



# Kubernetes - mise en œuvre

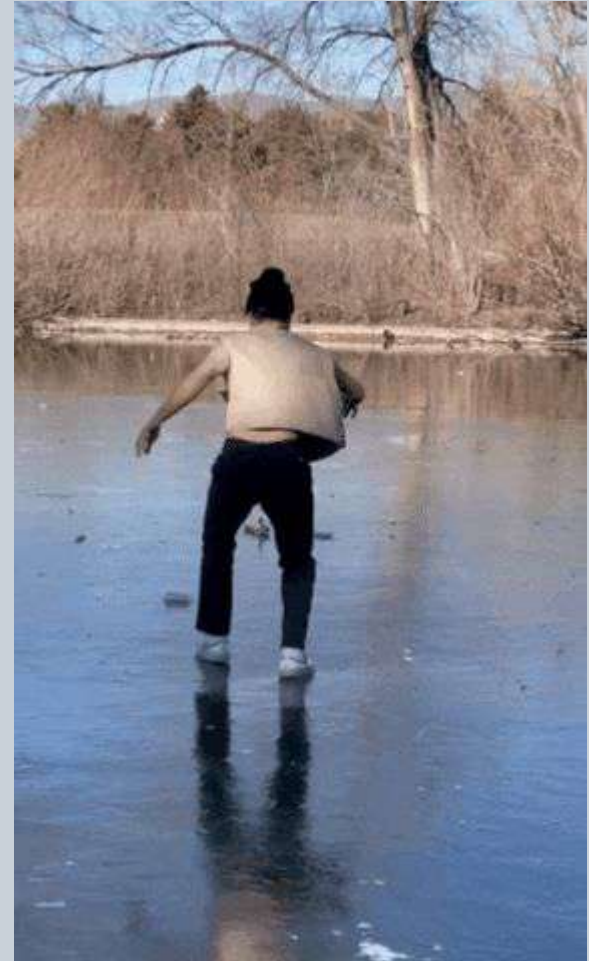
Mawaki PAKOUPETE

# \$ whoami

- Mon nom est ***Mawaki PAKOUPETE***
- Ingénieur Cloud & DevOps
  - ◆ Débutant dans l'administration système : Unix/Linux, Virtualisation...
  - ◆ Ingénieur systèmes & DevOps (Docker, Kubernetes, Cloud AWS, GCP, Azure...)
  - ◆ Ingénieur cloud travaillant pour une entreprise cloud qui utilise fortement Kubernetes
- Certifications : Elasticsearch, CKA, AWS, RHCE, RHCSA, VMware LPIC-1, ITIL
- J'utilise les technologies Docker et Kubernetes tous les jours
- Passionné par la formation depuis 7 ans

# Ice Breaker

- Prénom & Nom
- Entreprise actuelle & Rôle
- Attentes de la formation



\$ need --help

## Comment demander de l'aide ?

- Cours qui se veut interactif
- Pour toute question, levez la main et posez là
- N'importe quel moment
- Vous pouvez utiliser le Chat également

# Contenu du Cours

MATIN

APRES MIDI

## Jour 1

### Rappel Docker/Conteneurs

Qu'est-ce qu'un conteneur (Docker vs VM)  
Pourquoi les conteneurs ?  
commandes de base de Docker  
Dockerfile

### Introduction à Kubernetes

Qu'est ce que Kubernetes ?  
Différents orchestrateurs de conteneur ?  
Historique de Kubernetes  
Pourquoi Kubernetes ?

### Architecture de Kubernetes

Différents composants de Kubernetes ?  
Comment sont-ils reliés les uns aux autres ?  
Ressources & controllers  
Déployer un cluster Kubernetes

### Mise en oeuvre de Kubernetes

Vue d'ensemble des ressources K8s  
Contexte d'utilisation  
K8s API, API version,  
Fichier YAML

## Jour 2

### Ressources Workload K8s

Pods  
ReplicaSet, Deployment  
Stratégies de Déploiement  
StatefulSet, DaemonSet, Jobs

### Ressources Services & Networking K8s

Service (ClusterIP, NodePort, LB)  
DNS & Réseau en environnement K8s  
Ingress, Ingress Controllers

### Ressources de configuration & Stockage K8s

Persistent Volumes  
Persistent Volumes Claim  
Storage Classes  
ConfigMaps  
Secrets

### Ressources Cluster K8s Namespace

Role & Role Binding  
Cluster Role & Cluster Role Binding  
Service Account

## Jour 3

### Scheduling sur K8s

Kube-scheduler  
Taints, labels, annotations  
Affinity & AntiAffinity  
Liveness, Readiness, Startup probes

### Administration d'un Cluster K8s

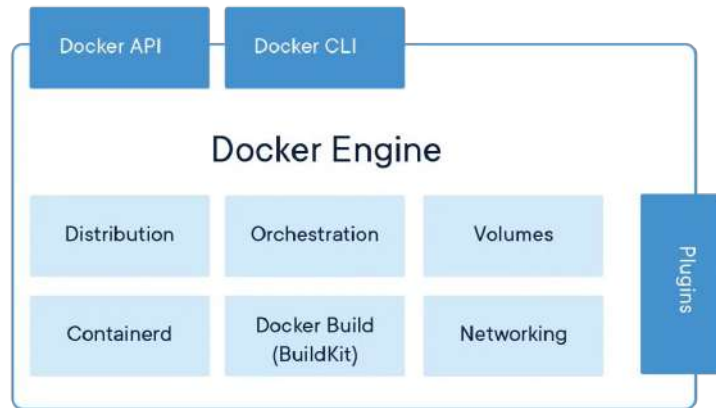
Kubeadm: Installation K8s  
Maintenance des noeuds

# Rappel Docker & Conteneurs

- Qu'est ce que Docker?
- Qu'est-ce qu'un conteneur (Docker vs VM) ?
- Historique de Docker
- Pourquoi les conteneurs ?

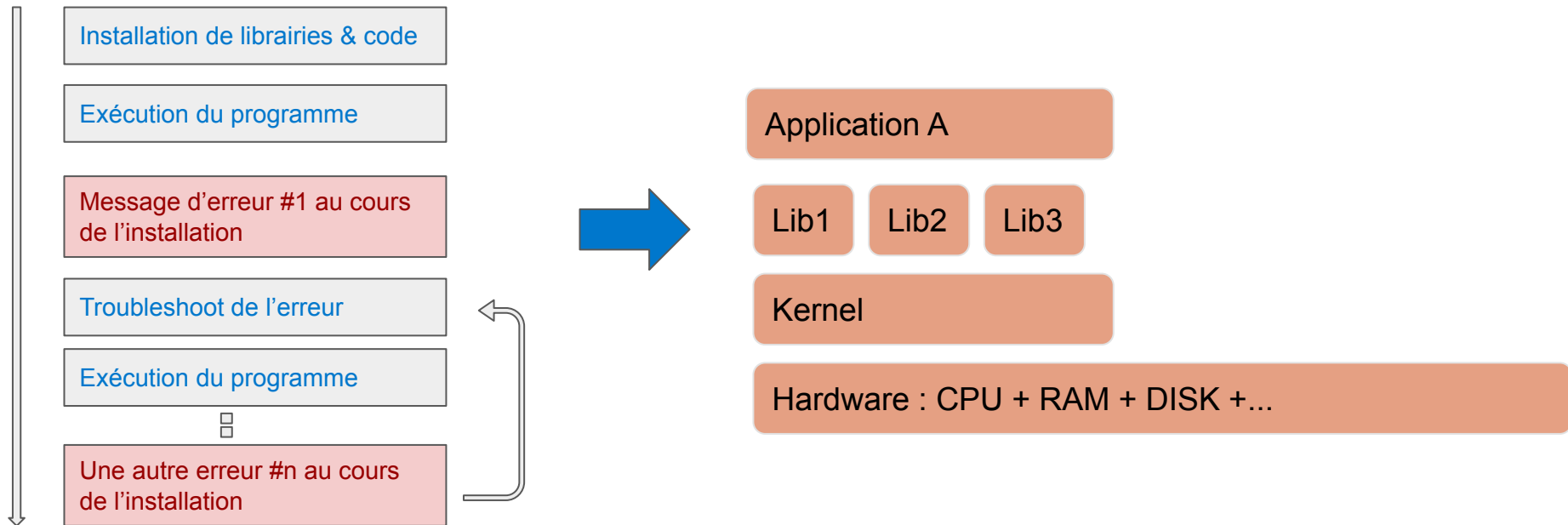
# Qu'est ce que Docker?

- Docker est une plateforme ouverte (une sorte d'écosystème) pour le développement, l'expédition et l'exécution d'applications.
- Docker permet de packager (images) et d'exécuter une application dans un environnement faiblement isolé appelé conteneur.



# Pourquoi les conteneurs ?

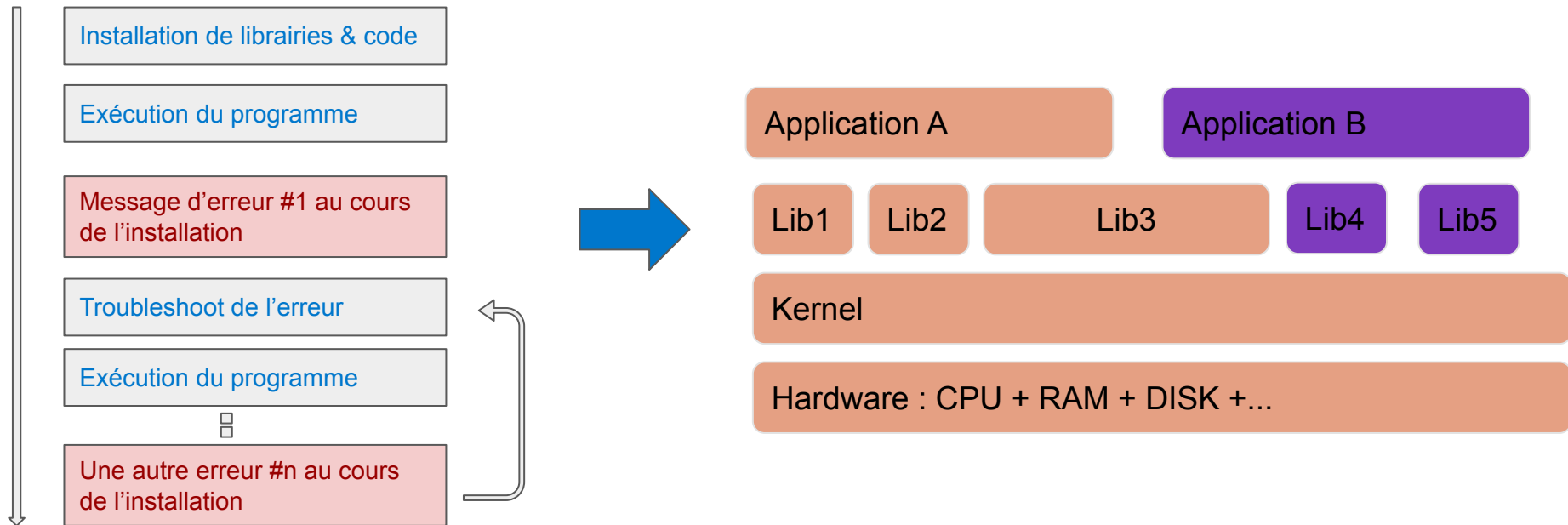
## Scénario 1 : Installation de l'application A





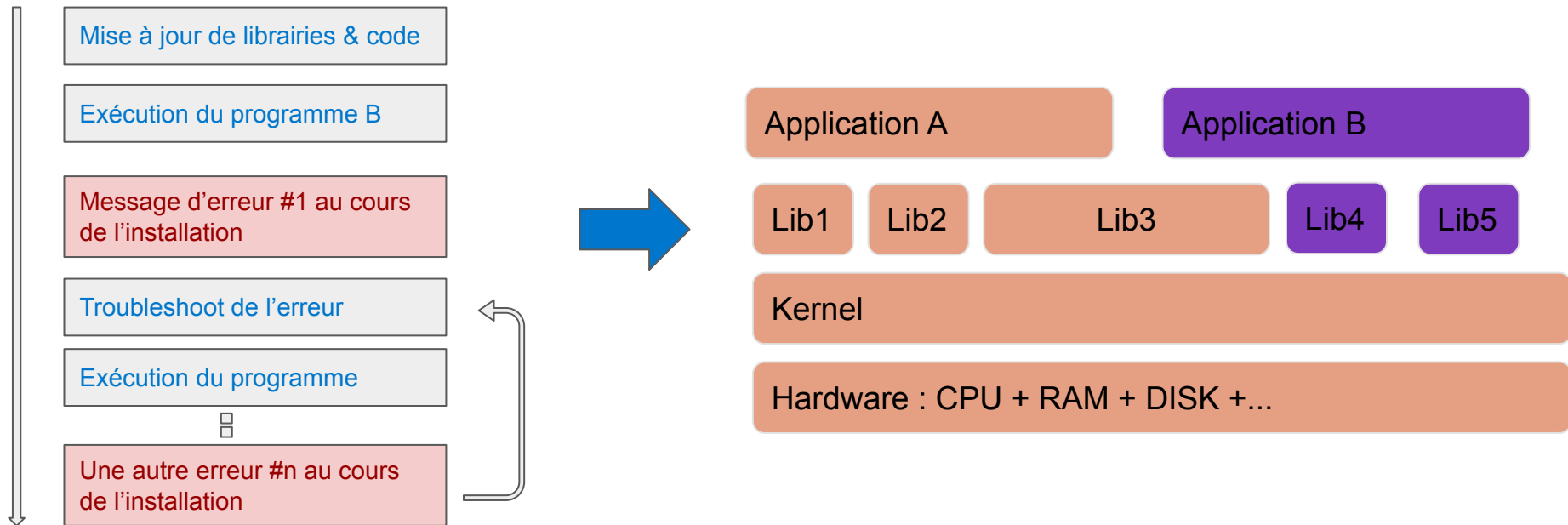
# Pourquoi les conteneurs ?

## Scénario 2 : Installation d'une seconde application B



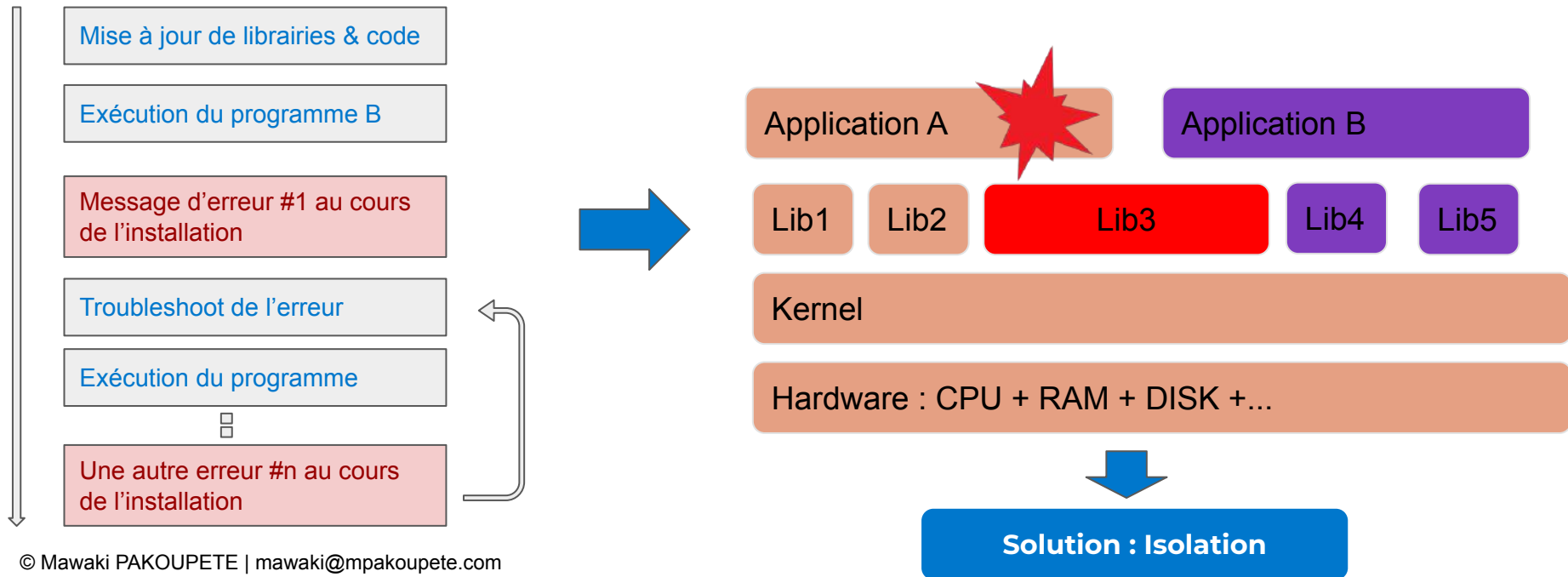
# Pourquoi les conteneurs ?

## Scénario 3 : Obligation de mise à jour de application B



# Pourquoi les conteneurs ?

Nouveau Problème : Application A Down : Lib3 - version incompatible

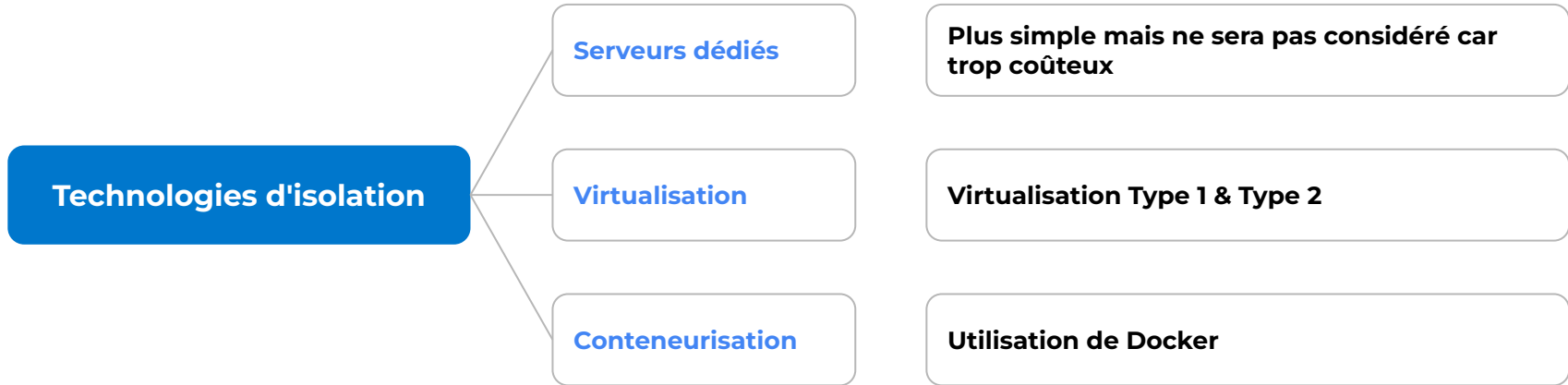


# Pourquoi les conteneurs ?

## **Autres inconvénients de l'exécution de plusieurs logiciels sur un seul serveur :**

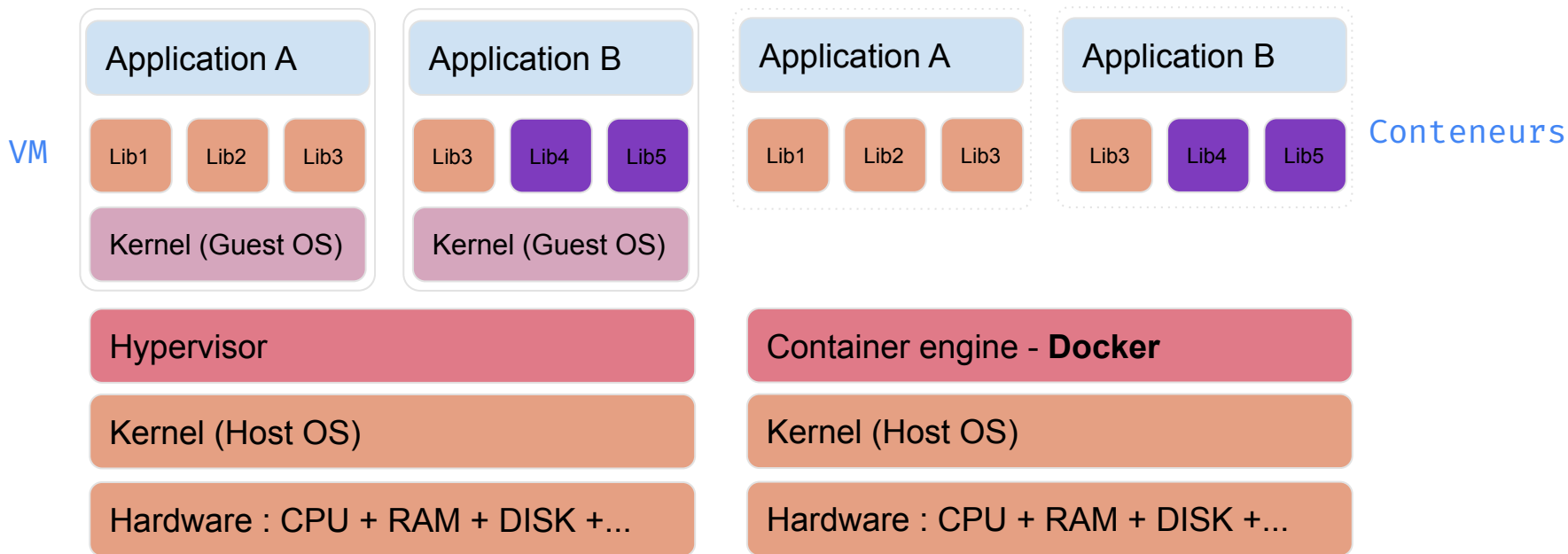
- Gestion et la maintenance séparée des applis ⇒ difficile sans défaillance.
- Risque de sécurité :
  - ◆ Surface d'attaque est plus grande
  - ◆ Principe de moindre privilège et de séparation des tâches rendu difficile.
- Serveur en panne ⇒ Toutes les Applis sont en pannes
- Problèmes d'installation fréquents
- Problèmes de compatibilité Librairies
- ...

# Quelle solution ?



# Conteneurs vs VM

## Virtualisation vs Conteneurisation



# Histoire des conteneurs

- 1979 : Unix V7 chroot ⇒ change le répertoire racine apparent pour le processus en cours d'exécution et fils
- 2000 : FreeBSD Jails ⇒ Permet aux admin sys. de partitionner un l'OS FreeBSD en plusieurs mini-os (Jails)
- 2004 : Solaris Containers ⇒ Implémentation de serveurs virtuels isolés au sein d'une seule instance de l'OS
- 2005 : OpenVZ ⇒ Similaire à Solaris Containers & LXC: exécuter plusieurs instances isolées du système d'exploitation, appelées conteneurs. Utilisé par les VPS
- **2006 : cgroups** ⇒ (Slide Dédié)
- **2008 : namespaces** ⇒ (Slide Dédié)

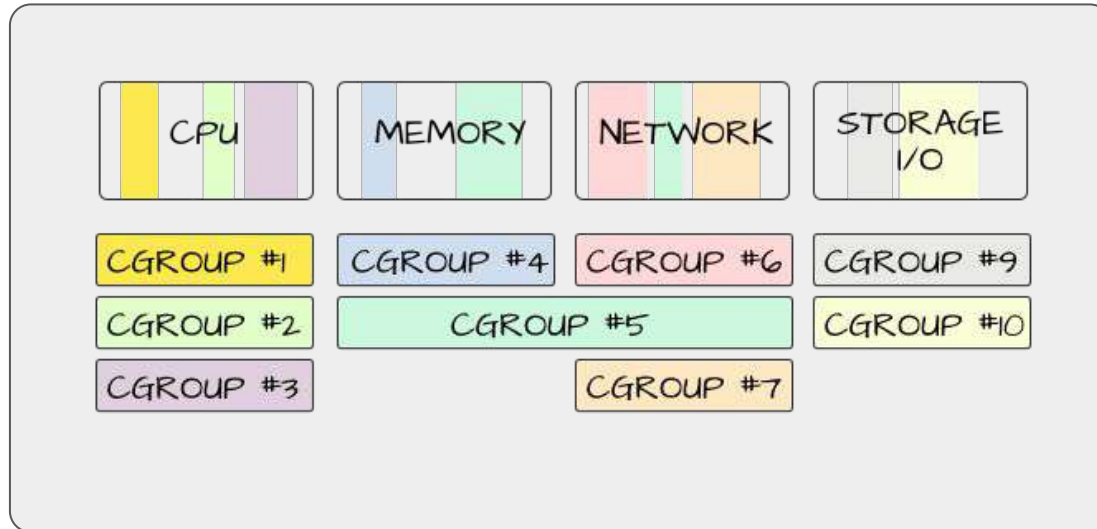
# Histoire des conteneurs

- (...Suite)
- 2008 : LXC ⇒ S'appuie sur chroot, cgroups & namespace pour virtualiser les applis. Très proche de Docker mais différent. Excellent article à ce sujet : <https://earthly.dev/blog/lxc-vs-docker/>
- 2011 : Warden (Cloudfoundry) ⇒ S'appuie sur cgroups & namespace. Different de Docker
- 2013 : Lmctfy (google) ⇒ Collabore avec Docker pour le développement de Libcontainer
- **2013 : Docker** ⇒ Technologie de Containerisation la plus utilisé et la plus répandue



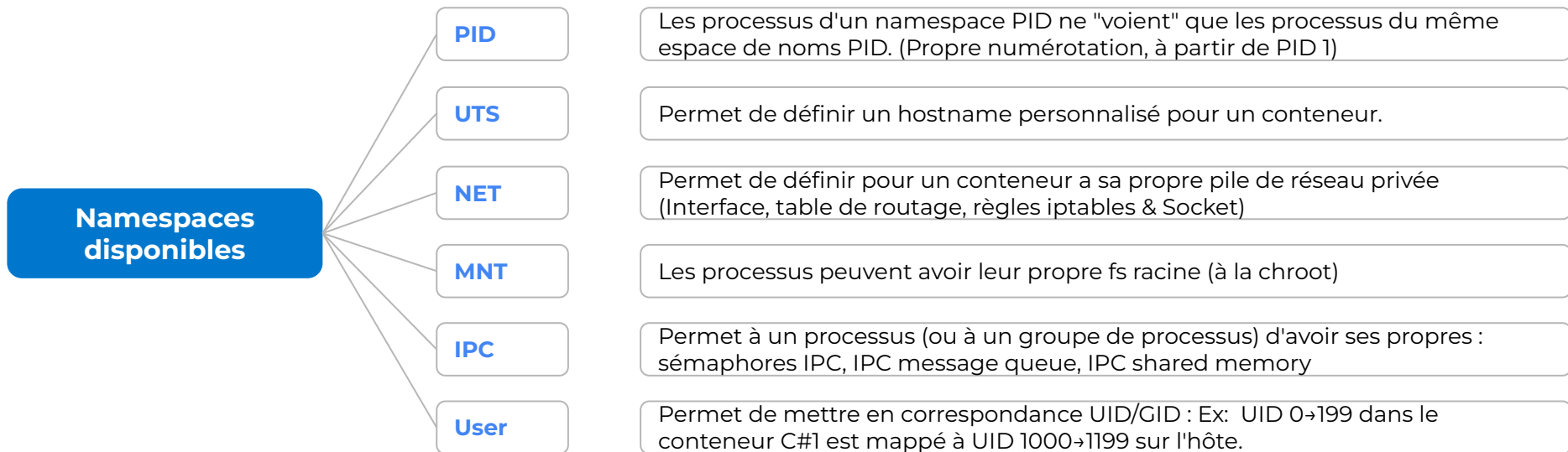
# Namespaces & Cgroups - Kesako

- ❑ **Cgroups (control groups)** : Développé à l'origine par Google et disponible depuis le noyau 2.6.24
- ➔ Permet la limitation, la comptabilité, le contrôle, la priorisation des ressources (CPU, Mémoire, E/S Disque, Réseaux, etc.)



# Namespaces & Cgroups - Kesako

- ❑ **Namespaces** : Développé à l'origine par Bell Lab et disponible depuis le noyau 2.4.19, permettent de disposer d'un environnement distinct. Chaque processus est associé à un espace de nom et ne peut voir ou utiliser que les ressources associées à cet espace de nom.



# Introduction à Kubernetes

- Qu'est ce que Kubernetes ?
- Différents orchestrateurs de conteneur ?
- Historique de Kubernetes
- Pourquoi les Kubernetes ?

# Qu'est ce que Kubernetes aka K8s?



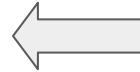
Orchestrateur de conteneurs



  
docker  
Containerd



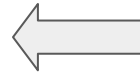
Plateforme de déploiement



Charts



Plateforme extensible



Nouvelles  
ressources



# Pourquoi un Orchestrateur ?

- Gestion du cycle de vie des conteneurs/applications
- Abstraction des hôtes et des cloud providers (clustering)
- Scheduling en fonction des besoins de l'application
- Exemples d'orchestrateur:
  - ◆ Docker Swarm
  - ◆ Rancher
  - ◆ Openshift
  - ◆ Mesos
  - ◆ Kubernetes

# Caractéristiques de Kubernetes

- Déploiement des conteneurs automatisée
- Capacités d'auto-réparation (Self-Healing)
- Déploiements et retours en arrière (rollouts & rollback) automatisés
- Scaling horizontale et équilibrage des charges (Load Balancing)
- Offre un environnement cohérent pour le développement, les tests et la production.
- L'infrastructure est faiblement couplée et chaque composant peut agir comme une unité distincte.
- Offre une plus grande densité d'utilisation des ressources
- Gestion centrée sur l'application
- Infrastructure auto-extensible
- Vous pouvez créer une infrastructure prévisible (IaC)

# Histoire de Kubernetes

- **2013 - 2014** - Créé par Google (3 eng.) Inspiré du système interne Borg de Google
- **Mi-2014** - Annoncé pour la première fois par Google
- **21 juillet 2015** - Open Source - Publication de Kubernetes v1.0
- **2015** - Cloud Native Computing Foundation(CNCF) est créé avec Linux Foundation pour aider à faire progresser la technologie des conteneurs
  - <https://www.cncf.io/> Découvrir les autres projets <https://landscape.cncf.io/>
- **21 Sept 2022** - Dernière version (actuelle): 1.25.2
  - Suivi des versions : <https://kubernetes.io/releases/>

# 7 raisons de s'intéresser à Kubernetes

## **/01** Leader dans l'orchestration de conteneurs

- > K8s reste le leader du marché de l'orchestration de conteneurs - **77%**

## **/02** La base des solutions de cloud computing modernes

- > K8s est considéré comme le Linux du Cloud moderne aujourd'hui

## **/03** Adoption Massive

- > L'enquête annuelle 2021 de la CNCF montre que plus de la moitié des entreprises (69 % en Europe, 55 % en Amérique du Nord, 54 % en Asie) utilisent Kubernetes en production.

## **/04** Devenue une des compétences les plus recherchées

- > Kubernetes est en tête des tendances dans la description des emplois DevOps.



# 7 raisons de s'intéresser à Kubernetes

## **/05** Future très prometteur

- > K8s est une technologie jeune et tranquille à l'avenir prometteur.  
Engouement autour du CNCF

## **/06** Le centre du projet CNCF

- > K8s a été la base de la fondation de CNCF et le premier projet qualifié de Graduated.

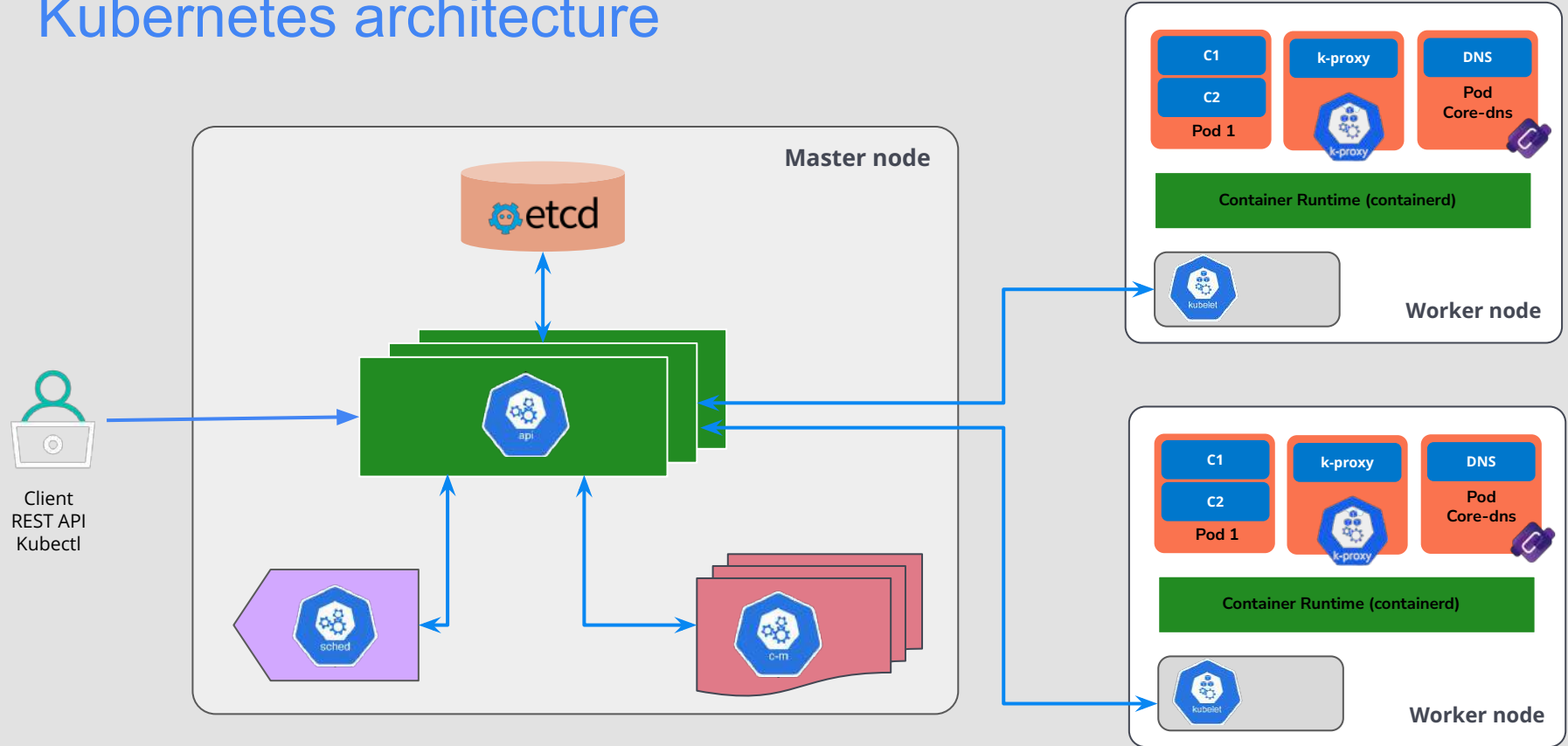
## **/07** Simple à apprendre

- > K8s est complexe mais facile à apprendre, bien documenté, vaste communauté open source.

