Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

adsis2-tf

Administración de Sistemas II

Autor 1:	Toral Pallás, Héctor - 798095
Grado:	Ingeniería Informática
Curso:	2022-2023

12 de abril de 2023

Índice

1. Resumen

2. Introducción y objetivos

En esta práctica, el objetivo principal es aprender a utilizar Kubernetes, una herramienta muy utilizada para la gestión y despliegue de aplicaciones en contenedores. Para ello, se va a utilizar una guía práctica que proporciona una serie de instrucciones para desplegar cuatro máquinas virtuales utilizando Vagrant, Virtualbox, K3s y kubectl.

En la sección 4.1 se explica la operativa que se va a llevar a cabo con Vagrant para desplegar las máquinas virtuales, y en la 4.2 se va a mostrar el script de aprovisionamiento de las maquinas virtuales y como este configura el cluster k3s.

Además de aprender a utilizar Kubernetes, también se va a analizar el despliegue automatizado de la infraestructura de máquinas virtuales, red, servicios y Kubernetes (K3s) mediante el estudio de los archivos Vagrantfile y provision.sh que se mecionan . Para ello, se van a realizar las secciones 1-12 del guión "Practical Introduction Kubernetes v1".

En resumen, esta práctica se trata de una introducción práctica al uso de Kubernetes y al despliegue automatizado de infraestructuras con Vagrant y programas shell.

3. Arquitectura, Vagrant y Kubernetes

Vagrant es una herramienta de virtualización que permite la creación y configuración automatizada de máquinas virtuales. Se utiliza comúnmente en el desarrollo de software para crear entornos de desarrollo portátiles y reproducibles.

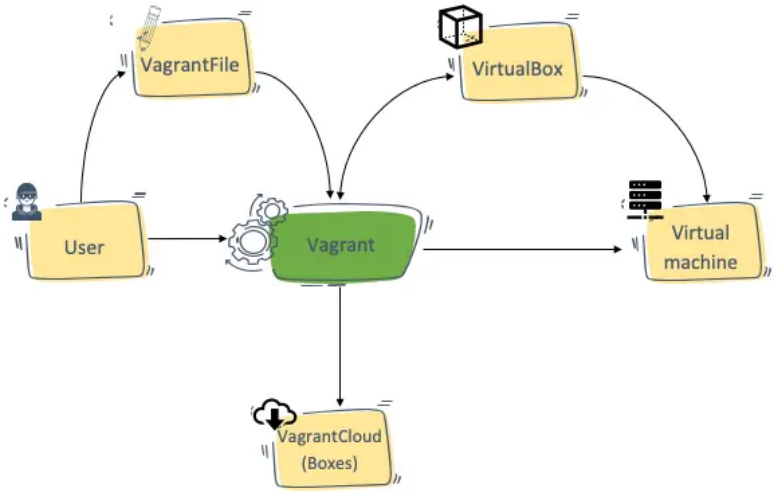

Figura 1: Arquitectura Kubernetes

Para definir la configuración de una máquina virtual con Vagrant, se utiliza un archivo de configuración llamado Vagrantfile. El Vagrantfile incluye información sobre la configuración de la máquina virtual, como la cantidad de memoria RAM, la capacidad de almacenamiento, la dirección IP, la configuración de red y otros parámetros.

Para crear nuevas máquinas virtuales con Vagrant, se utilizan las boxes. Las boxes son imágenes de máquinas virtuales que contienen el sistema operativo y otros paquetes de software preinstalados que se utilizan como base para la creación de nuevas máquinas virtuales.

El funcionamiento y la operación básica de Vagrant comienzan con la instalación de la herramienta en el sistema local. Una vez instalado, se debe crear un Vagrantfile para definir la configuración de la máquina virtual y seleccionar una box. Luego, se puede crear la máquina virtual utilizando el comando "vagrant up.[en] la línea de comandos.

Una vez creada la máquina virtual, se puede provisionar mediante scripts de shell como en nuestro caso mediante "provision.sh." herramientas de configuración como Ansible o Chef.

Por otro lado, Kubernetes es un sistema de orquestación de contenedores que se utiliza para automatizar la implementación, el escalado y la gestión de aplicaciones sobre contenedores. La arquitectura de Kubernetes se divide en dos partes principales: el plano de control (control plane) y los nodos de trabajo (worker nodes).

