



# Concepts de base de Kubernetes

Objets, Pods, Labels et mécanismes de gestion

# Vue d'ensemble des objets Kubernetes

## Qu'est-ce qu'un objet Kubernetes ?

Les objets Kubernetes sont des **entités persistantes** dans le système Kubernetes qui représentent l'état du cluster. Ils peuvent décrire :

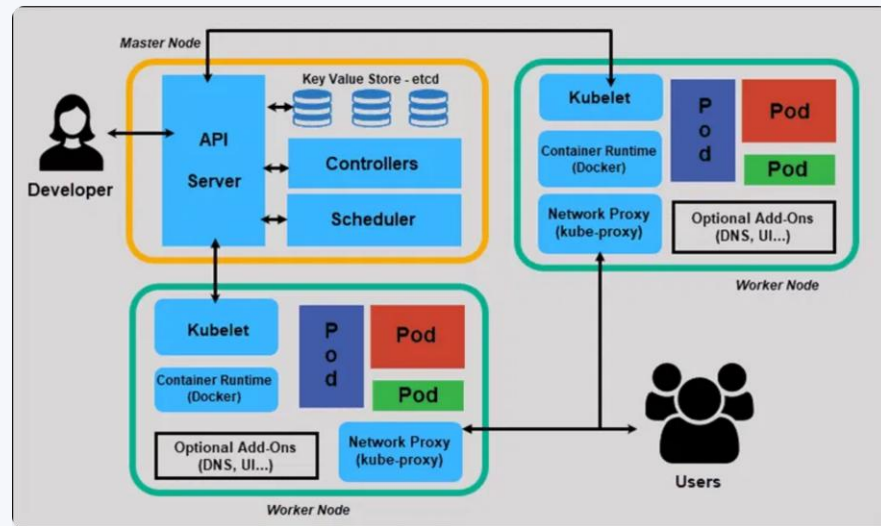
Quelles applications conteneurisées sont en cours d'exécution (et sur quels nœuds)

Les ressources disponibles pour ces applications

Les politiques autour du comportement de ces applications




## Structure d'un objet Kubernetes

- ✓ **spec** : État souhaité de l'objet (ce que vous voulez)
- ✓ **status** : État actuel de l'objet (fourni par Kubernetes)
- ✓ **metadata** : Informations d'identification (nom, UID, namespace)
- ✓ **kind** : Type d'objet (Pod, Service, Deployment, etc.)
- ✓ **apiVersion** : Version de l'API Kubernetes utilisée



# Créer un objet




## Méthodes de création

-  **Fichiers manifestes** : YAML ou JSON décrivant l'objet
-  **kubectl** : Outil en ligne de commande pour interagir avec Kubernetes
-  **API Kubernetes** : Accès programmatique via bibliothèques clientes




## Exemple de manifeste

```
# deployment.yaml apiVersion: apps/v1 kind: Deployment metadata:
name: nginx-deployment labels: app: nginx spec: replicas: 3 selector:
matchLabels: app: nginx template: metadata: labels: app: nginx
spec: containers: - name: nginx image: nginx:1.14.2 ports: -
containerPort: 80
```

## Commandes kubectl

-  `kubectl apply -f deployment.yaml`  
Crée ou met à jour un objet à partir d'un fichier
-  `kubectl create deployment nginx --image=nginx`  
Crée un déploiement directement en ligne de commande
-  `kubectl edit deployment/nginx`  
Modifie un objet existant dans un éditeur

## Validation

-  `kubectl get deployments`  
Liste les déploiements pour vérifier la création
-  `kubectl describe deployment nginx-deployment`  
Affiche les détails d'un déploiement spécifique
-  `kubectl get pods -l app=nginx`  
Liste les pods créés par le déploiement

# Pods : concept et création

## Qu'est-ce qu'un Pod ?

Un Pod est la **plus petite unité déployable** que vous pouvez créer et gérer dans Kubernetes. Il représente un groupe d'un ou plusieurs conteneurs avec des ressources partagées :

