
CONTAINER
CONTAINER
SELF-SERVICE
SERVICE CATALOG
(LANGUAGE RUNTIMES, MIDDLEWARE, DATABASES, ...)
BUILD AUTOMATION
DEPLOYMENT AUTOMATION
APPLICATION LIFECYCLE MANAGEMENT
(CI / CD)
CONTAINER ORCHESTRATION & CLUSTER MANAGEMENT
(KUBERNETES)
NETWORKING
STORAGE
REGISTRY
LOGS & METRICS
SECURITY
INFRASTRUCTURE AUTOMATION & COCKPIT
CONTAINER RUNTIME & PACKAGING
(DOCKER)
ATOMIC HOST
RED HAT ENTERPRISE LINUX
Physical
Virtual
Private
Public

Web Console - Homepage



CLI

```
> oc
OpenShift Client

This client helps you develop, build, deploy, and run your applications on any
OpenShift or Kubernetes compat
platform. It also includes th
under the 'adm' subcommand.

To create a new application,

  oc login https://mycluster.
  oc new-app centos/ruby-22-c
  oc logs -f bc/ruby-ex
```

CLI - Login

```
> oc login
OpenShift server [https://localhost:8443]: https://openshift.example.com

Username: developer
Authentication required for https://openshift.example.com (openshift)
Password: ******
Login successful.
```

```
> oc login -u developer -p developer
Login successful.
```

```
> oc logout
User, developer, logged out of https://openshift.example.com
```

REST API

```
> curl https://localhost:8443/oapi/v1/users \
    -H “Authorization: Bearer 27pZjHWn9GU5B5De4H3ICP6ZBd_7txQJMvz79dH0tsY”
```


```
"kind": "UserList",
 "apiVersion": "v1",
 "metadata": {
   "selfLink": "/oapi/v1/users",
 },
 "items": [
   {
     "metadata": {
       "name": "developer",
       "selfLink": "/oapi/v1/users/developer",
       "uid": "271e2b49-47f0-11e8-afb8-4a1a95a6dbc1",
       "resourceVersion": "1289",
       "creationTimestamp": "2018-04-24T18:49:00Z"
     },
     "identities": [
       "anypassword:developer"
     ],
     "groups": null
   },
   {
     "metadata": {
       "name": "mmumshad",
       "selfLink": "/oapi/v1/users/mmumshad",
```


```
> oc whoami -t
27pZjHWn9GU5B5De4H3ICP6ZBd_7txQJMvz79dH0tsY
```