# L'Architecture de Kubernetes

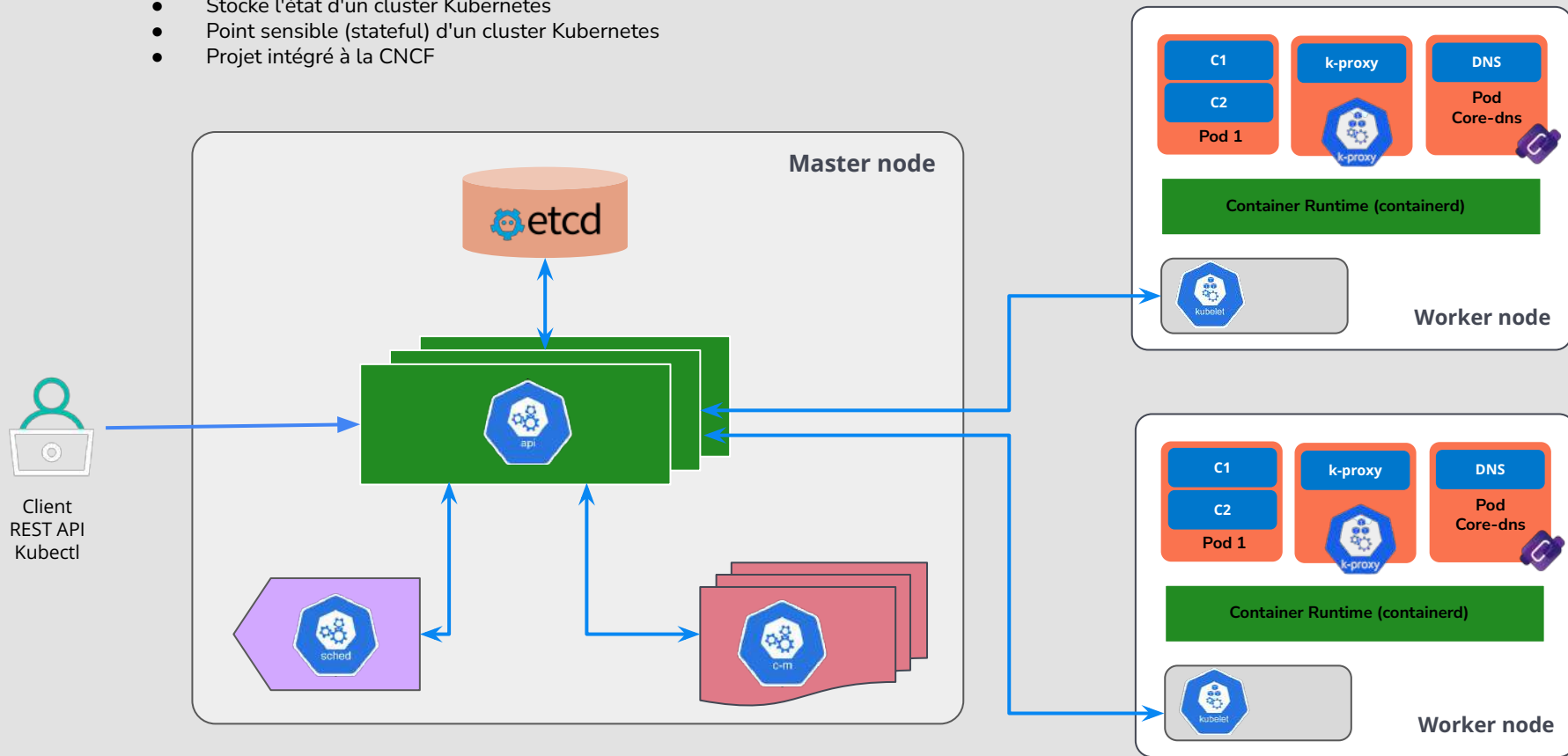
- Différents composants de Kubernetes ?
- Comment sont-ils reliés les uns aux autres ?
- Resources & controllers
- Déployer un cluster Kubernetes

# Kubernetes architecture



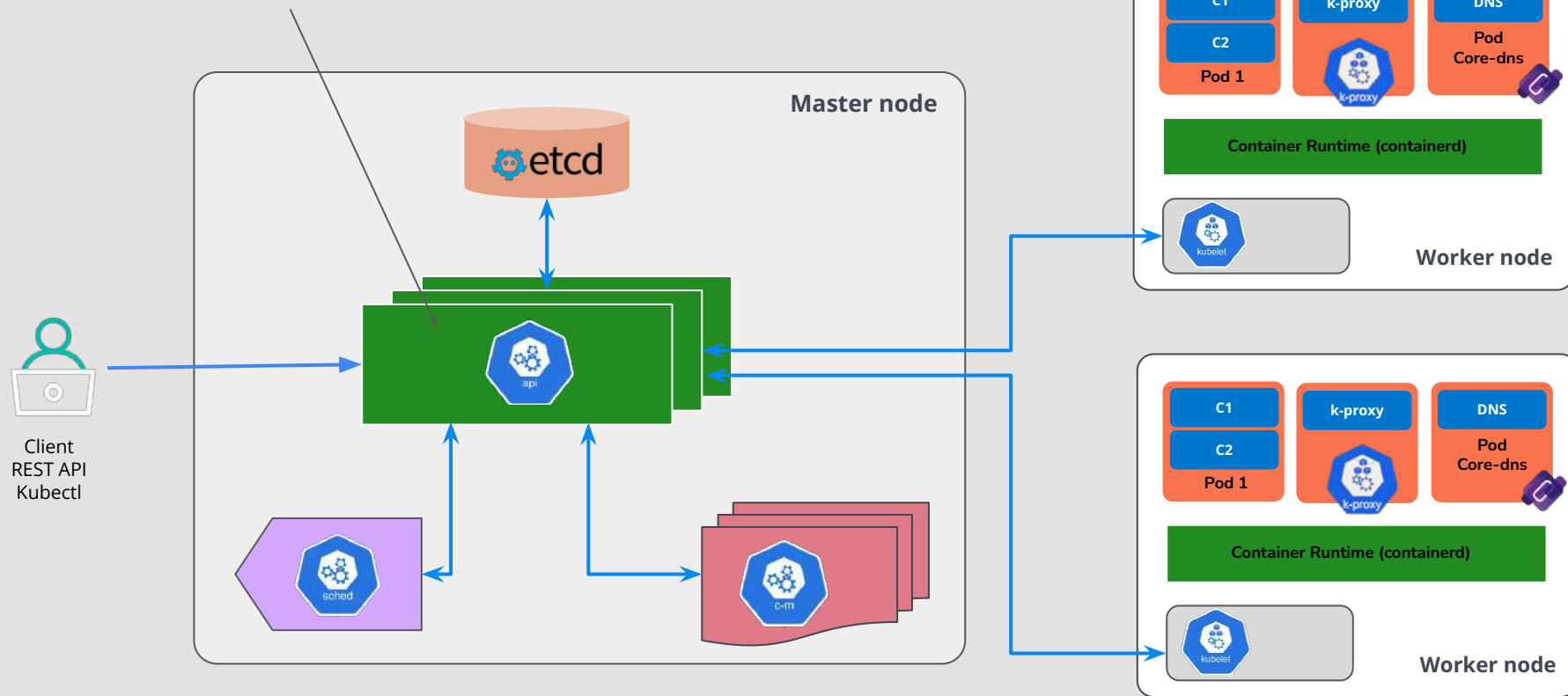
## etcd

- Base de données NoSQL distribué de type Clé/Valeur (Key Value Store)
- Stocke l'état d'un cluster Kubernetes
- Point sensible (stateful) d'un cluster Kubernetes
- Projet intégré à la CNCF



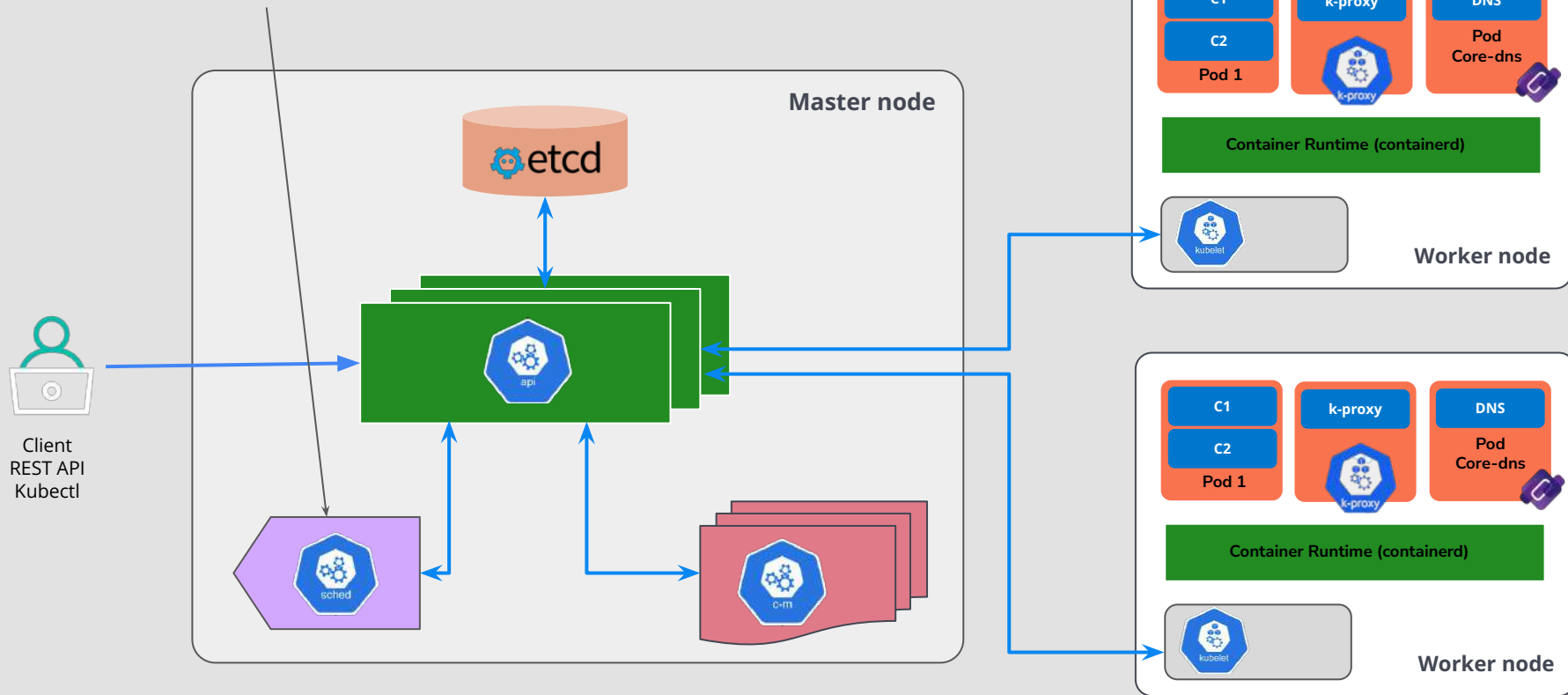
## Kube-apiserver

- Les configurations d'objets (Pods, Service, RC, etc.) se font via l'API server
- Un point d'accès à l'état du cluster aux autres composants via une API REST
- Tous les composants sont reliés à l'API server



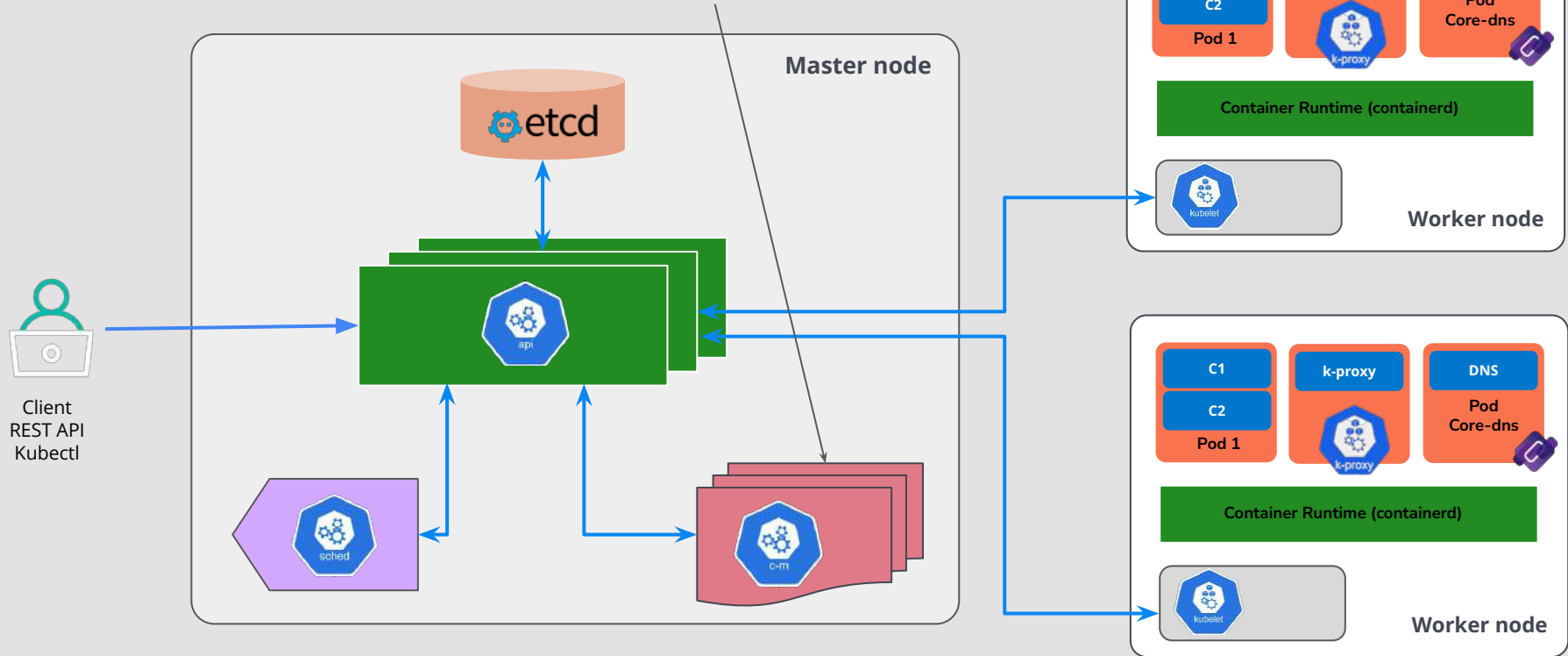
## Kube Scheduler

- Planifie les ressources sur le cluster
- En fonction de règles implicites (CPU, RAM, stockage disponible, etc.)
- En fonction de règles explicites (règles d'affinité et anti-affinité, labels, etc.)



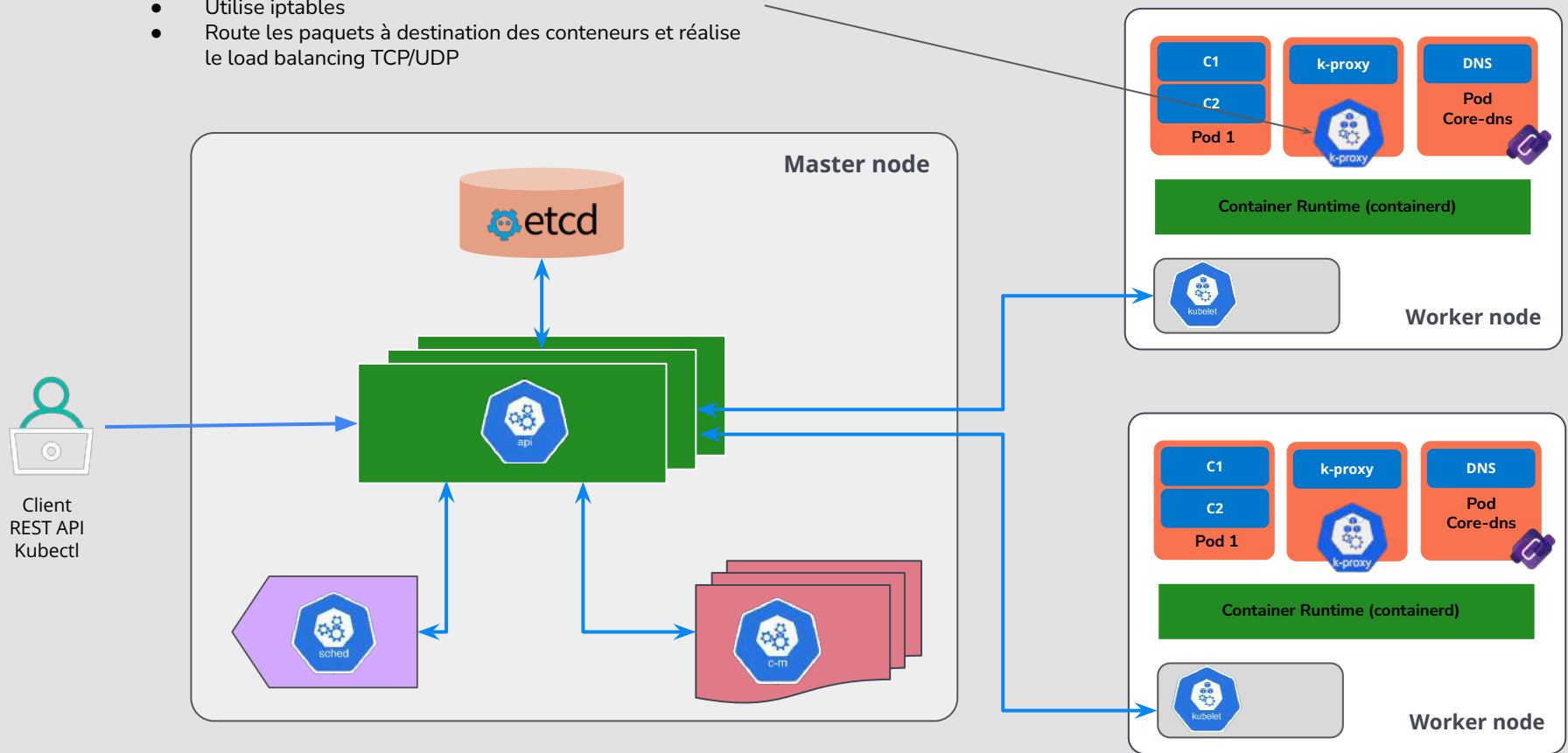
## Kube-controller-manager

- Boucle infinie qui contrôle l'état d'un cluster
- Effectue des opérations pour atteindre un état donné
- De base dans Kubernetes plusieurs controllers : replication controller, endpoints controller, namespace controller, service accounts controller...



## Kube Proxy

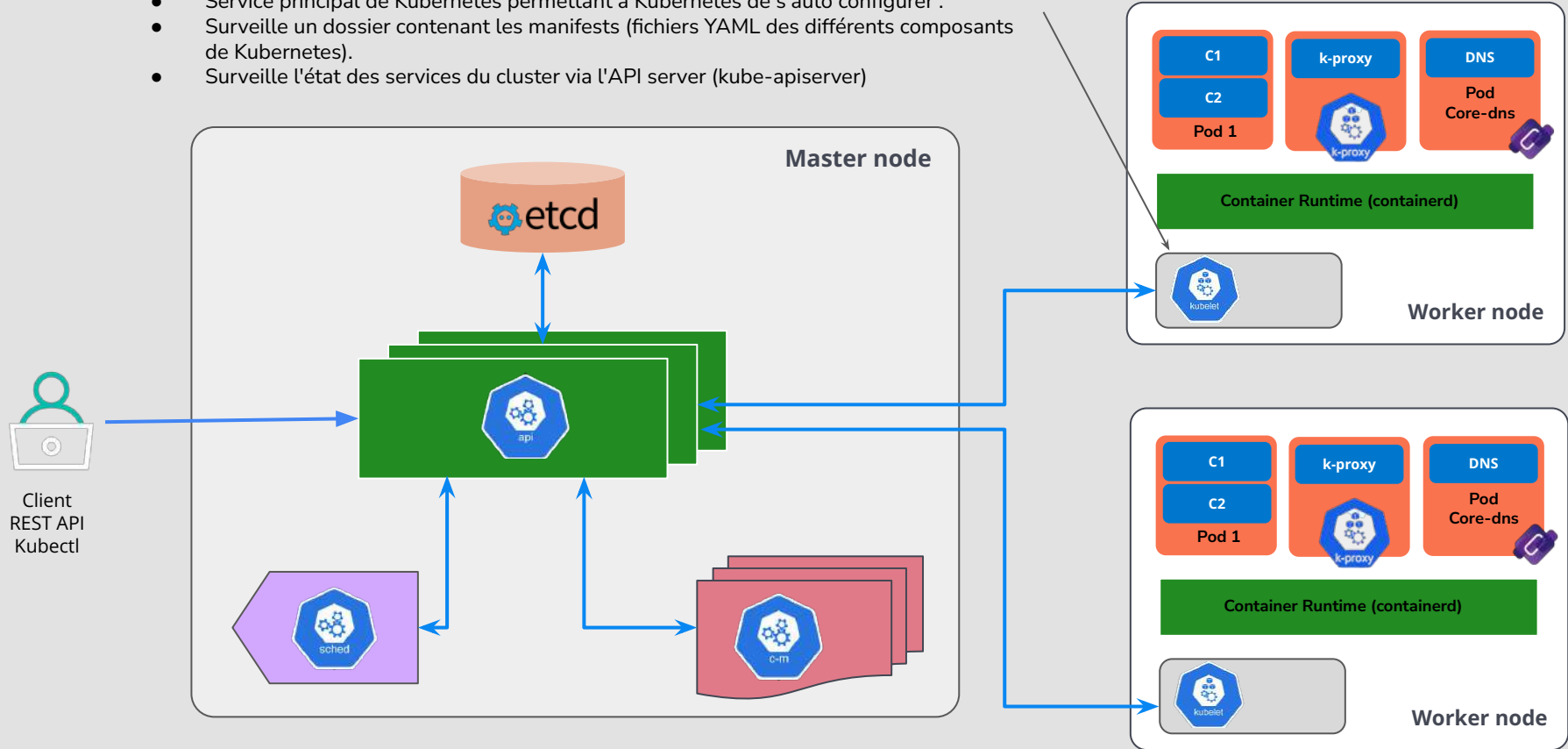
- Responsable de la publication des Services
- Utilise iptables
- Route les paquets à destination des conteneurs et réalise le load balancing TCP/UDP





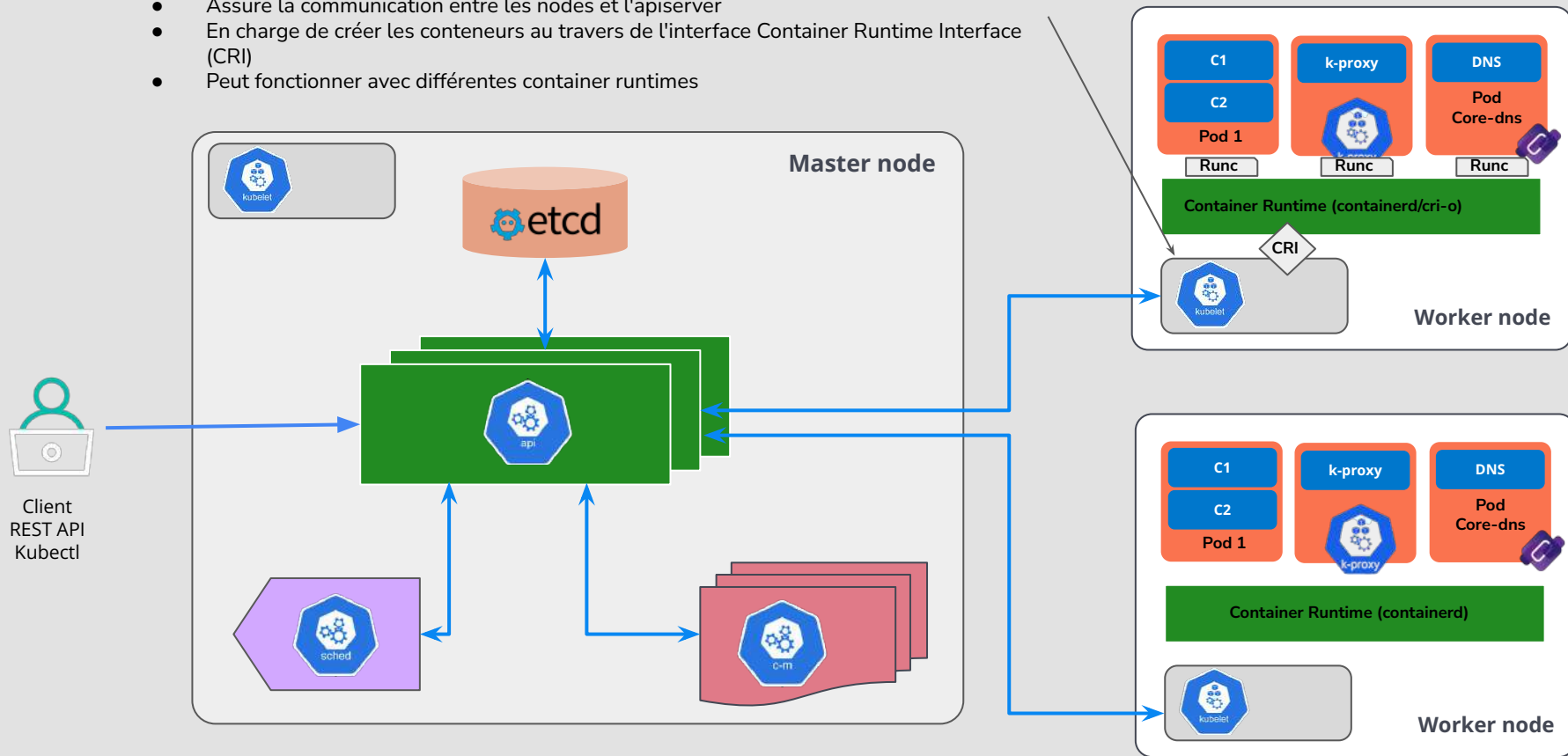
## Kubelet

- Agent fonctionnant sur chaque nœud, gère les conteneurs sur le nœud
- Service principal de Kubernetes permettant à Kubernetes de s'auto configurer :
- Surveille un dossier contenant les manifests (fichiers YAML des différents composants de Kubernetes).
- Surveille l'état des services du cluster via l'API server (kube-apiserver)



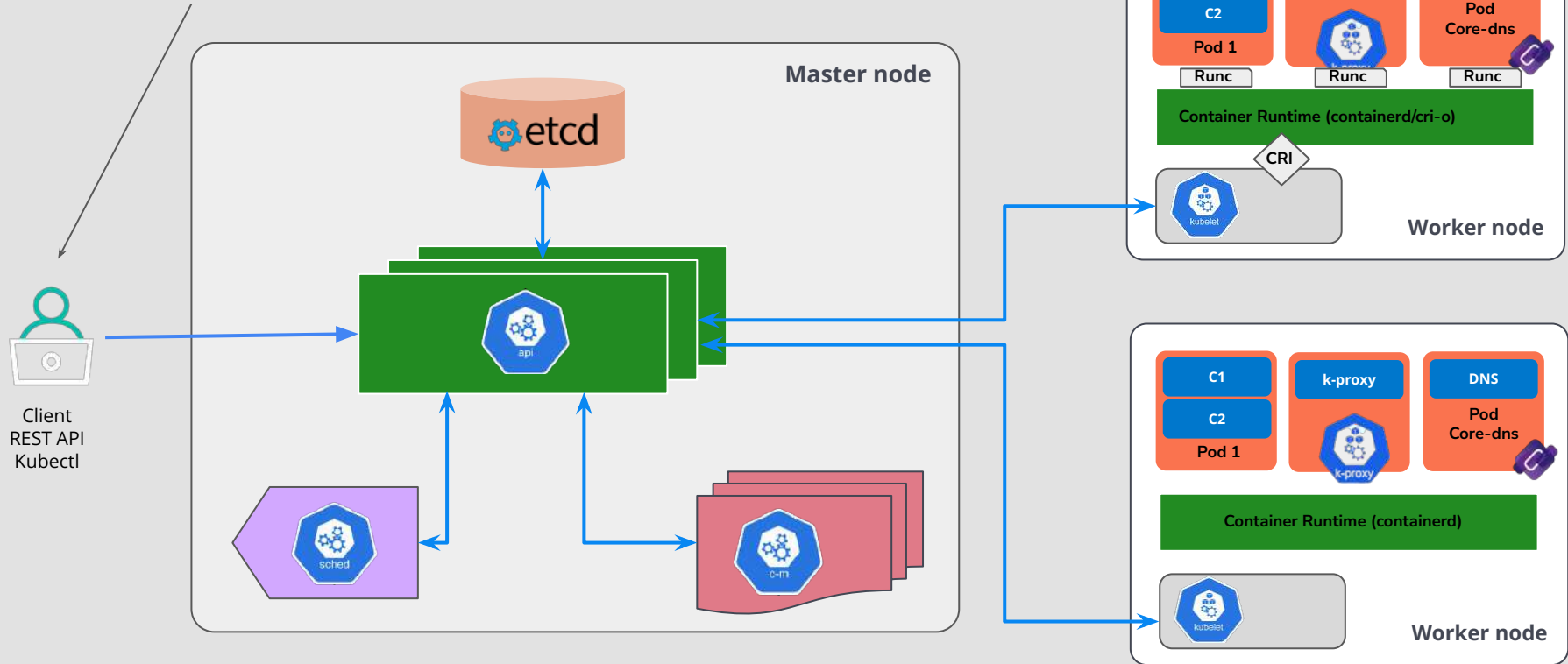
## Kubelet (Suite)

- Applique les modifications si besoin (upgrade, rollback).
- Assure la communication entre les nodes et l'apiserver
- En charge de créer les conteneurs au travers de l'interface Container Runtime Interface (CRI)
- Peut fonctionner avec différentes container runtimes

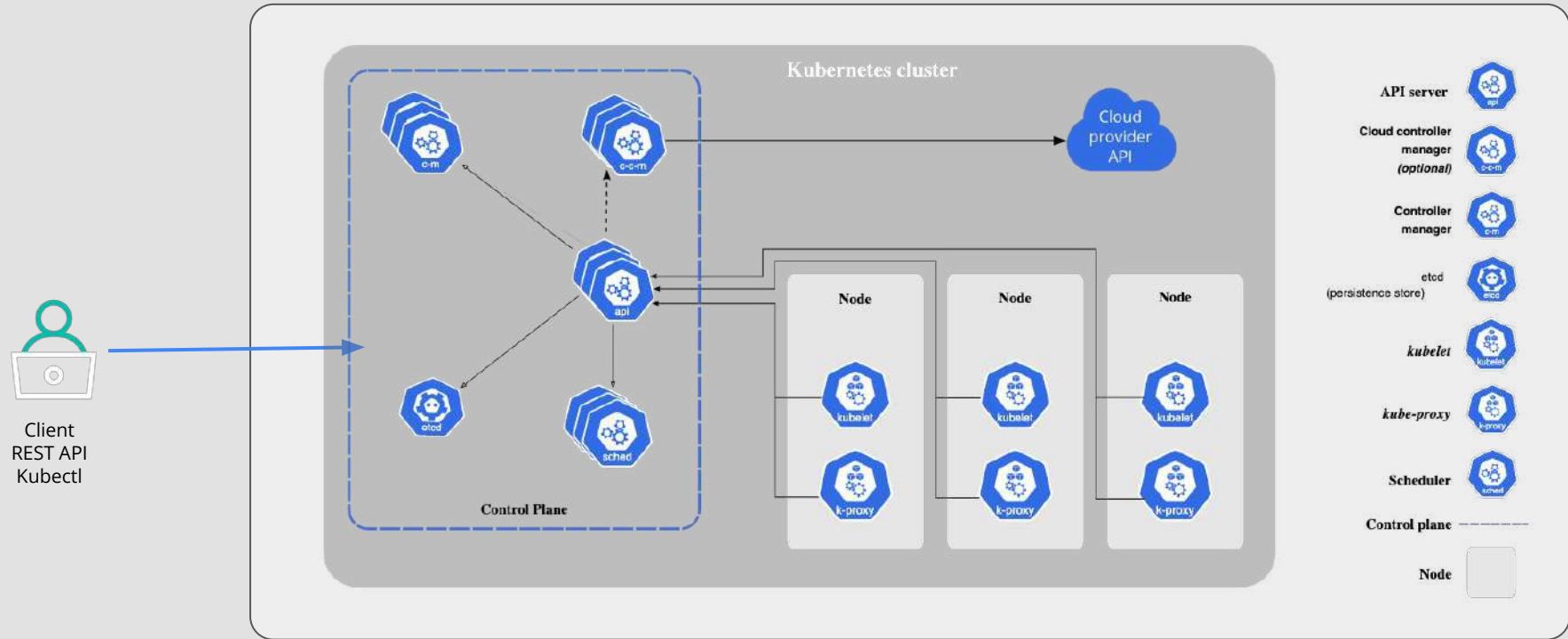


## Kubectl

- Ligne de commande permettant de piloter un cluster Kubernetes
- Syntaxes intuitives
- Référence <https://kubernetes.io/docs/reference/generated/kubectrl/kubectrl-commands>



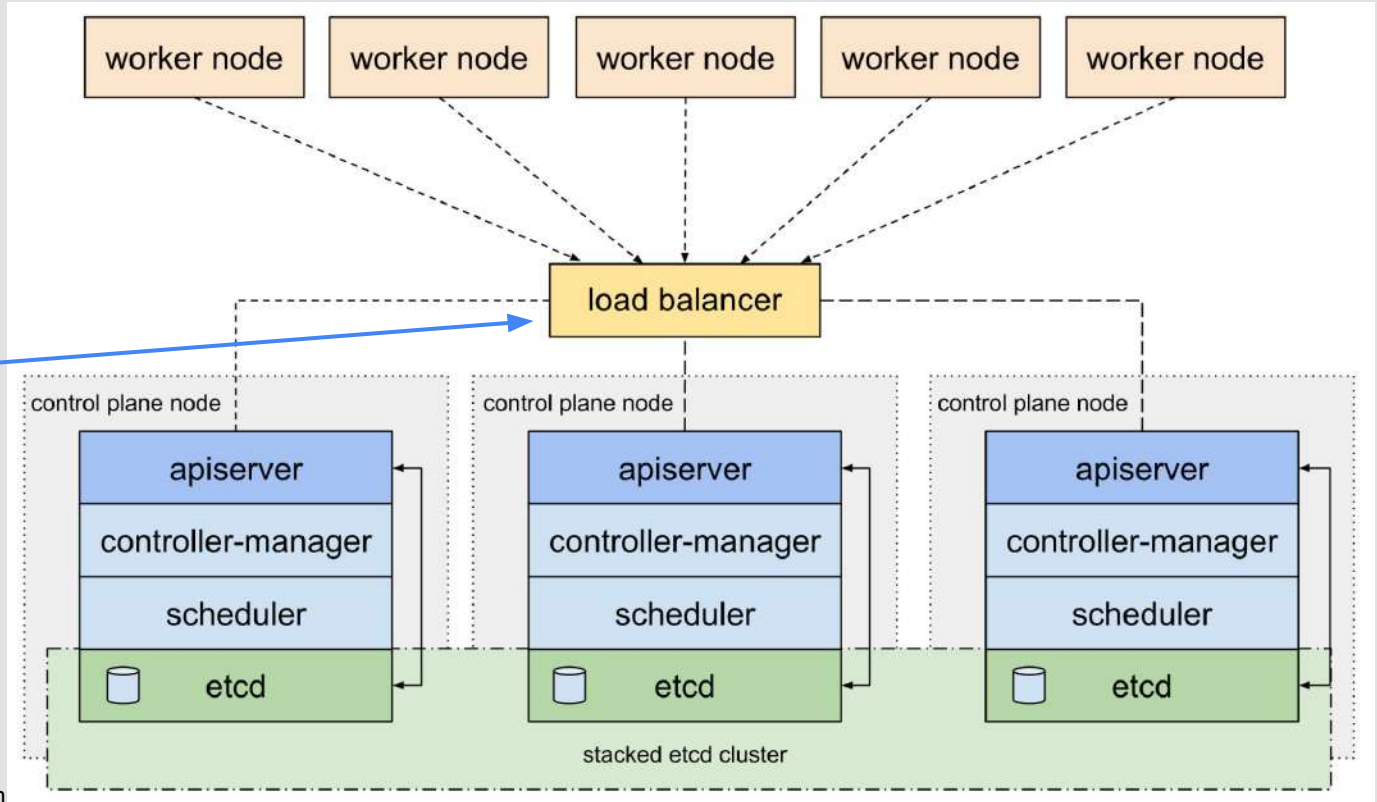
# Kubernetes dans le Cloud (GKE, ESK, AKS)



# Kubernetes (Haute Disponibilité)



Client  
REST API  
Kubectl



# Demo



- Exploration des Composants K8s
- Cluster On-premise
- Cluster dans le Cloud

# Mise en oeuvre de Kubernetes

- Vue d'ensemble des ressources K8s
- Contexte d'utilisation
- K8s API, API version,
- Fichier YAML

# Les objets Kubernetes

- **Objets Kubernetes** : Des entités persistantes dans le système Kubernetes (dans ETCD)
- K8s utilise ces entités pour représenter l'État de votre cluster.
- Objets peuvent décrire :
  - ◆ Les app conteneurisées en cours d'exécution (et sur quels nodes)
  - ◆ Les ressources disponibles pour ces app
  - ◆ Les politiques relatives au comportement de ces applications : politiques de redémarrage, politique d'upgrade, et de fault-tolerance.
  - ◆ ...

- Rest Api
- kubectl



Création  
Modification  
Suppression



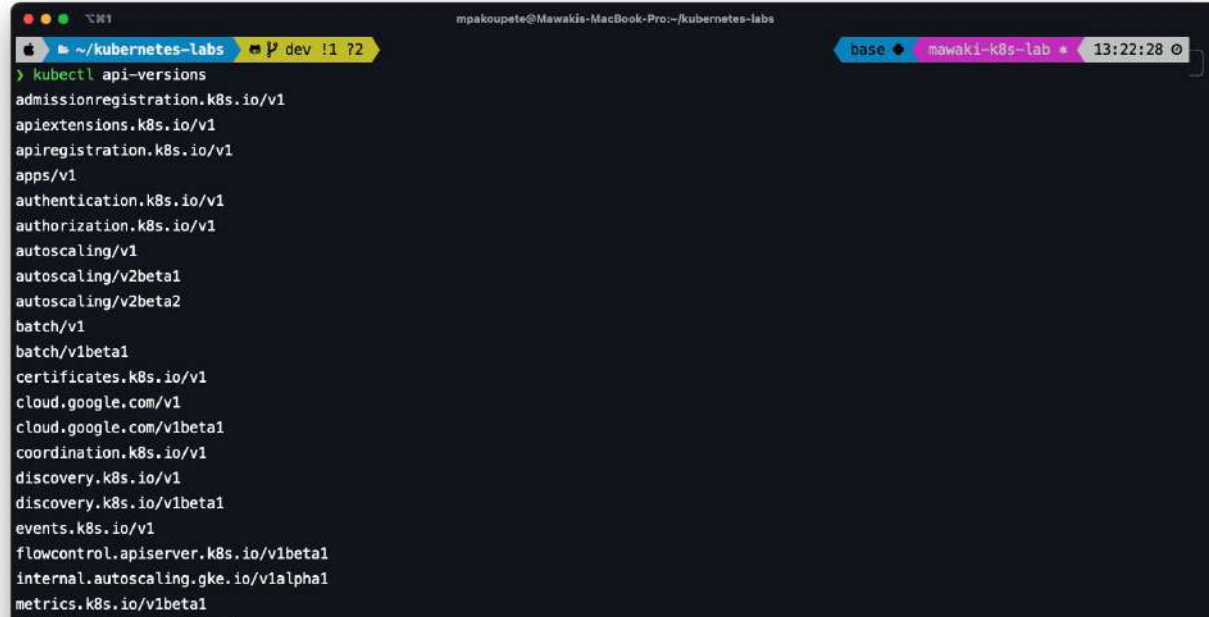
Api Server





# Les objets Kubernetes (Classement des ressources)

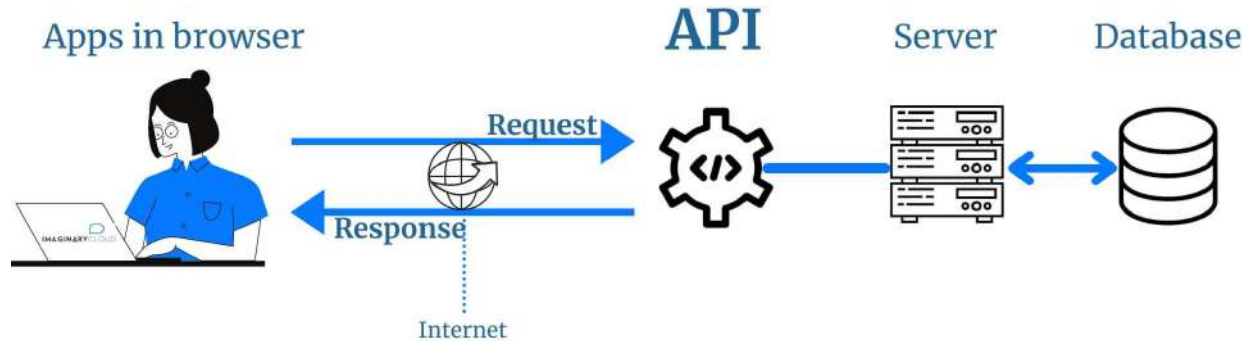
→ Les ressources sont classées par Version d'API Kubernetes

A terminal window with a dark background. The title bar shows 'mpakoupete@Mawaki-MacBook-Pro: ~/kubernetes-labs'. The terminal has two tabs: 'base' and 'mawaki-k8s-lab'. The command 'kubectl api-versions' has been executed, resulting in a list of Kubernetes API versions. The output is as follows:

```
> kubectl api-versions
admissionregistration.k8s.io/v1
apiextensions.k8s.io/v1
apiregistration.k8s.io/v1
apps/v1
authentication.k8s.io/v1
authorization.k8s.io/v1
autoscaling/v1
autoscaling/v2beta1
autoscaling/v2beta2
batch/v1
batch/v1beta1
certificates.k8s.io/v1
cloud.google.com/v1
cloud.google.com/v1beta1
coordination.k8s.io/v1
discovery.k8s.io/v1
discovery.k8s.io/v1beta1
events.k8s.io/v1
flowcontrol.apiserver.k8s.io/v1beta1
internal.autoscaling.gke.io/v1alpha1
metrics.k8s.io/v1beta1
```

# Les objets Kubernetes (Classement des ressources)

→ Les ressources sont classées par Version d'API Kubernetes



Une API spécifie les types de demandes qu'une application (page web ou application mobile) peut adresser à une autre et définit en outre la manière d'effectuer ces demandes, les formats de données à utiliser et les pratiques que les utilisateurs doivent suivre.

3 modèles principaux lors de la création d'une API : **RPC** (Remote Procedure Call), **REST** (Representational State Transfer) et **GraphQL**.

