TP7 : KUBERNETES

Partie 1 : Préparation de l'environnement & Lancement du Minikube

1. Nous allons tout d'abord récupérer la CLI kubectl, nécessaire à Minikube.
   Pour installer la dernière version stable de minikube sur Windows x86-64, allez sur le site : https://minikube.sigs.k8s.io/docs/start/
   Vous devez lancer l-instalation avec powershell
2. Lancer minikube avec Docker comme driver : **minikube start --driver=docker**
   Cette commande sert à démarrer une instance de Minikube en utilisant Docker comme moteur de virtualisation.
   Minikube : est un outil qui permet de faire tourner un cluster Kubernetes localement sur votre machine.
   Driver = docker : Cette option spécifie que Minikube doit utiliser Docker comme moteur de virtualisation pour créer et gérer les machines virtuelles nécessaires au cluster Kubernetes.
3. Lancez la commande **kubectl cluster-info**
   Cette commande fournit des informations sur les services principaux du cluster Kubernetes auquel vous êtes connecté. kubectl : C'est l'outil en ligne de commande pour interagir avec Kubernetes.
4. Lancez la commande **minikube dashboard**
   Cette commande sert à lancer l'interface graphique du tableau de bord Kubernetes (Kubernetes Dashboard) pour le cluster Minikube.
5. Lancez la commande **kubectl get nodes**
   Cette commande est utilisée pour afficher des informations sur les nœuds d'un cluster Kubernetes.

Partie 2 : Préparation de l'environnement & Lancement du Minikube

1. Lancez la commande
   kubectl create deployment first-deployment --image=katacoda/docker-http-server
   Cette commande est utilisée pour créer un déploiement dans un cluster Kubernetes.
   Un déploiement est une ressource Kubernetes qui gère un ensemble de répliques de pods pour assurer la disponibilité et la mise à jour de vos applications.
2. Lancez la commande **kubectl get deploy & kubectl get nodes & kubectl get pods**
3. Il est possible d'exposer le port du conteneur sur la VM minikube en utilisant un service de type NodePort. La commande suivante expose le port 80 du conteneur sur un port aléatoire de l'host : **kubectl expose deployment first-deployment --port=80 --type=NodePort**
4. Récupérer le port des services associés :
   kubectl get svc

Partie 3 : Scaler un deployment

1. La commande kubectl scale permet d'ajuster le nombre de réplica d'un déploiement:
   kubectl scale --replicas=3 deployment first-deployment
2. Listez ensuite les pods disponibles: **kubectl get pods**
3. Il y a maintenant 3 pods disponibles pour ce Deployment
   Décrivez ensuite l'objet de type Service associé et regarder la partie endpoints:
   Que remarquez-vous ?
   kubectl describe svc first-deployment
   kubectl get endpoints

Partie 4 : Installation du dashboard

1. Pour démarrer le dashboard sur minikube : **minikube addons enable dashboard**
2. Vérifier le déploiement du dashboard :
   kubectl -n kubernetes-dashboard get pods
   kubectl -n kubernetes-dashboard get svc
3. Comme pour notre premier déploiement, il est possible d'exposer le service associé au dashboard via un NodePort :
   kubectl -n kubernetes-dashboard edit svc kubernetes-dashboard

Partie 5 : Les fichiers YAML

DIFFERENCE ENTRE UN POD ET DEPLOIEMENT

1. Nous utiliserons l'image helloworld de Docker comme démonstrateur. Celle-ci expose un service web sur le port 80. Nous allons d'abord créer un objet de type Pod.
   Dans un fichier hello-world-pod.yaml :

```
apiVersion: v1
kind: Pod
metadata:
      name: helloworld
      labels:
            app: helloworld
spec:
      containers:
      - name: helloworld
        image: particule/helloworld
        ports:
        - containerPort: 80
```

2. Pour appliquer ce fichier sur le cluster :
   kubectl apply -f hello-world-pod.yaml
3. Vérifiez que votre pod est en marche avec **kubectl get pods**.
   Supprimez votre pod avec **kubectl delete pod nom_de_votre_pod**.
   Est-il correctement supprimé ?
4. Nous allons maintenant créer un objet de type Deployment
   Dans un fichier helloworld-de.yml.

```
 apiVersion: apps/v1
kind: Deployment
metadata:
  name: helloworld
  labels:
    app: helloworld
spec:
  replicas: 1
  selector:
    matchLabels:
      app: helloworld
  template:
```

```
metadata:
  labels:
    app: helloworld
spec:
  containers:
  - name: helloworld
    image: particule/helloworld
    ports:
    - containerPort: 80
```

5. Pour appliquer ce fichier sur le cluster : **kubectl apply -f helloworld-de.yml**
6. Vérifiez que votre pod est en marche avec kubectl get pods.
7. Supprimez votre pod avec **kubectl delete pod nom_de_votre_pod**.
   Est-il correctement supprimé ? Pourquoi ?