

**SERVIDORES WEB Y ALTA DISPONIBILIDAD (2021-2022)**  
GRADO EN INGENIERÍA INFORMÁTICA  
UNIVERSIDAD DE GRANADA

---

# Memoria Trabajo Final

---

Pedro Antonio Mayorgas Parejo y Adrián Acosa Sánchez

17 de junio de 2022

# Índice

<b>1</b>	<b>Abstract</b>	<b>3</b>
<b>2</b>	<b>Definición y despliegue de la infraestructura</b>	<b>5</b>
2.1	Instalación de Vagrant en Debian/Ubuntu . . . . .	5
2.2	Fichero Vagrantfile . . . . .	7
2.3	Configuración de los parámetros de red básicos . . . . .	9
2.4	Parámetros del proveedor . . . . .	10
2.5	Definición de los servidores . . . . .	11
2.6	Despliegue de máquinas del Vagrantfile . . . . .	12
<b>3</b>	<b>Ansible</b>	<b>15</b>
3.1	Instalación de Ansible en Debian/Ubuntu . . . . .	15
3.2	Inventario . . . . .	16
3.3	Ansible-playbooks . . . . .	17
3.4	Playbooks usados para el proyecto y para qué sirven. . . . .	18
3.4.1	Playbook de hosts.yml . . . . .	18
3.4.2	Playbook de SSL.yml . . . . .	21
3.4.3	Playbook de webservers.yml . . . . .	24
3.4.4	Playbook de loadbalancer.yml . . . . .	28
3.4.5	Playbook de databases.yml . . . . .	31
<b>4</b>	<b>Conclusiones</b>	<b>33</b>
<b>5</b>	<b>Bibliografía</b>	<b>35</b>

## 1. Abstract

Este proyecto, es una demostración de la mejora y automatización de los trabajos realizados durante el curso 2021 y 2022 en la asignatura de SWAP.

Durante los trabajos anteriores, hemos conseguido documentar y elaborar un sistema estático de una granja web. Pero a la hora de re-desplegarlo en cualquier entorno, tanto de producción como de desarrollo. Nos encontramos con que el despliegue consume muchos recursos humanos, económicos y de tiempo. Aún teniendo una memoria y datos previos del primer entorno desarrollado de manera manual que nos guíe y acelere el proceso.

Tampoco podemos contar con herramientas creadas con Scripting de Bash o alternativas, ya que son herramientas que solo se pueden desarrollar para especializarse en una cosa y dependen mucho de la línea de comandos en la que se esté ejecutando en el momento. Como ejemplo: tenemos que instalar Apache2, la instalación sabemos que se haría por SSH, pero si tenemos innumerables servidores web esperando y desplegados. El provisionamiento, con la herramienta creada a través de Bash que muy difícilmente es escalable. Además no contamos directamente con la gestión de errores y todo ello depende de la salida estándar STDOUT. Por lo que tendríamos que desarrollar un controlador de errores a partir de esa salida. Lo cual haría del proyecto menos viable aún y aún más especializado.

También en bash, al cambiar entre versiones de software la gestión de errores, métodos de instalación, etc... Quedaría automáticamente obsoleto (deprecated). Haciendo del proyecto poco reutilizable o ni siquiera se podrá reutilizar. Destinando a la basura tiempo de trabajo y dinero.

Como herramienta de despliegue de máquinas virtuales y instancias (llamadas por el software que usamos como **boxes**) usamos Vagrant que permite definir una infraestructura con código. O denominada correctamente como *Infraestructure As Code IaC o Orquestación*, que permite un despliegue de la misma configuración con un código en local en servidores con hypervisor, en workstations para probarlo, en los proveedores de Cloud. Con el objetivo de asegurar de que todo un equipo tanto de desarrollo como de infraestructura puedan trabajar bajo las mismas condiciones físicas.

Por ello Ansible es la herramienta de provisionamiento más importante y quizás el futuro o el punto de partida de otras herramientas mejoradas. Permite a través de código el control de los estados de instalación, control de los servicios disponibles en el sistema, realizar ajustes específicos a un grupo objetivo más pequeño, un sin fin de distintas soluciones que unidas permiten de manera **idempotente** que se de el mismo resultado sin importar qué versión de software de paquetería tenga, versión de servidor SQL, etc...

**Nota:** No vamos a pararnos a explicar los ajustes dados a cada servicio y a SSL. Son los mismos ajustes explicados en las prácticas anteriores disponibles en mi repositorio de memorias de SWAP. Disponible en la bibliografía.

## 2. Definición y despliegue de la infraestructura

La herramienta utilizada para tal propósito es Vagrant, esto nos permite asegurar los despliegues que son iguales en todos los sitios en los que se disponga del mismo Vagrantfile que es el fichero que define la infraestructura.