# Les objets Kubernetes (Classement des ressources)

→ Les ressources sont classées par Version d'API Kubernetes



NAME	SHORTNAME	APIVERSION	NAMESPACE	KIND
bindings		v1	true	Binding
componentstatuses	cs	v1	false	ComponentStatus
configmaps	cm	v1	true	ConfigMap
endpoints	ep	v1	true	Endpoints
events	ev	v1	true	Event
limitranges	limits	v1	true	LimitRange
namespaces	ns	v1	false	Namespace
nodes	no	v1	false	Node
persistentvolumeclaims	pvc	v1	true	PersistentVolumeClaim
persistentvolumes	pv	v1	false	PersistentVolume
Pods	po	v1	true	Pod
podtemplates		v1	true	PodTemplate
replicationcontrollers	rc	v1	true	ReplicationController
resourcequotas	quota	v1	true	ResourceQuota
secrets		v1	true	Secret
serviceaccounts	sa	v1	true	ServiceAccount
services	svc	v1	true	Service
mutatingwebhookconfigurations		admissionregistration.k8s.io/v1	false	MutatingWebhookConfiguration
validatingwebhookconfigurations		admissionregistration.k8s.io/v1	false	ValidatingWebhookConfiguration
customresourcedefinitions	crd,crds	apiextensions.k8s.io/v1	false	CustomResourceDefinition
apiservices		apiregistration.k8s.io/v1	false	APIService
controllerrevisions		apps/v1	true	ControllerRevision
daemonsets	ds	apps/v1	true	DaemonSet
deployments	deploy	apps/v1	true	Deployment
replicasets	rs	apps/v1	true	ReplicaSet

# Structure des objets Kubernetes

→ Dans le fichier **.yaml** de l'objet Kubernetes que vous souhaitez créer, vous devez définir les valeurs des champs suivants :

- **apiVersion** - La version de l'API Kubernetes que vous utilisez pour créer cet objet.
- **kind** - Le type d'objet que vous souhaitez créer.
- **metadata** - Données permettant d'identifier l'objet de manière unique. Ex : nom, UID et/ou namespace.
- **spec** - L'état que vous souhaitez pour l'objet.
  - Contenu différent en fonction des ressources

# Structure des objets Kubernetes (Ex : Ressource Pod)

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-demo
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
```

# Structure des objets Kubernetes (Ex : Ressource Pod)

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-demo
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
```

# Structure des objets Kubernetes (Ex : Ressource Pod)

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-demo
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
```

# Structure des objets Kubernetes (Ex : Ressource Pod)

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-demo
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
```



# Structure des objets Kubernetes (Ex : Ressource Pod)

```
apiVersion: v1
```

```
kind: Pod
```

```
metadata:
```

```
  name: nginx-demo
```

```
spec:
```

```
  containers:
```

```
  - name: nginx
```

```
    image: nginx:1.14.2
```

```
apiVersion: apps/v1
```

```
kind: Deployment
```

```
metadata:
```

```
  name: nginx-deployment
```

```
spec:
```

```
  selector:
```

```
    matchLabels:
```

```
      app: nginx
```

```
  replicas: 2
```

```
  template:
```

```
    metadata:
```

```
      labels:
```

```
        app: nginx
```

```
    spec:
```

```
      containers:
```

```
      - name: nginx
```

```
        image: nginx:1.14.2
```

# Les objets Kubernetes (Catégories)

## /01 Ressources Workload



- Pod
- Deployment
- ReplicaSet
- StatefulSet
- DaemonSet
- Jobs
- CronJob

## /03 Ressources Config & Stockage



- Persistent Volumes
- Persistent Volumes Claim
- Storage Classes
- ConfigMaps
- Secrets

## /02 Ressources Services & Networking



- Service
- Ingress - Ingress Controllers
- Network Policies

## /04 Ressources Cluster



- Namespace
- Role & Role Binding
- Cluster Role & Cluster Role Binding
- Service Account

## /05 Ressources Metadata



- Custom Resource Definition (CRD)
- PodTemplate

# Avoir un Cluster Kubernetes

Plusieurs moyens d'avoir un Cluster kubernetes : **On-permise** ou dans **le Cloud**

Installation On Premise



Minikube



Kind



Kubeadm

Offres Cloud



Azure Kubernetes Service (AKS)



Amazon Elastic Kubernetes Service (EKS)



Google Kubernetes Engine (GKE)



IBM Cloud  
Kubernetes Service



# Avoir un Cluster Kubernetes de test



Minikube

- Installation d'un hyperviseur
  - Virtualbox, VMWare, **HyperKit**, ...
- Installation de **kubectl** (Client pour communiquer avec le cluster via API server)
  - <https://kubernetes.io/docs/tasks/tools/install-kubectl/>
- Installation de **MiniKube**
  - <https://minikube.sigs.k8s.io/docs/start/>
- Démarrer Minikube et

```
$ minikube start
```

```
$ kubectl get pods -A
```



# Avoir un Cluster Kubernetes de test

<https://labs.play-with-k8s.com>

- Des plateformes de test disponibles sur Internet
  - Online playground Play With K8s a.k.a PWK
  - Créé par Marcos Lilljedahl et Jonathan Leibiusky
    - <https://labs.play-with-k8s.com>
  - Conditions : avoir un compte Docker ou GitHub
  - Mise en place d'un cluster en 1 minute avec le binaire **kubeadm**
    - 5 instances utilisables pendant 4h

# Les choix pour un cluster de production

## Installation On Premise



Kubeadm



**RANCHER**



VMware Tanzu



HPE Ezmeral  
Container Platform

## Offres Cloud



Azure Kubernetes Service (AKS)



Amazon Elastic Kubernetes Service (EKS)



Google Kubernetes Engine (GKE)



IBM Cloud  
Kubernetes Service



linode



DigitalOcean



Alibaba Cloud

# Context d'Utilisateur

```
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0F(...)==
    server: https://192.168.56.11:6443
    name: k8s-lab1
contexts:
- context:
    cluster: k8s-lab1
    namespace: default
    user: kubernetes-admin
    name: kubernetes-admin@k8s-lab1
current-context: kubernetes-admin@k8s-lab1
kind: Config
preferences: {}
users:
- name: kubernetes-admin
  user:
    client-certificate-data: LS0tLS1CRUdJTiBDRVJUSUZ(...)==
    client-key-data: LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSU(...)==
```

# Context d'Utilisateur (Exemple multi-context)

```
mpakoupete@Mawakis-MacBook-Pro:~/kubernetes-labs

~/kubernetes-labs 12 01:17:34
$ kubectl config current-context
gke_...-c_mawaki-k8s-lab

~/kubernetes-labs 12 01:17:50
$ kubectl config get-contexts
CURRENT  NAME
*        gke_...-c_cluster-1
          gke_...-c_mawaki-lab
          gke_...-c_mawaki-k8s
          gke_...-c_mawaki-k8s-lab
          gke_...-c_mawaki-lab
kind-kind

CLUSTER
docker-desktop
gke_...-c_cluster-1
gke_...-c_mawaki-lab
gke_...-c_mawaki-k8s
gke_...-c_mawaki-k8s-lab
gke_...-c_mawaki-lab
kind-kind

AUTHINFO
docker-desktop
gke_...-c_cluster-1
gke_...-c_mawaki-lab
gke_...-c_mawaki-k8s
gke_...-c_mawaki-k8s-lab
gke_...-c_mawaki-lab
kind-kind

NAMESPACE
gke_...-c_cluster-1
gke_...-c_mawaki-lab
gke_...-c_mawaki-k8s
gke_...-c_mawaki-k8s-lab
gke_...-c_mawaki-lab
kind-kind

~/kubernetes-labs 12 01:17:53
$ kubectl config use-context docker-desktop
Switched to context "docker-desktop".

~/kubernetes-labs 12 01:18:39
$ kubectl config get-contexts
CURRENT  NAME
*        docker-desktop
          gke_...-c_cluster-1
          gke_...-c_mawaki-lab
          gke_...-c_mawaki-k8s
          gke_...-c_mawaki-k8s-lab
          gke_...-c_mawaki-lab
kind-kind

CLUSTER
docker-desktop
gke_...-c_cluster-1
gke_...-c_mawaki-lab
gke_...-c_mawaki-k8s
gke_...-c_mawaki-k8s-lab
gke_...-c_mawaki-lab
kind-kind

AUTHINFO
docker-desktop
gke_...-c_cluster-1
gke_...-c_mawaki-lab
gke_...-c_mawaki-k8s
gke_...-c_mawaki-k8s-lab
gke_...-c_mawaki-lab
kind-kind

NAMESPACE
gke_...-c_cluster-1
gke_...-c_mawaki-lab
gke_...-c_mawaki-k8s
gke_...-c_mawaki-k8s-lab
gke_...-c_mawaki-lab
kind-kind
```



# Ressources Workload K8s

- Pods
- ReplicaSet
- Deployment
- Stratégies de Déploiement
- StatefulSet
- DaemonSet
- Jobs

# Pod

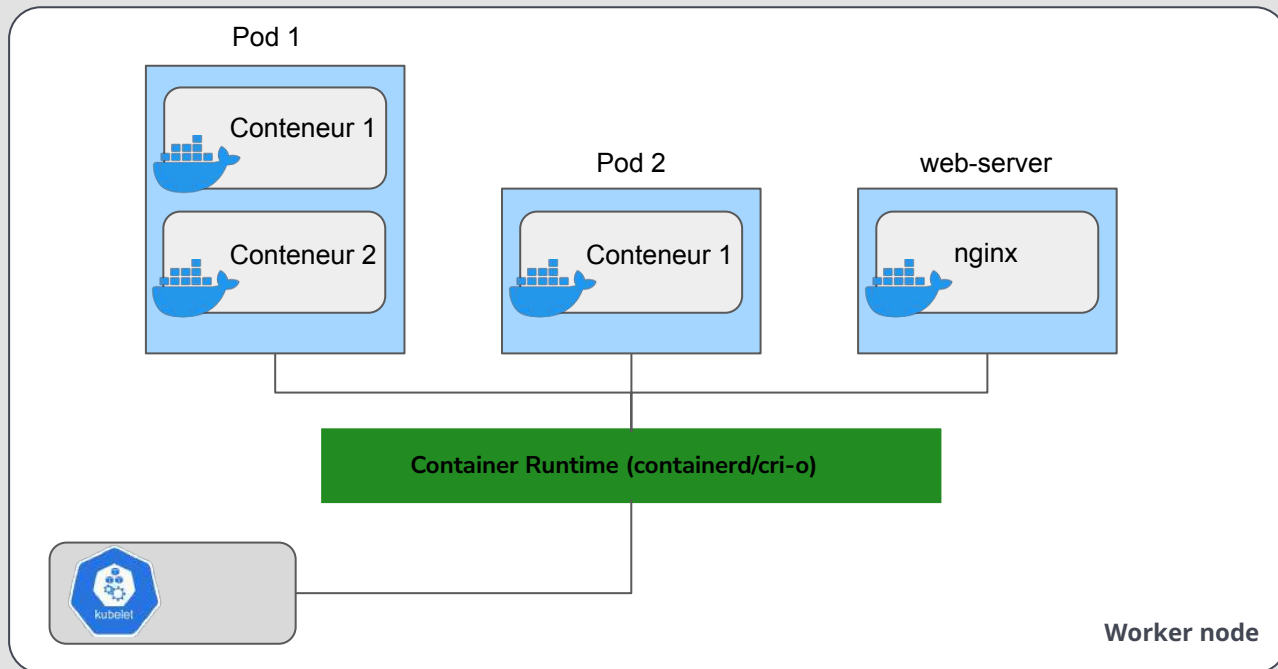
- Plus petites unités de calcul déployables que vous pouvez créer et gérer dans Kubernetes.
- Groupe d'un ou plusieurs conteneurs, avec :
  - ◆ Des ressources de stockage et de réseau partagées
  - ◆ Une spécification sur la façon d'exécuter les conteneurs.
- Pod est similaire à un ensemble de conteneurs avec des espaces de noms partagés et des volumes de système de fichiers partagés.
- Éphémère, lorsqu'un pod est arrêté, il ne peut pas être restauré.

```
apiVersion: v1
kind: Pod
metadata:
  name: web-server
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
```

# Pod

→ Partage dans un Pod :

- ◆ Même namespace réseau
- ◆ Même adresse IP
- ◆ Même Ports réseaux
- ◆ Même volumes de stockage partagés



# Pod (cycle de vie - État des Pods)

- **Pending (En attente)** : le pod a été créé et accepté par le cluster, mais un ou plusieurs de ses conteneurs ne sont pas encore actifs. Cette phase comprend le temps passé à planifier le pod sur un nœud et à télécharger des images.
- **Running (En cours d'exécution)** : le pod a été associé à un nœud et tous les conteneurs ont été créés. Au moins un conteneur est en cours d'exécution, de démarrage ou de redémarrage.
- **Succeeded (Réussite)** : tous les conteneurs du pod se sont correctement arrêtés. Les pods arrêtés ne redémarrent pas.
- **Failed (Échec)** : tous les conteneurs du pod se sont arrêtés, et au moins un conteneur s'est arrêté sur une défaillance. Un conteneur "échoue" s'il s'arrête dans un état différent de zéro.
- **Unknown (Inconnu)** : l'état du pod ne peut pas être déterminé.

# Pod (Démo)

```
> kubectl run web-server --image nginx  
pod/web-server created
```

```
> kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
web-server	0/1	Pending	0	5s

```
> kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
web-server	0/1	ContainerCreating	0	8s

```
> kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
web-server	1/1	Running	0	11s

```
>  
~/kubernetes-labs P dev !21 74 base • kubernet... 03:10:28  
> kubectl run busypod --image busybox -- sleep 5  
pod/busypod created  
  
~/kubernetes-labs P dev !21 74 base • kubernet... 03:10:32  
> kubectl get pods  
NAME      READY   STATUS    RESTARTS   AGE  
busypod    1/1     Running   0           4s  
web-server 1/1     Running   0          11m  
  
~/kubernetes-labs P dev !21 74 base • kubernet... 03:10:36  
> kubectl get pods  
NAME      READY   STATUS    RESTARTS   AGE  
busypod    1/1     Running   0           6s  
web-server 1/1     Running   0          11m  
  
~/kubernetes-labs P dev !21 74 base • kubernet... 03:10:38  
> kubectl get pods  
NAME      READY   STATUS    RESTARTS   AGE  
busypod    0/1     Completed 0           8s  
web-server 1/1     Running   0          11m
```

# Pod (cycle de vie - État des Conteneurs)

- **Running (En cours d'exécution)** : le conteneur s'exécute sans problème..
- **Terminated (Terminé)** : a commencé son exécution, puis s'est exécuté jusqu'au bout ou a échoué pour une raison quelconque.
- **Waiting (En attente)** : Si un conteneur n'est pas dans l'état Running ou Terminated, il est dans l'état Waiting.

# Pod (cycle de vie - État des Conteneurs)

État du Pod web-server

```
mpakoupete@Mawakis-MacBook-Pro: ~/kubernetes-labs
> kubectl describe pod web-server

Name:          web-server
Namespace:     default
Priority:       0
Node:          k8s-worker-2/192.168.56.13
Start Time:    Wed, 26 Oct 2022 03:06:55 +0200
Labels:        run=web-server
Annotations:   cni.projectcalico.org/podIP: 192.168.114.130/32
               cni.projectcalico.org/podIPs: 192.168.114.130/32
Status:        Running
IP:            192.168.114.130
IPs:
  IP: 192.168.114.130
Containers:
  web-server:
    Container ID:  containerd://1541e9e8b0d981e93ef68b6401ddebeb0d7a5ff984fd498010ffea7791eb2d2b
    Image:         nginx
    Image ID:      docker.io/library/nginx@sha256:24a1618e0ac445d24eda699881d7de1364e45641be98e9b1db5f7cad6b7b29b2
    Port:          <none>
    Host Port:     <none>
    State:         Running
                   Started:    Wed, 26 Oct 2022 03:06:57 +0200
    Ready:         True
    Restart Count: 0
    Environment:   <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-5kbnl (ro)
Conditions:
  Type           Status
  Initialized    True
  Ready          True
```

État du conteneur Nginx

# Pod (cycle de vie - État des Conteneurs)

État du Pod busypod

```
mpakoupete@Mawaki-MacBook-Pro:~/kubernetes-labs
> kubectl describe pod busypod
Name:          busypod
Namespace:     default
Priority:       0
Node:          k8s-worker-3/192.168.56.14
Start Time:    Wed, 26 Oct 2022 03:19:03 +0200
Labels:        run=busypod
Annotations:   cni.projectcalico.org/podIP: 192.168.117.195/32
               cni.projectcalico.org/podIPs: 192.168.117.195/32
Status:        Running
IP:            192.168.117.195
IPs:
  IP: 192.168.117.195
Containers:
  busypod:
    Container ID:  containerd://6ba1af636dab42f9e64d945e749d937d1448b6ca95315f6a99116ae8fc8bab5
    Image:          busybox
    Image ID:       docker.io/library/busybox@sha256:21f1dd2b8794a5f9c41a05b2857310e0ba33081ddaab2e134f37900a106d7abf
    Port:           <none>
    Host Port:      <none>
    Args:
      sleep
      10
    State:          Waiting
    Reason:         CrashLoopBackOff
    Last State:     Terminated
    Reason:         Completed
    Exit Code:      0
    Started:        Wed, 26 Oct 2022 03:52:27 +0200
    Finished:       Wed, 26 Oct 2022 03:52:37 +0200
    Ready:          false
```

État du conteneur



# Pod (Events - création de Conteneurs)

Events du Pod web-server

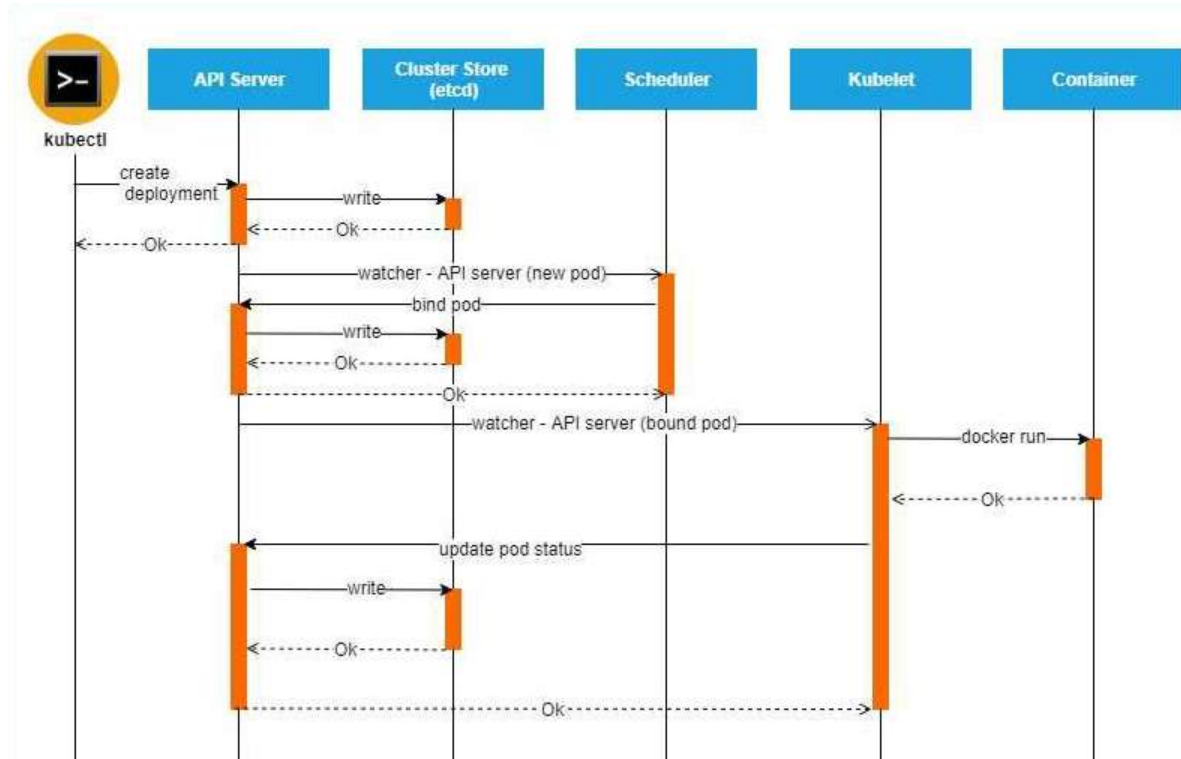
```
mpakoupete@Mawakis-MacBook-Pro:~/kubernetes-labs
~/kubernetes-labs dev 121 74 base • kubernetes-admin@k8s-lab 04:00:17
> kubectl describe pod web-server | grep -A20 ^Events
Events:
  Type       Reason      Age    From          Message
  ----       -
Normal      Scheduled   53m    default-scheduler Successfully assigned default/web-server to k8s-worker-2
Normal      Pulling     53m    kubelet       Pulling image "nginx"
Normal      Pulled      53m    kubelet       Successfully pulled image "nginx" in 2.150895676s
Normal      Created     53m    kubelet       Created container web-server
Normal      Started     53m    kubelet       Started container web-server

~/kubernetes-labs dev 121 74 base • kubernetes-admin@k8s-lab 04:00:30
> kubectl describe pod busypod2 | grep -A20 ^Events
Events:
  Type       Reason      Age    From          Message
  ----       -
Normal      Scheduled   95s    default-scheduler Successfully assigned default/busypod2 to k8s-worker-2
Normal      Pulled      92s    kubelet       Successfully pulled image "busybox" in 2.662020062s
Normal      Pulled      85s    kubelet       Successfully pulled image "busybox" in 954.840576ms
Normal      Pulling     66s (x3 over 94s) kubelet Pulling image "busybox"
Normal      Created     65s (x3 over 92s) kubelet Created container busypod2
Normal      Started     65s (x3 over 92s) kubelet Started container busypod2
Normal      Pulled      65s    kubelet       Successfully pulled image "busybox" in 926.819813ms
Warning     BackOff     59s (x2 over 79s) kubelet Back-off restarting failed container

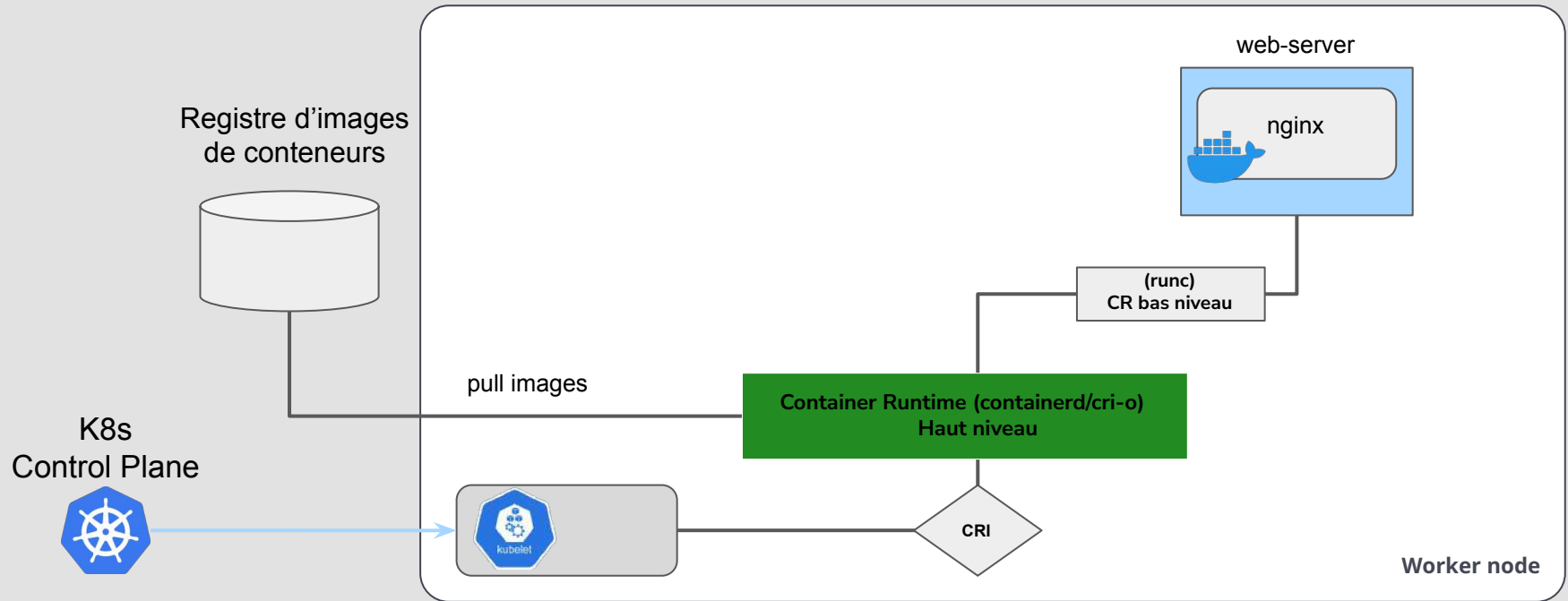
~/kubernetes-labs dev 121 74 base • 04:00:33
>
```

Events du Pod busypod2

# Pod (Séquence de création d'un Pod)



# Pod (mécanisme de création)

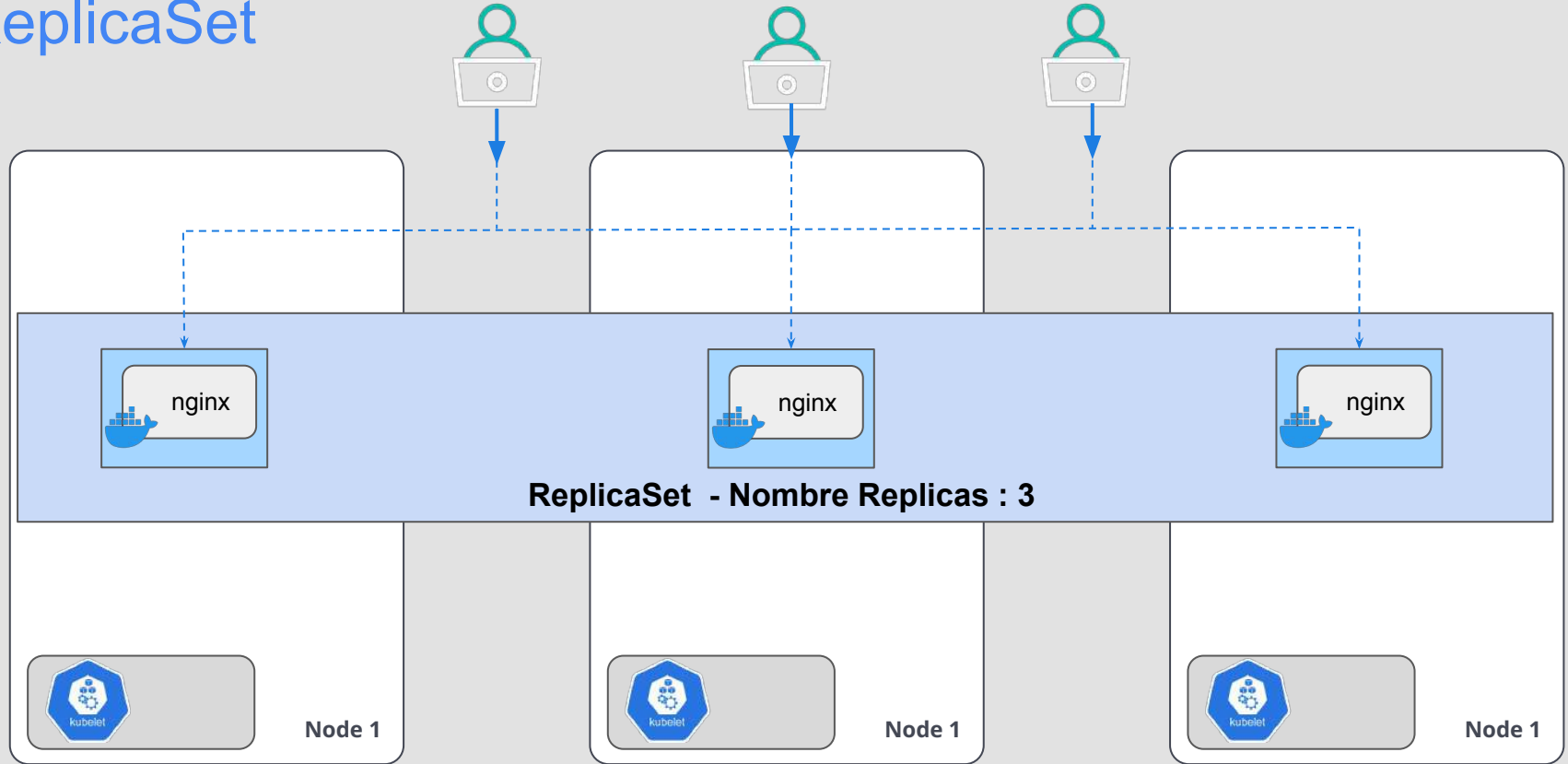


# ReplicaSet

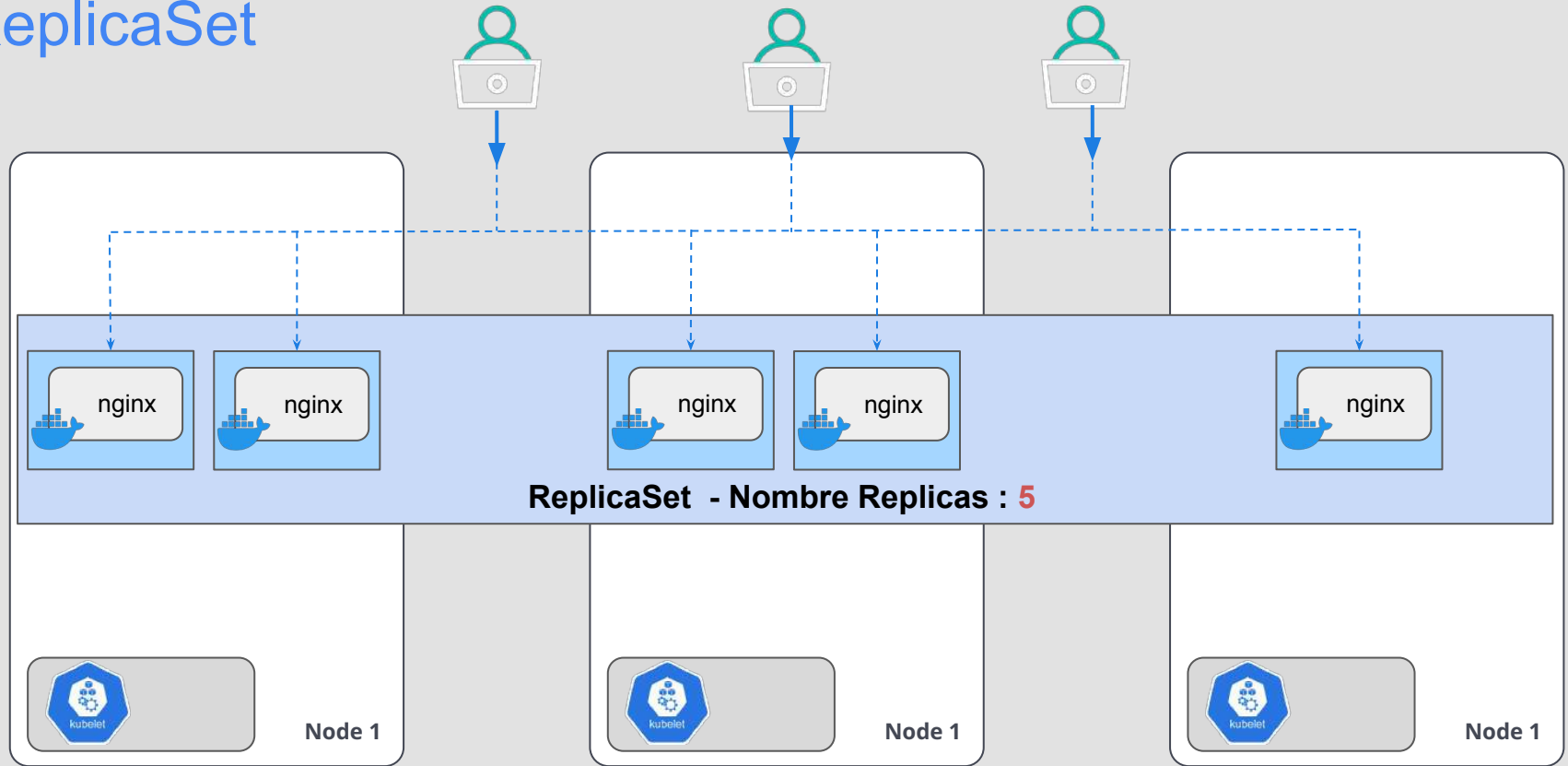
- Maintenir un ensemble stable de réplicas de Pods en fonctionnement à tout moment
  - ◆ **Replicas** : Nombre déterminé de Pods identiques dans le cluster.
  - ◆ **Selector** : spécifie comment identifier les Pods qu'il va répliquer
  - ◆ **Template** : spécifiant les données des nouveaux pods qu'il doit créer
- Recommandé d'utiliser les déploiements au lieu d'utiliser directement les ReplicaSets

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: frontend
  labels:
    app: web-server
    tier: frontend
spec:
  replicas: 6
  selector:
    matchLabels:
      tier: frontend
  template:
    metadata:
      labels:
        tier: frontend
    spec:
      containers:
        - name: nginx
          image: nginx:1.14.2
```

# ReplicaSet



# ReplicaSet



# ReplicaSet

```
~/kubernetes-labs dev !21 75
> kubectl create -f replicaset-frontend.yaml
replicaset.apps/frontend created
```

```
~/kubernetes-labs dev !21 75
> kubectl get replicaset
```

NAME	DESIRED	CURRENT	READY	AGE
frontend	3	3	3	17s

```
~/kubernetes-labs dev !21 75
> kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
frontend-k24km	1/1	Running	0	22s
frontend-p7h9s	1/1	Running	0	22s
frontend-sllmb	1/1	Running	0	22s
web-server	1/1	Running	1 (40m ago)	4d1h

```
~/kubernetes-labs dev !21 75
> kubectl scale --replicas=6 replicaset frontend
replicaset.apps/frontend scaled
```

```
~/kubernetes-labs dev !21 75
> kubectl get replicaset
```

NAME	DESIRED	CURRENT	READY	AGE
frontend	6	6	6	46s

```
~/kubernetes-labs dev !21 75
> kubectl describe replicaset frontend
```

Name: frontend  
Namespace: default  
Selector: tier=frontend  
Labels: app=web-server  
tier=frontend  
Annotations: <none>  
Replicas: 3 current / 3 desired  
Pods Status: 3 Running / 0 Waiting / 0 Succeeded / 0 Failed  
Pod Template:  
Labels: tier=frontend  
Containers:  
nginx:  
Image: nginx:1.14.2  
Port: <none>  
Host Port: <none>  
Environment: <none>  
Mounts: <none>  
Volumes: <none>

```
Events:
```

Type	Reason	Age	From	Message
Normal	SuccessfulCreate	48s	replicaset-controller	Created pod: frontend-k24km
Normal	SuccessfulCreate	48s	replicaset-controller	Created pod: frontend-p7h9s
Normal	SuccessfulCreate	48s	replicaset-controller	Created pod: frontend-sllmb

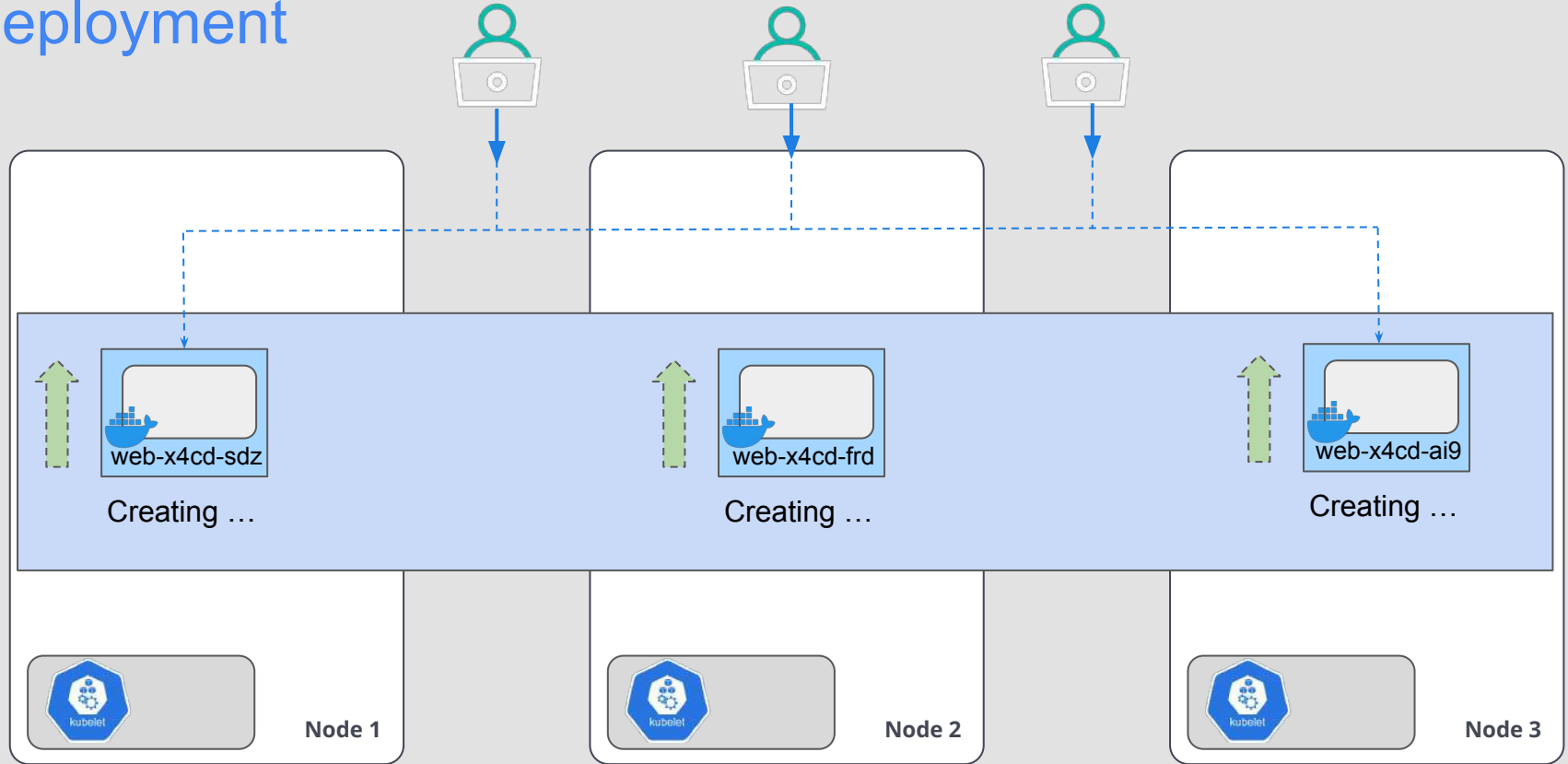
# Deployment

- Permet d'assurer le fonctionnement d'un ensemble de Pods
- Deployment ⇒ ReplicaSet ⇒ Pods
- Fonctionnalités :
  - ◆ Version ⇒ Création d'un nouveau ReplicaSet lors d'une mise à jour de l'application
  - ◆ Rollback ⇒ Retour à une révision antérieure du déploiement
- Différentes stratégies de mise à jour
  - ◆ Recreate
  - ◆ Rolling update
  - ◆ Blue / green
  - ◆ Canary
  - ◆ A/B testing

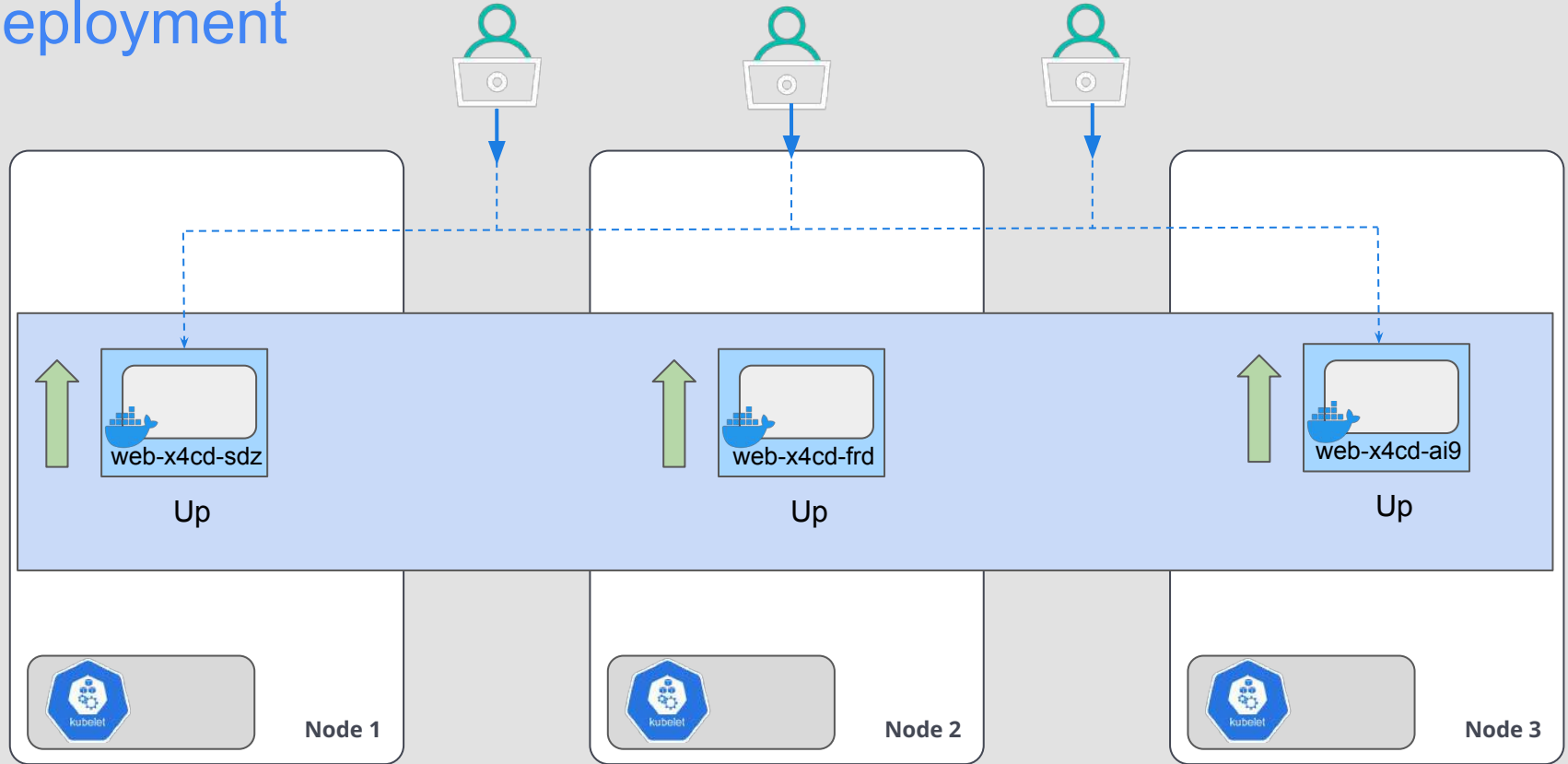
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
  labels:
    app: web-server
    tier: frontend
spec:
  replicas: 3
  selector:
    matchLabels:
      tier: frontend
  template:
    metadata:
      labels:
        tier: frontend
    spec:
      containers:
        - name: nginx
          image: nginx:1.14.2
          ports:
            - containerPort: 80
```



