Installation k8s

Il existe différentes solutions pour installer k8s. La méthode choisie consiste à utiliser l'outil kubeadm pour nous aider à configurer les différents outils nécessaires.
Plutôt que d'exécuter toutes les commandes d'installation de k8s dans une console, je vous les ai rassemblés dans des scripts ansible. Chaque script correspondant à une étape de l'installation.

Si vous êtes à l'école, il est impératif d'utiliser un vpn, comme protonvpn version gratuite.

1. Test connectivité

Avant de commencer l'installation, il est nécessaire de vérifier que toutes les machines sont joignables par Ansible

```
$ ansible-playbook test.yaml
```

2. Container runtime

Le runtime de conteneur, également appelé container engine, est un logiciel qui permet d'exécuter des conteneurs sur un système d'exploitation hôte. Il est responsable de la création, de l'exécution, de la gestion et de la suppression des conteneurs.

Il existe plusieurs types de runtimes de conteneurs, qui peuvent être classés en deux catégories principales :

- Les runtimes de conteneurs bas niveau : ces runtimes sont responsables de la création et de l'exécution des conteneurs, mais ils ne fournissent pas de fonctionnalités supplémentaires, telles que la gestion des ressources ou la surveillance. Les principaux runtimes de conteneurs bas niveau sont :
  - runc : runc est le runtime de conteneurs le plus populaire. Il est simple et efficace, et il est utilisé par de nombreux autres runtimes de conteneurs.
  - crun : crun est un runtime de conteneurs open source qui est développé par Red Hat. Il est basé sur runc, mais il apporte quelques améliorations en matière de performances et de sécurité.

- Les runtimes de conteneurs haut niveau : ces runtimes fournissent des fonctionnalités supplémentaires aux runtimes de conteneurs bas niveau, telles que la gestion des ressources, la surveillance ou la sécurité. Les principaux runtimes de conteneurs haut niveau sont :
  - Docker : Docker est le runtime de conteneurs le plus populaire. Il est facile à utiliser et à configurer, et il est bien intégré aux autres produits Docker.
  - Containerd : Containerd est un runtime de conteneurs open source qui est utilisé par de nombreux orchestrateurs de conteneurs, tels que Kubernetes.
  - CRI-O : CRI-O est un runtime de conteneurs open source qui est utilisé par Kubernetes.

Je vous ai fourni les scripts pour Containerd et CRI-O.
Pour la suite, j'utiliserai Containerd.

$ ansible-playbook containerd.yaml

Les commandes de bases de Containerd sont :

- pour lister les containers actifs (sur un invité):

$ sudo ctr -n k8s.io container ls

- pour lister les images présentes :

$ ctr -n k8s.io containers list

Pour plus d'information : https://platform9.com/docs/kubernetes/containerd-commands-and-info

3. Kube tools

Cette partie installe outils

- kubelet : composant qui s'exécute sur toutes les machines dans votre cluster et qui effectue des opérations telles que le démarrage des pods et des conteneurs.
- Kubeadm : commande d'amorçage du cluster
- kubectl : ligne de commande permettant de communiquer avec votre cluster.

$ ansible-playbook kube-tools.yaml

4. Master

Nous allons maintenant initialiser le master.

$ ansible-playbook master.yaml

Pour suivre la progression, il est nécessaire de se connecter sur le master sous root

