¿Qué es Vagrant?

Vagrant es una herramienta de código abierto, para creación de ambientes virtuales de desarrollo, disponible para *Windows*, *MacOS X* y *GNU/Linux*, que permite generar entornos de desarrollo reproducibles y compartibles de forma muy sencilla. Para ello, *Vagrant* crea y configura máquinas virtuales a partir de simples ficheros de configuración.

Basta con compartir el fichero de configuración de *Vagrant* (llamado "*Vagrantfile"*) con otro desarrollador para que, con un simple comando, pueda reproducir el mismo entorno de desarrollo. Esto es especialmente útil en equipos formados por varias personas, ya que asegura que todos los desarrolladores tienen el mismo entorno, con las mismas dependencias y configuración. Con *Vagrant*, compartir pesadas máquinas virtuales o el ya mítico "en mi ordenador funciona" (causado generalmente por diferentes configuraciones o versiones de software) son cosas del pasado.

Además, dado que la configuración de la máquina virtual es un simple fichero de texto plano, podemos incluir este fichero en nuestro repositorio en el control de versiones, junto con el resto del código del proyecto. De esta manera, un nuevo desarrollador que se incorpore al equipo simplemente tendrá que clonar el repositorio del proyecto y ejecutar *Vagrant* para tener el entorno de desarrollo montado y funcionando en cuestión de minutos.

Por defecto, *Vagrant* utiliza *VirtualBox* como motor de máquinas virtuales, aunque existe la opción de utilizar *VMWare Workstation* (*Windows*) o *VMWare Fusion* (*MacOS X*) con un plugin de pago.

Características fundamentales para entender Vagrant

1. Se instala en los equipos de los desarrolladores. Está pensado para entornos de desarrollo, ni siquiera la compañía que desarrolla Vagrant, lo recomienda en producción.
2. Es multiplataforma: Mac, Windows, CentOS y Debian.
3. Está pensado para montar entornos de desarrollo portables y reproducibles entre desarrolladores.
4. Por defecto, utiliza Virtual Box para virtualizar, pero funciona con cualquier software de virtualización.
5. Utiliza archivos de configuración Vagrantfile, con una sintaxis sencilla.
6. Los archivos de configuración Vagrantfile generan una *box* (máquina virtual) que puede ser compartida a través de repositorios .
7. Hay repositorios públicos de Vagrant boxes, por ejemplo este o este.

¿Para que NO sirve Vagrant?

A veces es más fácil entender una herramienta sabiendo para qué no lo puedes usar. En el caso de Vagrant:

1. No está pensado para gestionar la infraestructura de servidores (para esto está chef o puppet, ansible)
2. No está pensado para ejecutarse en un servidor, sino en el propio equipo del desarrollador
3. Y sobre todo, no está pensado para producción

¿Por qué usar Vagrant?

Llegados a este punto alguien se puede estar preguntando:

¿Por qué usar Vagrant y no directamente crear máquinas virtuales con VirtualBox? (u otro software)

Leer sobre sobre las **ventajas del paradigma DevOps** en general.

Motivos:

- Vagrant es una herramienta de consola, con lo cual permite **automatizar** procesos en lugar de crear máquinas virtuales y entornos de desarrollo manualmente
- El **lenguaje de scripting** de Vagrant permite compartir las configuraciones de cada proyecto, sin almacenar las máquinas virtuales completas, ahorrando espacio en disco.
- Se pueden almacenar las configuraciones en repositorios de código y que se mejoren de manera **colaborativa**
- **Evita dependencias** y cuellos de botella con departamentos de IT.
- Las máquinas virtuales se ejecutan en el equipo del desarrollador, de manera que tú (como desarrollador) **controlas todos los recursos**
- Se **simplifica** la infraestructura interna de la compañía, consumiendo menos recursos hardware (servidores)
- Se **"*estandariza*" y agiliza** la manera de crear entornos de desarrollo.

Uso básico de vagrant:

vagrant init, **vagrant up**, **vagrant status**, **vagrant halt** y **vagrant destroy.**

Creamos un directorio para cada proyecto, accedemos al proyecto y creamos un fichero de configuración básico de vagrant con:

```
vagrant init -m ubuntu/trusty64
```

Levantamos la máquina con, que copiará la imagen como una nueva máquina y la arrancará:

```
vagrant up
```

Accedemos a la máquina con:

```
vagrant ssh
```

Comprobamos en todo momento el estado de la máquina con:

```
vagrant status
```

Podemos parar la máquina con:

```
vagrant halt
```

Si de nuevo levantamos la máquina, se reutilizará la máquina anteriormente creada y los cambios que se hubieran efectuado en ella. Cuando no queramos volver a utilizar esa máquina la eliminamos permanentemente con:

```
 vagrant destroy
vagrant box ls
vagrant box remove centos/7 --box-version 1801.02
```

Laboratorio, trabajando con nuestra primera mv en vagrant:

Añadir una primera imagen (Ubuntu Trusty de 64 bits).

```
vagrant box add ubuntu/trusty64
```

Lanzando primera máquina

Uso básico de vagrant: vagrant init, vagrant up, vagrant status, vagrant halt y vagrant destroy.

Para definir un escenario utilizaremos un fichero llamado Vagrantfile, por cada escenario lo que hacemos es cambiarlo de directorio, cuando los escenarios están definidos en diferentes directorios no tendremos ningún problema de interferencias entre la mv

Creamos un directorio para cada proyecto, accedemos al proyecto y creamos un fichero de configuración básico de vagrant con:

```
C:\HashiCorp\Vagrant\prueba1

    vagrant init -m ubuntu/trusty64

    vagrant init -m bento/centos-7.1
```

Para listar los box que tenemos descargados:

$ vagrant box list

Levantamos la máquina con, que copiará la imagen como una nueva máquina y la arrancará:

```
    vagrant up
```

Este escenario consiste en una maquina de Ubuntu, conectada a una red interna que mediante un mecanismo de nat se conecta al exterior y nosotros desde la maquina anfitriona mediante comandos de vagrant podemos interectuar con ella, de esta forma nos ahorramos en crear una nueva mv, instalarle el sistema operativo, tener que configuraralo, etc….

Lo que esta haciendo ahora vagrant en coger la imagen, importarla, y realiza una serie de adaptaciones para que nosotros podamos utilizar y acceder a ellas, conecta la mv a una red interna que tiene acceso a través de nat, abrir una conexión en el host al puerto 2222, esto dependerá si esta en uso o no, y realizar una redirección de ese puerto al puerto 22 de la mv, para que nosotros a través del puerto 2222, podamos conectarnos a nuestra maquina virtual, una vez que hace eso, realiza la configuración de ssh y una serie de comprobaciones y monta una carpeta compartida que veremos mas adelante

Accedemos a la máquina con:

```
vagrant ssh
```

El usuario es vagrant y el password es vagrant, para comprobar que tenemos salida hacia fuera:

```
sudo apt-get update
```

Comprobamos en todo momento el estado de la máquina con:

```
vagrant status
```

Podemos parar la máquina con:

```
vagrant halt
```

Si de nuevo levantamos la máquina, se reutilizará la máquina anteriormente creada y los cambios que se hubieran efectuado en ella. Cuando no queramos volver a utilizar esa máquina la eliminamos permanentemente con:

```
vagrant destroy
```

Si ahora levantamos la mv denuevo con el comando vagrat up, veremos como levanta ya de forma mas rápida ya que la maquina virtual tiene el disco duro y ya esta creada.

Ejercicio: Lanzando una máquina ya configurada

Uso de una máquina ya configurada, por ejemplo **rasmus/php7dev** que contiene los paquetes necesarios para ejecutar una aplicación web con PHP7, en este caso nos descargaremos el box y el escenario.

