Vagrant

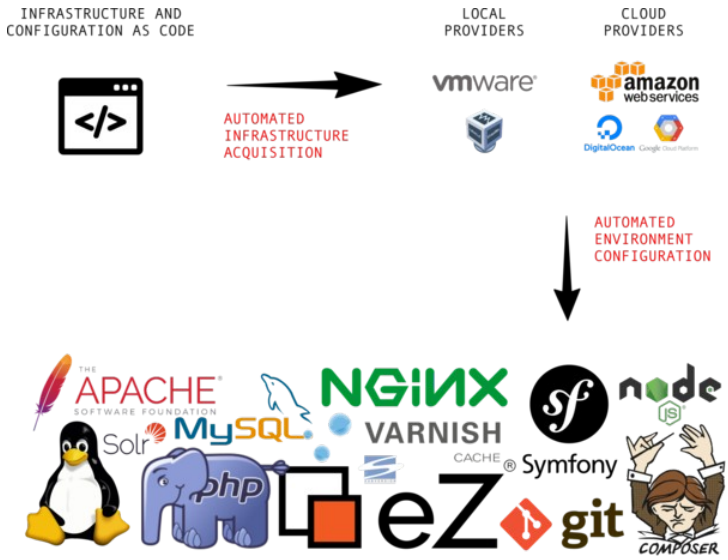
Vagrant es una herramienta de software que proporciona entornos de trabajo portables y reproducibles usando máquinas virtuales. Con Vagrant se puede conseguir de forma sencilla que todos los miembros de un grupo (p.e. desarrolladores) tengan el mismo entorno de trabajo, mismas versiones de software, dependencias, configuración, …, eliminando el famoso 'en mi máquina funciona' causado por trabajar con diferentes configuraciones o versiones de software.

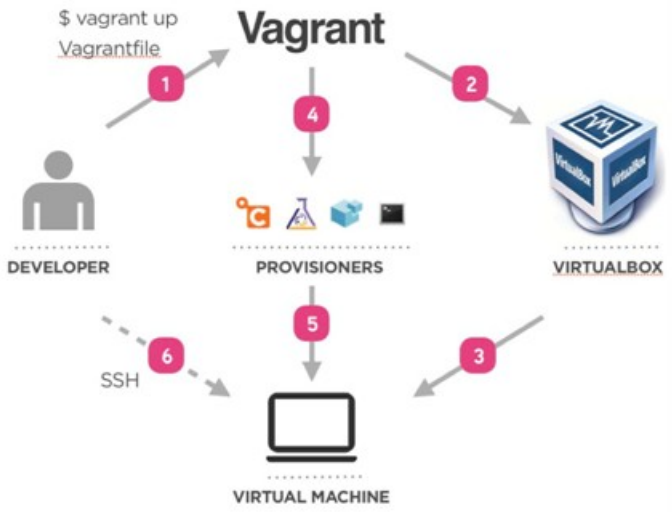
Vagrant se puede considerar una herramienta de *Infrastructure as code* (*IaC*), donde se aprovisiona y gestiona una infraestructura IT a través de código en vez de procedimientos estándar o manuales. Básicamente se tratan los servidores, bases de datos, redes y otra infraestructura como software. Iac permite automatizar el despliegue de una infraestructura de forma que sea un proceso consistente, repetible, rápido, sencillo y con ahorro de costes.

Terminología

En Vagrant se usan los siguientes términos:

- **Box**: imagen base compatible con un *provider* (p.e. Virtualbox) a partir de la que Vagrant creará nuevas máquinas virtuales.
- **Host**: equipo anfitrión en el que está instalado Vagrant.
- **Guest**: máquina virtual creada por Vagrant.
- **Providers**: sistema de virtualización a usar. Aunque Virtualbox es el provider por defecto, Vagrant puede trabajar con otros como KVM, VMWARE, ...
- **Plugins**: permiten añadir a Vagrant funcionalidades extras, como nuevos providers.[1] Por ejemplo, el plugin *vagrant-vbguest* permite actualizar automáticamente las Virtualbox Guest additions si fuese necesario.
- **Provisioners**: permiten automatizar la configuración de las máquinas virtuales instalando paquetes, configurando servicios o realizando otro tipo de tareas. Aunque su uso no es obligatorio; el no usarlos, limita a Vagrant al papel de describir e implementar una infraestructura, cosa que podría hacerse manualmente. Entre otros permite scripts (y comandos) de shell, Ansible, Chef y puppet.
- **Vagrantfile**: fichero que describe las máquinas necesarias (guests) para un proyecto y como configurarlas y aprovisionarlas. Su sintaxis se basa en Ruby y consta de un conjunto de directivas a las que se le asignan los valores que nos interesan.
- **Synced folders**: carpetas compartidas entre el host y los guests. Esto permite; por ejemplo, que se trabaje en el host con un IDE determinado y se prueben inmediatamente los desarrollos en un servidor virtual (guest).





1 https://github.com/hashicorp/vagrant/wiki/Available-Vagrant-Plugins

Escenario#1: Instalación y primer proyecto

Aunque en los repositorios de Ubuntu hay un paquete para instalar Vagrant, éste suele ser más antiguo que la versión oficial disponible en la página oficial de descargas[2]. Se siguen las instrucciones de instalación para Ubuntu:

```
$ curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo apt-key add -
$ sudo apt-add-repository "deb [arch=amd64] https://apt.releases.hashicorp.com $(lsb_release -cs) main"
$ sudo apt-get update && sudo apt-get install vagrant
```

Con `vagrant version` podremos saber que versión está instalada y si hay una versión más reciente:

```
manuel@pcmanuel:~$ vagrant version
Installed Version: 2.2.19
Latest Version: 2.2.19
```

Para nuestro primer proyecto con Vagrant creamos una carpeta y una vez dentro de ella ejecutamos el comando `vagrant init --minimal` para crear un archivo Vagrantfile básico con el que comenzar:

```
manuel@pcmanuel:~/Vagrant$ mkdir proyecto01
manuel@pcmanuel:~/Vagrant$ cd proyecto01
manuel@pcmanuel:~/Vagrant/proyecto01$ vagrant init --minimal
A `Vagrantfile` has been placed in this directory. You are now
ready to `vagrant up` your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
`vagrantup.com` for more information on using Vagrant.
manuel@pcmanuel:~/Vagrant/proyecto01$ cat Vagrantfile
Vagrant.configure("2") do |config|
  config.vm.box = "base"
end
manuel@pcmanuel:~/Vagrant/proyecto01$
```

Si en vez de `vagrant init --minimal` se hubiese ejecutado `vagrant init` se obtendría un Vagrantfile de ejemplo más completo con muchas directivas y comentarios explicativos.

