Kubernetes

manage application, not machines

N. Salleron B. Affes

Lundi 12 Février 2018

SCIENCES SORBONNE UNIVERSITÉ

Sommaire

SCIENCES SORBONNE UNIVERSITÉ

Historique

Une longue émergence

- Borg
  - Démarrage en 2004.
  - Développé en interne.
  - Manager de containers.
  - Objectif : réduction des coups en partageant machines et applications.
  - **Inconvéniant : notion de travail, gestion des ports**
  - **Non open-source.**

SCIENCES SORBONNE UNIVERSITÉ

Historique

Une longue émergence

- Borg
  - Démarrage en 2004.
  - Développé en interne.
  - Manager de containers.
  - Objectif : réduction des coups en partageant machines et applications.
  - **Inconvéniant : notion de travail, gestion des ports**
  - **Non open-source.**
- Omega
  - Fils de Borg.
  - Amélioration de l'écosystème apporté par Borg.
  - **Non open-source.**

SCIENCES SORBONNE UNIVERSITÉ

Historique

Une longue émergence

- Borg
  - Démarrage en 2004.
  - Développé en interne.
  - Manager de containers.
  - Objectif : réduction des coups en partageant machines et applications.
  - **Inconvéniant : notion de travail, gestion des ports**
  - **Non open-source.**
- Omega
  - Fils de Borg.
  - Amélioration de l'écosystème apporté par Borg.
  - **Non open-source.**
- Kubernetes
  - Adaptable à plusieurs infrastructure cloud.
  - **Open-source.**

SCIENCES SORBONNE UNIVERSITÉ

Introduction

Nom venant du Grec, crée par 3 ingénieurs de chez Google en 2014.

- *Orchestrateur* - Gestionnaire de conteneur.
- Exécute et manages des containers.
- Propose une API permettant la gestion de plusieurs clouds (Google, Microsoft, Amazon, et pleins d'autres).
- 100% Open Source écrit en Go.

FIGURE – Logo de Kubernetes

Il permet de se focus sur les applications et non sur le déploiement. Google exécute 2 milliards de conteneurs par semaine avec ces systèmes.

SCIENCES SORBONNE UNIVERSITÉ

Introduction

Nom venant du Grec, crée par 3 ingénieurs de chez Google en 2014.

- *Orchestrateur* - Gestionnaire de conteneur.
- Exécute et manages des containers.
- Propose une API permettant la gestion de plusieurs clouds (Google, Microsoft, Amazon, et pleins d'autres).
- 100% Open Source écrit en Go.

FIGURE – Logo de Kubernetes

Il permet de se focus sur les applications et non sur le déploiement. Google exécute 2 milliards de conteneurs par semaine avec ces systèmes.

Dernière version : 1.9.3 (sortie il y a 3 jours)

SCIENCES SORBONNE UNIVERSITÉ

Introduction

Nom venant du Grec, crée par 3 ingénieurs de chez Google en 2014.

- *Orchestrateur* - Gestionnaire de conteneur.
- Exécute et manages des containers.
- Propose une API permettant la gestion de plusieurs clouds (Google, Microsoft, Amazon, et pleins d'autres).
- 100% Open Source écrit en Go.

FIGURE – Logo de Kubernetes

Il permet de se focus sur les applications et non sur le déploiement. Google exécute 2 milliards de conteneurs par semaine avec ces systèmes.

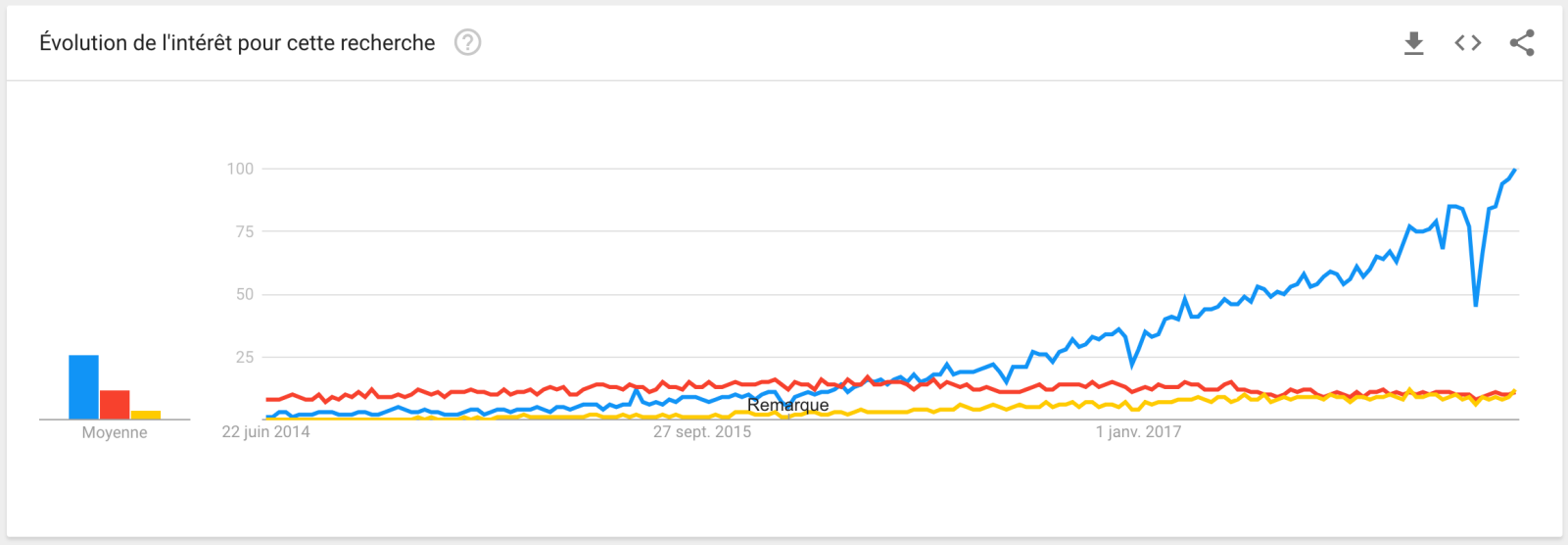
Dernière version : 1.9.3 (sortie il y a 3 jours)

