POLYTECHNIQUE MONTRÉAL

INF8480 - Systèmes répartis et infonuagique

TP6 - Service infonuagique tolérant aux pannes

Chargés de laboratoire :
Pierre-Frederick Denys

Automne 2020 - V4.0

1 Introduction

1.1 Prérequis

— **Processus serveurs pour l'infonuagique :** Services pour les machines virtuelles et conteneurs, les fichiers ou les applications. Exemples de Amazon EC2 et OpenStack, et de Kubernetes et Docker.
— Tolérance aux pannes. Modèles de pannes. Tolérance aux pannes par la réplication. Modèles de réplication. Ordonnancement des requêtes et cohérence.

1.2 But du TP

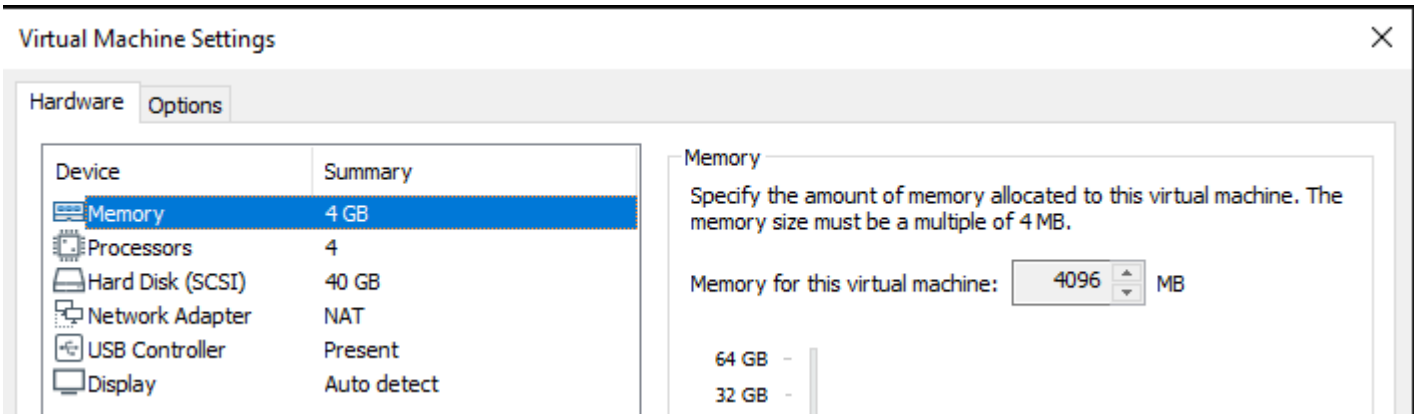
— Introduction à Kubernetes et à ses concepts
— Orchestration de conteneurs en production
— Mise en place d'un système tolérant aux pannes sur kubernetes

Le déploiement va s'effectuer sur une machine virtuelle comme cela se fait dans l'industrie. La machine virtuelle est en local sur vos machines, mais vous ne pouvez y accéder que par SSH.

Attention
Attention, il est crucial de bien lire l'énoncé en entier avant de commencer, afin de comprendre le but global du TP et connaître le but de chaque commande.

1.3 Mise en place de la VM

— Le TP est à réaliser sur votre ordinateur personnel, et fonctionne sur Windows, Mac et Linux. Vous n'avez pas de configuration particulière à faire et un seul logiciel à installer, qui est **VMWare**. Pour obtenir le logiciel, vous devez suivre ces instructions `https://www.polymtl.ca/gigl/sites/gigl.amigow.polymtl.ca/files/vmwareetudiantsgigl.pdf`. Vous n'avez pas besoin d'être sur le réseau de polytechnique, n'y d'utiliser le VPN. Vous obtiendrez une licence d'un an renouvelable pour les logiciels suivants :
  — Sur MAC, le logiciel VMware Fusion 11.x
  — Sur Windows, le logiciel VMware Workstation 15
  — Sur Linux, le logiciel VMware Workstation 15
— Vous pouvez également utiliser la version d'essai de VMware Worstation
— Vous devez télécharger l'archive de la machine virtuelle en cliquant sur **télécharger** en haut de l'écran du lien disponible sur Moodle.
— Lancer VMware, et cliquer sur `file>open` et choisir le fichier `.ovf`.
— La machine virtuelle lancée, elle affiche son IP, **vous devez utiliser le nom d'utilisateur `inf8480`** (et pas `ubuntu`) le fichier `vminf8480_key` (fichier de clé privé disponible sur moodle) pour vous connecter en ssh sur la machine (option -i comme dans les tp précédents). Vous **ne pouvez pas** vous connecter directement dans l'interface de VMWare.



