TP - Kubernetes

Brahim Hamdi

Préparation de l'environnement Linux

1. Cloner le dépôt git suivant :

   git clone https://github.com/brahimhamdi/k8s-lab

   - Démarrez les 3 VMs vagrant.

     cd k8s-lab

     vagrant up

2. Sur les 3 VMs :

   - Installez docker-ce

     sudo apt update

     sudo apt install apt-transport-https ca-certificates curl software-properties-common

     curl -fsSL https://download.docker.com/linux/ubuntu/gpg |sudo apt-key add -

     sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"

     sudo apt update

     sudo apt install docker-ce

   - Installez kubectl, kubeadm et kubelet

     curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg |sudo apt-key add

     sudo apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"

     sudo apt install kubeadm kubectl kubelet

   - Désactivez le swap

     sudo swapoff -a

     remarque : désactivez le montatge automatique du swap (fichier /etc/fstab)

Initialisation du cluster

3. Initialisez le cluster sur la VM Master.

   sudo kubeadm init --apiserver-advertise-address=192.168.205.10 --pod-network-cidr=10.244.0.0/16

   - Y a-t-il des erreurs ? *Oui, nombre de CPU*

     sudo kubeadm init --apiserver-advertise-address=192.168.205.10 --pod-network-cidr=10.244.0.0/16 `--ignore-preflight-errors=NumCPU

4. Exécutez les commandes nécessaires pour utiliser kubectl sans sudo.

   - Affichez l'état du cluster

kubectl cluster-info

- Donnez la liste des namespaces

  - default

  - kube-node-lease

  - kube-public

  - kube-system

- Affichez tous les Pods de tous les namaspaces. Quels sont les IPs des DNS.

  kubectl get pod –all-namespaces

  2 coredns : 10.244.0.2 et 10.244.0.3

- Affichez la liste des nodes qui appartiennent au cluster

  kubectl get node -o wide

5. Joignez worker1 et worker2 au cluster.

- Vérifiez sur Master que les 2 workers sont prêts.

Création de Pod

6. Créez un Pod exécutant un conteneur nginx en utilisant la ligne de commande.

   kubectl run nginx --image=nginx --restart=Never

- Vérifiez que le pod a été bien crée.

  kubectl get pod

- Sur quel nœud est-il crée ? Donnez son IP.

  Sur worker1

  IP : 10.244.1.2

- Testez son interface web.

  Sur worker1 : curl http://10.244.1.2

- Supprimez le Pod

  kubectl delete pod/nginx

7. Créez le fichier pod1.yml

   vim pod1.yml

- Ecrivez dans pod1.yml le code nécessaire qui permet de créer le pod de la question précédente.

  apiVersion: v1
  kind: Pod
  metadata:

```
    name: web
spec:
    containers:
- name: nginx
  image: nginx
```

◦ Appliquez le fichier yaml.

kubectl apply -f pod1.yml

◦ Donnez l'IP, le nœud et une description détaillée du pod.

Sur worker1

IP : 10.244.1.3

◦ Supprimez le pod.

kubectl delete pod/web

ou

kubectl delete -f pod1.yml

Création de Deployment

8. Créez le fichier deploy1.yml.

vim deploy1.yml

◦ Décrivez dans deploy1.yml le déploiement de 2 replicas du serveur web nginx.

apiVersion: apps/v1

◦ kind: Deployment

◦ metadata:

◦ name: deploy1

◦ labels:

◦ app: web

◦ spec:

◦ replicas: 2

◦ selector:

◦ matchLabels:

◦ app: web

◦ template:

◦ metadata:

◦ labels:

◦ app: web

◦ spec:

- containers:
- - name: nginx
  image: nginx

- Appliquez le fichier yml.
- Donnez les IP des pod, les nœuds et une description détaillée du déploiement et de chaque pod.
  - Y a t-il un problème ?
- Donnez 2 méthodes pour augmenter le nombre de replicas à 3.