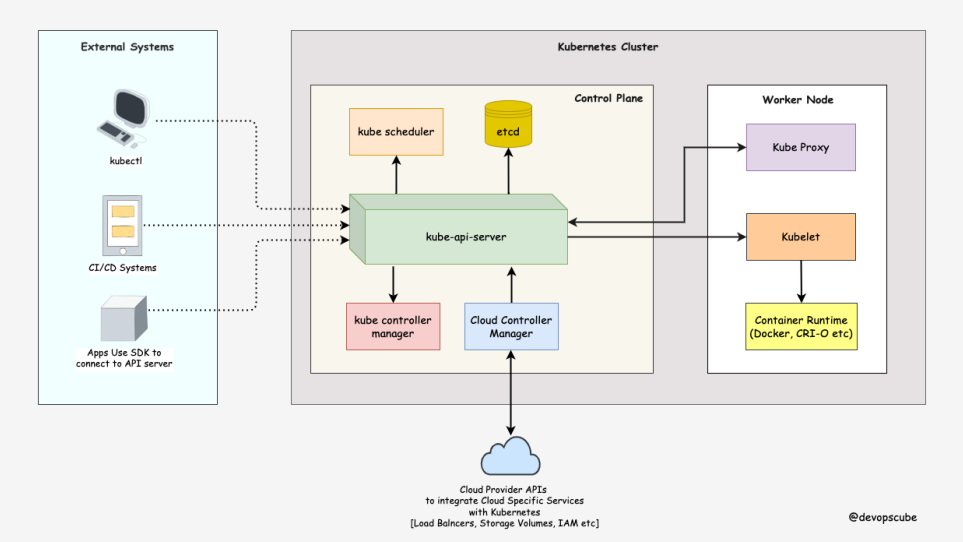

Figura 2: Arquitectura Kubernetes

El plano de control es responsable de administrar el clúster de Kubernetes y sus componentes. Incluye los siguientes componentes:

- **kube-apiserver**: Es el punto de entrada para todas las solicitudes de administración del clúster. Expone la API de Kubernetes y proporciona una interfaz para administrar el clúster.
- **etcd**: Es un almacén de datos distribuido que se utiliza para almacenar el estado del clúster, como los objetos de Kubernetes y la configuración de los componentes.
- **kube-scheduler**: Es responsable de asignar los pods a los nodos de trabajo disponibles en el clúster. Selecciona el nodo adecuado para programar el pod y garantiza que se cumplan los requisitos de recursos y políticas.
- **Kube Controller Manager**: Es un proceso que ejecuta los controladores de Kubernetes en segundo plano. Los controladores son responsables de garantizar que el estado deseado del clúster se mantenga. Por ejemplo, el controlador de réplicas garantiza que se mantenga el número deseado de réplicas de un pod.
- Cloud Controller Manager (CCM): Este componente es opcional y se utiliza para integrar el clúster de Kubernetes con proveedores de servicios en la nube, como AWS, Google Cloud Platform y Azure.

Los nodos de trabajo (workers) son los servidores que ejecutan los pods y sus contenedores. Cada nodo de trabajo tiene los siguientes componentes:

- **Kubelet**: Es el agente de nodo que se ejecuta en cada nodo de trabajo y se comunica con el plano de control. Es responsable de garantizar que los pods se ejecuten correctamente en el nodo de trabajo y de informar al plano de control sobre el estado de los pods.
- **Kube-proxy**: Es responsable de la configuración de red en cada nodo de trabajo. Es un proxy de red que redirige el tráfico de red a los pods y se comunica con el plano de control para garantizar que se cumplan las políticas de red.
- **Container Runtime**: Es el motor de contenedores que se utiliza para ejecutar los contenedores. Kubernetes es compatible con varios motores de contenedores, como Docker, containerd, CRI-O, entre otros.

4. Revisión de código

Para configurar un clúster de Kubernetes utilizando K3s, se ha hecho uso de dos archivos importantes: el Vagrantfile y el provision.sh.

