Fase 4 - Activitat 9.2: Entorno de Multi-Màquina de Vagrant. Instal·lació de la implementació microk8s de Kubernetes.

0- Identificació del grup i activitat:

Curs: ASIX2
Projecte: GP2 DevOps i Cloud Computing
Fase: 4
Activitat: 9.2
Grup:
Membres:

1- Introducció i objectius de l'activitat 9.2

a) Lectura de les especificacions de l'activitat.
b) Alguns conceptes bàsics i inicials de kubernetes.
c) Definint de multiples hosts dins d'un mateix fitxer Vagrantfile. Creació d'un entorn multi-màquina.
d) Instal·lació de la implementació microk8s de kubernets sobre multiples màquines definides amb un fitxer Vagrantfile.

2- Conceptes de Kubernetes

a) Què és kubernetes?

És una eina open-source d'orquestració de contenidors. Això vol dir que és una eina que permet agrupar multiples ordinadors físics i virtual formant un cluster a on els contenidors es puguin desplegar fàcilment. El cluster semblarà un únic sistema sobre el qual es depleguen els contenidors.

b) Quin és el propòsit d'utilitzar kubernetes?

Que un entorno de producció a on l'aplicació ha d'estar disponible als usuaris finals es puguin garantir:

- L'escalabilitat
- Tolerancia a les fallades: L'aplicació continua funcionant sense interrupció encara que una part del sistema (per exemple, un o més contenidors) falli.
- Alta Disponiblitat: Habilitat de l'apliació de respondre sempre a qualsevol petició reduint al màxim el temps de no disponiblitat.
- Optima utilització de recursos (CPU, RAM, xarxa, espai de disc,....)
- Seamless Updates → Actualitzacions sense interrupció (les actualitzacions estan disponibles i seran efectives a la primera reinicialització)
- Rollback → Facilitat de tornar a una versió anterior si falla una actualització sense temps de no disponiblitat.
- Poder implementar política de seguretat d'accés a les aplicacions dins dels contenidors
- Balanceig de carrega
- Els contenidors poden comunicar-se amb un altre contenidors que s'executi en un altre host físic/virtual.
- Assignar a cada moment la quantitat de contenidors adequats a cada host del cluster en funció de la disponibilitat del seus recursos (cpu, ram, etc...).

c) Que és microk8s?

És una implementació de kubernetes. N'existeixen altres com per exemple minikube o k3s. Aquesta implementació té avantatges d'instal·lació, gestió del cluster que fa una opció interessant. A més a més està preparat per poder ser desplegat en sistemes i entorns en producció i té certificació CNCF que assegura que compleix totes les especificacions d'una eina d'orquestració de contenidors kubernetes.

d) Quins són els components bàsics de kubernetes?

Dins de **kubernetes** poden diferenciar els següents elements:

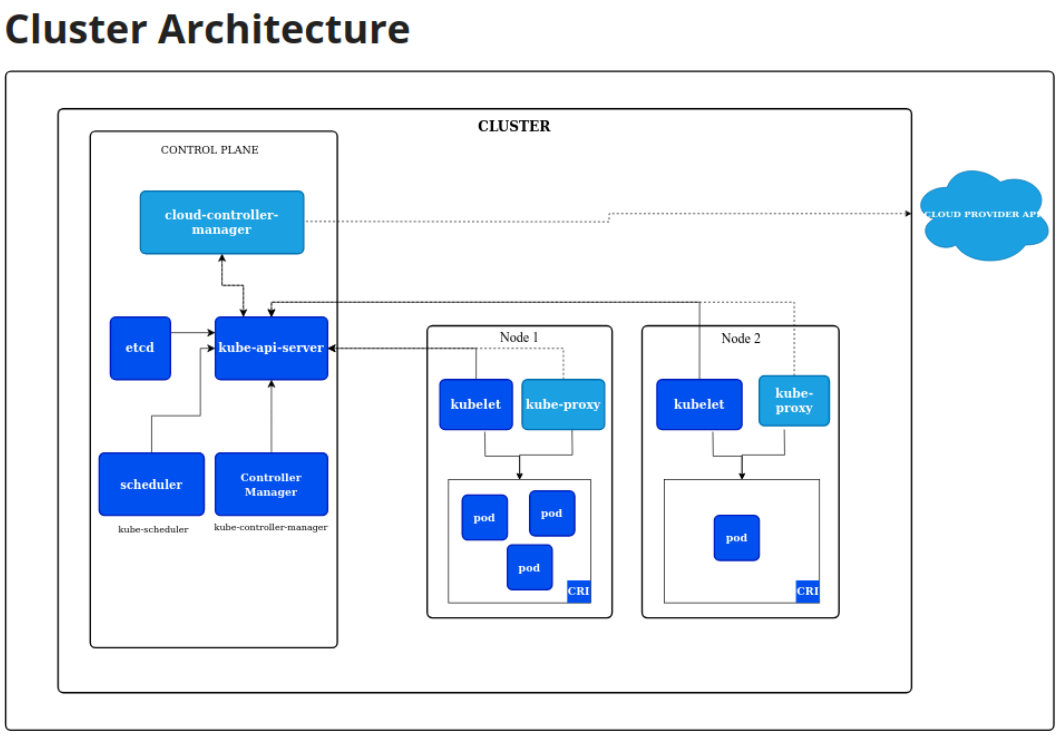
- El **cluster**: agrupació multiples ordinadors físics i virtuals que treballen coordinadament formant un únic sistema sobre el qual es depleguen i gestionen els contenidors.
- Els **nodes o workers**: cadascún dels ordinador físic o virtuals del cluster sobre els quals es despleguen els contenidors.
- Els **pods**:
  - És el conjunt de contenidors amb l'aplicació, la seva xarxa, volums compartits i una especificació de com han d'executar-se.
  - Dins d'un pod, a més dels contenidors amb l'apliació hi ha en marxa
  - Dins d'un **node** poden tenir en marxa 1 o més **pods**.
- El gestor del cluster anomenat **control plane**. Aquest component pot estar situat a un node o distribuït entre diversos nodes de manera que si falla un encara pugui donar servei amb la resta de nodes. El control plane gestiona els **nodes** i **pods** del cluster.

En **kubernetes** el element més petit que podem gestionar és un **pod**. Si volem escalar una aplicació haurem de crear nous pods. Si cal redistribuir la carrega, caldra veure de quin **node** s'eliminen uns pods i a quin altre **node** es posen en marxa altres **pods**.

Cada **node** té una sèrie de **components** que s'executen en 2n terme que li permeten controlar els seus **pods** i estar en contacte via **control plane** amb la resta del cluster. També té instruccions de la implementació de kubernetes que permet als usuaris interactuar via **control plane** amb la resta del cluster i amb els **pods** de la màquina.

El **control plane** té una sèrie de components que s'executaran en 2n terme (i potser de manera disitribuida) que li permeten controlar el cluster. L'usuari també té instrcuions per poder actuar sobre el **control plane**.

Un diagrama bàsic d'un cluster és aquest:



3- Definint de multiples hosts dins d'un mateix fitxer Vagrantfile. Creació d'un entorn multi-màquina.

a) D'acord amb allò que hem vist a l'apart anterior, veurem que per poder fer una activitat que tingui com a propòsit desplegar una aplicació utilitzant **kubernetes** ens calen uns quants nodes (com a mínim 3 o 4), que en el nostre cas seran màquines virtuals.

Si volem crear mútiples màquines virtuals aprofitant l'eina **vagrant**, ens caldrà:

- Crear mols directoris amb un fitxer Vagrantfile a cadascún d'ells.
- Configurar cada fitxer per separat i assegurar-nos que tinguin connexió entre ells via xarxa.
- Inciar o aturar cada màquina per separt.

Ja es veu que estem fent una feina repetitiva, que fa perdre molt el temps i a on es poden produir fàcilment molts errors per equivocacions.

Afortunadament l'eina **vagrant** disposa té dins del seu llenguatge de definició de màquines virtuals la possibilitat d'implementar bucles, estructures condicional, treballar amb variables i constants per poder crear amb un únic fitxer Vagrantfile un entorn multi-màquina.

Utilitzant les opcions multi-màquina:

- Només ens cal un únic fitxer Vagrantfile
- Podem crear, iniciar, aturar, modificar i esborrar múltiples màquines amb una única ordre.