Nuestro Vagrantfile contiene la directiva `config.vm.box` que sirve para especifiar que box se va a usar como plantilla para crear la máquina virtual. Editamos el fichero e indicamos que queremos usar Ubuntu Xenial de 64 bits[3]:

```
manuel@pcmanuel:~/Vagrant/proyecto01$ nano Vagrantfile
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/focal64"
end
```

Una vez indicada la imagen, procedemos a ejecutar `vagrant up` para lanzar el proyecto y crear el entorno de trabajo. Se destacan en negrita algunos pasos importantes de la operación:

```
manuel@pcmanuel:~/Vagrant/proyecto01$ vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Box 'ubuntu/focal64' could not be found. Attempting to find and install...
    default: Box Provider: virtualbox
    default: Box Version: >= 0
==> default: Loading metadata for box 'ubuntu/focal64'
    default: URL: https://vagrantcloud.com/ubuntu/focal64
==> default: Adding box 'ubuntu/focal64' (v20211026.0.0) for provider: virtualbox
    default: Downloading: https://vagrantcloud.com/ubuntu/boxes/focal64/versions/20211026.0.0/
providers/virtualbox.box
Download redirected to host: cloud-images.ubuntu.com
==> default: Successfully added box 'ubuntu/focal64' (v20211026.0.0) for 'virtualbox'!
==> default: Importing base box 'ubuntu/focal64'...
==> default: Matching MAC address for NAT networking...
==> default: Checking if box 'ubuntu/focal64' version '20211026.0.0' is up to date...
```

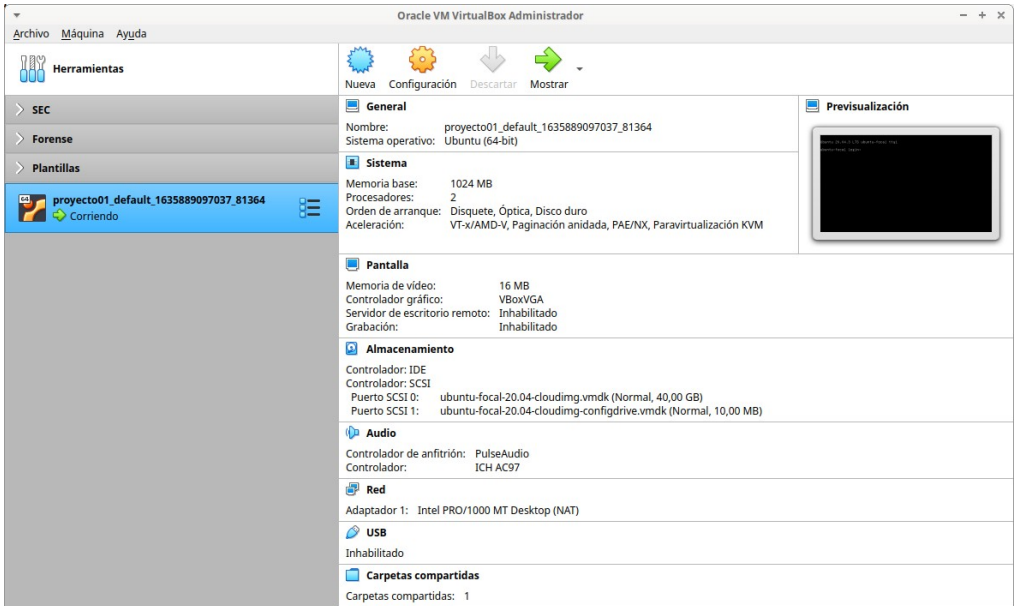
2 https://www.vagrantup.com/downloads.html

3 En https://app.vagrantup.com/boxes/search tenemos todo un catálogo de boxes a nuestra disposición

```
==> default: Setting the name of the VM: Vagrant_default_1635888730255_96549
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
==> default: Forwarding ports...
    default: 22 (guest) => 2222 (host) (adapter 1)
==> default: Running 'pre-boot' VM customizations...
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
    default: SSH address: 127.0.0.1:2222
    default: SSH username: vagrant
    default: SSH auth method: private key
    default:
    default: Vagrant insecure key detected. Vagrant will automatically replace
    default: this with a newly generated keypair for better security.
    default:
    default: Inserting generated public key within guest...
    default: Removing insecure key from the guest if it's present...
    default: Key inserted! Disconnecting and reconnecting using new SSH key...
==> default: Machine booted and ready!
==> default: Checking for guest additions in VM...
==> default: Mounting shared folders...
    default: /vagrant => /home/manuel/Vagrant/proyecto01
manuel@pcmanuel:~/Vagrant/proyecto01$
```

Revisando la salida vemos que:

- Al no tener una copia local del box ubuntu/focal64 se procede a su descarga de un repositorio de imágenes. Una vez descargado ya no habrá que descargarlo nuevamente del repositorio, por lo que se acelerará todo el proceso de creación de los proyectos que usen este box.
- Se crea una máquina virtual en Virtualbox de nombre `Vagrant_default_1635889097037_81364` con una interfaz de red en modo NAT, lo que le permite tener salida a Internet (pero no acceso desde otros equipos).
- Se configura una redirección de puertos (*forwarding ports*) mapeando el puerto 2222 del host con el 22 del guest.
- Se configura el acceso por ssh a la máquina virtual.
- Se crea una carpeta compartida entre el host y el guest (`/vagrant => /home/manuel/Vagrant/proyecto01`). En el guest la carpeta `/vagrant` se corresponde con la carpeta del proyecto, en mi caso la `/home/manuel/Vagrant/proyecto01` del host.



El resultado es un entorno de trabajo con una máquina Ubuntu Server 20.04 con una interfaz en modo NAT, una carpeta compartida con el host, accesible por ssh desde el puerto 2222 del host y corriendo en modo *headless* (en segundo plano).

Podemos ver el estado de las máquinas de un entorno con `vagrant status`

```
manuel@pcmanuel:~/Vagrant/proyecto01$ vagrant status
Current machine states:

default                   running (virtualbox)

The VM is running. To stop this VM, you can run `vagrant halt` to
shut it down forcefully, or you can run `vagrant suspend` to simply
suspend the virtual machine. In either case, to restart it again,
simply run `vagrant up`.
```

Para acceder y trabajar en la máquina virtual podemos hacer uso de la conexión ssh usando `vagrant ssh`

```
manuel@pcmanuel:~/Vagrant/proyecto01$ vagrant ssh
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-89-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Tue Nov  2 21:41:48 UTC 2021

  System load:  0.07              Processes:               119
  Usage of /:   3.3% of 38.71GB   Users logged in:         0
  Memory usage: 19%               IPv4 address for enp0s3: 10.0.2.15
  Swap usage:   0%

0 updates can be applied immediately.

vagrant@ubuntu-focal:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 02:c3:e3:75:4c:36 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic enp0s3
       valid_lft 86200sec preferred_lft 86200sec
    inet6 fe80::c3:e3ff:fe75:4c36/64 scope link
       valid_lft forever preferred_lft forever
vagrant@ubuntu-focal:~$ exit
logout
Connection to 127.0.0.1 closed.
manuel@x99:~/Vagrant/proyecto01$
```