Users

Regular

developer

System

system:admin

system:master

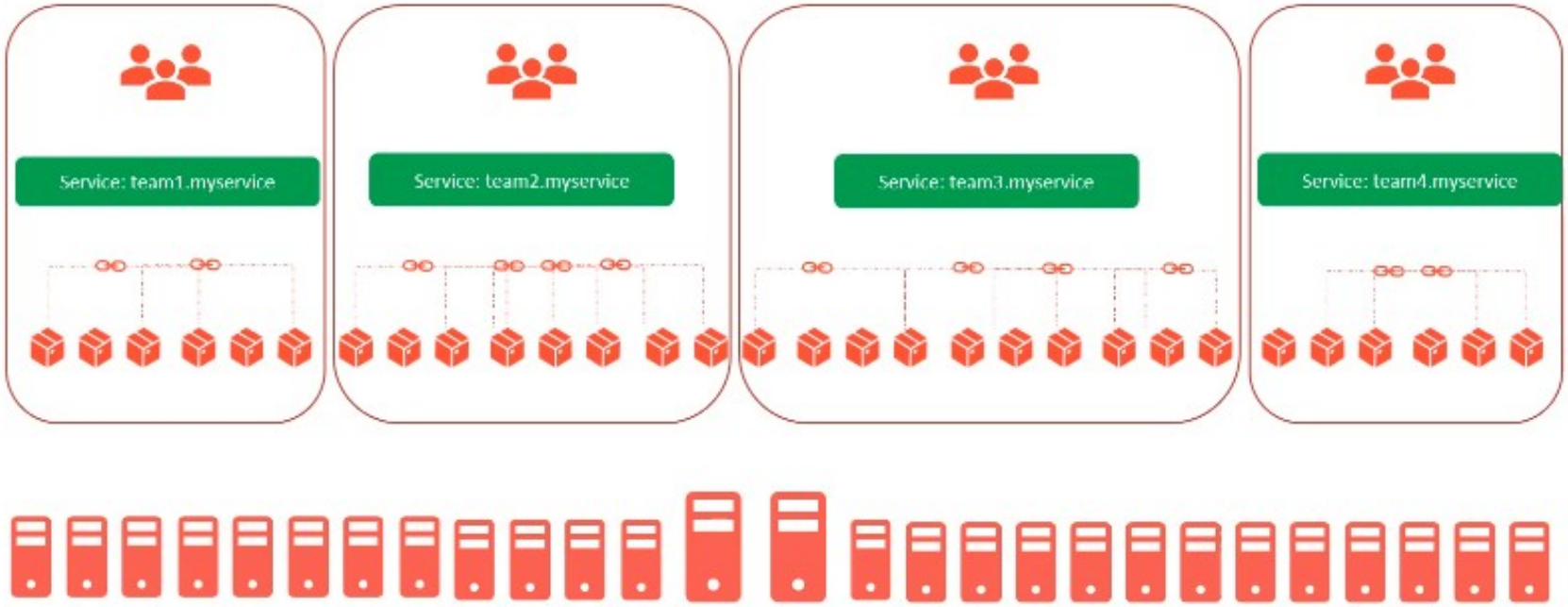
Service

system:serviceaccount:proj1:db_user

C:\WINDOWS\system32\cmd.exe - cmd

```
c:\minishift-1.16.1-windows-amd64>oc get users
NAME        UID                                    FULL NAME   IDENTITIES
developer   758ed92d-4c3b-11e8-8098-e2e0de05d311               anypassword:developer

c:\minishift-1.16.1-windows-amd64>oc get users
NAME         UID                                    FULL NAME   IDENTITIES
developer    758ed92d-4c3b-11e8-8098-e2e0de05d311               anypassword:developer
developer2   ba585475-4c43-11e8-8098-e2e0de05d311               anypassword:developer2

c:\minishift-1.16.1-windows-amd64>oc get users
NAME            UID                                    FULL NAME   IDENTITIES
administrator   4343778b-4c44-11e8-8098-e2e0de05d311               anypassword:administrator
developer       758ed92d-4c3b-11e8-8098-e2e0de05d311               anypassword:developer
developer2      ba585475-4c43-11e8-8098-e2e0de05d311               anypassword:developer2

c:\minishift-1.16.1-windows-amd64>oc adm policy add-cluster-role-to-user cluster-admin administrator
cluster role "cluster-admin" added: "administrator"

c:\minishift-1.16.1-windows-amd64>
```

Projects


Service: team1.myservice
Service: team2.myservice
Service: team3.myservice
Service: team4.myservice