"manage application, not machines" - Tim Hockin

SCIENCES SORBONNE UNIVERSITÉ

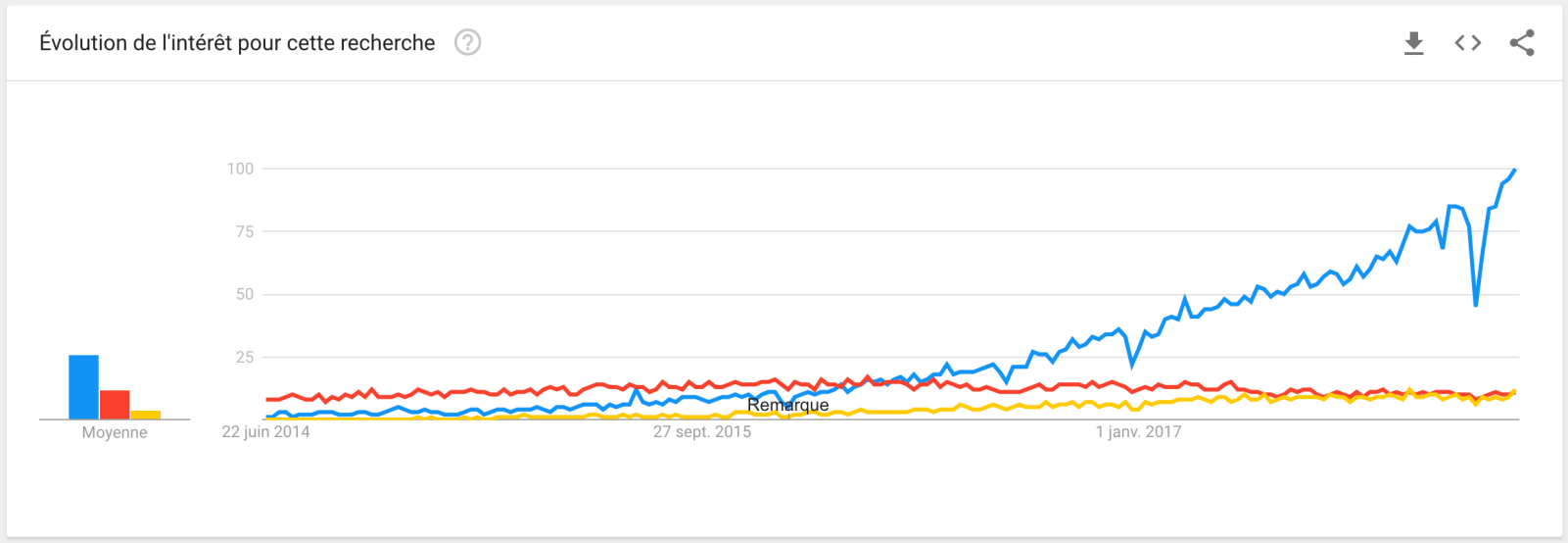
Popularité

Évolution des recherches entre Kubernetes, Mesos, Docker Swarm



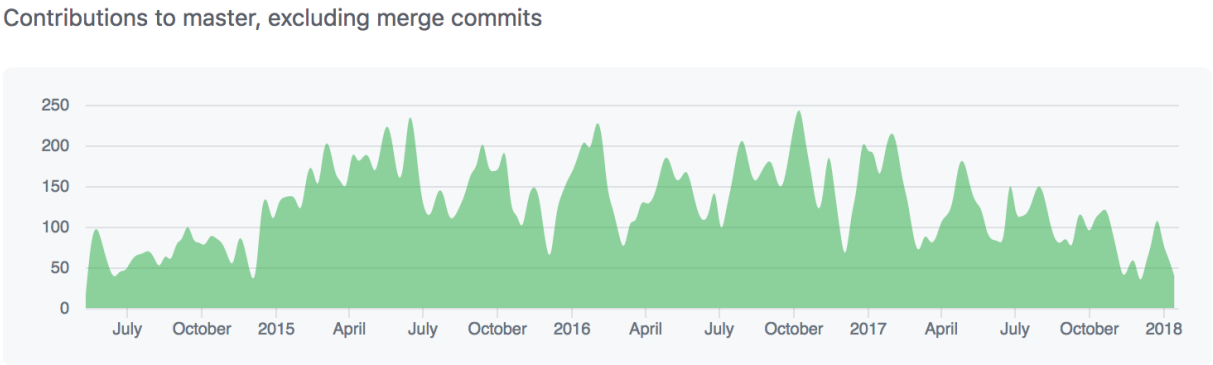
Popularité

Évolution des recherches entre Kubernetes, Mesos, Docker Swarm



Une communauté très active :

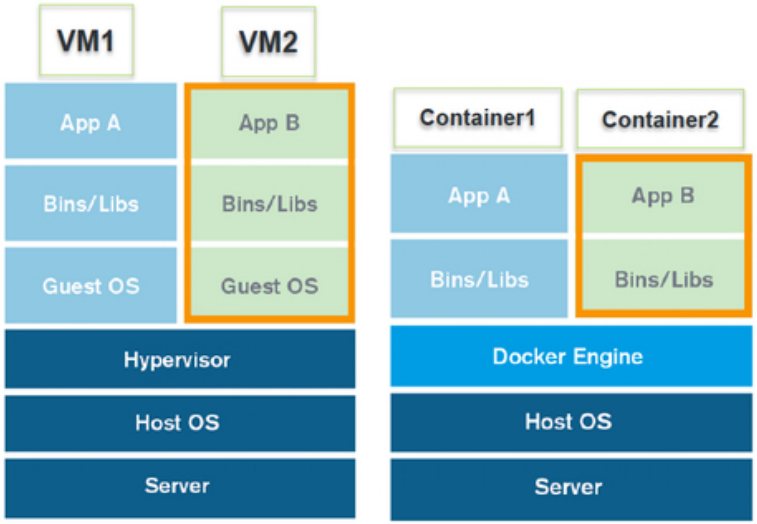
- Actuellement 61000 commits avec plus de 1500 contributeurs



SCIENCES SORBONNE UNIVERSITÉ


docker

Docker est un conteneur léger, permettant de l'isolation entre les processus.



- Retire le coût de la virtualisation (pas de gestion hardware)
- Retire le coût d'exécution de plusieurs OS.

SCIENCES SORBONNE UNIVERSITÉ

Docker se base sur deux technologies du noyau :

- CGroups
- Namespace

Docker se base sur deux technologies du noyau :

- CGroups
- Namespace

Control Groups

Feature kernel qui permet de contrôler, limité et isoler l'usage des ressources pour un processus ou une collection de processus.

SCIENCES SORBONNE UNIVERSITÉ

Docker se base sur deux technologies du noyau :

- CGroups
- Namespace

Control Groups

Feature kernel qui permet de contrôler, limité et isoler l'usage des ressources pour un processus ou une collection de processus.

CGroups Isolation

- Quantitative Isolation : Les CGroups ne peuvent pas avoir plus de pages que la limite imposé.
- Qualitative Isolation : Les CGroups doivent accéder à leur mémoire comme si elles étaient seules sur la machine.

SCIENCES SORBONNE UNIVERSITÉ

Docker se base sur deux technologies du noyau :

- CGroups
- Namespace

Control Groups

Feature kernel qui permet de contrôler, limité et isoler l'usage des ressources pour un processus ou une collection de processus.

CGroups Isolation

- Quantitative Isolation : Les CGroups ne peuvent pas avoir plus de pages que la limite imposé.
- Qualitative Isolation : Les CGroups doivent accéder à leur mémoire comme si elles étaient seules sur la machine.

Namespace

Feature linux qui permet de créer une vue local pour les ressources d'un systèmes. Les ressources en dehors du namespace ne sont pas visible.

Kubernetes

SCIENCES SORBONNE UNIVERSITÉ

Pods

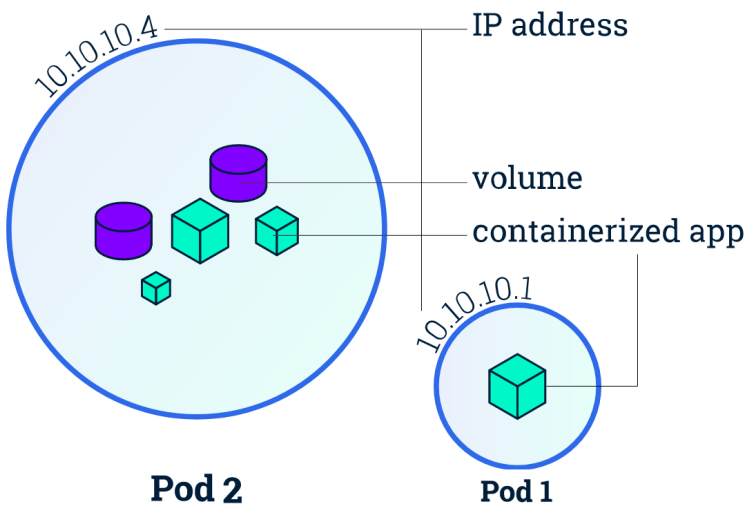


FIGURE – Les Pod dans Kubernetes

Caractéristiques du Pod

- Unité de base de l'ordonnancement.

SCIENCES SORBONNE UNIVERSITÉ

Pods

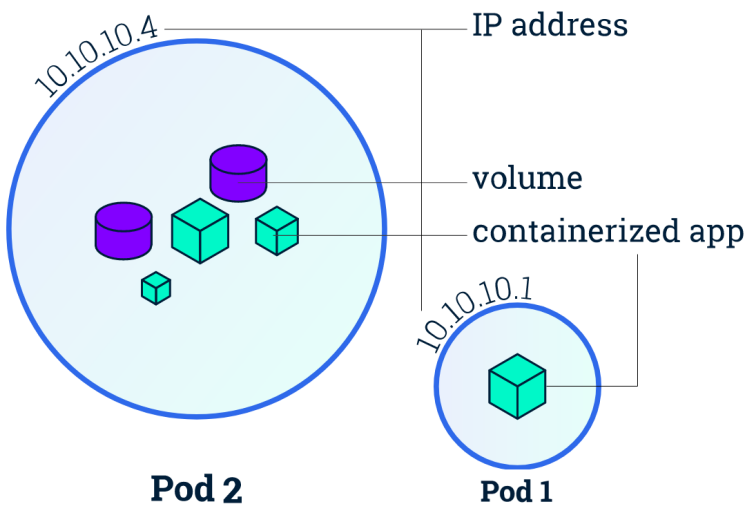


FIGURE – Les Pod dans Kubernetes

Caractéristiques du Pod

- Unité de base de l'ordonnancement.
- Vue abstraite de composants conteneurisés.

SCIENCES SORBONNE UNIVERSITÉ

Pods

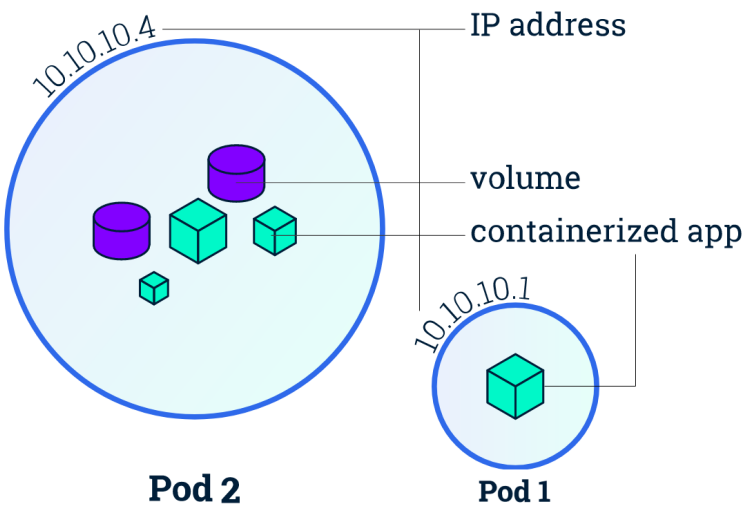


FIGURE – Les Pod dans Kubernetes

Caractéristiques du Pod

- Unité de base de l'ordonnancement.
- Vue abstraite de composants conteneurisés.
- Il peut regrouper 1 ou * conteneurs.
  => Couplage fort.