Con `vagrant halt` se detienen las Mvs del entorno:

```
manuel@pcmanuel:~/Vagrant/proyecto01$ vagrant halt
==> default: Attempting graceful shutdown of VM...
manuel@pcmanuel:~/Vagrant/proyecto01$ vagrant status
Current machine states:

default                   poweroff (virtualbox)
```

```
The VM is powered off. To restart the VM, simply run `vagrant up`
```

Para arrancar nuevamente la máquina hacemos vagrant up

```
manuel@pcmanuel:~/Vagrant/proyecto01$ vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Checking if box 'ubuntu/focal64' version '20211026.0.0' is up to date...
==> default: Clearing any previously set forwarded ports...
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
==> default: Forwarding ports...
    default: 22 (guest) => 2222 (host) (adapter 1)
==> default: Running 'pre-boot' VM customizations...
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
    default: SSH address: 127.0.0.1:2222
    default: SSH username: vagrant
    default: SSH auth method: private key
==> default: Machine booted and ready!
==> default: Checking for guest additions in VM...
==> default: Mounting shared folders...
    default: /vagrant => /home/manuel/Vagrant/proyecto01
==> default: Machine already provisioned. Run `vagrant provision` or use the `--provision`
==> default: flag to force provisioning. Provisioners marked to run always will still run.
manuel@pcmanuel:~/Vagrant/proyecto01$
```

Para reiniciar el entorno usaremos `vagrant reload`. Un punto a tener en cuenta es que con `vagrant up` y `vagrant reload`, se vuelve a leer el Vagrantfile aplicándose los cambios de configuración en caso de haberlos (salvo el aprovisionamiento que sólo se hace la primera vez).

Si el día de mañana queremos destruir completamente el entorno usaremos `vagrant destroy`, borrándose las máquinas y toda la configuración asociada a ellas (carpetas compartidas, redirección de puertos, …):

```
manuel@pcmanuel:~/Vagrant/proyecto01$ vagrant destroy
    default: Are you sure you want to destroy the 'default' VM? [y/N] y
==> default: Forcing shutdown of VM...
==> default: Destroying VM and associated drives…
manuel@pcmanuel:~/Vagrant/proyecto01$
```

Por supuesto, al conservar el Vagrantfile podemos reconstruir un nuevo entorno igual al original en pocos minutos con `vagrant up`.

Escenario#2: Servidor LAMP

En este escenario montaremos con Vagrant un entorno LAMP (Linux Apache Mariadb/MySQL PHP) compartiendo una carpeta del host con el `DocumentRoot` del servidor web Apache. No sólo se creará la máquina virtual, sino que se automatizará la instalación y configuración del entorno (instalar Apache, última versión de PHP, Mariadb y habilitar acceso por http/https).

Empezamos creando una nueva carpeta para el proyecto y hacemos un `vagrant init --minimal` o `vagrant init` para comenzar a trabajar sobre el Vagrantfile:

```
manuel@pcmanuel:~/Vagrant$ mkdir WEB
manuel@pcmanuel:~/Vagrant$ cd WEB
manuel@pcmanuel:~/Vagrant/WEB$ vagrant init --minimal
manuel@pcmanuel:~/Vagrant/WEB$ nano Vagrantfile
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/focal64"
  config.vm.hostname = "web"
  config.vm.network "private_network", ip: "192.168.56.10"
```

```
  config.vm.network "forwarded_port", guest: 80, host: 8080
  config.vm.network "forwarded_port", guest: 443, host: 4430
  config.vm.provider "virtualbox" do |vb|
     vb.name = "web"
     vb.gui = true
     vb.memory = "1024"
     vb.cpus = 2
     vb.linked_clone = true
  end
  config.vm.post_up_message = "Para acceder ejecuta vagrant ssh"
end
```

Añadimos nuevas directivas:

- `config.vm.hostname = "web"` configura el nombre de la máquina virtual a web.
- `config.vm.network "private_network", ip: "192.168.56.10"` añade una nueva interfaz de red a la máquina virtual con la IP 192.168.56.10/24. Esta nueva interfaz de red está asociada a una red sólo anfitrión (host-only) de Virtualbox, por lo que el equipo guest tendrá conexión directa tanto con el host como con cualquier máquina virtual conectada a esta red sólo anfitrión.

  Es importante darse cuenta que la máquina virtual tendrá dos interfaces de red:

  - La primera en modo NAT que le permitirá acceder a Internet y ser accesible mediante `vagrant ssh`.
  - La segunda en modo red sólo anfitrión.

- `config.vm.network "forwarded_port", guest: 80, host: 8080` mapea el puerto 8080 del host con el 80 de la máquina virtual, permitiendo acceder desde cualquier equipo al servidor web de la máquina virtual por http a través de la IP del host puerto 8080.
- `config.vm.network "forwarded_port", guest: 443, host: 4430` mapea el puerto 4430 del host con el 443 de la máquina virtual, permitiendo acceder desde cualquier equipo al servidor web de la máquina virtual por https a través de la IP del host puerto 4430.
- `config.vm.provider "virtualbox" do |vb|` permite definir características extras de la máquina al usar Virtualbox como *provider*:

  - `vb.name = "web"` configura el nombre de la máquina virtual en la interfaz de Virtualbox.
  - `vb.gui = true` permite iniciar la máquina en modo normal (no *headless*); es decir, al arrancar la máquina se hará visible su ventana en Virtualbox.
  - `vb.memory = "1024"` permite indicar la memoria RAM asignada a la máquina virtual.
  - `vb.cpus = 2` permite indicar el número de CPUs asignadas a la máquina virtual.
  - `vb.linked_clone = true` permite usar la clonación enlazada para acelerar el proceso de creación de la máquina virtual. Por defecto al crearse una máquina se hace una clonación completa a partir de la imagen del sistema operativo descargado, que en el caso de boxes grandes ralentiza todo el proceso.

- `config.vm.post_up_message = "Para acceder ejecuta vagrant ssh"` permite mostrar un mensaje al terminar el `vagrant up`.

Ahora podríamos hacer un `vagrant up` y crearíamos la máquina virtual con las configuraciones anteriores pero sin tener el sistema LAMP operativo. Para automatizar la instalación del sistema LAMP y permitir la compartición del `DocumentRoot` entre el host y el guest haremos uso de un *provisioner* y de las *synced folders* por lo que modificamos el Vagrantfile:

```
manuel@pcmanuel:~/Vagrant/WEB$ nano Vagrantfile
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/focal64"
  config.vm.hostname = "web"
  config.vm.network "private_network", ip: "192.168.56.10"
  config.vm.network "forwarded_port", guest: 80, host: 8080
  config.vm.network "forwarded_port", guest: 443, host: 4430
```

```
  config.vm.provider "virtualbox" do |vb|
     vb.name = "web"
     vb.gui = true
     vb.memory = "1024"
     vb.cpus = 2
     vb.linked_clone = true
  end
  config.vm.synced_folder "www", "/var/www/html"
  config.vm.provision "shell", path: "lamp.sh"
  config.vm.post_up_message = "Para acceder ejecuta vagrant ssh"
end
```

Las nuevas opciones de configuración:

- `config.vm.synced_folder "www", "/var/www/html"` comparte la carpeta `www` del host con el guest (donde aparecerá montada en `/var/www/html`)
- `config.vm.provision "shell", path: "lamp.sh"` indica que en el primer inicio de la máquina se ejecutará el script `lamp.sh`. Como ya se indicó antes existen diferentes provisioners además de los shell scripts (Ansible, Puppet, …).

