Mise en situation professionnelle	**EasyLinux**
	Cours : Kubernetes
Sujet : Deployer une application	Numéro : 5 à 10 Version : 1.0

Objectifs :

Préparer votre poste de travail

Prérequis :

aucun

Principales tâches à réaliser :

11 Cluster

11.1 Installer les serveurs GlusterFS

Démarrer la machine Gateway, configurer ses interfaces réseaux. (vboxnet sur 50)
Créer 3 clones liés de la machine Gluster-Srv, Configurer la carte réseau et ajouter un disque dur
Lancer Install et suivre les instruction

11.2 Installer le master

```
sudo kubeadm init --apiserver-advertise-address 192.168.50.200 --service-dns-domain
orsys.lan --pod-network-cidr=10.244.0.0/16
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Couche réseau

```
kubectl apply -f weave-kube.yaml
```

Nous avons un petit cluster, on ne va pas se passer du master, autoriser le master à scheduler

```
kubectl taint nodes master node-role.kubernetes.io/master-
```

NB :

Copier la ligne :

```
kubeadm join 192.168.50.200:6443 --token sh0mk3.ax38kryztq7z7044 --discovery-token-ca-
cert-hash sha256:115bf0abb1b4c686da05bfc024bb68b1ca7a27ae77eb0664e2faf3b62e58b1b3
```

Autoriser notre registre local

```
# mkdir -p certs.d/registry.formation.local:5000
# cd /etc/docker/certs.d/registry.formation.local:5000
# wget http://registry.formation.local/certs/ca.crt
```

Installer le serveur Nfs sur le serveur :

apt-get install nfs-server

mkdir /data

11.3 Installer les minions

Faire la même chose sur les 3 minions

Faire un clone lié de K8s-Minion

Le renommer

```
$ sudo vi /etc/hostname
$ cat /etc/hostname
Minion-1
```

Modifier le fichier /etc/hosts

$ cat /etc/hosts

```
127.0.0.1    Minion-1
127.0.0.1    localhost
127.0.1.1    Minion-1

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
192.168.50.200  Master
192.168.50.201  Minion-1
192.168.50.202  Minion-2
192.168.50.203  Minion-3
192.168.50.205  Minion
```

Modifier l'adresse ip

```
$ sudo vi /etc/network/interfaces
$ cat /etc/network/interfaces
# This file describes the network interfaces available on your system
...
auto eth1
iface eth1 inet static
      address 192.168.50.201
      netmask 255.255.255.0
#VAGRANT-END
```

Relancer la machine,

Joindre le cluster

```
sudo kubeadm join 192.168.50.200:6443 --token sh0mk3.ax38kryztq7z7044 --discovery-token-
ca-cert-hash sha256:115bf0abb1b4c686da05bfc024bb68b1ca7a27ae77eb0664e2faf3b62e58b1b3
```

Autoriser le registre local, ajouter l'adresse IP du formateur

```
$ vi /etc/hosts
```

Puis copier le certificat

```
$ sudo mkdir -p /etc/docker/certs.d/registry.formation.local:5000
$ su -
Password:
# cd /etc/docker/certs.d/registry.formation.local\:5000/
# wget registry.formation.local/certs/ca.crt
--2019-03-25 12:26:06--  http://registry.formation.local/certs/ca.crt
Resolving registry.formation.local (registry.formation.local)... 192.168.1.36
Connecting to registry.formation.local (registry.formation.local)|192.168.1.36|:80...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 2074 (2.0K) [application/x-x509-ca-cert]
Saving to: ‘ca.crt’

ca.crt                              100%
[=============================================================>]   2.03K  --.-KB/s    in
0s

2019-03-25 12:26:06 (260 MB/s) - ‘ca.crt’ saved [2074/2074]
```

11.4 Validation

Vous pouvez suivre le bon avancement de l'installation :

```
$ watch -n 2 kubectl get pod -n kube-system -o wide
```

puis

```
$ kubectl get nodes
NAME           STATUS   ROLES    AGE    VERSION
minion-1       Ready    <none>   109s   v1.13.4
master         Ready    master   23m    v1.13.4
```

Puis lancer un deploiement

```
$ kubectl create deployment kuard --image=registry.formation.local:5000/easylinux/kuard
deployment.apps/kuard created
$ kubectl expose deployment/kuard --type="NodePort" --port 8080
service/kuard exposed
$ kubectl get pod
NAME                     READY   STATUS    RESTARTS   AGE
kuard-76756c9d86-5vlds   1/1     Running   0          71s
$ kubectl scale deployment/kuard --replicas='3'
deployment.extensions/kuard scaled
$ kubectl get pod
NAME                     READY   STATUS              RESTARTS   AGE
kuard-76756c9d86-5vlds   1/1     Running             0          116s
kuard-76756c9d86-f4z2l   0/1     ContainerCreating   0          4s
kuard-76756c9d86-p8xmn   0/1     ContainerCreating   0          4s
```

11.5 GlusterFS

Pour nos stockages réseaux, nous allons installer GlusterFS

a Endpoints

```
apiVersion: v1
kind: Endpoints
metadata:
  name: gluster-endpoints
subsets:
- addresses:
  - ip: 192.168.50.210
  ports:
  - port: 1
    protocol: TCP
- addresses:
  - ip: 192.168.50.211
  ports:
  - port: 1
    protocol: TCP
- addresses:
  - ip: 192.168.50.212
  ports:
  - port: 1
    protocol: TCP
```

b Service glusterFS

```
apiVersion: v1
kind: Service
metadata:
  name: gluster-service
spec:
  ports:
- port: 1
```

c PersistentVolume

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: gluster-pv
spec:
  capacity:
    storage: 1Gi
  accessModes:
  - ReadWriteMany
  glusterfs:
    endpoints: gluster-endpoints
    path: /Vol0
    readOnly: false
  persistentVolumeReclaimPolicy: Retain
```

d persistentVolumeClaim

Pour utiliser le volume créé :

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gluster-claim
spec:
  accessModes:
  - ReadWriteMany
  resources:
     requests:
       storage: 1Gi
```