UNL VIRTUAL

Tecnicatura Universitaria en Software Libre

Administración GNU/Linux III

Anexo 1
Vagrant

Autor
Ing. Maximiliano Boscovich

FICH UNL

Anexo 1: Vagrant

Vagrant es un workflow para construir y administrar entornos de máquinas virtuales de una manera sencilla y unificada. Esta centrado en la automatización del proceso de creación y configuración de máquinas virtuales, agilizando los tiempos que estas tareas demandan. A su vez resuelve el problema de "funciona en mi maquina" que a menudo se presenta en los procesos de testeo y puesta en producción de los servicios, dado que nos permite contar con una configuración muy parecida en todos los entornos, sin tener que repetirnos en la creación de los mismos.

Pero no demos tantas vueltas y vayamos al grano.

Vagrantfile: el descriptor de la maquina virtual

Como hemos visto en la unidad 1, Vagrant basa la configuración de la MV que utilizaremos en este archivo denominado Vagrantfile. Dentro del mismo podemos definir casi todas las características que deseamos de una maquina virtual, es decir, configuración de la memoria, el nombre de la MV, cantidad de procesadores, la red, el sistema operativo, etc.

Para crear un archivo de definición vacío, debemos ejecutar

```
vagrant init
```

Esto creará en el directorio donde estamos el archivo Vagrantfile. Veamos lo que contiene...

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

# All Vagrant configuration is done below. The "2" in Vagrant.configure
# configures the configuration version (we support older styles for
# backwards compatibility). Please don't change it unless you know what
# you're doing.
Vagrant.configure("2") do |config|
 # The most common configuration options are documented and commented below.
 # For a complete reference, please see the online documentation at
 # https://docs.vagrantup.com.

 # Every Vagrant development environment requires a box. You can search for
 # boxes at https://vagrantcloud.com/search.
 config.vm.box = "base"

 # Disable automatic box update checking. If you disable this, then
 # boxes will only be checked for updates when the user runs
 # `vagrant box outdated`. This is not recommended.
 # config.vm.box_check_update = false
```

```
  # Create a forwarded port mapping which allows access to a specific port
  # within the machine from a port on the host machine. In the example below,
  # accessing "localhost:8080" will access port 80 on the guest machine.
  # config.vm.network "forwarded_port", guest: 80, host: 8080

  # Create a private network, which allows host-only access to the machine
  # using a specific IP.
  # config.vm.network "private_network", ip: "192.168.33.10"

  # Create a public network, which generally matched to bridged network.
  # Bridged networks make the machine appear as another physical device on
  # your network.
  # config.vm.network "public_network"

  # Share an additional folder to the guest VM. The first argument is
  # the path on the host to the actual folder. The second argument is
  # the path on the guest to mount the folder. And the optional third
  # argument is a set of non-required options.
  # config.vm.synced_folder "../data", "/vagrant_data"

  # Provider-specific configuration so you can fine-tune various
  # backing providers for Vagrant. These expose provider-specific options.
  # Example for VirtualBox:
  #
  # config.vm.provider "virtualbox" do |vb|
  #   # Display the VirtualBox GUI when booting the machine
  #   vb.gui = true
  #
  #   # Customize the amount of memory on the VM:
  #   vb.memory = "1024"
  # end
  #
  # View the documentation for the provider you are using for more
  # information on available options.