Uno de los pasos previos a lanzar el `vagrant up` será crear la carpeta local `www`

```
manuel@pc-profe:~/Vagrant/WEB$ mkdir www
```

El otro paso previo es preparar el script `lamp.sh` que se creará en la misma carpeta del Vagrantfile y automatizará todo el proceso de añadir repositorios, actualizar listado de paquetes, actualizar el equipo, instalar y configurar mariadb y apache, crear unas páginas index.php, info.php y descargar adminer.php.

```
manuel@pc-profe:~/Vagrant/WEB$ nano lamp.sh
#!/bin/bash
#echo "-----------------"
#echo "Añadiendo apt-cacher"
#echo "Acquire::http { Proxy \"http://10.0.5.7:8080\";};" > /etc/apt/apt.conf.d/01apt-cacher-ng

echo "-----------------"
echo "Actualizando repositorios y equipo"
apt update
apt full-upgrade -y

echo "-----------------"
echo "Instalando apache-php-mysql"
apt install net-tools apache2 php7.4 libapache2-mod-php7.4 php7.4-common php7.4-mbstring php7.4-xmlrpc
php7.4-soap php7.4-gd php7.4-xml php7.4-intl php7.4-mysql php7.4-cli php7.4-zip php7.4-curl mysql-
client mysql-server -y

echo "-----------------"
echo "Habilitando SSL"
a2enmod ssl
a2ensite default-ssl.conf

echo "-----------------"
echo "Creando index.php"
cat > /var/www/html/index.php << 'EOF'
<?php
echo "
<html>
<head>
    <title>Sitio web en ${_SERVER["SERVER_ADDR"]}</title>
</head>
```

```
<body>
<h1>Mi Sitio Web</h1>
Est&aacute;s en ${_SERVER["SERVER_ADDR"]}
</body>
</html>";
?>
EOF

echo "-----------------"
echo "Creando info.php"
cat > /var/www/html/info.php << 'EOF'
<?php
phpinfo();
?>
EOF

echo "-----------------"
echo "Preparando adminer.php"
wget -O /var/www/html/adminer.php https://www.adminer.org/latest.php

echo "-----------------"
echo "Reiniciando apache"
systemctl restart apache2.service

echo "------ FIN ------"
```

Una vez preparada la carpeta compartida y el script lanzamos el proceso y tras unos minutos tendremos un entorno LAMP igual, fácil y rápidamente reproducible para todos los miembros de nuestro equipo de trabajo:

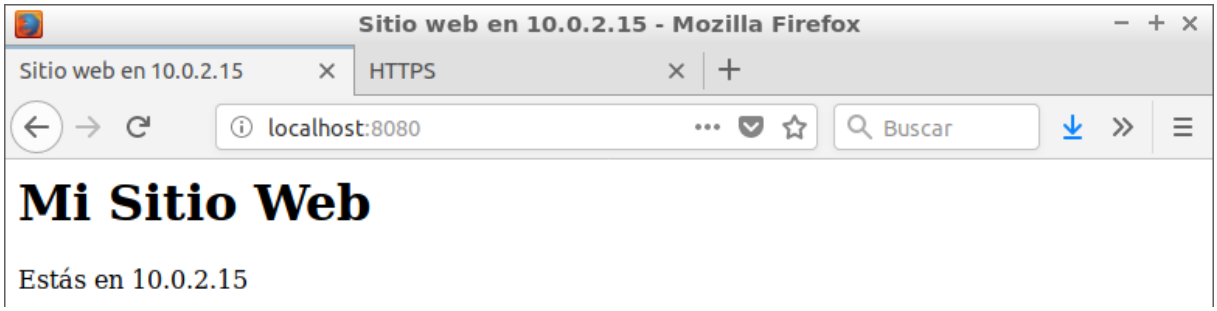
```
manuel@pcmanuel:~/Vagrant/WEB$ vagrant up
manuel@pcmanuel:~/Vagrant/WEB$ vagrant ssh
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-89-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Tue Nov  2 22:15:15 UTC 2021

  System load:  0.36              Processes:               127
  Usage of /:   5.1% of 38.71GB   Users logged in:         0
  Memory usage: 57%               IPv4 address for enp0s3: 10.0.2.15
  Swap usage:   0%                IPv4 address for enp0s8: 192.168.56.10


0 updates can be applied immediately.
vagrant@web:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 02:c3:e3:75:4c:36 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic enp0s3
       valid_lft 86152sec preferred_lft 86152sec
```

```
    inet6 fe80::c3:e3ff:fe75:4c36/64 scope link
       valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:5e:f2:36 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.10/24 brd 192.168.56.255 scope global enp0s8
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe5e:f236/64 scope link
       valid_lft forever preferred_lft forever

vagrant@web:~$ cat /proc/cpuinfo | grep processor | wc -l
2
vagrant@web:~$ free -h
              total        used        free      shared  buff/cache   available
Mem:           992M         44M        733M        3.1M        214M        797M
Swap:            0B          0B          0B
vagrant@web:~$ sudo netstat -putan
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:33060         0.0.0.0:*               LISTEN      14057/mysqld
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN      14057/mysqld
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      590/systemd-resolve
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      822/sshd: /usr/sbin
tcp        0      0 10.0.2.15:22            10.0.2.2:55724          ESTABLISHED 16878/sshd: vagrant
tcp6       0      0 :::80                   :::*                    LISTEN      16785/apache2
tcp6       0      0 :::22                   :::*                    LISTEN      822/sshd: /usr/sbin
tcp6       0      0 :::443                  :::*                    LISTEN      16785/apache2
udp        0      0 127.0.0.53:53           0.0.0.0:*                           590/systemd-resolve
udp        0      0 10.0.2.15:68            0.0.0.0:*                           1880/systemd-networ
vagrant@web:~$ curl 192.168.56.10

<html>
<head>
    <title>Sitio web en 192.168.56.10</title>
</head>
<body>
<h1>Mi Sitio Web</h1>

Est&aacute;s en 192.168.56.10
</body>
</html>
vagrant@web:~$ sudo mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 8.0.27-0ubuntu0.20.04.1 (Ubuntu)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
```

```
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.02 sec)

mysql>  exit
vagrant@web:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            474M     0  474M   0% /dev
tmpfs            99M  948K   98M   1% /run
/dev/sda1        39G  1.6G   38G   4% /
tmpfs           491M     0  491M   0% /dev/shm
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           491M     0  491M   0% /sys/fs/cgroup
/dev/loop0       62M   62M     0 100% /snap/core20/1169
/dev/loop1       68M   68M     0 100% /snap/lxd/21835
/dev/loop2       33M   33M     0 100% /snap/snapd/13640
vagrant         240G  126G  114G  53% /vagrant
var_www_html    240G  126G  114G  53% /var/www/html
tmpfs            99M     0   99M   0% /run/user/1000
vagrant@web:~$ ls -lhF /var/www/html
total 4.0K
total 476K
-rw-r--r-- 1 vagrant vagrant 466K May 14 05:40 adminer.php
-rw-r--r-- 1 vagrant vagrant  185 Nov  2 22:13 index.php
-rw-r--r-- 1 vagrant vagrant   20 Nov  2 22:13 info.php
vagrant@web:~$
```

