Configuración cluster Kubernetes

Contenido

[Introducción 1](#_Toc68003983)

[Instalación de recursos 1](#_Toc68003984)

[Docker 1](#_Toc68003985)

[Kubeadm 1](#_Toc68003986)

[Gcloud 2](#_Toc68003987)

# Introducción

Para configurar Kubernetes, primero debemos tener en cuenta donde lo vamos a configurar. En mi caso creé 3 máquinas virtuales con el sistema operativo [**Ubuntu**](https://ubuntu.com/download/desktop), la versión **20.04**. Para arrancar estas máquinas virtuales, utilicé la aplicación **VMware** Workstation 16 Player.

Una vez creadas las máquinas virtuales, ejecutaremos diferentes comandos dependiendo del tipo de nodo que vaya a ser dentro del sistema Kubernetes. Aun así, primero instalaremos todos los recursos necesarios para la creación y uso del cluster.

# Instalación de recursos

Ejecutaremos estos comandos en todos los nodos.

## Docker

Primero, instalaremos Docker:

> *sudo apt install docker.io*

Si queremos comprobar si Docker se ha instalado: > *docker --version*

Después habilitaremos Docker:

> *sudo systemctl enable docker*

## Kubeadm

Antes de instalar Kubeadm añadiremos la clave de Kubernetes con el siguiente comando:

> *curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add*

Despues, añadiremos el repositorio:

> *sudo apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"*

Ahora sí instalaremos kubeadm:

> *sudo apt install kubeadm*

Si queremos comprobar si Kubeadm se ha instalado: > *kubeadm version*

## Gcloud

Para instalar Gcloud:

> sudo snap install google-cloud-sdk

# Configuración nodo maestro

## **Desactivar la memoria de intercambio**

Para utilizar Kubernetes en las máquinas virtuales, debemos desactivar la memoria de intercambio. Por comando se puede hacer de una manera muy sencilla:

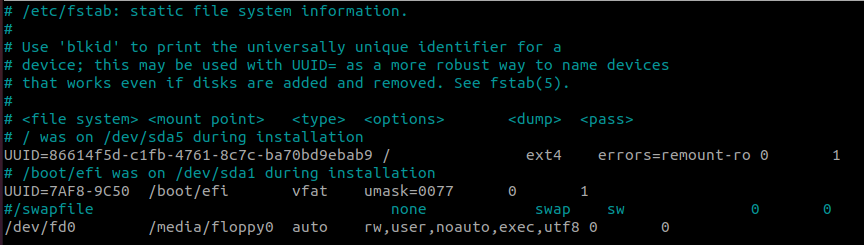
> sudo swapoff -a

Aun así, esto solo desactiva la memoria temporalmente. Es decir, si reiniciamos las máquinas virtuales, tendremos que ejecutar este comando cada vez. Para hacer la desactivación definitiva, tendremos que modificar un archivo: */etc/fstab*.

Para ello, utilizaremos la herramienta *nano*:

> sudo nano /etc/fstab

Dentro del archivo comentaremos la línea donde aparece *swapfile* añadiendo un ‘*#*’ al principio:



Por último, guardaremos el archivo con ‘Crtl+X’, ‘Y’ y ‘Enter’.

## Inicializar el nodo maestro

Para comenzar, le daremos un nombre al nodo maestro, nosotros le hemos llamado “*node1*”, pero se le puede nombrar de cualquier forma (“*master-node*”, “*main-node*” …)

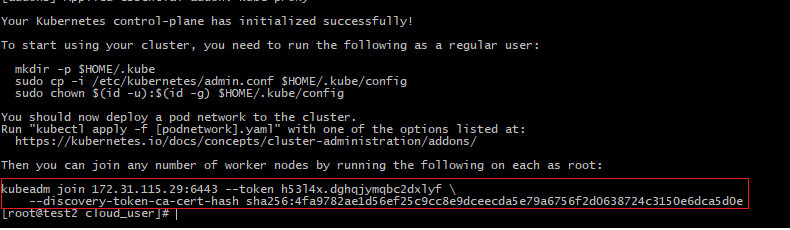
> sudo hostnamectl set-hostname node1

Ahora, sí inicializaremos el nodo maestro. Con este comando, arrancaremos el cluster de kubernetes:

> sudo kubeadm init --pod-network-cidr=10.244.0.0/16

**¡IMPORTANTE!**

Con el comando anterior se creará el cluster, y nos mostrará información que debemos guardar. A parte de decirnos los siguientes comandos que debemos ejecutar, los cuales veremos después, no muestra el **token** y **discovery-token** de nuestro cluster. Es muy importante guardar la información de dentro del cuadro rojo, ya que siempre que queramos importar un nuevo nodo, lo necesitaremos.



