Vagrant

Índice

1. ¿Qué es Vagrant?

Vagrant es un software que se interpone entre nuestro de poveedor de máquinas virtuales y nostros, permitiendonos, a través de un fichero de configuración de texto plano, crear entornos de desarrollo y/o de pruebas por medio de máquinas virtuales de forma sencilla y declarativa.

Vagrant está basado en el lenguaje ***Ruby*** que es de alto nivel y muy sencillo de entender.

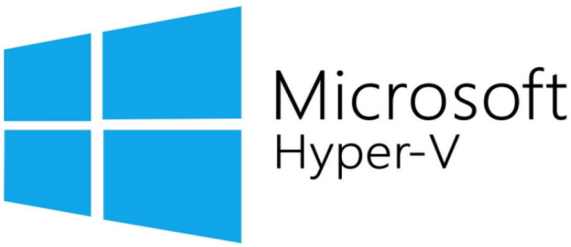
Lo que hacemos con **vagrant** no es crear una máquina virtual desde cero, sino que le pasamos un fichero de configuración indicándole que cree una o varias máquinas virtuales junto con las

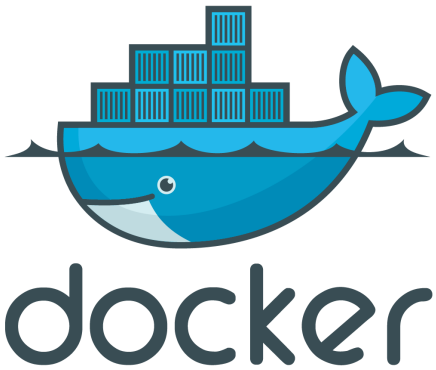
configuraciones que nosotros querramos, por ejemplo una máquina virtual con 4 GB de memória ram, 2 cpus, que esté conectada a 2 redes virtuales, y que al crear la máquina virtual ejecute un scritp que contiene la instalación de diferentes servicios.

Una cosa que hay que aclarar es que **vagrant** no es un proveedor de máquinas virtuales, de hecho **vagrant** necesita proveedores de máquinas virtuales.

1.1. Proveedores

Vagrant actualmente tiene compatibilidad con distintos proveedores, los cuales son: Virtualbox, Hyper-V, docker, livbirt y VMware





1.2. Vagrant's Workflow

El funcionamiento de vagrants (workflow) se divide en 2, los cuales son el ámbito y las boxes ("imagenes")

- **environment workflow:**
  - Vagrantfile:

    Es el fichero donde indicamos qué imagen (Box) usaremos para la máquina virtual y que configuración usaremos, se puede crear haciendo un `vagrant init` o creando el archivo manualmente

  - vagrant up

    Con `vagrant up` lo que conseguimos es *"levantar"* la máquina virtual una vez tenemos el archivo **Vagrantfile** con dicha configuración especificada.

- vagrant halt

  Con `vagrant halt` lo que hacemos es parar la máquina que está activa.

- vagrant ssh

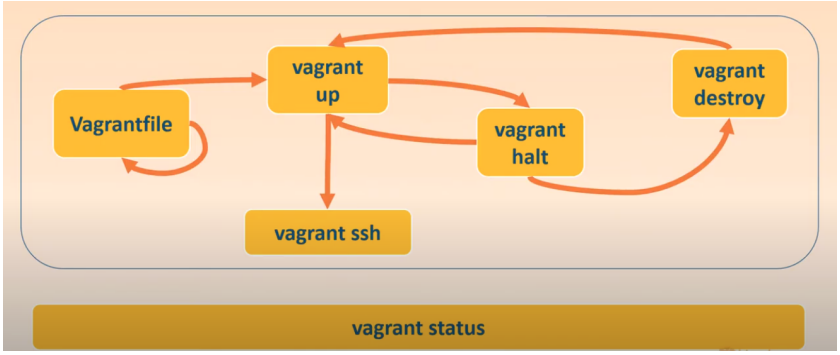
  Con `vagrant ssh` lo que hacemos es conectarnos por medio de ssh a la máquina virtual para así poder administrarla desde dentro.

- vagrant destroy

  Esta opción lo que hace, cuando ya no se necesite este ámbito que hemos creado con **Vagrant**, lo elimina junto con las configuraciones e instalaciones que hayamos hecho en este.
  No borra el fichero Vagrantfile

- vagrant status

  Está opción lo que nos permite es monitorizar la máquina o ámbito con el que estemos trabajando



- **box workflow:**

  - vagrant box add:

    Está subopción sirve para añadir un box a vagrant, un box, entre otras elementos, está formado por una imagen y un archivo de cofiguración.