Con los comandos anteriores podemos comprobar que se está ejecutando un servidor LAMP en una máquina virtual con 1GB de RAM, 2 cpus, 2 interfaces de red y con el `DocumentRoot` del servidor web accesible vía la carpeta local `www`. Si modificamos el fichero `index.php` (o añadimos nuevas páginas) en la carpeta `www` del host (o en `/vagrant_www` del guest), éstas podrán ser accesibles vía el servidor web Apache.

El servidor web es accesible directamente por la red sólo anfitrión y a través de la IP del host por los puertos configurados:



Si no queremos que la carpeta del proyecto (donde está el Vagrantfile) se monte automáticamente en `/vagrant` podemos añdir en el Vagrantfile:

```
config.vm.synced_folder ".", "/vagrant", disabled: true
```

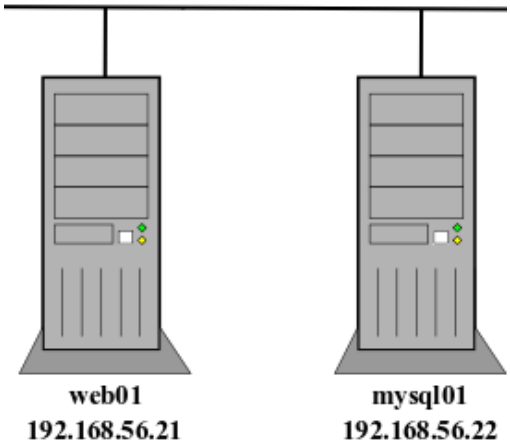
A veces pueden aparecer problemas de permisos en las carpetas compartidas; ya que por defecto, Vagrant le asigna como propietario/grupo a `vagrant/vagrant`. Añadiendo `owner` y `group` en la definición de la carpeta compartida podremos asignar el propietario/grupo adecuado; por ejemplo:

```
config.vm.synced_folder "www", "/vagrant_www", owner: "root", group: "root"
```

Para terminar, indicar que el aprovisionamiento ocurre únicamente en ciertos momentos de la vida de un entorno:

- En el primer `vagrant up` cuando se crea en entorno se ejecuta el aprovisionamiento. Una vez creado el entorno, `vagrant up` únicamente se limita a iniciar las máquinas, salvo que se añada la opción `--provision`.
- Si se hace un `vagrant provision` en un entorno en ejecución.
- Cuando se reinicia un entorno con `vagrant reload --provision`.

Escenario#3: Cluster Web – Base de datos



En esta ocasión vamos a ver como Vagrant puede usarse para levantar rápidamente entornos complejos para pruebas o prácticas en clase. Crearemos un entorno con dos máquinas virtuales, una que actuará como servidor web y otra como servidor de base de datos. Aparecen varios problemas que habrá que solucionar:

- En el Vagrantfile habrá que definir las dos máquinas virtuales.
- Tanto el servidor web como el de base de datos serán directamente accesibles desde el equipo anfitrión; y además, el servicio web http/https será accesible desde cualquier lugar a través de los puertos 8080/tcp y 4430/tcp del anfitrión.

Creamos una carpeta para el nuevo entorno y editamos el Vagrantfile:

```
manuel@pc-profe:~/Vagrant$ mkdir web_db
manuel@pc-profe:~/Vagrant$ cd web_db/
manuel@pc-profe:~/Vagrant/web_db$ vagrant init --minimal
manuel@pc-profe:~/Vagrant/web_db$ nano Vagrantfile
Vagrant.configure("2") do |config|
   config.vm.box = "ubuntu/focal64"
# WEB
  config.vm.define "web01" do |web01|
    # web01.vm.box = "bento/ubuntu-20.04"
    web01.vm.hostname = "web01"
    web01.vm.network "private_network", ip: "192.168.56.21", netmask: "255.255.255.0"
    web01.vm.network "forwarded_port", guest: 80, host: 8080
    web01.vm.network "forwarded_port", guest: 443, host: 4430
    web01.vm.synced_folder "www", "/var/www/html"
    web01.vm.provision "shell", path: "apache.sh"
    web01.vm.provider "virtualbox" do |vb|
      vb.name = "web01"
      vb.memory = "1024"
      vb.cpus = 1
      vb.linked_clone = true
      vb.customize ["modifyvm", :id, "--groups", "/Cluster"]
    end
  end
# DB
  config.vm.define "mysql01" do |mysql01|
    # mysql01.vm.box = "bento/ubuntu-20.04"
    mysql01.vm.hostname = "mysql01"
    mysql01.vm.network "private_network", ip: "192.168.56.22", netmask: "255.255.255.0"
    mysql01.vm.provision "shell", path: "mysql.sh"
    mysql01.vm.provider "virtualbox" do |vb|
      vb.name = "mysql01"
      vb.memory = "1024"
```

```
      vb.cpus = 2
      vb.linked_clone = true
      vb.customize ["modifyvm", :id, "--groups", "/Cluster"]
    end
  end
end
```