### 2.1. Instalación de Vagrant en Debian/Ubuntu

La instalación en sistemas Linux basados en Debian consiste en la ejecución de los siguientes comandos:

- 1 \$ curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo apt-key add -
- 2 \$ sudo apt-add-repository "deb [arch=amd64] https://apt.releases.hashicorp.com \$(lsb\_release -cs) main"
- 3 \$ sudo apt-get update && sudo apt-get install vagrant

Instalación de libvirt y la librería de Vagrant Libvirt necesarios en sistemas Debian:

- 1 \$ sudo apt-get build-dep vagrant ruby-libvirt
- 2 \$ sudo apt-get install qemu libvirt-daemon-system libvirt-clients ebtables dnsmasq-base
- 3 \$ sudo apt-get install libxslt-dev libxml2-dev libvirt-dev zlib1g-dev ruby-dev
- 4 \$ sudo apt-get install libguestfs-tools

```
adrian@debian:~$ curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo apt-key add -
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
adrian@debian:~$ sudo apt-add-repository "deb [arch=amd64] https://apt.releases.hashicorp.com $(lsb_release -cs) main"
adrian@debian:~$ sudo apt-get update && sudo apt-get install vagrant
Hit:1 http://deb.debian.org/debian bullseye InRelease
Hit:2 http://packages.microsoft.com/repos/code stable InRelease
Get:3 https://apt.releases.hashicorp.com bullseye InRelease [11.2 kB]
Hit:4 https://dl.google.com/linux/chrome/deb stable InRelease
Get:5 https://apt.releases.hashicorp.com bullseye/main amd64 Packages [57.6 kB]
Fetched 68.8 kB in 2s (31.0 kB/s)
Reading package lists... Done
W: https://apt.releases.hashicorp.com/dists/bullseye/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATION section in apt-key(8) for details.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libarchive-tools nfs-kernel-server racc rsync ruby-bcrypt-pbkdf ruby-builder ruby-childprocess ruby-co
ncurrent ruby-ed25519 ruby-erubi ruby-excon ruby-ffi ruby-fog-core ruby-fog-json ruby-fog-libvirt
  ruby-fog-xml ruby-formatador ruby-i18n ruby-libvirt ruby-listen ruby-log4r ruby-mime-types ruby-mime-t
ypes-data ruby-mini-portile2 ruby-multi-json ruby-net-scp ruby-net-sftp ruby-net-ssh ruby-nokogiri
  ruby-oj ruby-pkg-config ruby-rb-inotify ruby-vagrant-cloud ruby-zip
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  vagrant
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 41.5 MB of archives.
After this operation, 117 MB of additional disk space will be used.
Get:1 https://apt.releases.hashicorp.com bullseye/main amd64 vagrant amd64 2.2.19 [41.5 MB]
Fetched 41.5 MB in 4s (11.4 MB/s)
Selecting previously unselected package vagrant.
(Reading database ... 243038 files and directories currently installed.)
Preparing to unpack .../vagrant_2.2.19_amd64.deb ...
Unpacking vagrant (2.2.19) ...
Setting up vagrant (2.2.19) ...
```

Figura 1: Instalación de Vagrant en Debian 11.

```

adrian@debian:~$ sudo apt-get install qemu-system-x86 libvirt-daemon-system libvirt-clients ebttables dns
masq-base libxslt1-dev libxml2-dev libvirt-dev zlib1g-dev ruby-dev libguestfs-tools
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
qemu-system-x86 is already the newest version (1:7.0+dfsg-7).
The following additional packages will be installed:
  augeas-lenses bsdmainutils db-util fonts-lato guestfish guestfs-tools guestmount hfsplus icoutils
  icu-devtools kpartx ldmtool libafflib0v5 libaugeas0 libbfiol libc-dev-bin libc-devtools libc6-dev
  libc6-dev libconfig9 libcrypt-dev libdate-manip-perl libdpkg-perl libewf2 libfile-fcntllock-perl libgmp-dev
  libgmpxx4ldbl libguestfs-hfsplus libguestfs-perl libguestfs-reiserfs libguestfs-xfs libguestfs0
  libhfs0 libhivex0 libicu-dev libintl-perl libintl-xs-perl libjs-jquery libltdl-1.0-0 libnetpbm11
  libnsl-dev librpm9 librpmio9 libruby3.0 libstring-shellquote-perl libsys-virt-perl libtirpc-dev
  libtsk19 libvhd11 libvirt-daemon-config-network libvirt-daemon-config-nwfilter
  libvirt-daemon-system-systemd libvmdk1 libwin-hivex-perl libxar9 linux-libc-dev lsscsi manpages-dev
  mdevctl ncal netpbm pkg-config rake rpcsvc-proto rpm-common ruby ruby-net-telnet ruby-rubygems
  ruby-webrick ruby-xmlrpc ruby3.0-dev ruby3.0-doc rubygems-integration scrub sleuthkit
  supermin syslinux virt-p2v zerofree
Suggested packages:
  augeas-doc calendar whois vacation mailutils libterm-readline-gnu-perl | libterm-readline-perl-perl
  augeas-tools glibc-doc debian-keyring gcc | c-compiler bzip2 gmp-doc libgmp10-doc libmpfr-dev
  libguestfs-gfs2 libguestfs-jfs libguestfs-nifs libguestfs-rescue libguestfs-rsync libguestfs-zfs
  icu-doc libvirt-login-shell auditd open-iscsi pm-utils systemd zfsutils dpkg-dev ri bundler
  autopsy mac-robber
The following NEW packages will be installed:
  augeas-lenses bsdmainutils db-util dnsmasq-base ebttables fonts-lato guestfish guestfs-tools
  guestmount hfsplus icoutils icu-devtools kpartx ldmtool libafflib0v5 libaugeas0 libbfiol
  libc-dev-bin libc-devtools libc6-dev libconfig9 libcrypt-dev libdate-manip-perl libdpkg-perl libewf2
  libfile-fcntllock-perl libgmp-dev libgmpxx4ldbl libguestfs-hfsplus libguestfs-perl
  libguestfs-reiserfs libguestfs-tools libguestfs-xfs libguestfs0 libhfs0 libhivex0 libicu-dev
  libintl-perl libintl-xs-perl libjs-jquery libltdl-1.0-0 libnetpbm11 libnsl-dev librpm9 librpmio9
  libruby3.0 libstring-shellquote-perl libsys-virt-perl libtirpc-dev libtsk19 libvhd11 libvirt-clients
  libvirt-daemon-config-network libvirt-daemon-config-nwfilter libvirt-daemon-system
  libvirt-daemon-system-systemd libvirt-dev libvmdk1 libwin-hivex-perl libxar9 linux-libc-dev libxslt1-dev
  libxar9 linux-libc-dev lsscsi manpages-dev mdevctl ncal netpbm pkg-config rake rpcsvc-proto
  rpm-common ruby ruby-dev ruby-net-telnet ruby-rubygems ruby-webrick ruby-xmlrpc ruby3.0-dev ruby3.0-doc
  rubygems-integration scrub sleuthkit supermin syslinux virt-p2v zerofree zlib1g-dev
0 upgraded, 89 newly installed, 0 to remove and 28 not upgraded.
Need to get 86.5 kB/59.3 MB of archives.
After this operation, 243 MB of additional disk space will be used.
Do you want to continue? [Y/n]

