HashiCorp

Vagrant

Vagrant y VirtualBox

Vagrant es una herramienta de código abierto que permite la creación y gestión de entornos virtualizados.

Permite definir y configurar máquinas virtuales y entornos a través de archivos de configuración (IaC).

Para lograr la virtualización, necesitamos un hipervisor, que es un software que permite ejecutar múltiples máquinas virtuales en un único hardware físico.

VirtualBox es uno de los hipervisores más utilizados y permite crear y gestionar máquinas virtuales de manera eficiente.

Diferencias entre Docker y Vagrant

Docker

- Docker se enfoca en contenedores ligeros y portables.
- Tecnología de contenedores para encapsular aplicaciones y dependencias.
- Requiere menos recursos en comparación con máquinas virtuales.
- Ideal para empaquetar aplicaciones y ejecutarlas de manera consistente en diferentes entornos.

Vagrant

- Vagrant se centra en la gestión de máquinas virtuales completas.
- Utiliza hipervisores como VirtualBox o VMware para crear máquinas virtuales.
- Enfoque más pesado en recursos debido al uso de máquinas virtuales.
- Permite replicar entornos de producción completos en máquinas virtuales.

Instalación de VirtualBox

Para comenzar, necesitamos instalar VirtualBox en tu sistema.

El instalador de VirtualBox peude descargarse desde el sitio web oficial de VirtualBox (https://www.virtualbox.org/).

En el caso de Linux, también puedes instalar VirtualBox desde los repositorios oficiales de tu distribución:

```
sudo apt update
sudo apt install virtualbox
```

Instalación de Vagrant

Una vez instalado VirtualBox, instalaremos Vagrant en el sistema.

Instalación de Vagrant (Linux)

```
sudo apt update
sudo apt install vagrant
```

Instalación de Vagrant (MacOS)

```
brew install --cask vagrant
```

Instalación de Vagrant (Windows)

Descarga el instalador de Vagrant desde el sitio web oficial de Vagrant (https://www.vagrantup.com/) y ejecuta el instalador.

Creación de una máquina Vagrant

Primero, crea un directorio de trabajo para tus máquinas Vagrant.

```
mkdir mi_proyecto_vagrant
cd mi_proyecto_vagrant
```

Dentro del directorio de trabajo, inicializa un proyecto Vagrant:

```
vagrant init
```

Este comando crea un archivo llamado `Vagrantfile`:

- Configuración básica que define características como la imagen base y los recursos de la máquina virtual
- Plantilla para personalizar la configuración de tu máquina virtual

Descarga de una imagen de máquina virtual

Podemos usar Vagrant para descargar imágenes de máquinas virtuales de proveedores como HashiCorp, Atlas o Vagrant Cloud.

https://app.vagrantup.com/boxes/search

Para descargar una imagen (box) de máquina virtual compatible con Vagrant:

```
vagrant box add [nombre_box]
vagrant box add ubuntu/jammy64
```

Vagrant añadirá la imagen a tu colección local de boxes.

```
vagrant box list
```

`vagrant box remove [nombre_box]` permite eliminar una imagen de la colección.

Configuración del archivo Vagrantfile (I)

```
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/jammy64"
  config.vm.network "private_network", type: "dhcp"
end
```

- `Vagrant.configure("2")`: versión de la configuración de Vagrant
- `config.vm.box`: imagen de la máquina virtual que se utilizará como base (la intentará descargar si no está disponible localmente)
- `config.vm.network`: configura la red de la máquina virtual, podemos indicar una ip específica de la siguiente forma:

```
  config.vm.network "private_network", ip: "192.168.1.50"
```

Configuración del archivo Vagrantfile (II)

```
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/bionic64"
  config.vm.network "private_network", type: "dhcp"
  config.vm.provider "virtualbox" do |vb|
    vb.memory = "1024"
    vb.cpus = 2
  end
end
```

- `config.vm.provider` : configura el proveedor de virtualización (en este caso, VirtualBox)
- `vb.memory` : configura la cantidad de memoria RAM asignada a la máquina virtual
- `vb.cpus` : configura la cantidad de CPUs asignadas a la máquina virtual

Configuración del archivo Vagrantfile (III)

```
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/bionic64"
  config.vm.network "private_network", type: "dhcp"
  config.vm.network "forwarded_port", guest: 80, host: 8080
end
```

- `config.vm.network "forwarded_port"` : configura el reenvío de puertos para permitir el acceso desde el host local a un servicio en la máquina virtual (por ejemplo, una página servida con nginx)

Se pueden redireccionar tantos puertos como sea necesario:

```
  config.vm.network "forwarded_port", guest: 8080, host: 8080
  config.vm.network "forwarded_port", guest: 8081, host: 8081
```

Esto será necesario para acceder a la máquina virtual desde el navegador web del host local.