Attention

Si votre PC dispose de moins de 16G de RAM, vous pouvez baisser la RAM allouée à la VM à 4GB avant de la démarrer dans les paramètres de celle-ci.

2 Partie 1 : Découverte de Kubernetes et mise en place d'un cluster

Avant de commencer le TP, répondez aux questions du quiz moodle afin de vérifier que vous ayez compris les différents éléments de Kubernetes.

2.1 Objectif

L'installation d'un cluster kubernetes (k8s) implique normalement l'installation de nombreux composants sur des noeuds différents. Pour faciliter l'installation et les tests, **microk8s** est un package "all-in-one" utilisable sur un seul noeud, et qui permet d'avoir un cluster openstack opérationnel en quelques minutes.

2.2 Installer microk8s

Connectez vous à la vm en effectuant un tunnel sur le port 8001 (la raison sera expliquée par la suite) : **(commande à exécuter dans une console de votre PC)**

```
ssh -L 8001:localhost:8001 -i <fichier_cle_privee> inf8480@<ip_vm>
```

Installer le paquet `microk8s` : (snap est un gestionnaire de paquet permettant d'installer des applications avec toutes leurs dépendances incluses)

```
sudo apt-get update &&  sudo apt-get upgrade
sudo apt-get install snapd
sudo snap install microk8s --classic
```

```
sudo usermod -a -G microk8s inf8480 #permet d'ajouter notre
   utilisateur au groupe microk8s
sudo chown -f -R inf8480 ~/.kube
```

Déconnectez vous de la console ssh, puis re-connectez vous.

2.3 Configurer microk8s

Voir la liste des services disponibles :

```
microk8s.enable --help
```

Activer les différents services :

```
microk8s.enable dashboard dns istio registry storage ingress
```

Démarrer le cluster :

```
microk8s.start
```

Visualiser les différents services (ressemble à docker ps) :

```
microk8s.kubectl get all --all-namespaces
```

Un dashboard permet de monitorer le cluster kubernetes, autoriser la connexion au travers d'un proxy :

```
microk8s.kubectl proxy &
```

et se connecter à l'adresse :

```
http://localhost:8001/api/v1/namespaces/kube-system/services/https:
   kubernetes-dashboard:/proxy/
```

Récupérer le token associé :

```
 token=$(microk8s.kubectl -n kube-system get secret | grep default-
    token | cut -d " " -f1)
microk8s.kubectl -n kube-system describe secret $token
```

Dans la fenêtre d'authentification, choisir le bouton token, Et copier-coller le token obtenu dans la commande précédente.

Vous devriez avoir accès au dashboard kubernetes.

Le dashboard vous permet de visualiser l'état en temps réel de votre cluster, et d'en effectuer le monitoring. Ce dashboard qui est personnalisable et est souvent utilisé par des systèmes en production.

3 Partie 2 : mise en place d'un service web sur Kubernetes

Un Ingress peut fournir un équilibrage de charge, une terminaison TLS et un hébergement virtuel basé sur un nom.

3.1 Premier déploiement

Déployer une application sur le cluster kubernetes :

```
microk8s.kubectl run tp6-web --image=gcr.io/google-samples/hello-app
    :1.0 --port=8080
```

Exposer le déploiement :

```
microk8s.kubectl expose pod/tp6-web --target-port=8080 --type=
    NodePort
```

Dans le dashboard, chercher l'ip du service que vous venez de créer, et lancer dans la console la commande : (installer `curl`)

```
curl http://<ip_service>:8080
```

Elle devrait vous retourner un message suivi du nom d'hôte. Vous pouvez également obtenir l'ip avec la commande :

```
microk8s.kubectl get service tp6-web
```

3.2 Second déploiement

Déployer une seconde application sur le cluster kubernetes :

```
microk8s.kubectl run tp6-web2 --image=gcr.io/google-samples/hello-app
    :2.0 --port=8080
```