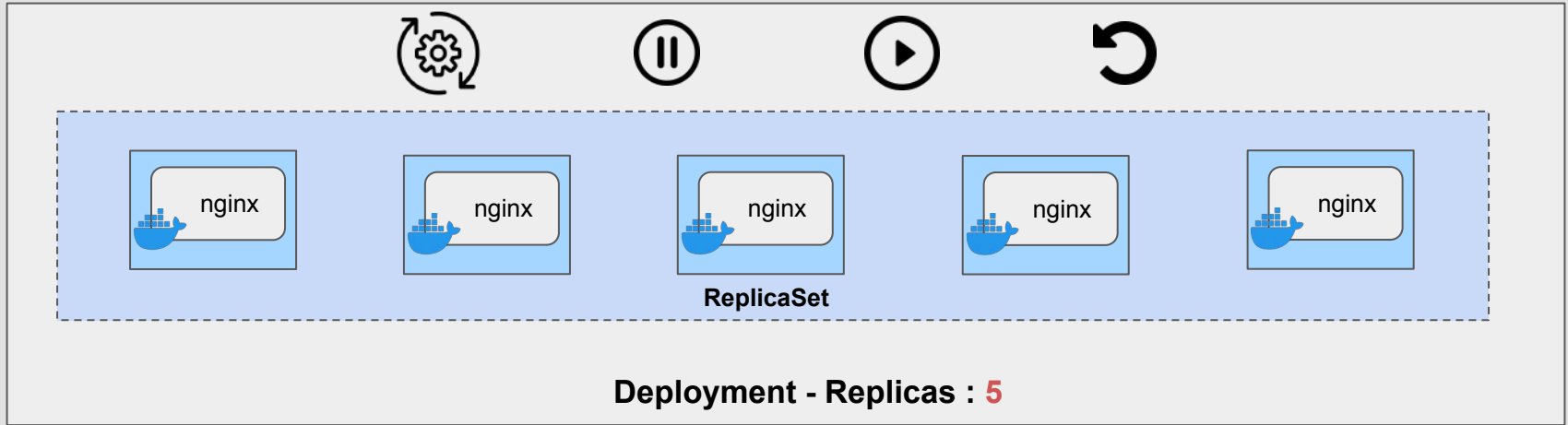
# Deployment



# Deployment



# Deployment



# Deployment

```
~/kubernetes-labs dev 121 76
> kubectl create -f deployment-frontend.yaml
deployment.apps/frontend created
```

```
~/kubernetes-labs dev 121 76
> kubectl get deployments
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
frontend	3/3	3	3	9s

```
~/kubernetes-labs dev 121 76
> kubectl get replicaset
```

NAME	DESIRED	CURRENT	READY	AGE
frontend-5bbbfcf84f	3	3	3	16s

```
~/kubernetes-labs dev 121 76
> kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
frontend-5bbbfcf84f-6fqxv	1/1	Running	0	20s
frontend-5bbbfcf84f-758fr	1/1	Running	0	20s
frontend-5bbbfcf84f-s2qb7	1/1	Running	0	20s

```
~/kubernetes-labs dev 121 76
> kubectl describe deployments frontend
```

Name: frontend  
Namespace: default  
CreationTimestamp: Sun, 30 Oct 2022 03:48:47 +0100  
Labels: app=web-server, tier=frontend  
Annotations: deployment.kubernetes.io/revision: 1

Selector: tier=frontend  
Replicas: 3 desired | 3 updated | 3 total | 3 available | 0 unavailable  
StrategyType: RollingUpdate

MinReadySeconds: 0  
RollingUpdateStrategy: 25% max unavailable, 25% max surge

Pod Template:

Labels: tier=frontend  
Containers:

nginx:  
Image: nginx:1.14.2  
Port: 80/TCP  
Host Port: 0/TCP  
Environment: <none>  
Mounts: <none>  
Volumes: <none>

Conditions:

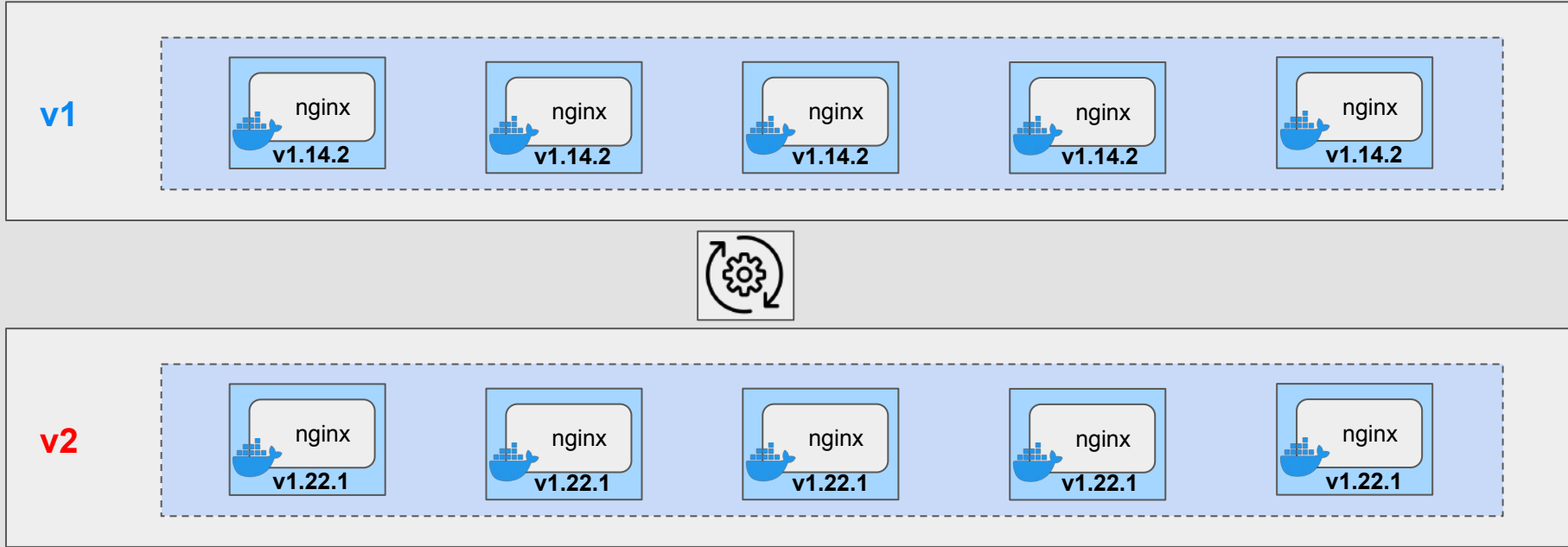
Type	Status	Reason
Available	True	MinimumReplicasAvailable
Progressing	True	NewReplicaSetAvailable

OldReplicaSets: <none>  
NewReplicaSet: frontend-5bbbfcf84f (3/3 replicas created)

Events:

Type	Reason	Age	From	Message
Normal	ScalingReplicaSet	55s	deployment-controller	Scaled up replica set frontend-5bbbfcf84f to 3

# Deployment (Update & Rollback)





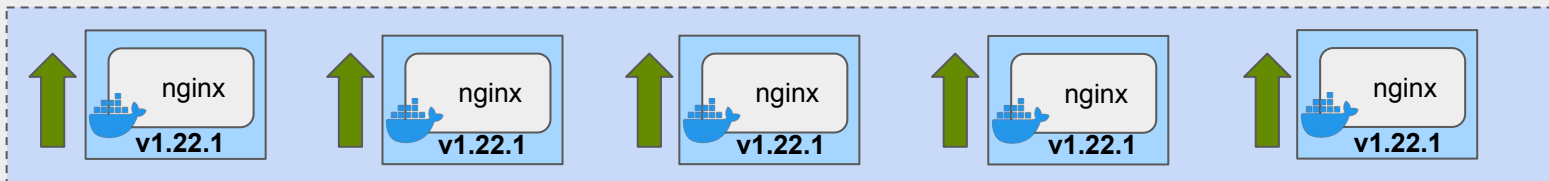
Application  
Down

# Stratégies de Déploiement (Recreate)

v1



v2



# Stratégies de Déploiement

## → Créer

```
$ kubectl create -f deployment-frontend.yaml
```

## → Regarder le status

```
$ kubectl describe deployment frontend
```

## → Mettre à jour

```
$ kubectl apply -f deployment-frontend.yaml  
$ kubectl set image deployment frontend nginx=nginx:1.22.1
```

## → Voir le status & Rollback

```
$ kubectl rollout history deployment frontend  
$ kubectl rollout status deployment frontend  
$ kubectl rollout undo deployment frontend
```



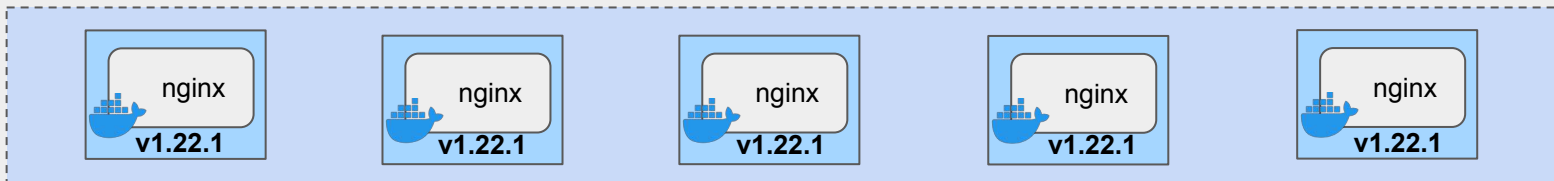
Application  
Down

# Stratégies de Déploiement (Recreate)

v1



v2





# Stratégies de Déploiement (Recreate)

```
> kubectl create -f deployment-frontend.yaml
deployment.apps/frontend created
```

```
> kubectl describe deployments frontend
```

```
Name:          frontend
Namespace:     default
CreationTimestamp: Sun, 30 Oct 2022 04:31:17 +0100
Labels:        app=web-server
               tier=frontend
Annotations:   deployment.kubernetes.io/revision: 1
Selector:      tier=frontend
```

```
Replicas:      3 desired | 3 updated | 3 total | 3 available | 0 unavailable
StrategyType:   Recreate
MinReadySeconds: 0
```

```
Pod Template:
  Labels:  tier=frontend
  Containers:
```

```
  nginx:
    Image:      nginx:1.14.2
    Port:       80/TCP
    Host Port:  0/TCP
    Environment: <none>
    Mounts:      <none>
    Volumes:     <none>
```

Conditions:

Type	Status	Reason
Available	True	MinimumReplicasAvailable
Progressing	True	NewReplicaSetAvailable

OldReplicaSets: <none>

NewReplicaSet: frontend-5bbbfcf84f (3/3 replicas created)

Events:

Type	Reason	Age	From	Message
Normal	ScalingReplicaSet	4s	deployment-controller	Scaled up replica set frontend-5bbbfcf84f to 3

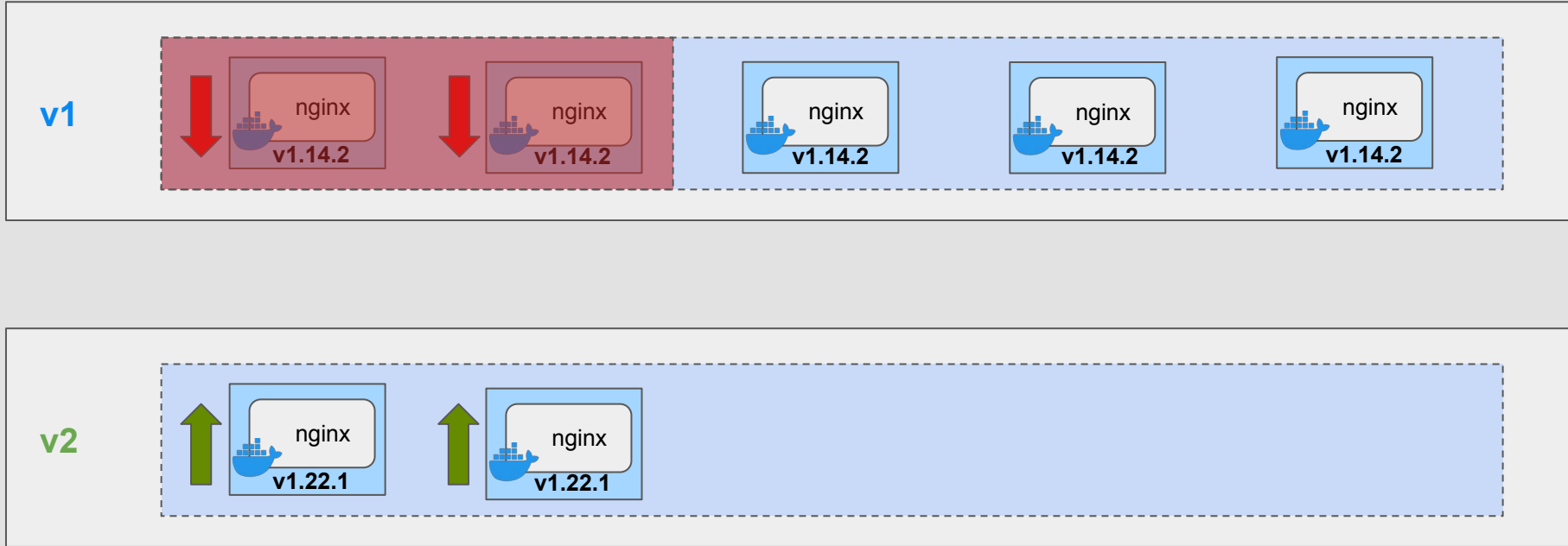
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
  labels:
    app: web-server
    tier: frontend
spec:
  replicas: 3
  strategy:
    type: Recreate
  selector:
    matchLabels:
      tier: frontend
  template:
    metadata:
      labels:
        tier: frontend
    spec:
      containers:
        - name: nginx
          image: nginx:1.14.2
          ports:
            - containerPort: 80
```

# Stratégies de Déploiement (Recreate)

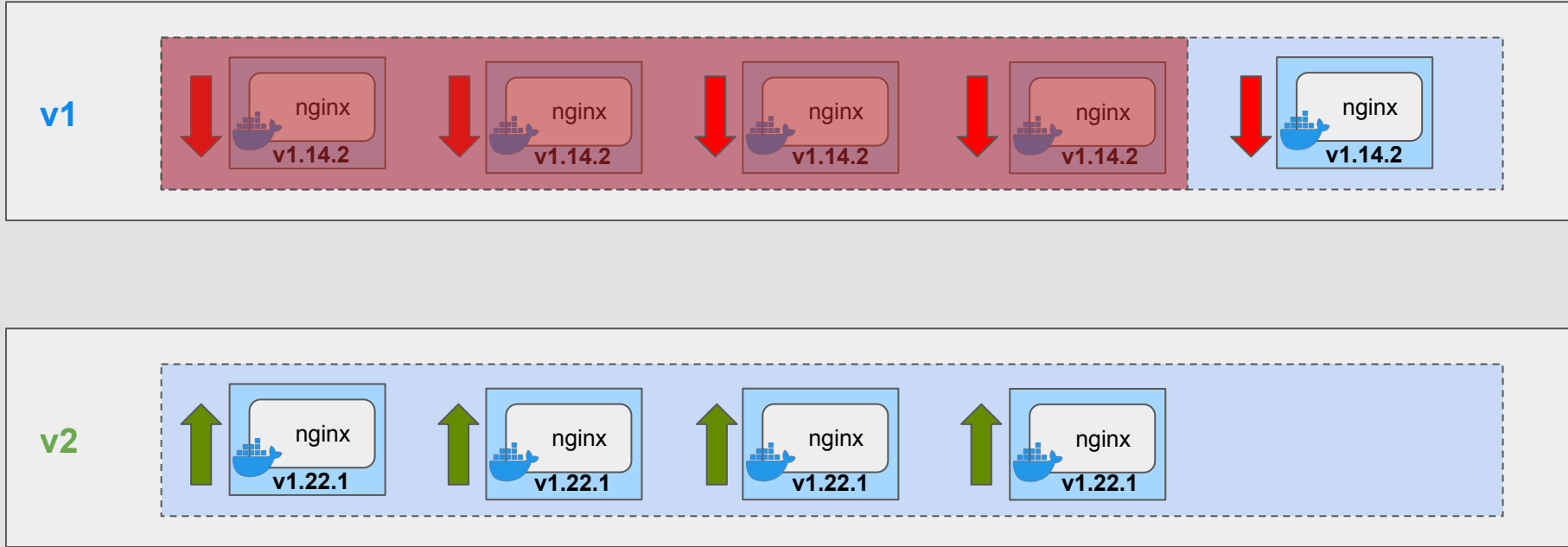
```
~/kubernetes-labs dev 121 ?6  
> kubectl set image deployment frontend nginx=nginx:1.22.1  
deployment.apps/frontend image updated
```

```
> kubectl describe deployments frontend  
Name: frontend  
Namespace: default  
CreationTimestamp: Sun, 30 Oct 2022 04:31:17 +0100  
Labels: app=web-server  
tier=frontend  
Annotations: deployment.kubernetes.io/revision: 2  
Selector: tier=frontend  
Replicas: 3 desired | 3 updated | 3 total | 3 available | 0 unavailable  
StrategyType: Recreate  
MinReadySeconds: 0  
Pod Template:  
  Labels: tier=frontend  
  Containers:  
    nginx:  
      Image: nginx:1.22.1  
      Port: 80/TCP  
      Host Port: 0/TCP  
      Environment: <none>  
      Mounts: <none>  
      Volumes: <none>  
Conditions:  
  Type      Status  Reason  
  ----      -  
  Available  True    MinimumReplicasAvailable  
  Progressing True    NewReplicaSetAvailable  
OldReplicaSets: <none>  
NewReplicaSet: frontend-77db9c8569 (3/3 replicas created)  
Events:  
  Type      Reason      Age   From          Message  
  ----      -  
  Normal    ScalingReplicaSet  4m11s deployment-controller Scaled up replica set frontend-5bbbfcf84f to 3  
  Normal    ScalingReplicaSet  10s   deployment-controller Scaled down replica set frontend-5bbbfcf84f to 0 from 3  
  Normal    ScalingReplicaSet  10s   deployment-controller Scaled up replica set frontend-77db9c8569 to 3
```

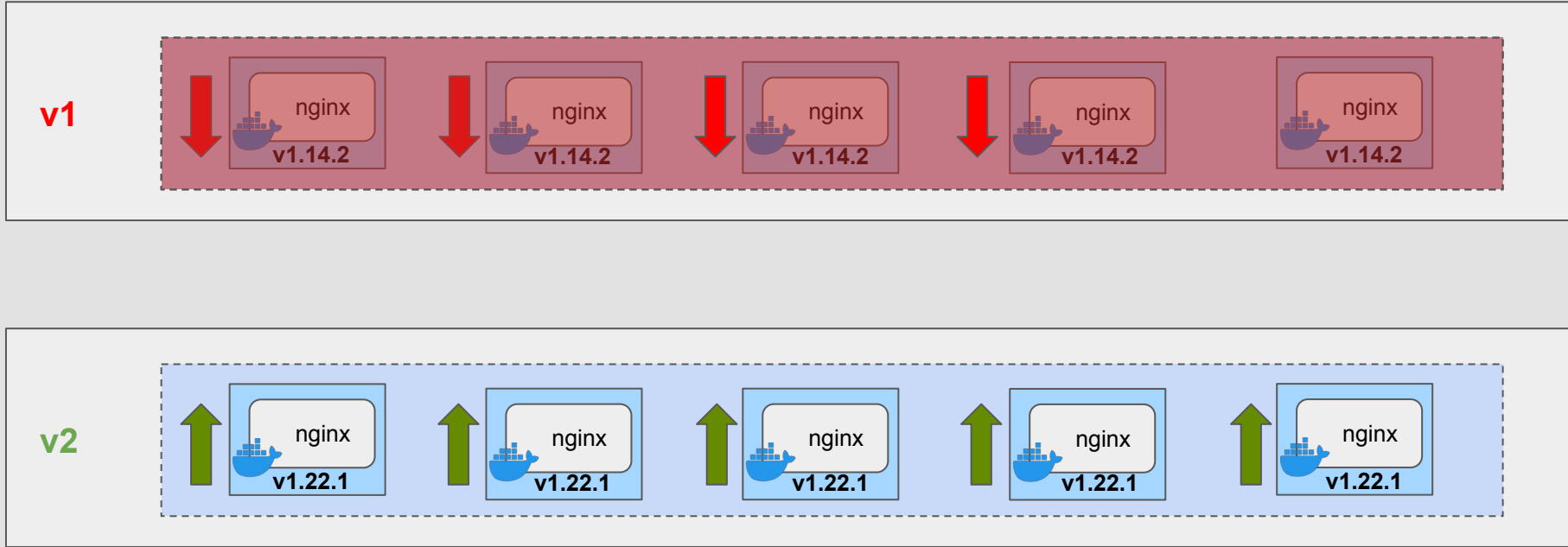
# Stratégies de Déploiement (Rolling Update)



# Stratégies de Déploiement (Rolling Update)



# Stratégies de Déploiement (Rolling Update)

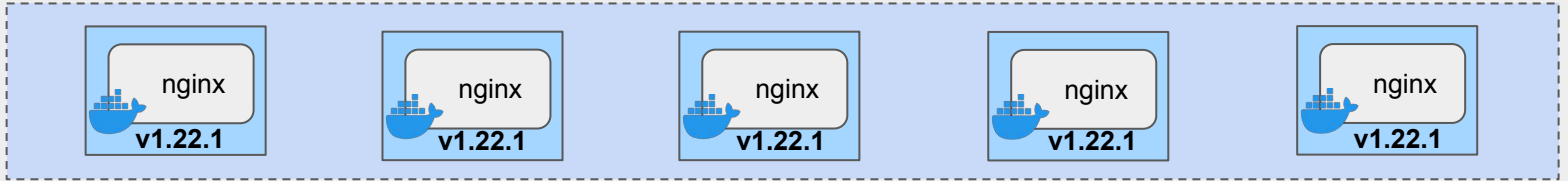


# Stratégies de Déploiement (Rolling Update)

v1



v2



# Stratégies de Déploiement (Rolling Update)

```
~/kubernetes-labs  dev 121 76
> kubectl describe deployments frontend
Name:          frontend
Namespace:     default
CreationTimestamp: Sun, 30 Oct 2022 03:48:47 +0100
Labels:        app=web-server
               tier=frontend
Annotations:   deployment.kubernetes.io/revision: 1
Selector:      tier=frontend
Replicas:      3 desired | 3 updated | 3 total | 3 available | 0 unavailable
StrategyType:  RollingUpdate
MinReadySeconds: 0
RollingUpdateStrategy: 25% max unavailable, 25% max surge
Pod Template:
  Labels:  tier=frontend
  Containers:
    nginx:
      Image:      nginx:1.14.2
      Port:       80/TCP
      Host Port:  80/TCP
      Environment: <none>
      Mounts:      <none>
      Volumes:     <none>
Conditions:
  Type           Status  Reason
  ----           -
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets: <none>
NewReplicaSet:  frontend-5bbbfcf84f (3/3 replicas created)
Events:
  Type           Reason              Age   From                  Message
  ----           -
  Normal         ScalingReplicaSet   55s   deployment-controller Scaled up replica set frontend-5bbbfcf84f to 3
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
  labels:
    app: web-server
    tier: frontend
spec:
  replicas: 3
  # Par défaut strategy = RollingUpdate
  # Peut être spécifié
  selector:
    matchLabels:
      tier: frontend
  template:
    metadata:
      labels:
        tier: frontend
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

# Stratégies de Déploiement (Rolling Update)

~/kubernetes-labs dev !21 76

```
> kubectl create -f deployment-frontend.yaml
deployment.apps/frontend created
```

~/kubernetes-labs dev !21 76

```
> kubectl get rs
```

NAME	DESIRED	CURRENT	READY	AGE
frontend-5bbbfcf84f	3	3	2	1s

~/kubernetes-labs dev !21 76

```
> kubectl set image deployment frontend nginx=nginx:1.22.1
deployment.apps/frontend image updated
```

~/kubernetes-labs dev !21 76

```
> kubectl get rs
```

NAME	DESIRED	CURRENT	READY	AGE
frontend-5bbbfcf84f	2	2	2	16s
frontend-77db9c8569	2	2	1	2s

~/kubernetes-labs dev !21 76

```
> kubectl get rs
```

NAME	DESIRED	CURRENT	READY	AGE
frontend-5bbbfcf84f	2	2	2	17s
frontend-77db9c8569	2	2	1	3s

~/kubernetes-labs dev !21 76

```
> kubectl get rs
```

NAME	DESIRED	CURRENT	READY	AGE
frontend-5bbbfcf84f	0	0	0	19s
frontend-77db9c8569	3	3	3	5s

```
kubectl describe deployments frontend
```

```
name: frontend
namespace: default
creationTimestamp: Sun, 30 Oct 2022 04:50:00 +0100
labels:
  app=web-server
  tier=frontend
annotations:
  deployment.kubernetes.io/revision: 2
selector:
  tier=frontend
replicas: 3 desired | 3 updated | 3 total | 3 available | 0 unavailable
strategyType: RollingUpdate
minReadySeconds: 0
rollingUpdateStrategy: 25% max unavailable, 25% max surge
```

```
Pod Template:
```

```
Labels: tier=frontend
```

```
Containers:
```

```
nginx:
  Image:      nginx:1.22.1
  Port:      80/TCP
  Host Port: 0/TCP
  Environment: <none>
  Mounts:      <none>
  Volumes:      <none>
```

```
Conditions:
```

Type	Status	Reason
Available	True	MinimumReplicasAvailable
Progressing	True	NewReplicaSetAvailable

```
OldReplicaSets: <none>
```

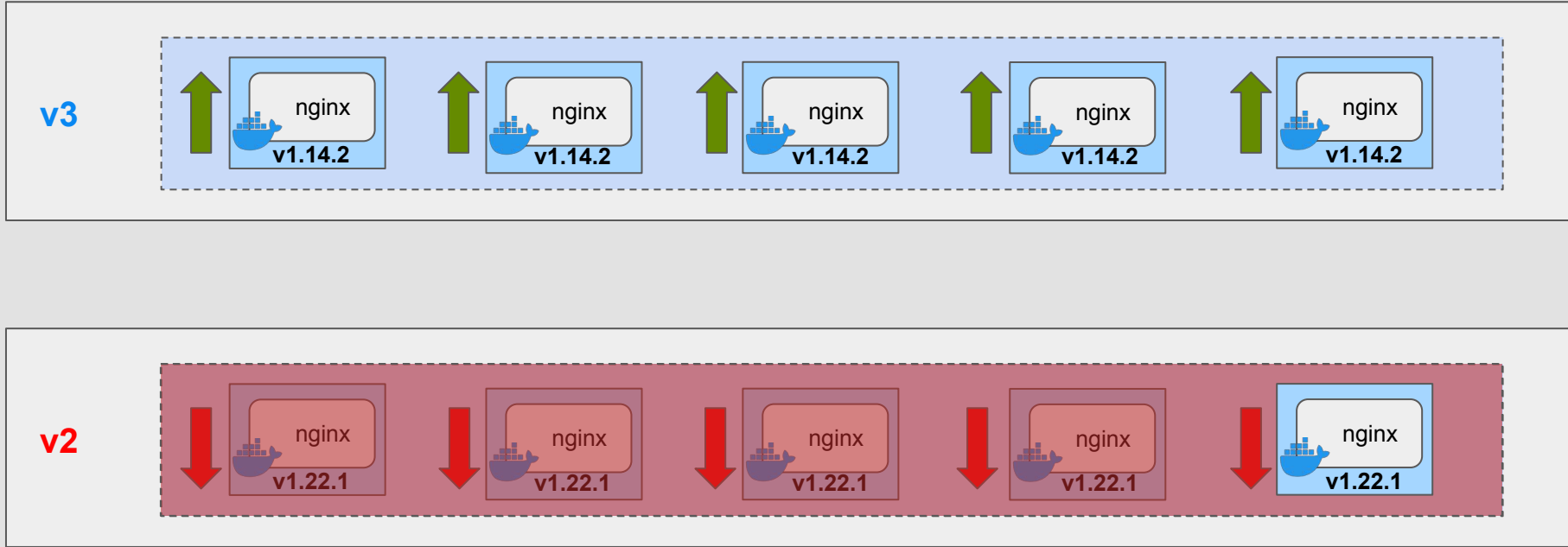
```
NewReplicaSet: frontend-77db9c8569 (3/3 replicas created)
```

```
Events:
```

Type	Reason	Age	From	Message
Normal	ScalingReplicaSet	55s	deployment-controller	Scaled up replica set frontend-5bbbfcf84f to 3
Normal	ScalingReplicaSet	41s	deployment-controller	Scaled up replica set frontend-77db9c8569 to 1
Normal	ScalingReplicaSet	39s	deployment-controller	Scaled down replica set frontend-5bbbfcf84f to 2 from 3
Normal	ScalingReplicaSet	39s	deployment-controller	Scaled up replica set frontend-77db9c8569 to 2 from 1
Normal	ScalingReplicaSet	38s	deployment-controller	Scaled down replica set frontend-5bbbfcf84f to 1 from 2
Normal	ScalingReplicaSet	38s	deployment-controller	Scaled up replica set frontend-77db9c8569 to 3 from 2
Normal	ScalingReplicaSet	36s	deployment-controller	Scaled down replica set frontend-5bbbfcf84f to 0 from 1



# Stratégies de Déploiement (RollBack)



# Stratégies de Déploiement (RollBack)

```
~/kubernetes-labs dev !21 76
> kubectl rollout history deployment frontend
deployment.apps/frontend
REVISION  CHANGE-CAUSE
1          <none>
2          <none>
```

```
~/kubernetes-labs dev !21 76
> kubectl rollout undo deployment frontend
deployment.apps/frontend rolled back
```

```
~/kubernetes-labs dev !21 76
> kubectl get rs
```

NAME	DESIRED	CURRENT	READY	AGE
frontend-5bbbfcf84f	3	3	3	8m10s
frontend-77db9c8569	0	0	0	7m56s

```
~/kubernetes-labs dev !21 76
> kubectl rollout history deployment frontend
deployment.apps/frontend
REVISION  CHANGE-CAUSE
2          <none>
3          <none>
```

```
> kubectl describe deployments frontend
Name:          frontend
Namespace:     default
CreationTimestamp:  Sun, 30 Oct 2022 04:50:00 +0100
Labels:        app=web-server
               tier=frontend
Annotations:   deployment.kubernetes.io/revision: 3
Selector:      tier=frontend
Replicas:      3 desired | 3 updated | 3 total | 3 available | 0 unavailable
StrategyType:   RollingUpdate
MinReadySeconds: 0
RollingUpdateStrategy: 25% max unavailable, 25% max surge
Pod Template:
  Labels:  tier=frontend
  Containers:
    nginx:
      Image:      nginx:1.14.2
      Port:       80/TCP
      Host Port:  0/TCP
      Environment: <none>
      Mounts:      <none>
      Volumes:     <none>
  Conditions:
    Type           Status  Reason
    ----           -
    Available       True    MinimumReplicasAvailable
    Progressing     True    NewReplicaSetAvailable
  OldReplicaSets: <none>
  NewReplicaSet:  frontend-5bbbfcf84f (3/3 replicas created)
Events:
```

Type	Reason	Age	From	Message
Normal	ScalingReplicaSet	12m	deployment-controller	Scaled up replica set frontend-5bbbfcf84f to 3
Normal	ScalingReplicaSet	12m	deployment-controller	Scaled up replica set frontend-77db9c8569 to 1
Normal	ScalingReplicaSet	12m	deployment-controller	Scaled down replica set frontend-5bbbfcf84f to 2 from 3
Normal	ScalingReplicaSet	12m	deployment-controller	Scaled up replica set frontend-77db9c8569 to 2 from 1
Normal	ScalingReplicaSet	12m	deployment-controller	Scaled down replica set frontend-5bbbfcf84f to 1 from 2
Normal	ScalingReplicaSet	12m	deployment-controller	Scaled up replica set frontend-77db9c8569 to 3 from 2
Normal	ScalingReplicaSet	12m	deployment-controller	Scaled down replica set frontend-5bbbfcf84f to 0 from 1
Normal	ScalingReplicaSet	4m44s	deployment-controller	Scaled up replica set frontend-5bbbfcf84f to 1 from 0
Normal	ScalingReplicaSet	4m43s	deployment-controller	Scaled down replica set frontend-77db9c8569 to 2 from 3
Normal	ScalingReplicaSet	4m41s (x4 over 4m43s)	deployment-controller	(combined from similar events): Scaled down replica set frontend-77db9c8569 to 0 from 1

# DaemonSet

- Assure que tous les noeuds exécutent une copie du pod
- Ne connaît pas la notion de replicas.
- Utilisé pour des besoins particuliers comme :
  - ◆ L'exécution d'agents de collection de logs
    - Ex: filebeat, metricbeat, fluentd...
  - ◆ L'exécution de pilotes pour du matériel
    - Ex: nvidia-plugin
  - ◆ L'exécution d'agents de supervision
    - Ex: Prometheus node exporter

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: filebeat
  labels:
    agent: filebeat
spec:
  selector:
    matchLabels:
      agent: filebeat
  template:
    metadata:
      labels:
        agent: filebeat
    spec:
      containers:
        - name: filebeat
          image: docker.elastic.co/beats/filebeat:8.4.3
```

# DaemonSet

```
~/kubernetes-labs dev !21 77
> kubectl create -f daemonset-filebeat.yaml
daemonset.apps/filebeat created

~/kubernetes-labs dev !21 77
> kubectl get daemonsets
```

NAME	DESIRED	CURRENT	READY	UP-TO-DATE	AVAILABLE	NODE SELECTOR	AGE
filebeat	3	3	3	3	3	<none>	10s

```
~/kubernetes-labs dev !21 77
> kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
k8s-master-1	Ready	control-plane	9d	v1.25.3
k8s-worker-1	Ready	<none>	9d	v1.25.3
k8s-worker-2	Ready	<none>	9d	v1.25.3
k8s-worker-3	Ready	<none>	9d	v1.25.3

```
~/kubernetes-labs dev !21 77
> kubectl get pods -o wide
```

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED NODE
filebeat-b2bmn	1/1	Running	0	26s	192.168.114.158	k8s-worker-2	<none>
filebeat-srtj9	1/1	Running	0	26s	192.168.117.221	k8s-worker-3	<none>
filebeat-x8fcl	1/1	Running	0	26s	192.168.118.27	k8s-worker-1	<none>

```
~/kubernetes-labs dev !21 77
> kubectl describe daemonsets filebeat
```

Name: filebeat

Selector: agent=filebeat

Node-Selector: <none>

Labels: agent=filebeat

Annotations: deprecated.daemonset.template.generation: 1

Desired Number of Nodes Scheduled: 3

Current Number of Nodes Scheduled: 3

Number of Nodes Scheduled with Up-to-date Pods: 3

Number of Nodes Scheduled with Available Pods: 3

Number of Nodes Missscheduled: 0

Pods Status: 3 Running / 0 Waiting / 0 Succeeded / 0 Failed

Pod Template:

Labels: agent=filebeat

Containers:

filebeat:

Image: docker.elastic.co/beats/filebeat:8.4.3

Port: <none>

Host Port: <none>

Environment: <none>

Mounts: <none>

Volumes: <none>

Events:

Type	Reason	Age	From	Message
Normal	SuccessfulCreate	2m28s	daemonset-controller	Created pod: filebeat-b2bmn
Normal	SuccessfulCreate	2m28s	daemonset-controller	Created pod: filebeat-x8fcl
Normal	SuccessfulCreate	2m28s	daemonset-controller	Created pod: filebeat-srtj9

