Introducción a Vagrant

Grado en Ingeniería Informática

Pablo García-Sánchez

Departamento de Ingeniería Informática
Universidad de Cádiz
Introducción a Vagrant

Curso 2016 – 2017

Indice

1. Introducción
2. Instalación de Vagrant
3. Usar Vagrant
4. Terminar con Vagrant

Sección 1 | Introducción

¿Qué es Vagrant?

Vagrant es una herramienta para construir y gestionar entornos de máquinas virtuales en un único flujo de trabajo. Con un flujo de trabajo fácil de usar y un enfoque en la automatización, Vagrant reduce el tiempo de configuración del entorno de desarrollo, aumenta la paridad de la producción y hace que los "funcionaba en mi máquina"sean una reliquia del pasado.

¿Por qué Vagrant?

- Vagrant proporciona entornos de trabajo fáciles de configurar, reproducibles y portátiles, construidos sobre tecnología estándar del sector y controlados por un único flujo de trabajo coherente para ayudar a maximizar la productividad y la flexibilidad.
- Las máquinas se suministran sobre VirtualBox, VMware, AWS o cualquier otro proveedor. Herramientas de aprovisionamiento estándar del sector, como scripts shell, Chef o Puppet, pueden instalar y configurar automáticamente el software en la máquina virtual.

Para desarrolladores

- Vagrant aislará las dependencias y su configuración dentro de un único entorno desechable y consistente, sin sacrificar ninguna de las herramientas con las que está acostumbrado a trabajar (editores, navegadores, depuradores, etc.).
- Una vez que alguien crea un único archivo Vagrant, sólo tiene que usar `vagrant up` y todo estará instalado y configurado para que se pueda trabajar.
- Otros miembros del equipo crean sus entornos de desarrollo a partir de la misma configuración, por lo que tanto si se trabaja en Linux, Mac OS X o Windows, todos los miembros de su equipo ejecutan código en el mismo entorno, con las mismas dependencias y configurados de la misma forma.

Para DevOps

- Vagrant ofrece un entorno desechable y un flujo de trabajo consistente para desarrollar y probar scripts de gestión de infraestructura.
- Se puede probar rápidamente cosas como scripts de shell, cookbooks Chef, módulos Puppet, y más utilizando virtualización local como VirtualBox o VMware.
- Luego, con la misma configuración, puede probar estos scripts en nubes remotas como AWS o RackSpace con el mismo flujo de trabajo.
- Ahorra el uso de scripts personalizados para reciclar instancias de EC2, y hacer malabares con las instrucciones de SSH en varias máquinas.

Vagrant versus herramientas CLI

- Los software de virtualización como VirtualBox y VMware vienen con utilidades de línea de comandos para administrar el ciclo de vida de las máquinas en su plataforma. Muchas personas utilizan estas utilidades para escribir su propia automatización. Vagrant utiliza muchas de estas utilidades internamente.
- Sin embargo Vagrant se construye encima de estas utilidades de varias maneras, a la vez que proporciona un flujo de trabajo consistente. Vagrant soporta múltiples tipos de carpetas sincronizadas, múltiples proveedores para configurar la máquina, configuración SSH automática, creación de túneles HTTP en su entorno de desarrollo, y más. Todos ellos pueden configurarse utilizando un único y sencillo archivo de configuración.

Vagrant versus herramientas CLI (y 2)

- Las utilidades de línea de comandos proporcionadas por el software de virtualización a menudo cambian cada versión o tienen errores sutiles con soluciones provisionales. Vagrant detecta automáticamente la versión, utiliza los flags correctos y puede solucionar problemas conocidos. Así que si tu estás usando una versión de VirtualBox y un compañero de trabajo está usando una versión diferente, Vagrant seguirá trabajando consistentemente.
- Para flujos de trabajo altamente específicos que no cambian con frecuencia, puede ser beneficioso mantener scripts personalizados. Vagrant está dirigido a la creación de entornos de desarrollo, pero algunos usuarios avanzados todavía utilizan las herramientas CLI inferiores para hacer otras cosas manuales.

