Fase 4 - Activitat 9.2: Orquestració de contenidors amb Kubernetes. Desplegament d'aplicacions utilitzant Kubernetes.

0- Identificació del grup i activitat:

Curs: ASIX2
Projecte: PJ9 DevOps i Cloud Computing
Fase: 4
Activitat: 9.2
Grup/Individual: Individual
Membres/Alumne:

1- Introducció i objectius de l'activitat 9.2

a) Nous conceptes bàsics sobre Kubernetes.
b) Desplegament d'una aplicació sobre un cluster
c) Gestió de l'aplicació.

2- Nous conceptes bàsics de Kubernetes

- Què és un **Deployment** (desplegament)?:
  - Per mitjà d'un **Deployment** posem en marxa els **Pods** d'una aplicació sobre un cluster de **Kubernetes**. En altres paraules, instal·lem i posem en marxa l'aplicacío utilitzant **Kubernetes**.
  - Els **Deployments** no serveixen únicament per posar en marxa els Pods d'una aplicació. També ens permeten indicar les característiques dels **Pods** a posar en marxa. Això vol dir entre d'altres coses, indicar:
    - Imatges de contenidors utilitzades per crear els contenidors dels Pods.
    - El nom dels contenidors dels Pods.
    - Els volums i xarxes dins dels Pods.
    - Ports exposats per fer l'apliació disponible a l'usuari
    - Recursos de memòria RAM i CPU necessaris.
  - Els **Deployments** també s'utilitzen per gestionar els Pods. Això vol dir:
    - Proporcionar escalabilitat.
    - Permetre fer actualitzacions en temps real amb temps d'espera mínim o zero.
    - Permetre tornar a version posteriors (rollback) en temps real amb temps d'espera mínim o zero.
    - Proporcionar alta disponibilitat.
    - Monitoritzar l'estat dels Pods
    - Desplegar nous Pods i substituir-ne els antics si el seu estat no és el desitjat.
  - Els **Deployments** permeten automatizar la gestió (escalabilitat, actualitzacions, rollbacks, monitorització, substitució) dels Pods i això s'eviten molts errors humans, s'allibera temps dels administradors i aquestes tasques es fan més ràpidament.

- Què és un **Service** (Servei)?: Per mitjà d'un **Service** exposarem tots els Pods de l'apliació com si fosin un únic servei de xarxa i també podrem fer balanceig de carrega. D'aquesta manera, farem disponible i accessible l'aplicació com si fos una única aplicació sobre un únic servidor.

- Què és un **Ingress** (Servei)?: Un reverse-proxy de Kubernetes que serà necessari per poder accedir a un Service des de fora del cluster. Ingress redirecciona les peticions HTTP (o HTTPS) des de fora del cluster al Service que exposa l'aplicació a l'exterior del cluster.

- Què és un **manifest**?:
  - És un fitxer en format **YAML** (també por ser JSON) amb l'inventari (o sigui la llista) de tots els Pods, Services i Deployments que es volem desplegar-se sobre un cluster i les seves configuracions.
  - Amb un fitxer de manifest i una ordre de Kubernetes simple poden crear i gestionar fàcilment els Pods, Deployments, Services i un Ingress necessaris per fer accessibles aplicacions als usuaris.
  - Com sempre, aplicar el manifest és fàcil, allò que és complicat és la creació un fitxer de manifest.
  - El fitxers de manifest tenen normalment l'extensió **.yml** o **.yaml**.

3- Desplegament d'una aplicació utilitzant Kubernetes

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

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 **pj9f4a9.2**. Dins de **pj9f4a9.2** 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', '/KURBERNETES2', '--mac-address1', "0800278DC05"+"#{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 canvia 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 **node92**.
- El nom de domini dels nodes serà **clotfje.net**
- La RAM dels nodes serà **3072MB**
- Les CPUs assignades als nodes seran **3**
- Crearem **1** node
- Seleccionarem la targeta de xarxa WiFi a partir del seu identificador dins 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 la màquina virtual.

f) Executa **vagrant status** i comprova s'han creat la màquina **node921**.

g) Comprova que les màquines virtual són vísibles a **VirtualBox** dins d'un grup de nom **KUBERNETES2**.

h) Accedeix a les màquina virtual **node921**. Executa: **vagrant ssh node921**

i) Comprova dins de **node921** 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 nom de sistema del **node921**. Executa: **hostname --fqdn**. El nom serà **node921.fjeclot.net**.
- Comprova l'adreça IP de la interfície **eth1** executant: **ip addr show eht1**

3.2- Comprovació que el cluster de Kubernetes està funcionant

a) Dins de **node921** comprova que **microk8s** està en marxa. Executa: **microk8s status** i comprova que a la primera linia es mostra el missatge: **microk8s is running**.

b) Executa la següent ordre per habilitar l'add-on **dns**: **microk8s enable dns**

c) Dins de **node921** executa: **microk8s kubectl get nodes** i comprova que el resultat és aquest:

```
vagrant@node921:~$ microk8s kubectl get nodes
NAME      STATUS   ROLES    AGE   VERSION
node921   Ready    <none>   71s   v1.31.3
```

Si a la columna **STATUS** surt **Ready v**ol dir que el cluster està en funcionament.