# DaemonSet

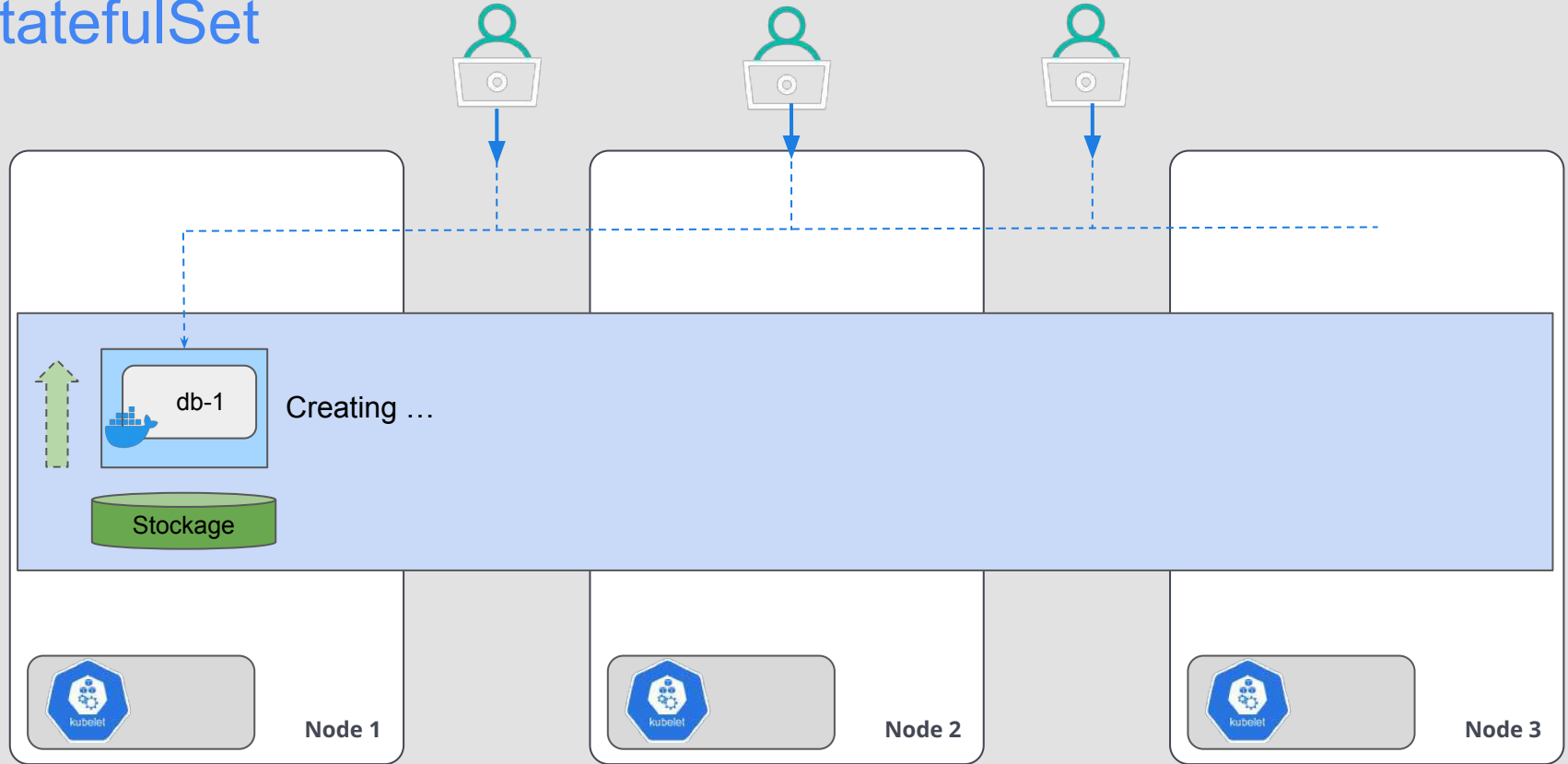


# StatefulSet

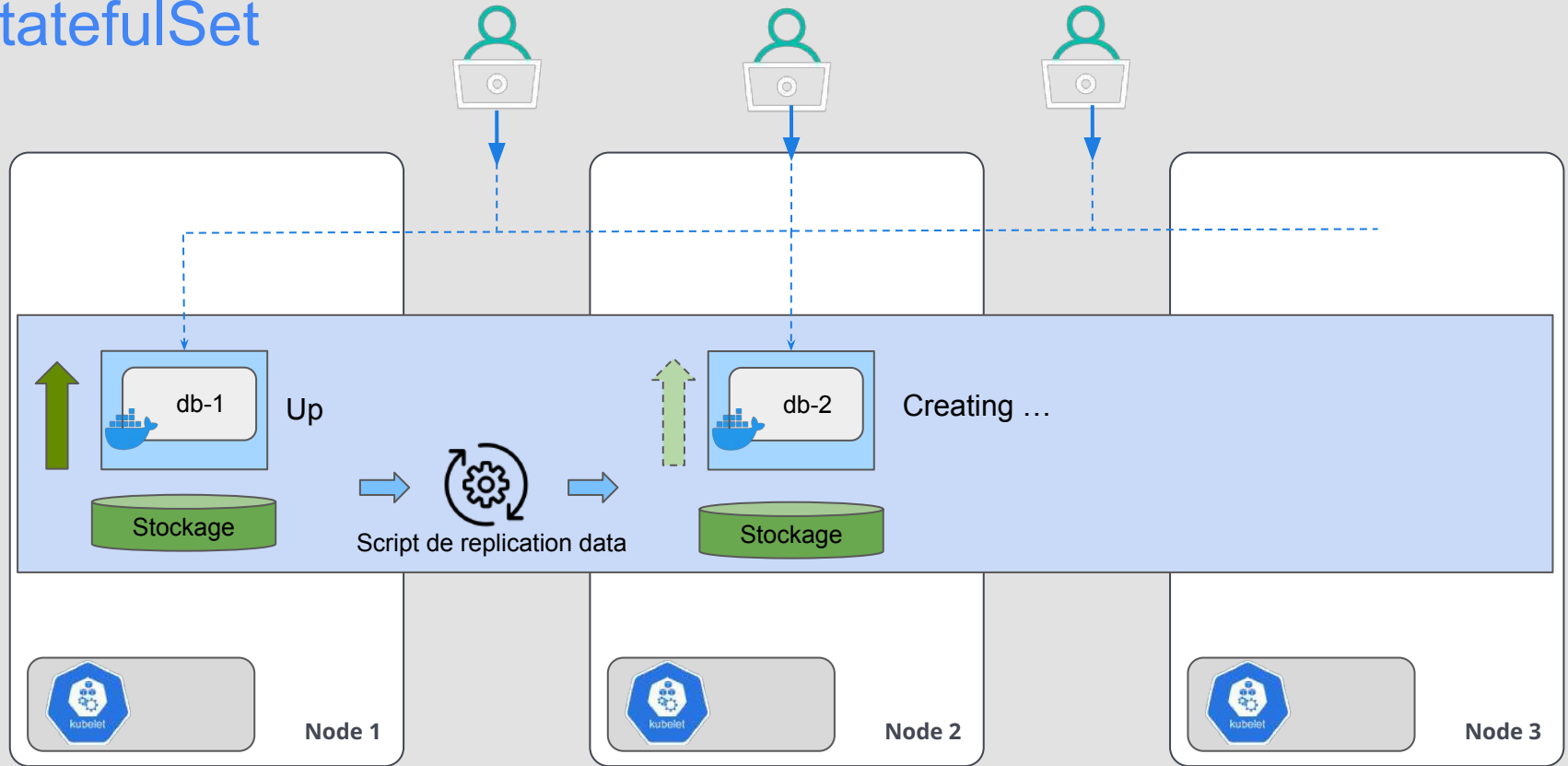
- Similaire au Deployment
  - ◆ Deployment ⇒ Conçus pour les App Stateless
  - ◆ StatefulSet ⇒ Pour les application à état (Stateful)
    - Ex: DB, Kafka, toute app qui stocke la donnée
- Les pods possèdent des identifiants uniques.
- Chaque replica de pod est créé par ordre d'index
- En général, nécessite un Persistent Volume et un Storage Class.
- Supprimer un StatefulSet ne supprime pas le PV associé

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: stateful-web
spec:
  selector:
    matchLabels:
      app: nginx
  serviceName: "nginx"
  replicas: 3 # by default is 1
  minReadySeconds: 10 # by default is 0
  template:
    metadata:
      labels:
        app: nginx
    spec:
      terminationGracePeriodSeconds: 10
      containers:
        - name: nginx
          image: registry.k8s.io/nginx-slim:0.8
          ports:
            - containerPort: 80
              name: web
          volumeMounts:
            - name: www
              mountPath: /usr/share/nginx/html
      volumeClaimTemplates:
        (...)
```

# StatefulSet

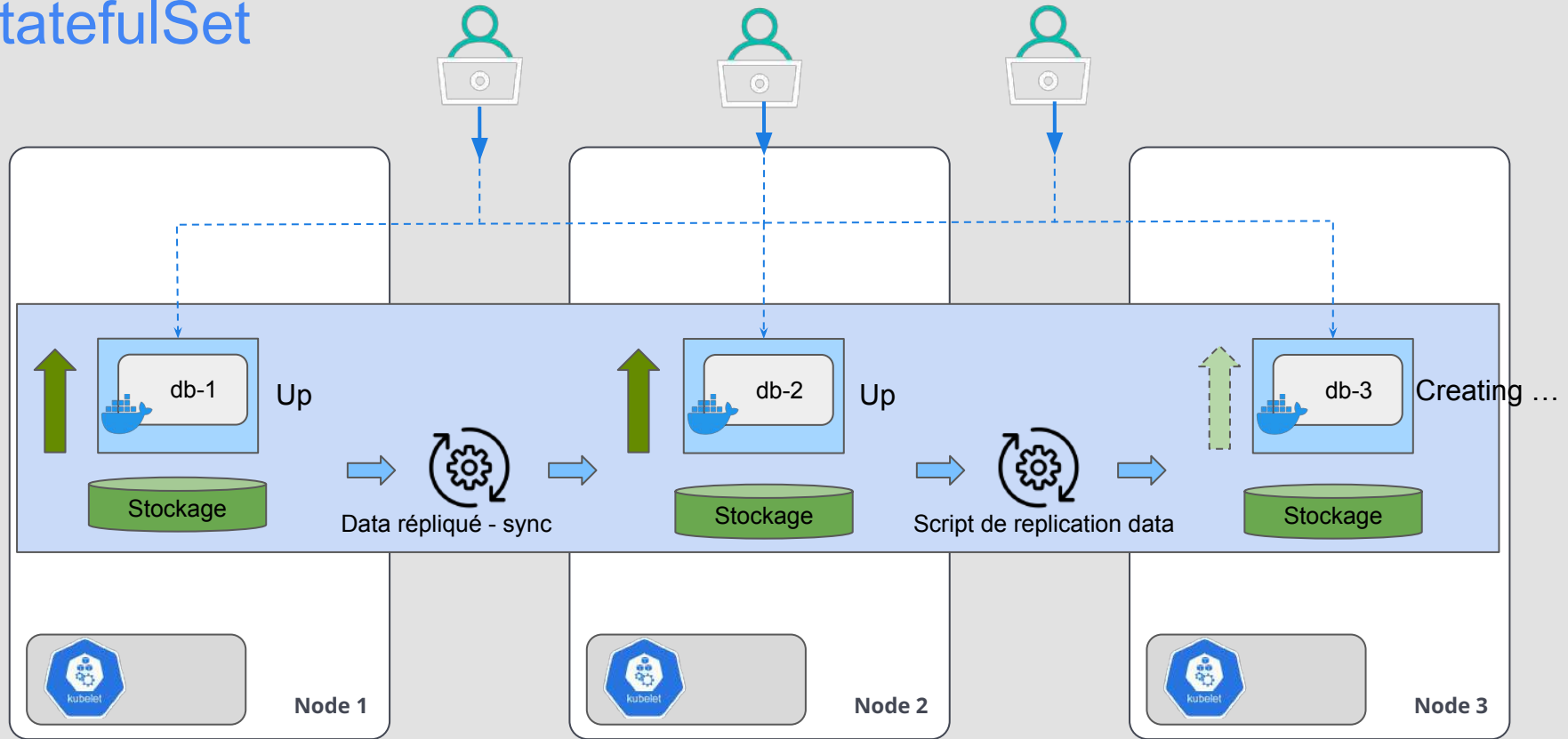


# StatefulSet

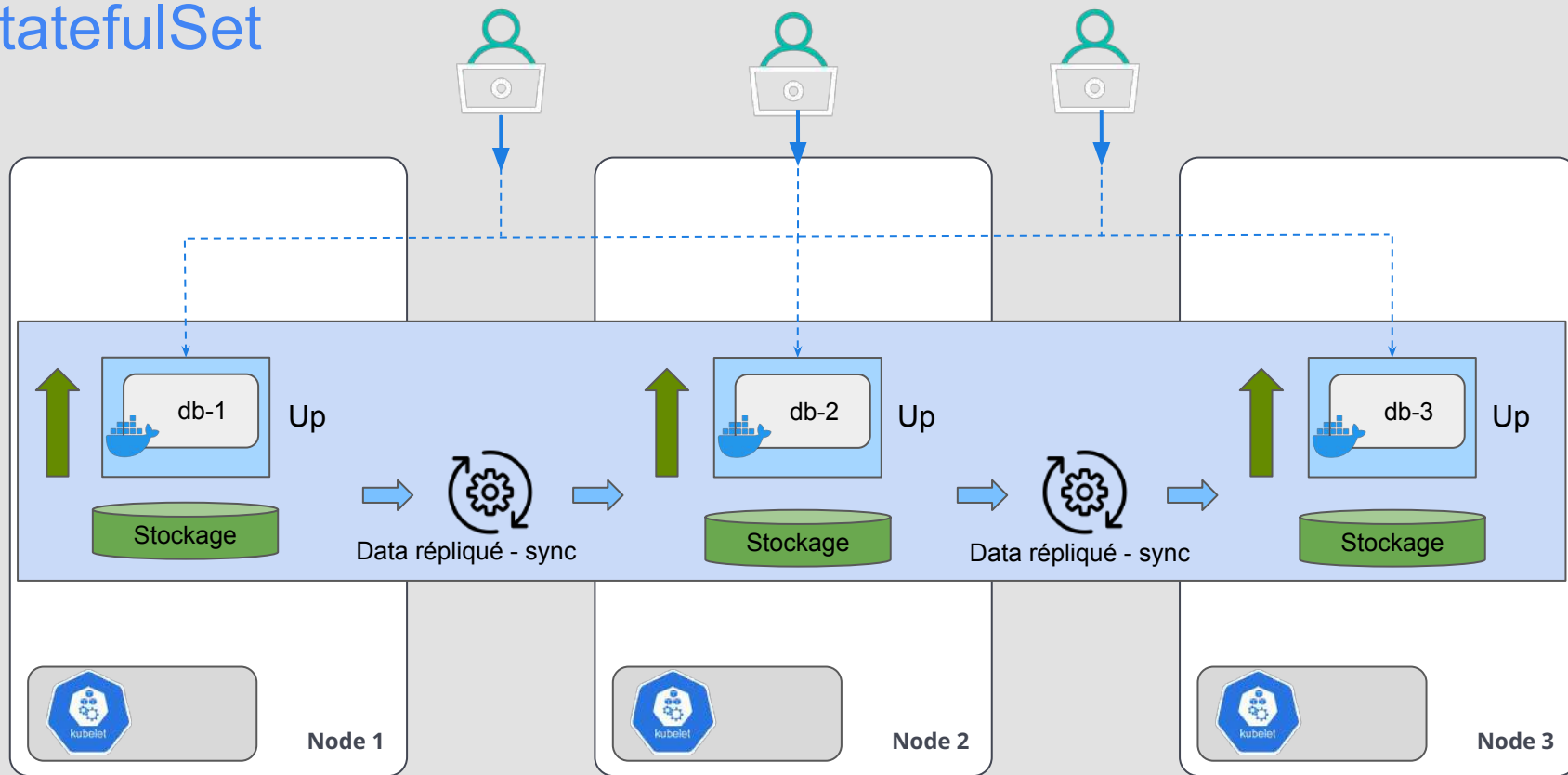




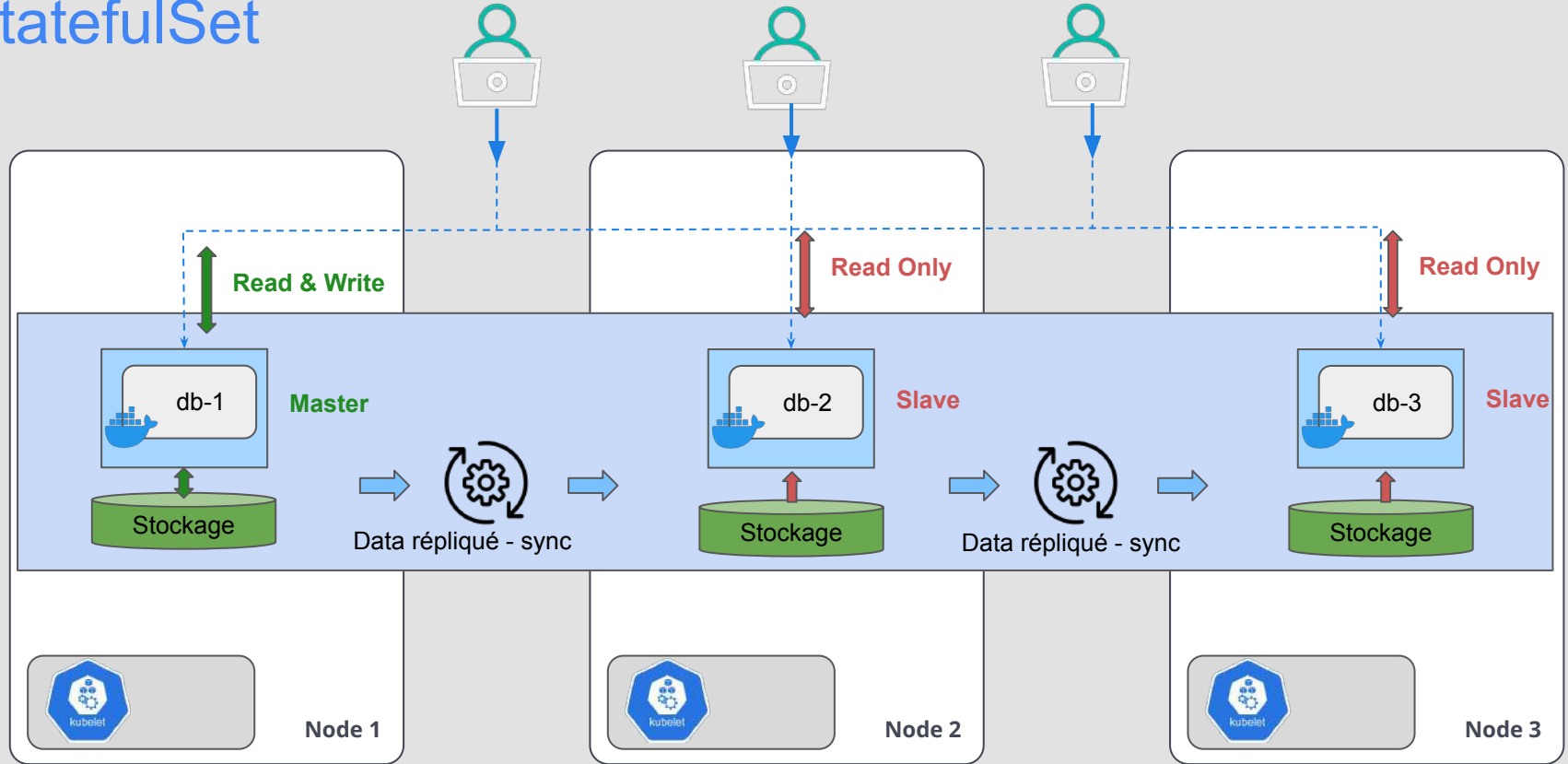
# StatefulSet



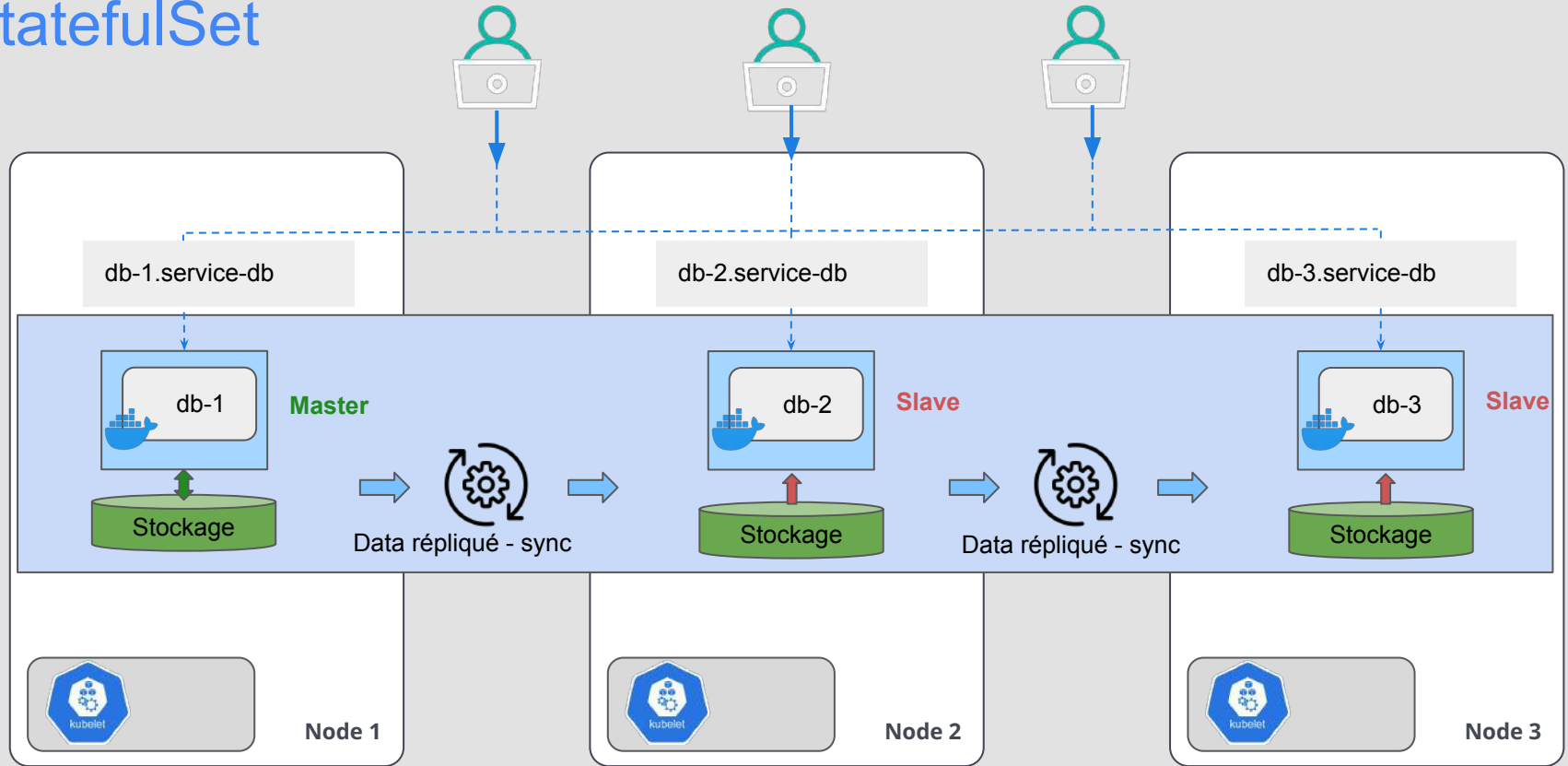
# StatefulSet



# StatefulSet



# StatefulSet



# StatefulSet

```
~/kubernetes-labs  P dev !21 711
> kubectl create -f stateful-webapp.yaml
statefulset.apps/stateful-web created
```

```
~/kubernetes-labs  P dev !21 711
> kubectl get statefulsets
NAME          READY   AGE
stateful-web  1/3     8s
```

```
~/kubernetes-labs  P dev !21 711
> kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
stateful-web-0 1/1     Running   0           14s
```

```
~/kubernetes-labs  P dev !21 711
> kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
stateful-web-0 1/1     Running   0           40s
stateful-web-1 1/1     Running   0           20s
```

```
~/kubernetes-labs  P dev !21 711
> kubectl get pods
NAME          READY   STATUS        RESTARTS   AGE
stateful-web-0 1/1     Running       0           40s
stateful-web-1 1/1     Running       0           20s
stateful-web-2 0/1     ContainerCreating 0           0s
```

```
~/kubernetes-labs  P dev !21 711
> kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
stateful-web-0 1/1     Running   0           43s
stateful-web-1 1/1     Running   0           23s
stateful-web-2 1/1     Running   0           3s
```

```
> kubectl describe statefulsets stateful-web
```

```
Name:          stateful-web
Namespace:     default
CreationTimestamp: Sun, 30 Oct 2022 06:54:22 +0100
Selector:      app=nginx
Labels:        <none>
Annotations:   <none>
Replicas:      3 desired | 3 total
Update Strategy: RollingUpdate
  Partition:    0
Pods Status:   3 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
```

```
  Labels: app=nginx
```

```
  Containers:
```

```
    nginx:
```

```
      Image:      registry.k8s.io/nginx-slim:0.8
```

```
      Port:       80/TCP
```

```
      Host Port:  0/TCP
```

```
      Environment: <none>
```

```
      Mounts:
```

```
        /usr/share/nginx/html from www (rw)
```

```
  Volumes:
```

```
    www:
```

```
      Type:      PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same namespace)
```

```
      ClaimName: my-claim
```

```
      ReadOnly:  false
```

```
  Volume Claims: <none>
```

```
Events:
```

Type	Reason	Age	From	Message
Normal	SuccessfulCreate	4m5s	statefulset-controller	create Pod stateful-web-0 in StatefulSet stateful-web successful
Normal	SuccessfulCreate	3m45s	statefulset-controller	create Pod stateful-web-1 in StatefulSet stateful-web successful
Normal	SuccessfulCreate	3m25s	statefulset-controller	create Pod stateful-web-2 in StatefulSet stateful-web successful

# Jobs

- Crée des pods et s'assurent qu'un certain nombre d'entre eux se terminent avec succès.
- Peut exécuter plusieurs pods en parallèle Si un node du cluster est en panne, les pods sont reschedulés vers un autre node.
- Cas d'usage :
  - ◆ Jobs de maintenance
  - ◆ Jobs de sauvegarde

```
apiVersion: batch/v1
kind: Job
metadata:
  name: pi
spec:
  template:
    spec:
      containers:
      - name: pi
        image: perl:5.34.0
        command: ["perl", "-Mbignum=bpi",
"-wle", "print bpi(2000)"]
        restartPolicy: Never
      backoffLimit: 4
```

# Cronjob

- Un CronJob permet de lancer des Jobs de manière planifiée.
- la programmation des Jobs se définit au format **Cron**
- le champ jobTemplate contient la définition de l'application à lancer comme Job.
- Cas d'usage :
  - ◆ Jobs régulier, planifié de maintenance
  - ◆ Jobs régulier, planifié de sauvegarde, de health check, de synchronisation, export/import de données...

```
apiVersion: batch/v1
kind: CronJob
metadata:
  name: hello
spec:
  schedule: "*/10 * * * *"
  jobTemplate:
    spec:
      template:
        spec:
          containers:
            - name: hello
              image: busybox:1.28
              imagePullPolicy: IfNotPresent
              command:
                - /bin/sh
                - -c
                - date; echo Hello from the
                  Kubernetes cluster
              restartPolicy: OnFailure
```

# Ressources Services & Networking K8s

- Service (CLusterIP, NodePort, LB)
- DNS
- Réseau en environnement K8s
- Ingress
- Ingress Controllers
- Network Policies



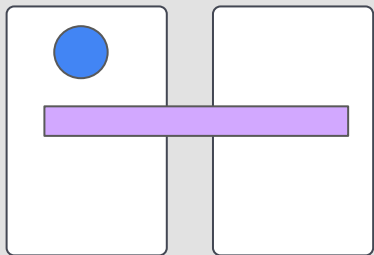
# Service

- Abstraction des Pods sous forme d'une IP virtuelle de Service ou nom DNS
- Rendre un ensemble de Pods accessibles depuis l'extérieur ou l'intérieur du cluster
- Load Balancing entre les Pods d'un même Service
- Sélection des Pods faisant parti d'un Service grâce aux labels
- Adresse IP persistante vs IP Pod changeant

# Types de Service

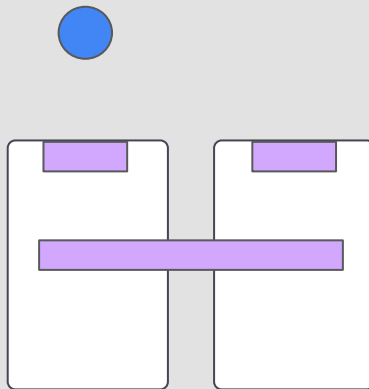
## ClusterIP

Exposition à l'intérieur du cluster



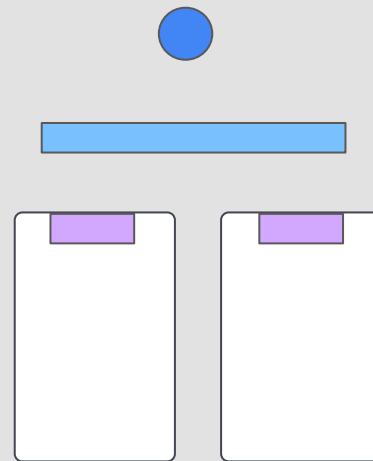
## NodePort

Exposition vers l'extérieur : chaque noeud ouvre un port statique et redirige le trafic vers le port indiqué

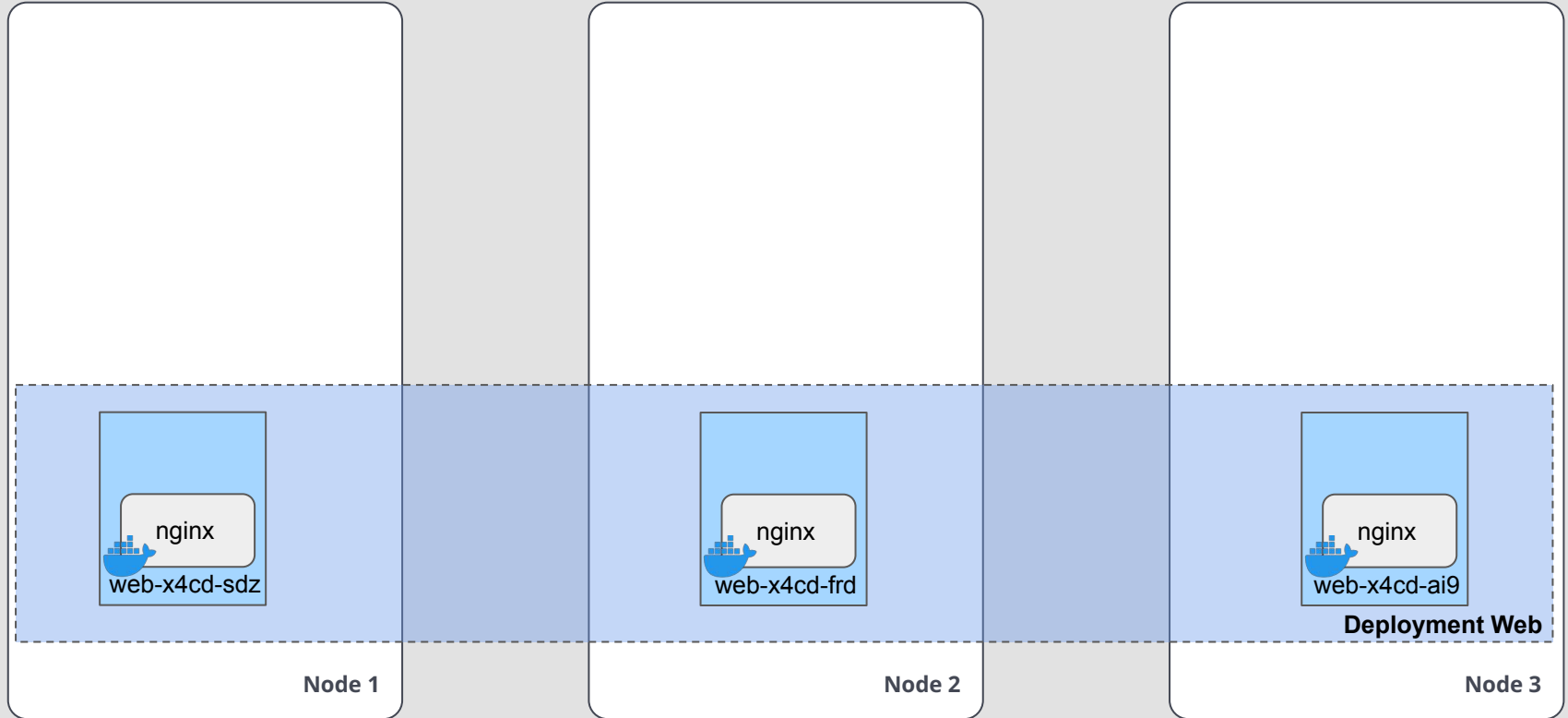


## LoadBalancer

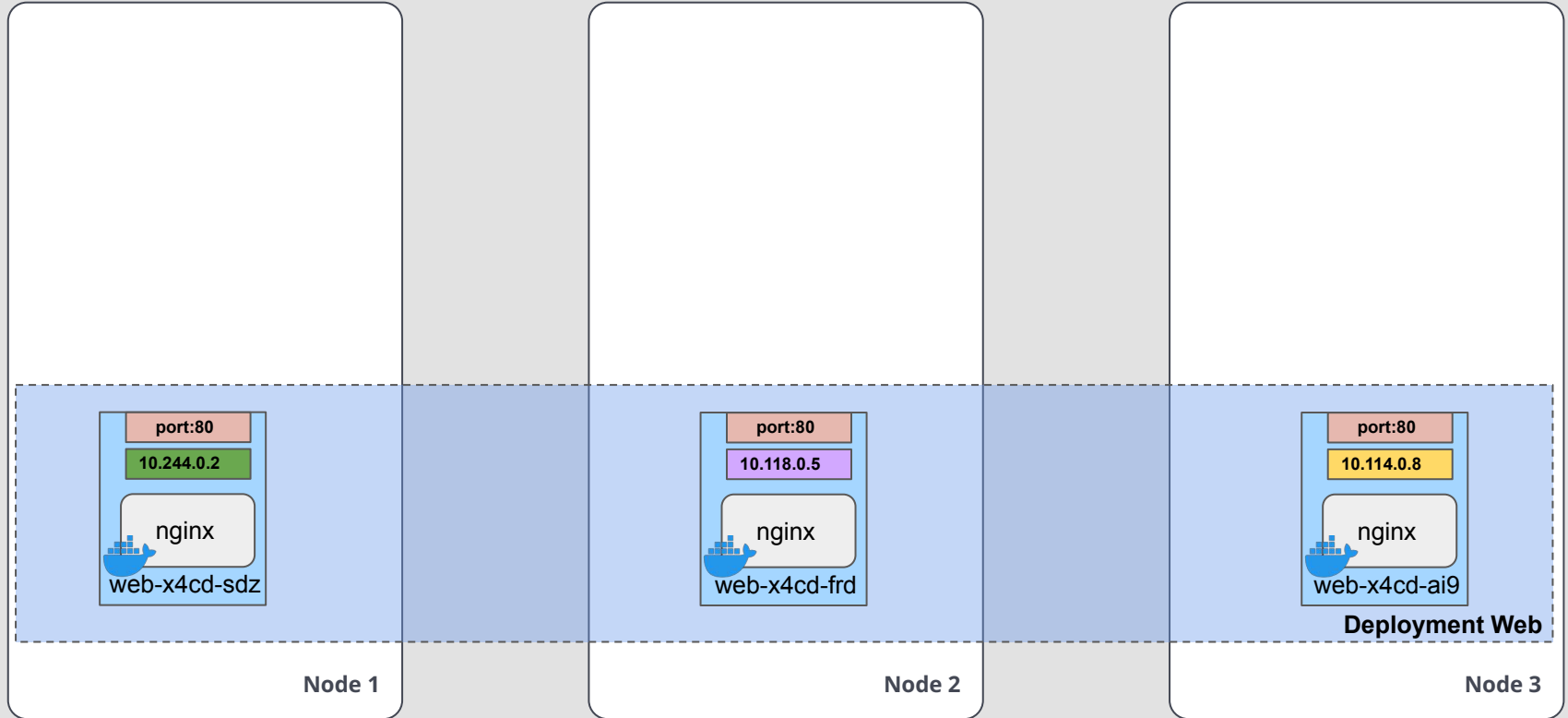
Exposition vers l'extérieur du cluster : en utilisant le LoadBalancer d'un cloud provider



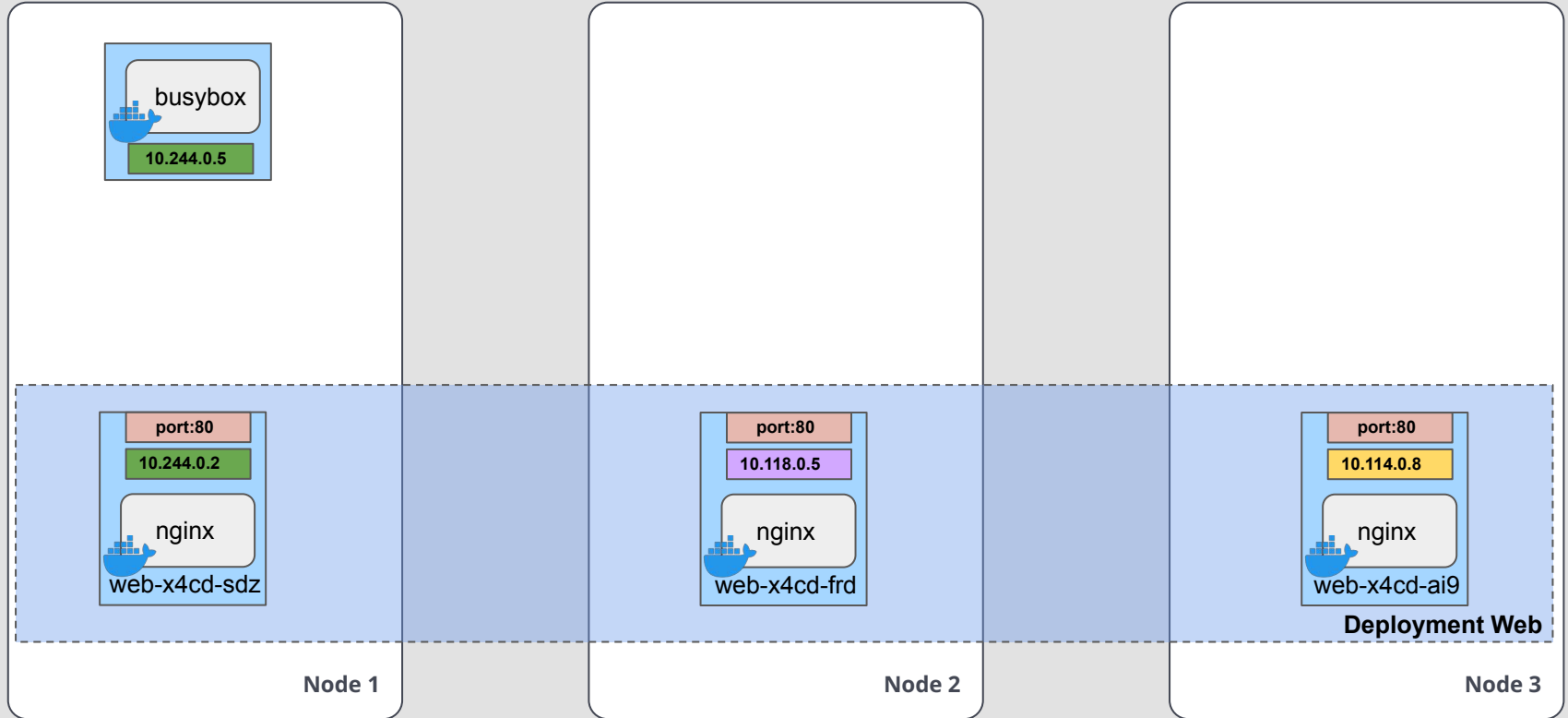
# Service



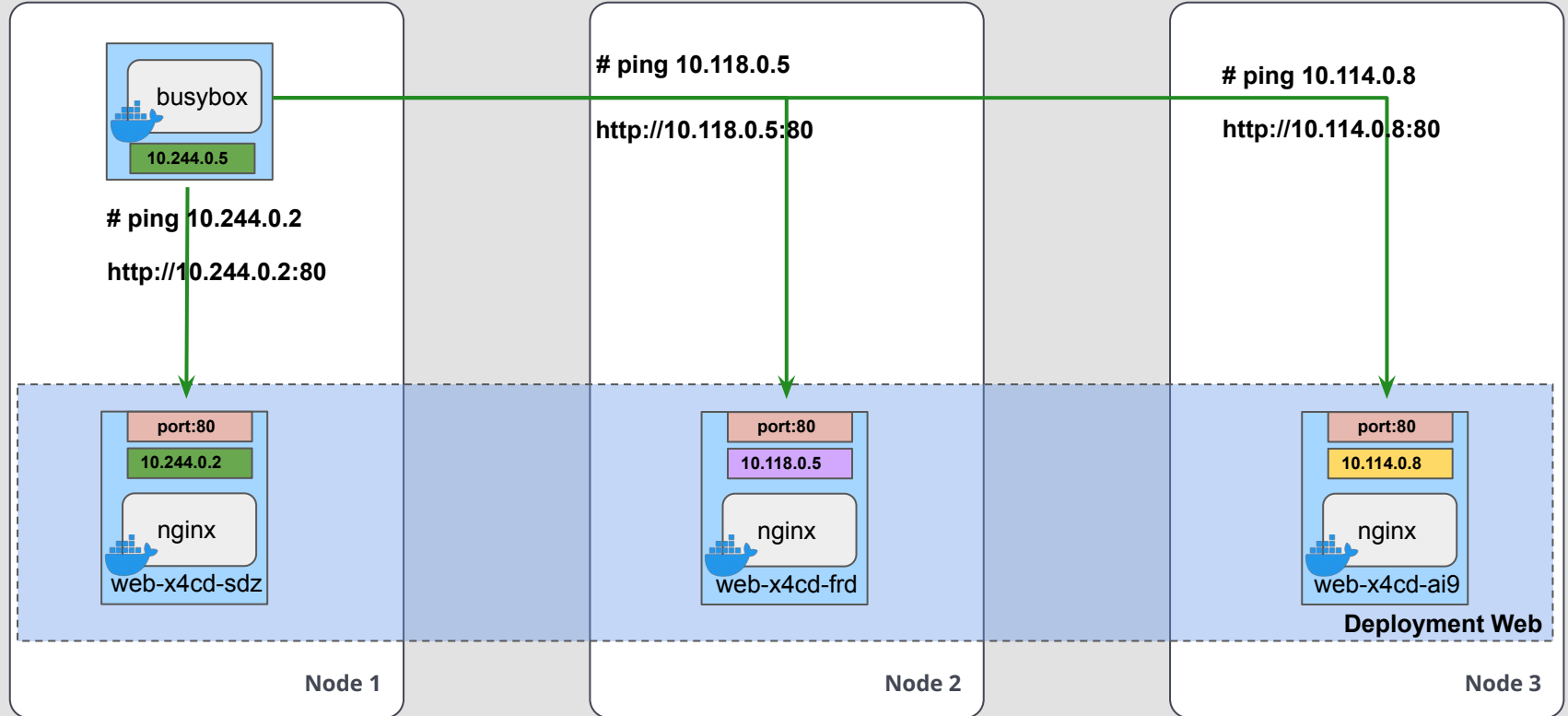
# Service



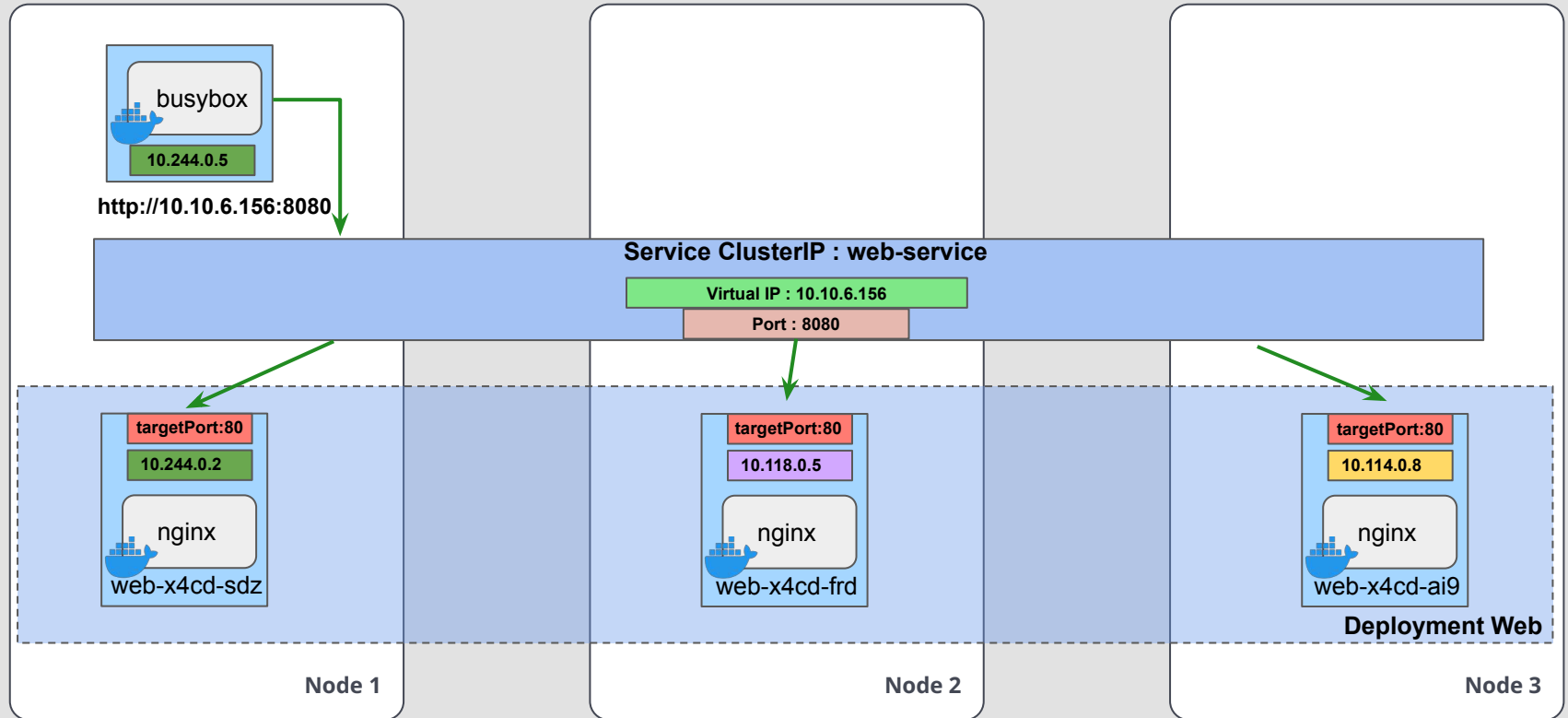
# Service



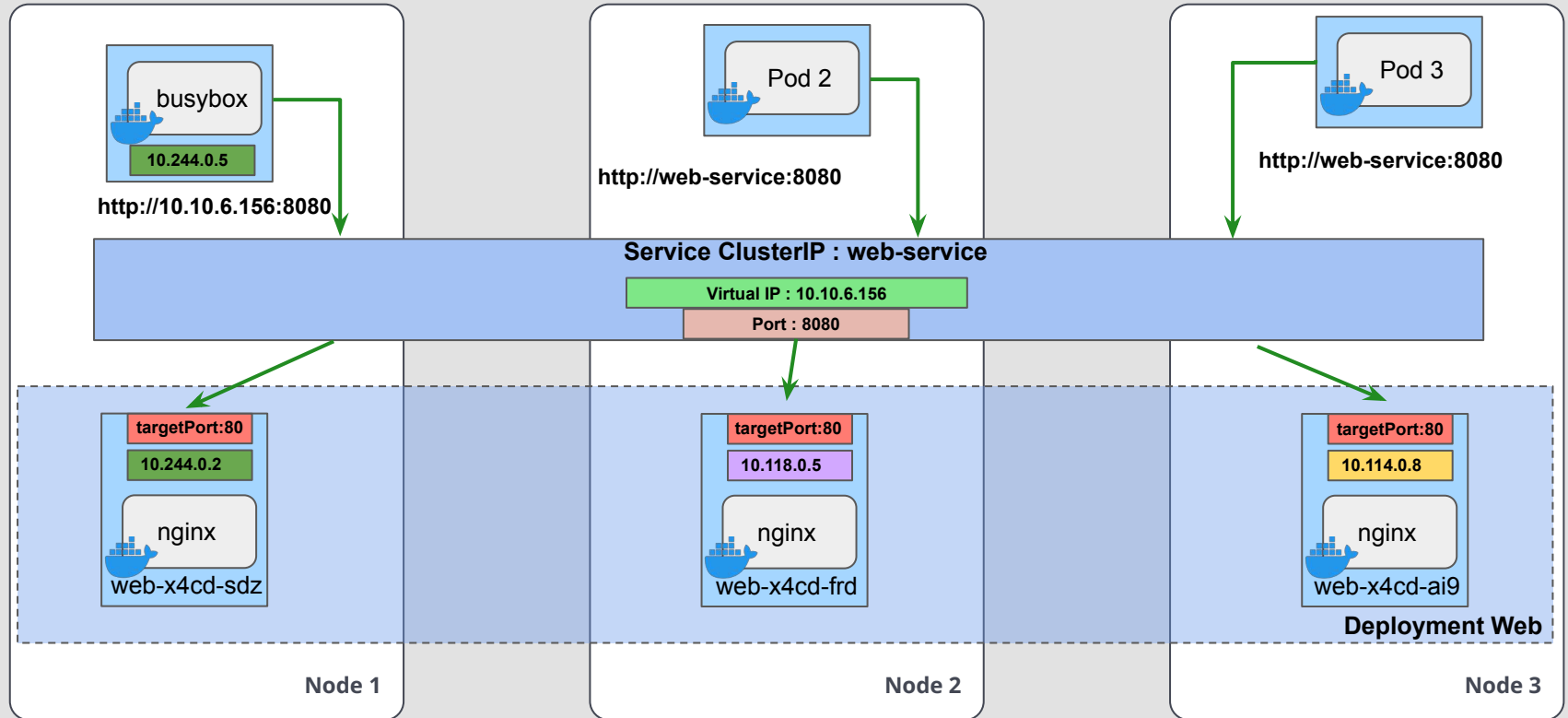
# Service



# Service ClusterIP



# Service ClusterIP





# Service ClusterIP

Indique les Pods que le service va regrouper (les Pods ayant les 2 labels mentionné)

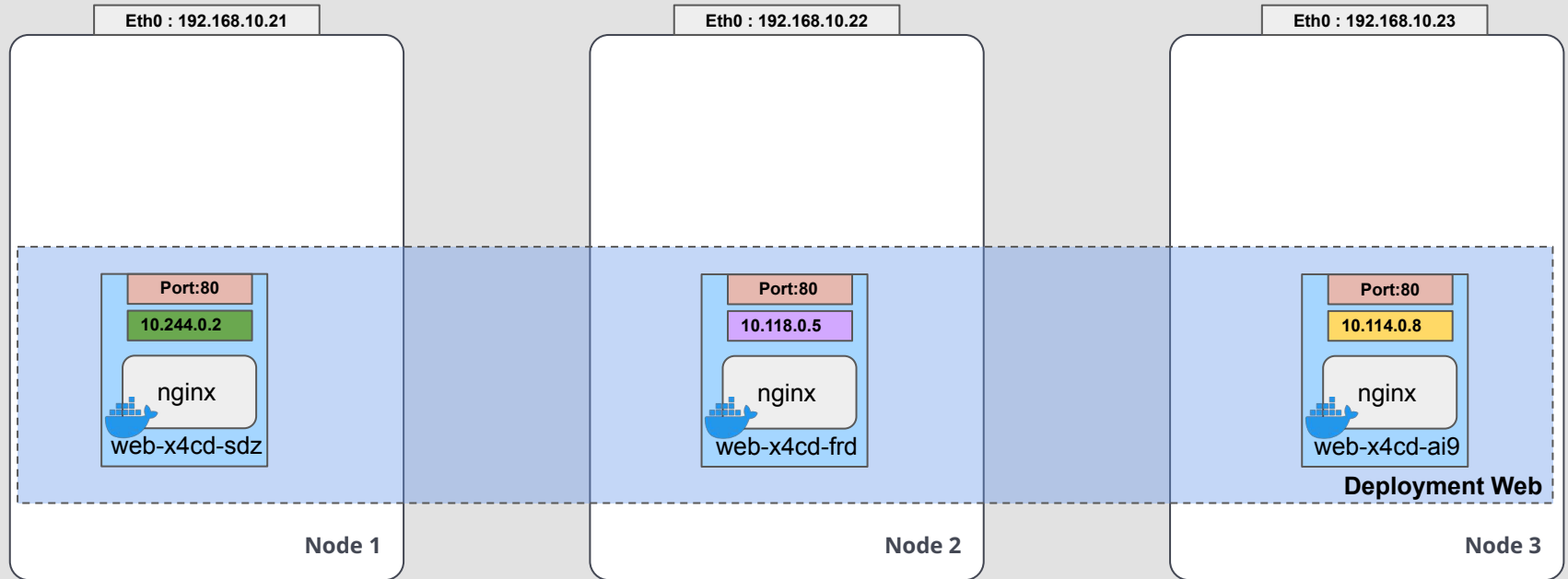
Type par défaut. D'autres Pods accèdent au service par son nom (DNS)

**Port :** le port exposé via l'IP du service  $\Rightarrow$  port 8080 exposé dans le Cluster

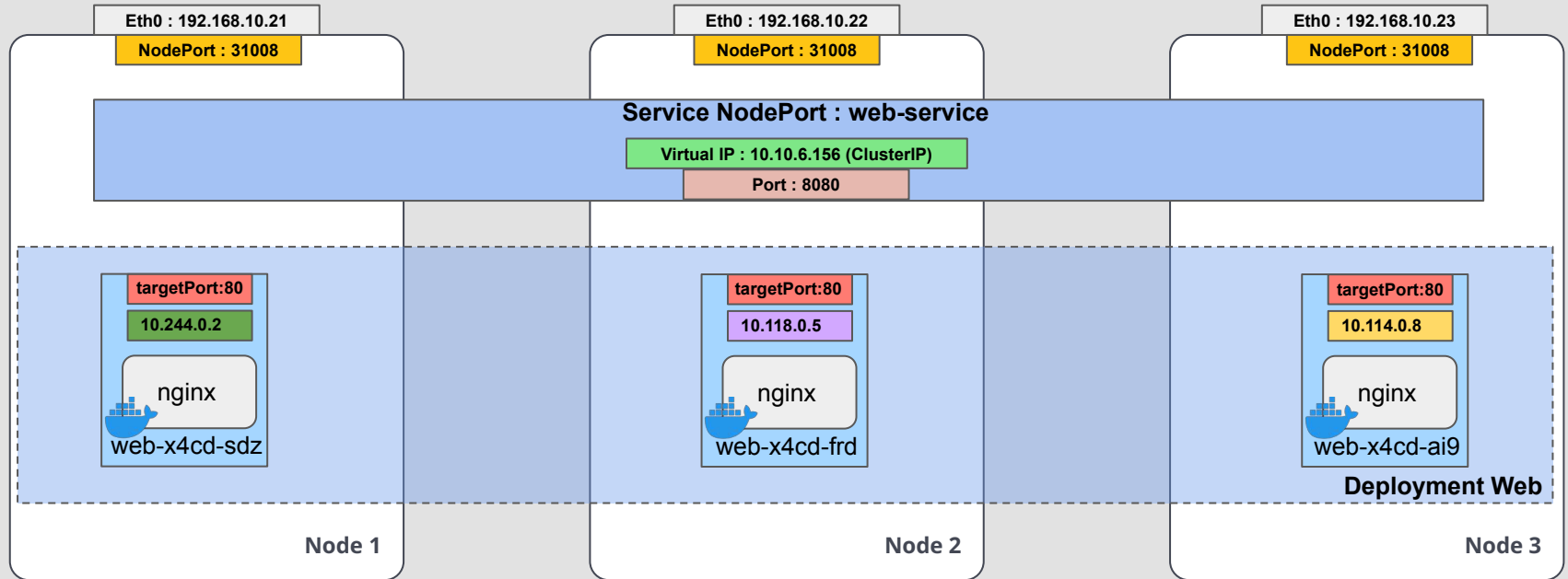
**targetPort :** le port du conteneur exposé  $\Rightarrow$  Les requêtes de ce service iront sur le port 80 des Pods du groupe

