Fase 4 - Activitat 9.1: Orquestració de contenidors amb Kubernetes. Conceptes bàsics. Creació de nodes i clusters

0- Identificació del grup i activitat:

Curs: ASIX2
Projecte: PJ9 DevOps i Cloud Computing
Fase: 4
Activitat: 9.1
Grup/Individual: Grupal
Membres/Alumne:

1- Introducció i objectius de l'activitat 9.1

a) Conceptes bàsics sobre Kubernetes.
b) Creació dels nodes del cluster amb el programari de contenidors i Kubernetes
c) Creació del cluster de Kubernetes
d) Desplegament d'una aplicació sobre el cluster
e) Gestió del cluster i l'aplicació.

2- Conceptes de Kubernetes

a) Què és Kubernetes?

Sobre **Kubernets** podem dir que:

- És una eina open-source d'orquestració de contenidors.
- Amb aquesta eina podem agrupar multiples hosts (ordinadors físics i virtuals) formant un cluster. El cluster semblarà un únic sistema sobre el qual serà fàcil desplegar multiples contenidors.
- El contenidors es distribuiran al llarg dels host sque formen part del cluster.
- El contenidors estaran intercomunicats dins del cluster tot i que formin part d'un host diferent.

b) Quin és el propòsit d'utilitzar Kubernetes?

El propòsit de **Kubernetes** és que, dins d'un entorn de producció a on una aplicació hagi d'estar disponible als usuaris finals, es pugui garantir:

- Poder desplegar i gestionar fàcilment l'aplicació posant en marxa desenes, centenars o milers de contenidors distribuits entre multiples hosts i intercomunicats.
- 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 dels hosts del cluster (CPU, RAM, xarxa, espai de disc,....)
- Seamless Updates i Rolling updates → Actualitzacions sense interrupció i temps d'espera zero.
- Rollback → Facilitat de tornar a una versió anterior si falla una actualització sense temps de no disponiblitat.
- Poder implementar polítiques de seguretat d'accés a les aplicacions dins dels contenidors
- Balanceig de carrega

c) Que és microk8s?

Sobre **microk8s** podem dir que:

- És una implementació de kubernetes. N'existeixen altres com per exemple **minikube** o **k3s**.
- Aquesta implementació és una opció interessant perquè proporciona documentació, eines d'instal·lació i eines de gestió de clusters que fan de **microk8s** una opció molt interessant tan per aprendre com per utilitzar **Kubernets** en entorns de producció.
- Està preparada per poder ser desplegat en sistemes i entorns en producció i té certificació CNCF i per tant, 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**: cadascún dels ordinador físics o virtuals del cluster sobre els quals es despleguen els contenidors.
- Els **pods**:
  - És la unitat mínima d'instal·lació d'una aplicació. Una aplicació necessita com a mínim un Pod per funcionar. Normalment, quan posem en marxa una aplicació ho fem sobre multiples Pods.
  - Dins d'un pod tenim els contenidors de l'aplicació, la seva xarxa, els volums compartits i el **.yml** de com han de desplagar-se els contenidors (ports, nom de contenidors, imatges, etc...).
  - Dins d'un **node** poden tenir en marxa 1 o més **pods**.
  - Són efimers. Això vol dir que si per qualsevol motiu un Pod deixa de funcionar, automàticament es posa en marxa un altre per substituir-lo.
- 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.

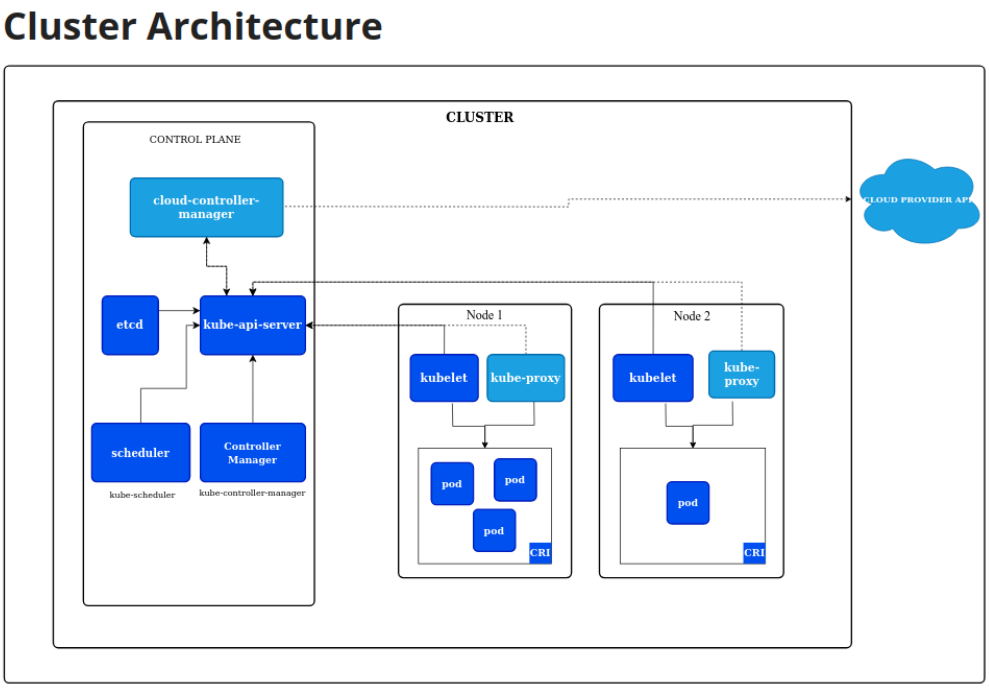
En **kubernetes** el element més petit que podem gestionar és un **pod**. Per exemple:

- Si volem escalar una aplicació haurem de crear nous pods o esborrar-ne.
- 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**.

En **kubernetes**:

- Cada **node** controla té les eines necessàries per controlar els seus **pods** i estar en contacte via **control plane** amb la resta del cluster.
- El **control plane** té les eines necessàries per controlar el cluster, els nodes i les comunicacions amb l'exterior del cluster. Generalment es troba dins d'un node però pot distribuir-se entre diferentes nodes del cluster.
- un **worker** és qualsevol node sobre el qual s'executa una aplicació i que no forma part del **control plane.**

Un diagrama bàsic d'un **cluster** és aquest:



3- Creació d'un cluster de Kubernetes

3.1- Creació dels nodes necessaris per establir un cluster de Kubernetes

Per poder utilitzar **Kubernetes** ens cal crear un **cluster** de **nodes**. Com que els **nodes** poden ser màquines físiques o virtuals crearem 3 màquines amb **Vagrant** de la mateixa manera que ho vam fer a l'activitat **pj9f4a7.5** en la qual varem aprendre a crear múltiples màquines virtuals utilitzant ún únic fitxer **Vagrantfile**.

a) Crea una carpeta de nom **a9** dins de la carpeta **f4** que es troba a **pj9**. A continuació, dins de la carpeta **a9** crea una carpeta de nom **pj9f4a9.1**. Dins de **pj9f4a9.1** crea un fitxer **Vagrantfile** amb aquest contingut:

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

# -*- mode: ruby -*-
# vi: set ft=ruby :

#################################
#### Definició de variables ####
#################################

IMATGE_BOX_NODES = "???????"
PROVIDER = "???????"
NUM_NODES = ???????
NOM_BASE_NODES="???????"
NOM_DOMINI_NODES="???????"
MEMORIA_RAM_NODES = ???????
NUM_CPUS_NODES = ???????
TARGETA_XARXA="???????"

################################################
#### Definició de les màquines. Provision ####
################################################

Vagrant.configure("2") do |config|

 ###############################
 #### Creació dels Nodes ####
 ###############################
 (1..NUM_NODES).each do |i|
```

```
    config.vm.define NOM_BASE_NODES+"#{i}" do |node|
      node.vm.box = IMATGE_BOX_NODES
      node.vm.hostname = NOM_BASE_NODES+"#{i}"+"."+NOM_DOMINI_NODES
      node.vm.network "public_network", bridge: TARGETA_XARXA
      node.vm.provider PROVIDER do |prov|
            prov.name = NOM_BASE_NODES+"#{i}"
            prov.memory = MEMORIA_RAM_NODES
            prov.cpus = NUM_CPUS_NODES
            prov.customize ['modifyvm', :id, '--clipboard', 'bidirectional', '--groups', '/KURBERNETES', '--mac-address1', "0800278DC04"+"#{i}"]
          end
    end
 end

 ##############################################################################################################
 #### Programari a instal·lar i instruccions a executar a les màquines virtuals durant la seva creació ####
 ##############################################################################################################
 config.vm.provision "shell", inline: <<-SHELL
   #### Instal·lació d'algunes eines de sistema
   sudo apt-get update -y
   sudo apt-get install -y net-tools whois aptitude git zip unzip curl
   #### Instal·lació de Docker
   #### Instal·lació de microk8s
   SHELL
end
```

b) Ara canviarem els paràmetres de configuració:

- Utilitzarem el box **debian/bookworm64**.
- Treballarem amb **virtualbox** com a eina de virtualització (provider).
- El nom base del sistema i l'identificador dins de VirtualBox dels nodes serà **node**.
- El nom de domini dels nodes serà **fjeclot.net**
- La RAM dels nodes serà **2048MB**
- Les CPUs assignades als nodes seran **2**
- Crearem **2** nodes
- Seleccionarem la targeta de xarxa WiFi a partir del seu identificador dins de VirtualBox. Executa l'ordre:
  - Windows (Powershell) → **VBoxManage.exe list bridgedifs | Select-String "Name"** → comprova el nom (Name) de les teves targetes de xarxa des del punt de vista de VirtualBox.
  - Linux → **VBoxManage list bridgedifs | grep ^Name** → comprova el nom (Name) de les teves targetes de xarxa des del punt de vista de VirtualBox.

c) Afegeix a la secció **provision** el programari **Docker**. Després de la línia *# Instal·lació de Docker* dins de **Vagrantfile** escriu:

```
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
```

d) Afegeix a la secció **provision** el programari **microk8s**. Després de la línia *# Instal·lació de microk8s* dins de **Vagrantfile** escriu:

```
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
```

e) Executa:
- **vagrant box update**
- **vagrant up**

i posa en marxa les màquines virtuals.

f) Executa **vagrant status** i comprova s'han creat les màquines **node1** i **node2**.

g) Comprova que les màquines virtual són vísibles a **VirtualBox** dins d'un grup de nom **KUBERNETES**.

h) Accedeix a les màquina virtual **node1**. Executa: **vagrant ssh node1**

i) Comprova dins de **node1** que:
- L'usuari **vagrant** es membre del grups **docker** i **micro8ks**. Executa: **id vagrant**
- El programari **micro8ks** i **docker** està disponible. Executa:
  - **microk8s version** (hauria de sortir `MicroK8s v1.31.3 revision 7449` o posterior)
  - **doker -v** (hauria de sortir `Docker version 27.4.1, build b9d17ea` o posterior)
- Els noms de sistema del **node1**. Executa: **hostname --fqdn**. Els nom han de ser **node1.fjeclot.net**.
- Comprova l'adreça IP i MAC de la interfície **eth1** executant: **ip addr show eht1**
- Comprova l'adreça MAC de la interfície **eth0** executant: **ip addr show eht0**

j) Surt de **node1** i accedeix a **node2**. Fes les mateixes comprovacions que has fet a **node1**. Assegura't que tot funciona correctament i que les adreces **MAC** d'**eth0** i **eth1** són diferents a les del **node1**. Assegura't també que l'adreça IP d'**eth1** és diferent.

i) Dins de **node1** assegura't d'habilitar els add-on **dns**. Executa: **microk8s enable dns**

3.2- Creació d'un cluster de Kubernetes a partir dels 2 nodes creats a l'apartat anterior

a) Dins de **node1** executarem la següent ordre per habilitat l'add-on **dns**:

microk8s enable dns

b) Després executarem:

microk8s add-node

i el resultat serà una cosa similar a això:

```
vagrant@node1:~$ microk8s add-node
From the node you wish to join to this cluster, run the following:
microk8s join 10.0.2.15:25000/cfd9bab7ce3e09980cab837a86259c39/fd1d8374c32b

Use the '--worker' flag to join a node as a worker not running the control plane, eg:
microk8s join 10.0.2.15:25000/cfd9bab7ce3e09980cab837a86259c39/fd1d8374c32b --worker

If the node you are adding is not reachable through the default interface you can use one of the following:
microk8s join 10.0.2.15:25000/cfd9bab7ce3e09980cab837a86259c39/fd1d8374c32b
microk8s join 192.168.1.36:25000/cfd9bab7ce3e09980cab837a86259c39/fd1d8374c32b
microk8s join 172.17.0.1:25000/cfd9bab7ce3e09980cab837a86259c39/fd1d8374c32b
vagrant@node1:~$
```

Ordre a executar en **node2** per unir-lo al cluster. L'adreça IP segurament serà diferent.

Ara **node1** ja actua com a control plane d'un cluster i amb les ordres indicades podem unir altres nodes al cluster.

c) Afegirem **node2** al **cluster** executant dins de **node2** l'ordre indicada a l'apartat **a)** , i el resultat serà similar a això:

```
vagrant@node2:~$ microk8s join 192.168.1.36:25000/cfd9bab7ce3e09980cab837a86259c39/fd1d8374c32b
Contacting cluster at 192.168.1.36
Waiting for this node to finish joining the cluster. .. .. .. ..
Successfully joined the cluster.
vagrant@node2:~$
```

d) Dins de **node1** executa:

microk8s kubectl get nodes

i el resultat hauria de ser:

```
vagrant@node1:~$ microk8s kubectl get nodes
NAME    STATUS   ROLES    AGE     VERSION
node1   Ready    <none>   15m     v1.31.3
node2   Ready    <none>   3m55s   v1.31.3
vagrant@node1:~$
```

Si a la columna **STATUS** surt **Ready** tant a **node1** com a **node2** és que ja tenim un cluster amb 2 nodes.

4- Lliurament de l'activitat

- Comprovació que es poden posar 2 màquines virtuals a partir del fitxer **Vagrantfile** creat.
- Comprovació que el programari microk8s està disponible a les 2 màquines virtuals
- Comprovació des de node1 que s'ha creat el cluster amb les 2 màquines virtuals.
- Data límit per obtenir el 100% de la nota: **dijous 9-1-25** a les **17.45**. (posteriorment és el 50%).