Instalamos la imagen:

```
vagrant box add rasmus/php7dev
```

If you have vagrant version < 1.5, you may run into "command was not invoked properly" error with vagrant box add rasmus/php7dev, then you can run it with the following explicit url:

vagrant box add "rasmus/php7dev"

https://vagrantcloud.com/rasmus/boxes/php7dev/versions/0.0.7/providers/virtualbox.box

Clonamos el repositorio que contiene el Vagrantfile configurado con las características de esta imagen:

```
git clone https://github.com/rlerdorf/php7dev.git
```

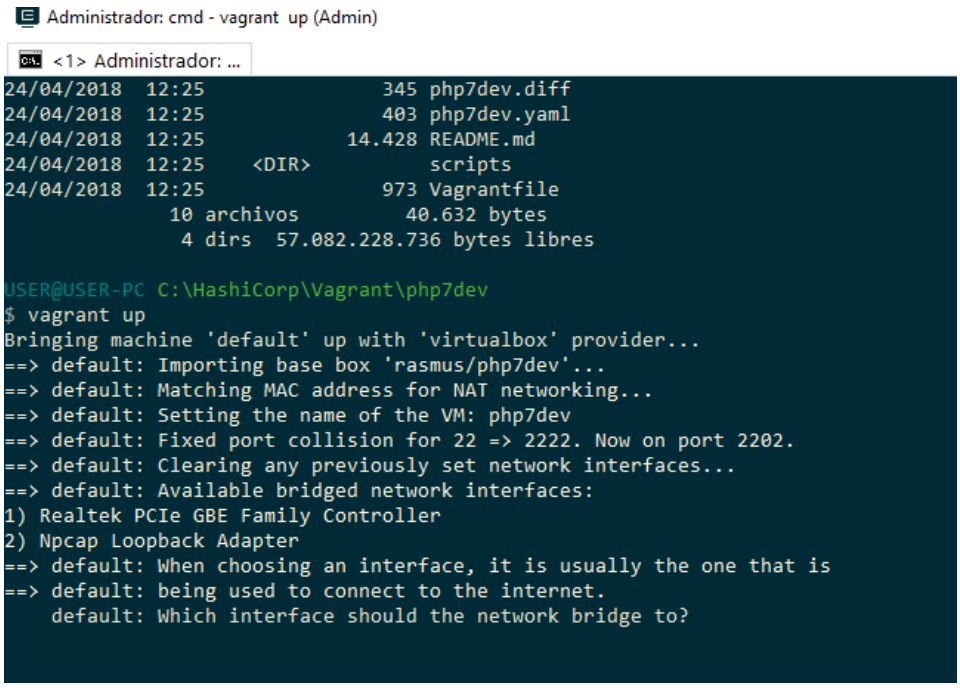
Accedemos al directorio “php7dev”, arrancamos la máquina y accedemos a ella:

```
cd php7dev

vagrant up
```

```
vagrant ssh
```

Duranta el arranque del escenario nos preguntara con que interface nos vamos a conectar en mi caso el interfaz 1.



Si queremos acceder a la web con php7 basta con poner en el navegador:

```
http://192.168.7.7
```

Ya que la configuración de Vagrant crea una red específica para esa máquina y la configura con esa IP estática.

192.168.7.7

PHP Version 7.0.0-dev

System	Linux php7dev 3.2.0-4-amd64 #1 SMP Debian 3.2.65-1+deb7u2 x86_64
Build Date	Mar 29 2015 14:31:00
Configure Command	'./configure' '--with-apxs2=/usr/bin/apxs2' '--enable-zend-signals' '--with-gd' '--without-pear' '--with-jpeg-dir=/usr' '--with-png-dir=/usr' '--with-vpx-dir=/usr' '--with-freetype-dir=/usr' '--with-t1lib=/usr' '--enable-gd-native-ttf' '--enable-exif' '--with-config-file-path=/etc/php7' '--with-config-file-scan-dir=/etc/php7/conf.d' '--with-mysql-sock=/var/run/mysqld/mysqld.sock' '--with-zlib' '--enable-phpdbg' '--with-gmp' '--with-zlib-dir=/usr' '--with-gettext' '--with-kerberos' '--with-imap-ssl' '--with-mcrypt=/usr/local' '--with-iconv' '--enable-sockets' '--with-openssl' '--with-pspell' '--with-pdo-mysql=mysqlnd' '--with-pdo-sqlite' '--enable-soap' '--enable-xmlreader' '--enable-phar=shared' '--with-xsl' '--enable-ftp' '--enable-cgi' '--with-curl=/usr' '--with-tidy' '--with-xmlrpc' '--enable-mbstring' '--enable-sysvsem' '--enable-sysvshm' '--enable-shmop' '--with-readline' '--enable-pcntl' '--enable-fpm' '--enable-intl' '--enable-zip' '--with-imap' '--with-mysqli=mysqlnd' '--enable-calendar' '--enable-debug' '--prefix=/usr/local/php70-debug'
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php7
Loaded Configuration File	/etc/php7/php.ini
Scan this dir for additional .ini files	/etc/php7/conf.d
Additional .ini files parsed	(none)
PHP API	20131218
PHP Extension	20141001
Zend Extension	320140815
Zend Extension Build	API320140815,NTS,debug
PHP Extension Build	API20141001,NTS,debug
Debug Build	yes
Thread Safety	disabled
Zend Signal Handling	enabled

Ejercicio: Directorio sincronizado

Las imágenes de Vagrant suelen crearse con un directorio sincronizado entre la máquina anfitriona y la máquina virtual, que en muchas ocasiones se encuentra en el directorio **/vagrant** de la máquina virtual y el directorio de trabajo de la máquina anfitriona (donde se encuentra el fichero Vagrantfile), esto es configurable pero inicialmente es muy cómodo utilizarlo así:

Arrancar una máquina virtual y comprobar la sincronización del directorio /vagrant

En la maquina virtual se crean en /vagrant

<1> vagrant@php7dev...

```
grant@php7dev:/vagrant$ df -h
lesystem                                      Size  Used Avail Use% Mounted on
otfs                                           16G  4.7G  9.7G  33% /
ev                                             10M     0   10M   0% /dev
pfs                                           101M  224K  101M   1% /run
ev/disk/by-uuid/38618e10-3074-407f-84f8-1c75edd749df   16G  4.7G  9.7G  33% /
pfs                                           5.0M     0  5.0M   0% /run/lock
pfs                                           201M     0  201M   0% /run/shm
ne                                            243G  195G   49G  81% /vagrant
grant@php7dev:/vagrant$
```

Y en la maquina anfitriona será el directorio de proyecto en este ejemplo estamos ulizando la box del laboratorio anterior php7dev, podemos crear un archivo desde la mv y desde el anfitrio y veremos como se sincroniza.

Gestión de imágenes de Vagrant (boxes)

Vagrant denomina box a un paquete que contiene la imagen de una máquina virtual con el formato adecuado para un determinado proveedor y algunos ficheros con metadatos.

Aquí utilizaremos de manera indistinta y premeditada el término “box” o imagen.

Cualquier usuario puede crearse sus propias imágenes de Vagrant aunque existe un sitio gestionado por Hashicorp donde pueden obtenerse multitud de imágenes:

https://atlas.hashicorp.com/boxes/search

Ahora las boxes se gestionan en Vagrant Cloud:

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

ADVERTENCIA : Cualquier usuario puede subir boxes a Atlas

En los últimos años las diferentes distribuciones de GNU/Linux han comenzado a distribuir imágenes de vagrant “oficiales” y son las que más se utilizan, en particular las de Ubuntu.

Ejercicio: descargar un box

En este ejercicio descargaremos el box llamado **bento/centos-7.1** ,desde:

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

Crearemos un nuevo escenario bajo un directorio llamador centos7, y iniciamos el proyecto y nos conectamos a la mv a través de ssh, utilizaremos el comando yum update, para actualizar la mv.

C:\HashiCorp\Vagrant\centos7

Ejercicio: Reempaquetar un box

Cuando nos descargamos una imagen para vagrant se guarda en el directorio, perso ya se descarga desempaquetada:

C:\Users\USER\.vagrant.d\boxes

Con el comando **vagrant box list**, se corresponderá con lo que tengamos en este directorio.

1. Accede a atlas y busca imágenes para virtualbox utilizando diferentes términos e instálalas en tu equipo
2. Obtener un box de una imagen previamente instalada:

```
3. vagrant box repackage centos/7 virtualbox 1703.01
```

Con esto podemos volver a conseguir el fichero box como el que descargamos en atlas y colocarlo en un servidor nuestro apache y desde este servidor distribuirlo para realizar instalaciones:

> vagrant box list

bento/centos-7.1 (virtualbox, 2.2.2)

centos/6 (virtualbox, 1801.01)

centos/7 (virtualbox, 1803.01)

rasmus/php7dev (virtualbox, 0)

ubuntu/trusty64 (virtualbox, 20180201.0.1)

ubuntu/trusty64 (virtualbox, 20180419.0.0)

Con este comando estamos rempaquetador la imagen, que es para virtualbox y la versión, si todo ha ido correcto, recordar que si estamos en Windows, tendremos que iniciar nuestro cmd con permisos de administrador:

```
> vagrant box repackage bento/centos-7.1  virtualbox   2.2.2

>dir

Directorio de C:\HashiCorp\Vagrant

26/04/2016  10:50       359.331.067 package.box
```

1. Borrar la imagen de bento/centos-7.1

```
2. vagrant box remove bento/centos-7.1
```

3. Instalar de nuevo el box de bento/centos-7.1 desde el fichero ".box" en lugar de hacerlo desde Atlas:

```
vagrant box add package.box --name centos7
```

Si ahora hacemos un vagrant box list tendríamos el nuevo box con el nombre centos7.