Exposer le déploiement :

```
microk8s.kubectl expose pod/tp6-web2 --target-port=8080 --type=
    NodePort
```

Tester le fonctionnement du second déploiement avec la commande :

```
curl http://<ip_service>:8080/v2
```

3.3 Service Ingress

Pour déployer un service de type ingress, **compléter** le ficher suivant `tp6-ingress.yaml` **(fourni sur Moodle pour plus de facilité)** :

```
apiVersion: networking.k8s.io/v1beta1
kind:
metadata:
  name: tp6-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$1
spec:
  rules:
  - host: web-tp6.info
    http:
      paths:
      - path: /

      - path: /v2/*
```

Déployer le service avec les commandes suivantes :

```
microk8s.kubectl apply -f tp6-ingress.yaml
```

Ajouter ensuite l'entrée `web-tp6.info` à votre fichier `/etc/hosts sur votre VM`.

```
127.0.0.1 web-tp6.info
```

Vous pourrez ensuite exécuter :

```
curl web-tp6.info:80
curl web-tp6.info:80/v2/
```

Vous venez de déployer une application web avec service de type ingress.

3.4 Service Ingress avec répartition de charge

Maintenant que vous êtes en mesure de déployer des services et déploiements avec kubernetes, vous allez créer un service web avec répartition de charge (et donc résilience aux pannes) composé de 5 pods.

Commencer par créer un service avec une application tournant dans 5 pods `service-5pods.yaml` : **(fourni sur Moodle pour plus de facilité)**

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app.kubernetes.io/name: load-balancer-example
  name: hello-world
spec:
  replicas: 5
  selector:
    matchLabels:
      app.kubernetes.io/name: load-balancer-example
  template:
```

```
    metadata:
      labels:
        app.kubernetes.io/name: load-balancer-example
    spec:
      containers:
      - image: gcr.io/google-samples/hello-app:1.0
        name: hello-world
        ports:
        - containerPort: 8080
```

Lancer ensuite le déploiement, et exposez le : **(ignorer le warning à la sortie de la première commande)**

```
microk8s.kubectl apply -f service-5pods.yaml
microk8s.kubectl get deployments hello-world
microk8s.kubectl expose deployment hello-world --type=LoadBalancer --
   name=my-service
```

Vous pouvez voir l'avancée de votre déploiement avec la commande suivante. (attendre que tous les pods soient up).

```
microk8s.kubectl get deployment
microk8s.kubectl describe deployments hello-world
```

Vous pouvez si besoin éditer le déploiement avec la commande :

```
microk8s.kubectl edit deployment.v1.apps/hello-world
```

Vous pouvez voir la liste des pods avec la commande :

```
microk8s.kubectl get pods --output=wide
```

Exécuter la commande afin d'associer une ip externe (celle de la vm) au service (normalement, c'est un service de load balancing externe qui effectue cette tache automatiquement, plus d'infos içi : `https://8gwifi.org/docs/kube-debug.jsp`). **Pensez à remplacer `<ip_vm>`!!!!.**

```
microk8s.kubectl patch svc my-service  -p '{"spec": {"type": "
   LoadBalancer", "externalIPs":["<ip_vm>"]}}'service/my-service
```

Exécuter plusieurs fois la commande suivante sur la VM ou accéder à l'adresse `http://<ip_vm>:8080/` depuis votre fureteur en forçant l'actualisation de la page avec CTRL + F5.

```
curl <ip_vm>:8080
```

Vous devriez voir le hostname changer, qui signifie qu'un pod différent vous répond à chaque fois. Bravo ! vous venez de créer un service avec répartition de charge !

Ainsi, dans une infrastructure réelle, comme les pods sont répartis entre plusieurs machines physiques (worker), il y a une grande tolérence aux pannes, qui est renforcée par la présence de plusieurs noeuds manager.

4 Remise

Exécuter **SUR LA VM** le script de correction (que vous pouvez copier à l'aide de la commande scp) avec le code header obtenu dans la question 1 du quiz. Si le fonctionnement de votre serveur de fichier est correct, le hash obtenu permet de valider la seconde question du quiz. **(ignorer le warning)**

```
./correction-tp6.sh.x  code_header_moodle
```

♢