b) Anem a crear un fixer **Vagrantfile multi-màquina**, amb el programari de contenidors Dockers i kubernetes. Crea dins d'una carpeta nova de nom **gp1f4a9** del directori de **projectes** de la **màquina física** el següent fitxer **Vagrantfile**:

```
IMATGE_BOX = "??????"
NODES = ??????
NOM_NODE = "???????"
MEMORIA = ??????
CPUS = ??????
TARGETA_XARXA = "??????"

Vagrant.configure("2") do |config|
 (1..NODES).each do |i|
  config.vm.define NOM_NODE+"#{i}" do |subconfig|
   subconfig.vm.box = IMATGE_BOX
   subconfig.vm.hostname = NOM_NODE+"#{i}"
   subconfig.vm.network "public_network", bridge: TARGETA_XARXA
   subconfig.vm.network "NAT", bridge: TARGETA_XARXA
   subconfig.vm.provider "virtualbox" do |v|
        v.memory = MEMORIA
        v.cpus = CPUS
        v.customize ['modifyvm', :id, '--clipboard', 'bidirectional', '--groups', '/KUBERNETES']
      end
  end
 end

 config.vm.provision "shell", inline: <<-SHELL
  sudo apt-get update -y
  sudo apt-get install -y net-tools
  sudo apt-get install -y whois
 SHELL
end
```

Ara canviarem els paràmetres de configuració:

- La imatge a utilitzar ha de ser **debian/bullseye64**.
- Crearem 4 nodes
- Els noms de sistema i de màquina virtual dels node seran node1, node2, etc..., de manera que el nom genèric de tots els nodes serà **node**.
- RAM de 1024MB
- Una CPU
- Seleccionarem la targeta de xarxa WiFi a partir del seu identificador dins de VirtualBoxx. Executa l'ordre:
  - Windows (Powershell): **VBoxManage.exe list bridgedifs | Select-String "Name"** i comprova el nom (Name) de les teves targetes de xarxa des del punt de vista de VirtualBox.
  - Linux: **VBoxManage list bridgedifs | grep ^Name** i comprova el nom (Name) de les teves targetes de xarxa des del punt de vista de VirtualBox.

Afegirem a la secció **provision** el programari de **contenidors** Dockers. Abans del SHELL final (segurament entre la línies 27 i 28) afegeix:

```
sudo apt-get -y install apt-transport-https ca-certificates curl gnupg2 software-properties-common
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/debian $(lsb_release -cs) stable"
sudo apt-get update -y
sudo sudo apt-get -y install docker-ce docker-ce-cli containerd.io docker-compose
sudo gpasswd -a vagrant docker
```

Finalment, afegirem a la secció **provision** el programari de la implementació **microk8s** de **kubernetes**. Una altra vegada, abans del SHELL final (segurament entre les línies 27 i 28) afegeix:

```
sudo apt-get install -y snapd
echo 'export PATH=/snap/bin:$PATH' >> /home/vagrant/.bashrc
source /home/vagrant/.bashrc
sudo snap install microk8s --classic
sudo gpasswd -a vagrant microk8s
sudo  chown -f -R vagrant /home/vagrant/.kube
echo "alias kubectl='microk8s kubectl'" >> /home/vagrant/.bashrc
source /home/vagrant/.bashrc
exit
```

A continuació:

- Executa **vagrant up** i comprova es creen i s'aprovisiona tot el programari i s'executen les instruccions de configuració.
- Executa **vagrant status i** comprova s'han creat les màquines **node1** a **node4**.
- Accedeix a les màquines virtuals. Executa:
  - **vagrant ssh node1** → per accedir a la màquina **node1**
  - **vagrant ssh node2** → per accedir a la màquina **node2**
  - **vagrant ssh node3** → per accedir a la màquina **node3**
  - **vagrant ssh node4** → per accedir a la màquina **node4**
- Comprova dins d'una màquina (per exemple **node1**) que el programari de **micro8ks** està instal·lat, configurat i tenim accés a les eines per poder utilitzar-lo. Executa aquestes ordres:
  - **ip a show dev eth1** → per comprovar l'adreça IP de l'equip.
  - **id vagrant** → per comprovar que l'usuari **vagrant** és membre de **micro8ks**.
  - **microk8s version** → versió **MicroK8s v1.28.3 revision 6089**
  - **kubectl get nodes** → Comprova l'estat del teu node (hauria d'estar ready però sense role).

d) Surt de la màquina virtual a on estiguis treballant i executa **vagrant halt -f** per aturar totes les màquines, des de la màquina física.

Lliurament de l'activitat

a) Mostra el contingut de **Vagrantfile**
b) Posa en marxa l'entorn **Multi-màquina**.
c) Accedeix al **node1** i comprova que es pot treallar amb **microk8s**.
d) Surt de **node1** i atura totes les màquines virtuals.
e) Data límit per obtenir el 100% de la nota: dimarts 21-11-23 a les 19.10.