

Título

Pruebas con Título

Daniel Moreno

May 6, 2019

IES Francesc de Borja Moll

Vagrant es una herramienta que permite crear entornos de desarrollo basados en máquinas virtuales.

La idea es crear entornos que sean lo más parecido o idénticos a los servidores de producción.

El objetivo es solucionar el problema de “en mi máquina funciona”.

Si definimos un entorno determinado, lo podemos ejecutar en cualquier lugar, sabiendo que cada vez que implemente en un sitio diferente se crearán las mismas máquinas y funcionarán de la misma forma.

Características de Vagrant:

- Agrega una capa por encima del software de virtualización (Virtualbox, VMware, etc).
- Está escrito en Ruby
- Inicialmente se construyó para ser utilizado con VirtualBox como hipervisor
- Actualmente soporta muchos más.

Vagrant permite crear entornos de desarrollo:

- Reproducibles
- Portables
- Ligeros

Para poner en marcha Vagrant necesitaremos, por lo menos:

- Descargar e instalar VirtualBox (si es el hipervisor elegido).
- Descargar e instalar Vagrant
- Descargar la box de vagrant que queramos

Una box es un archivo (tareado y gzipeado) parecida a una imagen de máquina virtual, lista para ejecutarse (sin necesidad de instalación). Las podemos:

- Descargar de diferentes proveedores
- Crear nosotros manualmente

Cada box contiene:

- Un Vagrantfile
- Una imagen de máquina virtual (vmdk)
- Un archivo OVF que define el hardware virtual del box
- Un archivo JSON que define que proveedor trabaja con la caja

Las boxes se pueden crear:

- Utilizando herramientas (packer.io, imagefactory)
- Manualmente a través del comando `vagrant package`.

Cosas que puedes hacer con Vagrant

Redactando correctamente un archivo de configuración podemos:

- Arrancar y parar máquinas
- Customizar las MV con tantos CPU cores, memoria como queramos
- Conectar por SSH a las diferentes máquinas con `vagrant ssh`
- Aprovisionar las máquinas con todo lo que se necesite, utilizando Puppet, chef, ansible, shell, etc.

La configuración del entorno que queremos implementar se realiza en un archivo de texto plano escrito en Ruby llamado `Vagrantfile` (similar a los `Dockerfiles`).

A través de este archivo podemos definir el entorno que queramos, y podemos:

- Configurar máquinas virtuales

Version utilizada

Para conocer la versión de Vagrant instalada:

```
vagrant version
```

Descargando boxes

Podemos consultar las boxes que hay publicadas en:

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

Para descargar una box de una máquina virtual concreta:

```
vagrant box ubuntu/precise64
```

Podemos listar las boxes que tenemos descargadas. Si una box no está descargada pero hacemos un vagrant up se descargará la primera vez.

```
vagrant box list
```

Inicializando configuraciones (Vagrantfile)

Creamos una carpeta para guardar los archivos de configuración, y una vez dentro inicializamos vagrant en esta carpeta:

```
vagrant init
```

Nos ha creado el archivo Vagrantfile que procederemos a adaptar a nuestras necesidades. Para hacer un init no tiene que existir ningún Vagrantfile.

Una vez definido el vagrantfile ya podemos levantar la/s MV:

```
vagrant up
```

Podemos ver qué MV tenemos levantadas con:

```
vagrant status
```

Conectando con las máquinas virtuales

Podemos conectar a la máquina con la un comando SSH de vagrant. Otra opción es desocultar la MV (que habrá arrancado en Virtualbox en modo oculto), o bien tirar de putty.

```
vagrant ssh
```

Mágicamente se nos meterá por ssh en la MV, sin pedir contraseña. Una vez dentro podemos hacer lo que queramos.

Si solo tenemos una MV levantada, se conectará a esta. Si tenemos más de una, pedirá el nombre de la MV.

Salimos con

```
exit
```

Parando máquinas

Para cerrar las máquinas virtuales por las buenas:

```
vagrant halt
```

Las podemos volver a levantar con un up.

Para destruir la máquina virtual (veremos que ya no aparece en Virtualbox)

```
vagrant destroy
```

Todas las configuraciones se guardan en una carpeta oculta llamada `.vagrant`. Estas configuraciones se pueden subir a github y llevar un control de versiones, al ser la configuración guardada en archivos de texto plano.

Las boxes se meten en la carpeta que tengamos definida en Virtualbox para almacenar las MV.

Aprovisionamiento

Cuando preparamos un entorno de desarrollo, se suele hacer de forma manual. Se ejecutan los comandos necesarios en el SO guest para personalizar nuestro entorno.

Esto lo podemos automatizar, almacenando los comandos a realizar en scripts especiales.

Estos scripts son los que se ejecutarán automáticamente al levantar las MV.

Este proceso de instalar y configurar software dentro del SO guest automáticamente se conoce como `aprovisionamiento`.

Existen diferentes herramientas para ello:

- Shell scripts
- Puppet
- Ansible
- Chef

Ejemplos de aprovisionamiento

shell

To-do ##### script To-do ##### ansible To-do

Proyectos en este repositorio

vagrant-ansible

La idea es aprovisionar máquinas virtuales a través de un playbook de ansible.

vagrant-mult

Automatizar el aprovisionamiento de varias MV en un único Vagrantfile

vagrant-wordpress

Automatizar y aprovisionar una MV con wordpress