3.3- Creació d'un Deployment dins d'un cluster de Kubernetes

3.3.1- Creació d'una imatge de Docker per fer el Deployment

a) Dins de **node921** crea una carpeta de nom **projectes**. Dins de la carpeta **projectes** crea una carpeta de nom **ppv**. Dins de la carpeta **ppv** crea una carpeta de nom **codi**.

b) Dins de la carpeta **ppv** descarrega un fitxer **Dockerfile** des de l'adreça URL:

https://raw.githubusercontent.com/asix2pj9/ppv/refs/heads/main/Dockerfile

c) Dins de la carpeta **codi** descarrega els arxius **index.html** i **ppv.php** que es troben a les adreces URL:

https://raw.githubusercontent.com/asix2pj9/ppv/refs/heads/main/codi/index.html

https://raw.githubusercontent.com/asix2pj9/ppv/refs/heads/main/codi/ppv.php

d) Crea una imatge de contenidor de nom **ippv** i versió **1.0**. Dins del directori **ppv** executa:

```
docker build -t ippv:1.0 .
```

3.3.2- Puja la imatge a Docker Hub

a) Crea un repository per les imatges de l'aplicació **ppv** amb les següents característiques:
- Namespace → El valor per defecte és correcte.
- Repository name: **ippv**
- Short description: **App ppv (pay per view)**
- Visibility: **Public**

b) Dins del directori **ppv** de la màquina virtual **node921** executa les següent ordres per associar les imatges locals **ippv** versió **1.0** al seu corresponent dins del repository **ippv** del teu compte de **Docker Hub**:

```
docker tag ippv:1.0 <nom_usuari_dockerhub>/ippv:1.0
```

c) Fes un **login** al teu compte de Docker Hub:

```
docker login -u <nom_usuari_dockerhub>
```

i si et pots validar dins de Docker Hub sortirà el missatge: `Login Suceeded`

d) Ara puja la imatge creada al dipòsit:

```
docker push <nom_usuari_dockerhub>/ippv:1.0
```

e) Refresqueu el vostre dipòsit de **Docker Hub** i comproveu que s'han pujat la imatge **ippv:1.0** dins del dipòsit **ippv** del teu compte.

f) Si s'has pogut pujar la imatge, ara ja pots fer un **logout** del teu compte de Docker Hub:

```
docker logout
```

i també surt del teu compte de **Docker Hub** des del navegador.

3.3.3- Creant un manifest i fent un Deployment a partir del manifest

a) Dins de la carpeta **ppv** de **node921** crea un fitxer de nom **ppv-deployment.yaml**. Dins d'aquest fitxer escriu el següent codi de desplegament de l'aplicació:

```
apiVersion: apps/v1
kind: Deployment # Especificica el recurs a desplegar: Deployment, Service , Pod, etc.
metadata:
  name: depl-ppv # Nom identificador del Deployment i nom base dels Pods en el sistema
spec: #spec del Deployment
  replicas: 3 # Quantitat de Pods a crear quan es fa el desplegament
  selector:
    matchLabels:
      app: pods-ppv # Etiqueta per selecciona els Pods que seran gestionats per aquest Deployment .
  template: # Configuració dels Pods
    metadata:
      labels:
        app: pods-ppv # Etiquetes assignades als Pods que els permet ser seleccionats per formar part del Deployment.
    spec: #spec dels Pods
      containers:
      - name: cppv # Nom base dels contenidors dins del Pod
        image: <nom_usuari_dockerhub>/ippv:1.0 # Imatge per crear contenidors
        ports:
        - containerPort: 80 # Port pel qual s'exposa l'aplicació.
```

NOTES:

- *Canvia <nom_usuari_dockerhub> pel nom del compte de **Dockerhub** utilitzat a l'apartat **3.3.2**.*
- *Això és un fitxer YAML i les sagnies són importants. Si copies i enganxes d'un PDF hauràs de comprovar els espais.*
- *Has d'assegurar-te que els comentaris queden al mateix lloc a on es veuen si copies i enganxes d'un PDF.*

Aquest manifest:

- Crearà un **Desployment** de nom **depl-ppv** dins del **namespace** identificat com **ns-ppv**.
- Crearà **3 Pods** de nom base **depl-ppv-**
- Cadascun dels **Pods** tindrà un contenidor de nom **cPpv** creat a partir de la imatge **ippv.1.0**.
- El port **80** de cada contenidor serà exposat a l'exterior per poder fer accessible l'aplicació.