  # Enable provisioning with a shell script. Additional provisioners such as
  # Puppet, Chef, Ansible, Salt, and Docker are also available. Please see the
  # documentation for more information about their specific syntax and use.
  # config.vm.provision "shell", inline: <<-SHELL
  #   apt-get update
  #   apt-get install -y apache2
  # SHELL
end
```

Podemos destacar 4 cosas en este archivo:

- Tienen unas cuantas opciones bien documentadas.
- Es un script de ruby.
- Menciona algo denominado box.
- Hay algo que se llama provisioning.

Si bien en el archivo se presentan las opciones más comunes, podemos configurar casi todas las opciones que podemos definir en Virtualbox.

Lo segundo lo destacamos por una simple cuestión, debemos respetar la sintaxis de este lenguaje, y el hecho de que sea un script resulta más que interesante.

En tercer lugar están los box. Estos son el corazón de Vagrant, y los veremos a continuación.

Por último tenemos lo que se denomina provisioning (aprovisionamiento) que también lo veremos en detalle a continuación.

Vagrant boxes

En lugar de crear una maquina virtual siempre desde cero (lo que es un proceso lento y tedioso), Vagrant usa imágenes base que permiten clonar rápidamente una maquina virtual.

Estas imágenes base son conocidas como "boxes", y especifican el entorno que usara Vagrant. Para decirlo de manera simplificada, es una maquina virtual ya instalada y configurada, que podemos utilizar para empezar ya desde este punto, y no tener que instalarla. En el caso de la Unidad 1, por ejemplo, el box que utilizamos era un Debian Jessie 9.3 con todos los discos que necesitábamos para trabajar en el Ejemplo Práctico 1.

Los box pueden ser versiones de diferentes distribuciones, ya listas para utilizarse, con diferentes arquitecturas y configuraciones. Incluso podemos crear nuestros propios boxes y subirlos a la nube para que estén disponibles desde cualquier lugar, ó para otros usuarios. Esto ha hecho muy popular a Vagrant, dado hay cientos de boxes libres que se pueden utilizar. Vagrant cuenta con un repositorio de boxes, disponible en https://app.vagrantup.com/boxes/search

Vagrant en acción

Supongamos que necesitamos probar un software en un CentOS versión 7 de 64bits, y no contamos con una maquina virtual para realizar este test. Podemos buscar en la URL anterior un box con CentOS, y luego editar el archivo Vagrantfile para que haga uso del mismo

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

Vagrant.configure("2") do |config|
  config.vm.box = "centos/7"
end
```

NOTA: Por simplificación fueron quitados los comentarios.

Luego simplemente parados en el directorio donde se encuentra el archivo Vagrantfile, ejecutamos "vagrant up", y dejamos que la magia suceda

```
mboscovich@tesla:~/vagrant/maquina-virtual-01$ vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Box 'centos/7' could not be found. Attempting to find and install...
    default: Box Provider: virtualbox
    default: Box Version: >= 0
==> default: Loading metadata for box 'centos/7'
    default: URL: https://vagrantcloud.com/centos/7
==> default: Adding box 'centos/7' (v1802.01) for provider: virtualbox
    default: Downloading: https://vagrantcloud.com/centos/boxes/7/versions/1802.01/providers/virtualbox.box
==> default: Successfully added box 'centos/7' (v1802.01) for 'virtualbox'!
==> default: Importing base box 'centos/7'...
==> default: Matching MAC address for NAT networking...
==> default: Checking if box 'centos/7' is up to date...
==> default: Setting the name of the VM: maquina-virtual-01_default_1521299496587_40088
==> 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: 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: No guest additions were detected on the base box for this VM! Guest
    default: additions are required for forwarded ports, shared folders, host only
    default: networking, and more. If SSH fails on this machine, please install
    default: the guest additions and repackage the box to continue.
    default:
    default: This is not an error message; everything may continue to work properly,
    default: in which case you may ignore this message.
==> default: Rsyncing folder: /home/mboscovich/vagrant/maquina-virtual-01/ => /vagrant
```

Durante este proceso, Vagrant busco localmente si había un box llamado "centos/7", como no lo encontró lo busco en https://vagrantcloud.com/ y lo descargo. Finalmente creo en VirtualBox una maquina virtual con esta imagen base que descargo, y listo, ya contamos con un CentOS versión 7, listo para que lo usemos. Si ahora hacemos en el mismo directorio "vagrant ssh" podremos ingresar al mismo y corroborarlo. Incluso si iniciamos Virtualbox, veremos que la maquina virtual fue creada.

```
mboscovich@tesla:~/vagrant/maquina-virtual-01$ vagrant ssh
[vagrant@localhost ~]$ rpm --query centos-release
centos-release-7-4.1708.el7.centos.x86_64
```

Tan simple como eso...

