LES SERVICES

**Exercice déployer une application Flask pour Kubernetes**

Ce projet consiste à déployer une application web en utilisant Kubernetes. L'application web est construite avec Python et Flask, utilisant Redis pour la mise en cache et MySQL comme base de données. Voici les étapes générales pour comprendre et déployer l'application:

**Étape 1 : Préparation de l'Environnement**

1. Cloner le dépôt Git : Commencez par cloner le dépôt contenant le code de l'application :

2. Construire l'image Docker : À partir du code source Python existant, vous construirez une image Docker.

**Étape 2 : Déploiement sur Kubernetes**

1. Pod de Base de Données : Déployez un pod MySQL en utilisant le fichier db-pod.yml.

2. Service de Base de Données : Créez un service Kubernetes pour MySQL avec db-svc.yml pour permettre aux autres pods de se connecter à la base de données.

3. Pod Web : Déployez le pod de l'application web utilisant web-pod.yml ou web-deployment.yaml pour gérer les instances multiples du pod.

4. Service Web : Configurez un service pour l'application web avec web-svc.yml pour exposer l'application à l'extérieur du cluster Kubernetes.

**Étape 3 : Vérification et Utilisation de l'Application**

1. Vérifiez les Pods et Services : Utilisez kubectl get pods et kubectl get svc pour vous assurer que tout est correctement déployé.

2. Accéder à l'Application : Utilisez kubectl describe svc web pour trouver l'adresse IP et le port pour accéder à l'application.

3. Initialiser la Base de Données : Faites une requête à l'endpoint /init pour initialiser la base de données.

4. Ajouter des Données : Utilisez curl pour envoyer des données à l'application, qui seront ajoutées à MySQL et mises en cache dans Redis.

5. Accéder aux Données : Faites des requêtes pour récupérer les données. Lorsque vous accédez aux données pour la deuxième fois, elles seront servies à partir du cache Redis (indiqué par '(c)').

**Informations Complémentaires**

- Le fichier app.py contient le code de l'application Flask, qui gère les routes pour initialiser la base de données, ajouter des utilisateurs et récupérer des données utilisateur.

- docker-compose.yml est utilisé pour définir et exécuter des applications multi-conteneurs avec Docker.

- Les fichiers YAML dans le répertoire deploy sont des spécifications pour déployer les composants de l'application dans Kubernetes.

Pour suivre ce tutoriel, vous devez avoir une compréhension de base de Docker, de Kubernetes, et des commandes de la ligne de commande Unix/Linux. Vous devez également avoir accès à un cluster Kubernetes, soit localement via kind, minikube, soit via un service cloud comme Google Kubernetes Engine (GKE), Amazon EKS, ou Azure AKS.

A diagram of a service

Description automatically generated

sudo -s

kind delete cluster

kind create cluster

kubectl cluster-info --context kind-kind

git clone https://github.com/gbhure/docker-demo.git

cd docker-demo/service-demo

cd build

docker-compose up -d (optionnel - docker build . -t hrehouma1/flaskapp:1.2)

docker images

docker tag redis:latest hrehouma1/redis:1.2

docker tag mysql:latest hrehouma1/mysql:1.2

docker tag your\_web\_image\_name:your\_tag hrehouma1/flaskapp:1.2

EXEMPLE:

docker images

docker tag build\_web hrehouma1/flaskapp:1.2

docker push hrehouma1/redis:1.2

docker push hrehouma1/mysql:1.2

docker push hrehouma1/flaskapp:1.2

docker ps

docker images

cd ../deploy

kubectl apply -f db-deployment.yaml

kubectl apply -f db-svc.yml

kubectl apply -f web-deployment.yaml

kubectl apply -f web-svc.yml

kubectl get pods

kubectl get svc

kubectl get deployments

kubectl logs web-58d5b669c5-nkgrj (nom à remplacer)

kubectl logs mysql (nom à remplacer)

export NODE\_IP=$(kubectl get nodes -o jsonpath='{.items[0].status.addresses[?(.type=="InternalIP")].address}')

export NODE\_PORT=$(kubectl get svc web -o jsonpath='{.spec.ports[0].nodePort}')

curl http://$NODE\_IP:$NODE\_PORT/init

curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "1", "user":"Aakash M"}' http://$NODE\_IP:$NODE\_PORT/users/add

curl http://$NODE\_IP:$NODE\_PORT/users/1

kubectl delete -f db-deployment.yaml

kubectl delete -f db-svc.yml

kubectl delete -f web-deployment.yaml

kubectl delete -f web-svc.yml

kind delete cluster

exit

exit

az group delete --name k8s-cluster --yes --no-wait