SCIENCES SORBONNE UNIVERSITÉ

Pods

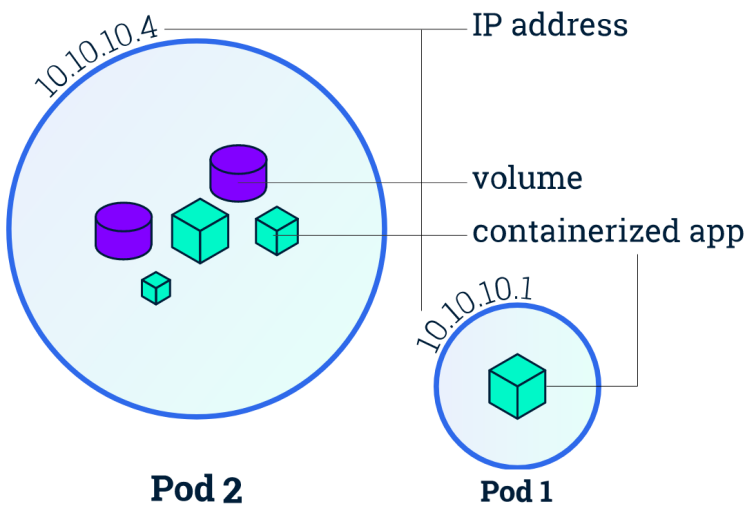


FIGURE – Les Pod dans Kubernetes

Caractéristiques du Pod

- Unité de base de l'ordonnancement.
- Vue abstraite de composants conteneurisés.
- Il peut regrouper 1 ou * conteneurs. => Couplage fort.
- Chaque pod possède une adresse IP unique (limité au cluster).

SCIENCES SORBONNE UNIVERSITÉ

Pods

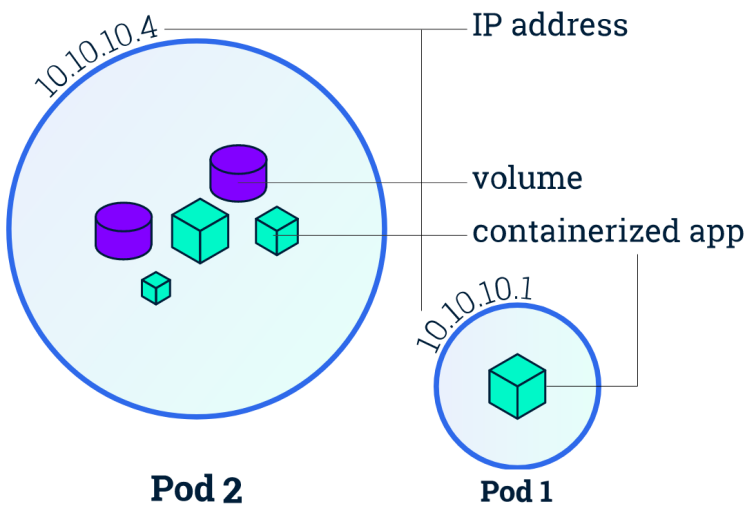


FIGURE – Les Pod dans Kubernetes

Caractéristiques du Pod

- Unité de base de l'ordonnancement.
- Vue abstraite de composants conteneurisés.
- Il peut regrouper 1 ou * conteneurs. => Couplage fort.
- Chaque pod possède une adresse IP unique (limité au cluster).
- Un Pod peut définir un volume. Il a la même durée de vie que le Pod.

SCIENCES SORBONNE UNIVERSITÉ

Pods

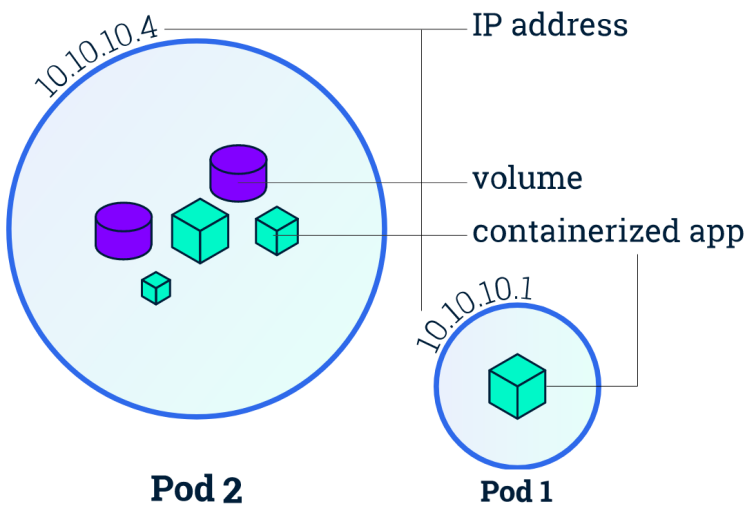


FIGURE – Les Pod dans Kubernetes

Caractéristiques du Pod

- Unité de base de l'ordonnancement.
- Vue abstraite de composants conteneurisés.
- Il peut regrouper 1 ou * conteneurs.
  => Couplage fort.
- Chaque pod possède une adresse IP unique (limité au cluster).
- Un Pod peut définir un volume. Il a la même durée de vie que le Pod.

Bénéfices du pod :

- Plusieurs conteneurs dans 1 Pod
  => Processus qui ont besoin d'interroger un autre processus avec une faible latence.

SCIENCES SORBONNE UNIVERSITÉ

Pods

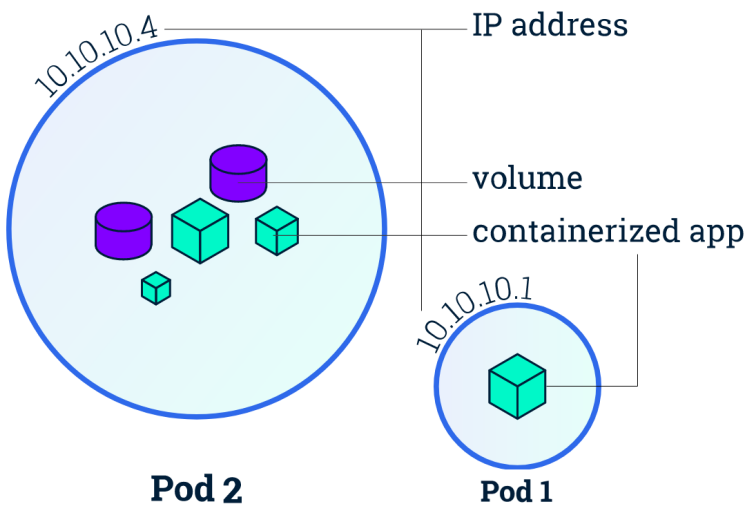


FIGURE – Les Pod dans Kubernetes

Caractéristiques du Pod

- Unité de base de l'ordonnancement.
- Vue abstraite de composants conteneurisés.
- Il peut regrouper 1 ou * conteneurs.
  => Couplage fort.
- Chaque pod possède une adresse IP unique (limité au cluster).
- Un Pod peut définir un volume. Il a la même durée de vie que le Pod.

Bénéfices du pod :

- Plusieurs conteneurs dans 1 Pod
  => Processus qui ont besoin d'interroger un autre processus avec une faible latence.
- Utilisable sous plusieurs environnements (fichier de configuration indépendant de la plateforme)

SCIENCES SORBONNE UNIVERSITÉ

Pods

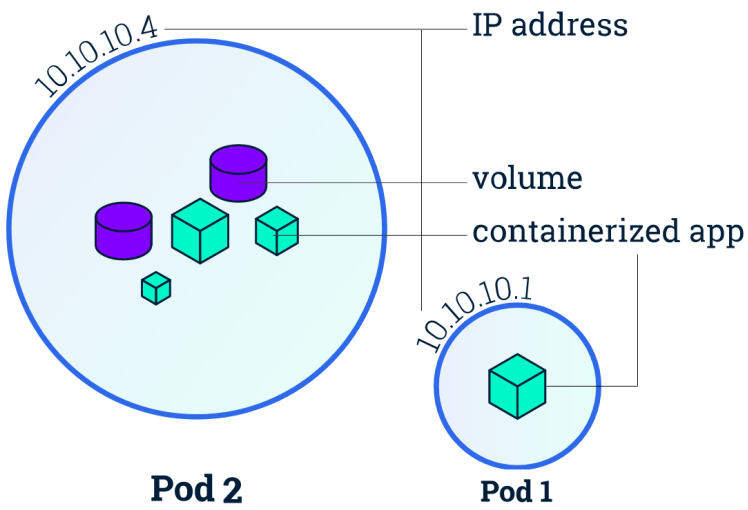


FIGURE – Les Pod dans Kubernetes

Caractéristiques du Pod

- Unité de base de l'ordonnancement.
- Vue abstraite de composants conteneurisés.
- Il peut regrouper 1 ou * conteneurs.
  => Couplage fort.
- Chaque pod possède une adresse IP unique (limité au cluster).
- Un Pod peut définir un volume. Il a la même durée de vie que le Pod.