4.1. Vagranfile

El Vagrantfile es un archivo de configuración utilizado por Vagrant que define la configuración de los nodos de cluster de Kubernetes. En este se especifica la distribución del sistema operativo, las características de cada máquina virtual y la configuración de red. También se establecen las opciones de proveedor de virtualización, y se realiza el aprovisionamiento de cada nodo mediante un script shell llamado provision.sh en nuestro caso. Además, se configura un trigger que copia la configuración de acceso al cluster en un PATH para que las herramientas como kubectl puedan acceder a él, solo en la máquina virtual maestra.

4.2. provision.sh

Por otro lado, el archivo provision.sh es un script de shell que se utiliza para realizar la configuración del clúster de Kubernetes en cada nodo de la máquina virtual. En este archivo se especifican las acciones necesarias para instalar y configurar K3s, establecer la comunicación entre los nodos del clúster y configurar los recursos de red necesarios.

El script comienza estableciendo la zona horaria en Europa/Madrid y cambiando el nombre del host, el cual se establece en /etc/hosts. Luego, se copia el archivo binario k3s en la ruta /usr/local/bin/.

Para los nodos tipo "maestro", se ejecuta el comando de instalación de k3s para el servidor y se configura el adaptador de red para la red virtual Flannel. Además, se especifica la dirección IP y el nombre del nodo, se deshabilitan los controladores de ingreso y balanceadores de carga de servicio, se aplica una restricción de nodo y se copia el archivo de configuración k3s.yaml en el directorio /etc/rancher/k3s/k3s.yaml.

En el caso de nodos tipo .ªgente", se ejecuta el comando de instalación de k3s para el agente, especificando la dirección IP y el nombre del nodo, la dirección IP del servidor principal y el token de conexión del clúster. También se especifica el adaptador de red para la red virtual Flannel.

5. Problemas

E0428 09:31:46.617551 22277 memcache.go:265] couldn't get current server API group list: Get "https://127.0.0.1:6443/api?tim dial tcp 127.0.0.1:6443: connect: connection refused E0428 09:31:46.617807 22277 memcache.go:265] couldn't get current server API group list: Get "https://127.0.0.1:6443/api?timeout=32s": dial tcp 127.0.0.1:6443: connect: connection refused E0428 09:31:46.619045 22277 memcache.go:265] couldn't get current server API group list: Get "https://127.0.0.1:6443/api?timeout=32s": dial tcp 127.0.0.1:6443: connect: connection refused E0428 09:31:46.620390 22277 memcache.go:265] couldn't get current server API group list: Get "https://127.0.0.1:6443/api?timeout= dial tcp 127.0.0.1:6443: connect: connection refused E0428 09:31:46.621747 22277 memcache.go:265] couldn't get current server API group list: Get "https://127.0.0.1:6443/api?timeout=32s": dial tcp 127.0.0.1:6443: connect: connection refused The connection to the server 127.0.0.1:6443 was refused - did you specify the right host or port?

El problema reside en que al realizar la asignación de IPs no ha sido posible hacerlas únicas por lo que no podremos comunciarnos desde la máquina host con las maquinas [m, w1, w2, w3]. La manera de solucionarlo o evitar problemas en la realización de las diferentes pruebas es hacer **vagrant ssh m** y ejecutar todo desde el nodo maestro.

