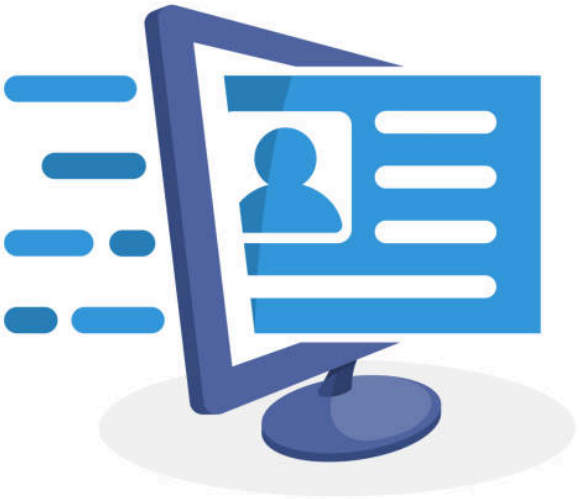
ADMINISTRACIÓN DE SISTEMAS 2

Memoria proyecto
Parte I

Grupo Viernes Mañana



15/04/2024

GUILLERMO BAJO LABORDA, 842748@unizar.es

Universidad Zaragoza
1542

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

Resumen

La primera parte del proyecto se centra en el despliegue y análisis de un cluster Kubernetes mediante el uso de Vagrant sobre VMs de Virtualbox. El objetivo principal es familiarizarse con el entorno de Kubernetes y comprender su funcionalidad y organización. Se despliegan diferentes servicios y aplicaciones, estudiando sus conceptos y recursos asociados. Además, se analiza el despliegue automatizado de VMs, red, servicios y Kubernetes mediante los archivos Vagrantfile y provision.sh.

Introducción y objetivos

Esta primera parte del proyecto tiene como objetivo principal el análisis del despliegue de un cluster Kubernetes, así como el aprendizaje y familiarización del entorno. Para ello, se ha desplegado un cluster Kubernetes utilizando Vagrant sobre VMs de Virtualbox. Se ha hecho especial énfasis en comprender la arquitectura y los elementos clave del despliegue de Vagrant y Kubernetes, además de implementar servicios y aplicaciones en Kubernetes para analizar los conceptos y recursos asociados. Asimismo, se ha llevado a cabo un estudio del despliegue inicial de VMs, red, servicios y Kubernetes mediante los ficheros *Vagrantfile* y *provision.sh.*

Arquitectura de elementos relevantes

La arquitectura del despliegue se basa en la combinación de dos tecnologías clave: Vagrant y Kubernetes. **Vagrant** facilita la configuración de entornos de desarrollo mediante la automatización del despliegue de máquinas virtuales. En esta primera parte del proyecto, Vagrant se encarga de provisionar las VMs de Virtualbox según las especificaciones definidas en el archivo Vagrantfile, lo que incluye la asignación de recursos de hardware, la configuración de la red y la instalación de software adicional necesario para el entorno Kubernetes.

Por otro lado, **Kubernetes** proporciona la infraestructura necesaria para la orquestación de contenedores y la gestión de aplicaciones distribuidas en el cluster. K3s simplifica la implementación de Kubernetes al reducir la complejidad y el consumo de recursos, lo que lo hace especialmente adecuado para entornos de desarrollo y pruebas. En esta arquitectura, Kubernetes se encarga de coordinar la ejecución de contenedores en las VMs provisionadas por Vagrant, asegurando la alta disponibilidad, escalabilidad y rendimiento de las aplicaciones desplegadas.

Aplicaciones desplegadas sobre Kubernetes

A continuación comentaremos algunas de las aplicaciones desplegadas sobre Kubernetes, así como los diferentes recursos utilizados.

Pods

El pod es la unidad más pequeña de implementación que puede contener uno o más contenedores. Los contenedores dentro de un Pod comparten recursos y red, lo que facilita la comunicación entre ellos. Los Pods son gestionados dinámicamente por Kubernetes y pueden ser escalados, replicados y administrados según las necesidades de la aplicación.

Se pueden lanzar Pods utilizando el comando *kubectl run,* especificando la imagen del contenedor y los puertos a exponer, o usando un manifiesto para hacerlo de forma más automatizada. Los recursos en un Pod, como CPU y RAM, pueden ser limitados y solicitados mediante la configuración de recursos en el archivo de manifiesto YAML. Los Pods pueden ser eliminados con el comando kubectl delete pod, pero es recomendable utilizar Deployments para una gestión más controlada y supervisada.