- vagrant box update:

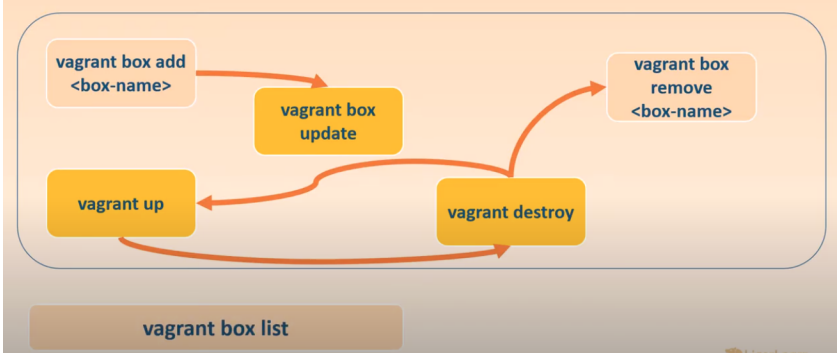
  Esta subopción sirve para actualizar un box que ya tengamos en nuestra máquina.

- vagrant box remove:

  Esta subopción sirve para borrar boxes que ya tengamos en nuestra máquina.

- Vagrant box list:

  Esta subopción sirve para listar los boxes que tenemos descargados.

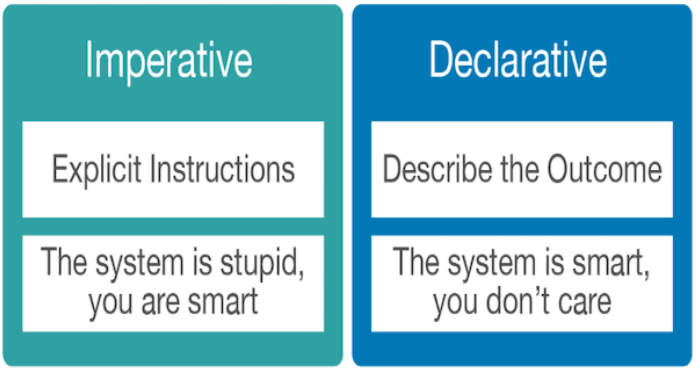


2. ¿Por qué Vagrant?

Las principales razones del por qué usar **vagrant** son:

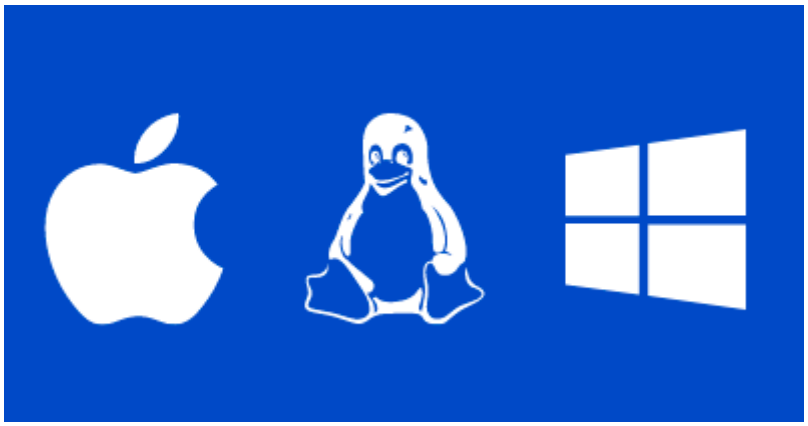
- **Es declarativo:**

  Una configuración declarativa quiere decir que nosotros le indicamos el objetivo al que queremos llegar, por ejemplo como en el apartado anterior le indicamos a vagrant que queremos que lance una máquina virtual con un determinado hardware y configuración. En comparación con una declaración imperativa que viene siendo indicarle paso por paso lo que queremos hacer.



- **Es portable:**

  Con que es portable se refiere a que podemos compartir dicho fichero de configuración con otras personas, estás personas pueden trabajar en diferentes sistemas operativos o en diferentes versiones del sistema operativo.
  Para **Vagrant** esto no es problema ya que es *cross-platform*, es decir, crea un ámbito de trabajo único para cada una de las personas gracias al uso de máquina virtualbes.