Otras configuraciones del archivo Vagrantfile

La clave SSH se genera automáticamente al crear una máquina virtual Vagrant.

Si deseas personalizar la clave SSH, puedes hacerlo en el archivo Vagrantfile:

```
Vagrant.configure("2") do |config|
  config.ssh.insert_key = false
end
```

- `config.ssh.insert_key = false`: desactiva la inserción automática de la clave SSH
- Este paso es opcional y no es necesario a menos que desees personalizar la clave SSH
- Insertar la clave SSH automáticamente **puede tomar más tiempo** al iniciar la máquina

Inicio de la máquina virtual

Inicia tu máquina virtual Vagrant con el siguiente comando:

```
vagrant up
```

Si la imagen indicada en el archivo `Vagrantfile` no está disponible localmente, Vagrant la intentará descargar automáticamente.

Una vez finalizada la descarga, Vagrant creará la máquina virtual y la iniciará.

Si ejecutas `vagrant up` después de modificar la configuración de una máquina virtual en el archivo Vagrantfile, Vagrant aplicará las modificaciones al iniciar la máquina virtual.

Algunas configuraciones pueden requerir ciertas acciones adicionales, como la reconstrucción de la máquina virtual.

Acceso la máquina virtual

Accede a la máquina virtual a través de SSH con el siguiente comando:

```
vagrant ssh
```

Podemos salir de la sesión SSH con el comando `exit`.

Administración de la máquina virtual (I)

vagrant halt

`vagrant halt` se utiliza para apagar una máquina Vagrant:

```
vagrant halt
```

- Conserva el estado de la máquina virtual y la detiene temporalmente
- Útil cuando deseas apagar completamente la máquina virtual y no necesitas que consuma recursos mientras está apagada
- Puedes retomar tu trabajo más tarde sin perder cambios en el sistema.

Para reiniciar la máquina, basta con ejecutar el comando `vagrant up`.

Administración de la máquina virtual (II)

vagrant suspend

`vagrant suspend` se utiliza para suspender una máquina Vagrant:

```
vagrant suspend
```

- La máquina virtual se detiene, pero a diferencia de `vagrant halt`, se guarda en un estado de suspensión
- El estado de la máquina se conserva en el disco, pero no se ejecuta ningún proceso en segundo plano
- Útil para ahorrar recursos del sistema mientras no se usa la máquina

Puede reanudarse con `vagrant up`.

Administración de la máquina virtual (III)

vagrant destroy

El comando `vagrant destroy` se utiliza para eliminar una máquina virtual Vagrant y todos sus recursos asociados:

```
vagrant destroy
```

Este proceso es irreversible y no podremos recuperar la máquina virtual una vez eliminada.

Aprovisionamiento con Vagrantfile

El aprovisionamiento es el proceso de configuración de una máquina virtual con herramientas y aplicaciones.

Vagrantfile permite definir el aprovisionamiento de la máquina virtual:

```
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/bionic64"
  config.vm.network "private_network", type: "dhcp"
  config.vm.provision "shell", inline: <<-SHELL
    apt-get update
    apt-get install -y apache2
  SHELL
end
```

- `config.vm.provision "shell"`: ejecuta comandos de shell en la máquina virtual durante el aprovisionamiento

Además de shell, Vagrant permite utilizar otros aprovisionadores como Ansible, Chef, Docker, Puppet y Salt.