Bien, si nos deslogueamos y ejecutamos "vagrant destroy" podremos eliminar dicha MV

```
mboscovich@tesla:~/vagrant/maquina-virtual-01$ 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...
mboscovich@tesla:~/vagrant/maquina-virtual-01$
```

Ahora cambiemos algunas opciones de la MV que crearemos, cambiemos el nombre, la cantidad de cpus y la memoria...también configuremos una IP para dicho host. Para esto modifiquemos el Vagrantfile de la siguiente manera

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

Vagrant.configure("2") do |config|
  config.vm.box = "centos/7"
  config.vm.network "private_network", ip: "192.168.33.10"
  config.vm.provider "virtualbox" do |vb|
     vb.memory = "1024"
     vb.cpus = 2
     vb.name = "CentOS versión 7 (Vagrant)"
  end
end
```

Si ahora ejecutamos en el directorio donde esta el vagrantfile, "vagrant up" nuevamente veremos lo siguiente

```
mboscovich@tesla:~/vagrant/maquina-virtual-01$ vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Importing base box 'centos/7'...
==> default: Matching MAC address for NAT networking...
==> default: Checking if box 'centos/7' is up to date...
==> default: Setting the name of the VM: CentOS versión 7 (Vagrant)
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
    default: Adapter 2: hostonly
==> 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: No guest additions were detected on the base box for this VM! Guest
    default: additions are required for forwarded ports, shared folders, host only
    default: networking, and more. If SSH fails on this machine, please install
    default: the guest additions and repackage the box to continue.
    default:
    default: This is not an error message; everything may continue to work properly,
    default: in which case you may ignore this message.
==> default: Configuring and enabling network interfaces...
==> default: Rsyncing folder: /home/mboscovich/vagrant/maquina-virtual-01/ => /vagrant
```

Revisando, podemos ver que:

- Vagrant ahora encontró el box "centos/7" (ya se bajo y cacheo en la ejecución anterior)
- Configuró el nombre de la máquina virtual a "Centos versión 7 (Vagrant)"
- Agrego 2 placas de red a la MV, una que utilizará el método NAT para que podamos hacer "vagrant ssh", y otra que agrega a la MV la IP 192.168.33.10.

Si vamos a virtualbox, veremos que se configuro la MV con los 2 procesadores, y 1024Mb de memoria RAM que definimos.

Jugando un poco más con Vagrant

Ahora supongamos que necesitamos probar el mismo software, pero en un Debian 9 de 64bits. Para esto, podemos modificar el Vagrantfile con lo siguiente y ejecutar vagrant up

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

Vagrant.configure("2") do |config|
  config.vm.box = "debian/jessie64"
  config.vm.network "private_network", ip: "192.168.33.11"
  config.vm.provider "virtualbox" do |vb|
    vb.memory = "1024"
    vb.cpus = 2
    vb.name = "Debian Jessie (Vagrant)"
  end
end
```

Con esto tendremos que, simplemente cambiando una línea, hemos creado una maquina virtual idéntica a la anterior, pero con un Debian Jessie (identica en las especificaciones de hardware).

Proveedores (Providers)

En Vagrant, los proveedores son los backends utilizados para crear las máquinas virtuales. El backend predeterminado es Virtualbox, pero podemos utilizar cualquiera de los siguientes:

- VirtualBox
- Docker
- VMWare
- Hyper-V
- AWS

Esto aporta una gran versatilidad, dado que podemos probar una maquina virtual en VirtualBox por ejemplo, y luego cuando ya se encuentre en el estado que deseamos, podemos hacer que corra en AWS ó Docker, simplemente modificando el vagrantfile. Imagínense correr en el entorno de desarrollo la MV en Vitualbox, en testing correrla en Docker, y luego en producción pasarla a AWS, todo desde un mismo archivo de configuración, y sin tener que cambiar casi nada, solo un par de líneas.

