# Vagrant

**Vagrant** es una herramienta para la creación y configuración de entornos de desarrollo vitalizados

Necesita Virtual Box

## Creación de proyecto

### Configurar proxy

1. Comandos para Windows
   1. set http\_proxy=http://10.110.8.42:8080
   2. set https\_proxy=http://10.110.8.42:8080
2. Comandos para Linux
   1. export http\_proxy= http://10.110.8.42:8080
   2. export https\_proxy= http://10.110.8.42:8080
3. Para terminar
   1. vagrant plugin install vagrant-proxyconf
   2. set VAGRANT\_HTTP\_PROXY="%http\_proxy%"

### Vagrantfile

* Dónde se encuentra el directorio base del proyecto.
* Describe el tipo de máquina y recursos que necesitamos para ejecutar nuestro proyecto, así como el software que necesitamos instalar y cómo queremos acceder a él.

Para crearlo usaremos el comando: **$ vagrant init**

### Box

**Descargar el Box**

Vagrant usa una imagen de base, que puede clonar rápidamente. A estas imágenes que se utilizan como base es a lo que se denominan boxes.

Comando: vagrant box add (nombre) [url del box]

Añadir box:

**$ vagrant box add Ubuntu/Xenial64 https://app.vagrantup.com/ubuntu/boxes/xenial64**

Ubuntu/Xenial64 es el nombre que se le pone a la máquina, tiene que estar en concordancia con el Box.

Esta descarga se hace una sola vez, ya que estas imágenes se comparten entre todos los proyectos. Es decir cada proyecto hace referencia a un box por su nombre pero los proyectos en ningún caso modifican esta imagen de base, por lo que se puede compartir sin problema.

**Listar los Box**

Para listar los box instalados en nuestro Vagrant

**$vagrant box list**

**Usar el box**

Decirle al proyecto que se base en el box para crear la máquina virtual.

Sustituir en *Vagrantfile*: **config.vm.box = "base"** por **config.vm.box = "ubuntu/xenial64"**

### Levantar y entrar en la máquina virtual

**$ vagrant up**

### Guest Additions

**$ vagrant plugin install vagrant-vbguest**

Para probar que se han instalado correctamente, paramos y arrancamos la maquina:

**$ vagrant halt**

**$ vagrant up**

### Entrando por SSH

**$ vagrant ssh**

El directorio /vagrant de nuestra máquina virtual es un directorio compartido con el directorio de nuestro proyecto en la máquina física.

## Configurar varias máquinas virtuales

Lo que hay que hacer es modificar el fichero *VagrantFile*

1. Con **config.vm.define :***[NombreMaquina]* **do |***[NombreMaquina]***|**
2. Dentro de la configuración de la maquina añadir:

*[NombreMaquina]***.vm.host\_name = "***[NombreMaquina]*"

1. Se finaliza la configuración de una maquina con el comando: **end**
2. Ejempo de VagrantFile con 2 máquinas diferentes

# -\*- mode: ruby -\*-

# vi: set ft=ruby :

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

config.vm.define :nodo1 do |nodo1|

nodo1.vm.box = "precise32"

nodo1.vm.hostname = "nodo1"

nodo1.vm.network :private\_network, ip: "10.1.1.101"

end

config.vm.define :nodo2 do |nodo2|

nodo2.vm.box = "precise32"

nodo2.vm.hostname = "nodo2"

nodo2.vm.network :public\_network,:bridge=>"wlan0"

end

end

## Instalar software para que lo vean mis compañeros

Crear el Script bootstrap.sh con este contenido:

*#!/usr/bin/env bash*

*apt-get update*

*apt-get install -y apache2*

*rm -rf /var/www*

*ln -fs /vagrant /var/www*

Se trata de un script de Unix donde estamos diciendo al sistema de paquetes que instale un Apache Web Server, y que el directorio que tiene que servir como contenido web es nuestro directorio compartido /vagrant.

Editamos el fichero *Vagrantfile* y añadimos (antes del end):

**config.vm.provision :shell, :path => "bootstrap.sh"**

Para que el cambio tenga efecto reiniciamos la maquina: **$ vagrant reload –provisioning**

Comprobamos que haya ido bien:

**$ vagrant ssh**

**vagrant@precise64:~$ wget -qO- 127.0.0.1**

Debería mostrar la salida por defecto del Apache

## Configuración de la red

Editar el fichero *Vagrantfile*  y buscar la opción: *config.vm.network* donde hay comentadas 3 posibilidades:

* *port forwarding* – Hacer un mapeo de puertos para que el tráfico que llega a un puerto del host (máquina física) se redirija a un puerto del guest (máquina virtual).
* *private network* – Crear otra red privada que sólo es accesible desde el host.
* *bridged network* – Hacer que el guest se comporte como otra máquina más dentro de la red del host.

Para descomentarlo solo es necesario borrar **#**

Después es necesario reiniciar la maquina

Nos preguntara qué interfaz de red del host queremos usar para el guest.

## Despedida y cierre

Distintas maneras de apagar la maquina:

* **vagrant suspend** – Queda el ordenador suspendido. La ventaja: Es más rápido y se queda guardado el estado todos los programas y memoria en ese justo memento. El inconveniente: consume más espacio en disco de nuestro host.
* **vagrant halt** – Apagar el ordenador. Inconvenientes: Más lento que el método anterior, y cuando arrancamos de nuevo la máquina virtual todos los servicios se vuelven a arrancar de cero. Ventajas: Consume menos disco duro de nuestro host.
* **vagrant destroy** – Borra todo rastro de la máquina virtual de nuestro host, ahorrando así espacio en disco. Inconvenientes: Al volver arrancar tiene que volver a hacer todo el proceso de configuración, y tarda mucho más. No sería recomendable salvo que sea un proyecto con el que ya no vamos a trabajar.

## Bibliografía

[**https://www.adictosaltrabajo.com/tutoriales/vagrant-install/**](https://www.adictosaltrabajo.com/tutoriales/vagrant-install/) **(**Autor: Alejandro Pérez García)

[**https://www.josedomingo.org/pledin/2013/09/gestionando-maquinas-virtuales-con-vagrant/**](https://www.josedomingo.org/pledin/2013/09/gestionando-maquinas-virtuales-con-vagrant/)(Contenido adicional)

[**https://www.vagrantup.com/docs/cli/up.html**](https://www.vagrantup.com/docs/cli/up.html)(Documentación oficial)