```

Figura 2: Instalación de Vagrant libvirt provider en Debian 11.

Una vez instalado Vagrant, procederemos a crear nuestro fichero de despliegue llamado **Vagrantfile** por defecto, que nos da una idea general de cómo trabaja Vagrant con los proveedores y boxes. Además de generar el entorno que necesita Vagrant para trabajar ya que tiene una carpeta oculta con sus configuraciones dentro de la carpeta de trabajo.

```
1 $ vagrant init
```

```

peter@mark-5:~/Documents/working/pruebaVagrant$ vagrant init
A 'Vagrantfile' has been placed in this directory. You are now
ready to 'vagrant up' your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
'vagrantup.com' for more information on using Vagrant.
peter@mark-5:~/Documents/working/pruebaVagrant$ cat Vagrantfile
# -*- mode: ruby -*-
# vi: set ft=ruby :

# All Vagrant configuration is done below. The "2" in Vagrant.configure
# configures the configuration version (we support older styles for
# backwards compatibility). Please don't change it unless you know what
# you're doing.
Vagrant.configure("2") do |config|
  # The most common configuration options are documented and commented below.
  # For a complete reference, please see the online documentation at
  # https://docs.vagrantup.com.

  # Every Vagrant development environment requires a box. You can search for
  # boxes at https://vagrantcloud.com/search.
  config.vm.box = "base"

  # Disable automatic box update checking. If you disable this, then
  # boxes will only be checked for updates when the user runs
  # 'vagrant box outdated'. This is not recommended.
  # config.vm.box_check_update = false

  # Create a forwarded port mapping which allows access to a specific port
  # within the machine from a port on the host machine. In the example below,
  # accessing "localhost:8080" will access port 80 on the guest machine.
  # NOTE: This will enable public access to the opened port
  # config.vm.network "forwarded_port", guest: 80, host: 8080

```

Figura 3: Creando un Vagrantfile por defecto.