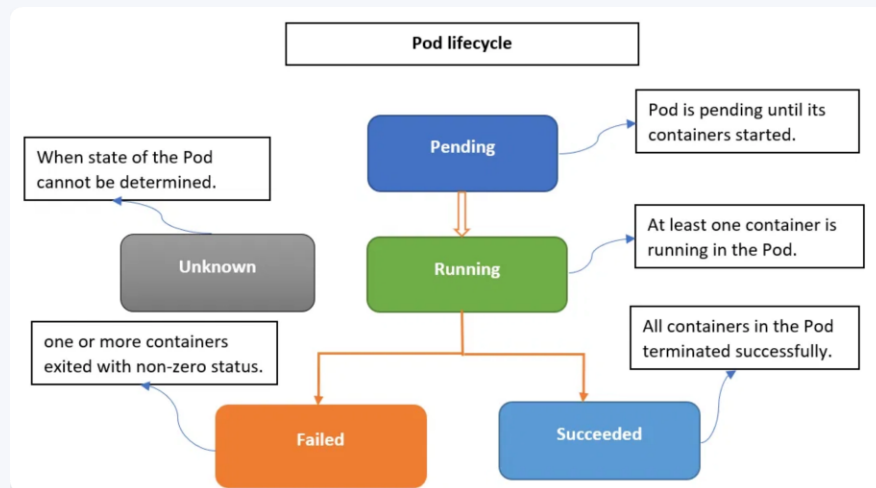
Stockage partagé (volumes)

Ressources réseau (adresse IP unique)

Spécification sur comment exécuter les conteneurs

## Caractéristiques clés

- ✓ **Modèle d'hôte logique** : Les conteneurs dans un Pod sont comme des processus sur une même machine
- ✓ **Co-localisation** : Tous les conteneurs d'un Pod s'exécutent toujours sur le même nœud
- ✓ **Contexte partagé** : Les conteneurs partagent le même espace de noms Linux et peuvent communiquer via localhost
- ✓ **Éphémère** : Les Pods sont conçus pour être des entités relativement éphémères et jetables



## Exemple de création

```
# pod-simple.yaml  apiVersion: v1 kind: Pod metadata: name: nginx-
pod labels: app: nginx spec: containers: - name: nginx image:
nginx:1.14.2 ports: - containerPort: 80
```

# Pods : interaction et cycle de vie

## Phases du cycle de vie d'un Pod

### -- Pending

Le Pod a été accepté par Kubernetes mais un ou plusieurs conteneurs ne sont pas encore créés. Cela inclut le temps d'attente pour le téléchargement des images.

### ● Running

Le Pod est lié à un nœud et tous ses conteneurs ont été créés. Au moins un conteneur est en cours d'exécution, de démarrage ou de redémarrage.

### ● Succeeded

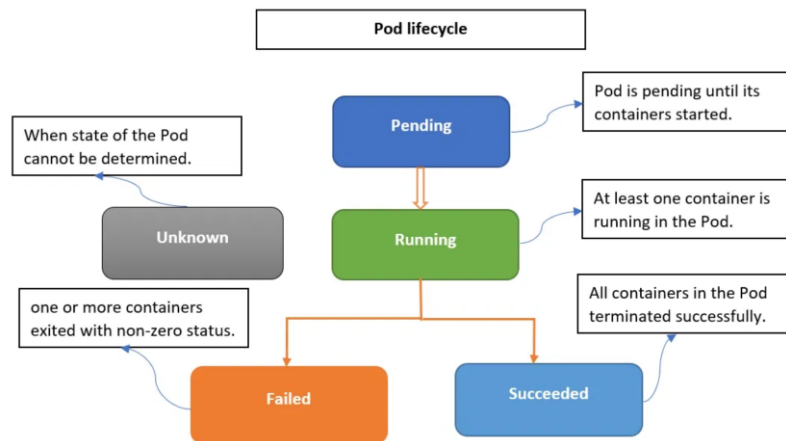
Tous les conteneurs du Pod se sont terminés avec succès et ne seront pas redémarrés.

### ● Failed

Tous les conteneurs du Pod se sont terminés et au moins un conteneur s'est terminé en échec.

### ● Unknown

L'état du Pod ne peut pas être déterminé, généralement en raison d'un problème de communication avec le nœud hôte.



## Interaction avec les Pods

- `kubectl get pods` - Liste tous les pods dans le namespace courant
- `kubectl describe pod <nom-du-pod>` - Affiche les détails d'un pod spécifique
- `kubectl logs <nom-du-pod>` - Affiche les logs d'un pod
- `kubectl exec -it <nom-du-pod> -- /bin/bash` - Exécute une commande dans un pod
- `kubectl port-forward <nom-du-pod> 8080:80` - Transfère un port local vers un port du pod

# Labels et Selectors

## Labels

Les Labels sont des paires clé/valeur attachées aux objets Kubernetes qui permettent d'identifier des attributs significatifs.

```
metadata: labels: app: nginx tier: frontend environment: production
```