Ejercicio: Actualización de imágenes

Vagrant a través de atlas nos facilita un mecanismo, mediante el cual nos facilitan imágenes y además que los usuarios los van actualizando funcionalidades, corrigiendo errores y nosotros podremos actualizarlas

Con el comando **vagrant box outdated** y el parámetro **-h** para obtener la ayuda:

$vagrant box outdated -h

1) Comprobar si están actualizadas las imágenes, del directorio donde estoy trabajando y que exista un Vagrantfile:

   a) C:\HashiCorp\Vagrant\ubuntu-trusty64

- Si le pasamos el parámetro **–global** comprobara de todas las versiones:

```
vagrant box outdated --global
```

En el caso de imágenes desactualizadas se pueden eliminar con el comando, recordar que cuando actualizamos una imagen no se elimina la anterior, sino que mantiene la imagen desactualizada y se descarga la nueva imagen actualizada:

```
vagrant box prune
```

2) Para actualizar el box correspondiente a un escenario se utiliza (sólo válido si se ha descargado de Atlas o similar):

```
vagrant box update
```

Ejercicio: Crear un box desde una maquina virtual

El comando **package** de vagrant, crear un box de vagrant a partir de una maquina virtual de VirtualBox que se este ejecutando, nos puede interesar en que si tenemos una mv ya con toda la configuración, la puedo pasar a vagrant

Crear un box de Vagrant desde una máquina virtual de VirtualBox.

Obtenernos el nombre de la máquina virtual, el ejecutable vboxmanage se encuentra:

C:\Program Files\Oracle\VirtualBox

```
VBoxManage list vms
```

La “empaquetamos” como un box de vagrant con:

```
vagrant package --base "Nombre de la MV" --output maquina.box
```

Ejercicio: Obtener un BOX desde una OVA

Lo que realizaremos en este laboratorio será abrir la ova, en nuestro VirtualBox y luego con el subcomando package comvertirlo a vagrant, podríamos seguir la siguiente guía de github:

https://github.com/crohr/ebarnouflant/issues/7

En ocasiones, los proveedores de distribución (como UCS) solo le brindan archivos .ova de VirtualBox para probar su software. A continuación, le mostramos cómo puede importar de manera fácil y no interactiva un archivo .ova a .box para utilizarlo con Vagrant.

C:\Program Files\Oracle\VirtualBox\VBoxManage.exe

$ VBoxManage import c:\HashiCorp\Vagrant\UCS-Virtualbox-Demo-Image.ova --vsys 0 --eula accept

```
$ VBoxManage list vms
"UCS 4.3" {1f1213f4-3861-4f5d-96db-8abbba6bd2ee}
```

So todo ha ido bien el box resultante estará:

C:/Program Files/Oracle/VirtualBox/UCS.box

Copiamos el archivo UCS.box al directorio C:\HashiCorp\Vagrant y lanzamos el comando:

#And add it to the list of your local Vagrant boxes:

```
$ vagrant box add UCS.box --name UCS
$ vagrant box list
UCS             (virtualbox, 0)
```

Para finalizar creamos un directorio y un Vagrantfile para iniciar el proyecto:

C:\HashiCorp\Vagrant\ucs

Vagrantfile

```
Vagrant.configure("2") do |config|
  config.vm.box = "UCS"
end
```

Ahora podemos iniciar nuestro box:

C:\HashiCorp\Vagrant\ucs

```
$ vagrant up
```

VAGRANTS FILES

La configuración de un escenario concreto se realiza de forma bastante simple mediante modificaciones en este fichero, que está escrito en formato Ruby.

Realmente la configuración que se aplica es la aplicación en serie de varios Vagranfiles, tal como se explica en Load Order and Merging , aunque lo más habitual es que se cargue el Vagrantfile que incluye el box y el que exista en el directorio de trabajo, siendo este último el que se modifica en la mayoría de los casos.

Modificaciones de la máquina virtual

Se configuran en el espacio de nombres "**config.vm**", prefijo que antecede a los parámetros en este caso, por ejemplo para modificar el hostname de la máquina utilizaríamos:

```
...

config.vm.hostname = "maquina1"

...
```

El resto de parámetros que se pueden modificar los encontramos en la documentación de Vagrant:Machine Settings , dejando en nuestro caso para secciones posterior algunos de los aspectos que necesitan más desarrollo como la configuración de la red o la configuración integrada de la máquina virtual mediante shell scripts o mediante aplicaciones como ansible o puppet.

Parámetros relativos a las características de hardware de la máquina virtual dependen del proveedor en Vagrant y en el caso de VirtualBox se definen mediante una subsección, pero como en casos anteriores, consideramos más interesante hacerlo mediante algunos ejercicios.

En nuestro laboratorio arrancamos el escenario con **centos7** realizamos la modificación en el Vagrantfile y llamamos a la mv como maquina1.

Ejercicio: Modificar el nombre de la Maquina Virtual

En nuestro laboratorio arrancamos el escenario con **centos7** realizamos la modificación en el Vagrantfile y llamamos a la mv como maquina1.

Vagrantfile

```
Vagrant.configure("2") do |config|
 config.vm.box = "bento/centos-7.1"
 config.vm.hostname = "maquina1"
end
```

Ejercicio: Modificar Hardware Maquina Virtual

Realiza las modificaciones apropiadas en un Vagrantfile para cambiar el nombre de la máquina virtual, la memoria RAM asignada y el número de núcleos virtuales.

```
...
config.vm.provider "virtualbox" do |vb|
  vb.name = "nombre"
  vb.memory = "512"
  vb.cpus = 2
end
....
```

En el laboratorio realizamos sobre el **escenario centos7**, cada driver tiene su propia configuración, en nuestro caso estamos utilizando el de facto para vagrant que es VirtualBox:

Vagrantfile

```
Vagrant.configure("2") do |config|
  config.vm.box = "bento/centos-7.1"
  config.vm.hostname = "maquina1"
  config.vm.provider "virtualbox" do |vb|
     vb.name = "maquina1"
     vb.memory = "512"
     vb.cpus = 2
 end
end
```

Iniciamos la mv y comprobamos como se ve en el inventario de VirtualBox

```
$ vagrant up
$ VBoxManage list vms
```

Si entramos dentro de la vm:

```
[vagrant@maquina2 ~]$ free -m
              total        used        free      shared  buff/cache   available
Mem:            489          66         188           4         233         323
[vagrant@maquina2 ~]$ free -m
```

Veremos que tenemos 2 cpus cores:

[vagrant@maquina2 ~]$ cat /proc/cpuinfo

Ejercicio: Máquina virtual con interfaz gráfica

Como la forma habitual de gestionar máquinas virtuales en Vagrant es mediante la línea de comandos y accediendo a ellas a través de ssh, no tiene mucho sentido que se arranque una interfaz gráfica, pero en algunas ocasiones es conveniente. Modifica un fichero Vagrantfile para que se inicie la interfaz gráfica de usuario al levantar la máquina y ver como esta arrancado nuestra mv.

```
...
config.vm.provider "virtualbox" do |vb|
  vb.gui = true
end
...
```

Vagrantfile

```
Vagrant.configure("2") do |config|
  config.vm.box = "bento/centos-7.1"
  config.vm.hostname = "maquina3"
  config.vm.provider "virtualbox" do |vb|
     vb.name = "maquina3"
     vb.memory = "512"
     vb.cpus = 2
     vb.gui = true
 end
end
```

EJERCICIO: APROVISIONAMIENTO LIGERO

Aprovisionamiento ligero (*thin provisioning*) es una técnica muy utilizada en diferentes sistemas de virtualización y consiste en crear un disco de imagen de máquina virtual que incluya sólo las modificaciones respecto a una imagen base, consiguiendo un ahorro significativo de espacio en disco a costa de una pequeña penalización en rendimiento. **Configura un Vagrantfile** para que se realice aprovisionamiento ligero.

```
....

config.vm.provider "virtualbox" do |vb|

  vb.name = "ligera"

  vb.linked_clone = true

end

...
```

En este laboratorio nos creamos un nuevo escenario basado en el **escenario de centos7**, copiamos la carpeta y realizamos la siguiente modificación en Vagrantfile:

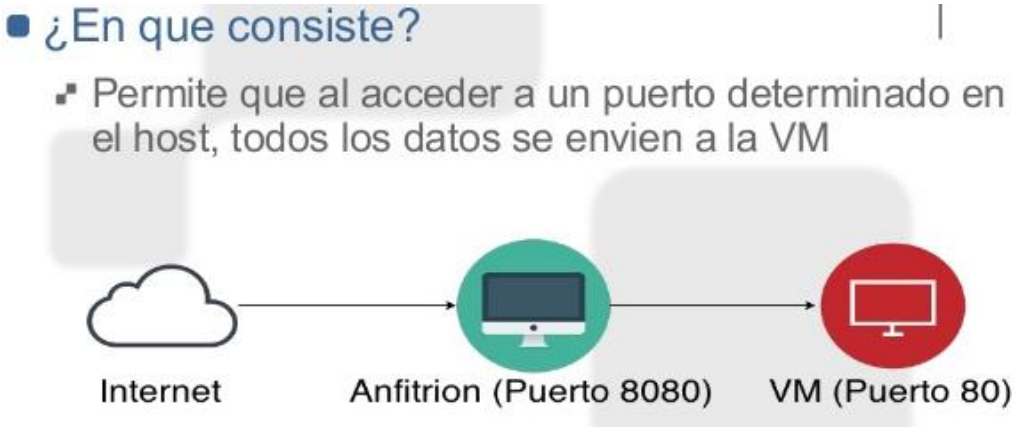
Vagrantfile

```
Vagrant.configure("2") do |config|
  config.vm.box = "bento/centos-7.1"
  config.vm.hostname = "ligera"
  config.vm.provider "virtualbox" do |vb|
     vb.name = "ligera"
     vb.memory = "512"
     vb.cpus = 2
          vb.linked_clone = true
 end
end
```

Si arrancamos los dos escenarios y comprobamos el tamaño de los discos duros de los dos escenarios, veremos la diferencia.

C:\Users\USER\VirtualBox VMs

Ejercicio: Redirección de puertos



Una funcionalidad muy útil y sencilla es la redirección de puertos de la red por defecto que utiliza Vagrant (red interna con NAT). Configura un Vagrantfile para que las peticiones al puerto 8080/tcp de la máquina anfitriona se redirijan al puerto 80/tcp de la máquina virtual.

La documentación completa se encuentra en Forwarded Ports .

```
...

config.vm.network "forwarded_port", guest: 80, host: 8080

...
```

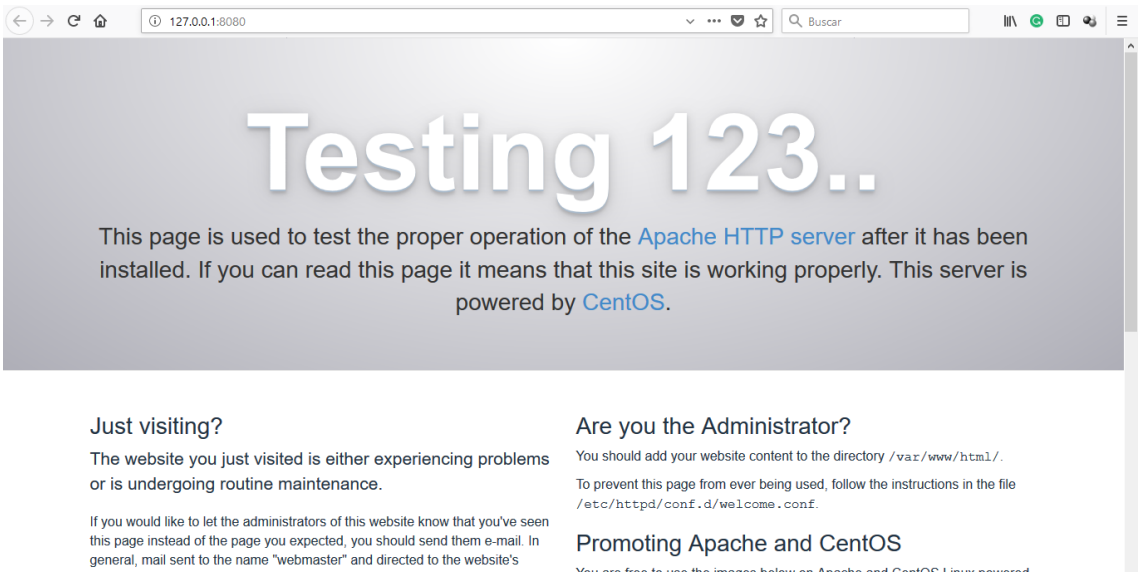
Podemos ver en todo momento los puertos que se han redireccionado con la instrucción:

```
vagrant port
```

Estos cambios se pueden realizar sobre una máquina ya funcionando y para que se apliquen se utiliza la opción:

```
vagrant reload
```

Para nuestro laboratorio arrancamos **el apache** en nuestro proyecto **centos7**, y mapeamos el puerto 8080 en el invitado y a continuación tendremos que llegar al apache de la mv, desde el localhost del invitado y desde la ip del invitado a través del puerto 8080.




VAGRANT

EJERCICIO: AÑADIR DISCO ADICIONAL (No realizar, lo realizaremos mediante el pluguin vagrant-persistent-storage)

Configura una máquina virtual que tenga un disco adicional de 5 GiB.

Editamos el fichero Vagrantfile e incluímos las líneas:

```
....
config.vm.provider "virtualbox" do |vb|
  file_to_disk = 'tmp/disk.vdi'
  unless File.exist?(file_to_disk)
    vb.customize ['createhd',
                  '--filename', file_to_disk,
                  '--size', 5 * 1024]
  end
  vb.customize ['storageattach', :id,
                '--storagectl', 'SATAController',
                '--port', 1,
                '--device', 0,
                '--type', 'hdd',
                '--medium', file_to_disk]
end
...
```

En el directorio de trabajo podremos ver que se ha creado un fichero en formato vdi::

Vagrantfile:

```
Vagrant.configure("2") do |config|
  config.vm.box = "bento/centos-7.1"
  config.vm.hostname = "maquina3"
  config.vm.network "forwarded_port", guest: 80, host: 8080
  config.vm.provider "virtualbox" do |vb|
    vb.name = "maquina3"
    vb.memory = "512"
    vb.cpus = 2
        vb.gui = true
    file_to_disk = 'C:\HashiCorp\Vagrant\disk.vdi'
    unless File.exist?(file_to_disk)
      vb.customize ['createhd',
                '--filename', file_to_disk,
                '--size', 5 * 1024]

end
vb.customize ['storageattach', :id,
```

```
            '--storagectl', 'SATAController',
            '--port', 1,
            '--device', 0,
            '--type', 'hdd',
            '--medium', file_to_disk]


  end
end
```

```
ls -hl tmp/
total 2,0M
-rw------- 1 alberto alberto 3,0M abr 23 10:42 disk.vdi
```

Y desde la máquina virtual veremos un disco adicional de 500GiB:

```
lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda      8:0    0    40G  0 disk
└─sda1   8:1    0    40G  0 part /
sdb      8:16   0   500G  0 disk
```

NOTA: Este tipo de configuraciones en las que se pone de forma explícita las características de la máquina virtual no son ni mucho menos generales, en el caso anterior se está poniendo de forma concreta el puerto SATA al que conectar el disco y el

nombre del controlador SATA, características que pueden variar de una máquina virtual a otra. Suele ser conveniente obtener previamente información de las características de la máquina virtual, que en el caso de VirtualBox se puede hacer con el siguiente comando de VBoxManage:

```
VBoxManage showvminfo NOMBREDELAMV
```

INTRODUCCIÓN A LA LINEA DE COMANDOS

https://www.vagrantup.com/docs/cli/

nit

- *vagrant init [box-name] [box-url]*
- Cria o arquivo Vagrantfile com as configurações do box que você informou no [box-name], caso você não tenha o box ainda adicionado é obrigatório passar o [box-url] para que ele possa baixa-lo

Up

- *vagrant up*
  - Cria e inicia a instancia após o comando *vagrant init*

Reload

- *vagrant reload*
  - Reinicia a instancia do box ativo

Suspend

- *vagrant suspend*
  - Stopa a instancia ativa, congelando seu estado atual

Resume

- *vagrant resume*
  - Ativa a instancia suspensa, até então, pelo comando *vagrant suspend*

Halt

- *vagrant halt*
  - Manda um comando para desligar a instancia ativa, finalizando todos os processos antes de finalizar

Destroy

- *vagrant destroy*
  - Destroy a instancia ativa

SSH

- *vagrant ssh*
  - Acessa a instancia ativa via ssh

Status

- *vagrant status*
  - Informa o status atual da instancia

Box

- *vagrant box add name url*
  - Adiciona um novo box a sua lista de box
- *vagrant box list*
  - Lista suas box para utilização
- *vagrant box remove name*
  - Remove um box da lista de box

Ejercicio: Suspender y reanudar una maquina (Comandos de Vagrant)

Suspende una máquina virtual que esté arrancada, comprueba el estado y reanúdala, cuando se suspende la mv se suspende a disco, se guarda la imagen de la mv, tal y como estaba y su estado de memoria se almacena en disco, de manera de que se pueda restaurar exactamente el estado que tenia la maquina, no solo con lo que teneia en disco, sino con lo que tenia en memoria ram .

