**Etapes projet 1**

**Création de 4 machines sous hyper-v**

Workstation 2g ram 1 CPU 15g disc

Master 4g ram 2 CPU 30g disc

Worker1 4g ram 2 CPU 30g disc

Worker2 4g ram 2 CPU 30g disc

**La première étape est de rendre les clés ip statique**

vi /etc/netplan/00-installer-config.yaml

avec la config suivante

# This is the network config written by 'subiquity'

network:

  renderer: networkd

  ethernets:

eth0:

   addresses:

     - 10.20.206.76/24

   nameservers:

     addresses: [4.2.2.2, 8.8.8.8]

   routes:

     - to: default

       via: 10.20.206.254

  version: 2

**Pour l'installation d'ansible on a commencé par installer python3**

apt-get install python3-pip

**Puis on a fait un git clone**

git clone <https://github.com/kubernetes-sigs/kubespray.git>

**Après on a lancé l'installation à partir du fichier requirements.txt existant dans le répertoire clone gît**

pip install -U -r requirement.txt

**On a créé par la suite un l'inventaire Hosts.yaml dans le répertoire de travail**

all:

  vars:

#pipelining : True

ansible\_become: True

   #ansible\_user: workstation

   #ansible\_password: workstation

#ansible\_sudo\_pass: workstation

ansible\_port: 22

   #ansible\_ssh\_private\_key\_file: /root/ansible/nodes.key

ansible\_ssh\_common\_args: -o StrictHostKeyChecking=no -o ControlMaster=auto -o ControlPersist=30m -o ConnectionAttempts=100 -o UserKnownHostsFile=/dev/null

ansible\_python\_interpreter: /usr/bin/python3

  hosts:

master:

   ansible\_host: 10.20.206.124

   ansible\_user: master

   ansible\_password: master

   ansible\_sudo\_pass: master

worker1:

   ansible\_host: 10.20.206.127

   ansible\_user: worker1

   ansible\_password: worker1

   ansible\_sudo\_pass: worker1

worker2:

   ansible\_host: 10.20.206.76

   ansible\_user: worker2

   ansible\_password: worker2

   ansible\_sudo\_pass: worker2

**Et le fichier user\_ssh.yaml qui est sensé créer l'utilisateur et la clé ssh**

---

- hosts: localhost

  become: True

  tasks:

- name : Création des clés SSH de gestion

   community.crypto.openssh\_keypair:

     path : /root/ansible/nodes.key

     size : 2048

- name : chmod keypair

   ansible.builtin.file:

     path: /root/ansible/nodes.key

     mode: 0600

     owner: workstation

     group: workstation

- hosts: all

  become: True

  vars\_prompt:

- name: "my\_password"

   prompt: "Enter password"

   private: yes

   encrypt: "sha512\_crypt"

   confirm: yes

   salt\_size: 7

  tasks:

- debug:

     msg: password is {{ my\_password }}

- name: Ensure group "ansible" exists

   ansible.builtin.group:

     name: ansible

     state: present

- name: Add the user 'ansible' with a specific uid and a primary group of 'ansible'

   ansible.builtin.user:

     name: ansible

     comment: Ansible User

     group : ansible

     createhome: yes

     shell: /bin/bash

     password : "{{ my\_password }}"

- name: Allow password authentication

   ansible.builtin.lineinfile:

     path: /etc/ssh/sshd\_config

     regexp: "^PasswordAuthentication"

     line: "PasswordAuthentication yes"

     state: present

   notify: restart sshd

# - name: Enable Root Login

#  lineinfile:

#    dest: /etc/ssh/sshd\_config

#    regexp: '^PermitRootLogin'

#    line: "PermitRootLogin yes"

#    state: present

#    backup: yes

#   notify: restart sshd

- name: Add the user 'ansible' in sudoers

   community.general.sudoers:

     name: ansible

     user: ansible

     commands: ALL

     nopassword: yes

     state: present

- name: Add public Key in autorized key

   ansible.posix.authorized\_key:

     user : ansible

     state: present

     key: "{{ lookup('file', '/root/ansible/nodes.key.pub') }}"

  handlers:

- name: restart sshd

   service:

     name: sshd

     state: restarted

**On ping vers les 3 machines en lançant la commande**

ansible all -i hosts.yml -m ping

**On lance le playbook comme suit :**

ansible-paybook -i hosts.yml user-ssh-key.yml –check on check et on lance sans check

**Une fois le lancement réussi on crée un nouveau fichier hosts2 en décochant les variable, mots de passe et la clé ssh qui vient d'être créé au lancement du playbook**

all:

  vars:

#pipelining : True

ansible\_become: True

ansible\_user: ansible

   #ansible\_password: workstation

   #ansible\_sudo\_pass: workstation

ansible\_port: 22

ansible\_ssh\_private\_key\_file: /root/ansible/nodes.key

ansible\_ssh\_common\_args: -o StrictHostKeyChecking=no -o ControlMaster=auto -o ControlPersist=30m -o ConnectionAttempts=100 -o UserKnownHostsFile=/dev/null

ansible\_python\_interpreter: /usr/bin/python3

  hosts:

master:

   ansible\_host: 10.20.206.124

worker1:

   ansible\_host: 10.20.206.127

worker2:

   ansible\_host: 10.20.206.76

**On relance un nouveau ping**

ansible all -i host2.yml -m ping

**Le playbook est lancé à ce moment employant le répertoire host2 avec la commande**

ansible-paybook -i host2.yml user-ssh-key.yml

Qui va nous générer un répertoire **~/kubespray/inventory/mycluster** contentant le fichier hosts qu'on doit modifier en rajoutant la partie des variables de notre répertoire hosts.yaml et le noms de nos différentes machines

all:

  vars:

#pipelining : True

ansible\_become: True

ansible\_user: ansible

   #ansible\_password: workstation

   #ansible\_sudo\_pass: workstation

ansible\_port: 22

ansible\_ssh\_private\_key\_file: /root/ansible/nodes.key

ansible\_ssh\_common\_args: -o StrictHostKeyChecking=no -o ControlMaster=auto -o ControlPersist=30m -o ConnectionAttempts=100 -o UserKnownHostsFile=/dev/null

ansible\_python\_interpreter: /usr/bin/python3

  hosts:

master:

   ansible\_host: 10.20.206.124

   ip: 10.20.206.124

   access\_ip: 10.20.206.124

worker1:

   ansible\_host: 10.20.206.127

   ip: 10.20.206.127

   access\_ip: 10.20.206.127

worker2:

   ansible\_host: 10.20.206.76

   ip: 10.20.206.76

   access\_ip: 10.20.206.76

  children:

kube\_control\_plane:

   hosts:

     master:

kube\_node:

   hosts:

     worker1:

     worker2:

etcd:

   hosts:

     master:

k8s\_cluster:

   children:

     kube\_control\_plane:

     kube\_node:

calico\_rr:

   hosts: {}

**On doit ensuite faire un ping à partir du fichier hosts en question pour être sûr que tout a été fait correctement**

ansible all -i inventory/mycluster/hosts.yaml -m ping

**Lancement des commandes suivantes pour installer les collections:**

ansible-galaxy collection install community.general

ansible-galaxy collection install community.crypto

**Suivre les instructions dans le lien :**

https://kubespray.io/#/

**la commande finale pour lancer le cluster à partir du repertoir kubespray**

ansible-playbook -i inventory/mycluster/hosts.yaml  --become --become-user=root cluster.yml

Plus d’info <https://github.com/kubernetes-sigs/kubespray#requirements>

**Installation kubectl dans la workstation**

Réinstallation des machines **master et worker1 et 2** pour octroyer la mémoire totale lors de l’installation

Edition des différentes adresses ip et les remettre en statique

Relancer l’Install du cluster kubernetes

Copie du fichier config de la machine master /etc/kubernetes/admin.conf à la machine workstation dans le fichier du configuration en le renommant « ~/.kube/config » ensuite le modifier et mettre l’ip du « Master » pour définir le master du cluster

**Ajout d’un nœuds’ au cluster en exécutant le playbook scale**

Première étape est de créer une nouvelle machine et rajouter l’adresse ip dans l’inventaire hosts.yaml

On effectue le ping pour s’assurer que la nœud est connecté

Et puis la commande pour rajouter le nœud au cluster

**root@workstation:~/kubespray# ansible-playbook -i inventory/mycluster/hosts.yaml scale.yaml**

On vérifie si le Node à bien été rajouté

kubectl get nodes \*ou bien

root@workstation:~/kubespray# ansible-inventory -i inventory/mycluster/hosts.yaml –graph

**Pour supprimer le Node**

root@workstation:~/kubespray# ansible-playbook -i inventory/mycluster/hosts.yaml remove-node.yaml -e ‘node=nodetest‘ -b

**On retire le Node par la suite de l’inventaire ou on le commente**

**Install de helm**

helm repo add grafana https://grafana.github.io/helm-charts

helm repo update

**Création d’un namespace**

kubectl create namespace monitoring

kubectk get namespace

**Lister tous les pods**

kubectl get pods -A

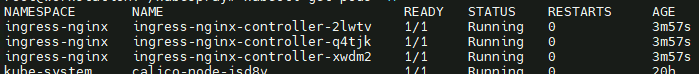
Activation du Dashboard et nginx-ingress plus les métriques de monitoring de cluster :

**Modification du fichier addons.yml dans le chemin pour les rajouts de services à notre cluster Dashboard :**

/root/kubespray/inventory/mycluster/group\_vars/k8s\_cluster/addons.yml

**Lancement de l’Install**

ansible-playbook -i /root/kubespray/inventory/mycluster/hosts.yaml cluster.yml --tags=apps



**Création des fichiers et lancement des dashboards et creation du token**

Dashbord\_ingress

Dasgbord-cluster-role-bending

dashboard-serviceaccount

Service-account-token

Kubectl apply -f ‘chaqu’un’ des fichiers’

**On recupere le token**

cmd de récupération du token : kubectl -n kube-system describe secret $(kubectl -n kube-system get secret | grep admin-user-token | awk '{print $1}')

**Générer la clé token**

**Se connecter sur les différent Dashboard**

ex : https://grafana.10.20.206.124.nip.io

**Vérifier les installations**

kubectl get svc -n kube-system

kubectl get svc -A

**Install de loki**

helm upgrade --install loki grafana/loki-stack --namespace monitoring

helm -n monitoring upgrade --install loki grafana/loki-distributed