## Selectors

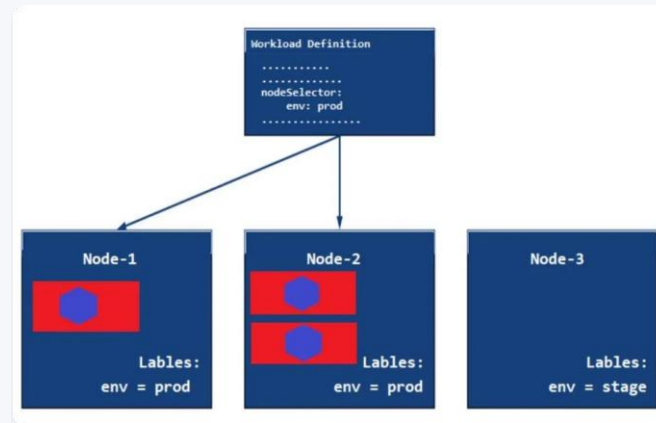
Les sélecteurs de labels permettent d'identifier un ensemble d'objets dans Kubernetes.

### Sélecteurs basés sur l'égalité :

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

### Sélecteurs basés sur l'ensemble :

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



## Cas d'utilisation

- ✓ **Services** : Cibler des pods spécifiques pour le routage du trafic
- ✓ **Déploiements** : Identifier les pods à gérer et mettre à jour
- ✓ **Filtrage** : `kubectl get pods -l environment=production`





# Travailler avec les ReplicationControllers

## Qu'est-ce qu'un ReplicationController ?

Un **ReplicationController** assure qu'un nombre spécifié de répliques de pod sont en cours d'exécution à tout moment. Il remplace automatiquement les pods qui échouent, sont supprimés ou sont arrêtés.

C'est similaire à un superviseur de processus, mais au lieu de superviser des processus individuels sur un seul nœud, le ReplicationController supervise plusieurs pods sur plusieurs nœuds.

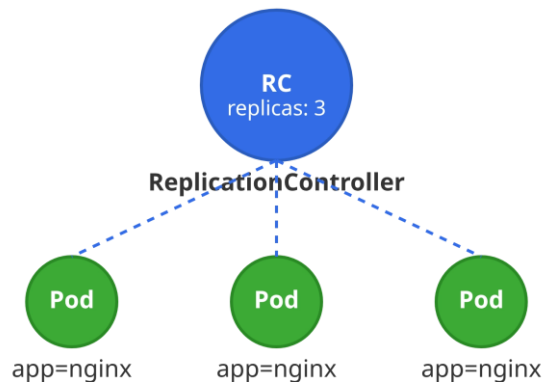
## Caractéristiques principales

-  **Haute disponibilité** : Maintient le nombre souhaité de pods en fonctionnement
-  **Sélection par labels** : Identifie les pods à gérer via des labels
-  **Template de pod** : Définit le modèle pour créer de nouveaux pods
-  **Équilibrage** : Supprime les pods excédentaires ou en crée de nouveaux si nécessaire

## Exemple de ReplicationController

```
# replication-controller.yaml  apiVersion: v1 kind:
ReplicationController metadata: name: nginx-rc spec: replicas: 3
selector: app: nginx template: metadata: labels: app: nginx spec:
containers: - name: nginx image: nginx:1.14.2 ports: - containerPort:
80
```

## Fonctionnement d'un ReplicationController



# ReplicaSets, Deployments et autres contrôleurs

## ReplicaSet

Nouvelle génération de ReplicationController avec sélecteurs basés sur l'ensemble. Maintient un ensemble stable de pods répliqués.

**Cas d'utilisation :** Généralement géré par un Deployment.

## Deployment

Objet de niveau supérieur qui gère les ReplicaSets et fournit des mises à jour déclaratives aux Pods.

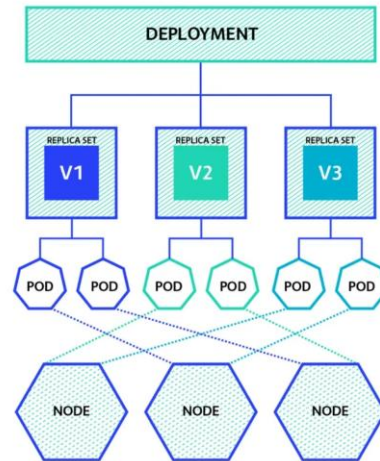
**Cas d'utilisation :** Déploiements avec mises à jour progressives et rollbacks.

## DaemonSet

Assure qu'une copie d'un pod s'exécute sur tous les nœuds du cluster.

## Job / CronJob

Exécute des pods qui effectuent une tâche jusqu'à sa complétion.