6. Anexos

Vagrantfile utilizado.

```
# Distribuci n de sistema operativo
OSD = 'ubuntu/bionic64'
# OSD = 'debian/buster64'

# Identificador definido al comienzo de la asignatura
W = 7

# Configuraci n base de los nodos
MASTER = "192.168.1.#{W}9"
NODES = [
  { hostname: 'm', type: "master", ip: MASTER, mem: 1000, m: MASTER },
  { hostname: 'w1', type: "worker", ip: "192.168.1.#{W}1", mem: 1000, m: MASTER },
  { hostname: 'w2', type: "worker", ip: "192.168.1.#{W}2", mem: 1000, m: MASTER },
  { hostname: 'w3', type: "worker", ip: "192.168.1.#{W}3", mem: 1000, m: MASTER },
]

Vagrant.configure("2") do |config|
  NODES.each do |node|
    config.vm.define node[:hostname] do |nodeconfig|
      nodeconfig.vm.box = OSD
      nodeconfig.vm.hostname = node[:hostname]

      # DOCS: Vagrant: [networking, public_network]
      nodeconfig.vm.network :public_network,
              # Permite conectar las m quinas virtuales a la misma red que el host.
              # Interfaz por defecto a la que hacer el bridge.
              bridge: "wlp0s20f3",

              ip: node[:ip],

              # Especifica una red privada como red interna en VirtualBox
              # virtualbox__intnet: true,

              # Esto no est  prefijado por "virtualbox__", pero es espec fico de
    VirtualBox.
              # (NIC) Network Interface Card
              nic_type: "virtio"


      nodeconfig.vm.provider "virtualbox" do |v|
        v.customize ["modifyvm", :id, "--memory", node[:mem], "--cpus", "1"]

        # Establece el interfaz de red predeterminado
        v.default_nic_type = "virtio"

=begin
```

```
  if node[:type] == "worker"
    nm = node[:hostname]
    unless File.exist?("disk-#{nm}.vdi")
      v.customize ["storagectl", :id, "--name", "VboxSata", "--add", "sata"]
    end
    unless File.exist?("disk-#{nm}.vdi")
      v.customize ["createmedium", "--filename", "disk-#{nm}.vdi","--size", 20*1024]
    end
  end
=end
      end

      nodeconfig.vm.boot_timeout = 400

      # Aprovisionamiento del nodo mediante un script shell
      nodeconfig.vm.provision "shell",
        path: 'provision.sh',
        args: [ node[:hostname], node[:ip], node[:m], node[:type], "#{W}" ]

      # Copia la configuraci n de acceso al cluster en un PATH para que herramientas
      # como kubectl sep n como acceder a este
      if node[:type] == "master"
        nodeconfig.trigger.after :up do |trigger|
          trigger.run = {inline: "sh -c 'cp k3s.yaml /home/hector/.kube/config'"}
        end
      end
    end
  end
end
```

Script para el aprovisionamiento de las boxes de vagrant.

```
#!/bin/bash -x

HOSTNAME=$1
NODEIP=$2
MASTERIP=$3
NODETYPE=$4
W=$5  # Identificador definido al comienzo de la asignatura

timedatectl set-timezone Europe/Madrid

cd /vagrant

echo $1 > /etc/hostname
hostname $1

# Bloque de comandos para modificar el fichero /etc/hosts
MASTER="192.168.1.${W}9"
{ echo 192.168.1.${W}9 m; echo 192.168.1.${W}1 w1; echo 192.168.1.${W}2 w2
  echo 192.168.1.${W}3 w3; cat /etc/hosts
} > /etc/hosts.new
mv /etc/hosts{.new,}

cp k3s /usr/local/bin/

if [ $NODETYPE = "master" ]; then
  INSTALL_K3S_SKIP_DOWNLOAD=true \
  ./install.sh server \
  --token "wCdC16AlP8qpqqI53DM6ujtrfZ7qsEM7PHLxD+Sw+RNK2d1oDJQQOsBkIwy5OZ/5" \
  --flannel-iface enp0s8 \  # interfaz de red para (Flannel) red virtual ejecutada sobre la
    red f sica.
  --bind-address $NODEIP \
  --node-ip $NODEIP --node-name $HOSTNAME \
  --disable traefik \       # Deshabilita el controlador ingress.
  --disable servicelb \     # Deshabilita el balanceador de carga de servicios.
  --node-taint k3s-controlplane=true:NoExecute  # Especifica restricci n en el nodo.
  #--advertise-address $NODEIP
  #--cluster-domain   cluster  .l o c a l
  #--cluster-dns "10.43.0.10"

  cp /etc/rancher/k3s/k3s.yaml /vagrant
```

```
40
41 else
42   INSTALL_K3S_SKIP_DOWNLOAD=true \
43   ./install.sh agent --server https://${MASTERIP}:6443 \
44   --token "wCdC16AlP8qpqqI53DM6ujtrfZ7qsEM7PHLxD+Sw+RNK2d1oDJQQOsBkIwy5OZ/5" \
45   --node-ip $NODEIP --node-name $HOSTNAME --flannel-iface enp0s8
46 fi
```

7. Bilbiografía

- Vagrant - networking
- Vagrant - public_network
- Persistent Volumes
- K3s - Lightweight Kubernetes
- Vagrant Documentation
- Kubernetes index
- Kubernetes Documentation Concepts
- Command line tool (kubectl)