Para suspenderla a disco:

```
$vagrant suspend
```

Comprobamos su estado:

```
$vagrant status
Current machine states:

default                   saved (virtualbox)

To resume this VM, simply run `vagrant up`.
```

Y se reanuda con "**vagrant up**" o con "**vagrant resume**", este último no hace ningún tipo de comprobación sobre la máquina, el box, etc. simplemente reanuda la máquina desde su estado de suspensión.

El comando **vagrant up**, realizaría una serie de comprobaciones sobre el vagranfile, si a camniado respecto a cuando se suspendio.

Ejercicio: Configuracion SSH

Comprueba la configuración ssh de una máquina vagrant, lo que muestra es la salida es un fichero de configuración de cliente ssh, seria el mismo que tenemos en un Linux en el .ssh/config, con esto veremos la configuración que esta utilizando vagrant para conectarse a través de ssh.va

```
$ vagrant ssh-config
Host default
  HostName 127.0.0.1
  User vagrant
  Port 2222
  StrictHostKeyChecking no
  StrictHostKeyChecking no
  PasswordAuthentication no
  IdentityFile /home/alberto/Prueba1/.vagrant/machines/default/virtualbox/private_key
  IdentitiesOnly yes
  LogLevel FATAL
```

Lo que nos dice es que se esta conectando al puerto 2222, con el usuario vagrant al 127.0.0.1, no esta guardando la clave publica del servidor ssh cuando se conecta

(**StrictHostKeyChecking no**), no utilizaremos el fichero de hosts conocidos (**StrictHostKeyChecking no**)

No utiliza autenticación a través de contraseña (**PasswordAuthentication no**), y lo que utiliza es una clave privada que vagrant a creado automáticamente

(home/alberto/Prueba1/.vagrant/machines/default/virtualbox/private_key)

Si estamos utilizando Windows:

C:\HashiCorp\Vagrant\centos7\.vagrant\machines\default\virtualbox

Ejercicio: Gestionar instantánea

Un caso típico de uso de instantáneas, es cuando pretendemos probar un nuevo software, para lo que antes de modificar nada, hacemos una *instantánea* de la MV y luego ya instalamos el software, lo configuramos, lo probamos y si hay problemas o no nos gusta, usamos la *instantánea* para regresar al estado en el que partimos inicialmente.

1. Lanza una máquina vagrant, realiza alguna modificación, guarda una instantánea, realiza una segunda modificación y restaura la máquina desde la instantánea.

   Creamos una máquina, accedemos a ella e instalamos un paquete (por ejemplo apache", redirigimos el puerto 8080/tcp de la anfitriona al 80/tcp de la máquina virtual. Realizamos una instantánea (*snapshot*) de la máquina:

   vagrant snapshot save apache-limpio
   ==> apache-limpio: Snapshotting the machine as 'apache-limpio'...
   ==> apache-limpio: Snapshot saved! You can restore the snapshot at any time by
   ==> apache-limpio: using `vagrant snapshot restore`. You can delete it using
   ==> apache-limpio: `vagrant snapshot delete`.

   Podemos comprobar en el directorio de máquinas virtuales de VirtualBox el contenido del directorio Snapshots.

   Realizamos una modificación del index.html y restauramos la máquina al estado que tenía cuando se hizo la instantánea:

   vagrant snapshot restore apache-limpio

2. Existe un método más sencillo de realizar instantáneas, es mediante el comando "vagrant snapshot push" que va almacenando instantáneas cada vez que se invoca y se recupera la más reciente con "vagrant snapshot pop".

En nuestro laboratorio levantamos la mv, basada en el box

$ vagrant box list

bento/centos-7.1 (virtualbox, 2.2.2)

Nos conectamos a la mv:

$ vagrant sshd

Y nos convertimos en root, con el comando (recordar que el pass es vagrant):

$su –

Comprobamos que no tenemos apache instalado:

[root@localhost ~]# rpm -qa httpd

En este momento tomaremos una instantánea de la mv:

$ vagrant snapshot save sin-apache

==> default: Snapshotting the machine as 'sin-apache'...

==> default: Snapshot saved! You can restore the snapshot at any time by

==> default: using `vagrant snapshot restore`. You can delete it using

==> default: `vagrant snapshot delete`.

A continuación nos conectamos a la mv y instalamos y iniciamos apache a través de yum:

[root@localhost ~]# yum install httpd

[root@localhost ~]# systemctl start httpd

[root@localhost ~]# systemctl enable httpd

Le redirigimos el puerto localmente, para conectarnos al apache:

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

  ...
```

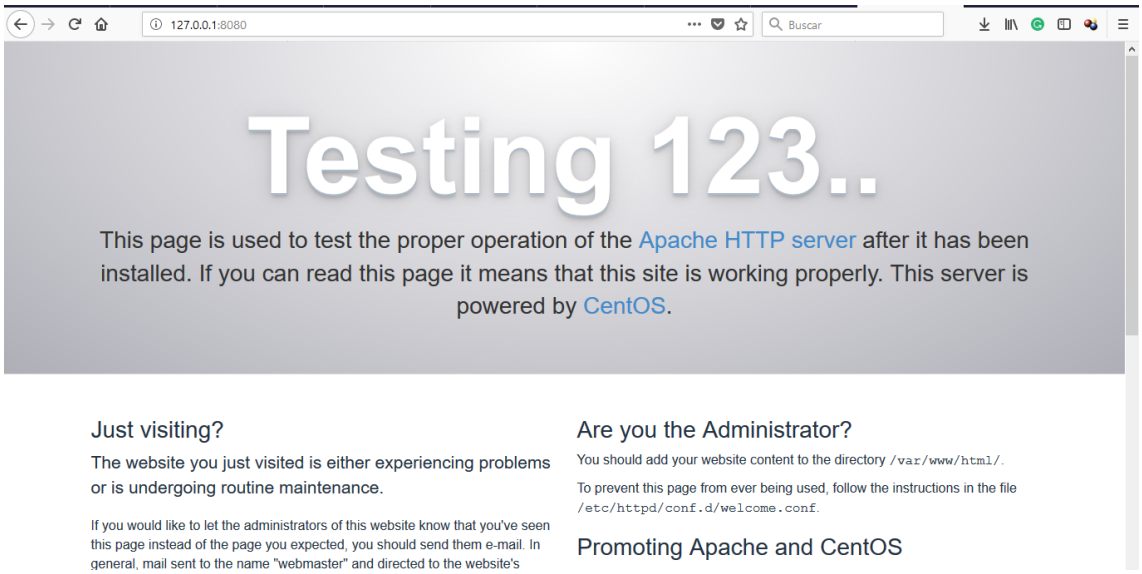
Podemos ver en todo momento los puertos que se han redireccionado con la instrucción:

```
vagrant port
```

Estos cambios se pueden realizar sobre una máquina ya funcionando y para que se apliquen se utiliza la opción:

```
vagrant reload
```

Comprobamos el correcto funcionamiento de todos los pasos desde nuestro cliente:



Ruta donde la mv tiene los snapshots:

C:\Users\USER\VirtualBox VMs\centos7_default_1524994473221_95791\Snapshots

Ahora procederemos a volver al estado en en que no teníamos apache:

$ vagrant snapshot restore sin-apache

Comprobamos que ya no tenemos el apache configurado ni el puerto expuesto:

$ vagrant port

  22 (guest) => 2222 (host)

Por ultimo eliminamos el Snapshots:

$ vagrant snapshot -h

$ vagrant snapshot list

$ vagrant snapshot delete sin-apache

Podemos comprobar que el snpshost ya no esta:

C:\Users\USER\VirtualBox VMs\centos7_default_1524994473221_95791\Snapshots

Redes en Vagrant

Cuando utilizamos una máquina virtual, aunque no definamos ninguna red en el fichero Vagrantfile, ya sabemos que en la inmensa mayoría de los casos vagrant realiza los pasos necesarios para que esa máquina virtual sea accesible a través de una red virtual desde el equipo anfitrión y tenga acceso a Internet. Esta configuración lógicamente va a variar en función del proveedor, pero en el caso de VirtualBox se trata de una red de tipo NAT con el direccionamiento inicial 10.0.2.0/24.

Red privada

En algunas ocasiones tendremos la necesidad de utilizar un direccionamiento IP específico en una máquina virtual o añadir una red virtual adicional, por lo que en muchos casos es adecuado añadir una red privada, tal como se explica en Private Networks .

Una red privada es una red que vagrant va a conectar la mv con un direcrionamiento privado, y nosotros accederemos a esa maquina poniéndole un direccionamiento estatico, se utiliza sobre todo cuando queremos tener conectividad desde el anfitrión a la mv a través de una ip, o bien entre varias maquinas virtuales, por ejemplo una maquina virtual con un servidor web y otra con una base de datos, queremos que el servidor web tenga acceso hacia fuera pero que la base de datos no tenga acceso hacia afuera, pero entre la base de datos y el servidor web tenemos conexión privada y que solo el servidor web acceda al puerto de conexión de la base de datos, pero solo desde el servidor web

Red pública

Para vagrant una red pública es una red en modo puente (*bridged network*) conectada al exterior, típicamente sería poner en la misma red la máquina anfitriona y la máquina virtual. Más detalles en Public Networks.

Una red publica en vagrant, es engancharse a la misma red que la maquina anfitriona, en modo puente, que si a la red en la que se encuentra nuestra maquina anfitriona es una red 192.168.1.1 y tenemos un servidor dhcp externo, será este servidor dhcp el que le proporciona ip a la mv y esta mv será accesible desde toda la red y no solo desde la maquina anfitriona y no será necesario realizar redirecion de puertos.

Ejercicio: Red privada

En este laboratorio, lo realizaremos sobre centos7, veremos la inclusión en el archivo Vagranfile, de una red privada, de manera que la maquina vagrant a demás de utilizar la red vagrant que configura a todas la maquinas, va a añadirnos un interfaz de red mas, conectado a la red privada que nosotros configuremos y esta red estará conectda tanto la maquina anfitriona, como la maquina de vagrant

Configura un Vagrantfile en el que la máquina virtual tenga la dirección IP estática 192.168.33.10

Editamos el fichero Vagrantfile y añadimos la línea:

```
...
config.vm.network "private_network", ip: "192.168.33.10"
...
```

Al acceder por ssh a la máquina (utilizando la red NAT como habitualmente) podemos ver que la máquina virtual tiene dos interfaces de red, una en la red NAT con dirección IP del rango 10.0.2.0/24 y otra la dirección IP estática que hemos asignado:

```
ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
group default
    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: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:98:b7:7b brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe98:b77b/64 scope link
       valid_lft forever preferred_lft forever
3: eth1:
<BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state
UP group default qlen 1000
    link/ether 08:00:27:68:02:01 brd ff:ff:ff:ff:ff:ff
    inet 192.168.33.10/24 brd 192.168.33.255 scope global eth1
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe68:201/64 scope link
       valid_lft forever preferred_lft forever
```

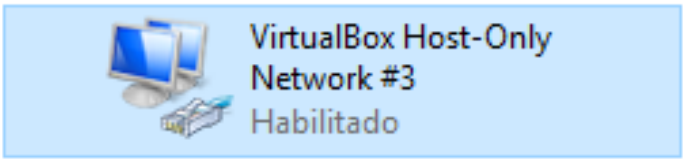
Y en la tabla de encaminamiento podemos ver que la máquina virtual sale a Internet por la red NAT:

```
ip r
default via 10.0.2.2 dev eth0
10.0.2.0/24 dev eth0  proto kernel  scope link  src 10.0.2.15
192.168.33.0/24 dev eth1  proto kernel  scope link  src
192.168.33.10
```

Podemos acceder a la máquina virtual utilizando la nueva red privada, ya que la máquina anfitriona también está conectada a esa red:

```
   ssh -i .vagrant/machines/default/virtualbox/private_key
vagrant@192.168.33.10
```

En la maquina anfitriona crea un interfaz con la ip 192.168.33.1, a través de la cual nos conectamos entre el anfitrio y la mv de vagrant utilizando esta red.



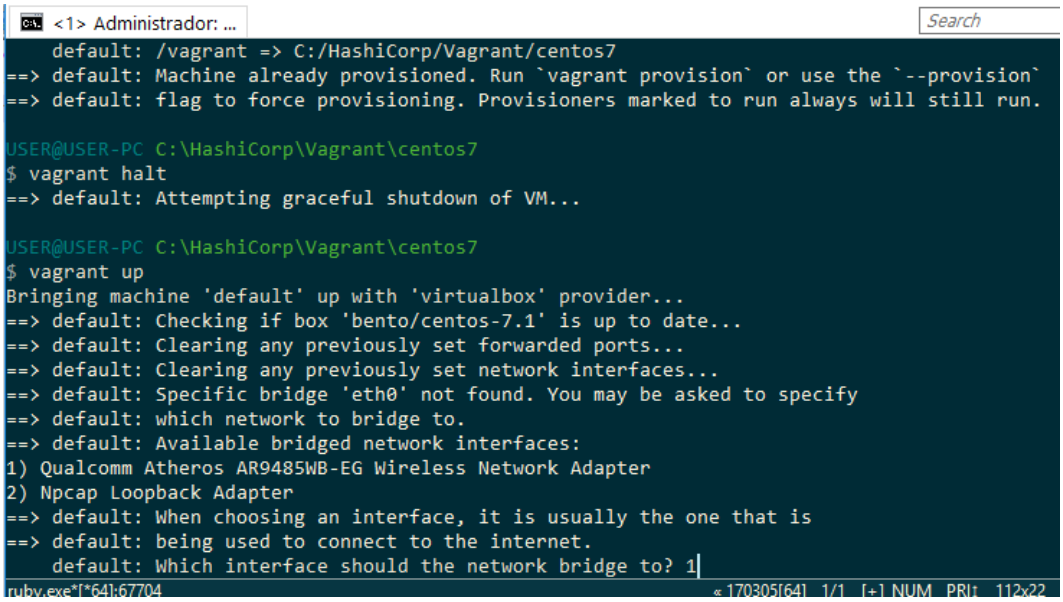
Ejercicio: Red pública

Configura un entorno de vagrant para que la máquina virtual esté conectada en modo puente a la misma red que la máquina anfitriona.

```
config.vm.network  "public_network", bridge: "eth0"
```

Accedemos a la máquina y vemos las interfaces de red así como la configuración de /etc/network/interfaces:

```
...
#VAGRANT-BEGIN
# The contents below are automatically generated by Vagrant. Do not
modify.
auto eth1
iface eth1 inet dhcp
    post-up route del default dev $IFACE || true
#VAGRANT-END
```



Ejemplo de configuración de vagrantfile, para salir desde mi portátil con una conexión wifi:

Vagrantfile

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

  config.vm.box = "bento/centos-7.1"

  config.vm.network  "public_network", bridge: "wlan0"

end
```

En la maquina virtual de vagrant:

[vagrant@localhost ~]$ ip a

2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000

link/ether 08:00:27:f6:b0:07 brd ff:ff:ff:ff:ff:ff

inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic enp0s3

valid_lft 86125sec preferred_lft 86125sec

inet6 fe80::a00:27ff:fef6:b007/64 scope link

valid_lft forever preferred_lft forever

3**: enp0s8**: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000

link/ether 08:00:27:04:9a:d6 brd ff:ff:ff:ff:ff:ff

inet **192.168.1.18/24** brd 192.168.1.255 scope global dynamic enp0s8

valid_lft 3334sec preferred_lft 3334sec

inet6 fe80::a00:27ff:fe04:9ad6/64 scope link

valid_lft forever preferred_lft forever

Entornos multimáquinas

Se puede extender la configuración de un escenario virtual con vagrant a aquellos casos en los que haya más de una máquina virtual y esto se hace con estructuras como la siguiente en el Vagrantfile:

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

  config.vm.define "maquina1" do |m1|

    m1.vm.box = "ubuntu/trusty64"

  end

  config.vm.define "maquina2" do |m2|

    m2.vm.box = "ubuntu/trusty64"

  end

end
```

Al existir más de una máquina virtual, muchas de las instrucciones que hemos visto hasta ahora deben incluir el nombre de la máquina como parámetro, por ejemplo si hiciéramos:

```
vagrant ssh

This command requires a specific VM name to target in a multi-VM environment.
```

Que deberíamos cambiar por algo como:

```
vagrant ssh maquina1
```

Redes

Cuando se utilizan entornos multi-máquinas es muy habitual utilizar redes privadas y configurar la comunicación entre las máquinas del escenario mediante direcciones IP estáticas.

EJERCICIO: MULTIMÁQUINAS

Crea un escenario para una aplicación web en la que ubicamos en una máquina el servidor web conectado a una red pública exterior y una máquina con la base de datos en una máquina no accesible desde el exterior. Para que ambas máquinas estén interconectadas entre sí, crea la red privada 10.0.100.0/24 a la que estarán conectadas ambas máquinas.

Nos basaremos en el box bento/centos-7.1 y estamos en el escenarios centos7:

```
Vagrant.configure("2") do |config|
  config.vm.define "web" do |nodo1|
    nodo1.vm.box = "bento/centos-7.1"
    nodo1.vm.hostname = "web"
    nodo1.vm.network "public_network", bridge: "eth0"
    nodo1.vm.network "private_network", ip: "10.0.100.101"
  end
  config.vm.define "db" do |nodo2|
    nodo2.vm.box = "bento/centos-7.1"
    nodo2.vm.hostname = "db"
    nodo2.vm.network "private_network", ip: "10.0.100.102"
  end
end
```

Vagrantfile

```
Vagrant.configure("2") do |config|
 config.vm.define "web" do |nodo1|
    nodo1.vm.box = "bento/centos-7.1"
    nodo1.vm.hostname = "web"
    nodo1.vm.network "public_network", bridge: "eth0"
    nodo1.vm.network "private_network", ip: "10.0.100.101"
  end
  config.vm.define "db" do |nodo2|
    nodo2.vm.box = "bento/centos-7.1"
    nodo2.vm.hostname = "db"
    nodo2.vm.network "private_network", ip: "10.0.100.102"
  end
 end
```

Como se puede observar en la configuración las maquinas se llaman nodo1 y nodo2.

Les hemos puesto a las maquinas virtuales el hostname de web y db, ambos utilizan el mismo box bento/centos-7.1, las maquinas están conectada en puente a través del interfaz de red en mi caso de la wifi, y de modo interno se comunican entre ellas a través de una red privada.

$ vagrant up

$ vagrant ssh web

```
[vagrant@web ~]$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
   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 pfifo_fast state UP qlen 1000

link/ether 08:00:27:f6:b0:07 brd ff:ff:ff:ff:ff:ff

inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic enp0s3

valid_lft 86113sec preferred_lft 86113sec

inet6 fe80::a00:27ff:fef6:b007/64 scope link

valid_lft forever preferred_lft forever

3**: enp0s8**: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000

link/ether 08:00:27:7c:f2:0f brd ff:ff:ff:ff:ff:ff

inet 192.168.1.19/24 brd 192.168.1.255 scope global dynamic enp0s8

valid_lft 3296sec preferred_lft 3296sec

inet **192.168.1.20/24** brd 192.168.1.255 scope global secondary dynamic enp0s8

valid_lft 3342sec preferred_lft 3342sec

inet6 fe80::a00:27ff:fe7c:f20f/64 scope link

valid_lft forever preferred_lft forever

4: **enp0s9**: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000

link/ether 08:00:27:60:c5:01 brd ff:ff:ff:ff:ff:ff

inet **10.0.100.101/24** brd 10.0.100.255 scope global enp0s9

valid_lft forever preferred_lft forever

En la maquina virtual para db, se puede observar que solo tiene el interfaz de la red privada, para comunicarse solamente desde el servidor web, con lo que nos es accesible desde nuestra red local, es decir solo es accesible desde la red privada del servidor web:

$ vagrant ssh db

[vagrant@db ~]$ ip a

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN

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 pfifo_fast state UP qlen 1000

link/ether 08:00:27:f6:b0:07 brd ff:ff:ff:ff:ff:ff

inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic enp0s3

valid_lft 86063sec preferred_lft 86063sec

inet6 fe80::a00:27ff:fef6:b007/64 scope link

valid_lft forever preferred_lft forever

3: **enp0s8**: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000

link/ether 08:00:27:8b:d2:da brd ff:ff:ff:ff:ff:ff

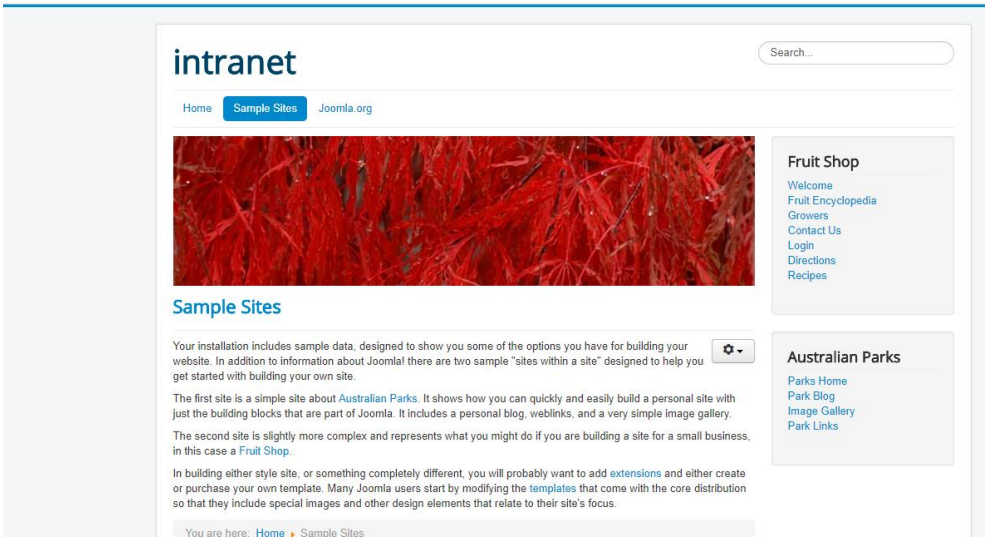
inet **10.0.100.102/24** brd 10.0.100.255 scope global enp0s8

valid_lft forever preferred_lft forever

Para la correcta comprensión de este laboratorio, el formador guiara como instalar Joomla, en el servidor web y que la base de datos este en el servidor db, tras la correcta configuración, pararemos las mv y cuando se vuelvan a levantar todo tiene que estar funcionando correctamente:

Desde nuestro navegador de Windows:

http://10.0.100.101/intranet



Paramos y iniciamos el proyecto:

USER@USER-PC C:\HashiCorp\Vagrant\centos7

$ vagrant halt

$ vagrant up

Y comprobamos que todo esta correcto y nos podemos conectar a la web de Jooma.

http://10.0.100.101/intranet/

CONFIGURACIÓN INTEGRADA

Vagrant denomina aprovisionamiento (*provisioning*) a los procesos de configuración de la máquina virtual una vez que esta ha finalizado el arranque, procesos que pueden incluir la ejecución de comandos de shell, la ejecución de un shell script completo o la utilización de herramientas de la gestión de la configuración (*configuration management systems*) como puppet, chef o ansible.

Este proceso se realiza típicamente la primera vez que se inicia un entorno vagrant, cuando se invoca directamente desde la línea de comandos con “vagrant provision” o cuando se recarga un entorno de vagrant con la opción “—provision”, en el resto de casos en los que volvamos a iniciar un entorno no se volverá a configurar.

El campo de las herramientas de la gestión de la configuración es muy amplio y su explicación detallada sale totalmente del ámbito de este curso. En el caso de vagrant la solución que se ha encontrado es muy razonable, ya que en lugar de desarrollar una nueva herramienta para competir con las cuatro soluciones más utilizadas hoy en día: ansible, puppet, chef y salt, la gente de Hashicorp ha optado por una solución muy adecuada y razonable: integrarlas todas con vagrant.

EJERCICIO: ACTUALIZACIÓN AUTOMÁTICA DE LA MÁQUINA

En este laboratorio vamos a integrar en el Vagrantfile, lo que se demomina fase dos, en este laboratorio vamos a lanzar un script, que lo que va a realizar es actualizar todo nuestro sistema, cuando se inicie la maquina virtual, utilizaremos el escenario **ubuntu/trusty64.**

Crea un entorno vagrant sencillo en el que tras arrancar una máquina se realice una actualización de la lista de paquetes y se efectúen las actualizaciones de aquellos paquetes necesarios mediante apt-get

Creamos el fichero actualizar.sh ubicado en el mismo directorio que el Vagrantfile y con el siguiente contenido:

```
#!/bin/bash

apt-get update

DEBIAN_FRONTEND=noninteractive apt-get -y upgrade
```

Creamos el siguiente Vagrantfile:

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

  config.vm.box = "ubuntu/trusty64"

  config.vm.provision "shell", path: "actualizar.sh"

end
```

Ejercicio: Actualización automática de la máquina con Ansible

1. Crea un entorno idéntico al anterior, realizando las mismas tareas, pero ahora utilizando ansible, que debe estar instalado en la máquina anfitriona.

   Creamos un fichero site.yml de configuración de ansible para que se realice la actualización de paquetes:

```
---

- hosts: all

  sudo: True

  tasks:

    - name: Ensure system is updated

      apt: update_cache=yes upgrade=yes
```

   Y el correspondiente Vagrantfile sería:

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

  config.vm.box = "ubuntu/trusty64"

  config.vm.provision "ansible" do |ansible|

    ansible.playbook = "site.yml"

  end

end
```

2. Más que un ejercicio, el siguiente es un ejemplo de configuración integrada de vagrant con ansible, desplegando un clúster de balanceo de carga mediante un round-robin en un servidor DNS

Clona el siguiente repositorio y sigue las instrucciones que se incluyen en el fichero README.md:

https://github.com/albertomolina/ejemplo-ansible-vagrant

PLUGINS

La funcionalidad de Vagrant es extensible mediante plugins, desarrollados por la propia Hashicorp o por terceros. La documentación de Vagrant describe de forma detallada los conceptos básicos para el desarrollo de los mismos:

https://www.vagrantup.com/docs/plugins/

La arquitectura que tiene vagrant, muchos de los componentes con los que estamos trabajando son plugins, podríamos hablas de plugins integrados o plugins adicionales, cuando nosotros instalamos vagrant muchas de las funcionalidades que estamos utilizando, de provedores, configuración integrada, son realmente pluguins que ya están integrados en la arquitectura de vagrant y la diferencia de otros pluguins, son otros pluguins, bien desarrollado por Hasicorp o por empresas de terceros

EJERCICIO: VAGRANT PARA UTILIZAR AWS COMO PROVEEDOR

Siguiendo con la metodología de este curso, vamos a mostrar algunos ejemplos del uso de plugins mediante algunos ejercicios.