```
apiVersion: v1
kind: Service
metadata:
  name: web-service
spec:
  selector:
    app: web-server
    tier: frontend
  type: ClusterIP
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 80
```

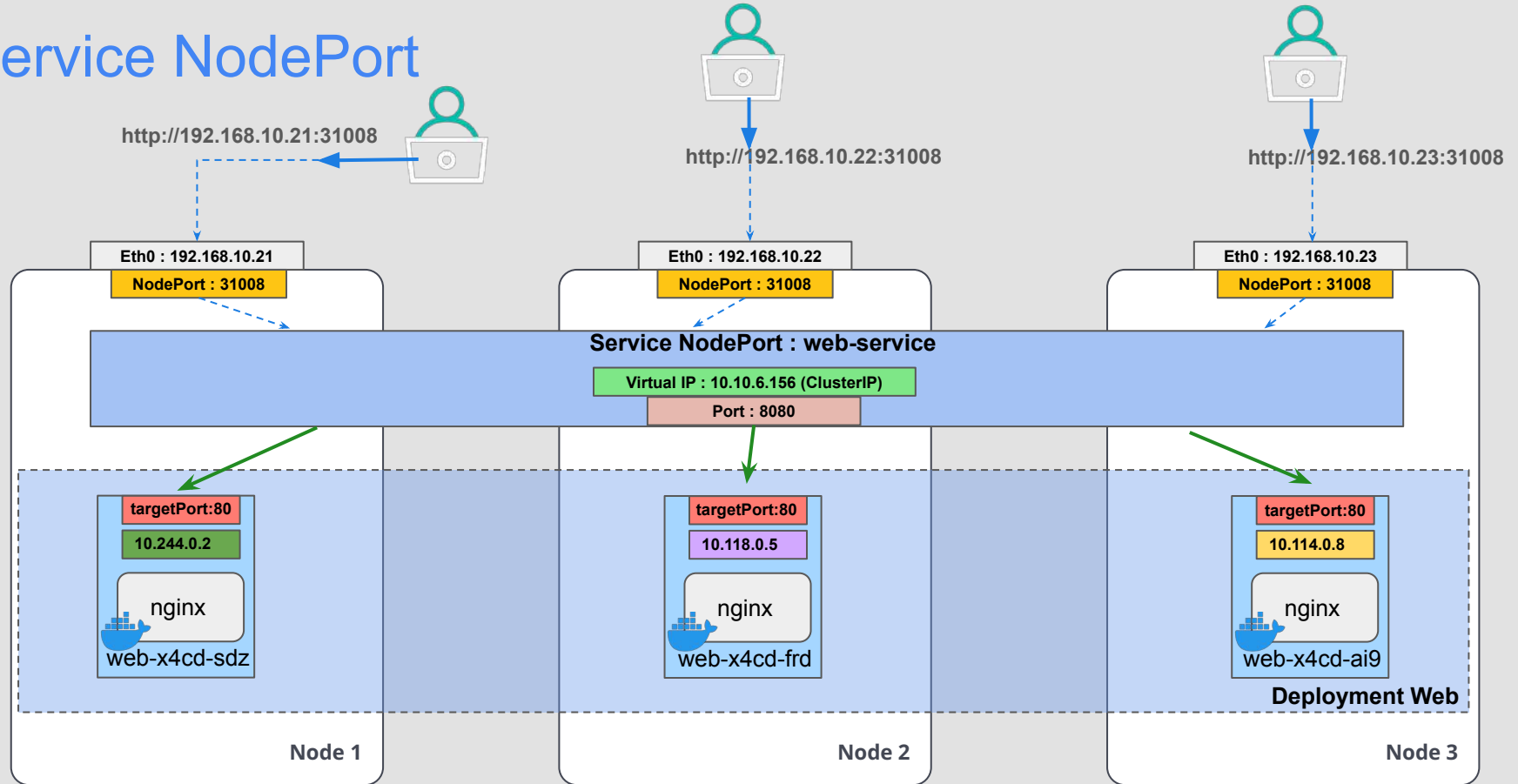
# Service NodePort



# Service NodePort



# Service NodePort



# Service NodePort

Indique les Pods que le service va regrouper (les Pods ayant les 2 labels mentionné)

Type par défaut : ClusterIP

**Port** : le port exposé via l'IP du service interne ClusterIP ⇒ port 8080 exposé dans le Cluster (Requêtes internes)

**targetPort** : le port du conteneur exposé ⇒ Les requêtes de ce service iront sur le port 80 des Pods du groupe

**nodePort** : le port du worker node qui sera exposé ⇒ (Requêtes externes) : choix entre : **30000 - 32767**

```
apiVersion: v1
kind: Service
metadata:
  name: web-service
spec:
  selector:
    app: web-server
    tier: frontend
  type: NodePort
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 80
      nodePort: 31008
```

# Service NodePort

```
mpakoupete@Mawakis-MacBook-Pro:~/kubernetes-labs
~/kubernetes-labs dev 121 713
> kubectl create -f clusterip-service.yaml --save-config
service/web-service-interne created

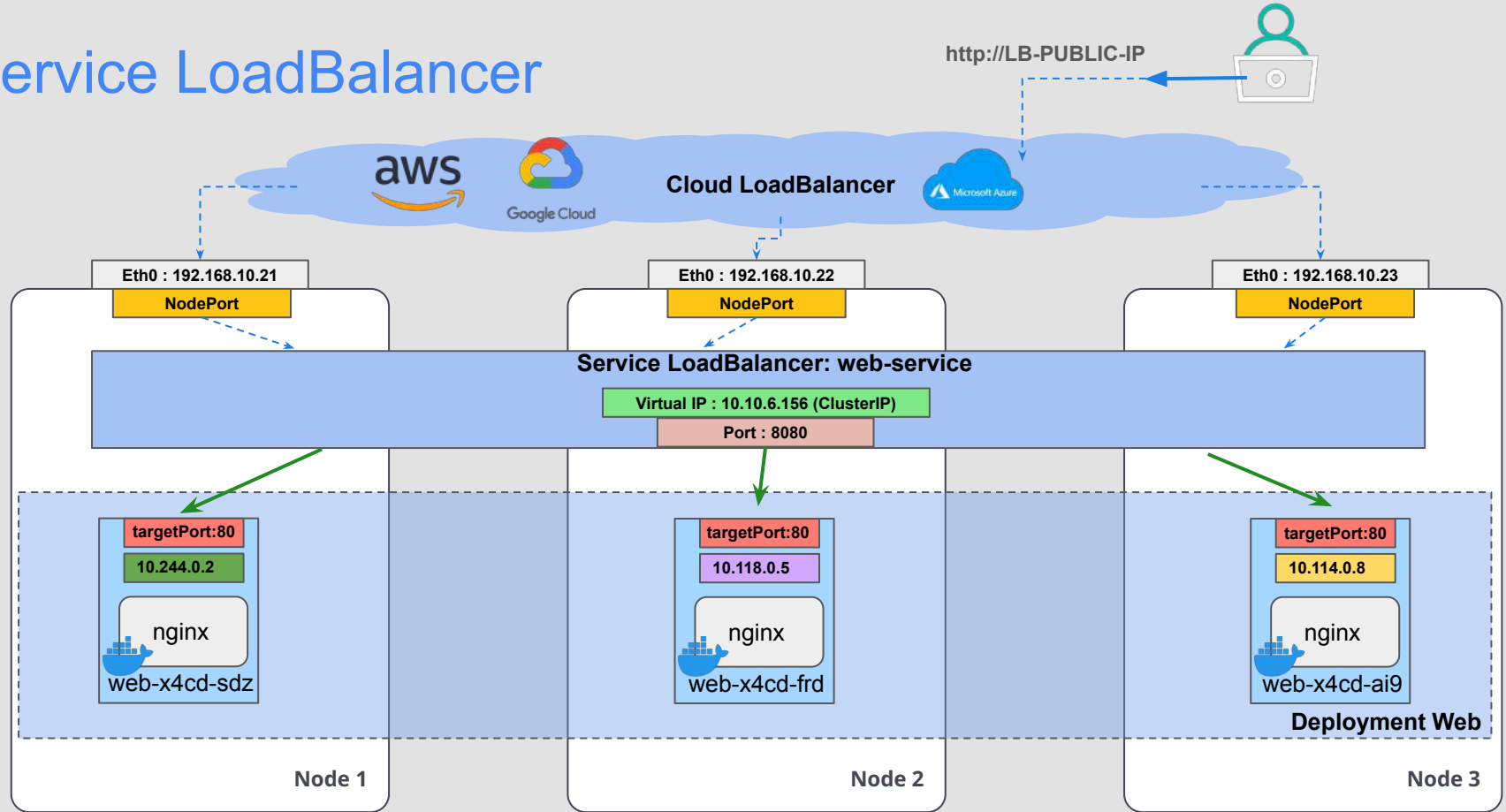
~/kubernetes-labs dev 121 713
> kubectl create -f nodeport-scv.yaml --save-config
service/web-service created

~/kubernetes-labs dev 121 713
> kubectl get service
NAME                TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
kubernetes           ClusterIP   10.96.0.1     <none>         443/TCP          10d
web-service          NodePort    10.100.219.198 <none>         8080:31008/TCP   8s
web-service-interne ClusterIP   10.98.252.242 <none>         8080/TCP         18s

~/kubernetes-labs dev 121 713
> kubectl get nodes -o wide
NAME                STATUS    ROLES    AGE   VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION   CONTAINER-RUNTIME
k8s-master-1        Ready     control-plane  10d   v1.25.3   192.168.56.11 <none>         Ubuntu 20.04.4 LTS   5.4.0-110-generic containerd://1.6.8
k8s-worker-1        Ready     <none>      10d   v1.25.3   192.168.56.12 <none>         Ubuntu 20.04.4 LTS   5.4.0-110-generic containerd://1.6.8
k8s-worker-2        Ready     <none>      10d   v1.25.3   192.168.56.13 <none>         Ubuntu 20.04.4 LTS   5.4.0-110-generic containerd://1.6.8
k8s-worker-3        Ready     <none>      10d   v1.25.3   192.168.56.14 <none>         Ubuntu 20.04.4 LTS   5.4.0-110-generic containerd://1.6.8

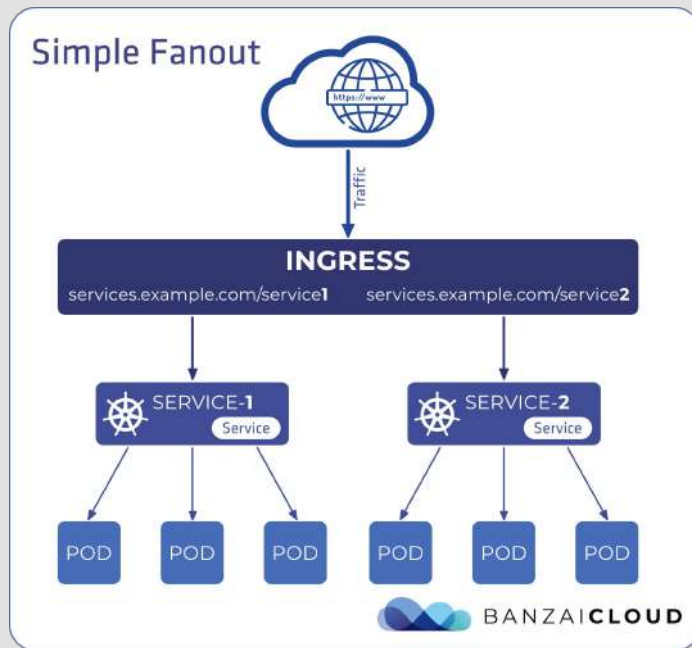
~/kubernetes-labs dev 121 713
> curl http://192.168.56.12:31008
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
  body {
    width: 35em;
    margin: 0 auto;
    font-family: Tahoma, Verdana, Arial, sans-serif;
```

# Service LoadBalancer



# Ingress

- Ensemble de règles permettant d'exposer un Service à l'extérieur d'un cluster Kubernetes
- Différents cas d'usage
  - ◆ Fournir des URLs visibles permettant d'accéder à un ou des Service Kubernetes
  - ◆ Routage HTTP (Ex: hôtes virtuels basés sur le nom)
  - ◆ Avoir des terminaison TLS
  - ◆ Faire du Load Balancing
- Nécessite qu'un **Ingress Controller** soit déployé
  - ◆ Container exécutant un processus de contrôle
  - ◆ Load balancer managé par kubernetes
  - ◆ Ex: nginx-controller, Traefik, HAProxy... cloud : GCE





# Ingress

**host:** les requêtes envoyé vers l'uri

<http://www.k8s-lab.com> seront forwardées vers les services en fonction du path

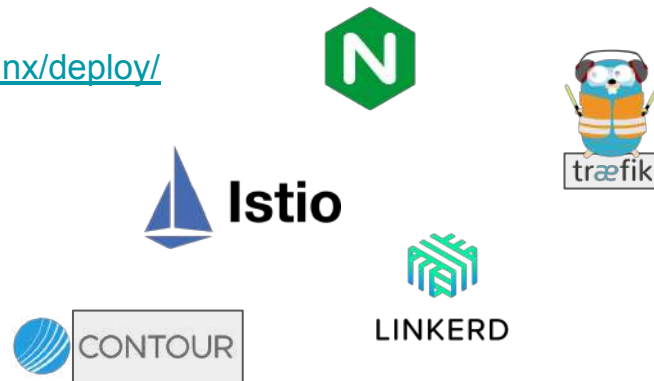
**Path:** les requêtes envoyé vers l'uri

<http://www.k8s-lab.com/frontend> seront forwardées vers le services [web-service-interne](#)

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: web-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - host: "www.k8s-lab.com"
    http:
      paths:
      - path: /frontend
        pathType: Prefix
        backend:
          service:
            name: web-service-interne
            port:
              number: 8080
      - path: /backend
        pathType: Prefix
        backend:
          service:
            name: api-service
            port:
              number: 80
```

# Ingress Controllers

- Pour que la ressource Ingress fonctionne, le cluster doit avoir un contrôleur Ingress en cours d'exécution. EX :
- NGINX Controller : <https://kubernetes.github.io/ingress-nginx/deploy/>
- Traefik : <https://github.com/containous/traefik>
- Istio ingress : <https://github.com/istio/istio>
- Linkerd : <https://github.com/linkerd/linkerd>
- Contour : <https://www.github.com/heptio/contour/>



# Ingress Controllers

```
mpakoupete@Mawaki-MacBook-Pro:~/kubernetes-lab
~/kubernetes-labs P dev 121 714
> kubectl create -f web-ingress.yaml
ingress.networking.k8s.io/web-ingress created

~/kubernetes-labs P dev 121 714
> kubectl get ingress
NAME CLASS HOSTS ADDRESS PORTS AGE
web-ingress <none> www.k8s-lab.com 80 26s

~/kubernetes-labs P dev 121 714
> kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.4.0/deploy/static/provider/cloud/deploy.yaml
namespace/ingress-nginx created
serviceaccount/ingress-nginx created
serviceaccount/ingress-nginx-admission created
role.rbac.authorization.k8s.io/ingress-nginx created
role.rbac.authorization.k8s.io/ingress-nginx-admission created
clusterrole.rbac.authorization.k8s.io/ingress-nginx created
clusterrole.rbac.authorization.k8s.io/ingress-nginx-admission created
rolebinding.rbac.authorization.k8s.io/ingress-nginx created
rolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
configmap/ingress-nginx-controller created
service/ingress-nginx-controller created
service/ingress-nginx-controller-admission created
deployment.apps/ingress-nginx-controller created
job.batch/ingress-nginx-admission-create created
job.batch/ingress-nginx-admission-patch created
ingressclass.networking.k8s.io/nginx created
validatingwebhookconfiguration.admissionregistration.k8s.io/ingress-nginx-admission created

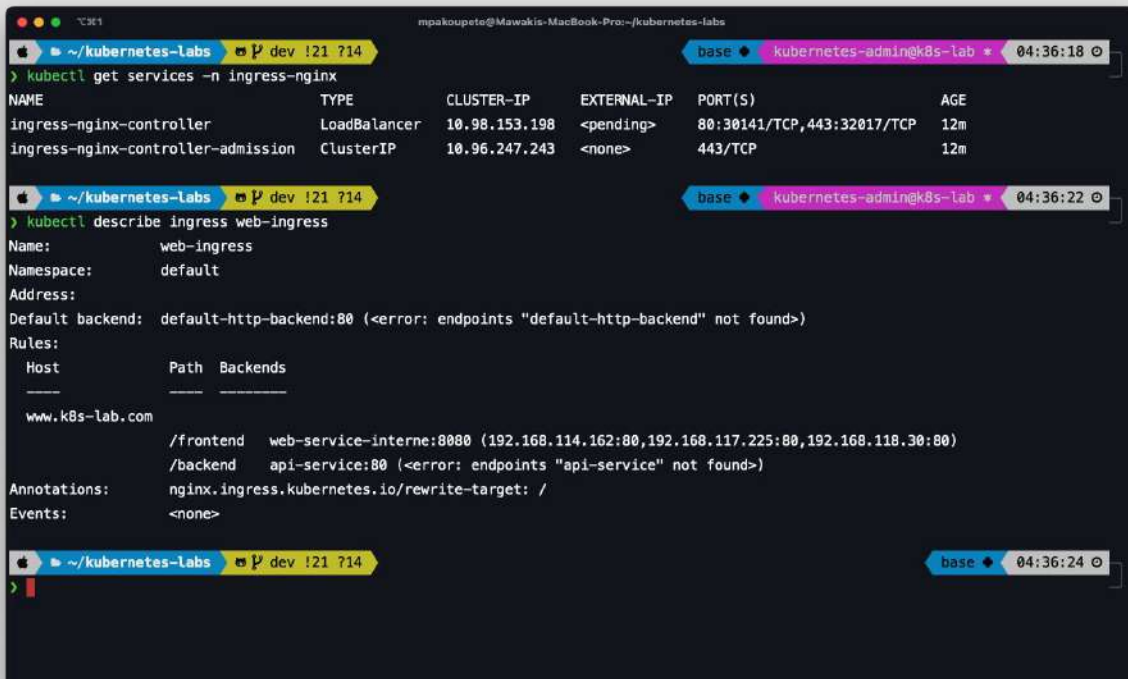
~/kubernetes-labs P dev 121 714
> kubectl get pods -n ingress-nginx
NAME READY STATUS RESTARTS AGE
ingress-nginx-admission-create-687hs 0/1 Completed 0 52s
ingress-nginx-admission-patch-r97b8 0/1 Completed 1 52s
ingress-nginx-controller-7844b9db77-q7cw4 1/1 Running 0 52s
```

# Ingress Controllers

**External-IP:** IP externe manquant parce que nous ne sommes pas dans un cluster Kubernetes Cloud (AWS, GCP, Azure...)

**/frontends:** Backends correspond aux IP+Ports des conteneurs exposé par le service web-service-interne

**/backend:** Backends error car nous n'avons pas de service api-service



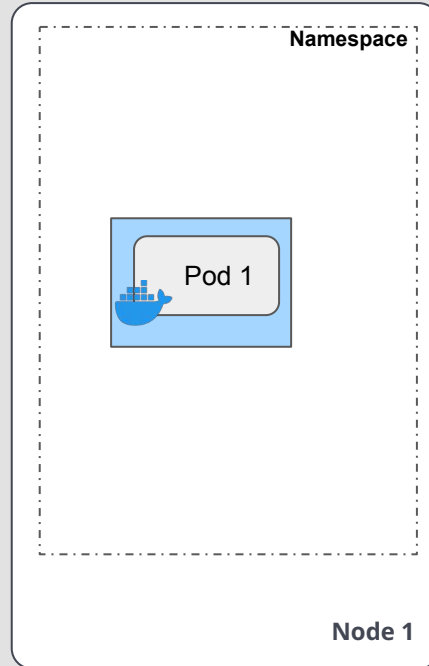
```
mpakoupete@Mawakis-MacBook-Pro:~/kubernetes-labs
~/kubernetes-labs P dev !21 714 base  kubernet...admin@k8s-lab 04:36:18
> kubectl get services -n ingress-nginx
NAME                                TYPE        CLUSTER-IP    EXTERNAL-IP  PORT(S)          AGE
ingress-nginx-controller            LoadBalancer  10.98.153.198  <pending>    80:30141/TCP,443:32017/TCP  12m
ingress-nginx-controller-admission  ClusterIP     10.96.247.243  <none>       443/TCP           12m

~/kubernetes-labs P dev !21 714 base  kubernet...admin@k8s-lab 04:36:22
> kubectl describe ingress web-ingress
Name:          web-ingress
Namespace:     default
Address:
Default backend: default-http-backend:80 (<error: endpoints "default-http-backend" not found>)
Rules:
  Host        Path  Backends
  ---  ---  ---
  www.k8s-lab.com
    /frontend  web-service-interne:8080 (192.168.114.162:80,192.168.117.225:80,192.168.118.30:80)
    /backend   api-service:80 (<error: endpoints "api-service" not found>)
Annotations:  nginx.ingress.kubernetes.io/rewrite-target: /
Events:       <none>
```

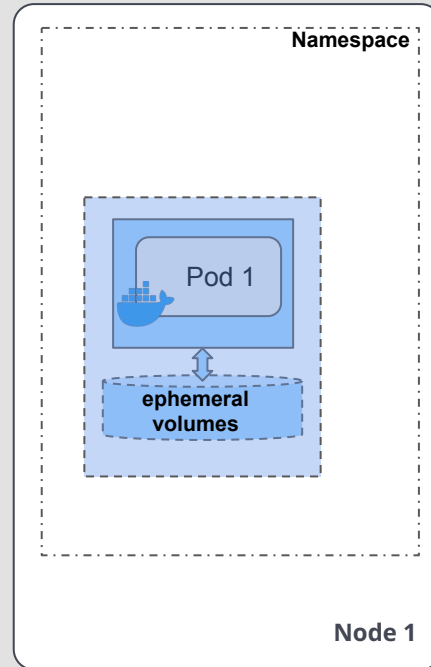
# Ressources de configuration & Stockage K8s

- Persistent Volumes
- Persistent Volumes Claim
- Storage Classes
- ConfigMaps
- Secrets

