UNIVERSITE DE SOUSSE Institut Supérieur d'Informatique et des Techniques de Communication Hammam Sousse		جـامعة سوسة المعهد العالي للإعلامية وتقنيات الاتصال بحمام سوسة

TP2 : Introduction à Kubernetes **A.U. : 2020/2021**

Matière : Administration et Sécurité des Réseaux
Enseignant : Salah Gontara
Classe : 2DNI

OBJECTIF(S):

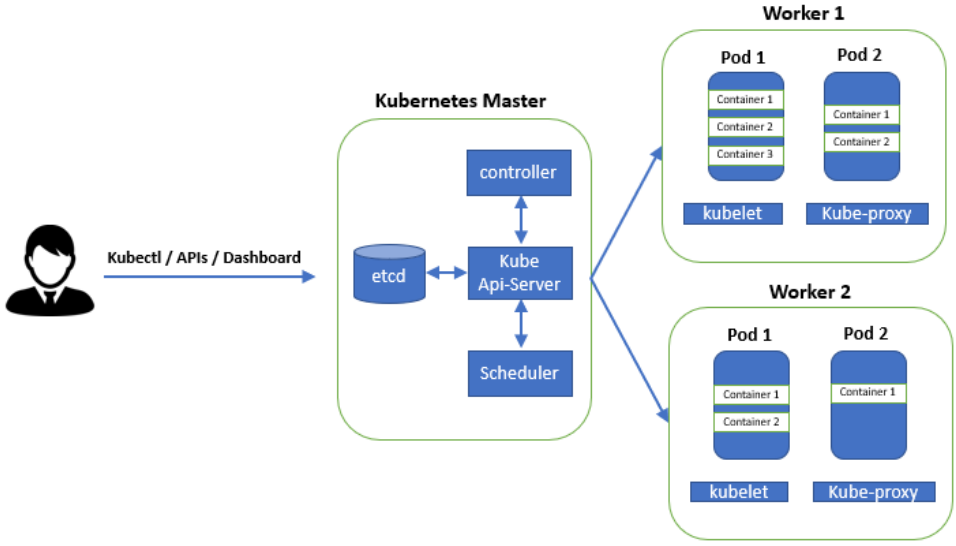
- Comprendre le fonctionnement de base de Kubernetes et Minikube
- Interagir efficacement (en mode debug) avec les pods
- Créer des déploiements avec et sans fichiers de configuration YAML

OUTILS UTILISÉS :

Kubernetes (Minikube), Nginx, Mongo

Préambule:

Kubernetes (K8s) est un système open-source permettant d'automatiser le déploiement, la mise à l'échelle et la gestion des applications conteneurisées. Il fonctionne avec toute une série de technologies de conteneurisation, et est souvent utilisé avec Docker. Il a été conçu à l'origine par Google, puis offert à la Cloud Native Computing Foundation. Kubernetes suit une architecture client-serveur. Il est possible d'avoir une configuration multi-maître (pour une haute disponibilité), mais par défaut, il y a un seul serveur maître qui agit comme un nœud de contrôle et un point de contact.



Architecture Kubernetes

Partie 1 : Les commandes de base

1. Démarrez le cluster Minikube :
   sudo minikube start --driver=none

   NB : si vous utilisez linux nativement, vous devez inclure un hyperviseur et utiliser les commandes suivantes sans 'sudo' :
   minikube start --vm-driver=docker

2. Affichez la version de Minikube :
   sudo kubectl version

3. Listez les nœuds de votre cluster :
   sudo kubectl get nodes

4. Affichez le status de votre cluster :
   sudo minikube status

5. Effacez votre cluster et redémarrez le en mode debugging.
   sudo minikube delete
   sudo minikube start --driver=none --alsologtostderr
   sudo minikube status

   NB : si vous utilisez linux nativement, n'oubliez pas d'inclure un hyperviseur et utiliser les commandes sans 'sudo' :
   minikube start --vm-driver=docker -v=7 --alsologtostderr

Partie 2 : Les déploiements simples

1. Listez les pods de votre cluster :
   sudo kubectl get pods

2. Listez les services de votre cluster :
   sudo kubectl get services

3. Créez un déploiement simple de Nginx :
   sudo kubectl create deployment nginx-depl --image=nginx

4. Confirmez la création de ce déploiement :
   sudo kubectl get deployment

5. Confirmez la création de ce déploiement :
   sudo kubectl get deployment

6. Affichez les réplications de vos pods déployés, s'ils existent :
   sudo kubectl get replicaset

7. Créez un déploiement simple de MongoDB :
   sudo kubectl create deployment mongo-depl --image=mongo

8. Affichez les logs de vos pods déployés :
   sudo kubectl logs *nom_pod*

   NB : Les noms des pods sont dégagés de : **sudo kubectl get pods**

9. Accédez à l'un des pods et confirmez le système
   sudo kubectl exec -it *nom_pod* -- bin/bash

   NB : Les noms des pods sont dégagés de : **sudo kubectl get pods**

10. **Affichez la description des pods déployés :**
    sudo kubectl describe pod *nom_pod*

11. Effacez les déploiements :
    sudo kubectl delete deployment mongo-depl
    sudo kubectl delete deployment nginx-depl

Partie 3 : Les déploiements configurés

1. Créez le fichier YAML de configuration suivante :
   vim nginx-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.16
        ports:
        - containerPort: 8080
```

2. Démarrez le déploiement en vous basant sur le fichier de configuration :
   sudo kubectl apply -f nginx-deployment.yaml

3. Affichez les informations sur ce déploiement:
   sudo kubectl get pod
   sudo kubectl get deployment

4. Effacez le déploiement configuré :
   sudo kubectl delete -f nginx-deployment.yaml