Bénéfices du pod :

- Plusieurs conteneurs dans 1 Pod
  => Processus qui ont besoin d'interroger un autre processus avec une faible latence.
- Utilisable sous plusieurs environnements (fichier de configuration indépendant de la plateforme)
- Mortel : un container peut mourir.

SCIENCES SORBONNE UNIVERSITÉ

Label et Selector

Label

- Méta-données arbitraire attaché à un objet.
- Forme (K :V)
- Représente généralement une identité.

Selector

- Permet de sélectionner plusieurs objets.
- API supporte deux types de selector :
  - equality-based : "==", "=", " !="
  - set-based : "in", "notin", "exists"

```
"labels": {
  "key1" : "value1",
  "key2" : "value2"
}
```

FIGURE – Exemple K :V format JSON

```
environment = production
tier != frontend
```

FIGURE – Selector *"equality-based"*

```
environment in (production, qa)
tier notin (frontend, backend)
```

FIGURE – Selector *"set-based"*

SCIENCES SORBONNE UNIVERSITÉ

Exemple d'utilisation

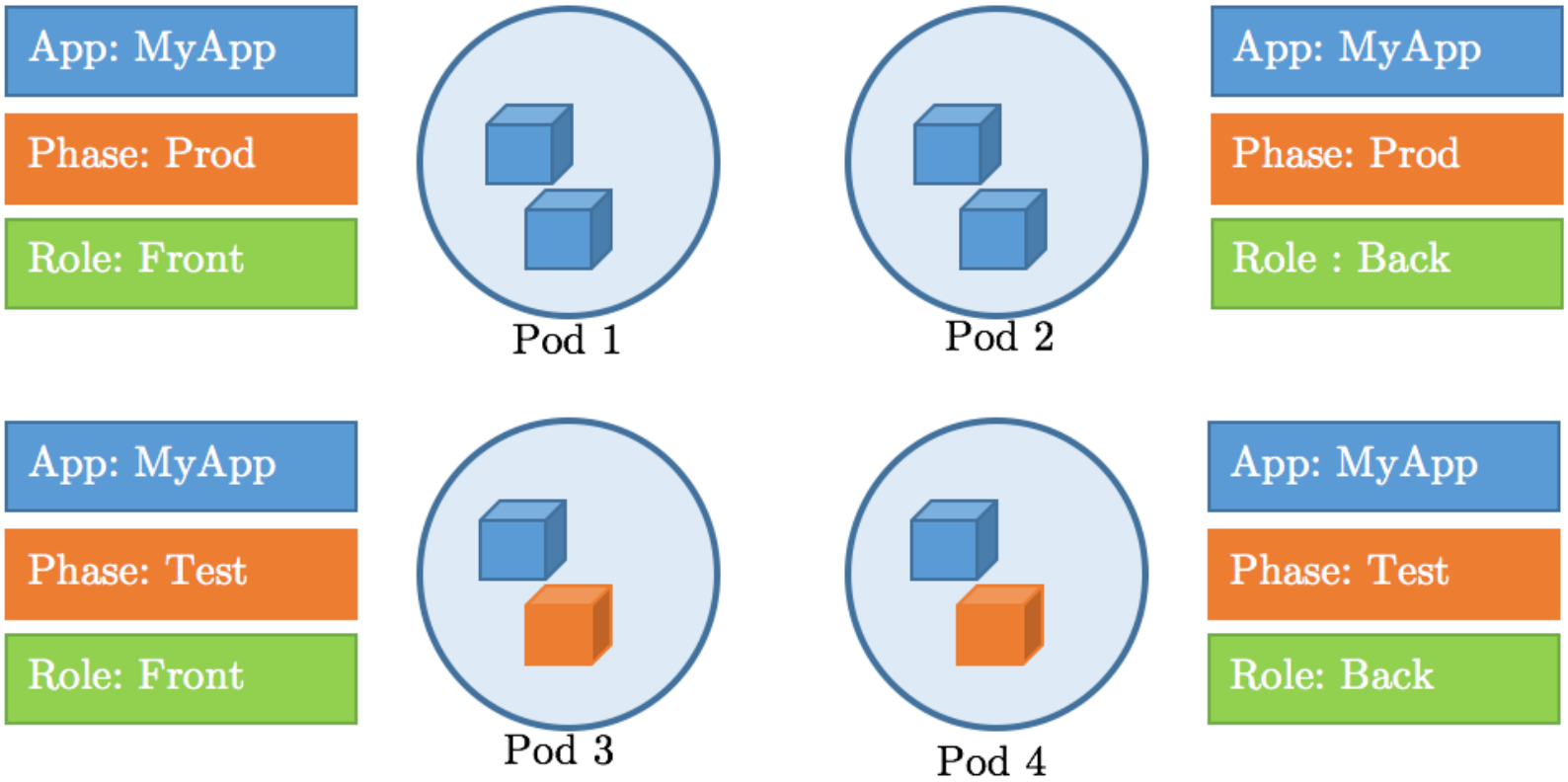


FIGURE – Exemple avec différents selectors

Exemple d'utilisation

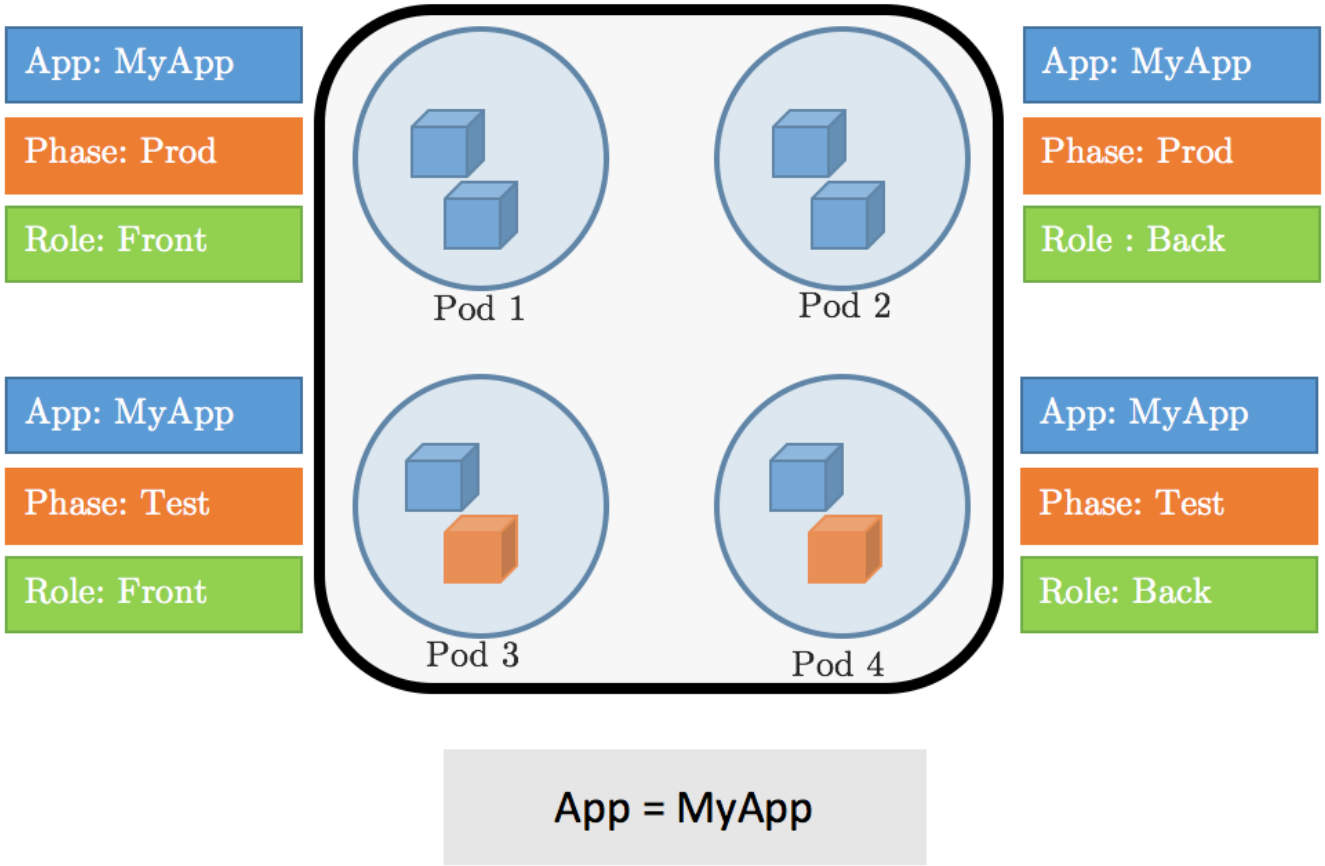


FIGURE – Exemple avec différents selectors

Exemple d'utilisation

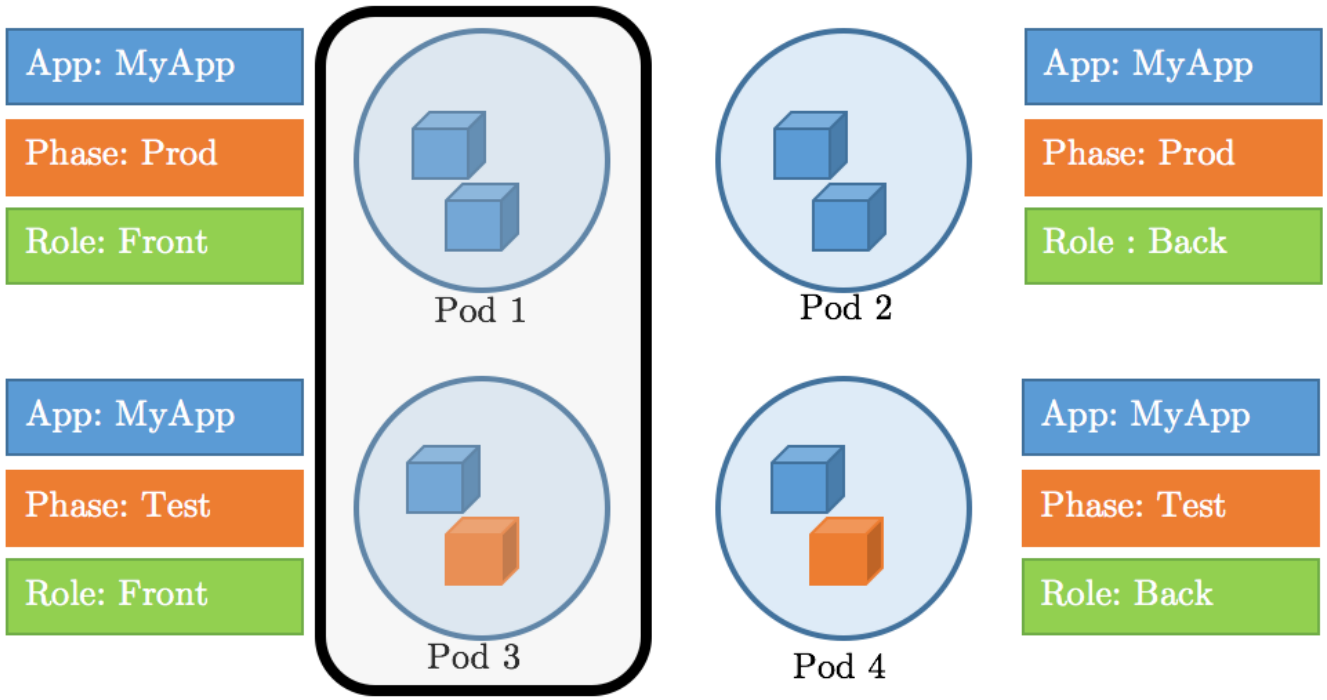


FIGURE – Exemple avec différents selectors

Exemple d'utilisation

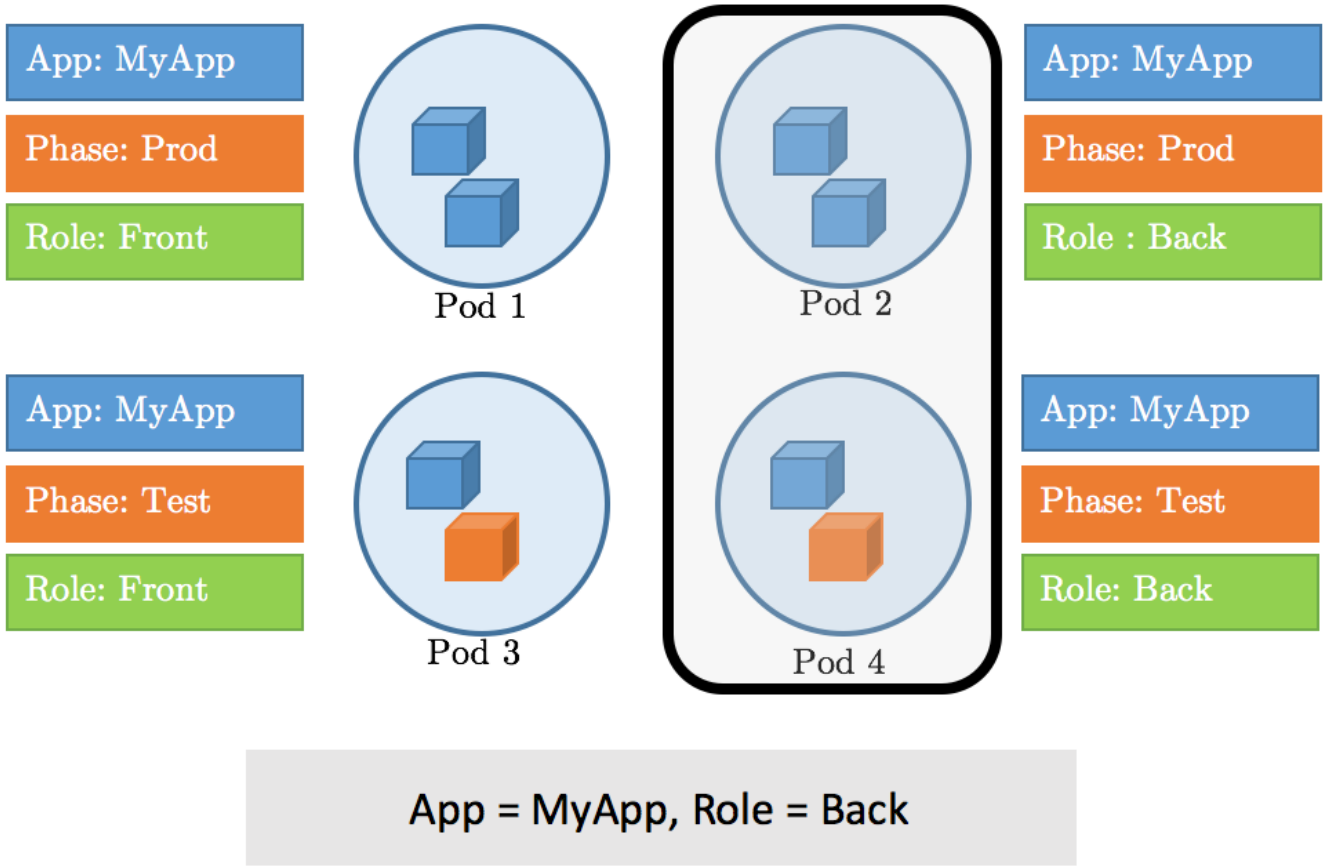


