Gérer une infrastructure agile et flexible.

Table des matières

Installation K3S

Installé k3s

curl -sfL https://get.k3s.io | sh -

Vérifiez que K3s fonctionne correctement

sudo systemctl status k3s

Vérifiez que vous pouvez utiliser kubectl

kubectl get nodes

Création du fichier de déploiement

Créez un fichier nommé goweb-deploy.yaml

nano goweb-deploy.yaml

Copiez le contenu suivant dans ce fichier

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    run: goweb
  name: goweb-deployment-rc
  namespace: default
spec:
  replicas: 4
  selector:
    matchExpressions:
      - {key: run, operator: In, values: [goweb]}
      - {key: environment, operator: NotIn, values: [production]}
  strategy:
    rollingUpdate:
      maxSurge: 15%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      labels:
        run: goweb
        environment: dev
    spec:
      containers:
      - image: gr0unz/goweb:V1
        imagePullPolicy: Always
        name: goweb
        ports:
        - containerPort: 8080
          protocol: TCP
```

Création du déploiement

Appliquez le fichier de déploiement

sudo kubectl create -f goweb-deploy.yaml

Vérification du déploiement

Vérifiez que le déploiement a été créé

sudo kubectl get deployments -n default

```
ubuntu22@ubuntu22-virtual-machine:~$ sudo kubectl get deployments -n default
NAME                 READY   UP-TO-DATE   AVAILABLE   AGE
goweb-deployment-rc  4/4     4            4           17h
```

Création du fichier de service

Créez un fichier nommé goweb-svc.yaml

nano goweb-svc.yaml

Copiez le contenu suivant dans ce fichier

```yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    run: goweb
  name: goweb-svc
  namespace: default
spec:
  externalTrafficPolicy: Cluster
  ports:
  - nodePort: 32520
    port: 80
    protocol: TCP
    targetPort: 8080
  selector:
    run: goweb
  sessionAffinity: None
  type: NodePort
```

Création du service

Appliquez le fichier de service

sudo kubectl create -f goweb-svc.yaml

Vérification du service

Vérifiez que le service a été créé

sudo kubectl get services -n default

```
ubuntu22@ubuntu22-virtual-machine:~$ sudo kubectl get services -n default
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
goweb-svc    NodePort    10.43.242.226   <none>        80:32520/TCP   17h
kubernetes   ClusterIP   10.43.0.1       <none>        443/TCP        17h
```

Vérification des pods

Vérifiez les pods créés par le déploiement

sudo kubectl get pods -n default

```
ubuntu22@ubuntu22-virtual-machine:~$ sudo kubectl get pods -n default
NAME                                  READY   STATUS    RESTARTS      AGE
goweb-deployment-rc-78d77ff877-n5gh4  1/1     Running   1 (20m ago)   17h
goweb-deployment-rc-78d77ff877-qck8t  1/1     Running   1 (20m ago)   17h
goweb-deployment-rc-78d77ff877-s52mq  1/1     Running   1 (20m ago)   17h
goweb-deployment-rc-78d77ff877-s6zrg  1/1     Running   1 (20m ago)   17h
```

Accès à l'application

Pour accéder à votre application, vous aurez besoin de l'adresse IP de votre nœud K3s. Obtenez-la avec

ip addr show

Cherchez l'adresse IP de votre interface réseau principale (souvent eth0 ou ens33). Ensuite, ouvrez un navigateur et accédez à

http://<ADRESSE_IP_DU_NOEUD>:32520

- Home
- News
- Contact
- About

Go Web Server Info

Hostname:

goweb-deployment-rc-78d77ff877-qck8t

Os Version

"Alpine Linux"

Interface: eth0

IP Address: 10.42.0.19/24

IP Address: fe80::6805:4cff:fe5d:1f1c/64

Hardware Address: 6a:05:4c:5d:1f:1c

Modification du nombre de réplicas

Pour changer le nombre de réplicas de 4 à 2

sudo kubectl scale deployment goweb-deployment-rc --replicas=2

Vérifiez que le nombre de pods a été réduit

sudo kubectl get pods -n default

```
ubuntu22@ubuntu22-virtual-machine:~$ sudo kubectl get pods -n default
NAME                                 READY   STATUS    RESTARTS      AGE
goweb-deployment-rc-78d77ff877-s52mq   1/1     Running   1 (28m ago)   17h
goweb-deployment-rc-78d77ff877-s6zrg   1/1     Running   1 (28m ago)   17h
```

Livrable : Déploiement Kubernetes

Définition de Kubernetes

Kubernetes est une plateforme open-source d'orchestration de conteneurs conçue pour automatiser le déploiement, la mise à l'échelle et la gestion des applications conteneurisées.

Architecture de Kubernetes

L'architecture de Kubernetes se compose de deux parties principales :

- Le plan de contrôle (Control Plane) : Gère l'état global du cluster.
- Les nœuds (Nodes) : Machines de travail qui exécutent les applications.

Le plan de contrôle comprend

- kube-apiserver
- etcd
- kube-scheduler
- kube-controller-manager

Chaque nœud comprend

- kubelet
- kube-proxy
- Container runtime (comme Docker)

Définition d'un Node

Un Node est une machine de travail dans Kubernetes, qui peut être une machine physique ou virtuelle. Les Nodes hébergent les Pods qui sont les composants de l'application.

Définition d'un Pod

Un Pod est la plus petite unité déployable dans Kubernetes. Il représente un processus en cours d'exécution dans votre cluster et peut contenir un ou plusieurs conteneurs.

Principales étapes pour réaliser le déploiement

- Création du fichier de déploiement YAML
- Application du fichier de déploiement avec `kubectl create -f`
- Création du fichier de service YAML
- Application du fichier de service avec `kubectl create -f`
- Vérification du déploiement et du service avec `kubectl get`
- Accès à l'application via l'adresse IP du nœud et le NodePort

Définition de haute disponibilité

La haute disponibilité se réfère à la capacité d'un système à fonctionner de manière continue sans interruption. Dans Kubernetes, cela est réalisé par la réplication des composants critiques et la distribution des charges de travail sur plusieurs nœuds.

Définition d'un fichier YAML

YAML (YAML Ain't Markup Language) est un format de sérialisation de données lisible par l'homme. Dans Kubernetes, les fichiers YAML sont utilisés pour définir les ressources comme les déploiements, les services, etc.

Attributs du ReplicaSet dans le fichier YAML

Les attributs du ReplicaSet présents dans le fichier YAML sont :

- `replicas`: Définit le nombre de réplicas du pod
- `selector`: Spécifie comment le ReplicaSet identifie les pods à gérer
- `template`: Définit le modèle pour créer de nouveaux pods

Image Docker utilisée

L'image Docker utilisée dans cet exercice, comme spécifiée dans le fichier YAML, est :

gr0unz/goweb:V1