

# Tarea de introducción a Vagrant

## Introducción

Aunque nuestro enfoque será en *Docker*, Vagrant es otra tecnología muy popular que conviene conocer. Es por ello que se nos encargó la tarea de generar un entorno de prueba con una aplicación *Hello, World* bajo **Spring Boot** usando esta tecnología.

## Metodología

### Instalación de Vagrant y sus requisitos

Vagrant es un software multiplataforma que debe ser instalado en el entorno de trabajo. Además, por debajo hace uso de máquinas virtuales, por lo que también hay que tener el software adecuado para este fin. Una opción preferida y la que usé yo fue VirtualBox.

Una vez instalada, se accede a ella mediante el comando **vagrant**

A screenshot of a terminal window showing the Vagrant help text. The terminal has a dark blue background with a faint circuit pattern. The text is white and lists various commands and their descriptions. The commands listed include: box, destroy, global-status, halt, help, init, login, package, plugin, port, powershell, provision, push, rdp, reload, resume, snapshot, ssh, ssh-config, status, suspend, up, validate, and version. Each command is followed by a brief description of its function. For example, 'box' manages boxes: installation, removal, etc., and 'up' starts and provisions the vagrant environment.

```
C:\Users\jgonzalez> vagrant
Usage: vagrant [options] <command> [<args>]

-v, --version          Print the version and exit.
-h, --help             Print this help.

Common commands:
box                    manages boxes: installation, removal, etc.
destroy               stops and deletes all traces of the vagrant machine
global-status         outputs status Vagrant environments for this user
halt                 stops the vagrant machine
help                 shows the help for a subcommand
init                 initializes a new Vagrant environment by creating a Vagrantfile
login                log in to HashiCorp's Vagrant Cloud
package              packages a running vagrant environment into a box
plugin               manages plugins: install, uninstall, update, etc.
port                 displays information about guest port mappings
powershell           connects to machine via powershell remoting
provision             provisions the vagrant machine
push                deploys code in this environment to a configured destination
rdp                  connects to machine via RDP
reload               restarts vagrant machine, loads new Vagrantfile configuration
resume              resume a suspended vagrant machine
snapshot            manages snapshots: saving, restoring, etc.
ssh                  connects to machine via SSH
ssh-config           outputs OpenSSH valid configuration to connect to the machine
status              outputs status of the vagrant machine
suspend             suspends the machine
up                  starts and provisions the vagrant environment
validate            validates the Vagrantfile
version              prints current and latest Vagrant version

For help on any individual command run "vagrant COMMAND -h"
```

Figure 1: vgt

## VagrantFile

La configuración de la máquina virtual viene dada por el VagrantFile, del que caben destacar las siguiente configuraciones:

- Presentación de puertos: para poder acceder a la máquina fácilmente desde el navegador del *host*

```
config.vm.network "forwarded_port", guest: 8080, host: 8081, host_ip: "127.0.0.1"
config.vm.network "forwarded_port", guest: 80, host: 81, host_ip: "127.0.0.1"
```

- Creación de directorios e instalación de paquetes: necesarios para correr nuestra aplicación. **Notar el comentario del proxy**, que es necesario tenerlo en cuenta al instalar los paquetes.

```
config.vm.provision "shell", inline: <<-SHELL
echo "Configuring my c00l app..."
echo "Checking if the app directory exists..."
if [ -d "/home/vagrant/javaApp" ]; then
  echo "Dir already exists!"
else
  echo "Dir not found! Creating..."
  mkdir /home/vagrant/javaApp
  echo "Do not forget permissions!!"
  sudo chown -R vagrant:vagrant /home/vagrant/javaApp/
  sudo chmod +xrw /home/vagrant/javaApp
fi
#export http_proxy=""
#export https_proxy=""
apt-get update
apt-get install -y maven apache2
SHELL
```

- Copia del código y configuración: necesario para poder correr la aplicación al arrancar la máquina

```
config.vm.provision "file", source: "app.jar", destination: "/home/vagrant/javaApp/app.jar"
config.vm.provision "shell", inline: <<-SHELL
echo "Preparing the application..."
cd /home/vagrant/javaApp/
echo "Setting JAVA_HOME"
export JAVA_HOME="/usr/lib/jvm/default-java"
echo "Running..."
java -jar app.jar &
SHELL
```

## Configuración y uso de Vagrant

Aunque hay más comandos y conceptos, habiendo ensamblado el fichero anterior, levantar la aplicación es tan sencillo como ejecutar un **vagrant up**, que iniciará y ejecutará la máquina virtual con su aplicación.

Es importante destacar que estos pasos **requieren descargas desde Internet**, por lo que si nos encontramos detrás de un proxy, deberemos tener las variables de entorno debidamente configuradas, de forma similar a:

```
set http_proxy=http://user:password@host:port
```

```
set https_proxy=%http_proxy%
```

Con estas variables de entorno seteadas, podemos instalar un plugin de vagrant ya preparado para trabajar con proxy, mediante: `vagrant plugin install vagrant-proxyconf`