FIGURE – Exemple avec différents selectors

Exemple d'utilisation

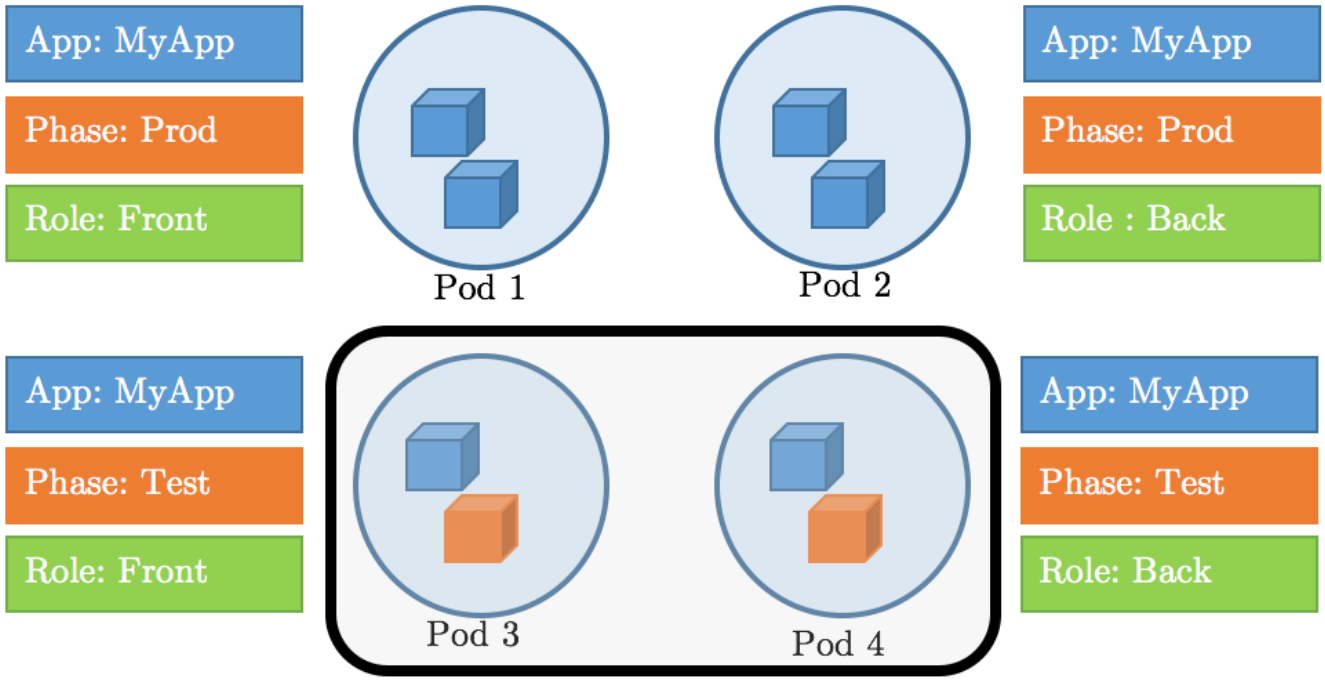


FIGURE – Exemple avec différents selectors

SCIENCES SORBONNE UNIVERSITÉ

Les ReplicatSet

Objectifs

- Gère l'unité basique dans Kubernete, le Pod.
- Il s'assure que le nombre de Pod voulu est présent.
  - Groupe les Pods via des Selectors
  - Si n < LIMIT : start Pod
  - Si n > LIMIT : kill Pod
- Les Pods répliqués n'ont pas d'identité propre.
- Ce sont des consommables.
- Peut-être utilisé pour une mise à l'échelle horizontale automatisé.

```yaml
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: frontend-scaler
spec:
  scaleTargetRef:
    kind: ReplicaSet
    name: frontend
  minReplicas: 3
  maxReplicas: 10
  targetCPUUtilizationPercentage: 50
```

FIGURE – Mise à l'échelle horizontale automatique

Attention

Les ReplicatSet sont déconseiller pour faire de la mise à jour continue.
=> A utiliser seulement pour des applications n'ayant pas besoin de mise à jour

Deployment et Service

Deployment

- Possède et gère 1 ou plusieurs Replica Sets.
- Permet la mise à jour continue avec 3 paramètres :
  - minReadySeconds
  - maxSurge
  - maxUnavailable
- Permet également le rollback après deployment.

Service

- Une abstraction d'un ensemble logique de pod et d'une politique d'accès aux Pods.
- Adapté aux micro-services
- Load-balancing