## 2.2. Fichero Vagrantfile

El fichero Vagrantfile del proyecto es como sigue:

```
1  # -*- mode: ruby -*-
2  # vi: set ft=ruby :
3
4  VAGRANTFILE_API_VERSION = "2"
5
6  Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
7  # General Vagrant VM configuration.
8  config.ssh.insert_key = true
9  # config.ssh.private_key_path = ["~/.ssh/id_rsa", "~/.vagrant.
    d/insecure_private_key"]
10 config.ssh.forward_agent = true
11 config.nfs.functional = false
12
13 config.vm.provision "file", source: "~/.ssh/id_rsa.pub", destination: "~/.
    ssh/admin_rsa.pub"
14 config.vm.provision "shell", inline: <<-EOC
15     echo "nameserver 8.8.8.8" > /etc/resolv.conf
16     cat /home/vagrant/.ssh/admin_rsa.pub >> /home/vagrant/.
        ssh/authorized_keys
17     rm /home/vagrant/.ssh/admin_rsa.pub
18     sudo ip route del default via 192.168.121.1
19     sudo ip route add default via 192.168.100.1
20     echo "finished"
21 EOC
22
23 config.vm.provider :libvirt do |libvirt|
24     libvirt.cpus = 2
25     libvirt.memory = 2048
26     libvirt.cputopology :sockets => '1', :cores => '2', :threads => '1'
27     libvirt.driver = "kvm"
28     # libvirt.storage_pool_name = "DISK_IMG"
29 end
30
31 # Web Server 1
32 config.vm.define :m1 do |app|
33     app.vm.box = "generic/debian11"
34     app.vm.hostname = "m1.swapfinal"
35
36     app.vm.network "private_network", ip: "192.168.100.4", netmask: "
        255.255.255.0", libvirt__network_name: "ansible",
        libvirt__dhcp_enabled: false, libvirt__host_ip: '192.168.100.1'
37 end
38
39 # Web Server 2
40 config.vm.define :m2 do |app|
41     app.vm.box = "generic/debian11"
42     app.vm.hostname = "m2.swapfinal"
43
44     app.vm.network "private_network", ip: "192.168.100.5", netmask: "
        255.255.255.0", libvirt__network_name: "ansible",
```

```

45         libvirt__dhcp_enabled: false , libvirt__host_ip: '192.168.100.1'
46     end
47     # Web Server 3
48     config.vm.define :m3 do |app|
49         app.vm.box = "generic/debian11"
50         app.vm.hostname = "m3.swapfinal"
51
52         app.vm.network "private_network", ip: "192.168.100.6", netmask: "
            255.255.255.0", libvirt__network_name: "ansible",
            libvirt__dhcp_enabled: false , libvirt__host_ip: '192.168.100.1'
53     end
54
55     # Web Server 4
56     config.vm.define :m4 do |app|
57         app.vm.box = "generic/debian11"
58         app.vm.hostname = "m4.swapfinal"
59
60         app.vm.network "private_network", ip: "192.168.100.7", netmask: "
            255.255.255.0", libvirt__network_name: "ansible",
            libvirt__dhcp_enabled: false , libvirt__host_ip: '192.168.100.1'
61     end
62
63     # Web Server 5
64     config.vm.define :m5 do |app|
65         app.vm.box = "generic/debian11"
66         app.vm.hostname = "m5.swapfinal"
67
68         app.vm.network "private_network", ip: "192.168.100.8", netmask: "
            255.255.255.0", libvirt__network_name: "ansible",
            libvirt__dhcp_enabled: false , libvirt__host_ip: '192.168.100.1'
69     end
70
71     # Web Server 6
72     config.vm.define :m6 do |app|
73         app.vm.box = "generic/debian11"
74         app.vm.hostname = "m6.swapfinal"
75
76         app.vm.network "private_network", ip: "192.168.100.9", netmask: "
            255.255.255.0", libvirt__network_name: "ansible",
            libvirt__dhcp_enabled: false , libvirt__host_ip: '192.168.100.1'
77     end
78
79     # LoadBalancer
80     config.vm.define :m0 do |lb|
81         lb.vm.box = "generic/debian11"
82         lb.vm.hostname = "m0.swapfinal"
83
84         lb.vm.network "private_network", ip: "192.168.100.2", netmask: "
            255.255.255.0", libvirt__network_name: "ansible",
            libvirt__dhcp_enabled: false , libvirt__host_ip: '192.168.100.1'
85     end
86
87     # DB server

```



```

88     config.vm.define :db1 do |db|
89         db.vm.box = "rockylinux/8"
90         db.vm.hostname = "m20.swapfinal"
91
92         db.vm.network "private_network", ip: "192.168.100.20", netmask: "
           255.255.255.0", libvirt__network_name: "ansible",
           libvirt__dhcp_enabled: false, libvirt__host_ip: '192.168.100.1'
93     end
94
95     # DB server 2
96     config.vm.define :db2 do |db|
97         db.vm.box = "rockylinux/8"
98         db.vm.hostname = "m21.swapfinal"
99
100        db.vm.network "private_network", ip: "192.168.100.21", netmask: "
           255.255.255.0", libvirt__network_name: "ansible",
           libvirt__dhcp_enabled: false, libvirt__host_ip: '192.168.100.1'
101    end
102 end

```

### 2.3. Configuración de los parámetros de red básicos

El Vagrantfile permite un provisionamiento básico que utilizamos para configurar parámetros de red.