Vagrant versus Docker

- Vagrant es una herramienta enfocada en proporcionar un flujo de trabajo consistente en el entorno de desarrollo a través de múltiples sistemas operativos. Docker es una gestión de contenedores que puede ejecutar software de forma consistente siempre y cuando exista un sistema de contenedorización.
- Los contenedores son generalmente más ligeros que las máquinas virtuales, por lo que el arranque y la parada de los contenedores es extremadamente rápida. Docker utiliza la funcionalidad nativa de contenedorización en macOS, Linux y Windows.
- Actualmente, Docker carece de soporte para ciertos sistemas operativos (como BSD). Si la implementación de destino es uno de estos sistemas operativos, Docker no proporcionará la misma paridad de producción que una herramienta como Vagrant. Vagrant permite ejecutar un entorno de desarrollo Windows en Mac o Linux.

Vagrant versus Docker (y 2)

- Para entornos pesados de microservicio, Docker puede ser atractivo porque puede iniciar fácilmente una sola Docker VM y muchos contenedores por encima de ella muy rápidamente. Vagrant también puede hacer esto con el proveedor de Docker. Un beneficio principal para Vagrant es un flujo de trabajo consistente, pero hay muchos casos en los que un flujo de trabajo de Docker puro tiene más sentido.
- Tanto Vagrant como Docker tienen una vasta biblioteca de "imágenes" o "cajas" realizadas por la comunidad para elegir.

Sección 2 | Instalación de Vagrant

Instalar Vagrant

- `https://www.vagrantup.com/downloads.html`
- Es necesario tener instalado git y VirtualBox

Init y up

```
$ vagrant init hashicorp/precise64
$ vagrant up
```

- Se crea un fichero Vagrantfile
- El resto de boxes se puede consultar en `https://app.vagrantup.com/boxes/search`.
- Si se abre VirtualBox puede verse en funcionamiento.

Sección 3 | Usar Vagrant

Conexión por SSH

```
vagrant ssh
```

Se puede comprobar la carpeta compartida.

Aprovisionamiento

bootstrap.sh

```
#!/usr/bin/env bash

apt-get update
apt-get install -y apache2
if ! [ -L /var/www ]; then
  rm -rf /var/www
  ln -fs /vagrant /var/www
fi
```

Actualizar el fichero Vagrant

```
Vagrant.configure("2") do |config|
  config.vm.box = "hashicorp/precise64"
  config.vm.provision :shell, path: "bootstrap.sh"
end
```

Recargamos

```
$ vagrant reload --provision
$ vagrant ssh
...
vagrant@precise64:~$ wget -qO- 127.0.0.1
```

Actualizar puertos

```
Vagrant.configure("2") do |config|
  config.vm.box = "hashicorp/precise64"
  config.vm.provision :shell, path: "bootstrap.sh"
  config.vm.network :forwarded_port, guest: 80, host: 4567
end
```

Probar `http://127.0.0.1:4567`

Sección 4 | Terminar con Vagrant

Desmantelamos

- `vagrant suspend` guardará el estado de funcionamiento actual de la máquina y la detendrá. Se reanudará desde donde lo dejó usando `vagrant up`.
  - Ventaja: Tarda 5 a 10 segundos.
  - Inconveniente: la máquina virtual sigue consumiendo espacio de disco y requiere aún más espacio de disco para almacenar todo el estado de la RAM.
- `vagrant halt` apagará el sistema operativo huésped y apagará la máquina.
  - Ventaja: apagará limpiamente la máquina, preservando el contenido del disco y permitiendo que se inicie de nuevo de forma limpia.
  - Inconveniente: tiempo extra para arrancar desde un arranque en frío, y la máquina huésped todavía consume espacio en disco.

Desmantelamos (y 2)

- `vagrant destroy` Detiene la máquina huésped, la apaga y elimina todos los discos duros huéspedes.
  - Ventaja: no queda ningún residuo en su máquina. El espacio en disco y la RAM consumidos por el equipo invitado se recuperan y el equipo host se deja limpio.
  - Inconveniente: al hacer vagrant up tardará un poco más de tiempo ya que tiene que reimportar la máquina y reabastecerla.
- Ojo: No elimina realmente el archivo box descargado. Para eliminarlo completamente, utilizar el comando `vagrant box remove`.

Otros proveedores

```
$ vagrant up --provider=vmware_fusion
$ vagrant up --provider=aws
```

Referencias

- https://www.vagrantup.com/intro/getting-started/networking.html
- https://app.vagrantup.com/boxes/search