Aprovisionamiento (Provisioning)

Los aprovisionadores en vagrant permiten dotar de ciertas particularidades a la maquina virtual instalada, como pueden ser instalar automáticamente software y alterar su configuración entre otras cosas, todo durante el proceso de inicio de vagrant (vagrant up). Son útiles para adaptar los box (que como vimos son imágenes básicas de un S.O) a nuestras necesidades de forma completamente automática. Por supuesto que podemos usar Vagrant para crear una maquina virtual e instalar y configurar a mano su software, pero usando aprovisionadores, se automatiza este proceso y se puede repetir tantas veces sea necesario. Pensemos por ejemplo en que necesitamos 2 máquinas virtuales, una con un servidor web y otra con un motor de base de datos, podríamos usar el mismo box (por ejemplo un Debian Jessie) y luego mediante los aprovisionadores, en una maquina instalar y configurar el servidor web y en la otra el motor de bases de datos. Incluso si contamos con varios entornos (como integración, testing y producción), repetir este proceso en los 3 entornos, utilizando solo 2 recetas que correrán los aprovisionadores. Lo más importante, es que este proceso no requerirá la intervención humana, simplemente definimos que box utilizar, configuramos que aprovisionador se debe utilizar, y que debe realizar el mismo y listo, Vagrant se encargara de descargar e instalar el box, y luego "correr" el aprovisionador para adaptar dicho box a las necesidades particulares de esa máquina virtual. Incluso si estamos probando, podemos destruir esta instancia (vagrant destroy) y probar de nuevo todo el proceso ejecutando simplemente "vagrant up"...vamos, que si esto no es una maravilla no se que puede serlo :P.

Vagrant nos da la posibilidad de utilizar multiples aprovisionadores, como puede ser un siemple script de bash, a algo mas avanzado como puede ser Ansible. Entre los aprovisionadores que soporta se encuentran:

- Shell
- Ansible
- CFEngine
- Chef
- Docker
- Puppet
- Salt

Entre otros.

Usando el aprovisionador Shell

Veamos un ejemplo sencillo, al vagranfile anterior agreguemosle un aprovisionador que ejecute un script de shell que se encargue de instalar el servidor web apache: Editemos el archivo Vagrantfile para que quede de la siguiente manera

```
Vagrant.configure("2") do |config|
  config.vm.box = "debian/jessie64"
  config.vm.network "private_network", ip: "192.168.33.11"
  config.vm.provider "virtualbox" do |vb|
    vb.memory = "1024"
    vb.cpus = 2
    vb.name = "Debian Jessie (Vagrant)"
  end
  config.vm.provision "shell", inline: "sudo apt install apache2 -y"
end
```

Simplemente hemos agregado la opción

```
config.vm.provision "shell", inline: "sudo apt install apache2 -y"
```

Que le dice a vagrant que debe utilizar el aprovisionador "shell", y que este debe recibir como parámetros la orden "sudo apt install apache2 -y" (el -y es para que no nos pregunte nada y solo ejecute la instalación).