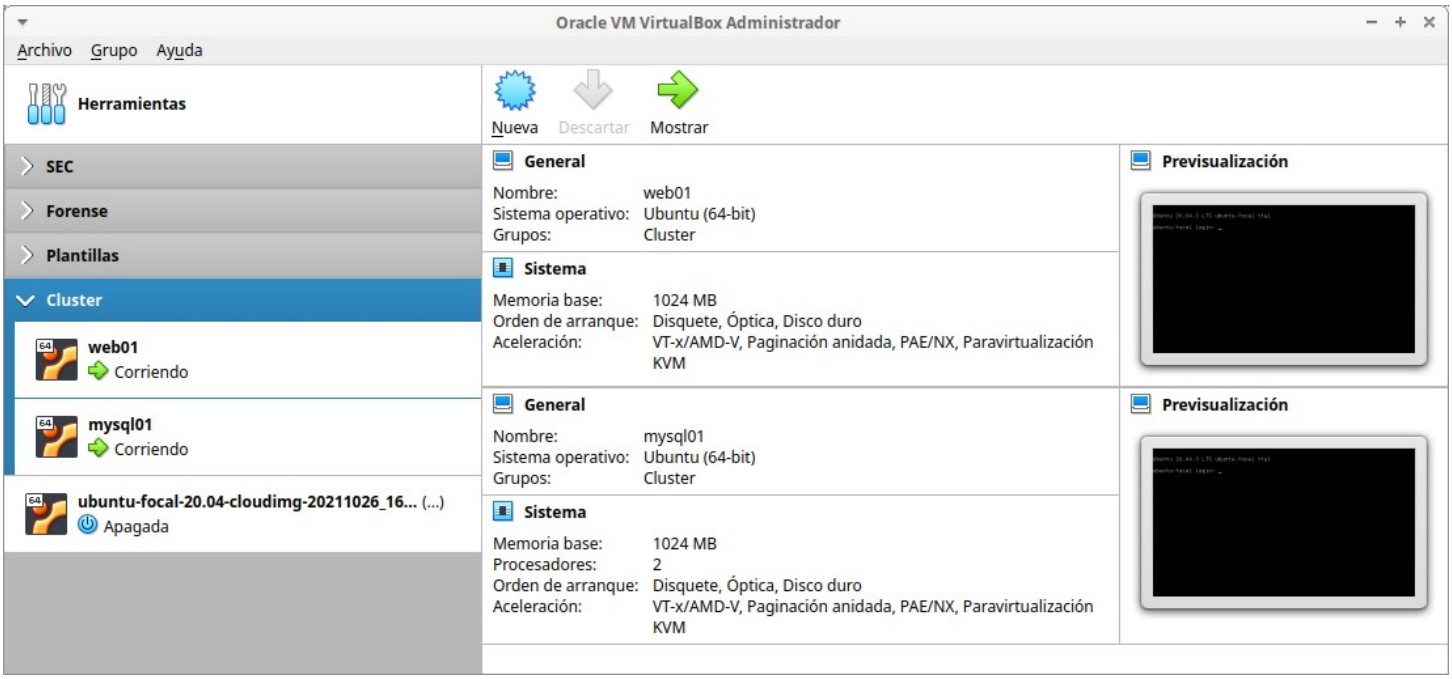
Analizando el Vagrantfile vemos que:

- Hay dos secciones donde se definen las características particulares de las máquinas del entorno:
  - `config.vm.define "web01" do |web01|` usada para definir el servidor web.
  - `config.vm.define "mysql01" do |mysql01|` usada para definir el servidor de base de datos.
- La directiva `config.vm.box = "ubuntu/focal64"` está fuera de las secciones de web01 y mysql01 por lo que se aplica a todas las máquinas. En este caso ambos servidores correrán un Ubuntu 20.04 creado a partir del box `ubuntu/focal64`. Es posible usar box diferentes indicándolo en la sección correspondiente (`web01.vm.box` y `mysql01.vm.box`).
- Los dos servidores, además de la interfaz en modo NAT que por defecto crea Vagrant, están conectados a la red sólo anfitrión 192.168.56.0/24.
- Mediante las directivas `web01.vm.network "forwarded_port"` se hace accesible el servicio http/https a través de los puertos 8080/tcp y 4430/tcp del anfitrión.
- `vb.customize ["modifyvm", :id, "--groups", "/Cluster"]` permite agrupar las máquinas virtuales de este entorno en un grupo llamado `Cluster`, visible en la interfaz de Virtualbox.
- Los scripts `apache.sh` y `mysql.sh` se encargan de la instalación del servidor web Apache y de Mariadb.

Una vez terminado el `vagrant up` se puede comprobar como se han creado las dos máquinas y se puede acceder desde el servidor web al servidor de base de datos:

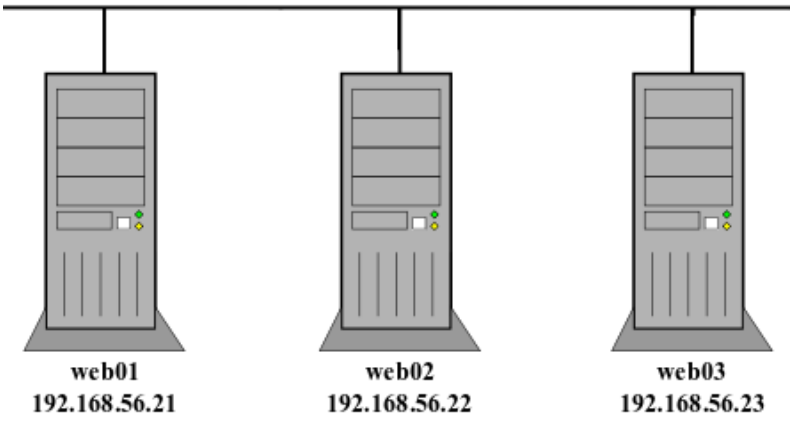
```
manuel@pc-profe:~/Vagrant/web_db$ vagrant up
manuel@pc-profe:~/Vagrant/web_db$ vagrant ssh web01
vagrant@web01:~$ mysql -u wordpress -p -h 192.168.56.22
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.27-0ubuntu0.20.04.1 (Ubuntu)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| wordpress          |
+--------------------+
2 rows in set (0.01 sec)

mysql> exit
```

Como se ve, para acceder a las máquinas del escenario se ejecuta el comando `vagrant ssh <nombre_maquina>`.



Escenario#4: Cluster de servidores web

Aprovechándonos del hecho de que el Vagrantfile está basado en Ruby, en un entorno donde haya varias máquinas virtuales de configuraciones similares es posible crear un bucle para definirlas. Suponiendo que nuestro entorno se corresponde con la figura, una posible solución sería:




```
manuel@pc-profe:~/Vagrant$ mkdir cluster
manuel@pc-profe:~/Vagrant$ cd cluster/
manuel@pc-profe:~/Vagrant/cluster$ vagrant init --minimal
manuel@pc-profe:~/Vagrant/cluster$ nano Vagrantfile
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/focal64"

# WEB
  (1..3).each do |i|
    config.vm.define "web0#{i}" do |node|
        node.vm.hostname = "web0#{i}"
        node.vm.network "private_network", ip: "192.168.56.2#{i}", netmask: "255.255.255.0"
        node.vm.synced_folder "web0#{i}", "/var/www/html"
        node.vm.provision "shell", path: "apache.sh"
        node.vm.provider "virtualbox" do |vb|
            vb.name = "web0#{i}"
            # vb.gui = true
```

```
            vb.memory = "1024"
            vb.cpus = 1
            vb.linked_clone = true
            vb.customize ["modifyvm", :id, "--groups", "/Cluster"]
        end
    end
  end
end
```

- `(1..3).each do |i|` define un bucle con tres iteraciones.
- `#{i}` permite ir personalizando la configuración de las distintas máquinas del bucle:
  - Nombre de las máquinas: web01, web02 y web03.
  - Direcciones IP: 192.168.56.21, 192.168.56.22 y 192.168.56.23.
  - Carpetas compartidas: web01, web02 y web03.