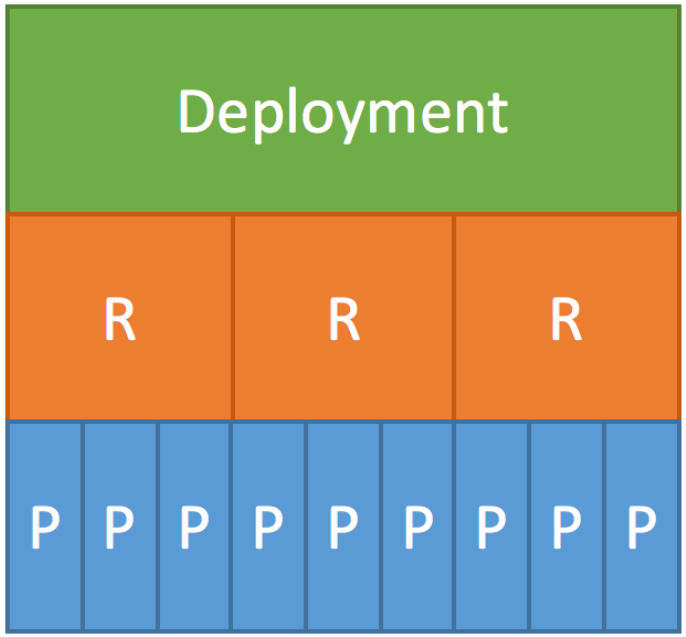


FIGURE – Relation entre Objet Deployment, ReplicatSet et Pod

SCIENCES SORBONNE UNIVERSITÉ

Exemple du Rolling-Update

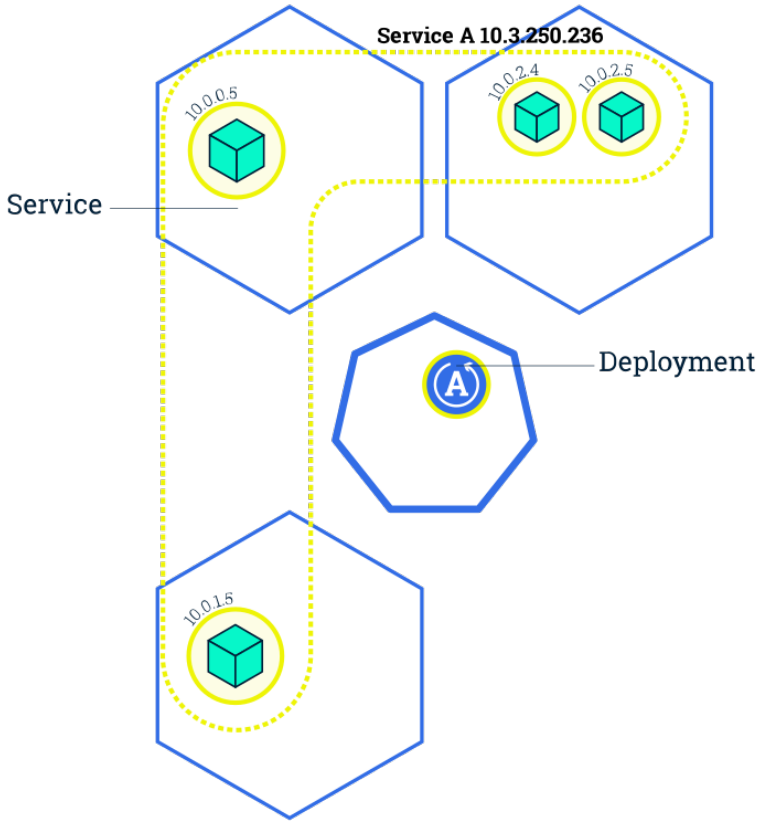


FIGURE – Exemple du Rolling Update tiré de la documentation Kubernetes

SCIENCES SORBONNE UNIVERSITÉ

Exemple du Rolling-Update

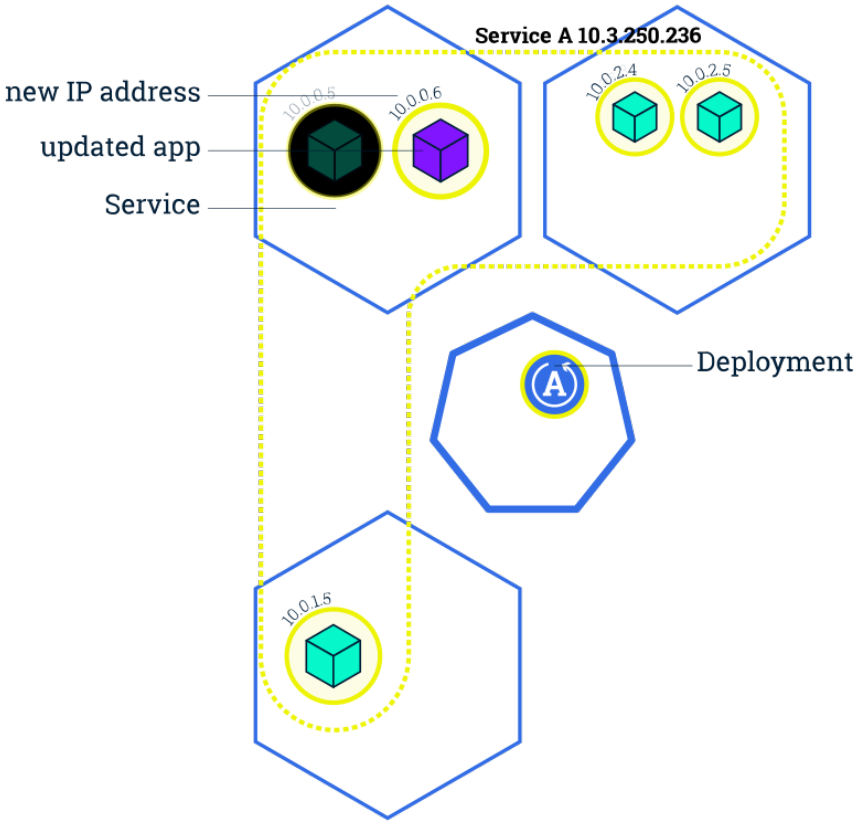


FIGURE – Exemple du Rolling Update tiré de la documentation Kubernetes

SCIENCES SORBONNE UNIVERSITÉ

Exemple du Rolling-Update

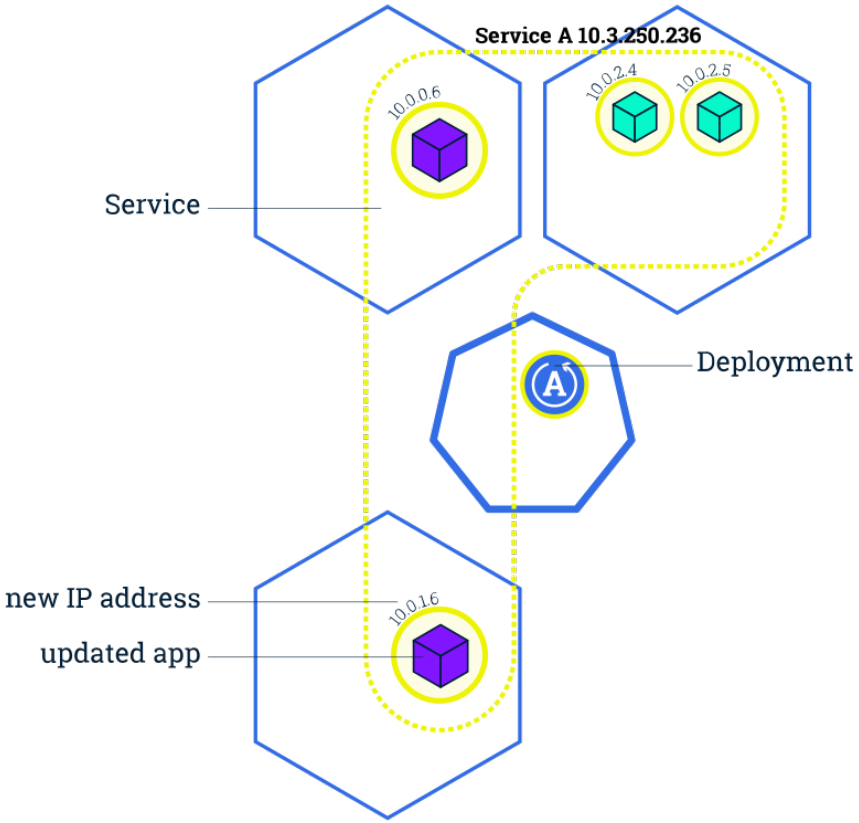


FIGURE – Exemple du Rolling Update tiré de la documentation Kubernetes

SCIENCES SORBONNE UNIVERSITÉ

Exemple du Rolling-Update

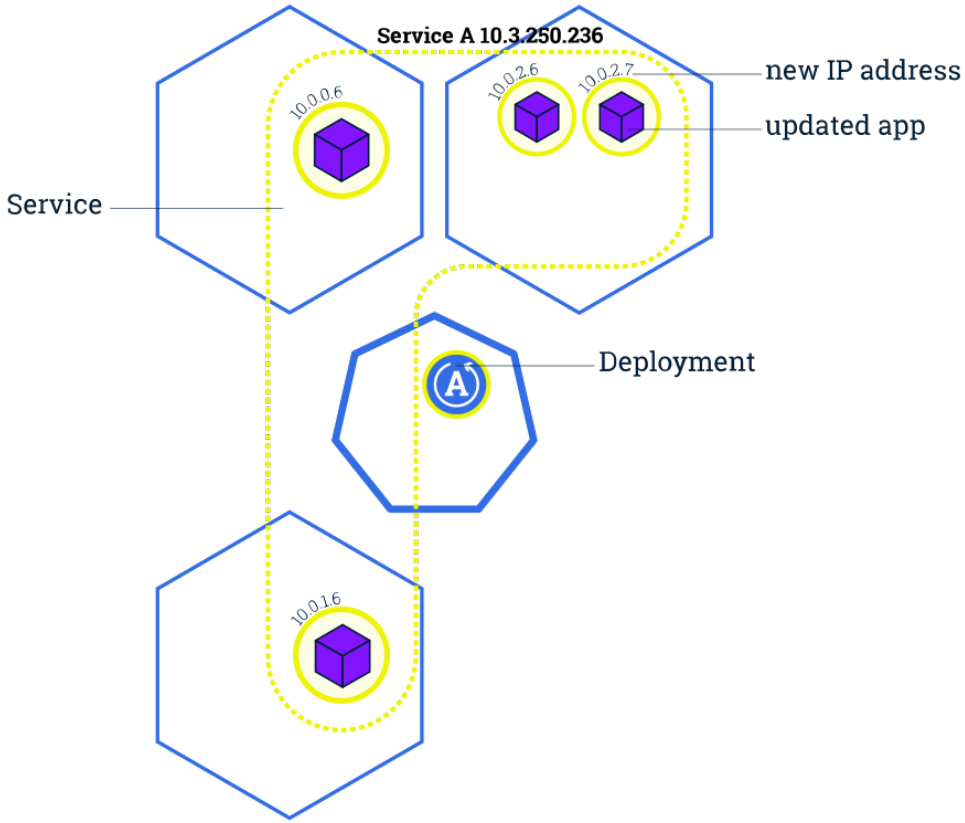


FIGURE – Exemple du Rolling Update tiré de la documentation Kubernetes

SCIENCES SORBONNE UNIVERSITÉ

Kubernetes Node i.e Worker Node

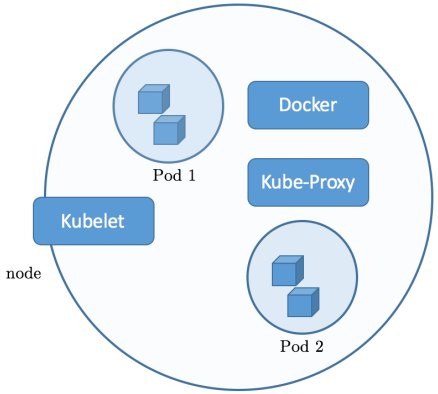


FIGURE – Exemple de node

Rôle du node

- Les pods sont exécuté dans des nodes.
- Il contient les services de gestion et de communications entre les containers.
- Il assigne les ressources au containers qui sont ordonnancé par le master node.