Ahora si ejecutamos un vagrant destroy y luego un vagrant up, la maquina virtual se creará con un Debian Jessie y luego en la misma se instalará apache2, todo sin tener que ingresar ni una sola vez a la misma.

```
mboscovich@tesla:~/vagrant/maquina-virtual-02$ vagrant destroy
mboscovich@tesla:~/vagrant/maquina-virtual-02$ vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Importing base box 'debian/jessie64'...
==> default: Matching MAC address for NAT networking...
==> default: Checking if box 'debian/jessie64' is up to date...
==> default: Setting the name of the VM: Debian Jessie (Vagrant)
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
    default: Adapter 2: hostonly
==> 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: No guest additions were detected on the base box for this VM! Guest
    default: additions are required for forwarded ports, shared folders, host only
    default: networking, and more. If SSH fails on this machine, please install
    default: the guest additions and repackage the box to continue.
    default:
    default: This is not an error message; everything may continue to work properly,
    default: in which case you may ignore this message.
==> default: Configuring and enabling network interfaces...
==> default: Installing rsync to the VM...
==> default: Rsyncing folder: /home/mboscovich/vagrant/maquina-virtual-02/ => /vagrant
==> default: Running provisioner: shell...
    default: Running: inline script
==> default: stdin: is not a tty
==> default:
==> default: WARNING: apt does not have a stable CLI interface yet. Use with caution in scripts.
==> default: Reading package lists...
==> default: Building dependency tree...
==> default:
==> default: Reading state information...
==> default: The following extra packages will be installed:
==> default:   apache2-bin apache2-data apache2-utils libapr1 libaprutil1
==> default:   libaprutil1-dbd-sqlite3 libaprutil1-ldap liblua5.1-0 ssl-cert
==> default: Suggested packages:
==> default:   apache2-doc apache2-suexec-pristine apache2-suexec-custom openssl-blacklist
==> default: The following NEW packages will be installed:
==> default:   apache2 apache2-bin apache2-data apache2-utils libapr1 libaprutil1
==> default:   libaprutil1-dbd-sqlite3 libaprutil1-ldap liblua5.1-0 ssl-cert
==> default: 0 upgraded, 10 newly installed, 0 to remove and 27 not upgraded.
==> default: Need to get 1,951 kB of archives.
==> default: After this operation, 6,377 kB of additional disk space will be used.
==> default: Get:1 http://httpredir.debian.org/debian/ jessie/main libapr1 amd64 1.5.1-3 [95.3 kB]
==> default: Get:2 http://httpredir.debian.org/debian/ jessie/main libaprutil1 amd64 1.5.4-1 [86.2 kB]
==> default: Get:3 http://httpredir.debian.org/debian/ jessie/main libaprutil1-dbd-sqlite3 amd64 1.5.4-1 [19.1 kB]
==> default: Get:4 http://httpredir.debian.org/debian/ jessie/main libaprutil1-ldap amd64 1.5.4-1 [17.2 kB]
==> default: Get:5 http://httpredir.debian.org/debian/ jessie/main liblua5.1-0 amd64 5.1.5-7.1 [108 kB]
==> default: Get:6 http://httpredir.debian.org/debian/ jessie/main apache2-bin amd64 2.4.10-10+deb8u11 [1,038 kB]
==> default: Get:7 http://httpredir.debian.org/debian/ jessie/main apache2-utils amd64 2.4.10-10+deb8u11 [195 kB]
==> default: Get:8 http://httpredir.debian.org/debian/ jessie/main apache2-data all 2.4.10-10+deb8u11 [163 kB]
==> default: Get:9 http://httpredir.debian.org/debian/ jessie/main apache2 amd64 2.4.10-10+deb8u11 [208 kB]
==> default: Get:10 http://httpredir.debian.org/debian/ jessie/main ssl-cert all 1.0.35 [20.9 kB]
==> default: dpkg-preconfigure: unable to re-open stdin: No such file or directory
==> default: Fetched 1,951 kB in 6s (293 kB/s)
==> default: Selecting previously unselected package libapr1:amd64.
(Reading database ... 28340 files and directories currently installed.)
==> default: Preparing to unpack .../libapr1_1.5.1-3_amd64.deb ...
==> default: Unpacking libapr1:amd64 (1.5.1-3) ...
==> default: Selecting previously unselected package libaprutil1:amd64.
==> default: Preparing to unpack .../libaprutil1_1.5.4-1_amd64.deb ...
==> default: Unpacking libaprutil1:amd64 (1.5.4-1) ...
==> default: Selecting previously unselected package libaprutil1-dbd-sqlite3:amd64.
==> default: Preparing to unpack .../libaprutil1-dbd-sqlite3_1.5.4-1_amd64.deb ...
==> default: Unpacking libaprutil1-dbd-sqlite3:amd64 (1.5.4-1) ...
==> default: Selecting previously unselected package libaprutil1-ldap:amd64.
==> default: Preparing to unpack .../libaprutil1-ldap_1.5.4-1_amd64.deb ...
==> default: Unpacking libaprutil1-ldap:amd64 (1.5.4-1) ...
==> default: Selecting previously unselected package liblua5.1-0:amd64.
==> default: Preparing to unpack .../liblua5.1-0_5.1.5-7.1_amd64.deb ...
```

```
==> default: Unpacking liblua5.1-0:amd64 (5.1.5-7.1) ...
==> default: Selecting previously unselected package apache2-bin.
==> default: Preparing to unpack .../apache2-bin_2.4.10-10+deb8u11_amd64.deb ...
==> default: Unpacking apache2-bin (2.4.10-10+deb8u11) ...
==> default: Selecting previously unselected package apache2-utils.
==> default: Preparing to unpack .../apache2-utils_2.4.10-10+deb8u11_amd64.deb ...
==> default: Unpacking apache2-utils (2.4.10-10+deb8u11) ...
==> default: Selecting previously unselected package apache2-data.
==> default: Preparing to unpack .../apache2-data_2.4.10-10+deb8u11_all.deb ...
==> default: Unpacking apache2-data (2.4.10-10+deb8u11) ...
==> default: Selecting previously unselected package apache2.
==> default: Preparing to unpack .../apache2_2.4.10-10+deb8u11_amd64.deb ...
==> default: Unpacking apache2 (2.4.10-10+deb8u11) ...
==> default: Selecting previously unselected package ssl-cert.
==> default: Preparing to unpack .../ssl-cert_1.0.35_all.deb ...
==> default: Unpacking ssl-cert (1.0.35) ...
==> default: Processing triggers for man-db (2.7.0.2-5) ...
==> default: Processing triggers for systemd (215-17+deb8u7) ...
==> default: Setting up libapr1:amd64 (1.5.1-3) ...
==> default: Setting up libaprutil1:amd64 (1.5.4-1) ...
==> default: Setting up libaprutil1-dbd-sqlite3:amd64 (1.5.4-1) ...
==> default: Setting up libaprutil1-ldap:amd64 (1.5.4-1) ...
==> default: Setting up liblua5.1-0:amd64 (5.1.5-7.1) ...
==> default: Setting up apache2-bin (2.4.10-10+deb8u11) ...
==> default: Setting up apache2-utils (2.4.10-10+deb8u11) ...
==> default: Setting up apache2-data (2.4.10-10+deb8u11) ...
==> default: Setting up apache2 (2.4.10-10+deb8u11) ...
==> default: Enabling module mpm_event.
==> default: Enabling module authz_core.
==> default: Enabling module authz_host.
==> default: Enabling module authn_core.
==> default: Enabling module auth_basic.
==> default: Enabling module access_compat.
==> default: Enabling module authn_file.
==> default: Enabling module authz_user.
==> default: Enabling module alias.
==> default: Enabling module dir.
==> default: Enabling module autoindex.
==> default: Enabling module env.
==> default: Enabling module mime.
==> default: Enabling module negotiation.
==> default: Enabling module setenvif.
==> default: Enabling module filter.
==> default: Enabling module deflate.
==> default: Enabling module status.
==> default: Enabling module reqtimeout.
==> default: Enabling conf charset.
==> default: Enabling conf localized-error-pages.
==> default: Enabling conf other-vhosts-access-log.
==> default: Enabling conf security.
==> default: Enabling conf serve-cgi-bin.
==> default: Enabling site 000-default.
==> default: Setting up ssl-cert (1.0.35) ...
==> default: Processing triggers for libc-bin (2.19-18+deb8u10) ...
==> default: Processing triggers for systemd (215-17+deb8u7) ...

==> default: Machine 'default' has a post `vagrant up` message. This is a message
==> default: from the creator of the Vagrantfile, and not from Vagrant itself:
==> default:
==> default: Vanilla Debian box. See https://app.vagrantup.com/debian for help and bug reports
```