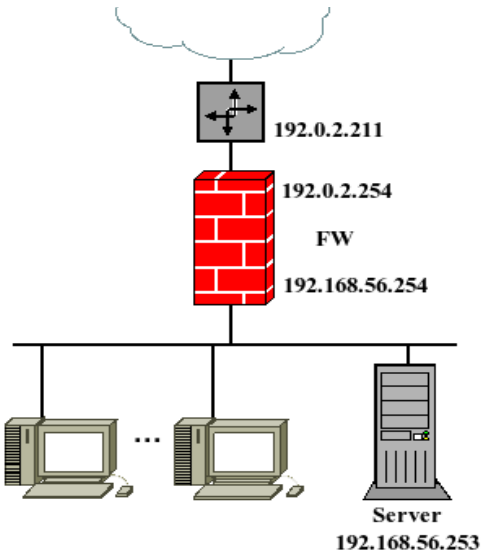
Creamos las carpetas compartidas y lanzamos el vagrant up:

```
manuel@pc-profe:~/Vagrant/cluster$  mkdir web0{1..3}
manuel@pc-profe:~/Vagrant/cluster$ ls -1
Vagrantfile
web01
web02
web03
manuel@pc-profe:~/Vagrant/cluster$ vagrant up
```

Escenario#5: Entorno con varias máquinas y redes

En esta ocasión vamos a ver como Vagrant puede levantar rápidamente entornos complejos con varias máquinas conectadas a redes diferentes. Crearemos un entorno con dos máquinas virtuales, una que actuará como router-firewall y otra como un servidor web interno. Aparecen varios problemas que habrá que solucionar:

- En el Vagrantfile habrá que definir las dos máquinas virtuales.
- El router-firewall FW sale a Internet a través de un router 192.0.2.211. Aquí el problema está en que Vagrant siempre crea una interfaz NAT que permite al guest salir al exterior a través del host y que no se puede eliminar ya que es necesaria para el funcionamiento de Vagrant (p.e. para hacer `vagrant ssh`).
- El servidor interno queremos que salga a Internet a través del router-firewall FW y no a través del equipo anfitrión. Igual que antes, tenemos el problema de la interfaz NAT.
- Los servicios corriendo en el servidor interno serán accesibles desde el exterior a través del router-firewall FW; por tanto, en FW habrá que configurar unas reglas iptables de tipo DNAT que hagan que las peticiones http/https destinadas a su interfaz externa se redirijan al servidor interno. Una vez más, no nos interesa la interfaz NAT creada por defecto ni para que FW tenga acceso a Internet (usará su propio router 192.0.2.211) ni para abrir puertos (se hará directamente en FW y no en el anfitrión con `config.vm.network "forwarded_port"`).



Creamos una carpeta para el nuevo entorno y editamos el Vagrantfile:

```
manuel@pc-profe:~/Vagrant$ mkdir fw-server
manuel@pc-profe:~/Vagrant$ cd fw-server/
manuel@pc-profe:~/Vagrant/fw-server$ vagrant init --minimal
manuel@pc-profe:~/Vagrant/fw-server$ nano Vagrantfile
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/focal64"
# Definición del router-firewall FW
  config.vm.define "fw" do |fw|
    fw.vm.hostname = "FW"
    fw.vm.network "public_network", bridge: "enp2s0", ip: "192.0.2.254", netmask: "255.255.255.0"
    fw.vm.network "private_network", ip: "192.168.56.254", netmask: "255.255.255.0"
```

```
    # eliminar default gw - red NAT creada por defecto
    fw.vm.provision "shell",
        run: "always",
        inline:  "ip route del default"
    # default router
    fw.vm.provision "shell",
        run: "always",
        inline: "ip route add default via 192.0.2.211"
    fw.vm.provider "virtualbox" do |vb|
        vb.name = "FW"
        # vb.gui = true
        vb.memory = "1024"
        vb.cpus = 1
        vb.linked_clone = true
        vb.customize ["modifyvm", :id, "--groups", "/FW-Server"]
    end
  end

# Definición del servidor interno server
  config.vm.define "server" do |server|
    server.vm.hostname = "server"
    server.vm.network "private_network", ip: "192.168.56.253", netmask: "255.255.255.0"
    server.vm.synced_folder "www", "/var/www/html"
    server.vm.provision "shell", path: "lamp.sh"
    # eliminar default gw - red NAT creada por defecto
    server.vm.provision "shell",
        run: "always",
        inline:  "ip route del default"
    # default router
    server.vm.provision "shell",
        run: "always",
        inline: "ip route add default via 192.168.56.254"
    server.vm.provider "virtualbox" do |vb|
        vb.name = "server"
        # vb.gui = true
        vb.memory = "1024"
        vb.cpus = 2
        vb.linked_clone = true
        vb.customize ["modifyvm", :id, "--groups", "/FW-Server"]
    end
  end
end
```

Analizando el Vagrantfile vemos que:

- Hay dos secciones donde se definen las características particulares de las máquinas del entorno:
  - `config.vm.define "fw" do |fw|` usada para definir el router-firewall FW.
  - `config.vm.define "server" do |server|` usada para definir el servidor interno.
- La directiva `config.vm.box = "ubuntu/focal64"` está fuera de las secciones de fw y server por lo que se aplica a todas las máquinas. En este caso tanto el router-firewall FW como el servidor interno correrán un Ubuntu 20.04 creado a partir del box `ubuntu/focal64`. Es posible usar box diferentes indicándolo en la sección correspondiente (`fw.vm.box` y `server.vm.box`).
- En el router-firewall vamos a tener 3 interfaces de red:
  - Una de tipo NAT creada automáticamente por Vagrant y que por el box usado se llamará eth0.

- Una de tipo puente (bridge) con IP 192.0.2.254/24 que se llamará eth1 y estará asociada a la interfaz enp2s0 del equipo anfitrión: `fw.vm.network "public_network", bridge: "enp2s0", ip: "192.0.2.254", netmask: "255.255.255.0"`. `enp2s0` es el nombre asignado a la interfaz real del host anfitrión[4].
- Una de tipo sólo anfitrión con IP 192.168.56.254/24 que se llamará eth2: `fw.vm.network "private_network", ip: "192.168.56.254", netmask: "255.255.255.0"`

- En el router-firewall para configurar la pasarela predeterminada correcta se ejecutarán en cada inicio (`run: "always"`) dos comandos shell (`server.vm.provision "shell"`):
  - Uno que borra la pasarela predeterminada creada por defecto por Vagrant y asociada a la red NAT: `inline: "ip route del default"`
  - Uno que añade la pasarela predeterminada correcta: `inline: "ip route add default via 192.0.2.211"`

  En la documentación oficial de Vagrant proponen la siguiente solución, algo más compleja y dependiente del nombre asignado a la interfaz por el sistema operativo guest:

```
# default router
fw.vm.provision "shell",
  run: "always",
  inline: "route add default gw 192.0.2.211"
# eliminar default gw en eth0 - red NAT creada por defecto
fw.vm.provision "shell",
  run: "always",
  inline:  "eval `route -n | awk '{ if ($8 == \"eth0\" && $2 != \"0.0.0.0\") print \"route del default
gw \" $2; }'`"
```