Labels

Las etiquetas en Kubernetes son identificadores clave-valor adjuntos a objetos, como Pods, ndos o Servicios, que se utilizan para organizar recursos. Permiten la categorización de recursos, facilitando la gestión de objetos en el clúster. Conocemos comandos para asignarlas, filtrar por etiquetas, y más.

Nodos

Un nodo en Kubernetes es una máquina donde se ejecutan los componentes de un clúster, como los pods. Los nodos son responsables de ejecutar las cargas de trabajo asignadas por Kubernetes, como aplicaciones y servicios. Cada nodo tiene asignado un rol específico, como maestro o trabajador, y puede ser etiquetado para controlar dónde se ejecutan los pods.

Logging

Los logs en Kubernetes son esenciales para comprender el comportamiento de las aplicaciones y el estado del clúster. Permite acceder a la salida generada por los contenedores, lo que facilita la depuración de problemas y el monitoreo del rendimiento. Además, es posible visualizar registros de pods que ya han finalizado su ciclo de vida, lo que resulta útil para la investigación de problemas pasados. La gestión de registros en Kubernetes es clave para mantener la eficiencia del clúster.

Servidor API de Kubernetes

El acceso al servidor API Kubernetes es útil para propósitos de exploración o pruebas directas. Una opción para lograr esto es mediante la creación de un proxy hacia el entorno local utilizando el comando *kubectl proxy --port=8080.* Esto permite consultar la API utilizando herramientas como curl. Además, para explorar las versiones y recursos admitidos por la API, se pueden utilizar los comandos *kubectl api-versions* y *kubectl api-resources*, respectivamente.

Deployments

Un deployment en Kubernetes es un objeto que administra un conjunto de réplicas de un conjunto de Pods. Proporciona una forma de definir cómo se deben crear y actualizar esos Pods en un clúster de Kubernetes. El ReplicaSet se crea automáticamente cuando se crea un Deployment en Kubernetes. Un ReplicaSet es responsable de garantizar que un número específico de réplicas de un conjunto de Pods esté en ejecución en todo momento. Durante el proceso de implementación, Kubernetes gestiona la creación de nuevas réplicas con la versión actualizada del pod y la terminación de las réplicas antiguas, proporcionando así una implementación controlada y segura de cambios en la aplicación. Además, Kubernetes registra el historial de implementaciones, lo que facilita la supervisión y el seguimiento de los cambios realizados en el clúster.

Servicios

Un Service en Kubernetes proporciona una abstracción para los Pods al ofrecer direcciónes IP virtuales. Un Service permite a los clientes conectarse de manera confiable a los contenedores que se ejecutan en el Pod utilizando la VIP. No es una dirección IP real conectada a una interfaz de red, sino que su propósito es simplemente enrutar el tráfico hacia uno o más Pods. La tarea de mantener actualizada la asignación entre la VIP y los Pods corresponde a kube-proxy (encargado de gestionar las reglas IPtables), un proceso que se ejecuta en cada nodo y que consulta al servidor API para conocer los nuevos servicios en el clúster.Este enfoque asegura que los clientes puedan acceder de manera confiable a los servicios, independientemente de los cambios en la infraestructura del clúster.

Descubrimiento de servicios

El descubrimiento de servicios en Kubernetes permite determinar cómo conectarse a un servicio. Se prefiere el descubrimiento basado en DNS, que ofrece una forma flexible y genérica de conectar servicios en todo el clúster. En el ámbito de descubrimiento de servicios tenemos los jump pods, que se crean con el propósito específico de realizar conexiones o ejecutar comandos dentro de un clúster de Kubernetes. Estos pods actúan como un punto de partida o "salto" para acceder a otros recursos dentro del clúster, como servicios o pods en diferentes namespaces.

Port Forwarding

El port forwarding es una técnica que permite redirigir el tráfico de un puerto local de tu máquina hacia un servicio específico dentro del clúster de Kubernetes. Se suele utilizar para acceder rápidamente a servicios internos del clúster desde un entorno local.Un aspecto importante es recordar que el port forwarding no debe usarse para el tráfico de producción, ya que está diseñado para propósitos de desarrollo y experimentación.

Namespaces

Los namespaces en Kubernetes proporcionan un ámbito para los recursos de Kubernetes, dividiendo tu clúster en unidades más pequeñas. Son como espacios de trabajo compartidos entre varios usuarios. Muchos recursos, como los pods y los servicios, están dentro de un namespace, mientras que otros, como los nodos, no lo están ya que son globales para todo el clúster. Los namespaces son útiles para organizar y gestionar recursos de manera más ordenada, permitiendo a los desarrolladores y administradores trabajar de forma más eficiente y segura. Se pueden crear, listar y eliminar namespaces para separar y controlar el acceso a los recursos en tu clúster.