En caso de que no queramos destruir la maquina virtual, simplemente podemos ejecutar en la carpeta que contiene el vagrantfile lo siguiente

```
mboscovich@tesla:~/vagrant/maquina-virtual-02$ vagrant provision
```

Usando el aprovisionador Ansible

Pongamoslé mas pimienta a la cosa, y ahora utilicemos como aprovisionador Ansible para hacer lo anterior de instalar apache2. Pero antes que nada, primero debemos asegurarnos de tener instalado Ansible

```
mboscovich@tesla:~/vagrant/maquina-virtual-02$ sudo apt install ansible
```

Ahora creemos en el directorio donde esta el Vagrantfile, un archivo llamado playbook.yml con lo siguiente

```
---
- hosts: all

      tasks:
      - name: Instalar Apache
            become: true
            apt:
                  name: apache2
                  state: present
                  update-cache: yes
```

Y modifiquemos el vagrantfile para que lo invoque

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

Vagrant.configure("2") do |config|
  config.vm.box = "debian/jessie64"
  config.vm.network "private_network", ip: "192.168.33.11"
  config.vm.provider "virtualbox" do |vb|
    vb.memory = "1024"
    vb.cpus = 2
    vb.name = "Debian Jessie (Vagrant)"
  end
  config.vm.provision "ansible" do |ansible|
    ansible.playbook = "playbook.yml"
  end
end
```

Si lo ejecutamos con vagrant provision, podemos ver lo siguientes

```
mboscovich@tesla:~/vagrant/maquina-virtual-02$ vagrant provision
==> default: Running provisioner: ansible...
    default: Running ansible-playbook...

