Dans cet exercice, vous allez mettre en place un cluster Kubernetes à l'aide de l'utilitaire *kubeadm*.

Création des VMs

Afin de créer les VMs qui seront utilisées dans le cluster, vous allez utiliser Multipass, un outil qui permet de créer des machines virtuelles très simplement.

Installez Multipass (celui-ci est disponible pour Windows, Linux et MacOS) puis lancez les commandes suivantes pour créer 2 machines virtuelles nommées *master* et *worker*

```
multipass launch -n master
multipass launch -n worker
```

Installation de Kubectl

Assurez-vous d'avoir installé *kubectl* sur la machine depuis laquelle vous avez lancé les commandes Multipass. *kubectl* permet de communiquer avec un cluster Kubernetes depuis la ligne de commande. Note: vous pouvez vous reporter à l'exercice installation de kubectl pour l'installation de *kubectl*.

Initialisation du cluster

Une fois les VMs *master* et *worker* créées, vous allez initialiser le cluster.

Lancez tout d'abord un shell sur le node *master*:

```
multipass shell master
```

Depuis ce shell lancez la commande suivante, celle-ci installe les dépendances nécessaires sur le master (container runtime et quelques packages)

```
ubuntu@master:~$ curl -sSL https://luc.run/kubeadm/latest/master.sh | bash
```

Lancez ensuite la commande qui initialize le cluster:

```
ubuntu@master:~$ sudo kubeadm init --ignore-preflight-errors=NumCPU,Mem
```

Note: le flag *--ignore-preflight-errors* permet de forcer l'installation même si les requirements en terme de CPU et Memoire ne sont pas respectés (cette option ne sera évidememnt pas utilisée pour la mise en place d'un cluster de production)

Après quelques dizaines de secondes, vous obtiendrez alors une commande qui vous servira, par la suite, à ajouter un node worker au cluster qui vient d'être créé.

Exemple de commande retournée (les tokens que vous obtiendrez seront différents):

```
sudo kubeadm join 192.168.64.40:6443 --token xrtqvq.9zmmzjx16b4jc4q8 --discovery-token
```

Toujours depuis le node master, récupérer le fichier kubeconfig pour l'utilisateur courant (*ubuntu*):

```
ubuntu@master:~$ mkdir -p $HOME/.kube
ubuntu@master:~$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
ubuntu@master:~$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Ajout d'un node worker

Une fois le cluster initialisé, vous allez ajouter un node worker.

Lancez tout d'abord un shell sur *worker*:

```
multipass shell worker
```

Depuis ce shell lancez la commande suivante, celle-ci installe les dépendances nécessaires sur le worker:

```
ubuntu@worker:~$ curl -sSL https://luc.run/kubeadm/latest/worker.sh | bash
```

Lancez ensuite la commande retournée lors de l'étape d'initialisation (*sudo kubeadm join* ...) afin d'ajouter le node *worker* au cluster.

```
sudo kubeadm join 192.168.64.40:6443 --token xrtqvq.9zmmzjx16b4jc4q8 --discovery-token
```

Note: si vous avez perdu la commande d'ajout de node, vous pouvez la générer avec la commande suivante (à lancer depuis le node master)

```
ubuntu@master:~$ sudo kubeadm token create --print-join-command
```

Après quelques dizaines de secondes, vous obtiendrez rapidement une confirmation indiquant que la VM *worker* fait maintenant partie du cluster::

```
This node has joined the cluster
* Certificate signing request was sent to apiserver and a response was received.
```

```
* The Kubelet was informed of the new secure connection details.
...
```

Etat du cluster

Listez à présent les nodes du cluster avec la commande suivante:

```
ubuntu@master:~$ kubectl get nodes
NAME     STATUS     ROLES                  AGE     VERSION
master   NotReady   control-plane,master   6m12s   v1.23.3
worker   NotReady   <none>                 2m28s   v1.23.3
```

Les nodes sont dans l'état *NotReady*, cela vient du fait qu'aucun plugin network n'a été installé pour le moment.

Plugin network

Afin que le cluster soit opérationnel il est nécessaire d'installer un plugin network. Plusieurs plugins sont disponibles (WeaveNet, Calico, Flannel, Cilium, ...), chacun implémente la spécification CNI (Container Network Interface) et permet notamment la communication entre les différents Pods du cluster.

Dans cet exercice, vous allez installer le plugin WeaveNet. Utilisez pour cela La commande suivante:

```
ubuntu@master:~$ kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kub
```

Plusieurs ressources sont alors créées pour mettre en place cette solution de networking:

```
serviceaccount/weave-net created
clusterrole.rbac.authorization.k8s.io/weave-net created
clusterrolebinding.rbac.authorization.k8s.io/weave-net created
role.rbac.authorization.k8s.io/weave-net created
rolebinding.rbac.authorization.k8s.io/weave-net created
daemonset.apps/weave-net created
```

Note: l'article suivant donne une bonne comparaison des plugins network les plus utilisées: https://objectif-libre.com/fr/blog/2018/07/05/comparatif-solutions-reseaux-kubernetes/, celui-ci effectue un benchmark des différentes solutions https://itnext.io/benchmark-results-of-kubernetes-network-plugins-cni-over-10gbit-s-network-updated-august-2020-6e1b757b9e49

Après quelques secondes, les nodes apparaitront dans l'état *Ready*

```
ubuntu@master:~$ kubectl get nodes
NAME     STATUS   ROLES                  AGE     VERSION
master   Ready    control-plane,master   11m     v1.23.3
worker   Ready    <none>                 7m33s   v1.23.3
```

Le cluster est prêt à être utilisé.

Récupération du context

Afin de pouvoir dialoguer avec le cluster via le binaire *kubectl* que vous avez installé sur votre machine locale, il est nécessaire de récupérer le fichier de configuration généré lors de l'installation.

Pour cela, il faut récupérer le fichier */etc/kubernetes/admin.conf* présent sur le node master et le copier sur votre machine locale.

Avec Multipass vous pouvez récupérer le fichier de configuration avec la commande suivante (il sera alors sauvegardé dans le fichier *kubeconfig* du répertoire courant):

```
multipass exec master -- sudo cat /etc/kubernetes/admin.conf > kubeconfig
```

Une fois que le fichier est présent en local, il faut simplement indiquer à *kubectl* ou il se trouve en positionnant la variable d'environnement *KUBECONFIG*:

```
export KUBECONFIG=$PWD/kubeconfig
```

Listez une nouvelle fois les nodes du cluster.

```
$ kubectl get nodes
NAME     STATUS   ROLES                  AGE   VERSION
master   Ready    control-plane,master   13m   v1.23.3
worker   Ready    <none>                 10m   v1.23.3
```

Vous pouvez à présent communiquer avec le cluster depuis la machine locale et non depuis une connexion ssh sur le master.

En résumé

Le cluster que vous avez mis en place dans cet exercice contient un node master et 1 node worker. Il est également possible avec kubeadm de mettre en place un cluster HA avec plusieurs nodes master en charge de la gestion de l'état du cluster. Kubeadm est l'une des référence pour la mise en place et la gestion de cluster en environnement de production.