SCIENCES SORBONNE UNIVERSITÉ

Kubernetes Node i.e Worker Node

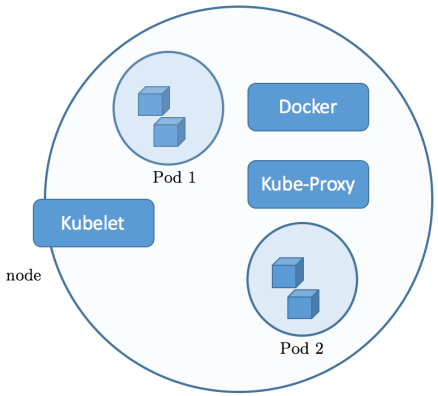


FIGURE – Exemple de node

Rôle du node

- Les pods sont exécuté dans des nodes.
- Il contient les services de gestion et de communications entre les containers.
- Il assigne les ressources au containers qui sont ordonnancé par le master node.

Kubelet

- Il est responsable de l'exécution (start/stop/maintenance)
- Il surveille l'état d'un pod, communique l'état du node au master node.
- Interaction avec un moteur Docker sous-jacent pour démarrer des conteneurs.

SCIENCES SORBONNE UNIVERSITÉ

Kubernetes Node i.e Worker Node

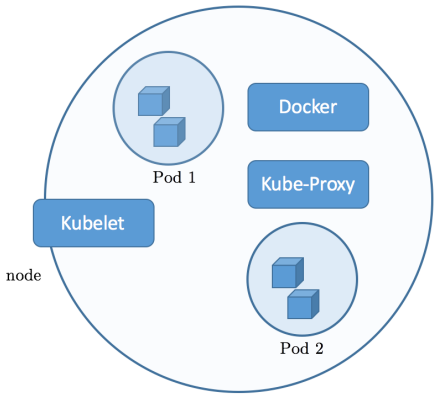


FIGURE – Exemple de node

Rôle du node

- Les pods sont exécuté dans des nodes.
- Il contient les services de gestion et de communications entre les containers.
- Il assigne les ressources au containers qui sont ordonnancé par le master node.

Kubelet

- Il est responsable de l'exécution (start/stop/maintenance)
- Il surveille l'état d'un pod, communique l'état du node au master node.
- Interaction avec un moteur Docker sous-jacent pour démarrer des conteneurs.