PLAY [all] *********************************************************************

TASK [Gathering Facts] *********************************************************
ok: [default]
```

```
TASK [Instalar Apache] ***************************************************
ok: [default]

PLAY RECAP *********************************************************************
default                    : ok=2    changed=0    unreachable=0    failed=0
```

Como verán podemos hacer casi cualquier cosa con la integración de estas dos herramientas.

Pongamos más pimienta aún

Si recordamos lo que vimos en la unidad 6, y ahora le agregamos como complemento Vagrant, podemos tener en cuestión de unos pocos minutos un wordpress instalado y configurado, listo para usarse. Recordemos el playbook que definimos en la unidad 6

```
---
- hosts: all
  vars:
    wordpress_db_name: 'wordpress_db'
    wordpress_db_user: 'wordpress'
    wordpress_db_password: 'superpassword'
    wordpress_db_host: 'localhost'
    wordpress_content_dir: '/var/lib/wordpress/wp-content'

  tasks:
  - name: Instalar Worpress y sus dependencias
    become: true
    apt:
      name: "{{ item }}"
      state: present
      update-cache: yes
    with_items:
      - wordpress
      - curl
      - apache2
      - mysql-server
      - python-mysqldb

  - name: Copiar configuración de Wordpress para apache2
    become: true
    copy:
      src: wp.conf
      dest: /etc/apache2/sites-available/wp.conf
      owner: root
      group: root
      mode: 0644

  - name: Habilitar sitio de Wordpress (crear enlace simbólico)
    become: true
    file:
      src: /etc/apache2/sites-available/wp.conf
      dest: /etc/apache2/sites-enabled/wp.conf
      owner: root
      group: root
      state: link
    notify:
```

```
    - Recargar apache

- name: Configurar wordpress
  become: true
  template:
    src: wordpress_config.j2
    dest:  /etc/wordpress/config-nodo1.intranet.php
    owner: www-data
    group: www-data
    mode: 0644

- name: Crear usuario wordpress
  become: true
  mysql_user:
    name: '{{ wordpress_db_user}}'
    password: '{{ wordpress_db_password }}'
    priv: '*.*:ALL'
    state: present

- name: Crear base de datos de wordpress
  become: true
  mysql_db:
    name: '{{ wordpress_db_name }}'
    login_user: '{{ wordpress_db_user }}'
    login_password: '{{ wordpress_db_password }}'
    login_host: 'localhost'
    state: present

handlers:
  - name: Recargar apache
    become: true
    service:
      name: apache2
      state: reloaded
```

Si guardamos este archivo con el nombre playbook.yml dentro del directorio que contiene el vagrantfile, y también creamos los siguientes 2 archivos que necesita el playbook (wp.conf y wordpress_config.j2) en el mismo directorio,

Archivo wp.conf

```
Alias /wp/wp-content /var/lib/wordpress/wp-content
Alias /wp /usr/share/wordpress
<Directory /usr/share/wordpress>
    Options FollowSymLinks
    AllowOverride Limit Options FileInfo
    DirectoryIndex index.php
    Require all granted
</Directory>
<Directory /var/lib/wordpress/wp-content>
    Options FollowSymLinks
    Require all granted
</Directory>
```

Archivo wordpress_config.j2

```
<?php
define('DB_NAME', '{{ wordpress_db_name }}');
define('DB_USER', '{{ wordpress_db_user }}');
define('DB_PASSWORD', '{{ wordpress_db_password }}');
define('DB_HOST', '{{ wordpress_db_host }}');
define('WP_CONTENT_DIR', '{{ wordpress_content_dir }}');
?>
```

luego ejecutando vagrant provision podemos convertir nuestra maquina virtual en un wordpress completamente configurado

```
mboscovich@tesla:~/vagrant/maquina-virtual-02$ vagrant provision
==> default: Running provisioner: ansible...
    default: Running ansible-playbook...