Ejercicios

1. Comprueba los plugins instalados en vagrant

```
vagrant plugin list

vagrant-share (1.1.7, system)
```

2. Instala el plugin de vagrant para utilizar AWS como proveedor (desarrollado también por M. Hashimoto).

   Este plugin se encuentra en

   Github: https://github.com/mitchellh/vagrant-aws

La instalación del plugin es realmente sencilla, basta con hacer:

```
vagrant plugin install vagrant-aws

Installing the 'vagrant-aws' plugin. This can take a few minutes...

Fetching: ipaddress-0.8.3.gem (100%)

Fetching: formatador-0.2.5.gem (100%)

Fetching: excon-0.55.0.gem (100%)

Fetching: fog-core-1.43.0.gem (100%)

Fetching: fog-xml-0.1.3.gem (100%)

Fetching: fog-json-1.0.2.gem (100%)

Fetching: trollop-2.1.2.gem (100%)

Fetching: CFPropertyList-2.3.5.gem (100%)

Fetching: rbvmomi-1.11.0.gem (100%)

Fetching: fission-0.5.0.gem (100%)

Fetching: inflecto-0.0.2.gem (100%)

Fetching: xml-simple-1.1.5.gem (100%)

Fetching: fog-digitalocean-0.3.0.gem (100%)

Fetching: fog-xenserver-0.3.0.gem (100%)

Fetching: fog-vsphere-1.9.1.gem (100%)

Fetching: fog-voxel-0.1.0.gem (100%)
```

```
Fetching: fog-vmfusion-0.1.0.gem (100%)

Fetching: fog-terremark-0.1.0.gem (100%)

Fetching: fog-storm_on_demand-0.1.1.gem (100%)

Fetching: fog-softlayer-1.1.4.gem (100%)

Fetching: fog-serverlove-0.1.2.gem (100%)

Fetching: fog-sakuracloud-1.7.5.gem (100%)

Fetching: fog-riakcs-0.1.0.gem (100%)

Fetching: fog-radosgw-0.0.5.gem (100%)

Fetching: fog-rackspace-0.1.5.gem (100%)

Fetching: fog-profitbricks-3.0.0.gem (100%)

Fetching: fog-powerdns-0.1.1.gem (100%)

Fetching: fog-openstack-0.1.20.gem (100%)

Fetching: fog-local-0.3.1.gem (100%)

Fetching: fog-google-0.1.0.gem (100%)

Fetching: fog-ecloud-0.3.0.gem (100%)

Fetching: fog-dynect-0.0.3.gem (100%)

Fetching: fog-dnsimple-1.0.0.gem (100%)

Fetching: fog-cloudatcost-0.1.2.gem (100%)

Fetching: fog-brightbox-0.11.0.gem (100%)
```

```
Fetching: fog-aws-1.3.0.gem (100%)

Fetching: fog-atmos-0.1.0.gem (100%)

Fetching: fog-aliyun-0.1.0.gem (100%)

Fetching: iniparse-1.4.2.gem (100%)

Fetching: fog-1.40.0.gem (100%)

------------------------------

Thank you for installing fog!



IMPORTANT NOTICE:

If there's a metagem available for your cloud provider,

e.g. `fog-aws`,

you should be using it instead of requiring the full fog collection

to avoid

unnecessary dependencies.



'fog' should be required explicitly only if:

- The provider you use doesn't yet have a metagem available.

- You require Ruby 1.9.3 support.

------------------------------
```

```
Fetching: vagrant-aws-0.7.2.gem (100%)

Installed the plugin 'vagrant-aws (0.7.2)'!
```

Tal como se indica en la documentación, es necesario instalar un box "dummy" para que no nos dé un error de formato el Vagrantfile:

```
vagrant box add dummy https://github.com/mitchellh/vagrant-aws/raw/master/dummy.box
```

Se podría utilizar directamente poniendo las credenciales de acceso en el fichero Vagrantfile, pero parece más razonable utilizar un entorno de AWS estándar como el siguiente:

```
/home/alberto/.aws

├── config

└── credentials
```

Siendo el contenido del fichero config:

```
[default]

region=eu-west-1
```

Y del fichero credentials:

```
[default]

aws_access_key_id = ...

aws_secret_access_key = ...
```

Como último paso previo, cargamos en el agente ssh de nuestro entorno la clave privada ssh que queramos utilizar en AWS, por ejemplo:

```
ssh-add ~/.ssh/clave-aws.pem
```

Y utilizamos el siguiente fichero Vagrantfile que lanza una instancia de tipo m3.medium en la región escogida, le asigna una dirección IP pública y le instala rsync para utilizar el directorio compartido:

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

  config.vm.box = "dummy"

  config.ssh.keys_only = false

  config.vm.provider :aws do |aws, override|

    aws.keypair_name = "clave-aws"

    aws.ami = "ami-3291be54"

    override.ssh.username = "admin"

  end

end
```

EJERCICIO: ALMACENAMIENTO PERSISTENTE

Utiliza el plugin **vagrant-persistent-storage** para incluir almacenamiento permanente sin necesidad de utilizar parámetros de VBoxManage en bruto (del laboratorio descrito en la pagina 24 de este manual .

Utilizaremos es escenario **centos7**

Este plugin está disponible en Github y no es “oficial”:

https://github.com/kusnier/vagrant-persistent-storage

Lo instalamos de forma idéntica al caso anterior:

```
vagrant plugin install vagrant-persistent-storage
```

Y añadimos las siguientes líneas a nuestro fichero Vagrantfile (para un disco adicional de hasta 1 GiB, con sistema de ficheros ext4 y montado en el directorio /var/lib/mysql):

```
...

config.persistent_storage.enabled = true

config.persistent_storage.location = "tmp/disk.vdi"

config.persistent_storage.size = 1000

config.persistent_storage.mountname = 'mysql'

config.persistent_storage.filesystem = 'ext4'

config.persistent_storage.mountpoint = '/var/lib/mysql'

config.persistent_storage.volgroupname = 'myvolgroup'

...
```

```
$ vagrant global-status
```

Vagrantfile

```
Vagrant.configure("2") do |config|
  config.vm.box = "bento/centos-7.1"
  config.vm.network "forwarded_port", guest: 80, host: 8080
  config.persistent_storage.enabled = true
  config.persistent_storage.location = "disk.vdi"
  config.persistent_storage.size = 1000
  config.persistent_storage.mountname = 'mysql'
  config.persistent_storage.filesystem = 'ext4'
  config.persistent_storage.mountpoint = '/var/lib/mysql'
  config.persistent_storage.volgroupname = 'myvolgroup'
end
```

Podemos comprobar en la ruta de la creación de la máquina virtual la creación del disco de 1GB:

C:\HashiCorp\Vagrant\centos7

Una vez que la mv se inicia con el comando **vagrant up**, podemos ver el punto de montaje en nuestras mv de Linux **/var/lib/mysql**

$ vagrant ssh

```
vagrant@127.0.0.1's password:

Last login: Mon Apr 30 08:42:54 2018 from 10.0.2.2
```

```
[vagrant@localhost ~]$ df -h
Filesystem                Size  Used Avail Use% Mounted on
/dev/mapper/centos-root    39G  773M   38G   2% /
devtmpfs                  220M     0  220M   0% /dev
tmpfs                     229M     0  229M   0% /dev/shm
tmpfs                     229M  4.3M  225M   2% /run
tmpfs                     229M     0  229M   0% /sys/fs/cgroup
/dev/sda1                 497M  120M  378M  24% /boot
/dev/mapper/myvolgroup-mysql  961M  2.5M  893M   1% /var/lib/mysql
none                      243G  218G   25G  90% /vagrant
```