$ ssh etudiant@10.54.56.100
$ sudo bash

puis de lister les pods présent dans le cluster

$ watch kubectl get pod --all-namespaces -o wide

Cette phase est plus ou moins longue …. car différents containers sont téléchargés du réseau.
À la fin, vous devriez avoir ceci sur votre console :

```
Every 2.0s: kubectl get pod --all-namespaces -o wide                                                masterv2.m222.org: Mon Jan 22 14:17:15 2024

NAMESPACE          NAME                                              READY   STATUS    RESTARTS   AGE     IP              NODE                 NOMINATED NODE   READINESS GATES
calico-apiserver   calico-apiserver-786474f5dc-7jkdr                 1/1     Running   0          2m22s   192.168.54.133  masterv2.m222.org    <none>           <none>
calico-apiserver   calico-apiserver-786474f5dc-tqhs4                 1/1     Running   0          2m21s   192.168.54.134  masterv2.m222.org    <none>           <none>
calico-system      calico-kube-controllers-fbfd8f68d-5djkk           1/1     Running   0          4m45s   192.168.54.130  masterv2.m222.org    <none>           <none>
calico-system      calico-node-7ntx7                                 1/1     Running   0          4m46s   10.54.56.100    masterv2.m222.org    <none>           <none>
calico-system      calico-typha-744496d596-ds9s5                     1/1     Running   0          4m46s   10.54.56.100    masterv2.m222.org    <none>           <none>
calico-system      csi-node-driver-xrsgr                             2/2     Running   0          3m20s   192.168.54.131  masterv2.m222.org    <none>           <none>
kube-system        coredns-5dd5756b68-4cgnt                          1/1     Running   0          5m38s   192.168.54.129  masterv2.m222.org    <none>           <none>
kube-system        coredns-5dd5756b68-j7qcj                          1/1     Running   0          5m38s   192.168.54.132  masterv2.m222.org    <none>           <none>
kube-system        etcd-masterv2.m222.org                            1/1     Running   0          5m52s   10.54.56.100    masterv2.m222.org    <none>           <none>
kube-system        kube-apiserver-masterv2.m222.org                  1/1     Running   0          5m55s   10.54.56.100    masterv2.m222.org    <none>           <none>
kube-system        kube-controller-manager-masterv2.m222.org         1/1     Running   0          5m54s   10.54.56.100    masterv2.m222.org    <none>           <none>
kube-system        kube-proxy-g7w5l                                  1/1     Running   0          5m39s   10.54.56.100    masterv2.m222.org    <none>           <none>
kube-system        kube-scheduler-masterv2.m222.org                  1/1     Running   0          5m53s   10.54.56.100    masterv2.m222.org    <none>           <none>
tigera-operator    tigera-operator-66996c666d-4msj8                  1/1     Running   0          5m7s    10.54.56.100    masterv2.m222.org    <none>           <none>
```

Nous allons configurer notre hôte pour interagir avec le cluster.
La première étape consiste à installer l'outil kubectl sur l'hôte (https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/). Sur le master, regarder la

version de kubectl utilisée

```
$ kubectl version
```

Sur l'hôte, installer la version de kubectl, avec X.X.X la version et YYY ayant comme valeur amd64 ou bien arm64 selon le processeur utilisé.
ATTENTION : ces commandes sont à exécuter dans le répertoire «maison» de l'utilisateur (d'où la commande cd vide).

```
$ cd
$ wget https://dl.k8s.io/release/vX.X.X/bin/linux/YYY/kubectl
$ sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
$ mkdir .kube
$ scp etudiant@10.54.56.100:/home/etudiant/.kube/config /home/etudiant/.kube
```

Vérifier la taille du binaire de kubectl (environ 50m) et que vous avez accès aux informations du cluster sur l'hôte

```
$ watch kubectl get pod --all-namespaces -o wide
```

5. Nodes

En production, le master n'héberge que les containers d'administration. Les containers des utilisateurs sont obligatoirement sur les workers.
L'initialisation d'un worker s'exécute avec la commande

```
$ ansible-playbook nodes.yaml
```

6. Cluster tools

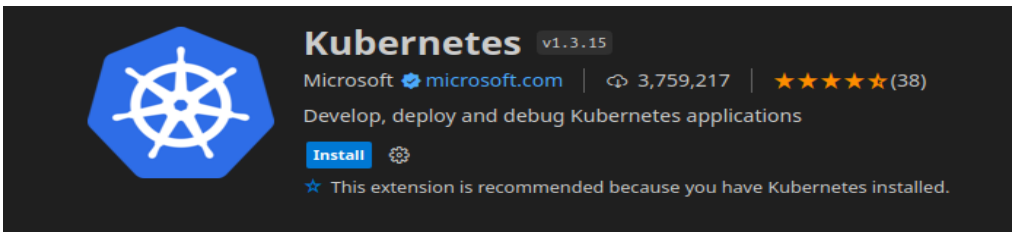
Cette partie installe Metallb

- MetalLb

Kubernetes ne propose pas d'implémentation d'équilibreurs de charge réseau (Services de type LoadBalancer) pour les clusters bare-metal. Si nous n'exécutons pas les containers sur une plateforme IaaS (GCP, AWS, Azure…), les LoadBalancers resteront indéfiniment dans l'état « en attente » une fois créés.
Pour corriger ce problème, MetalLB propose une implémentation d'équilibreur de charge réseau qui s'intègre aux équipements réseau standard, afin que les services externes sur les clusters bare-metal fonctionnent autant que possible.

