

Exercices

# Kubernetes (K8S)

---

Haruna Rashid yakubu

15 septembre 2024

## Exercice 1 : Déploiement d'une application simple dans un Pod

### Objectif :

Créer et déployer un pod Kubernetes exécutant une application web simple.

### Instructions :

- Création du fichier de configuration du pod :**
  - Créez un fichier YAML nommé `pod-simple-app.yaml`.
  - Dans ce fichier, définissez un pod qui utilise l'image Docker `nginx:latest`.
  - Assurez-vous que le pod est nommé `simple-web-pod`.
- Déploiement du pod :**
  - Utilisez la commande `kubectl` pour déployer le pod dans votre cluster.
- Vérification du déploiement :**
  - Vérifiez que le pod est en cours d'exécution avec la commande `kubectl get pods`.
  - Récupérez les logs du pod pour vous assurer que Nginx fonctionne correctement.
- Accès à l'application :**
  - Exposez le pod en utilisant un port-forwarding pour accéder à l'application web via votre navigateur.
  - Accédez à `http://localhost:8080` pour voir la page par défaut de Nginx.

### Points à considérer :

- Assurez-vous que votre cluster Kubernetes est opérationnel.
- Utilisez des commandes appropriées pour diagnostiquer les problèmes éventuels (`kubectl describe pod`, `kubectl logs`, etc.).

---

## Exercice 2 : Utilisation des Labels et Sélecteurs

## Objectif :

Gérer et sélectionner des pods à l'aide de labels et de sélecteurs.

## Instructions :

### 1. Ajout de labels au pod existant :

- Modifiez le fichier `pod-simple-app.yaml` pour ajouter les labels suivants au pod :

- `app: web`
- `env: dev`

### 2. Redéploiement du pod :

- Supprimez le pod existant avec `kubectl delete pod simple-web-pod`.
- Déployez à nouveau le pod avec les labels mis à jour.

### 3. Sélection des pods via les labels :

- Utilisez la commande `kubectl get pods --show-labels` pour afficher les pods avec leurs labels.
- Exécutez une commande pour sélectionner uniquement les pods avec `env=dev`.

### 4. Mise à jour des labels :

- Modifiez le label `env` du pod de `dev` à `production` sans redéployer le pod.

## Points à considérer :

- Familiarisez-vous avec les options de `kubectl` pour filtrer les ressources par labels.
- Utilisez la commande `kubectl label` pour mettre à jour les labels d'un pod existant.

---

## Exercice 3 : Mise en place d'un ReplicaSet

## Objectif :

Assurer la haute disponibilité de votre application en déployant un ReplicaSet.

## Instructions :

### 1. Création du fichier de configuration du ReplicaSet :

- Créez un fichier YAML nommé `replicaset-web.yaml`.
- Définissez un ReplicaSet qui gère 3 réplicas de l'image `nginx:latest`.

- Utilisez les labels `app: web` et `env: production` pour les pods.
- 2. **Déploiement du ReplicaSet :**
  - Utilisez `kubectl` pour déployer le ReplicaSet dans le cluster.
- 3. **Vérification du déploiement :**
  - Utilisez `kubectl get replicaset` pour vérifier que le ReplicaSet est déployé.
  - Vérifiez que 3 pods sont créés et en cours d'exécution.
- 4. **Test de la résilience :**
  - Supprimez un des pods manuellement.
  - Observez comment le ReplicaSet recrée automatiquement le pod pour maintenir le nombre de réplicas désiré.

**Points à considérer :**

- Comprenez comment les labels et les sélecteurs sont utilisés par le ReplicaSet pour gérer les pods.
- Utilisez `kubectl describe replicaset` pour obtenir plus d'informations sur le ReplicaSet.

---

## Exercice 4 : Création d'un Deployment avec Stratégies de Déploiement

**Objectif :**

Déployer une nouvelle version de l'application en utilisant un Deployment et en appliquant une stratégie de déploiement spécifique.

**Instructions :**

1. **Création du fichier de configuration du Deployment :**
  - Créez un fichier YAML nommé `deployment-web.yaml`.
  - Définissez un Deployment pour l'application web avec 3 réplicas.
  - Utilisez l'image `nginx:1.17` initialement.
  - Configurez la stratégie de déploiement en mode `RollingUpdate` avec un `maxUnavailable` de 1 et un `maxSurge` de 2.
2. **Déploiement de l'application :**
  - Déployez le Deployment avec `kubectl`.

### 3. Mise à jour de l'application :

- Mettez à jour l'image de l'application dans le fichier YAML vers `nginx:1.18`.
- Appliquez les modifications avec `kubectl apply`.

### 4. Observation de la stratégie de déploiement :

- Utilisez `kubectl rollout status deployment/<nom_du_deployment>` pour suivre le déploiement.
- Observez comment les pods sont mis à jour progressivement sans interruption du service.