1. Parte de SSH, DNS y Redes por defecto. Son parámetros de aprovisionamiento aplicados a todas las máquinas del Vagrantfile sin distinción.
  - a) Indicamos que queremos pasarle nuestro cliente local de SSH `forward_agent` para configurar las conexiones SSH para un posterior uso.
  - b) Indicamos la ruta de nuestra clave de SSH privada, que automáticamente Vagrant la utiliza para poder realizar conexiones. Esta parte se encuentra desactivada porque Vagrant no consigue trabajar bien con las claves SSH del usuario. Usa unas propias.
  - c) Desactivamos las conexiones de NFS, ya que Vagrant asegura que tengamos carpetas compartidas con las que comunicar ficheros al servidor, en nuestro caso no son necesarias y es una característica con muchos bugs y fallos.
  - d) Realizamos un aprovisionamiento general que consiste en que almacene bien la clave pública de SSH, esto es debido a que Vagrant puede no configurarla bien para nuestro cliente SSH para administrar con consola las máquinas. Entonces debemos poner esto para que podamos conectarnos correctamente, así como no borrar la clave pública con la cual Vagrant se conecta a través de la red de gestión para comprobar el estado de cada una de las máquinas.
  - e) Realizamos otro aprovisionamiento consistente en indicarle los servidores DNS correctos, ya que en la UGR su servidor DNS da muchos problemas para realizar conexiones con máquinas que no tengan el Certificado de conexión de

WiFi. O en otros sitios donde el DNS recibido por DHCP no funcione correctamente, ya que en KVM se lee del `/etc/resolv.conf` del sistema hypervisor.

- f) Finalmente el último aprovisionamiento es que le indicamos que desactive como Gateway la ip del “router” de la red de gestión de Vagrant que la usa el software para apagar, encender máquinas y comprobar sus estados. Configurando como ruta por defecto la de nuestra red preparada con nombre de **ansible** para evitar conflictos con la asignación dinámica de la red de gestión.

Todo ello se configura en el Vagrantfile de la siguiente manera:

```
1 # General Vagrant VM configuration.
2 config.ssh.insert_key = true
3 # config.ssh.private_key_path = ["~/.ssh/id_rsa", "~/.vagrant.
  d/insecure_private_key"]
4 config.ssh.forward_agent = true
5 config.nfs.functional = false
6
7 config.vm.provision "file", source: "~/.ssh/id_rsa.pub", destination: "~/.
  ssh/admin_rsa.pub"
8 config.vm.provision "shell", inline: <<-EOC
9   echo "nameserver 8.8.8.8" > /etc/resolv.conf
10   cat /home/vagrant/.ssh/admin_rsa.pub >> /home/vagrant/.
    ssh/authorized_keys
11   rm /home/vagrant/.ssh/admin_rsa.pub
12   sudo ip route del default via 192.168.121.1
13   sudo ip route add default via 192.168.100.1
14   echo "finished"
15 EOC
```

## 2.4. Parámetros del proveedor

Todas las **boxes** que se definan en el Vagrantfile tendrán las mismas características en cuanto al hardware virtualizado. En el fichero de configuración le indicamos al proveedor las características hardware que tienen como el número de núcleos de procesamiento, memoria RAM, topología de la CPU (esto solo se define para KVM por la alta especialización en emulación de QEMU. Sin embargo si es virtualbox el proveedor por defecto no haría falta hacer tanta especificación ya que es más básico) y el driver de virtualización a utilizar KVM que usa aceleración por hardware. Para realizar el despliegue de cada una de las **boxes** indicando las siguientes líneas, respectivamente:

```
1 config.vm.provider :libvirt do |libvirt|
2   libvirt.cpus = 2
3   libvirt.memory = 2048
4   libvirt.cputopology :sockets => '1', :cores => '2', :threads => '1'
5   libvirt.driver = "kvm"
6   # libvirt.storage_pool_name = "DISK_IMG"
7 end
```

## 2.5. Definición de los servidores

Consta de varias partes consistentes en:

1. Definimos el tipo de **box** que queremos. Un box es una imagen de Sistema Operativo preparada para trabajar con Vagrant y recibir todas los aprovisionamientos que se definan.
2. Definimos el hostname, esto se define en cada definición de máquina virtual por separado.
3. Definimos la IP estática, máscara de subred y la red privada que se crea a través del proveedor KVM que tiene de nombre como ansible. En la cual **no** tenemos DHCP para evitar que cualquier máquina futura tenga la misma IP que una estática.

La unica diferencia entre los servidores de bases de datos y el resto es que usamos otro tipo de **box** que utiliza RockyLinux 8 como se puede observar en el bloque de configuración siguiente:

### Box de Debian: Se aplica a los servidores web y sus balanceadores de carga

```
1  # Web Server 1
2  config.vm.define :ml do |app|
3    app.vm.box = "generic/debian11"
4    app.vm.hostname = "ml.swapfinal"
5
6    app.vm.network "private_network", ip: "192.168.100.4", netmask: "
      255.255.255.0", libvirt__network_name: "ansible",
      libvirt__dhcp_enabled: false, libvirt__host_ip: '192.168.100.1'
7  end
```

### Box de Rocky: Solo se aplica a los servidores que contienen un SGBD

```
1  # DB server
2  config.vm.define :db1 do |db|
3    db.vm.box = "rockylinux/8"
4    db.vm.hostname = "m20.swapfinal"
5
6    db.vm.network "private_network", ip: "192.168.100.20", netmask: "
      255.255.255.0", libvirt__network_name: "ansible",
      libvirt__dhcp_enabled: false, libvirt__host_ip: '192.168.100.1'
7  end
```