b) Ara farem el desplegament. Executarem dins de la carpeta **ppv** de **node921** l'ordre:

```
microk8s kubectl apply -f ppv-deployment.yaml
```

i el sistema hauria de mostrar el missatge: **`deployment.apps/depl-ppv created`**

c) Comprova l'estat del Ddeployment. Executa:

```
microk8s kubectl get deployment
```

i comprova que el resultat és:

```
vagrant@node921:~/projectes/ppv$ microk8s kubectl get deployment
NAME       READY   UP-TO-DATE   AVAILABLE   AGE
depl-ppv   3/3     3            3           35s
```

NOTA: Pot trigar des d'un uns pocs segons a uns minuts en mostrar el resultat perquè tots els Pods i els seus components necessiten un temps per ser creats i també perquè pot ser necessari descarregar imatges des de Docker Hub.

d) Finalment, comprova que l'estat dels Pods. Executa:

microk8s kubectl get pods

i comprova que el resultat és semblant a aquest:

```
vagrant@node921:~/projectes/ppv$ microk8s kubectl get pods
NAME                        READY   STATUS    RESTARTS   AGE
depl-ppv-59bcfdc985-h8mhx   1/1     Running   0          10m
depl-ppv-59bcfdc985-phhdk   1/1     Running   0          10m
depl-ppv-59bcfdc985-xpxwx   1/1     Running   0          10m
```

e) Si els resultats són correctes, llavors ja has fet un **Deployment**, o sigui has instal·lat i posat en marxa una aplicació utilitzant **Kubernetes**. Però encara l'aplicació no està disponible per poder ser utilitzada pels usuaris. Per fer disponible l'aplicació als usuaris, primer de tot cal crear un **Service** associat al **Deployment**.

3.4 Creació d'un Service a partir del Deployment de l'aplicacó

a) La creació d'un Service de Kubernetes és un pas necessari fer disponible una aplicació desplegaga amb Kubernetes i que permet exposar tots els Pods de l'apliació com si fosin un únic servei de xarxa i fer balanceig de carrega entre Pods

b) Dins de la màquina **node921** entra dins de la carpeta **ppv**. Crea un fitxer per crear un servei associat al desplegament fet a l'apartat anterior de nom **ppv-service.yaml**. Dins d'aquest fitxer escriu el següent codi:

```
apiVersion: v1
kind: Service  #Especificica el recurs a desplegar: Deployment, Service , Pod, etc.
metadata:
   name: serv-ppv # Nom del service que s'ha de crear
spec:
   type: NodePort
   selector:
      app: pods-ppv # Etiquetes identificadores dels Pods dins de ppv-deployment.yaml
   ports:
   - port: 4000 # Port del Service  que es redirigirà cap el port dels Pods.
     targetPort: 80 # Port pel qual els Pods exposen l'aplicació.
     nodePort: 30000 # Port creat dins de la màquina que es redirigirà cap el port del Service.
```

NOTES:

- *Això és un fitxer YAML i les sagnies són importants. Si copies i enganxes d'un PDF hauràs de comprovar els espais.*
- *Has d'assegurar-te que els comentaris queden al mateix lloc a on es veuen si copies i enganxes d'un PDF.*

Amb aquesta configuració crearem un **Service** amb aquestes característiques:

- Nom del servei: **serv-ppv**
- Pods utilitzats pel servei → Els pods **pods-ppv** definits dins de **ppv-deployment.yaml**.
- Redirecció de ports:

Port del node		Port del Service		Port dels Pods
30000/tcp	→	4000/tcp	→	80/tcp

Una connexió des de l'exterior al port **30000/tcp** del node **node921** serà redirigit al port **4000/tcp** del Service **ppv-serv** que el redigirirà al port **80/tcp** del Pod seleccionat a cada moment utilitzant el balanceig de carrega que ens proporciona el Service.

b) Ara crearem el Service. Executa dins de la carpeta **ppv** de **node921** l'ordre:

microk8s kubectl create -f ppv-service.yaml

i el sistema hauria de mostrar el missatge: **`service/serv-ppv created`**

c) Comprova que l'estat del Service. Executa:

microk8s kubectl get services

i comprova que el resultat és semblant a aquest:

```
vagrant@node921:~/projectes/ppv$ kubectl  get services
NAME         TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
kubernetes   ClusterIP   10.152.183.1     <none>        443/TCP          72m
serv-ppv     NodePort    10.152.183.199   <none>        4000:30000/TCP   2m8s
```

e) Si els resultats són correctes, llavors ja has creat un **Setvice**, pel **Deployment** creat a l'apartat anterior. Però encara l'aplicació no està disponible per poder ser utilitzada pels usuaris des de fora del cluster de Kubernetes. Per fer disponible l'aplicació als usuaris fora del cluster, cal posar en marxa un servei de **Reverse-Proxy** dins dels nodes del cluster.