#### Points à considérer :

- Comprenez les avantages d'utiliser un Deployment par rapport à un ReplicaSet.
  - Familiarisez-vous avec les commandes de gestion des rollouts (`kubectl rollout history`, `kubectl rollout undo`, etc.).
- 

## Exercice 5 : Gestion des Nœuds (Nodes)

#### Objectif :

Interagir avec les nœuds du cluster Kubernetes pour gérer les applications en cas de maintenance ou de pannes.

#### Instructions :

##### 1. Liste des nœuds :

- Utilisez `kubectl get nodes` pour lister tous les nœuds de votre cluster.

##### 2. Drainage d'un nœud :

- Choisissez un nœud (par exemple, `node-1`) et exécutez la commande `kubectl drain node-1 --ignore-daemonsets` pour évacuer tous les pods.
- Vérifiez que les pods sont reprogrammés sur d'autres nœuds.

##### 3. Marquage d'un nœud comme non planifiable :

- Exécutez `kubectl cordon node-1` pour empêcher de nouveaux pods d'être programmés sur ce nœud.

##### 4. Remise en service du nœud :

- Après la maintenance, exécutez `kubectl uncordon node-1` pour permettre à nouveau la planification des pods sur ce nœud.

### Points à considérer :

- Comprenez l'importance de gérer les nœuds lors des opérations de maintenance.
  - Assurez-vous de ne pas interrompre les applications critiques pendant ces opérations.
- 

## Exercice 6 : Déploiement d'une Application Multi-Conteneur dans un Pod

### Objectif :

Déployer un pod qui contient plusieurs conteneurs travaillant ensemble.

### Instructions :

1. **Création du fichier de configuration du pod multi-conteneurs :**
  - Créez un fichier YAML nommé `pod-multi-container.yaml`.
  - Définissez un pod nommé `web-with-logger` qui contient deux conteneurs :
    - Un conteneur `nginx` pour servir du contenu web.
    - Un conteneur `busybox` qui effectue un `tail -f` sur les logs d'accès de Nginx.
2. **Déploiement du pod :**
  - Déployez le pod avec `kubectl`.
3. **Vérification du fonctionnement :**
  - Accédez au service Nginx pour générer des logs.
  - Utilisez `kubectl logs` pour le conteneur `busybox` et vérifiez qu'il affiche les logs d'accès de Nginx.

### Points à considérer :

- Comprenez comment les conteneurs dans le même pod partagent le même réseau et le même espace de stockage.
  - Utilisez les champs `volume` et `volumeMounts` pour partager des données entre les conteneurs.
-

## Exercice 7 : Sélectionner des Pods avec des Sélecteurs Complexes

### Objectif :

Utiliser des sélecteurs complexes pour gérer des groupes de pods.

### Instructions :

- Déploiement de plusieurs pods avec différents labels :**
  - Créez plusieurs pods avec différentes combinaisons de labels :
    - Pod A : `app=web, env=dev`
    - Pod B : `app=web, env=production`
    - Pod C : `app=api, env=dev`
    - Pod D : `app=api, env=production`
- Sélection avec des sélecteurs :**
  - Utilisez des commandes `kubectl` pour sélectionner :
    - Tous les pods où `app=web`.
    - Tous les pods où `env=production`.
    - Tous les pods où `app=web et env=production`.
    - Tous les pods où `app` est `web ou api` et `env=dev`.
- Application de mises à jour ciblées :**
  - Exécutez une commande pour mettre à jour l'image des pods où `env=dev` sans affecter les pods en production.


### Points à considérer :

- Familiarisez-vous avec les opérateurs de sélecteurs (`=`, `==`, `!=`, `in`, `notin`).
- Utilisez les champs `spec.selector` dans vos fichiers YAML pour appliquer des sélecteurs complexes.

---

## Exercice 8 : Stratégies de Déploiement Avancées

### Objectif :



Mettre en œuvre des stratégies de déploiement avancées comme le Blue-Green Deployment ou le Canary Deployment.

**Instructions :**

**1. Blue-Green Deployment :**

- Déployez une version **v1** de votre application avec le label **version: v1**.
- Déployez une version **v2** de votre application avec le label **version: v2** sans supprimer la version **v1**.
- Configurez un service qui pointe initialement vers les pods avec **version: v1**.
- Basculez le service pour qu'il pointe vers les pods avec **version: v2** une fois que vous êtes prêt.

**2. Canary Deployment :**

- Déployez votre application principale avec 5 réplicas (version stable).
- Déployez une nouvelle version (canary) avec 1 réplique.
- Configurez le service pour distribuer le trafic entre les pods de manière à ce qu'environ 16% du trafic aille vers la version canary.
- Surveillez les performances et les erreurs de la version canary avant de déployer complètement.