Chapitre 1 - Installation

Installation avec kubeadm

Installation avec kubeadm

Installation de Docker

Pré-requis

Mettre à jour les fichiers /etc/hosts des 3 nodes

```
vi /etc/hosts
192.168.56.101 node1
192.168.56.102 node2
192.168.56.103 node3
```

Installation de Docker

Installer des packages pré-requis pour Docker

```
sudo yum install -y yum-utils net-tools
```

Ajouter le fichier repository sur les noeuds

```
sudo yum-config-manager  --add-repo   https://download.docker.com/linux/centos/docker-ce.repo
```

Installer les packages docker version 19.03.9 sur les 3 noeuds

```
sudo yum install docker-ce-19.03.9 docker-ce-cli-19.03.9 containerd.io -y
```

Démarrer et activer le daemon

```
sudo systemctl start docker
sudo systemctl enable docker
```

Ajouter le user "centos" au group "docker"

```
sudo usermod -aG docker centos
```

Passer en tant que user "centos" et vérifier la version de docker

```
su - centos
docker version
```

Installation de Kubernetes

Pré-requis

Sur le Master: Ouvrir les ports réseau requis

```
sudo firewall-cmd --permanent --add-port={6443,2379,2380,10250,10251,10252}/tcp
sudo firewall-cmd --reload
```

Sur les Workers: Ouvrir les ports réseau requis

```
sudo firewall-cmd --permanent --add-port={10250,30000-32767}/tcp
sudo firewall-cmd --reload
```

Sur tous les noeuds: Ouvrir les ports réseau pour Calico

```
sudo firewall-cmd --permanent --add-port={179,5473,443}/tcp
sudo firewall-cmd --reload
```

Sur tous les noeuds: Autoriser le trafic en mode bridge (exécuter en tant que root)

```
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
br_netfilter
EOF

cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
sudo sysctl --system
```

Sur tous les noeuds: ajouter le repository (exécuter en tant que root)

```
cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-\$basearch
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
exclude=kubelet kubeadm kubectl
EOF

# Set SELinux in permissive mode (effectively disabling it)
sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config

sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
sudo systemctl enable --now kubelet
```

Installation du Master

Sur le Master: Disable l swap

```
swapoff -a
sed -e '/swap/s/^/#/g' -i /etc/fstab
```

Sur le Master: initialiser le cluster Kubernetes

```
kubeadm init --apiserver-advertise-address=192.168.56.101 --pod-network-cidr=192.168.0.0/16
```

Transferer le fichier du config sur le poste client de l'utilisateur

```
scp /etc/kubernetes/admin.conf centos@poste:/home/centos/.kube/config
```

Installation de Calico

Sur le Client: Appliquer la spécification de Calico

```
kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml
```

Chapitre 2 – Configuration

5. Fichier de config

5.1 Ajout de fichier de config dans le HOME

Transférer le fichier de configuration

```
scp root@node1:/etc/kubernetes/admin.conf .kube/config
```

Visualier la configuration

```
kubectl config view
```

6. Variable d'environnement

6.1 KUBECONFIG

Créer la variable d'environnement avec les 2 fichiers de config des 2 clusters

```
export KUBECONFIG=/home/centos/.kube/config:/home/centos/.kube/admin.conf
```

Visualier la configuration

```
kubectl config view
```

7. Fusion de plusieurs fichiers de config

Créer la variable d'environnement avec les 2 fichiers de config des 2 clusters

```
export KUBECONFIG=/home/centos/.kube/config:/home/centos/.kube/admin.conf
```

Créer un fichier avec le mode raw de config view

```
kubectl config view --raw > new-config
mv new-config .kube/config
unset KUBECONFIG
```

Version Kubectl

```
PS C:\Users\pc> kubectl version
Client Version: version.Info{Major:"1", Minor:"20", GitVersion:"v1.20.0", GitCommit:"af46c47ce925f4c4ad5cc8d1fca46c7b77d13b3
8", GitTreeState:"clean", BuildDate:"2020-12-08T17:59:43Z", GoVersion:"go1.15.5", Compiler:"gc", Platform:"windows/amd64"}
Server Version: version.Info{Major:"1", Minor:"20", GitVersion:"v1.20.4", GitCommit:"e87da0bd6e03ec3fea7933c4b5263d151aafd07
c", GitTreeState:"clean", BuildDate:"2021-02-18T16:03:00Z", GoVersion:"go1.15.8", Compiler:"gc", Platform:"linux/amd64"}
PS C:\Users\pc>
```

Visualiser les nœuds

```
PS C:\Users\pc> kubectl get nodes
NAME           STATUS     ROLES                  AGE   VERSION
host13         NotReady   <none>                 88d
master-node1   Ready      control-plane,master   90d   v1.20.4
node2          Ready      <none>                 89d   v1.20.4
node3          Ready      <none>                 89d   v1.20.4
```

Rappel du travail demandé :