### Contrôleur

### Mise à jour

### État

### Sélecteur

ReplicaSet

Non

Sans état

Ensemble

Deployment

Progressive

Sans état

Ensemble

DaemonSet

Progressive

Sans état

Ensemble

Job

Non

Batch

Ensemble



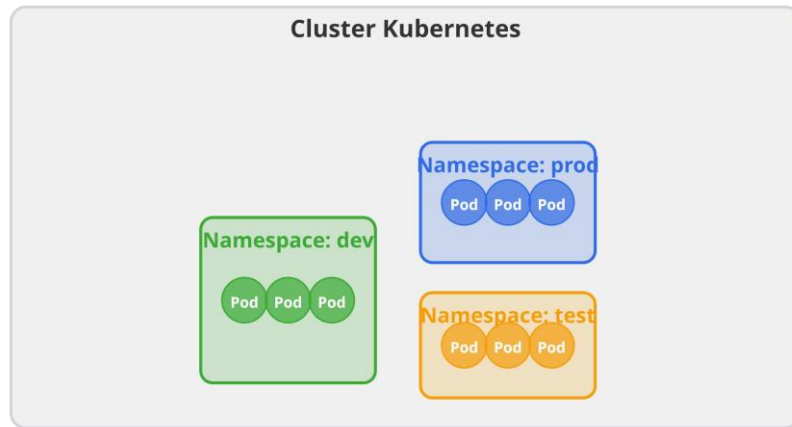
# Namespaces

## Qu'est-ce qu'un Namespace ?

Les Namespaces fournissent un **mécanisme d'isolation** pour les groupes de ressources au sein d'un même cluster Kubernetes. Ils offrent une portée pour les noms et permettent de diviser les ressources du cluster entre plusieurs utilisateurs.

Les noms des ressources doivent être uniques au sein d'un namespace, mais pas entre les namespaces. Les namespaces ne peuvent pas être imbriqués et chaque ressource Kubernetes ne peut être que dans un seul namespace.

## Organisation des ressources par Namespace



# Namespaces

## Namespaces initiaux

### default

Namespace par défaut pour les objets sans autre namespace spécifié

### kube-system

Namespace pour les objets créés par le système Kubernetes

### kube-public

Namespace lisible par tous les utilisateurs, utilisé pour les ressources visibles publiquement

### kube-node-lease

Namespace pour les objets de bail associés à chaque nœud pour la détection des défaillances

## Commandes utiles

- > `kubectl get namespaces` - Liste tous les namespaces
- > `kubectl create namespace dev` - Crée un nouveau namespace
- > `kubectl -n dev get pods` - Liste les pods dans le namespace "dev"
- > `kubectl config set-context --current --namespace=dev` - Définit le namespace par défaut

## Cas d'utilisation

- ✓ **Isolation des environnements** : Séparer dev, test, production
- ✓ **Séparation des équipes** : Allouer des ressources par équipe
- ✓ **Quotas de ressources** : Limiter les ressources par namespace
- ✓ **Politiques de sécurité** : Appliquer des règles différentes par namespace

# Conclusion et bonnes pratiques

## Points clés à retenir

- 📦 Les **objets Kubernetes** sont des entités persistantes qui représentent l'état du cluster
- 🔗 Les **Pods** sont la plus petite unité déployable dans Kubernetes
- 🔍 Les **Labels et Selectors** permettent d'organiser et de cibler des groupes d'objets
- 🔄 Les **contrôleurs** (ReplicaSets, Deployments) gèrent le cycle de vie des pods
- 🏠 Les **Namespaces** permettent d'isoler les ressources au sein d'un cluster

## Bonnes pratiques

- ✅ Utiliser des **Deployments** plutôt que des ReplicaSets ou des Pods directs
- ✅ Organiser les ressources avec des **labels cohérents** et significatifs
- ✅ Structurer les applications avec des **namespaces** pour isoler les environnements
- ✅ Définir des **limites de ressources** pour tous les conteneurs
- ✅ Implémenter des **sondes de vivacité et de préparation** pour une meilleure résilience

## Ressources pour approfondir

- 📖 [Documentation officielle Kubernetes](#)
- 🎓 [Tutoriels Kubernetes](#)
- 📖 [Aide-mémoire kubectl](#)
- 🔗 [Exemples de code Kubernetes](#)
- 👥 [Communauté Kubernetes](#)

## Prochaines étapes

- ➔ Explorer les **Services** et le **Networking** Kubernetes
- ➔ Comprendre le **stockage persistant** avec les PersistentVolumes
- ➔ Apprendre à configurer la **sécurité** et le **RBAC**
- ➔ Mettre en place des **pipelines CI/CD** avec Kubernetes
- ➔ Découvrir les **opérateurs** et l'extension de Kubernetes