## 2.6. Despliegue de máquinas del Vagrantfile

Una vez tenemos nuestro fichero de despliegue Vagrantfile completo con nuestras necesidades, procedemos a ejecutar el siguiente comando que levantará toda la infraestructura especificada:

```
1 $ vagrant up
```

La primera parte es similar a sistemas Docker. Vagrant comprueba si los ficheros de imagen de los box existen en el sistema. Si no existen se los descarga de los repositorios de imagenes de Vagrant. Antes de eso, indica que las máquinas que lee desde el fichero Vagrantfile están con KVM o libvirt que es la librería que se utiliza. Esto implica además que se puede ejecutar más de un proveedor distinto en el Vagrantfile.

```
peter@mark-5:~/Documents/working/AnsibleWebFarm$ vagrant up
Bringing machine 'm1' up with 'libvirt' provider...
Bringing machine 'm2' up with 'libvirt' provider...
Bringing machine 'm3' up with 'libvirt' provider...
Bringing machine 'm4' up with 'libvirt' provider...
Bringing machine 'm5' up with 'libvirt' provider...
Bringing machine 'm6' up with 'libvirt' provider...
Bringing machine 'm0' up with 'libvirt' provider...
Bringing machine 'db1' up with 'libvirt' provider...
Bringing machine 'db2' up with 'libvirt' provider...
==> m5: Checking if box 'generic/debian11' version '4.0.2' is up to date...
==> m4: Checking if box 'generic/debian11' version '4.0.2' is up to date...
==> m0: Checking if box 'generic/debian11' version '4.0.2' is up to date...
==> m3: Checking if box 'generic/debian11' version '4.0.2' is up to date...
==> m1: Checking if box 'generic/debian11' version '4.0.2' is up to date...
==> m6: Checking if box 'generic/debian11' version '4.0.2' is up to date...
==> m2: Checking if box 'generic/debian11' version '4.0.2' is up to date...
==> db2: Checking if box 'rockylinux/8' version '5.0.0' is up to date...
==> db1: Checking if box 'rockylinux/8' version '5.0.0' is up to date...
==> m0: Creating image (snapshot of base box volume).
==> m2: Creating image (snapshot of base box volume).
==> m6: Creating image (snapshot of base box volume).
==> db1: Creating image (snapshot of base box volume).
==> m6: Creating domain with the following settings...
==> m4: Creating image (snapshot of base box volume).
==> m1: Creating image (snapshot of base box volume).
```

Figura 4: Despliegue del Vagrantfile del proyecto - Obteniendo las imagenes y enlazando las máquinas box con el proveedor.

La siguiente parte del despliegue dependerá de cada proveedor. En nuestro caso, al usar KVM un emulador de hardware como QEMU se puede ver que se definen los parámetros del Hardware por defecto para una máquina virtual y los de la topología de la CPU. También el nombre de las máquinas virtuales definidas en el dominio de QEMU. Dependiendo del nº de máquinas definidas por nuestro código, la salida será muy larga ya que se ejecutan en paralelo todas las máquinas posibles.

```

==> db1: Creating domain with the following settings...
==> m1: -- Name: AnsibleWebFarm_m1
==> m3: Creating image (snapshot of base box volume).
==> m1: -- Description: Source: /home/peter/Documents/working/AnsibleWebFarm/Vagrantfile
==> db1: -- Name: AnsibleWebFarm_db1
==> m1: -- Domain type: kvm
==> m1: -- Cpus: 2
==> db1: -- Description: Source: /home/peter/Documents/working/AnsibleWebFarm/Vagrantfile
==> m0: Creating domain with the following settings...
==> m6: -- Name: AnsibleWebFarm_m6
==> m2: Creating domain with the following settings...
==> m1: -- CPU topology: sockets=1, cores=2, threads=1
==> m2: -- Name: AnsibleWebFarm_m2
==> db1: -- Domain type: kvm
==> m4: Creating domain with the following settings...
==> db1: -- Cpus: 2
==> m0: -- Name: AnsibleWebFarm_m0
==> m6: -- Description: Source: /home/peter/Documents/working/AnsibleWebFarm/Vagrantfile
==> m1: -- Feature: acpi
==> m2: -- Description: Source: /home/peter/Documents/working/AnsibleWebFarm/Vagrantfile

```

Figura 5: Despliegue del Vagrantfile del proyecto - Definiendo el hardware virtual con QEMU.

Ahora es cuando vagrant arranca las máquinas, le inyecta su clave pública para poder gestionar el Software las máquinas virtuales y controlar su estado.

```

==> db1: Removing insecure key from the guest if it's present...
==> m3: Waiting for machine to boot. This may take a few minutes...
m6:
m6: Vagrant insecure key detected. Vagrant will automatically replace
m6: this with a newly generated keypair for better security.
m3: SSH address: 192.168.121.188:22
m3: SSH username: vagrant
m3: SSH auth method: private key
m3: Warning: Connection refused. Retrying...
m2:
m2: Vagrant insecure key detected. Vagrant will automatically replace
m2: this with a newly generated keypair for better security.
db1: Key inserted! Disconnecting and reconnecting using new SSH key...
==> db1: Machine booted and ready!
==> db1: Setting hostname...