3.3.5- Utilització del reverse-proxy Ingress per accedir a l'aplicació des de fora del cluster

a) Per accedir a l'aplicació des de fora del cluster ens cal un servidor **reverse-proxy** que **redireccioni** les peticions HTTP (o HTTPS) des de fora del cluster al servei **serv-ppv** dins del cluster. **Kubernetes** en proporciona un servei de **reverse-proxy** anomenat Ingress que haurem d'habilitar i configurar per fer les redireccions.

b) Habilita **Ingress**. Executa dins de **ppv** de **node921** l'ordre:

microk8s enable ingress

i el sistema mostrarà un missatge com aquest:

```
Infer repository core for addon ingress
Enabling Ingress
....
....
Ingress is enabled
```

c) Configura **Ingress**. Dins de la carpeta **ppv** de **node921** crea un fitxer de nom **ppv-ingress.yaml**. Dins d'aquest fitxer escriu el següent codi de configuració:

```
apiVersion: networking.k8s.io/v1
kind: Ingress #Especificica el recurs a desplegar: Deployment, Service , Pod, etc.
metadata:
   name: proxy-ppv # Nom del service del recurs tipus ingress a crear
   annotations:
      spec.ingressClassName: "nginx"
spec:
   rules:
   - host: ppv.clotfje.net #Nom i domini amb el qual es pot accedir des de l'exterior
     http:
       paths:
       - path: /
         pathType: Prefix
         backend:
            service:
               name: serv-ppv # Nom del service del qual ha de fer de  reverse-proxy
               port:
                  number: 4000 # El mateix número que - port dins ppv-service.yaml
```

NOTES:

- *Això és un fitxer YAML i les sagnies són importants. Si copies i enganxes d'un PDF hauràs de comprovar els espais.*
- *Has d'assegurar-te que els comentaris queden al mateix lloc a on es veuen si copies i enganxes d'un PDF.*

Amb aquesta configuració, crearem un **reverse-proxy** que permet que qualsevol accés a **http://ppv.fjeclot.net:30000** és redireccioni al port **4000/tcp** del Service **serv-ppv** que autòmaticament es redirigirà al port **80/tcp** del qualsevol dels **Pods** de l'aplicació.

d) A continuació exexcuta dins de **ppv** de **node921** l'ordre:

```
microk8s kubectl create -f ppv-ingress.yaml
```

i el sistema hauria de mostrar el missatge: **`ingress.networking.k8s.io/proxy-ppv created`**

e) Comprova que l'estat del reverse-proxy. Executa:

```
microk8s kubectl get ingress
```

i el resultat hauria de ser:

```
vagrant@node921:~/projectes/ppv$ microk8s kubectl get ingress
NAME        CLASS    HOSTS             ADDRESS     PORTS   AGE
proxy-ppv   public   ppv.clotfje.net   127.0.0.1   80      4s
```

f) Troba l'adreça IP de la màquina virtual **node921**. Executa: **`ip -4 -br add show eth1`**

g) Dins de la teva màquina host:

- Si treballes amb Windows:
  - Obre amb **notepad** el fitxer **C:\Windows\System32\drivers\etc\hosts**
  - Afegeix al final del fitxer l'entrada la relació entre IP i nom de màquina.
  - Exemple:
    - **192.168.1.36 ppv.clotfje.net**
      S'ha de canviar **192.168.1.36** amb l'adreça obtinguda a l'apartat **f)**.
- Si treballes amb Linux:
  - Obre amb **nano** el fitxer **/etc/hosts**
  - Afegeix al final del fitxer l'entrada la relació entre IP i nom de màquina.
  - Exemple:
    - **192.168.1.36 ppv.clotfje.net**
      S'ha de canviar **192.168.1.36** amb l'adreça obtinguda a l'apartat **f)**.

h) Comprova que pots fer ping de la màquina host a la màquina virtual **node921**.

i) Des del navegador de la teva màquina host accedeix a **http://ppv.clotfje.net:30000** i comprova que pots accedir a l'aplicació.

j) Des d'una altra màquina física, modifica el fitxer hosts perquè apunti a la teva màquina virtual **node921**. Comprova que també es pot fer una connexió a **http:/ppv.clotfje.net:30000**.

Lliurament de l'activitat

a) Mostra el Deployment.
b) Mostra els Pods.
c) Mostra el Service.
d) Mostra el Reverse-Proxy.
e) Accedeix a l'aplicació des de la teva màquina física amb el navegador.
g) Accedeix a l'aplicació des d'una altra màquina física amb el navegador.
h) Data límit per obtenir el 100% de la nota: **dijous 16-1-25** a les **17.45**. Més tard, un 500%.