7. Tableau de gestion

Pour faciliter la gestion du cluster, installer dans visual studio code l'extension kubernetes



et rajouter l'utilitaire helm sur l'hôte

```
$ sudo snap install helm --classic
```

Vous pouvez aussi utiliser un tableau de bord sur un navigateur (https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard/). Pour cela installer le pods correspondant

```
$ kubectl apply -f
https://raw.githubusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yaml
```

Créer un utilisateur comme indiqué dans https://github.com/kubernetes/dashboard/blob/master/docs/user/access-control/creating-sample-user.md

Ce qui nous permet de nous connecter au tableau de bord.

```
$ kubectl proxy
```

et dans un navigateur
http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/

8. Registre docker privé

Pour faciliter le développement de nos conteneurs, nous allons installer un registre privé docker.
Sur le host, commençons par créer le groupe docker, auquel étudiant appartient. Il est important que le groupe docker soit créé avant l'installation de docker.

```
$ sudo groupadd docker
$ sudo  usermod -aG docker etudiant
$ sudo snap install docker
```

Pour que l'appartenance aux groupes soit prise en compte, arrêtez les VM proprement via « virt-manager » et redémarrer le PC.

Puis

```
$ ansible-playbook docker-registrery.yml
```

A la fin du script, regarder que vous avez bien un pod « deployement-docker-registrery-XXX » sur le nœud « node1v2.m222.org »

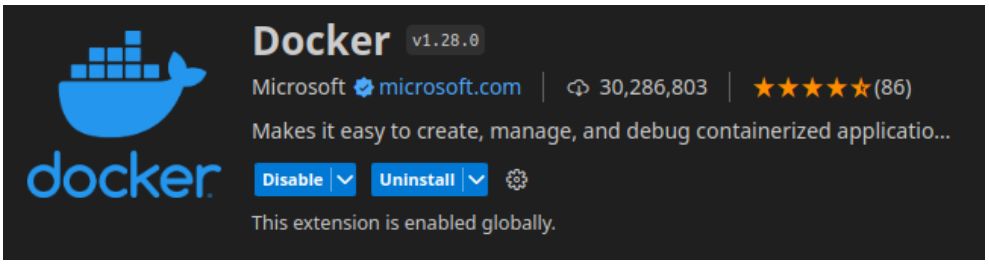
Le certificat n'est pas pris en compte dans k8s. Le plus simple est de relancer la partie de k8s qui s'en occupe :

```
$ kubectl -n kube-system delete pods kube-controller-manager-masterv2.m222.org
```

Rajouter le certificat masterRootCA.crt dans le répertoire /var/snap/docker/current/etc/docker/certs.d/10.54.56.39:5000 et vérifier que vous pouvez vous connecter au registre.

$ docker login -u etudiant -p etudiant 10.54.56.39:5000

Rajouter dans visual studio code l'extension



Pour vous permettre de gérer docker dans VSC. Il est alors nécessaire de mettre à jour les certificats pour que la connexion sur le registries en https://10.54.56.39:5000 fonctionne.
Les opérations sont les suivantes
copier masterRootCA.crt dans /usr/local/share/ca-certificates puis

$ update-ca-certificates

9. Sauvegarde de k8s

Il peut être nécessaire de repartir de cette configuration plus tard. Le plus simple est d'arrêter PROPREMENT les 4 VM, puis de prendre un snapshot de chaque VM sous kvm. De cette manière, vous pourrez restaurer cette configuration sans avoir a tous réinstaller.

Voilà! Vous avez maintenant une configuration k8s opérationnelle pour la suite du cours.