- Modifiez la page web de chaque pod pour distinguer l'une de l'autre.

- Testez la page web de chaque pod.
  - Comment exposer le service web de ce déploiement (3 pods) ?

Création de Service

9. Créez le fichier srv1.yml

- Décrivez dans ce fichier comment exposer le service web de déploiement deploy1 sur un port TCP de chaque nœud.
- Appliquez le fichier yaml.
- Sur quel port est exposé le service ?
- Accédez à l'interface web de l'ensemble des 3 pods.

Déploiement de Dockercoins

10. Supprimez tous le pod, le déploiement et le service que vous avez créé.

kubectl delete all --all

11. Dans ce qui suit on va déployer l'application Dockercoins sur le cluster k8s.

- Créez dans le fichier « dockercoins.yml » le code yaml permettant de :
  - déployer l'application Dockercoins.
  - Exposer les services redis, rng et hasher en interne du cluster
  - Exposer le service webui en externe du cluster

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webui
  labels:
    app: dockercoins
    tier: front
    environment: prod
```

```
spec:
  replicas: 1
  selector:
    matchLabels:
      app: dockercoins
      tier: front
  template:
    metadata:
      labels:
        app: dockercoins
        tier: front
        environment: prod
    spec:
      containers:
        - name: webui
          image: brahimhamdi/webui
---
apiVersion: v1
kind: Service
metadata:
  name: webui
  labels:
    app: dockercoins
    tier: front
    environment: prod
spec:
  selector:
    app: dockercoins
    tier: front
  type: NodePort
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
      nodePort: 30080
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis
  labels:
    app: dockercoins
    tier: db
    environment: prod
spec:
  replicas: 1
  selector:
    matchLabels:
      app: dockercoins
      tier: db
  template:
    metadata:
      labels:
        app: dockercoins
        tier: db
```

```
      environment: prod
  spec:
    containers:
      - name: redis
        image: redis
---
apiVersion: v1
kind: Service
metadata:
  name: redis
spec:
  selector:
    app: dockercoins
    tier: db
  type: ClusterIP
  ports:
    - protocol: TCP
      port: 6379
      targetPort: 6379
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: worker
  labels:
    app: dockercoins
    tier: worker
    environment: prod
spec:
  replicas: 1
  selector:
    matchLabels:
      app: dockercoins
      tier: worker
  template:
    metadata:
      labels:
        app: dockercoins
        tier: worker
        environment: prod
    spec:
      containers:
        - name: worker
          image: brahimhamdi/worker
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: rng
  labels:
    app: dockercoins
    tier: rng
    environment: prod
spec:
  replicas: 1
```

```
  selector:
    matchLabels:
      app: dockercoins
      tier: rng
  template:
    metadata:
      labels:
        app: dockercoins
        tier: rng
        environment: worker
    spec:
      containers:
        - name: rng
          image: brahimhamdi/rng
---
apiVersion: v1
kind: Service
metadata:
  name: rng
spec:
  selector:
    app: dockercoins
    tier: rng
  type: ClusterIP
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hasher
  labels:
    app: dockercoins
    tier: hasher
    environment: prod
spec:
  replicas: 1
  selector:
    matchLabels:
      app: dockercoins
      tier: hasher
  template:
    metadata:
      labels:
        app: dockercoins
        tier: hasher
        environment: prod
    spec:
      containers:
        - name: hasher
          image: brahimhamdi/hasher
---
apiVersion: v1
```

```
kind: Service
metadata:
  name: hasher
spec:
  selector:
    app: dockercoins
    tier: hasher
  type: ClusterIP
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

12. Modifiez le fichier dockercoins.yml en augmentant le nombre de replicas des workers, rng et hasher à 2.
13. Créez un autoscaling horizontal (hpa) pour le déploiement de hasher avec les spécifications suivantes :
    - nombre min de replicas : 1
    - nombre de réplicas : 5
    - poucentage d'utilisation de CPU : 50%