```

Figura 6: Despliegue del Vagrantfile del proyecto - Arranque, inyección SSH y fijando hostname.

Y tras terminar de configurar todo lo relacionado con la máquina, en el terminal nos aparecerá un mensaje como el siguiente indicando que ya podemos conectarnos

```

==> db2: Configuring and enabling network interfaces...
==> db1: Running provisioner: file...
db1: ~/.ssh/id_rsa.pub => ~/.ssh/authorized_keys
==> db1: Running provisioner: shell...
db1: Running: inline script
db1: finished
==> db2: Running provisioner: file...
db2: ~/.ssh/id_rsa.pub => ~/.ssh/authorized_keys
==> db2: Running provisioner: shell...
db2: Running: inline script
db2: finished
==> m2: Configuring and enabling network interfaces...
==> m2: Running provisioner: file...
m2: ~/.ssh/id_rsa.pub => ~/.ssh/authorized_keys
==> m2: Running provisioner: shell...
m2: Running: inline script
m2: finished

```

Figura 7: Despliegue del Vagrantfile del proyecto - Provisionamiento de la clave de SSH personal, cambiando DNS y Gateways.

Quedaría como infraestructura desplegada con Vagrant la que sigue:

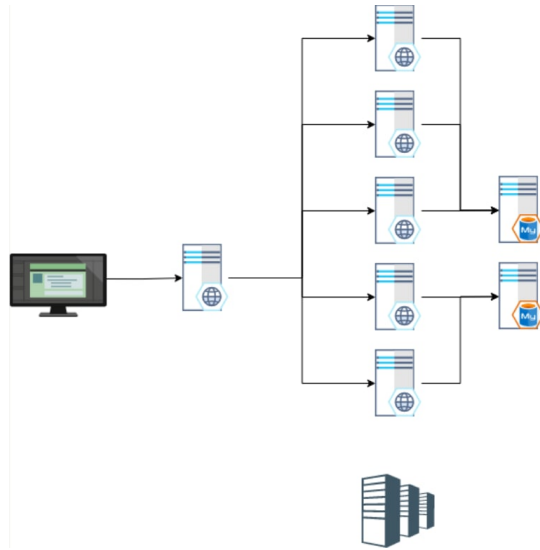


Figura 8: Despliegue del Vagrantfile del proyecto - infraestructura desplegada.

Virtual Machine Manager				
File Edit View Help				
<div> </div>				
Name	CPU usage	Host CPU usage	Memory usage	Disk I/O
▼ QEMU/KVM				
AnsibleWebFarm_db1 Running	<div></div>	<div></div>	<div></div>	<div></div>
AnsibleWebFarm_db2 Running	<div></div>	<div></div>	<div></div>	<div></div>
AnsibleWebFarm_m0 Running	<div></div>	<div></div>	<div></div>	<div></div>
AnsibleWebFarm_m1 Running	<div></div>	<div></div>	<div></div>	<div></div>
AnsibleWebFarm_m2 Running	<div></div>	<div></div>	<div></div>	<div></div>
AnsibleWebFarm_m3 Running	<div></div>	<div></div>	<div></div>	<div></div>
AnsibleWebFarm_m4 Running	<div></div>	<div></div>	<div></div>	<div></div>
AnsibleWebFarm_m5 Running	<div></div>	<div></div>	<div></div>	<div></div>
AnsibleWebFarm_m6 Running	<div></div>	<div></div>	<div></div>	<div></div>

Figura 9: Despliegue del Vagrantfile del proyecto - infraestructura desplegada visualiza-  
ción en Virt-Manager.

### 3. Ansible

Ansible es una herramienta idempotente **asegura el mismo resultado siempre**. Que permite orquestar y configurar desde un simple servidor hasta un datacenter completo, todo realizado a través de SSH que permite que podamos ejecutar las instrucciones u obtener información de monitoreo básica de manera segura.

#### 3.1. Instalación de Ansible en Debian/Ubuntu

Primero necesitamos instalar los repositorios de Ansible antes de proceder a su instalación. Para ello ejecutamos los siguientes comandos:

- 1 \$ `echo "deb http://ppa.launchpad.net/ansible/ansible/ubuntu focal main" |`  
    `sudo tee /etc/apt/sources.list.d/ansible.list`
- 2 \$ `sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 93C4A3FD7BB9C367`
- 3 \$ `sudo apt update`
- 4 \$ `sudo apt install ansible`