Si por algún casual se pierde esa información, se puede recuperar mediante este comando:

> kubeadm token create --print-join-command

Como se puede ver en la imagen, después de ejecutar el comando de *init* hay que ejecutar otros tres comandos:

> mkdir -p $HOME/.kube

> sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

> sudo chown $(id -u):$(id -g) $HOME/.kube/config

Antes de unir ningún nodo al cluster, debemos crear una red de pod para la comunicación entre nodos. Para ello, ejecutaremos este comando en el nodo maestro:

> sudo kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

## Configuración nodos esclavos

Desde los nodos esclavos, nos uniremos al cluster. Como hemos comentado al principio, tendremos que tener bien configurado el nodo, con todo lo necesario instalado.

Para unirnos al cluster, ejecutaremos el siguiente comando:

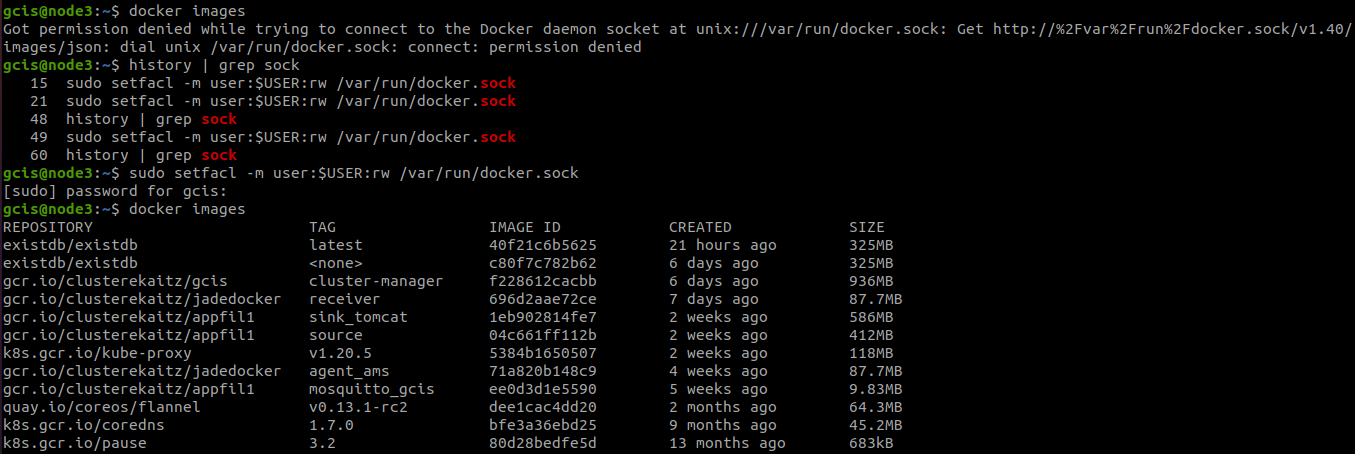
> kubeadm join <tuip> --token <tutoken> --discovery-token-ca-cert-hash sha256:<tuhash>

Si queremos saber las imágenes que tiene el nodo podremos verlas con este comando:

> docker images

A medida que vayamos desplegando aplicaciones, se irán descargando sus imágenes base en estos nodos. Si aparece el error de permiso denegado para el anterior comando, se podrá solucionar:

> sudo setfacl –m user:$USER:rw /var/run/docker.sock



Por último, para configurar el Google Cloud Registry usaremos este simple comando. Durante el proceso deberemos especificar datos como la cuenta de google o el nombre del proyecto.

> gcloud init

A continuación, deberemos configurarlo con Docker:

> gcloud auth configure-docker

Si queremos mostrar las configuraciones de Google Cloud del dispositivo:

> gcloud config list --all

# Comando para usar el cluster

Existen varios comandos muy interesantes a la hora de usar nuestro cluster.

Si queremos conseguir el listado de nodos que componen el cluster:

> kubectl get nodes

Si queremos conseguir el listado de pods que están desplegados en el cluster:

> kubectl get pods

Si queremos conseguir el listado de pods con más información:

> kubectl get pods –o wide

Si queremos conseguir información sobre un pod en concreto:

> kubectl describe pod <id del pod>

Si queremos conseguir el log de un pod (su consola), existen dos formas: la primera printea la consola en ese momento, y la segunda enseña la consola en directo, es decir, si la aplicación escribe algo se actualiza:

> kubectl logs <id del pod> (1)

> kubectl logs -f <id del pod> (2)

Si queremos entrar dentro de un pod en concreto:

> kubectl exec -it <id del pod> -- sh

Si queremos gestionar los deployments:

> kubectl delete/create/restart deploy <nombre del deploy>

> kubectl rollout restart deploy <nombre del deploy>

Si queremos desplegar utilizando archivos YAML:

> kubectl apply/delete –f archivo.yaml

Si queremos desplegar varios archivos YAML:

> kubectl apply/delete –f /path/Carpeta con YAML/