

KUBERNETES MULTINODO

INSTALACIÓN MULTIPASS

```
sudo apt install snapd
```

```
sudo snap install multipass
```

Creación De Las VMS

Crearemos con multipass 3 máquinas virtuales que serán los nodos, uno será el maestro y otros los otros dos los workers:

```
multipass launch 24.04 --name k3s-master --cpus 2 --memory 2G --disk 10G
```

```
multipass launch 24.04 --name k3s-worker1 --cpus 2 --memory 2G --disk 10G
```

```
multipass launch 24.04 --name k3s-worker2 --cpus 2 --memory 2G --disk 10G
```

Para ver las máquinas podemos ejecutar:

```
multipass list
```

Para borrarlas:

```
multipass delete --purge <name>
```

A las máquinas podemos acceder con ssh o con una opción que multipass nos ofrece:

```
multipass shell <name>
```

Instalación Y Configuración De K3S

Una vez entremos en la máquina virtual o donde queramos instalar k3s, lo haremos mediante el siguiente script:

```
curl -sfL https://get.k3s.io | sh -
```

Para que el usuario sin privilegios pueda utilizarlo y no haya que hacer uso de sudo cada vez que queramos ejecutar el binario kubectl, seguiremos las siguientes instrucciones:

```
mkdir -p ~/.kube
```

```
sudo cp /etc/rancher/k3s/k3s.yaml ~/.kube/config
```

```
sudo chown $USER:$USER ~/.kube/config
```

```
chmod 600 ~/.kube/config
```

```
ubuntu@k3s-master:~$ echo "export KUBECONFIG=~/.kube/config" >>
~/.bashrc
```

```
source ~/.bashrc
```

Para ver si el nodo está activo y que se ha creado:

```
kubectl get nodes
```

Configuración Multinodo

Vamos a consultar el token de nuestro nodo maestro, que será necesario para que los demás nodos se conecten a él:

```
sudo cat /var/lib/rancher/k3s/server/node-token
```

Vamos a copiar dicho token y después vamos a ejecutar el siguiente comando en cada nodo que queramos unir al nodo maestro:

```
curl -sfL https://get.k3s.io | K3S_URL=https://<ip_master>:6443  
K3S_TOKEN=<tu_token> sh -
```

Ahora podemos ver los 3 nodos:

```
ubuntu@k3s-master:~$ k3s kubectl get nodes -o wide  
NAME     STATUS   ROLES      AGE    VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE          KERNEL-VERSION   CONTAINER-RUNTIME  
k3s-master   Ready    control-plane   25m    v1.34.3+k3s3   10.147.215.101   <none>        Ubuntu 24.04.3 LTS   6.8.0-94-generic   containerd://2.1.5-k3s1  
k3s-worker1   Ready    <none>       4m28s   v1.34.3+k3s3   10.147.215.78    <none>        Ubuntu 24.04.3 LTS   6.8.0-94-generic   containerd://2.1.5-k3s1  
k3s-worker2   Ready    <none>       2m8s    v1.34.3+k3s3   10.147.215.207   <none>        Ubuntu 24.04.3 LTS   6.8.0-94-generic   containerd://2.1.5-k3s1
```

Configuración De Almacenamiento Mediante NFS Provisioner

Vamos a crear los volúmenes de forma dinámica y de modo que comparten almacenamiento sin importar el nodo. Primero añadimos el repositorio de nfs provisioner a helm:

```
helm repo add nfs-subdir-external-provisioner  
https://kubernetes-sigs.github.io/nfs-subdir-external-provisioner/
```

```
helm install nfs-subdir-external-provisioner  
nfs-subdir-external-provisioner/nfs-subdir-external-provisioner \  
--set nfs.server=10.147.215.207 \  
--set nfs.path=/srv/nfs/mariadb \  
--set storageClass.name=nfs-csi \  
--namespace nfs-provisioner \  
--create-namespace
```

Otra cosa que debemos tener en cuenta es que para que el volumen se monte en los contenedores del pod, tenemos que instalar en cada nodo el cliente nfs:

sudo apt install nfs-common, para no hacerlo de forma manual, podríamos automatizarlo con ansible, porque en este caso solo tenemos 3 nodos pero si tuvieramos 20 cambiaría la cosa.

Por tanto he configurado el siguiente script para sacar el inventario de ansible de forma automática:

```
#!/bin/bash
echo "[k8s_nodes]"
kubectl get nodes -o jsonpath='{range .items[*]}{.metadata.name}
ansible_host={.status.addresses[?(@.type=="InternalIP")].address}{"
"\n"}{end}'

echo -e "\n[k8s_nodes:vars]"
echo "ansible_user=ubuntu"
echo "ansible_become=yes"
```