  - Uno que añade la pasarela predeterminada correcta: `inline: "route add default gw 192.0.2.211"`
  - Uno que borra la pasarela predeterminada creda por defecto por Vagrant y asociada a la red NAT: inline: `"eval `route -n | awk '{ if ($8 == \"eth0\" && $2 != \"0.0.0.0\") print \"route del default gw \" $2; }'`"`

- En el servidor interno tendremos 2 interfaces de red:
  - Una de tipo NAT creada automáticamente por Vagrant y que por el box usado se llamará eth0.
  - Una de tipo sólo anfitrión con IP 192.168.56.253/24 que se llamará eth1: `fw.vm.network "private_network", ip: "192.168.56.253", netmask: "255.255.255.0"`

- En el servidor interno se repite lo explicado anteriormente para configurar la IP interna del router-firewall 192.168.56.254 como pasarela predeterminada.
- `vb.customize ["modifyvm", :id, "--groups", "/FW-Server"]` permite agrupar las máquinas virtuales de este entorno en un grupo llamado FW-Server , visible en la interfaz de Virtualbox.
- A modo de ejemplo, en la máquina FW no se hace ningún aprovisionamiento (salvo el necesario para configurar la pasarela predeterminada) y en el servidor interno el script de aprovisionamiento `lamp.sh` instala apache con soporte para php, crea un archivo de ejemplo `index.php` en el `DocumentRoot` y MySQL. Fijarse en la ubicación de `server.vm.provision "shell", path: "lamp.sh"` dentro del Vagrantfile para que se ejecute antes de cambiar el default gw (así se tendrá acceso a los repositorios aunque la máquina router-firewall no esté configurada).

Si iniciamos el entorno veremos como se crean las dos máquinas con la configuración indicada y agrupadas en la interface de Virtualbox dentro del grupo FW-server:

```
manuel@pc-profe:~/Vagrant/fw-server$ vagrant up
Bringing machine 'fw' up with 'virtualbox' provider...
Bringing machine 'server' up with 'virtualbox' provider...
==> fw: Cloning VM...
==> fw: Matching MAC address for NAT networking...
==> fw: Checking if box 'ubuntu/focal64' is up to date...
```

4 `VBoxManage list bridgedifs` lista el nombre de las interfaces del anfitrión para configurar adaptadores en modo puente

```
==> fw: Setting the name of the VM: FW
==> fw: Clearing any previously set network interfaces...
==> fw: Preparing network interfaces based on configuration...
    fw: Adapter 1: nat
    fw: Adapter 2: bridged
    fw: Adapter 3: hostonly
==> fw: Forwarding ports...
    fw: 22 (guest) => 2222 (host) (adapter 1)
==> fw: Running 'pre-boot' VM customizations...
==> fw: Booting VM...
==> fw: Waiting for machine to boot. This may take a few minutes...
    fw: SSH address: 127.0.0.1:2222
    fw: SSH username: vagrant
    fw: SSH auth method: private key
    fw:
    fw: Vagrant insecure key detected. Vagrant will automatically replace
    fw: this with a newly generated keypair for better security.
    fw:
    fw: Inserting generated public key within guest...
    fw: Removing insecure key from the guest if it's present...
    fw: Key inserted! Disconnecting and reconnecting using new SSH key...
==> fw: Machine booted and ready!
==> fw: Checking for guest additions in VM...
==> fw: Setting hostname...
==> fw: Configuring and enabling network interfaces...
==> fw: Mounting shared folders...
    fw: /vagrant => /home/manuel/Vagrant/fw-server
==> fw: Running provisioner: shell...
    fw: Running: inline script
==> fw: Running provisioner: shell...
    fw: Running: inline script
==> server: Cloning VM...
==> server: Matching MAC address for NAT networking...
==> server: Checking if box 'ubuntu/focal64' is up to date...
==> server: Setting the name of the VM: server
==> server: Fixed port collision for 22 => 2222. Now on port 2200.
==> server: Clearing any previously set network interfaces...
==> server: Preparing network interfaces based on configuration...
    server: Adapter 1: nat
    server: Adapter 2: hostonly
==> server: Forwarding ports...
    server: 22 (guest) => 2200 (host) (adapter 1)
==> server: Running 'pre-boot' VM customizations...
==> server: Booting VM...
==> server: Waiting for machine to boot. This may take a few minutes...
    server: SSH address: 127.0.0.1:2200
    server: SSH username: vagrant
    server: SSH auth method: private key
    server: Warning: Connection reset. Retrying...
    server:
    server: Vagrant insecure key detected. Vagrant will automatically replace
    server: this with a newly generated keypair for better security.
    server:
    server: Inserting generated public key within guest...
    server: Removing insecure key from the guest if it's present...
```

```
    server: Key inserted! Disconnecting and reconnecting using new SSH key...
==> server: Machine booted and ready!
==> server: Checking for guest additions in VM...
==> server: Setting hostname...
==> server: Configuring and enabling network interfaces...
==> server: Mounting shared folders...
    server: /vagrant => /home/manuel/Vagrant/fw-server
    server: /var/www/html => /home/manuel/Vagrant/fw-server/www
==> server: Running provisioner: shell...
    server: Running: /tmp/vagrant-shell20180522-8452-1xvasti.sh
    server: -----------------
    server: Actualizando repositorios y equipo
    server: -----------------
    server: Instalando apache-php-mysql
    server: -----------------
    server: Habilitando SSL
    server: -----------------
    server: Creando index.php
    server: -----------------
    server: Creando info.php
    server: -----------------
    server: Preparando adminer.php
    server: -----------------
    server: Reiniciando apache
    server: ------ FIN ------
==> server: Running provisioner: shell...
    server: Running: inline script
==> server: Running provisioner: shell...
    server: Running: inline script
manuel@pc-profe:~/Vagrant/fw-server$
```

Para acceder a las máquinas podemos especificar en `vagrant ssh` la que nos interesa:

```
manuel@pc-profe:~/Vagrant/fw-server$ vagrant ssh fw
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.4.0-126-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Mon Sep 26 11:48:06 UTC 2022

  System load:  0.93              Users logged in:         0
  Usage of /:   3.5% of 38.70GB   IPv4 address for enp0s3: 10.0.2.15
  Memory usage: 19%               IPv4 address for enp0s8: 192.0.2.254
  Swap usage:   0%                IPv4 address for enp0s9: 192.168.56.254
  Processes:    109


0 updates can be applied immediately.

New release '22.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

vagrant@FW:~$
```

Para server sería:

```
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.4.0-126-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Mon Sep 26 11:48:33 UTC 2022

  System load:  1.48              Processes:               129
  Usage of /:   5.3% of 38.70GB   Users logged in:         0
  Memory usage: 57%               IPv4 address for enp0s3: 10.0.2.15
  Swap usage:   0%                IPv4 address for enp0s8: 192.168.56.253


0 updates can be applied immediately.

New release '22.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


vagrant@server:~$
```

De la misma forma se podrían enviar órdenes a una máquina concreta del entorno: `vagrant {up|halt|reload|ssh} [name]`