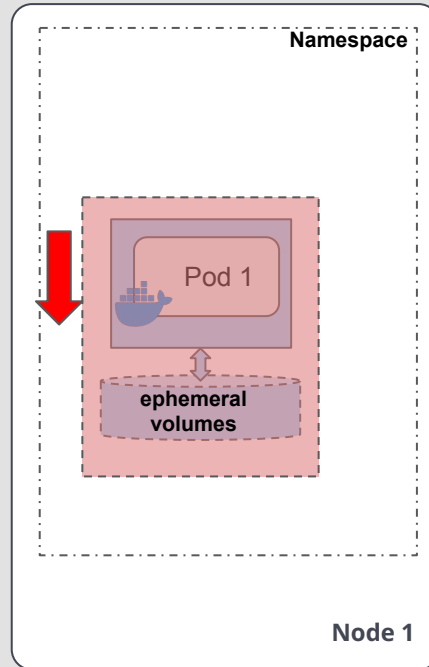
# Persistence de la donnée



# Persistance de la donnée

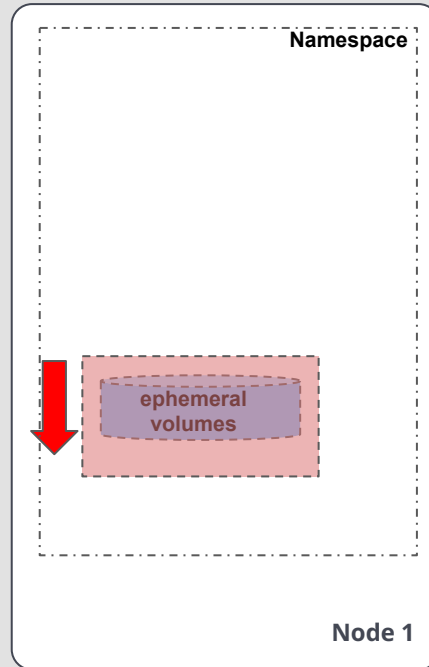


# Persistance de la donnée

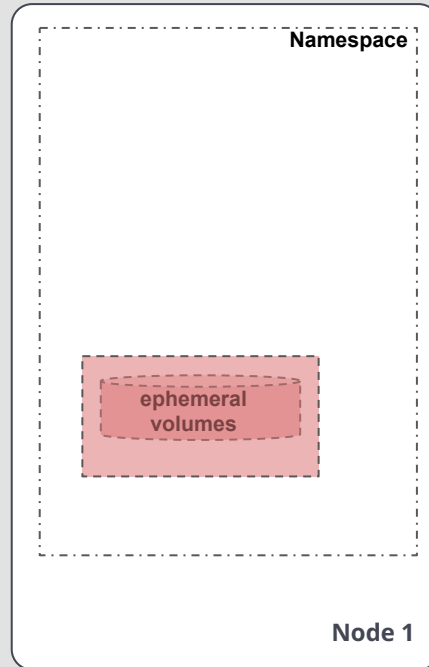




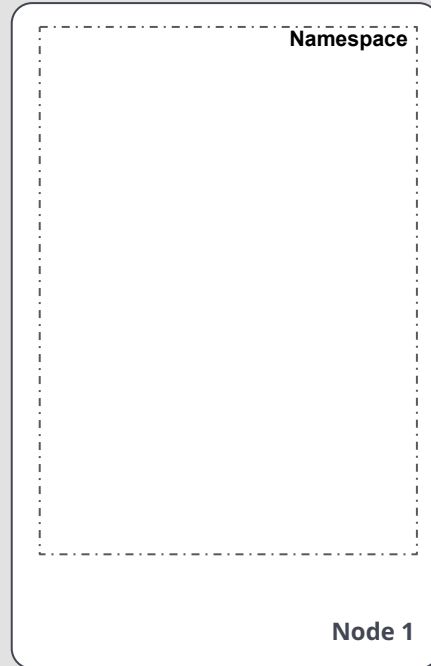
# Persistence de la donnée



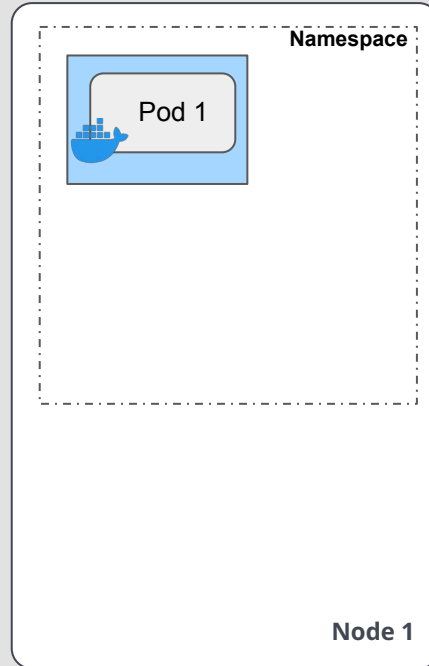
# Persistence de la donnée



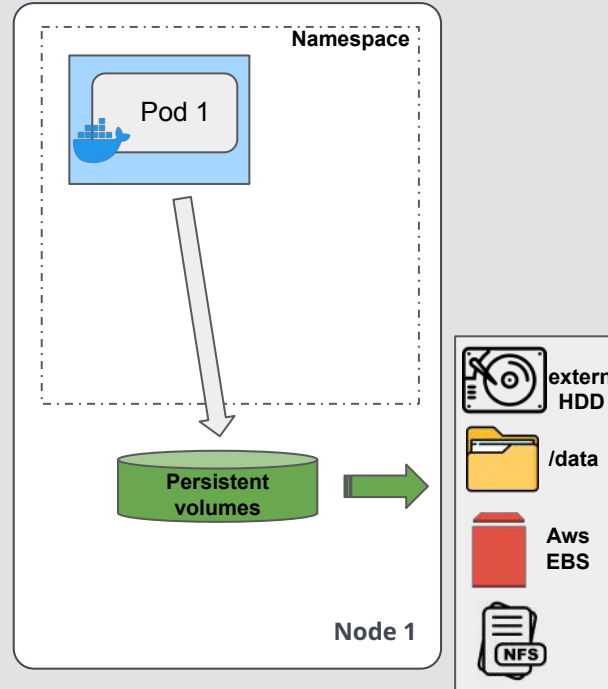
# Persistence de la donnée



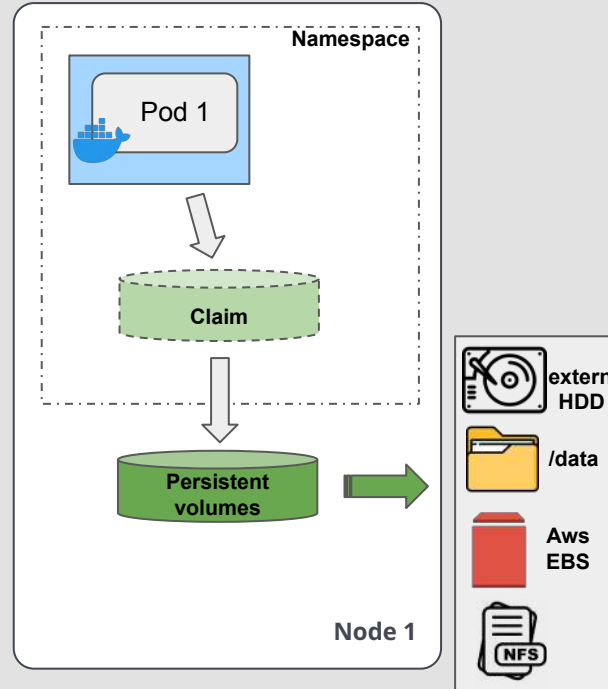
# Persistence de la donnée



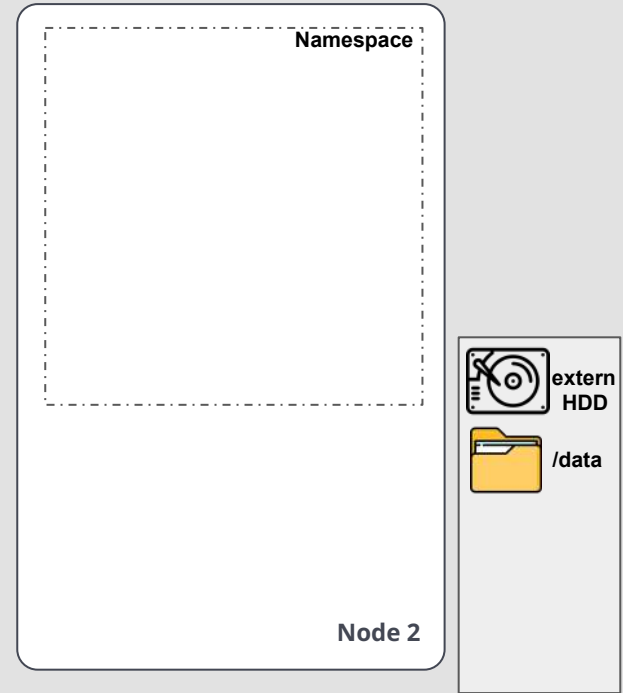
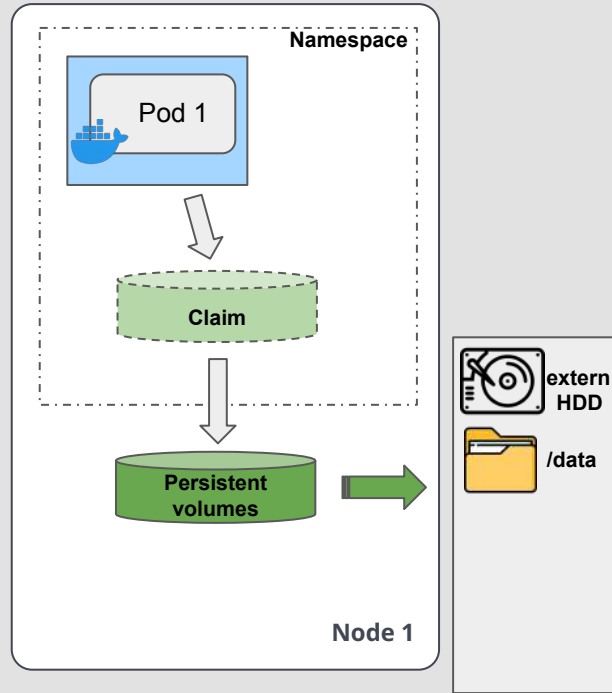
# Persistance de la donnée



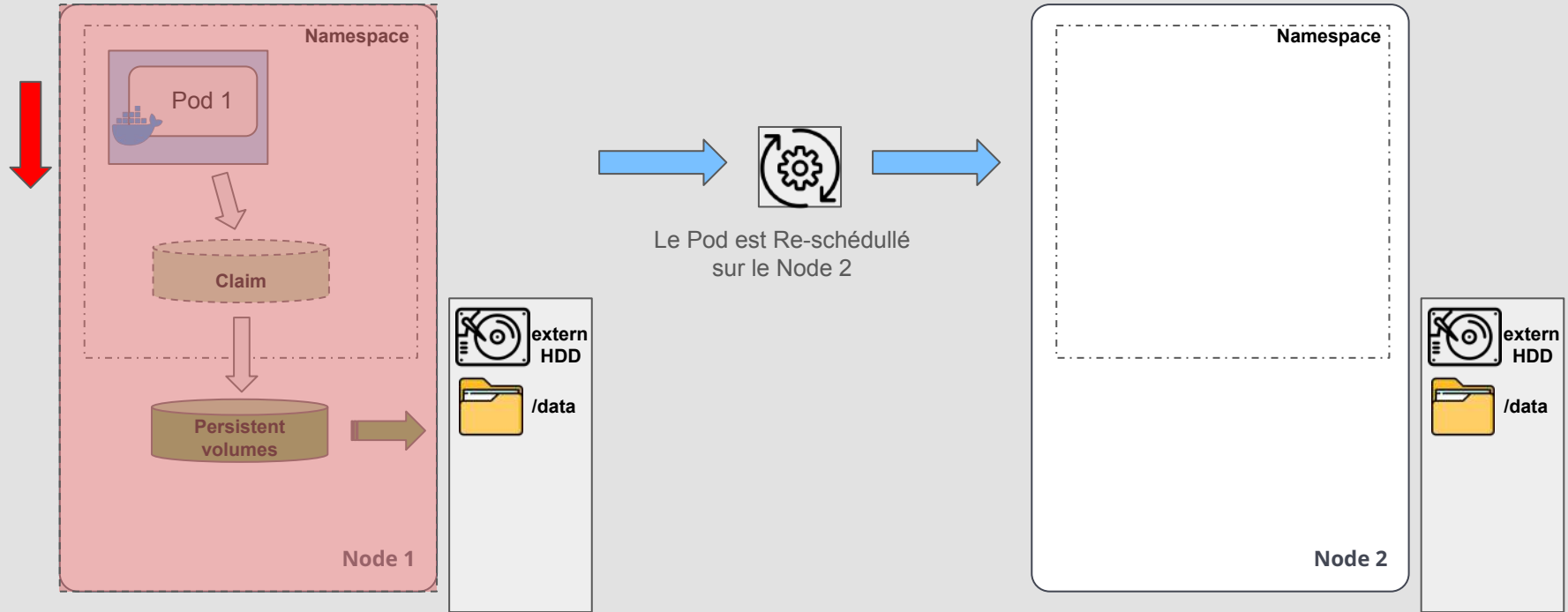
# Persistence de la donnée



# Persistence de la donnée (en Local)

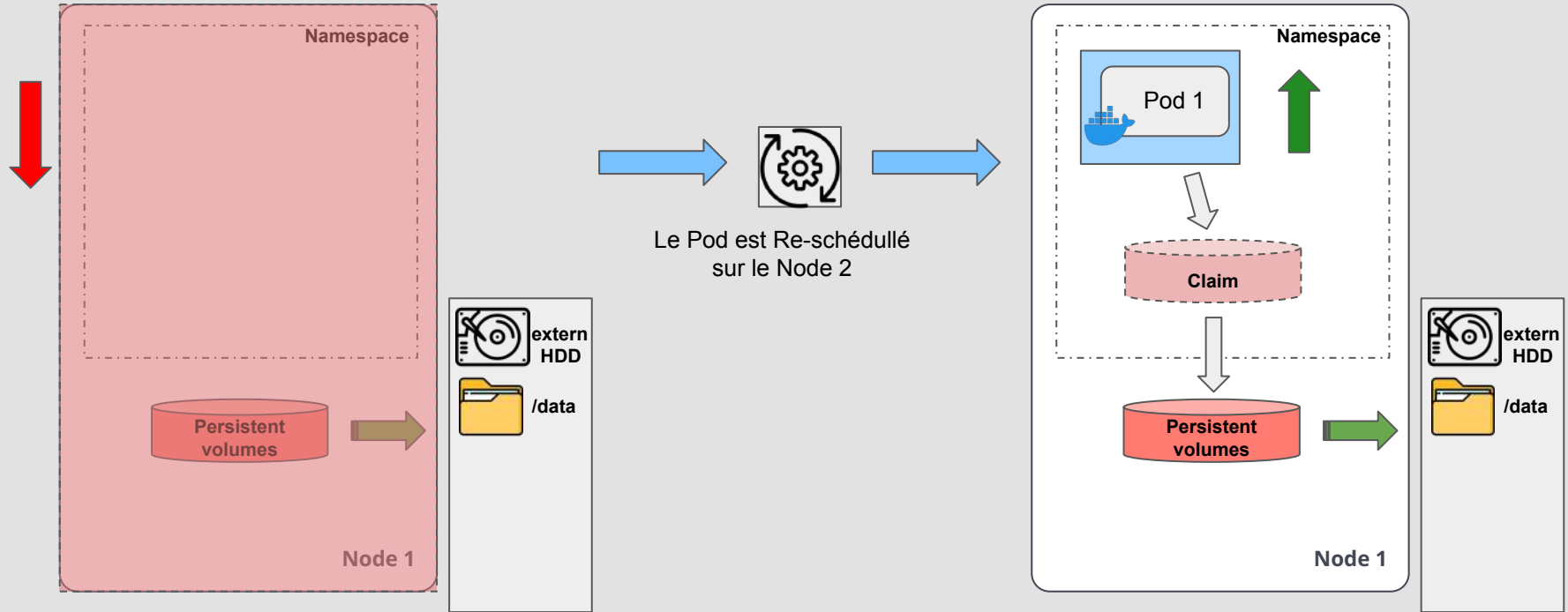


# Persistence de la donnée (en Local)

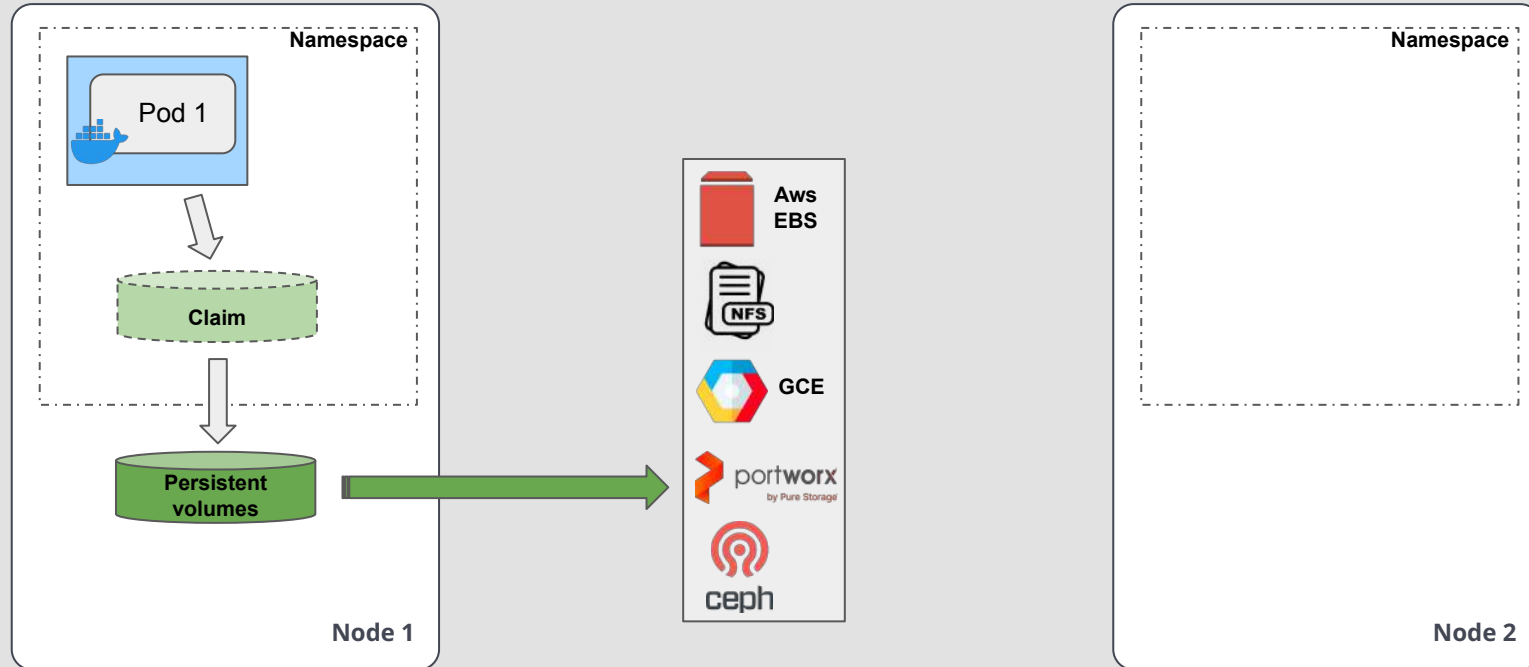




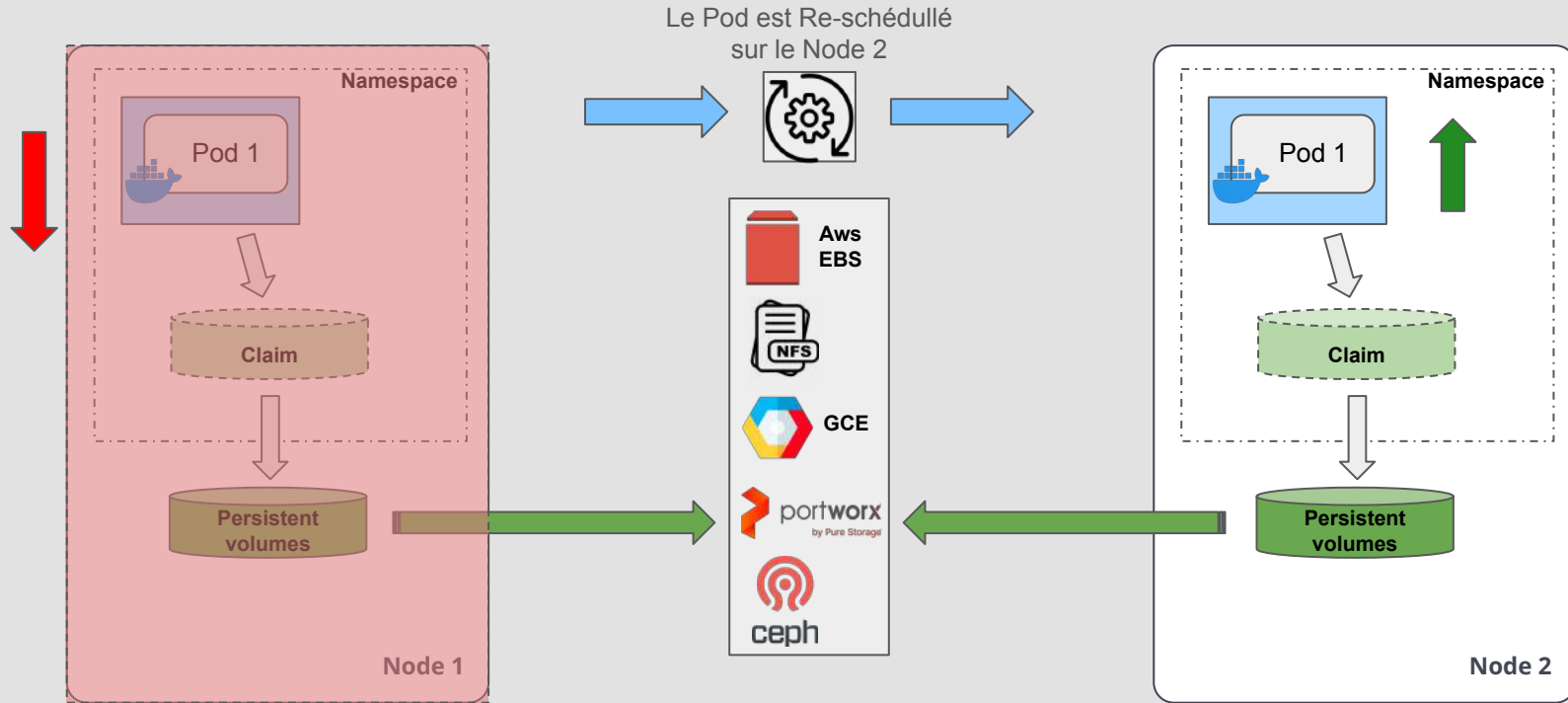
# Persistence de la donnée (en Local)



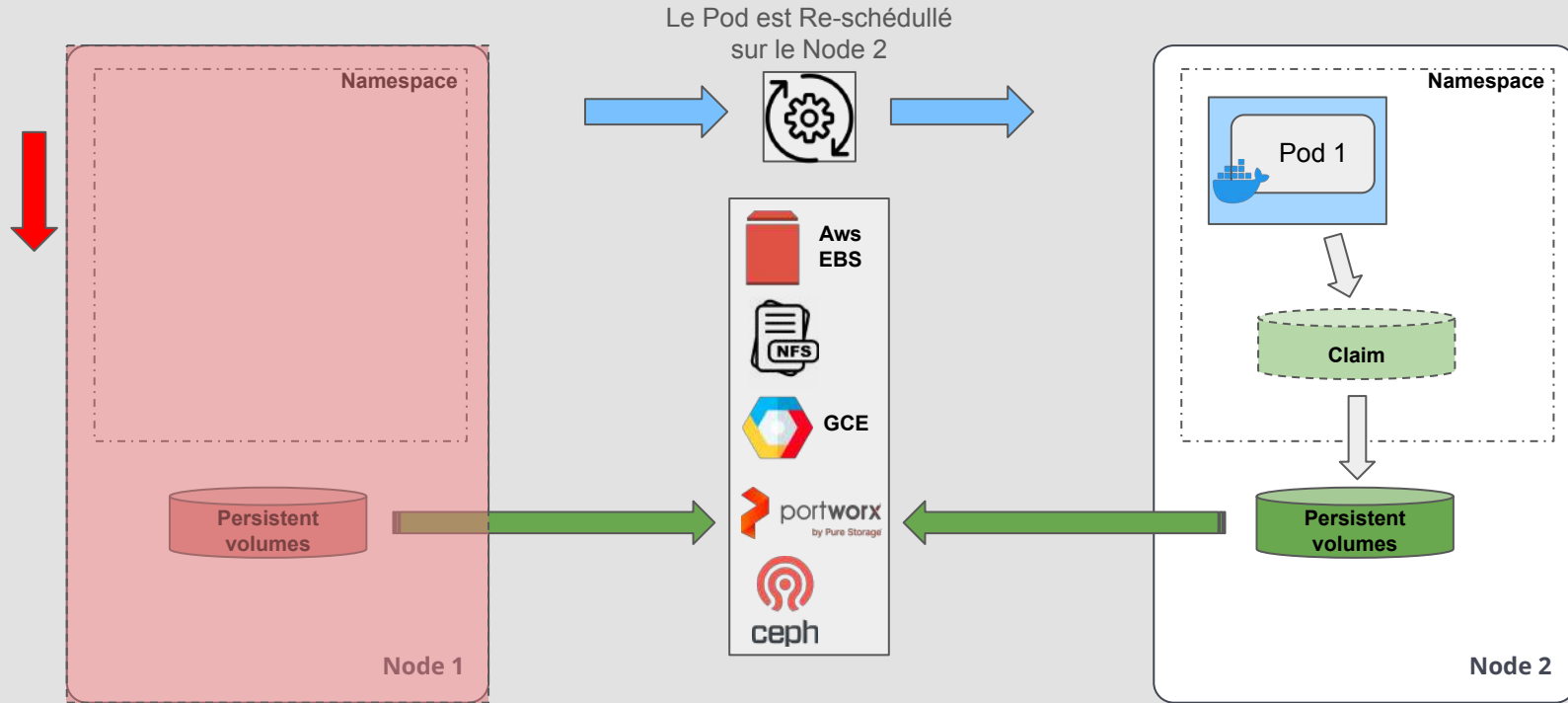
# Persistence de la donnée (Espace de stockage partagé)



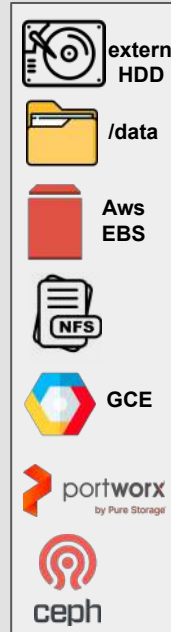
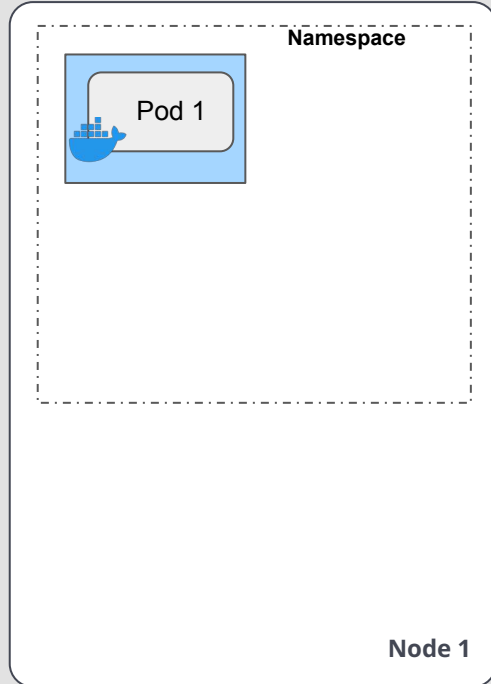
# Persistence de la donnée (Espace de stockage partagé)



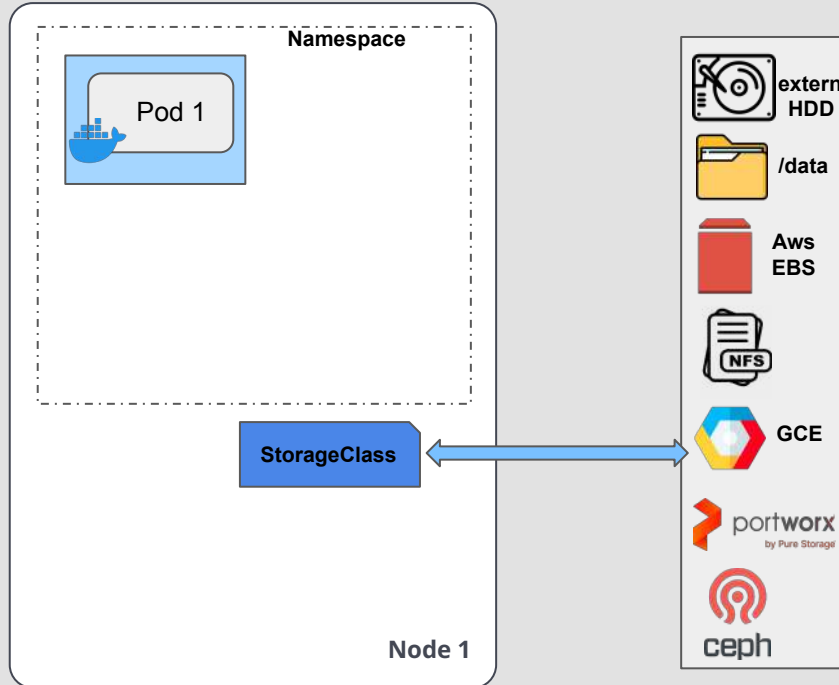
# Persistence de la donnée (Espace de stockage partagé)



# Provisionnement Dynamique (StorageClass )

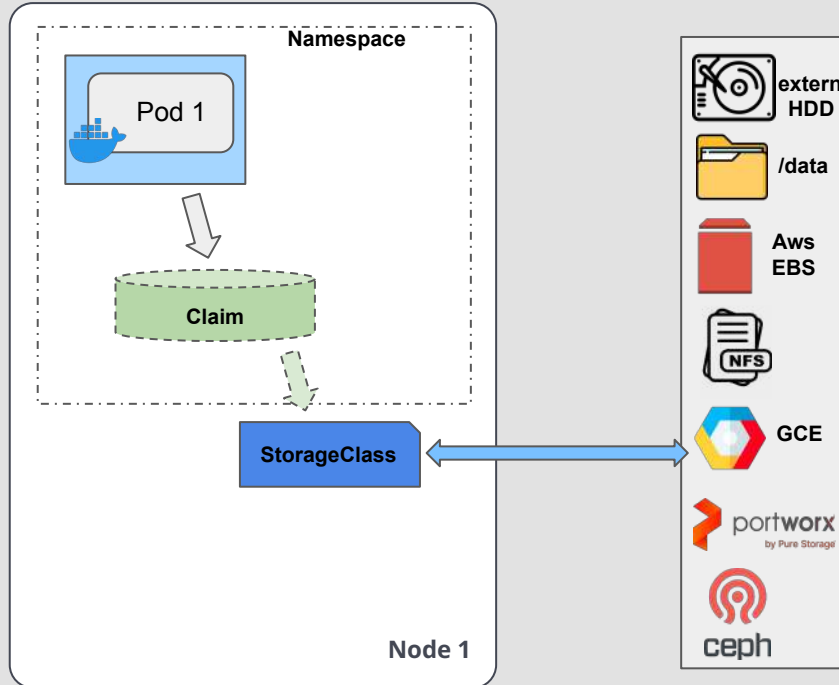


# Provisionnement Dynamique (StorageClass )



```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: fast
provisioner: kubernetes.io/gce-pd
parameters:
  type: pd-ssd
reclaimPolicy: Retain
allowVolumeExpansion: true
volumeBindingMode: WaitForFirstConsumer
```

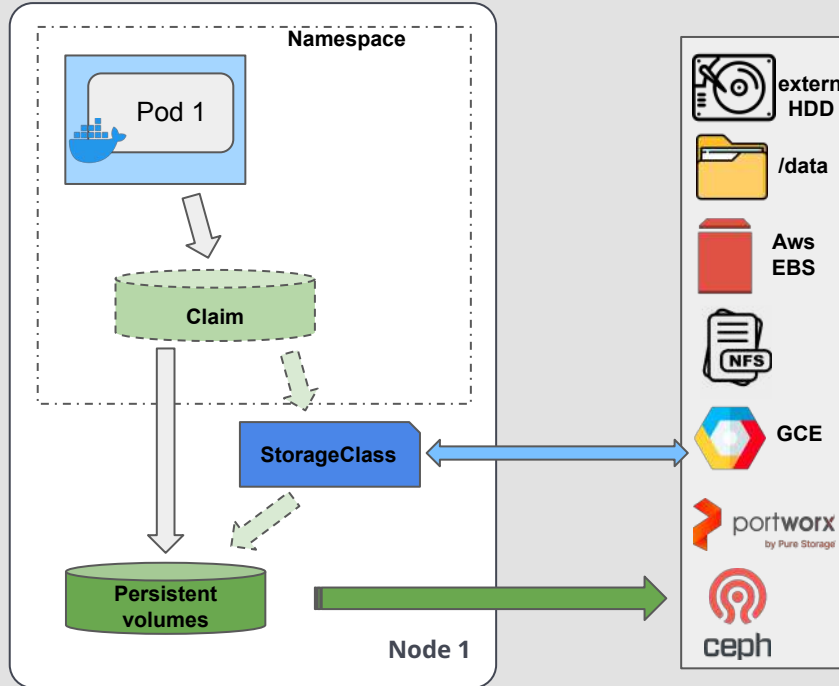
# Provisionnement Dynamique (StorageClass)



```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: fast
provisioner: kubernetes.io/gce-pd
parameters:
  type: pd-ssd
reclaimPolicy: Retain
allowVolumeExpansion: true
volumeBindingMode: WaitForFirstConsumer
```

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: claim1
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: fast
resources:
  requests:
    storage: 30Gi
```

# Provisionnement Dynamique (StorageClass)



```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: fast
provisioner: kubernetes.io/gce-pd
parameters:
  type: pd-ssd
reclaimPolicy: Retain
allowVolumeExpansion: true
volumeBindingMode: WaitForFirstConsumer
```

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: claim1
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: fast
  resources:
    requests:
      storage: 30Gi
```



# Persistent Volumes

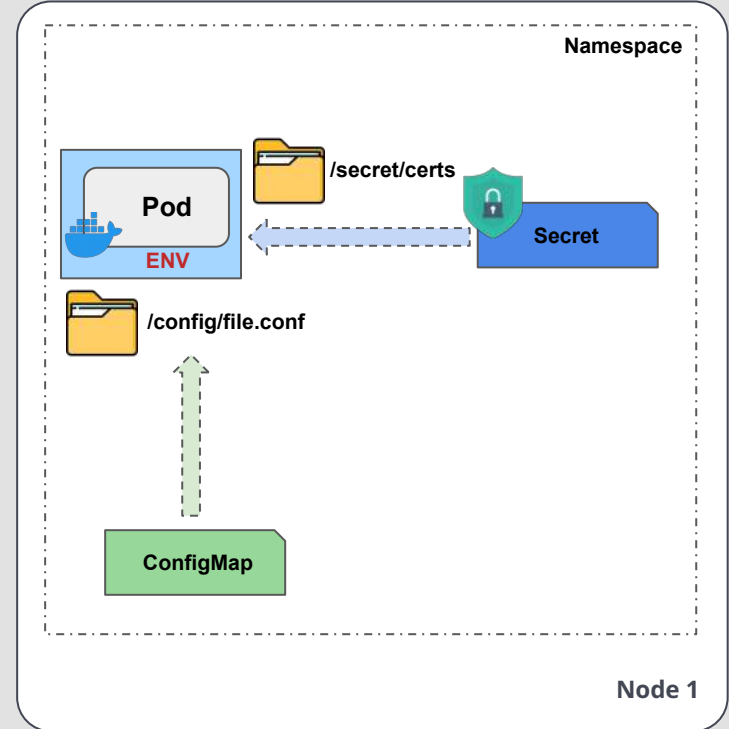
- Abstraction de la gestion des volumes
- Stockage provisionné statiquement ou dynamiquement (via une StorageClass)
- Type de stockage
  - ◆ Local
  - ◆ NFS / iSCSI / ...
  - ◆ GCEPersistentDisk
  - ◆ AWS EBS
  - ◆ AzureFile / AzureDisk
  - ◆ CephFS / Ceph Block Device

# Persistent Volumes

- Une demande de stockage
- Consomme un PersistentVolume
- Spécifie des contraintes supplémentaires
  - ◆ Taille
  - ◆ Type
  - ◆ Mode d'accès
- Création d'un binding entre PVC et PV

# ConfigMap & Secrets

- Définir des variables d'environnement
- Définir des secret (variable d'environnement masqué/opaque codé en base64)
- Monter un fichier (fichier de config, certificat, clé...) dans un Pod



# Secrets

- Objet permettant la protection des données sensible
  - ◆ Clé privée, mot de passe, URL sensible de connexion à un service tiers, ...
- Évite de définir ces informations dans les images de conteneur ou spécifications YAML de déploiement
  - ◆ Donne un meilleur contrôle sur leur utilisation
- Créé par un utilisateur ou par le système (clés d'accès à l'API)
- Utilisé dans un Pod
- Stocké dans etcd

# Types de Secrets

## → Generic

- ◆ Secret créé à partir d'un fichier, d'un répertoire ou d'une valeur littérale (Similaire à ConfigMap mais avec des valeurs masquées)

## → Docker-registry

- ◆ Secret utilisé pour l'authentification à un registre d'images de conteneurs(Ex: Aws ECR, Docker Private registry...)
- ◆ S'authentifier sur un registry Docker pour la Récupération des images privées

## → TLS

- ◆ Secret utilisé pour la gestion des clés (PKI)

# Secrets - Type generic

```
> kubectl create secret generic service-creds --from-literal username=mawaki --from-literal password=sEcretP@ss
secret/service-creds created
```

```
> kubectl get secret
NAME          TYPE      DATA   AGE
service-creds Opaque    2       7s
```

```
> kubectl describe secret service-creds
```

```
Name:          service-creds
Namespace:     default
Labels:        <none>
Annotations:   <none>
```

```
Type: Opaque
```

```
Data
====
password: 10 bytes
username: 6 bytes
```

```
> kubectl get secret service-creds -o yaml
apiVersion: v1
data:
  password: c0VjcmlVQ0UEBzcw==
  username: bWF3YWtp
kind: Secret
metadata:
  creationTimestamp: "2022-11-01T23:52:08Z"
  name: service-creds
  namespace: default
  resourceVersion: "408460"
  uid: 60b31b31-479e-4581-98e5-6181f19d71f6
type: Opaque
```

```
> kubectl get secret service-creds -o=jsonpath='{.data.username}' | base64 --decode
mawaki
```

```
> kubectl get secret service-creds -o=jsonpath='{.data.password}' | base64 --decode
sEcretP@ss
```

# Secrets - Type generic - utilisation (env)

```
> echo -n "mongodb+srv://dbadmin:secREtpAssw0rd@cluster0.grtxs.mongodb.net/?retryWrites=true&w=majority" | base64  
bW9uZ29kYitzcnY6Ly9kYmFkbWluOnNlY3JFdHBBc3N3MHJkQGNsdXN0ZlwlmdydHhzLm1vbmdvZGIubmV0Lz9yZXRYeVdyaXRlc310cnVlJnc9bWFqb3JpdHk=
```

```
apiVersion: v1  
kind: Secret  
metadata:  
  name: secret-mongodb  
type: Opaque  
data:  
  MONGO_CONNECTION_STRING: bW9uZ29kYitzcnY6  
  Ly9kYmFkbWluOnNlY3JFdHBBc3N3MHJkQGNsdXN0Z  
  XIwLmdydHhzLm1vbmdvZGIubmV0Lz9yZXRYeVdyaX  
  Rlc310cnVlJnc9bWFqb3JpdHk=
```

```
> kubectl create -f secret-mongodb.yaml  
secret/secret-mongodb created
```

# Secrets - Type generic - utilisation (env)

```
> kubectl apply -f pod-webapp.yaml  
pod/backend created
```

```
> kubectl exec -it backend -- bash  
root@backend:/# env | grep -i mongo  
MONGO_URL=mongodb+srv://dbadmin:secreTpaSsw0rd@cluster0.grtxs.mongodb.net/?retryWrites=true&w=majority  
root@backend:/#
```

```
apiVersion: v1  
kind: Pod  
metadata:  
  labels:  
    run: backend  
  name: backend  
spec:  
  containers:  
  - name: backend  
    image: backend-app:2.0  
    env:  
    - name: MONGO_URL  
      valueFrom:  
        secretKeyRef:  
          name: secret-mongodb  
          key: MONGO_CONNECTION_STRING
```



# Secrets - Type generic - utilisation (volume file)

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: backend
    name: backend
spec:
  containers:
  - name: backend
    image: nginx
    volumeMounts:
      - mountPath: "/etc/creds/mongodb"
        name: mongodb-creds
        readOnly: true
  volumes:
  - name: mongodb-creds
    secret:
      secretName: secret-mongodb
```

# Secrets - Type generic - utilisation (volume file)

```
> kubectl apply -f pod-webapp.yaml
pod/backend created
```

```
> kubectl exec -it backend -- bash
root@backend:/# ls /etc/creds/
mongodb
root@backend:/# ls /etc/creds/mongodb/
url
root@backend:/# cat /etc/creds/mongodb/url
mongodb+srv://dbadmin:secreTpaSsw@rd@cluster0.gtrxs.mongodb.net/?retryWrites=true&w=majorityroot@backend:/#
root@backend:/# env | grep -i mongo
root@backend:/#
```

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: backend
  name: backend
spec:
  containers:
  - name: backend
    image: nginx
    volumeMounts:
    - mountPath: "/etc/creds/mongodb"
      name: mongodb-creds
      readOnly: true
  volumes:
  - name: mongodb-creds
    secret:
      secretName: secret-mongodb
      items:
      - key: MONGO_CONNECTION_STRING
        path: url
```

## Secrets - Type docker-registry

```
> kubectl create secret docker-registry mpakoupete-docker-creds --docker-server=$PRIVATE_REGISTRY_URL
--docker-username=$USERNAME --docker-password=$DOCKER_PASSWD --docker-email=$DOCKER_EMAIL
```

```

$ kubectl get secret mpakoupete-docker-creds -o yaml
apiVersion: v1
data:
  .dockerconfigjson: eyJhdXN0YXZpd29yZCI6Ikt1Zm1pZUAyMTE3IiwiaWZlhaWwioiOiJ0YXdhb2IAbXBha291cGV0ZS5jb20iLCJhdXRoIjoiaWYlYlhaWwGey0TFjR1YwWLRwTmMyRnRhVlZBTWpFeE53PT0ifX19
kind: Secret
metadata:
  creationTimestamp: "2022-11-02T01:07:33Z"
  name: mpakoupete-docker-creds
  namespace: default
  resourceVersion: "415921"
  uid: f8aa26d5-481a-4c33-8628-3a9e4056954c
type: kubernetes.io/dockerconfigjson

```

```
apiVersion: v1
kind: Pod
metadata:
  name: custom-webapp
spec:
  containers:
    - name: react-front
      image: mpakoupete/client
  imagePullSecrets:
    - name: mpakoupete-docker-creds
```

# Secrets - Type docker-registry

```
Containers:
  react-front:
    Container ID:   containerd://634a852db251b1ddc21b8289a6f41c7fe588f42d20b59c486229171ae372279a
    Image:          mpakoupete/client
    Image ID:       docker.io/mpakoupete/client@sha256:7d40600d89a45bbace4c848175dc0c4dc8d8a0e3a2b78347170947710ce10dd1
    Port:          <none>
    Host Port:     <none>
    State:         Running
      Started:     Wed, 02 Nov 2022 02:13:22 +0100
    Ready:         True
    Restart Count:  0
    Environment:   <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-7xpl7 (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  kube-api-access-7xpl7:
    Type:          Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds: 3607
    ConfigMapName:  kube-root-ca.crt
    ConfigMapOptional: <nil>
    DownwardAPI:    true
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                  node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason      Age   From          Message
  ----    -
  Normal  Scheduled   19m   default-scheduler   Successfully assigned default/custom-webapp to k8s-worker-1
  Normal  Pulling     19m   kubelet         Pulling image "mpakoupete/client"
  Normal  Pulled      19m   kubelet         Successfully pulled image "mpakoupete/client" in 18.40880191s
  Normal  Created     19m   kubelet         Created container react-front
  Normal  Started     19m   kubelet         Started container react-front
```

```
apiVersion: v1
kind: Pod
metadata:
  name: custom-webapp
spec:
  containers:
  - name: react-front
    image: mpakoupete/client
  imagePullSecrets:
  - name: mpakoupete-docker-creds
```