Kube-proxy

- Routage du trafic vers le conteneur grâce à l'IP/Port et *répartiteur de charge*.

Kubernetes Master

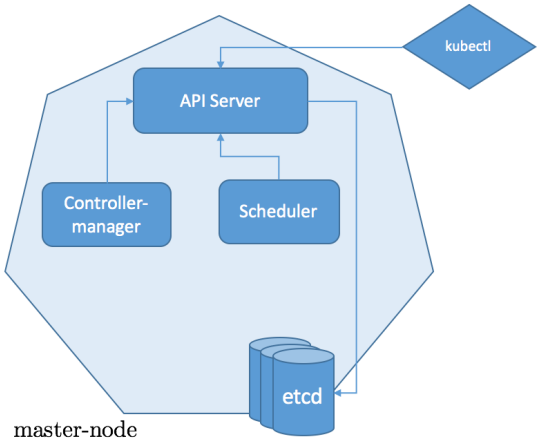


FIGURE – Exemple de master node

Master Node

Le master node est responsable du management du cluster Kubernetes.

- C'est le point d'entré de toutes les tâches administratives.
- Il est le responsable de *l'orchestration* des worker-nodes
- Il est composé de plusieurs sous éléments.

Master Node - l'API Server

- C'est le point d'entrée pour toutes les commandes utilisé pour contrôler le cluster.
- Il récupère les commandes (REST), les valides, et les exécutes.
- Le résultat de ces commandes est conservé dans *etcd*.

Kubernetes Master

Master Node - etcd

- etcd est utilisé pour partager la configuration et la découverte de service.
- API pour des opérations CRUD et permet aux nœuds de s'enregistrer

Master Node - kube-scheduler

Le scheduler permet le deployment de pods configurer et de services sur les nodes.

- Il possède les informations des ressources disponibles.
- Il possède également les informations requises pour l'exécution du service.
- Il choisit l'endroit du déploiement.

Master Node - kube-controller-manager

C'est un composant qui possède plusieurs controllers.

- Node Controller : Responsable de la bonne gestion des noeuds.
- Replication Controller : Responsable du maintient du bon nombre de pods pour chaque ReplicatSet objet du système.
- Endpoints Controller : Service qui associe service et pods.

Kubernetes Overview

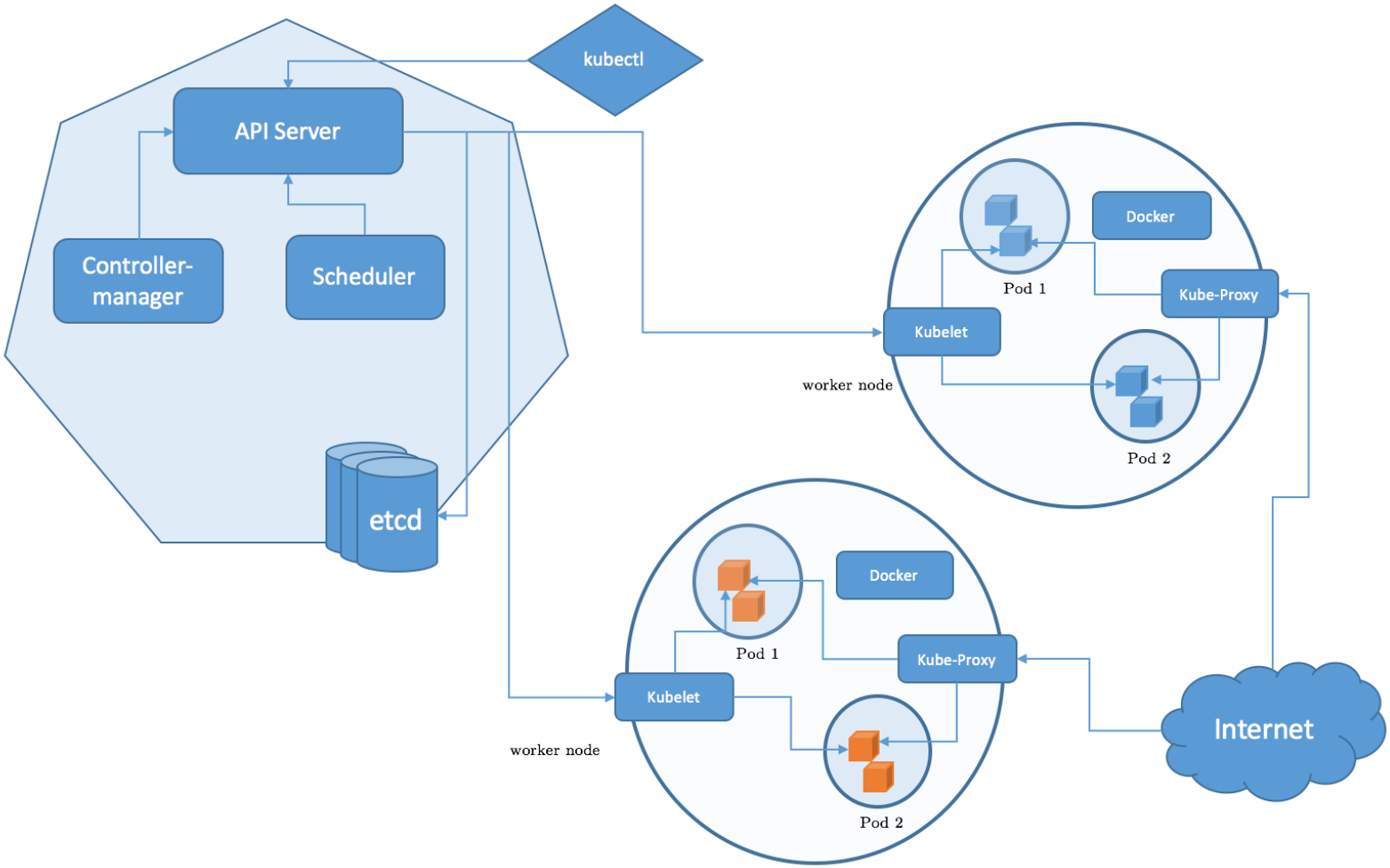


FIGURE – Overview

Comment Kubernetes Scheduler récupère-t-il un Pod ?

D'après le commit 5871b50 de https ://github.com/kubernetes/kubernetes/ il y a 3 grandes étapes :

```
// Etape 1 : lancement d'une goroutine sched.scheduleOne
go wait.Until(sched.scheduleOne, 0, sched.config.StopEverything)
```

SCIENCES SORBONNE UNIVERSITÉ

Comment Kubernetes Scheduler récupère-t-il un Pod ?

D'après le commit 5871b50 de https ://github.com/kubernetes/kubernetes/ il y a 3 grandes étapes :

```
// Etape 1 : lancement d'une goroutine sched.scheduleOne
go wait.Until(sched.scheduleOne, 0, sched.config.StopEverything)

func (sched *Scheduler) scheduleOne() { // Etape 2 : Récupération du prochain pod
    pod  := sched.config.getNextPod()
    // Autres élements du scheduler
}
```

SCIENCES SORBONNE UNIVERSITÉ

Comment Kubernetes Scheduler récupère-t-il un Pod ?

D'après le commit 5871b50 de https ://github.com/kubernetes/kubernetes/ il y a 3 grandes étapes :

```
// Etape 1 : lancement d'une goroutine sched.scheduleOne
go wait.Until(sched.scheduleOne, 0, sched.config.StopEverything)

func (sched *Scheduler) scheduleOne() { // Etape 2 : Récupération du prochain pod
     pod  := sched.config.getNextPod()
     // Autres élements du scheduler
}

func ( c *configFactory) getNextPod() *v1.Pod {
    pod, err  := c.podQueue.Pop() // Etape 3 : On retire le pod d'une file de Pod
    if err == nil {
        return pod
     }
     return nil
}
```

Comment Kubernetes Scheduler récupère-t-il un Pod ?

D'après le commit 5871b50 de https ://github.com/kubernetes/kubernetes/ il y a 3 grandes étapes :

```
// Etape 1 : lancement d'une goroutine sched.scheduleOne
go wait.Until(sched.scheduleOne, 0, sched.config.StopEverything)

func (sched *Scheduler) scheduleOne() { // Etape 2 : Récupération du prochain pod
     pod  := sched.config.getNextPod()
     // Autres élements du scheduler
}

func ( c *configFactory) getNextPod() *v1.Pod {
    pod, err  := c.podQueue.Pop() // Etape 3 : On retire le pod d'une file de Pod
    if err == nil {
        return pod
     }
     return nil
} //Les pods sont ajouté à la queue par un EventHandler
```

SCIENCES SORBONNE UNIVERSITÉ

Comment Kubernetes Scheduler place-t-il ce Pod ?

TODO

SCIENCES SORBONNE UNIVERSITÉ

Mon petit Replication Controller

Conclusion