Vagrant

Miguel Ortuño
Escuela de Ingeniería de Fuenlabrada
Universidad Rey Juan Carlos

Octubre de 2024

Vagrant

`https://www.vagrantup.com`

- Es una herramienta para construir y gestionar entornos de máquinas virtuales.
- Creado en 2010, es software libre, muy popular
- Funciona sobre Linux, FreeBSD, macOS, y Microsoft Windows

- Soporta las principales plataformas de virtualización: Docker, VirtualBox, VMware, AWS, Azure, entre otras.
  Vagrant las denomina *providers*
- Su función básica es reemplazar el interfaz (tanto gráfico como de texto) de estas plataformas, proporcionando un interfaz de texto, programable y homogéneo que permite preparar las máquinas, levantarlas, configurarlas, etc.
  - Lo fundamental es ser *homogéneo*, esto es, como administradores podemos configurar y gestionar muchas plataformas distintas, como si fueran la misma
- Usando Vagrant, resulta muy sencillo migrar entre diferentes tecnologías de virtualización
- Para la configuración, se integra con Ansible, Chef y Puppet, entre otras

Instalación

El paquete vagrant no está disponible en los repositorios oficiales de Ubuntu, Las instrucciones de instalación están disponibles en

`https://developer.hashicorp.com/vagrant/install?product_intent=vagrant`

Las órdenes son:

```
wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o
    /usr/share/keyrings/hashicorp-archive-keyring.gpg

echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg]
    https://apt.releases.hashicorp.com $(lsb_release -cs) main" |
    sudo tee /etc/apt/sources.list.d/hashicorp.list

sudo apt update

sudo apt install -y vagrant
```

Uso de Vagrant

Con Vagrant, es muy fácil crear y poner en marcha una máquina virtual, por ejemplo con VirtualBox

- Si no indicamos el *provider*, Vagrant usa VirtualBox. Es necesario haberlo instalado previamente
- No es necesario lanzar el GUI de VirtualBox, pero también podemos usarlo simultáneamente
- Todo lo relativo a una máquina virtual a manejar con vagrant se guarda en un directorio denominado *project directory*

Vagrant Box

- Vagrant cuenta con repositorios de imágenes preconfiguradas. Las denomina *boxes*. Hay *boxes* oficiales, y también cualquier usuario puede preparar sus boxes y hacerlos públicas gratuitamente
  - Se pueden preparar boxes privados, estos son de pago

Puesta en marcha de un Box

1. Creamos en nuestro *host* el *project directory*
2. Accedemos al *project directory*
3. Ejecutamos `vagrant init <NOMBRE_DE_BOX>`
   p.e.
   `vagrant init bento/ubuntu-24.04`
4. Encendemos la máquina
   `vagrant up`
5. Entramos en la máquina
   `vagrant ssh`
   De esta forma tenemos una sesión con el usuario *vagrant*, preconfigurado para poder lanzar procesos como root sin escribir contraseña

Observa que nunca indicamos con qué máquina queremos trabajar, basta con lanzar la orden `vagrant` desde el *project directory* que necesitemos en cada momento

- Vagrant redirecciona automáticamente un puerto del *host* al puerto 22 del *guest* para poder hacer ssh
  Si está libre, el 2222. Si no, usará otro. Lo indicará en el arranque de la máquina
  - Podemos entrar en el *guest* ejecutando en el *host* la orden
    `ssh -p 2222 usuario@localhost`
  - Pero por omisión no podemos autenticarnos mediante contraseña, sino mediante *authorized keys*
- Vagrant monta automáticamente el *project directory* del *host* en el directorio `/vagrant` del *guest*

Parada de una máquina

Tres formas distintas:

- `vagrant suspend`
  Duerme la máquina
- `vagrant halt`
  Para la máquina
- `vagrant destroy`
  Para la máquina, borra su imagen y todos sus ficheros

Naturalmente, estas instrucciones debemos ejecutarlas desde la máquina donde está vagrant, esto es, el *host*, no el *guest*

Fichero Vagrantfile

- El comando `vagrant init` crea automáticamente un fichero *Vagrantfile* en el *project directory*. En este fichero se especifican todos los detalles del *box*, incluyendo el provisionado
- Si ya disponemos de un fichero *Vagrantfile*, bien porque lo hayamos creado a mano desde cero o bien porque hayamos retocado un fichero *Vagrantfile* anterior, entonces omitimos el `vagrant init`
- Al ejecutar el comando `vagrant up`
  - Si la máquina no existía previamente en el *provider* (p.e. VirtualBox), Vagrant ordena al *provider* crear una máquina nueva, según lo especificado en el fichero *Vagrantfile*
  - Si la máquina ya existe, se pone en marcha, sin más

  En otras palabras: si modificamos el *Vagrantfile*, para que los cambios surgan efecto, es necesario destruir la máquina virtual.

Las opciones de configuración se escriben entre las líneas

```
Vagrant.configure("2") do |config|
```

y

```
end
```

Para cambiar el nombre de la máquina virtual (el nombre que usa el *provider*

- `config.vm.hostname= "MI_MAQUINA"`

Cambiar el nombre de host:

- `config.vm.define "MI_MAQUINA"  # sin '='`

Redireccionamiento de un puerto del *host* al *guest* al

- ```
  config.vm.network "forwarded_port", guest: 80, host: 8080,
      host_ip: "127.0.0.1"
  ```

Los boxes preconfigurados suelen tener un usuario *vagrant*, su claves se guardan en el *project directory*, en

`.vagrant/machines/default/virtualbox/private_key`
  ```

Provisionamiento de la máquina

- Se denomina *provisionar* la máquina a configurar de forma automática los ajustes que necesitemos en la máquina virtual
- La forma más sencilla es con scripts de shell, en entornos más exigentes podemos usar puppet, ansible, etc

En el *project directory* de Vagrant, en el fichero *Vagrantfile*, al final, encontraremos estas líneas, comentadas:

```
# config.vm.provision "shell", inline: <<-SHELL
#   apt-get update
#   apt-get install -y apache2
# SHELL
```

Para provisionar la máquina con comandos de shell

1. Descomentamos esas líneas
2. Reemplazamos las órdenes entre `<<-SHELL` y `SHELL` por las que necesitemos: instalación de paquetes, creación de grupos, usuarios, etc

Estos comandos se ejecutarán cada vez que preparemos una máquina con *vagrant up*

- Si creamos un usuario con `adduser` en el provisionamiento automático, es conveniente añadir la opción `--disabled-password` de la contraseña desde la entrada estándar
- Una forma de abrir sesión de un usuario que no tiene contraseña es que el *root* ejecute `su jperez`, y luego cambie la contraseña a mano
- Recuerda que también se pueden usar las órdenes
  `echo "jperez:sesamo" | chpassw`
  aunque este enfoque no es muy seguro, porque la contraseña quedaría visible en el fichero de provisionamiento