Volúmenes

Los volúmenes son básicamente directorios accesibles para todos los contenedores que se ejecutan en un pod. A diferencia del sistema de archivos local del contenedor, los datos en los volúmenes se conservan incluso cuando se reinicia el contenedor. El tipo de volumen determina el medio que respalda el volumen y su contenido, que puede ser local del nodo, de intercambio de archivos, específico del proveedor de la nube, de sistema de archivos distribuido o de propósito especial. Un tipo especial de volumen es PersistentVolume. En el siguiente ejemplo, creamos un pod con dos contenedores que comparten un volumen emptyDir para intercambiar datos. Podemos ver cómo se monta y se utiliza el volumen en cada contenedor del pod. Finalmente, eliminamos el pod, lo que también elimina el volumen compartido y su contenido.

Volúmenes persistentes

Un volumen persistente (PV) en Kubernetes es un recurso que se puede utilizar para almacenar datos de manera que persistan más allá de la vida útil de un pod. Está respaldado por un sistema de almacenamiento en red como NFS, o un sistema de archivos distribuido como Ceph. Para utilizar un PV, primero hay que reclamarlo mediante un reclamo de volumen persistente (PVC). El PVC solicita un PV con las especificaciones deseadas a Kubernetes y luego lo une a un pod donde se puede montar como un volumen.

Conceptos Vagrantfile y provision.sh

El Vagrantfile es un archivo de configuración utilizado por Vagrant para definir, configurar y aprovisionar entornos de desarrollo portátiles. En nuestro contexto, se utiliza para describir la infraestructura de máquinas virtuales que compondrán el entorno del clúster de Kubernetes. El archivo especifica detalles como el sistema operativo base, la configuración de red, los recursos de la máquina virtual y las acciones a realizar durante el aprovisionamiento. Además, permite definir múltiples nodos y sus configuraciones individuales. Gracias a Vagrant, podemos automatizar el despliegue y configuración del entorno de manera consistente, lo que facilita el desarrollo, pruebas y colaboración en proyectos de infraestructura.

En cambio, el script provision.sh es un script de aprovisionamiento utilizado para configurar y preparar un nodo en el clúster de Kubernetes. Este script es invocado desde el propio Vagrantfile para realizar el aprovisionamiento de cada una de las máquinas, y se ejecuta dentro de la máquina virtual durante el proceso de aprovisionamiento. El script se encarga de tareas como la configuración del nombre de host, la actualización de archivos de configuración del sistema, la instalación de paquetes adicionales y la configuración específica de K3s. Al utilizar un script de aprovisionamiento, podemos definir y mantener la configuración del nodo de forma reproducible, lo que facilita la creación y administración del clúster de Kubernetes de manera eficiente.

Problemas encontrados y su solución

Apenas se han encontrado problemas a lo largo de la realización de esta primera parte del proyecto de la asignatura. Esto se debe a su carácter introductorio y a la detallada guía que ha sido proporcionada.

Sin embargo, a la hora de lanzar las máquinas virtuales con Vagrant hubo diversas complicaciones. Tras ejecutar el comando *vagrant up*, las máquinas se ejecutaban a priori correctamente, pero al ejecutar el comando *kubectl get nodes*, no se obtenía respuesta alguna. Tras realizar diversas comprobaciones, se llegó a la conclusión de que este problema iba ligado a la red, dado que los problemas solo sucedían al estar trabajando bajo la red eduroam proporcionada por la universidad. En el resto de diferentes redes bajo las que se ha trabajado en esta parte del proyecto, no hubo problema alguno.

Anexo

Vagrantfile:

Unset

```
# Este archivo Vagrantfile está siendo utilizado para configurar y crear múltiples
máquinas virtuales utilizando Vagrant.

# Se definen las boxes que se utilizarán para crear las VMs. La caja Ubu  utiliza la
imagen base ubuntu/bionic64.
Ubu = 'ubuntu/bionic64'

# Se define la dirección IP del nodo master.
MASTER = '192.168.1.209'

# Lista nodos con las características de cada máquina virtual que se creará: hostname,
tipo, dirección IP, memoria asignada y dirección IP del nodo master.
NODES = [
  { hostname: 'm', type: "master", ip: MASTER, mem: 1000, m: MASTER },
  { hostname: 'w1', type: "worker", ip: '192.168.1.201', mem: 1000, m: MASTER },
  { hostname: 'w2', type: "worker", ip: '192.168.1.202', mem: 1000, m: MASTER },
  { hostname: 'w3', type: "worker", ip: '192.168.1.203', mem: 1000, m: MASTER },
]

# Configuración de las VMs utilizando el bloque Vagrant.configure. Se itera sobre cada
VM definida en NODES
Vagrant.configure("2") do |config|
    # Iteramos sobre cada nodo definido en NODES
    NODES.each do |node|

        # Configuración específica para cada nodo
        config.vm.define node[:hostname] do |nodeconfig|
            # Configuración de la box y hostname de la VM
            nodeconfig.vm.box = Ubu
            nodeconfig.vm.hostname = node[:hostname]

            # Configuración de la red pública
            nodeconfig.vm.network :public_network,
                    bridge: "wlp0s20f3", # Nombre del bridge de red
                    ip: node[:ip],       # Dirección IP del nodo
                    nic_type: "virtio"   # Tipo de interfaz de red

            # Configuración específica del proveedor VirtualBox
            nodeconfig.vm.provider "virtualbox" do |v|
                # Asignación de recursos de la VM
                v.customize ["modifyvm",:id,"--memory",node[:mem],"--cpus","1"]
                # Tipo de interfaz de red predeterminado
                v.default_nic_type = "virtio"
            end

            # Configuración del tiempo de arranque máximo
            nodeconfig.vm.boot_timeout = 400
```

```
            # Provisión de la VM utilizando un script shell
            nodeconfig.vm.provision "shell",
                # Ruta del script de provisionamiento
                path: 'provision.sh',
                 # Argumentos para el script
                args: [ node[:hostname], node[:ip], node[:m], node[:type] ]

            # Si el nodo es de tipo "master", se ejecuta un trigger después de que la
VM esté activa
            if node[:type] == "master"
                nodeconfig.trigger.after :up do |trigger|
                    # Se copia el archivo de configuración de k3s en el directorio
.kube de mi usuario personal
                    trigger.run = \
                        {inline: "sh -c 'cp k3s.yaml
/home/guillermobajo/.kube/config'"}
                end
            end
        end
    end
end
```

Provision.sh

Unset

```
#!/bin/bash -x
# Este script se utiliza para aprovisionar y configurar un nodo en un clúster de
Kubernetes utilizando K3s.

# Definición de variables
HOSTNAME=$1
NODEIP=$2
MASTERIP=$3
NODETYPE=$4

# Cnfiguración de la zona horaria
timedatectl set-timezone Europe/Madrid

cd /vagrant
# Cambiar el nombre del host
echo $1 > /etc/hostname
hostname $1

# Actualización del archivo /etc/hosts para incluir las direcciones IP y nombres de
host de todos los nodos
{ echo 192.168.1.201 w1; echo 192.168.1.202 w2
  echo 192.168.1.203 w3; cat /etc/hosts
} > /etc/hosts.new
mv /etc/hosts{.new,}
```

```
# Copiar el binario de K3s al directorio /usr/local/bin/
cp k3s /usr/local/bin/

# Instalación de K3s dependiendo del tipo de nodo

# Si el nodo es un nodo maestro, se instala K3s en modo servidor
if [ $NODETYPE = "master" ]; then
  INSTALL_K3S_SKIP_DOWNLOAD=true \
  ./install.sh server \
   # Token de acceso para unirse al clúster
  --token "wCdC16AlP8qpqqI53DM6ujtrfZ7qsEM7PHLxD+Sw+RNK2d1oDJQQOsBkIwy5OZ/5" \
  # Interfaz de red para Flannel
  --flannel-iface enp0s8 \
  # Dirección IP a la que se enlazará el servidor de K3s
  --bind-address $NODEIP \
   # Dirección IP y nombre del nodo
  --node-ip $NODEIP --node-name $HOSTNAME \
   # Deshabilitar Traefik y ServiceLB
  --disable traefik \
  --disable servicelb \
  --node-taint k3s-controlplane=true:NoExecute

  # Copiar el archivo de configuración de K3s para acceso remoto
  cp /etc/rancher/k3s/k3s.yaml /vagrant

else
  # Si el nodo es un nodo de trabajo, se instala K3s en modo agente
  INSTALL_K3S_SKIP_DOWNLOAD=true \
  # URL del servidor maestro
  ./install.sh agent --server https://${MASTERIP}:6443 \
  # Token de acceso para unirse al clúster
  --token "wCdC16AlP8qpqqI53DM6ujtrfZ7qsEM7PHLxD+Sw+RNK2d1oDJQQOsBkIwy5OZ/5" \
  # Dirección IP, nombre del nodo y configuración de Flannel
  --node-ip $NODEIP --node-name $HOSTNAME --flannel-iface enp0s8
fi
```