PLAY [all] *********************************************************************

TASK [Gathering Facts] *********************************************************
ok: [default]

TASK [Instalar Worpress y sus dependencias] ************************************
ok: [default] => (item=[u'wordpress', u'curl', u'apache2', u'mysql-server', u'python-mysqldb'])

TASK [Copiar configuración de Wordpress para apache2] **************************
changed: [default]

TASK [Habilitar sitio de Wordpress (crear enlace simbólico)] *******************
changed: [default]

TASK [Configurar wordpress] ****************************************************
changed: [default]

TASK [Crear usuario wordpress] *************************************************
changed: [default]

TASK [Crear base de datos de wordpress] ****************************************
changed: [default]

RUNNING HANDLER [Recargar apache] **********************************************
changed: [default]

PLAY RECAP *********************************************************************
default                    : ok=8    changed=6    unreachable=0    failed=0
```

Como verán, integrar Vagrant con un aprovisionador como Ansible nos brinda mucha flexibilidad y versatilidad, y puede permitirnos montar un entorno completamente funcional en un abrir y cerrar de ojos.

Probemos por ejemplo ahora ejecutar vagrant destroy y luego vagrant up

```
mboscovich@tesla:~/vagrant/maquina-virtual-02$ 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...
mboscovich@tesla:~/vagrant/maquina-virtual-02$ vagrant up
```

```
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Importing base box 'debian/jessie64'...
==> default: Matching MAC address for NAT networking...
==> default: Checking if box 'debian/jessie64' is up to date...
==> default: Setting the name of the VM: Debian Jessie (Vagrant)
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
    default: Adapter 2: hostonly
==> 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: No guest additions were detected on the base box for this VM! Guest
    default: additions are required for forwarded ports, shared folders, host only
    default: networking, and more. If SSH fails on this machine, please install
    default: the guest additions and repackage the box to continue.
    default:
    default: This is not an error message; everything may continue to work properly,
    default: in which case you may ignore this message.
==> default: Configuring and enabling network interfaces...
==> default: Installing rsync to the VM...
==> default: Rsyncing folder: /home/mboscovich/vagrant/maquina-virtual-02/ => /vagrant
==> default: Running provisioner: ansible...
    default: Running ansible-playbook...

PLAY [all] *********************************************************************

TASK [Gathering Facts] *********************************************************
ok: [default]

TASK [Instalar Worpress y sus dependencias] ************************************
changed: [default] => (item=[u'wordpress', u'curl', u'apache2', u'mysql-server', u'python-mysqldb'])

TASK [Copiar configuración de Wordpress para apache2] **************************
changed: [default]

TASK [Habilitar sitio de Wordpress (crear enlace simbólico)] *******************
changed: [default]

TASK [Configurar wordpress] ****************************************************
changed: [default]

TASK [Crear usuario wordpress] *************************************************
changed: [default]

TASK [Crear base de datos de wordpress] ****************************************
changed: [default]

RUNNING HANDLER [Recargar apache] **********************************************
changed: [default]
```

```
PLAY RECAP *********************************************************************
default                    : ok=8    changed=7    unreachable=0    failed=0


==> default: Machine 'default' has a post `vagrant up` message. This is a message
==> default: from the creator of the Vagrantfile, and not from Vagrant itself:
==> default:
==> default: Vanilla Debian box. See https://app.vagrantup.com/debian for help and bug reports
```

Imaginen la potencia de lo anterior, simplemente podemos llevar estos pocos archivos (el vagrantfile, playbook.yml, wp.conf y wordpress_config.j2) a cualquier lugar y levantar un wordpress completamente funcional. O se lo podemos pasar a un usuario para que empiece a testearlo o utilizarlo... lo único que deberá tener instalado en su PC es Virtualbox, Vagrant y Ansible, y estos 4 archivos :). Espero les pueda ser de utilidad.