Users

Regular

developer

System

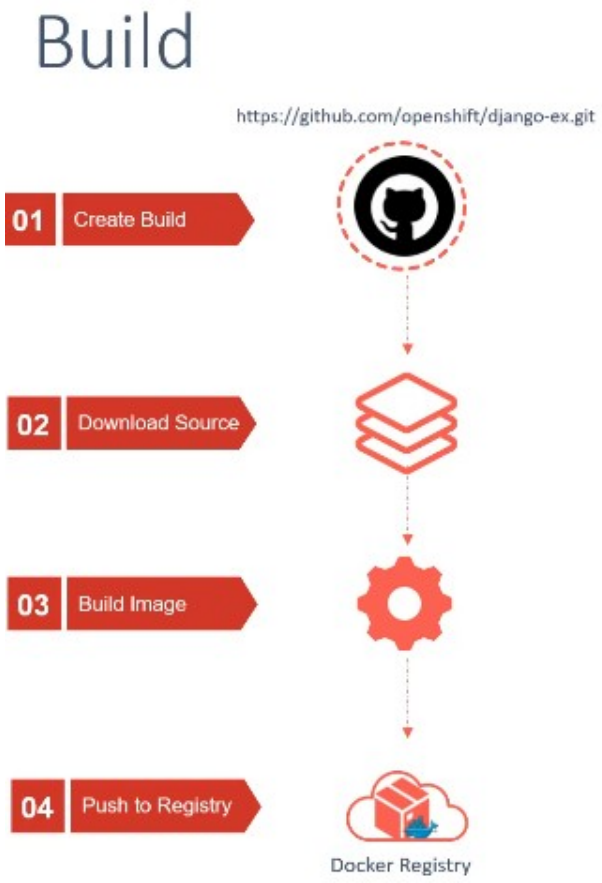
system:admin

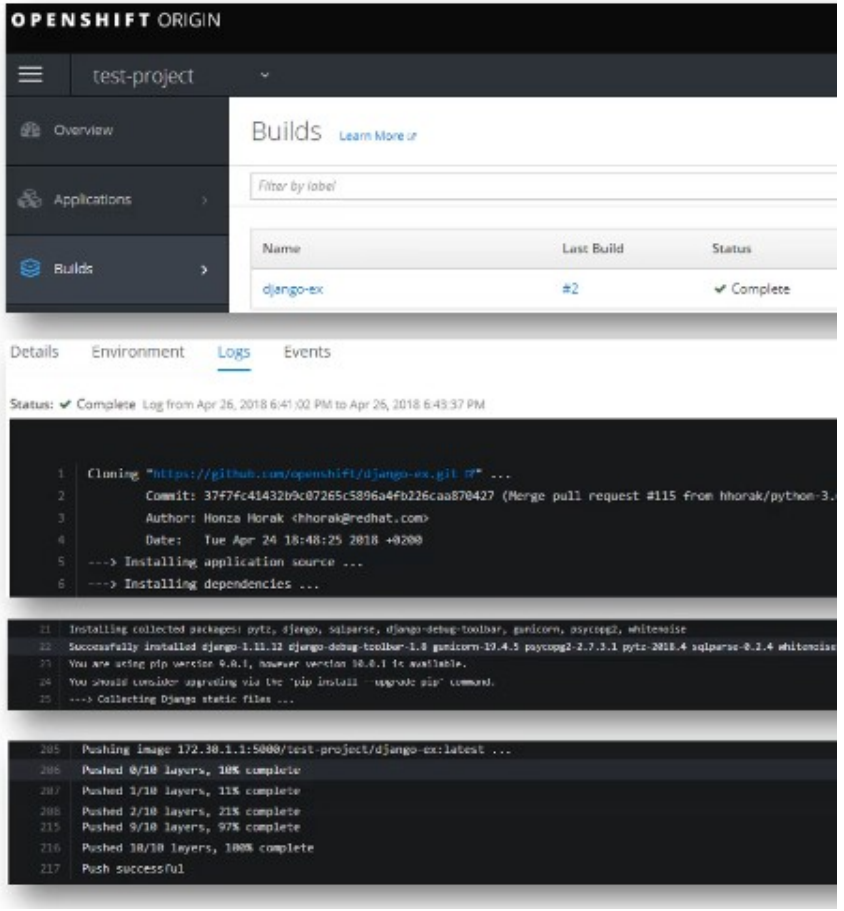
system:master

Service

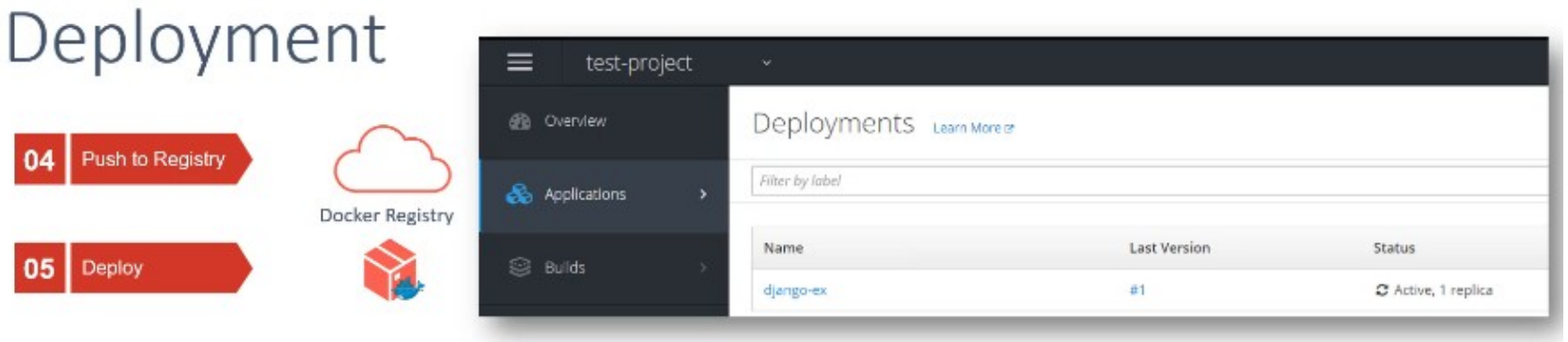
system:serviceaccount:proj1:db_user