Producteur

1. **Créer un daemonSet nommé producteur basé sur l'image alpine**
2. **Créer un volume**
3. **Monter le volume sur le conteneur**
4. **Ecrire dans le fichier index.html le nom du hostname et la date toutes les 60 secondes**

Consommateur

1. **Créer un deploiement nommé web (avec 3 replicas) basé sur l'image httpd**
2. **Monter le volume créé précédemment sur le chemin htdocs du serveur Apache**

Service NodePort

1. Créer un service nommé web de type NodePort qui rassemble les pods du deploiement web

Utilisation

1. **Depuis le poste de travail, exécuter toutes les minutes commande "curl node1:30000"**

Producteur

1. **Créer un daemonSet nommé producteur basé sur l'image alpine**

Daemon.set.yml

```yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: producteur
spec:
  selector:
    matchLabels:
      producteur: daemon
  template:
    metadata:
      labels:
        producteur: daemon
    spec:
      containers:
      - name: alpine-producteur
        image: alpine:3.12
        command: ["sleep","3600"]
```

Exécuter la commande : **kubectl apply -f .\daemonset.yml**

```
PS C:\Users\pc\Documents\Mastère ESI\Kubernetes\Exam_Final_K8s> kubectl apply -f .\daemonset.yml
daemonset.apps/producteur created
```

Visualiser le résultat : **kubectl get ds,pods -o wide**

```
PS C:\Users\pc\Documents\Mastère ESI\Kubernetes\Exam_Final_K8s> kubectl get ds,pods -o wide
NAME                          DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE   CONTAINERS          IMAGES         SELECTOR
daemonset.apps/producteur     2         2         2       1            2           <none>          15m   alpine-producteur   alpine:3.12    producteur=daemon

NAME                    READY   STATUS        RESTARTS   AGE     IP              NODE    NOMINATED NODE   READINESS GATES
pod/producteur-lnz2v    1/1     Running       0          9s      192.168.0.177   node3   <none>           <none>
pod/producteur-m2nbn    1/1     Terminating   0          3m29s   192.168.0.27    node2   <none>           <none>
```

2. **Créer un volume**

Création d'un fichier yml nommé **volume.yml** et y mettre le code suivant :

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: producteur-pv-volume
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/data"
```

Pour appliquer, il faut exécuter la commande : **kubectl apply -f .\volume.yml**

```
PS C:\Users\pc\Documents\Mastère ESI\Kubernetes\Exam_Final_K8s> kubectl apply -f .\volume.yml
persistentvolume/volume-producteur created
```

Pour visualiser le résultat : **kubectl get pv**

```
PS C:\Users\pc\Documents\Mastère ESI\Kubernetes\Exam_Final_K8s> kubectl get pv
NAME                CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   REASON   AGE
volume-producteur   10Gi       RWO            Retain           Available           manual                  8m15s
```

3. Monter le volume sur le conteneur

On va copier la partie ci-dessous dans le fichier daemonset.yml à la suite de la configuration du container alpine-producteur

```yaml
volumeMounts:
        - name: volume-producteur
          mountPath: /prod
      volumes:
      - name: volume-producteur
        hostPath:
          path: /prod
```

Ce qui nous donne :

```yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: producteur
spec:
  selector:
    matchLabels:
      producteur: daemon
  template:
    metadata:
      labels:
        producteur: daemon
    spec:
      containers:
      - name: alpine-producteur
        image: alpine:3.12
        command: ["sleep","3600"]
        volumeMounts:
        - name: volume-producteur
          mountPath: /prod
      volumes:
      - name: volume-producteur
        hostPath:
          path: /prod
```

Pour visualiser : **kubectl get pods -o wide**

```
PS C:\Users\pc\Documents\Mastère ESI\Kubernetes\Exam_Final_K8s> kubectl get pods -o wide
NAME                READY   STATUS    RESTARTS   AGE     IP              NODE    NOMINATED NODE   READINESS GATES
producteur-b5zkm    1/1     Running   0          7m19s   192.168.0.30    node2   <none>           <none>
producteur-lnz2v    1/1     Running   0          7m59s   192.168.0.177   node3   <none>           <none>
```

Pour vérifier que le volume est bien monté sur les pods : **kubectl exec -it producteur-b5zkm – sh**

kubectl exec -it producteur-lnz2v – sh puis faire un **ls** à la racine

```
PS C:\Users\pc\Documents\Mastère ESI\Kubernetes\Exam_Final_K8s> kubectl get pods -o wide
NAME                READY   STATUS    RESTARTS   AGE     IP              NODE    NOMINATED NODE   READINESS GATES
producteur-b5zkm    1/1     Running   0          11m     192.168.0.30    node2   <none>           <none>
producteur-lnz2v    1/1     Running   0          12m     192.168.0.177   node3   <none>           <none>
PS C:\Users\pc\Documents\Mastère ESI\Kubernetes\Exam_Final_K8s> kubectl exec -it producteur-b5zkm -- sh
/ # ls
bin    etc    lib    mnt    proc   root   sbin   sys    usr
dev    home   media  opt    prod   run    srv    tmp    var
/ # exit
PS C:\Users\pc\Documents\Mastère ESI\Kubernetes\Exam_Final_K8s> kubectl exec -it producteur-lnz2v -- sh
/ # ls
bin    etc    lib    mnt    proc   root   sbin   sys    usr
dev    home   media  opt    prod   run    srv    tmp    var
/ # exit
```

4. **Ecrire dans le fichier index.html le nom du hostname et la date toutes les 60 secondes**

Pour vérification que sur le conteneur : vérifier que dans le dossier **prod** il y a un fichier **index.html**