## Secrets - Type TLS

```
> openssl req -x509 -nodes -newkey rsa:2048 -days 365 -keyout myPrivateKey.pem -out myPublicKeyCert.pem
Generating a RSA private key
```

```
> kubectl create secret tls k8s-lab-pki-tls --cert myPublicKeyCert.pem --key myPrivateKey.pem
secret/k8s-lab-pki-tls created
```

```

$ kubectl get secrets k8s-lab-pki-tls -o yaml
apiVersion: v1
kind: Secret
metadata:
  creationTimestamp: "2022-11-02T01:31:59Z"
  name: k8s-lab-pki-tls
  namespace: default
  resourceVersion: "418335"
  uid: 739505ae-3a02-44ea-989b-2f27fafb6291
type: kubernet.es.io/tls
data:
  tls.crt: LS0tLS1CRUdJTS1BRVQJUSUzQ0FURS0tLS0tck1JSUQ2VENDQXRHOF3SUJBZ0tVMWkxMG81NnRwUkR6a2ZkS1k5
RvdfFb7FoZDJGcmFVbnQjPSE10YkdGaUxhTnZlVEF1RncweU1qRgkTURjd0U1UjRnRkZjHhRncweU16RXhNRE13TVRRNE16QmFNSI
pUwEwR0NTcUd0TSWlZcRFFQkFRVUFBNE1CRhdB2dnRUtBb0lCQVDFZApYUm91TzUzR3Avm1YyUjBjVXRyZjU1UkxhUzH1MjVlVlQ
VXZlY2V1UjQajhRjQnNmS9SEtTmxRR3l1WfJpb3VKChVxV1FCQTF1WmfZCndxdjRaeEtsWUJ0eDMwTjklX58CeU5MnNmW505
3FHU0l10MRRRUjDd1VBQTRAJQkFRQl1JVaVmVqNbxczFZNh1XN2jVVMxdvBRqcVAKVR1jY2d0MUUraWlAtTh2cXQweXg1ellGw
U0l0nm9HdhnS5mRkmdvtQjdhVEgRnQvEakNbvWNP0XF1JqV1dm1jV2UC9zSmdYQRpsdd1EeExDbEnve1hEMkFsd3R2mENtIU
tls.key: LS0tLS1CRUdJTS1BQkU1WQVRFIETfW50tLS0tck1JSUvZd0LCQRUBTJna3Foa21HOXcwQkFRRUZB0VNDQkdrt2d
VdKzd0hm9NML1QxV6xPMWlZV1J1THRFAURDR2JjZn1S2hJYTGkQ3FD0fRnE1VcWu1a0paUgoA0IzZ0pvUkhlU2psUd5Z
rL2svmFY0XZK2CNMNMkR272SaS0pJedNuTGQ2eEUzQUF6VzB0dVh6Wgowby9MQ0j1hUXRYZ1R5YU91UFFFRGVM5VYrN3F3eWZj
R4V2sreDN5eFBJajhDvNEwV3AaeXhFQd2ZRUF6cz05xeXvvBd5UkdQ2Hd0ClE3WjJMZ2xId1B2a005cURueG5YLz12bmhyQVZ
DHGSEhQhU0MzFWCwtMwXRadE5wZSRZmZkYkxzeGR5dyt2cWJaZ1J0cTZET1KXWke4en1ae19UaTjTajgzFgzS3ZrW5laZ
H1FFTE9aQ2Z2ycUjpeV5mNWbAhUSFLaT0N20GtB0Hh5SXZVh0Z5T23JUUtCZ1FDWQoyR2xWYXlydzh15TVzVghYzVdhMDLYT
FVExrcE1YL1nCPQTLFQ1VyM21nV1V2ZU11Zn50c1NHYXGtGY5dTBZ1VrTjU2WncXRAV2Cndhd0xMTEHuVzhyaZhLQkFLKzZjY
kind: Secret

```

```
apiVersion: v1
kind: Pod
metadata:
  name: client
spec:
  containers:
  - name: client
    image: busybox
    args:
    - "sleep"
    - "3000"
  volumeMounts:
  - mountPath: "/etc/tls/certs"
    name: tls
    readOnly: true
  volumes:
  - name: tls
    secret:
      secretName: k8s-lab-pki-tls
```

# Secrets - Type TLS

```
> kubectl create -f pod-client.yaml
pod/client created
```

```
> kubectl exec -it client -- sh
/ # ls /etc/tls/certs
tls.crt  tls.key
/ # cat /etc/tls/certs/tls.crt
-----BEGIN CERTIFICATE-----
MIID6TCCATGgAwIBAgIUxy/0o56tpRDzkfDJZh0rzhr22dowDQYJKoZIhvcNAQEL
BQAwYmxczAJBgNVBAYTAkZSMRYwFAYDVQQIDAI3bGUgZGUGRnJhbmNlMQ4wDAYD
VQQLHDAVQYXJpczERMABGAlUECgwISzhzLUXhYnMxYjFjAUBgNVBAMMSouazhLWxh
Yi5jb20xITAFBgkqhkiG9w0BCEQEWEm1hd2FraUBrOHMtbGFiLnNvbTAeFw0yMjEx
MDIwMTQ0MzBaFw0yMzExMDIwMTQ0MzBaMIGDMQswCQYDVQGEwJGUjEwMBQGA1UE
CwANSWwLIIGRlIEZyYw5jZTEOMAwGA1UEBwwFUGFyaXNkETAPBgNVBAQMCES4cy1M
YWJzMRwFAYDVQQDDA0qLms4cy1sYWIuY291MSEwHwYJKoZIhvcNAQkBFhjtYXdh
a21AazhLWxhYi5jb20wggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCd
rRou053Gp/3V2RPeUtb9TjE13bu1Yjn2G0C0u4InzB0Qc3dFtkEoxV540Zu/QV
Qsu6KF6k2XHsFMxJmXzcLAUBHJGg8dygtYo0bIj0mYkhwUA5cw7NBV+G7TVhngb
aE8GvbxMJsLcbAzhLhUg70EAVDypAoNlHdZZYPaT93wHtZoM/T1Tl01R3WRuLteId
G6bcfr97hIa8cqCwAgXIUqe5KJZpj8kB3gJoRHKSjL0GyeXRiouJpuqW0BA1eSas
wqv4ZxKsYBtX30M95TpBNMbs0KNeZqSECBfWam+PSshJ/HP0rnqYI4f9WqXn8Ip
xRXBT0LckT3Dj7apyTy5AgMBAAGjUzBRMB0GA1UdDgQWBQB9sLUpmUEecB0XL0NM
ZrNtG0121zAFBgNVHSMEGDAwQB9sLUpmUEecB0XL0NMZrNtG0121zAPBgNVHRMB
AfbEBTADAQH/MA0GCSqGSIb3DQEBCwUAA4IBAQCRIUieoBpqsY161W7bU1woLdjQp
EYcgctYE+iiZM8vqt0yx5zYFYiKyAjN4ZcDj/bQw5G4E219NejjGSpwIIFWA9tke
812/Ll08pLBq14nK8dJ7P+QJgVPMUw6g3pv2/GRkvH+Ucutqr9C8GLKUI/1XxRy
/yLZjd/2rTwmNgpyY0aDxws1akKwVXGwbGdle1INV9jgJhUnZzXpNsCxXy+k2Cy
1s8NfSnoSgZmbYSIh6oGuhrJddvkoB7GTH+8uDJComS09qeR2jWmf65vP/sJgXE
lt9DxLCLHozXD2ARwtvzcHPhFFlQJjHtCwsdLX3gt5pCUY0Ds+d8F2t7SjNjv
-----END CERTIFICATE-----
/ #
```

```
apiVersion: v1
kind: Pod
metadata:
  name: client
spec:
  containers:
  - name: client
    image: busybox
    args:
      - "sleep"
      - "3000"

    volumeMounts:
      - mountPath: "/etc/tls/certs"
        name: tls
        readOnly: true

  volumes:
  - name: tls

    secret:
      secretName: k8s-lab-pki-tls
```

# ConfigMap

- Découplage d'une application et de sa configuration
  - ◆ Pas de configuration dans le code applicatif
- Assure la portabilité
- 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

# ConfigMap - Utilisation (fichier)

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: filebeat-config
data:
  filebeat.yml: |-
    # Contenu du fichier
    filebeat.inputs:
    - type: container
      paths:
        - /var/log/containers/*.log
    processors:
    - add_cloud_metadata:
    - add_host_metadata:

cloud.id: ${ELASTIC_CLOUD_ID}
cloud.auth: ${ELASTIC_CLOUD_AUTH}
```

```
apiVersion: v1
kind: Pod
metadata:
  name: filebeat
spec:
  containers:
  - name: filebeat
    image:
      docker.elastic.co/beats/filebeat:8.4.3
    volumeMounts:
    - mountPath: /etc/filebeat.yml
      name: config-file
      subPath: filebeat.yml
      readOnly: true
  volumes:
  - name: config-file
    configMap:
      defaultMode: 0640
      name: filebeat-config
```



# ConfigMap - Utilisation (fichier)

```
> kubectl create -f filebeat-config.yaml
configmap/filebeat-config created
```

```
> kubectl exec -it filebeat -- sh
$ cat /etc/filebeat.yml
# Contenu du fichier
filebeat.inputs:
- type: container
  paths:
    - /var/log/containers/*.log
processors:
- add_cloud_metadata:
- add_host_metadata:

cloud.id: ${ELASTIC_CLOUD_ID}
cloud.auth: ${ELASTIC_CLOUD_AUTH}$
```

# ConfigMap - Utilisation (ENV)

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: myapp-configmap
data:
  ELASTIC_CLOUD_ID: cloud-id:TMzZCRVmN2M==
  ELASTIC_CLOUD_AUTH : admin:passw0rd892
```

```
> kubectl create -f myapp-configmap.yaml
configmap/myapp-variables created
```

```
> kubectl exec -it filebeat -- sh
$ env | grep -i elastic
ELASTIC_CONTAINER=true
ELASTIC_CLOUD_ID=cloud-id:TMzZCRVmN2M==
ELASTIC_CLOUD_AUTH=admin:passw0rd892
$ cat /etc/filebeat.yml
# Contenu du fichier
filebeat.inputs:
- type: container
  paths:
    - /var/log/containers/*.log
processors:
- add_cloud_metadata:
- add_host_metadata:

cloud.id: ${ELASTIC_CLOUD_ID}
cloud.auth: ${ELASTIC_CLOUD_AUTH}$
$
```

```
apiVersion: v1
kind: Pod
metadata:
  name: filebeat
spec:
  containers:
  - name: filebeat
    image:
      docker.elastic.co/beats/filebeat:8.4.3
    env:
      - name: ELASTIC_CLOUD_ID
        valueFrom:
          configMapKeyRef:
            name: myapp-configmap
            key: ELASTIC_CLOUD_ID
      - name: ELASTIC_CLOUD_AUTH
        valueFrom:
          configMapKeyRef:
            name: myapp-configmap
            key: ELASTIC_CLOUD_AUTH
    volumeMounts:
      - mountPath: /etc/filebeat.yml
        name: config-file
        subPath: filebeat.yml
        readOnly: true
    volumes:
      - name: config-file
        configMap:
          defaultMode: 0640
          name: filebeat-config
```

# Ressources Cluster K8s

- Namespace
- Role & Role Binding
- Cluster Role & Cluster Role Binding
- Service Account

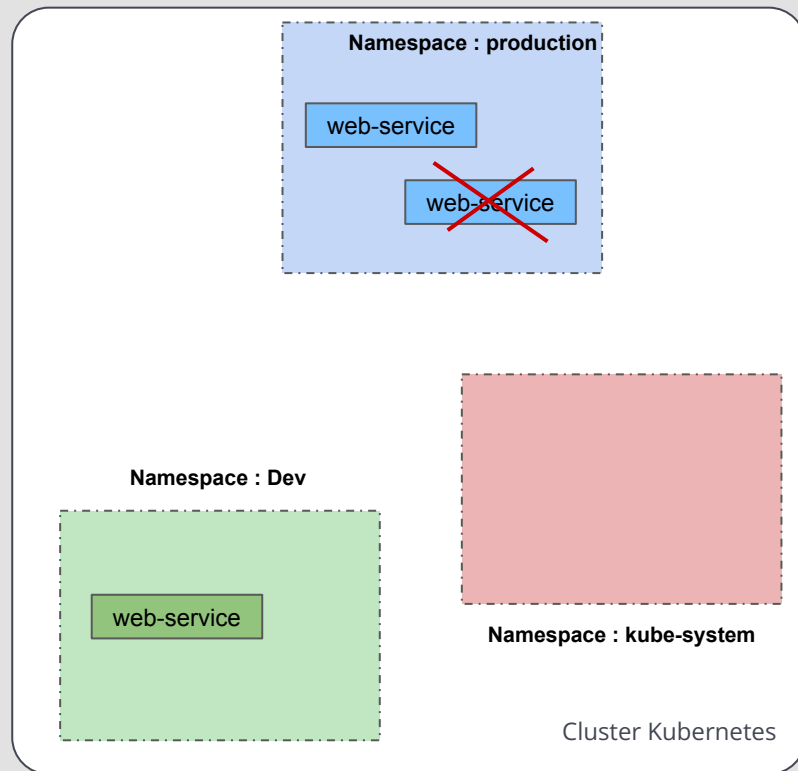
# Namespaces

- Mécanisme permettant d'isoler des groupes de ressources au sein d'un même cluster.
- Les noms des ressources doivent être uniques dans un namespace.
- Applicable que pour les objets dit “**namespaced objects**” vs “**cluster scope objects**”

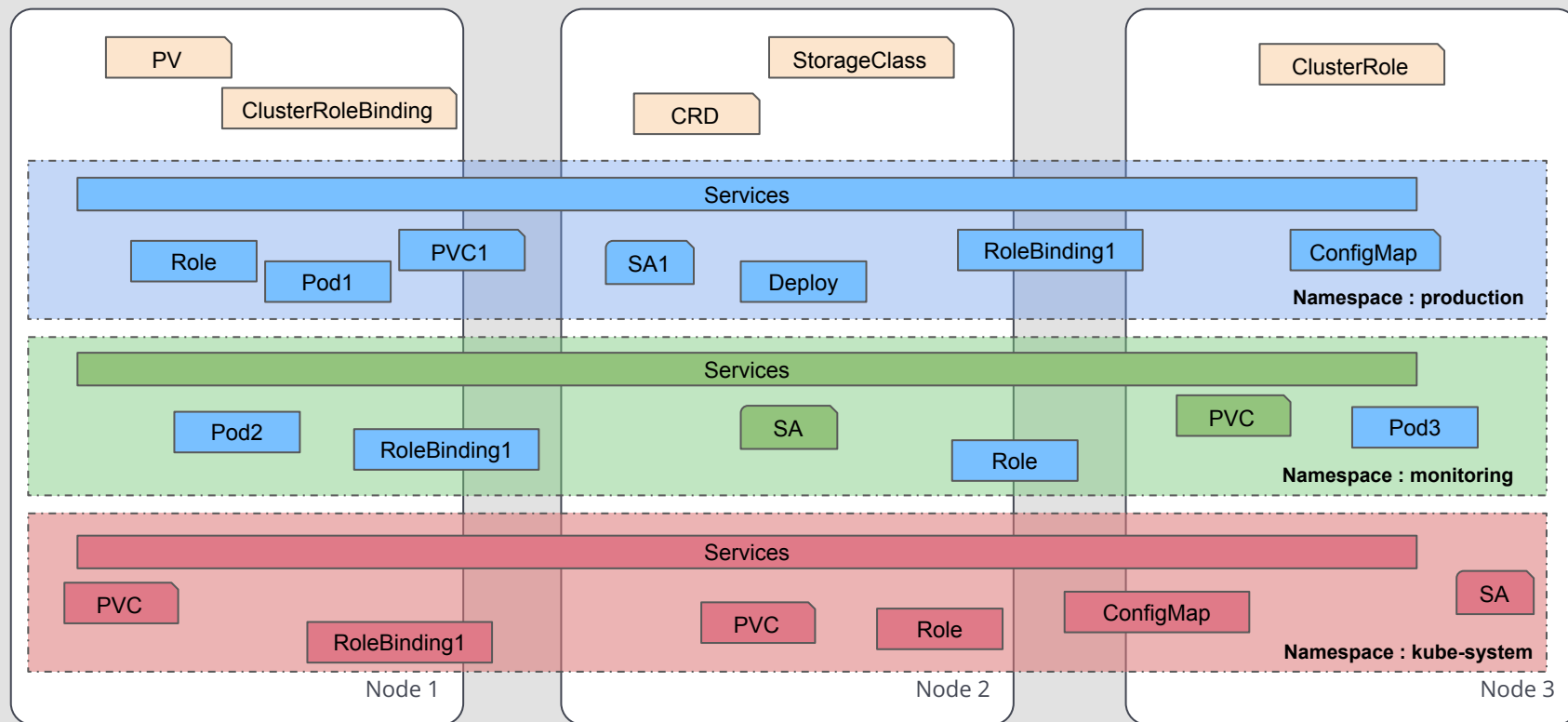
```
$ kubectl api-resources --namespaced=true
```

```
$ kubectl api-resources --namespaced=false
```

```
apiVersion: v1
kind: Namespace
metadata:
  name: production
```



# Namespaces



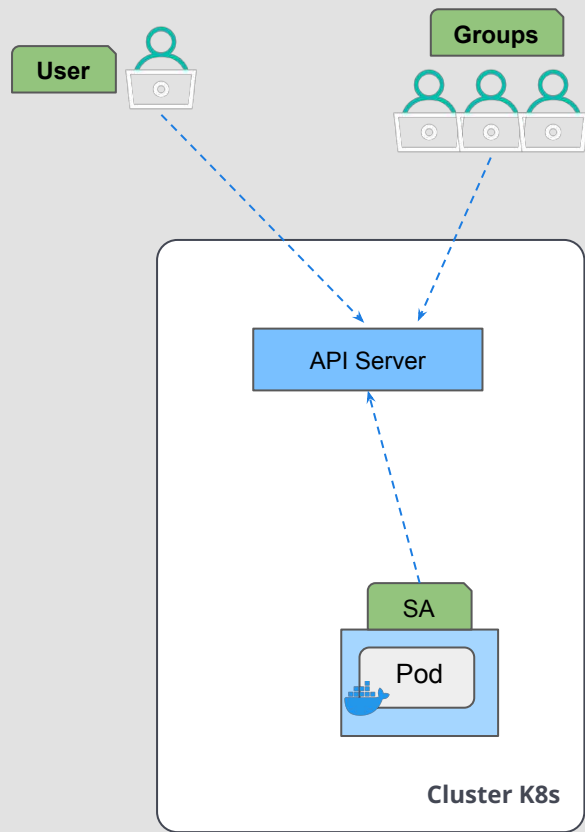
# Namespaces

→ Kubernetes démarre avec quatre espaces de noms initiaux :

- ◆ **Default** : namespace par défaut pour les objets dont namespace n'est pas spécifié.
- ◆ **kube-system** : namespace pour les objets créés par le système Kubernetes : api-server, scheduler...
- ◆ **kube-public** : namespace créé automatiquement et lisible par tout utilisateur (y compris non authentifiés : anonymous).
  - Principalement réservé à l'usage du cluster, dans le cas où certaines ressources doivent être visibles et lisibles publiquement dans l'ensemble du cluster.
  - Seulement une convention, pas une exigence.
- ◆ **Kube-node-lease** : Cet espace de nom contient les objets Lease associés à chaque node.
  - Utile pour Kubelet afin de détecter les nodes défaillants

# Autorisation : RBAC

- Role-Based Access Control
  - ◆ Règle permettant l'accès aux ressources d'un cluster
  - ◆ Appliquées aux Users, Group et ServiceAccount
- 4 types de ressources
  - ◆ Role
  - ◆ RoleBinding
  - ◆ ClusterRole
  - ◆ ClusterRoleBinding



# Service Account

- Utilisé par le process d'un Pod
- Un serviceAccount (SA) « default » pour chaque namespace
- Utilise un Secret pour l'authentification à l'API

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: myapp
automountServiceAccountToken: false
```

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
spec:
  serviceAccountName: myapp
  (...)
```



# Role

- Gestion des droits d'accès aux ressources dans un namespace défini
- Les permissions sont purement additives (il n'y a pas de règles Deny).
- Section Rules ⇒ Définir les privilèges:
  - ◆ **apiGroups** : les groupes API
    - `""` indique API group core "v1"
  - ◆ **Ressources** : La liste des ressources sur lequel les droit s'appliqueront
  - ◆ **Verbs** : Les actions/privilèges accordés

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default # namespace scope
  name: pod-reader
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "watch", "list"]
```

# Role Binding

- Associe les permissions définies dans un Rôle à un utilisateur (User, Group, ServiceAccount)
- Peut référencer un ClusterRole mais sera limité au namespace spécifié
- **Subjects** : list des users
  - ◆ Kind : peut être User, Group, ServiceAccount
- **roleRef** :
  - ◆ Kind : peut être Role, ClusterRole

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: read-pods
  namespace: default
subjects:
- kind: User
  name: jane # "name" is case sensitive
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: pod-reader
  apiGroup: rbac.authorization.k8s.io
```

# ClusterRole

- Gestion des droits d'accès aux ressources du cluster
- Non restreint à un namespace
- Les permissions sont purement additives (il n'y a pas de règles Deny).
- Section Rules ⇒ Définir les privilèges:
  - ◆ **apiGroups** : les groupes API
    - "" indique API group core "v1"
  - ◆ **Ressources** : La liste des ressources sur lesquelles les droits s'appliqueront
    - Peut mettre les ressources comme PVs
  - ◆ **Verbs** : Les actions/privilèges accordés

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  # "namespace" omis ClusterRoles n'est namespaced
  name: secret-reader
rules:
- apiGroups: [""]
  resources: ["secrets"]
  verbs: ["get", "watch", "list"]
```

# ClusterRole Binding

- Associe les permissions définies dans un ClusterRole à un utilisateur (User, Group, ServiceAccount)
- **Subjects** : list des users
  - ◆ Kind : peut être User, Group, ServiceAccount
- **roleRef** :
  - ◆ Le ClusterRole définissant les privilèges

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: read-secrets-global
subjects:
- kind: Group
  name: manager
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: secret-reader
  apiGroup: rbac.authorization.k8s.io
```

# Scheduling sur K8s

- Kube-scheduler
- Taints, labels, annotations
- Affinity & AntiAffinity
- Liveness, Readiness, Startup probes

# Scheduler - nodeName

- **nodeName** : choix explicite du worker node
  - ◆ Bypass les règles d'**affinité** ou le **nodeSelector**

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx
  nodeName: kube-01
```

# Labels

- Des étiquettes au format paires de **clé : valeur** qui sont attachées aux objets. Ex : pods, deployment, nodes,...
- Par défaut des labels sont appliqués aux nodes (**Masters** & **workers**)
- Utilisé pour organiser et sélectionner des sous-ensembles d'objets
- Peut avoir des préfixes sous forme de noms DNS (Ex: app.mydomain.com/tier : frontend)
- Les préfixes **kubernetes.io/** et **k8s.io/** sont réservés aux composants centraux de Kubernetes.

```
> kubectl get nodes --show-labels
```

NAME	STATUS	ROLES	AGE	VERSION	LABELS
k8s-master-1	Ready	control-plane	25m	v1.25.3	beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=k8s-master-1,kubernetes.io/os=linux,node-role.kubernetes.io/control-plane=,node.kubernetes.io/exclude-from-external-load-balancers=
k8s-worker-1	Ready	<none>	23m	v1.25.3	beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=k8s-worker-1,kubernetes.io/os=linux
k8s-worker-2	Ready	<none>	21m	v1.25.3	beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=k8s-worker-2,kubernetes.io/os=linux
k8s-worker-3	Ready	<none>	19m	v1.25.3	beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=k8s-worker-3,kubernetes.io/os=linux

# Scheduler - nodeSelector

- **nodeSelector** : choix explicite du noeud
- ◆ Forme recommandée, la plus simple pour la sélection des nodes
  - ◆ spécifier les **Labels de nœuds** que vous voulez pour le placement des Pods

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    env: test
spec:
  containers:
  - name: nginx
    image: nginx
    imagePullPolicy: IfNotPresent
  nodeSelector:
    instance-size: big
```



# Afinity

→ **requiredDuringSchedulingIgnoredDuringExecution** :

- ◆ Le scheduler ne peut planifier le Pod que si la règle est respectée.
- ◆ Similaire à **nodeSelector**, mais avec une syntaxe plus expressive.

```
apiVersion: v1
kind: Pod
metadata:
  name: with-node-affinity
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
          - matchExpressions:
              - key: topology.kubernetes.io/zone
                operator: In
                values:
                  - us-east1
                  - us-west1
      preferredDuringSchedulingIgnoredDuringExecution:
        - weight: 1
          preference:
            matchExpressions:
              - key: size_of_node
                operator: In
                values:
                  - big
    containers:
      - name: with-node-affinity
        image: registry.k8s.io/pause:2.0
```

# Afinity

→ **preferredDuringSchedulingIgnoredDuringExecution** :

- ◆ Le scheduler essaie de trouver un nœud qui répond à la règle.
- ◆ Si un nœud correspondant n'est pas disponible, l'ordonnanceur planifie quand même le Pod.

```
apiVersion: v1
kind: Pod
metadata:
  name: with-node-affinity
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
          - matchExpressions:
              - key: topology.kubernetes.io/zone
                operator: In
                values:
                  - us-east1
                  - us-west1
            preferredDuringSchedulingIgnoredDuringExecution:
              - weight: 1
                preference:
                  matchExpressions:
                    - key: size_of_node
                      operator: In
                      values:
                        - big
  containers:
    - name: with-node-affinity
      image: registry.k8s.io/pause:2.0
```

# AntiAffinity

- Permet de limiter les nœuds sur lesquels schedulé les Pods en fonction des Labels des **pods**.
- Cette règle **podAntiAffinity** indique au scheduler d'éviter de placer plusieurs réplicas avec l'étiquette **"app : store"** sur un seul nœud.
- **topologyKey**: label aussi attribué aux nodes (par un cloud provider en general)
  - ◆ Ex: empêche que les Pods soient programmés de manière aléatoire dans la même zone

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis-cache
spec:
  selector:
    matchLabels:
      app: store
  replicas: 3
  template:
    metadata:
      labels:
        app: store
    spec:
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            - labelSelector:
                matchExpressions:
                  - key: app
                    operator: In
                    values:
                      - store
              topologyKey: "kubernetes.io/hostname"
      containers:
        - name: redis-server
          image: redis:3.2-alpine
```

# Taints & Tolerations

- **Affinity** : propriété des pods attirant vers un ensemble de nodes VS **Taints** : Permettent à un nodes de repousser un ensemble de Pods.
- **Tolerations**: Appliquées aux pods; Permettent au Scheduler de bypasser les Taints
  - ◆ Scheduler peut programmer des pods sur des nodes avec des taints correspondants.

```
$ kubectl taint nodes node1 key1=value1:NoSchedule
```

- Ce taint appliqué a la clé : **key1**, la valeur : **value1**, et l'effet du taint **NoSchedule**.
- Signifie = aucun pod ne pourra se programmer sur node1 à moins qu'il n'ait une **tolérance** correspondante.

```
$ kubectl taint nodes node1 key1=value1:NoSchedule-
```

→ Retirer le taint

# Taints & Tolerations

→ Tolérance pour un pod est spécifié dans **PodSpec**.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    env: test
spec:
  containers:
  - name: nginx
    image: nginx
    imagePullPolicy: IfNotPresent
  tolerations:
  - key: "key1"
    operator: "Equal"
    value: "value1"
    effect: "NoSchedule"
```

# Liveness probes

## → livenessProbe

- ◆ Utilisé par Kubelet pour savoir quand démarrer un conteneur.
- ◆ `periodSeconds` : spécifie l'intervalle de temps de vérification de kubelet (Ex: chaque 5 secondes)

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

# Liveness probes

## → livenessProbe

- ◆ Verification HTTP
- ◆ Ex: si le path /healthz retourne un code de succès alors kubelet considère le conteneur UP et Healthy

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    test: liveness
  name: liveness-http
spec:
  containers:
    - name: liveness
      image: registry.k8s.io/liveness
      args:
        - /server
      livenessProbe:
        httpGet:
          path: /healthz
          port: 8080
          httpHeaders:
            - name: Custom-Header
              value: Awesome
          initialDelaySeconds: 3
          periodSeconds: 3
```

# Readiness probes

## → readinessProbe

- ◆ Utilisé par Kubelet pour savoir quand un conteneur est prêt à recevoir des requêtes.
- ◆ periodSeconds : spécifie l'intervalle de temps de vérification de kubelet (Ex: chaque 5 secondes)

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



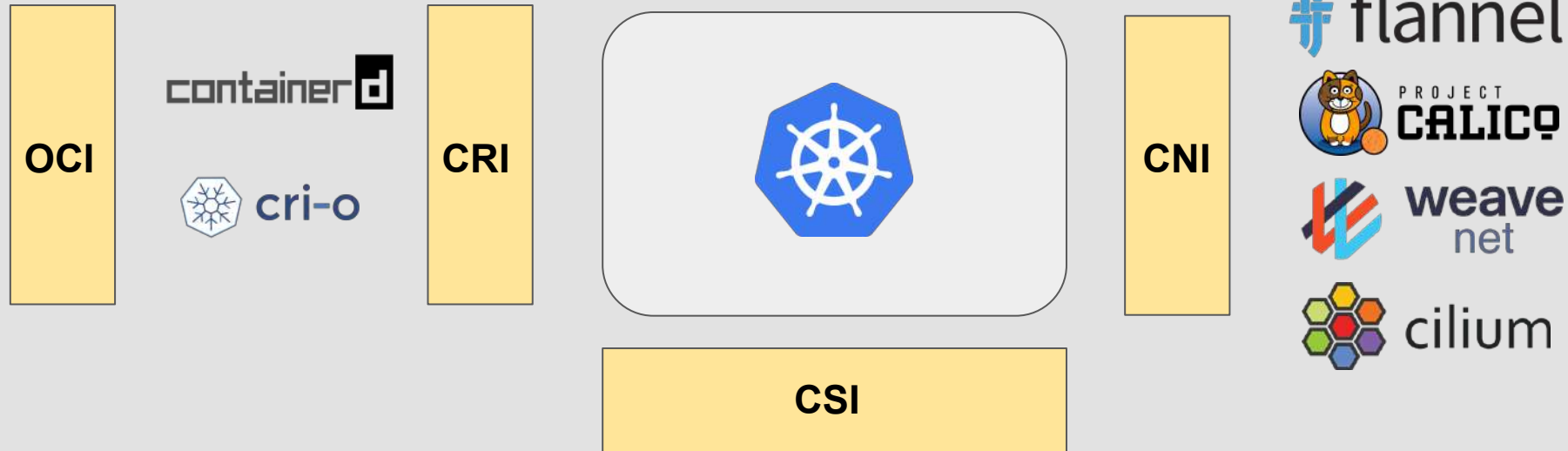
# Administration d'un Cluster K8s LAB - TP

- Kubeadm: Installation K8s
- Maintenance des noeuds

# Spécifications autour des conteneurs

- OCI
- CRI
- CNI
- CSI

# OCI - CRI - CNI - CSI



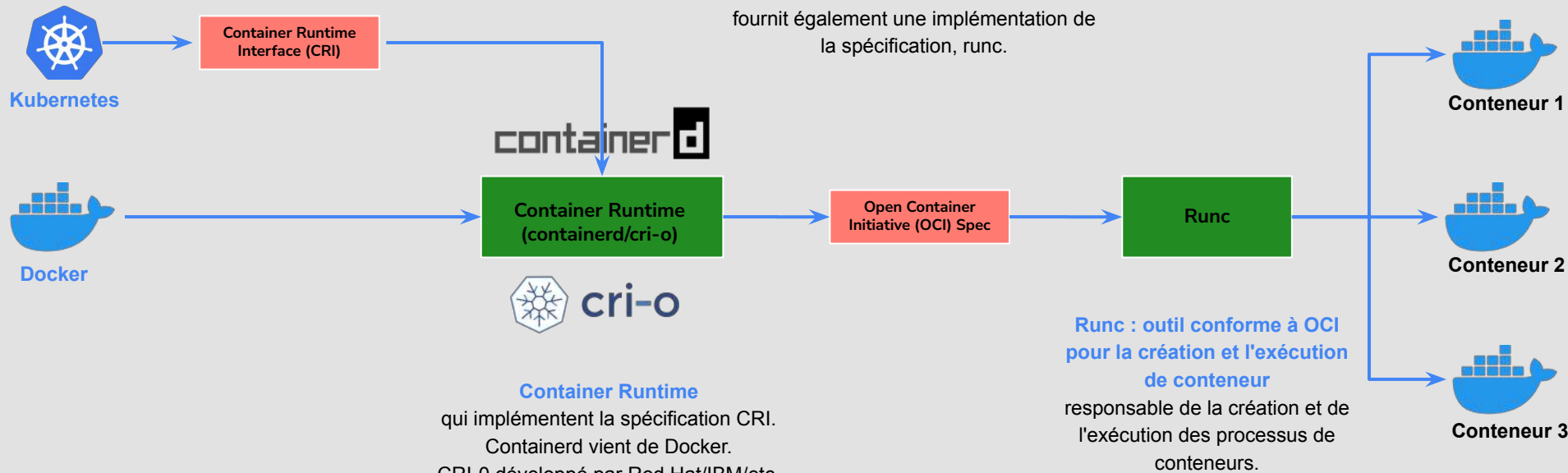
# CRI & OCI

## CRI est une API Kubernetes:

Définit la manière dont K8s  
interagit avec différents Container  
Runtime (Ex : containerd, cri-o, rkt...)

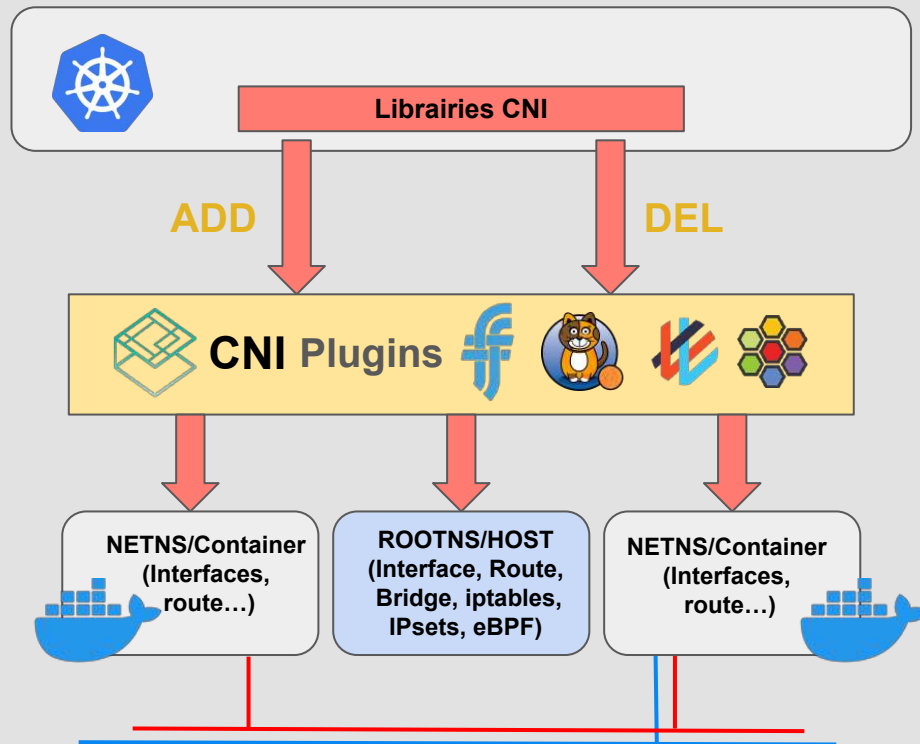
## OCI est une spécification pour les images de conteneurs et l'exécution des conteneurs

fournit également une implémentation de  
la spécification, runc.



# CNI

- Framework réseau qui permet la configuration dynamique des ressources de réseau
- Définit un protocole permettant aux Containers Runtimes d'émettre des requêtes aux plugins de réseau: Add, Del...
- Définit des types de données permettant aux plugins de renvoyer leurs résultats au runtime.
- Intégration transparente à Kubelet pour la [config automatique](#) du réseau entre Pods à l'aide d'un réseau underlay ou overlay.
- Offre des fonctionnalités non natives à K8s :
  - ◆ Isolation des Net namespace
  - ◆ Trafic inter Pods, Pods à nodes
  - ◆ Filtrage IP, Firewall, observability...



## → Flannel

- Initiative de CoreOS
- L'un des plugins les plus populaires
- Fournit une solution de tunneling VXLAN
- La configuration et la gestion sont très simples
- Manque de fonctionnalités avancées: ne prend pas en charge **politiques de réseau, Firewall**
- Dispose d'un mode appelé host-gw qui fournit une solution sans tunnel tant que les hôtes sont connectés avec une connectivité directe de couche 2.



## → Calico

- L'un des plugins les plus populaires
- Choix par défaut de la plupart des plateformes Kubernetes (kubespray, docker enterprise, etc.)
- Utilise BGP (vs VXLAN) et Bird, un démon appelé Felix configure les routes sur Bird
- Supporte l'encapsulation IP-IP si BGP ne peut être utilisé
- Supporte les **politiques de réseau**
- Utilise iptables pour le routage mais il peut être configuré pour utiliser le mode IPVS de kube-proxy
- Possède un outil CLI appelé calicoctl
- S'associe bien à des services de maillage (Service mesh) comme Istio: service mesh populaire de Kubernetes

## → Weave

- Fournit une solution de tunneling VXLAN
- Tous les nœuds sont connectés sous forme de maillage, ce qui lui permet de fonctionner sur des réseaux partiellement connectés
- N'est pas très évolutif en raison de la structure maillée
- Stocke les fichiers de configuration sur des pods au lieu de CRDs Kubernetes ou etcd
- Dispose d'une bibliothèque de cryptage (cryptage trafic)
- Supporte les politiques de réseau
- Dispose d'un outil CLI appelé weave





## → Cilium

- Fournit une solution de tunneling VXLAN mais il peut être utilisé avec **kube-router** pour fournir une solution sans tunneling
- Utilise eBPF et XDP (eXpress Data Path) pour le routage
- Prend en charge les stratégies de réseau
- Dispose également de Cilium Network Policies qui fournit des fonctionnalités niveau 4-7 OSI :
  - Pas encore prises en charge par les Network Policies de Kubernetes
  - Ex: filtres de requête HTTP, filtres DNS...
- Dispose d'un outil CLI appelé cilium
- Le noyau Linux doit être au moins 4.9.

**→ Kube Router**

- Utilise BGP
- Utilise IPVS pour le routage
- Plus simple et plus petit que Calico (un seul Daemonset vs Felix)
- Supporte les politiques de réseau

# CNI - Performance banchmark

CNI Benchmark August 2020 InfraBuilder	Config	Performances (bandwidth)				Resources consumption (cpu/ram)					Security features			
	MTU	Pod to Pod		Pod to Service		Idle	Pod to Pod		Pod to Service		Network Policies		Encryption	
	setting	TCP	UDP	TCP	UDP	none	TCP	UDP	TCP	UDP	in	out	activation	Performance
Antrea	auto	Very fast	Very fast	Very fast	Slow	Low	Low	Low	Low	Low	yes	yes	at deploy time	Slow
Calico	manual	Very fast	Very fast	Very fast	Fast	Low	Very low	Very low	Very low	Very low	yes	yes	anytime	Very fast
Canal	manual	Very fast	Very fast	Very fast	Very fast	Low	Very low	Very low	Very low	Very low	yes	yes	no	n/a
Cilium	auto	Fast	Very fast	Very fast	Very fast	High	High	High	High	High	yes	yes	at deploy time	Slow
Flannel	auto	Very fast	Very fast	Very fast	Very fast	Very low	Very low	Very low	Very low	Very low	no	no	no	n/a
Kube-OVN	auto	Fast	Very slow	Fast	Very slow	High	High	High	High	High	yes	yes	no	n/a
Kube-router	none	Slow	Very slow	Slow	Very slow	Low	Very low	Low	Very low	Low	yes	yes	no	n/a
Weave Net	manual	Very fast	Very fast	Very fast	Fast	Very low	Low	Low	Low	Low	yes	yes	at deploy time	Slow

<https://itnext.io/benchmark-results-of-kubernetes-network-plugins-cni-over-10gbit-s-network-updated-august-2020-6e1b757b9e49>

# CSI

→ Framework