Build OpenShift



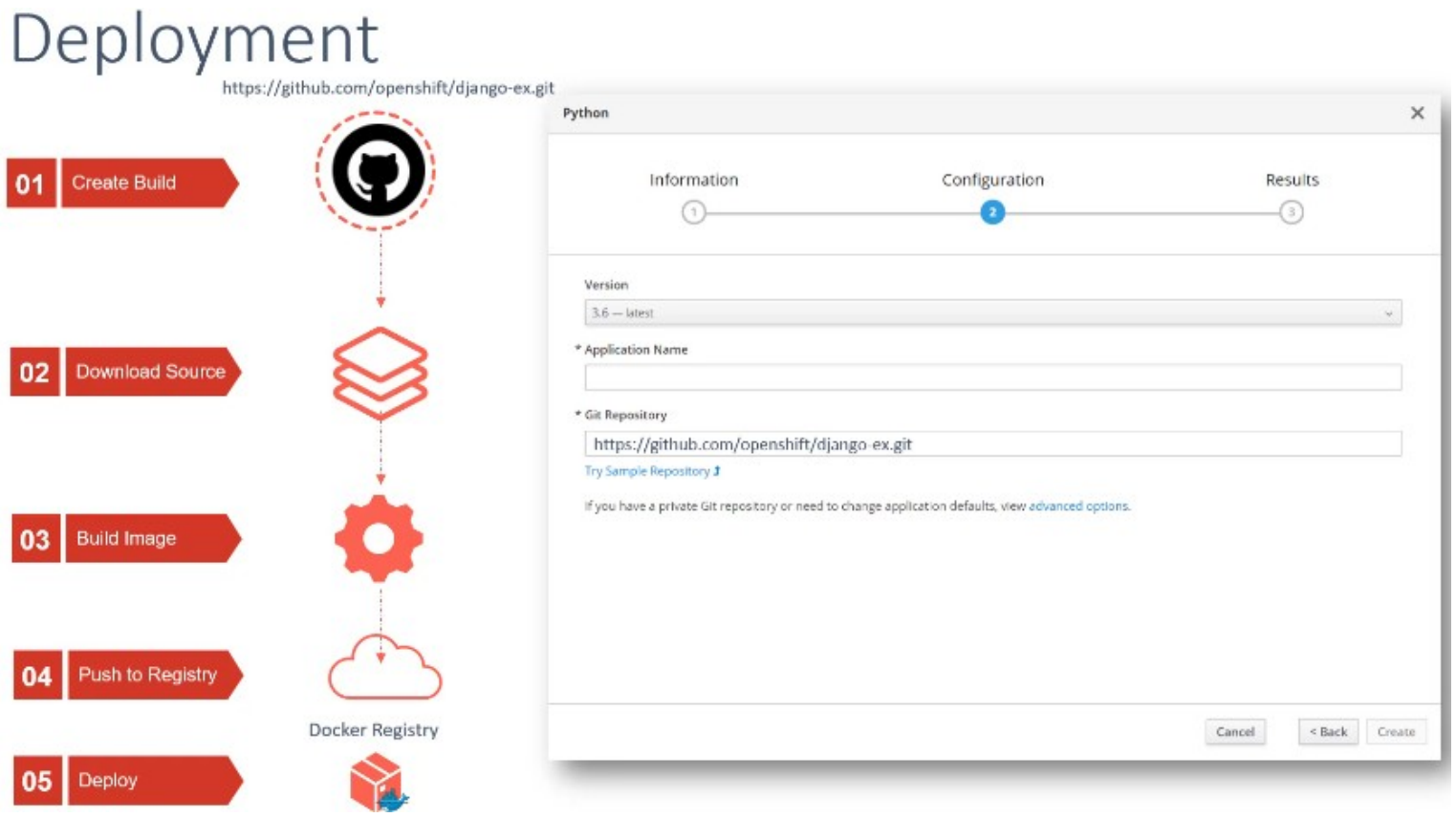


Deployment OpenShift




```
apiVersion: apps/v1
kind: Deployment
```

≠

```
apiVersion: apps.openshift.io/v1
kind: DeploymentConfig
```

En résumé : OpenShift