Vagrantfile con Docker (I)

Vagrant también permite crear entornos con contenedores Docker.

```
Vagrant.configure("2") do |config|
  config.vm.provider "docker" do |d|
    d.image = "nginx:latest"
    d.ports = ["8083:80"]
  end
end
```

- `d.image` : indica la imagen de Docker que se utilizará como base
- `d.ports` : configura el reenvío de puertos para permitir el acceso desde el host local a un servicio en el contenedor Docker (host:contenedor)

(Opcional) Podemos indicar que el proveedor es Docker al iniciar la máquina:

```
vagrant up --provider=docker
```

Vagrantfile con Docker (II)

La configuración de Vagrantfile también permite usar un Dockerfile para construir una imagen personalizada:

```
Vagrant.configure("2") do |config|
  config.vm.provider "docker" do |d|
    d.build_dir = "."
  end
end
```

- `d.build_dir = "."`: usa el Dockerfile en el directorio actual para construir la imagen

```
  FROM nginx:latest
  RUN apt-get update
  RUN apt-get install -y nano
```

Vagrantfile con Docker (III)

Para acceder a una máquina creada con Docker, en vez de usar `vagrant ssh`, usaremos el comando `docker exec` ya que realmente es un contenedor Docker:

```
docker exec -it [nombre_contenedor] bash
```

Las diferentes opciones de configuración al usar Docker como proveedor de Vagrant se pueden consultar en la documentación oficial:
https://developer.hashicorp.com/vagrant/docs/providers/docker/configuration

Directorio compartido (I)

- Una de las características clave de Vagrant es la capacidad de compartir archivos y directorios entre la máquina virtual y el host.
- Esto facilita el desarrollo y la colaboración al permitir que los archivos sean accesibles desde ambos entornos.
- Todos los archivos en el directorio del proyecto Vagrant en el host se comparten con la máquina virtual en el directorio `/vagrant`.
- Sin embargo, este directorio compartido no es persistente y se elimina cuando se destruye la máquina virtual.
- Para crear un directorio compartido persistente, debemos configurarlo en el archivo `Vagrantfile`.

Directorio compartido (II)

La configuración del directorio compartido se realiza en el archivo `Vagrantfile`, donde puedes especificar qué directorios del host deseas compartir con la máquina virtual y dónde deseas montarlos en la máquina virtual:

```
Vagrant.configure("2") do |config|
  # Otras configuraciones de la máquina virtual...

  # Configuración del directorio compartido
  config.vm.synced_folder "/mihost/directorio_host", "/directorio_virtual"
end
```

- `"/mihost/directorio_host"`: Ruta del directorio en tu host que deseas compartir.
- `"/directorio_virtual"`: Ruta en la máquina virtual donde deseas montar el directorio compartido.

Directorio compartido (III)

El directorio compartido ofrece varias ventajas:

- Los archivos del host son accesibles en la máquina virtual y viceversa
- Los archivos en el directorio compartido persisten incluso después de apagar o destruir la máquina virtual
- Vagrant monitorea y sincroniza los cambios en el host

Directorio compartido (IV)

Ejemplo de uso:

- Tienes un proyecto web y deseas editar el código desde el host
- Puedes configurar un directorio compartido para que los archivos de tu proyecto estén disponibles tanto en el host como en la máquina virtual:

```
config.vm.synced_folder "/ruta/al/proyecto", "/var/www/html"
```

Esto permitirá que la máquina Vagrant sirva los archivos modificados en el host.

Ejercicio 1

1. Crea una máquina virtual Vagrant estas especificaciones:
    - Sistema operativo Ubuntu, 2 GB de RAM, 2 CPUs virtuales
2. Aprovisiona la máquina virtual con nginx desde el archivo Vagrantfile.
3. Redirecciona el puerto 80 de la máquina virtual al puerto 8086 del host local.
4. Accede a la máquina virtual desde el navegador web del host local.

Ejercicio 2

1. Crea una máquina virtual Vagrant estas especificaciones:
    - Sistema operativo Ubuntu, 1 GB de RAM, 2 CPUs virtuales
2. Aprovisiona la máquina virtual con nginx desde el archivo Vagrantfile.
3. Indica la dirección IP estática `192.168.56.10` para la máquina virtual.
4. Crea un directorio compartido entre el host y la máquina virtual en `/var/www/html`.
   El directorio del host debe contener un archivo de imagen (png, jpg...).
5. Accede a la imagen desde el navegador web usando la IP estática.
6. Accede a la imagen desde el navegador web usando el puerto 8087.
7. Modifica el index.html de nginx sin acceder a la máquina virtual.