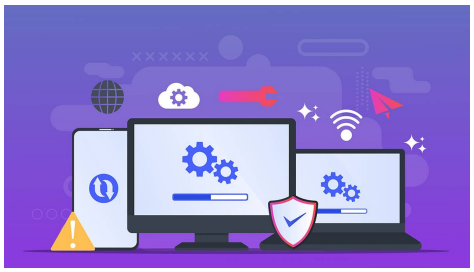


2.1. Para desarrolladores

Si eres un desarrollador o una dessarrolladora **vagrant** puede aislar las dependencias y sus respectivas configuraciones en un único ambiente de trabajo desechable por lo tanto no se sacrificará ninguna de las herramientas con las que estas acostumbrado/a a usar, ya sean editores, navegadores, debuggers, etc.

En caso de ser un grupo de desarroladores, una vez alguno de ellos haya creado un **Vagrantfile**, solo hace falta hacer `vagrant up` y todo estará instalado, configurado y listo para trabajar, en caso de que los otros miembros trabajen en otros sistemas operativos no resulta ningún problema ya que al final todos estarán usando el mismo ámbito de **Vagrant** y nos quitamos el problema de *"it works on my machine"*.

2.2. Para administradores



Vagrant te da un ámbito desechable y un workflow consistente para crear escenarios de pruebas en los cuales puedes probar scripts para la gestión de infraestructuras.

Por lo tanto puedes hacer pruebas fácil y rápidamente como *shell scripts, chef cookbooks, puppet modules* y más usando un proveedor de máquinas virtuales locales como VirtualBox o VMware, este proceso también se puede hacer en servidores remotos de la nube como AWS, usando la misma configuración y con el mismo workflow.

2.3. Para diseñadores

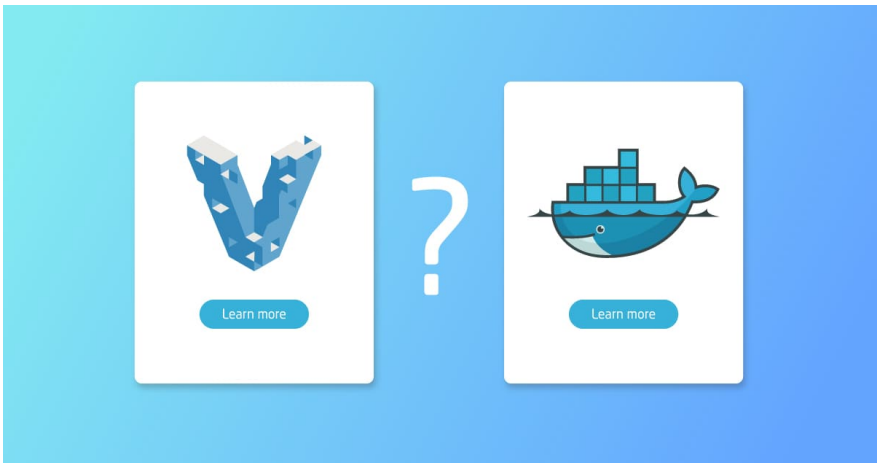
Una vez un desarroladaor haya configurado y preparado todo para la aplicación que necesite el diseñador, este no se tendrá que preocupar nunca más de configurar nada y se podrá centrar solo en desarrolar.

2.4. Para todo el mundo

Al final **Vagrant** está diseñado para crear ámbientes de virtualización de la manera más rápida y fácil, por lo tanto es útil para todo el mundo.

3. Vagrant vs. otros softwares

3.1. Vagrant vs. Docker



Docker depende en el sistema operativo del host, en cambio, **vagrant** incluye el sistema operativo como parte de su paquete, por lo tanto **vagrant** puede contener cualquier sistema operativo y por otra parte los contenedores de **Docker** se ejecutan en sistemas operativos de Linux.

3.2. Vagrant vs. Terraform

Con **Terraform** describes tu estructura completa como código y tienes la posibilidad de construir los *"resources"* a través de diferentes proveedores, por ejemplo, un servidor de base datos puede estar alojado en **Heroku** y los demás servidores pueden estar en **AWS**, **Terraform** lo que hará es lanzar todos estos servers en los diferentes proveedores en paralelo.

Vagrant en cambio es una herramienta que prporciona el *"framework"* y la confiuguración por tal de poder crear y gestionar ámbitos de desarrolo portable, estos ámbitos pueden estar en un host local o en la nube, y también son portables tanto para Windows, Mas OS y Linux.