```
PS C:\Users\pc\Documents\Mastère ESI\Kubernetes\Exam_Final_K8s> kubectl get pods
NAME                    READY   STATUS    RESTARTS   AGE
producteur-f8tct        1/1     Running   0          3s
producteur-rcdtr        1/1     Running   0          37s
```

```
PS C:\Users\pc\Documents\Mastère ESI\Kubernetes\Exam_Final_K8s> kubectl exec -it producteur-f8tct -- sh
/ # ls
bin    etc    lib    mnt    proc   root   sbin   sys    usr
dev    home   media  opt    prod   run    srv    tmp    var
/ # cat prod/index.html
Bonjour producteur-zpwwt, nous sommes le Wed Mar 10 10:29:13 UTC 2021
Bonjour producteur-f8tct, nous sommes le Wed Mar 10 10:29:13 UTC 2021
```

Consommateur

1. **Créer un déploiement nommé web (avec 3 réplicas) basé sur l'image httpd**
2. **Monter le volume créé précédemment sur le chemin htdocs du serveur Apache**

Création d'un ficher **deploy.yml** contenant les instructions suivantes (contient le montage du volume):

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web
spec:
  replicas: 3
  selector:
    matchLabels:
      app3: web
  template:
    metadata:
      labels:
        app3: web
    spec:
      containers:
      - name: apache
        image: httpd:2.4-alpine
        #Monter le volume sur le chemin htdocs du serveur Apache
```

```
        volumeMounts:
        - name: volume-producteur
          mountPath: /usr/local/apache2/htdocs/prod
      volumes:
      - name: volume-producteur
        hostPath:
          path: /prod
```

Vérification avec la commande : **kubectl get ds,pods -o wide** (avec l'apparition de 3 nouveaux pods

```
PS C:\Users\pc\Documents\Mastère ESI\Kubernetes\Exam_Final_K8s> kubectl get ds,pods -o wide
NAME                         DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE   CONTAINERS
       IMAGES         SELECTOR
daemonset.apps/producteur    2         2         2       2            2           <none>          53m   alpine-produ
cteur    alpine:3.12    producteur=daemon

NAME                       READY   STATUS    RESTARTS   AGE   IP              NODE    NOMINATED NODE   READINESS GA
TES
pod/producteur-5f6vp       1/1     Running   0          16m   192.168.0.34    node2   <none>           <none>
pod/producteur-dg2wp       1/1     Running   0          17m   192.168.0.179   node3   <none>           <none>
pod/web-86c8df459c-gbdb2   1/1     Running   0          68s   192.168.0.182   node3   <none>           <none>
pod/web-86c8df459c-kvgm7   1/1     Running   0          68s   192.168.0.180   node3   <none>           <none>
pod/web-86c8df459c-rctkf   1/1     Running   0          68s   192.168.0.181   node3   <none>           <none>
```

Vérification du montage du volume /prod dans les containers : **kubectl exec -it pod/web-86c8df459c-gbdb2 -- sh**

```
PS C:\Users\pc\Documents\Mastère ESI\Kubernetes\Exam_Final_K8s> kubectl exec -it pod/web-86c8df459c-gbdb2 -- sh
/usr/local/apache2 # ls
bin      build    cgi-bin  conf     error    htdocs   icons    include  logs     modules
/usr/local/apache2 # ls htdocs/prod/
index.html
```

Service NodePort

1. **Créer un service nommé web de type NodePort qui rassemble les pods du deploiement web**

```
apiVersion: v1
kind: Service
metadata:
  name: web
spec:
  selector:
    app: web
  type: NodePort
```

```
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
      nodePort: 30002
```

Exécuter : **kubectl apply -f .\service.yml**

```
PS C:\Users\pc\Documents\Mastère ESI\Kubernetes\Exam_Final_K8s> kubectl apply -f .\service.yml
service/web created
```

Verification **: kubectl get service -o wide**

```
NAME   TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
web    NodePort   10.105.48.145   <none>        80:30002/TCP   3m25s
```

Utilisation

1. **Depuis le poste de travail, exécuter toutes les minutes commande "curl node1:30002"**

Créer un script qui execute un curl toutes les minutes **curl.ps1**

```
while (true)
{
    Invoke-WebRequest 127.0.0.1:30002
    Start-Sleep 60
}
```

Exécuter le script