Ejecutando los comandos de arriba obtendremos lo siguiente:

```
adrian@debian:~$ sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 93C4A3FD7BB9C367
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
Executing: /tmp/apt-key-gpghome.eWBDLWzrq/gpg.1.sh --keyserver keyserver.ubuntu.com --recv-keys 93C4A3FD7BB9C367
gpg: key 93C4A3FD7BB9C367: public key "Launchpad PPA for Ansible, Inc." imported
gpg: Total number processed: 1
gpg:      imported: 1
adrian@debian:~$ sudo apt update
Hit:1 http://deb.debian.org/debian sid InRelease
Get:2 http://ppa.launchpad.net/ansible/ansible/ubuntu focal InRelease [18.0 kB]
Hit:3 http://packages.microsoft.com/repos/code stable InRelease
Hit:4 https://dl.google.com/linux/chrome/deb stable InRelease
Get:5 http://ppa.launchpad.net/ansible/ansible/ubuntu focal/main amd64 Packages [1.132 B]
Get:6 http://ppa.launchpad.net/ansible/ansible/ubuntu focal/main Translation-en [756 B]
Fetched 19.9 kB in 2s (9,928 B/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
28 packages can be upgraded. Run 'apt list --upgradable' to see them.
W: http://ppa.launchpad.net/ansible/ansible/ubuntu/dists/focal/InRelease: Key is stored in legacy truste
d.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATION section in apt-key(8) for details.
adrian@debian:~$ sudo apt install ansible
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  ansible-core python-babel-localedata python3-babel python3-bcrypt python3-invoke python3-jinja2
  python3-jmespath python3-kerberos python3-nacl python3-ntlm-auth python3-packaging python3-paramiko
  python3-requests-kerberos python3-requests-ntlm python3-resolveib python3-tz python3-winrm
  python3-xlrd python3-yaml sshpass
Suggested packages:
  python-invoke-doc python-jinja2-doc python-nacl-doc python3-gssapi
The following NEW packages will be installed:
  ansible ansible-core python-babel-localedata python3-babel python3-bcrypt python3-invoke
  python3-jinja2 python3-jmespath python3-kerberos python3-nacl python3-ntlm-auth python3-packaging
  python3-paramiko python3-requests-kerberos python3-requests-ntlm python3-resolveib python3-tz
  python3-winrm python3-xlrd python3-yaml sshpass
0 upgraded, 21 newly installed, 0 to remove and 28 not upgraded.
Need to get 22.2 MB/27.8 MB of archives.
After this operation, 352 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Figura 10: Instalación de Ansible.

## 3.2. Inventario

El entorno a controlar se puede definir en un fichero llamado **inventory.yml** que indicamos en las direcciones IP y los hostname para identificar a cada servidor particular. Así como crear grupos con los cuales podemos usar como objetivo de determinadas tareas.

En el proyecto usamos 3 tipos de grupos: webservers, loadbalancer y databases.

Esto nos permite lanzar a todas las direcciones de los contenedores, máquinas virtuales, servidores físicos (on-prem), etc...

Así como podemos agrupar los grupos con hostnames y direcciones en otros grupos padre simbólicos. Esto es útil en el caso de que necesitemos agruparlos para usar Playbooks que resuelvan necesidades específicas como por ejemplo: Realizar copias de seguridad de /var/log/nginx, para almacenar los logs de manera segura para tener un análisis del tráfico de los balanceadores y los servidores web.

**Nota:** Para que ansible pueda identificar el fichero de inventario. Necesitamos usar otro fichero llamado **ansible.cfg** en el cual tenemos que poner como parámetro el inventario que va a usarse para lanzar los Playbooks.

Fichero donde definimos el inventario a utilizar por parte de Ansible *ansible.cfg*

```
1 [ defaults ]
2 inventory = inventory.yml
3 host_key_checking = False
```

Fichero de inventarios. *inventory.yml*

```
1 ---
2   webservers:
3     hosts:
4       m1.swapfinal:
5         ansible_host: 192.168.100.4
6       m2.swapfinal:
7         ansible_host: 192.168.100.5
8       m3.swapfinal:
9         ansible_host: 192.168.100.6
10      m4.swapfinal:
11        ansible_host: 192.168.100.7
12      m5.swapfinal:
13        ansible_host: 192.168.100.8
14      m6.swapfinal:
15        ansible_host: 192.168.100.9
16
17   loadbalancer:
18     hosts:
19       m0.swapfinal:
20         ansible_host: 192.168.100.2
21
22   databases:
23     hosts:
24       m20.swapfinal:
25         ansible_host: 192.168.100.20
26       m21.swapfinal:
27         ansible_host: 192.168.100.21
28
```



```

29     datacenter:
30         children:
31             datacenterweb:
32             databases:
33         vars:
34             ansible_user: vagrant
35             ansible_ssh_private_key_file: ~/.ssh/id_rsa
36
37
38     datacenterweb:
39         children:
40             webserver:
41             loadbalancer:

```

En el fichero definimos desde la raíz del árbol de grupos hasta los hostnames como sigue:

- **datacenter** - Grupo padre de todos que apunta a todos los subgrupos del inventario. Útil para tareas de monitorización y comprobación de estados con un simple ping ejecutado desde comandos de Ansible Ad-hoc.
  - **datacenterweb** - Sub-Grupo hijo de datacenter que tiene de hijos solo a los sub-grupos webserver y loadbalancers.
    - **webserver** - Sub-Grupo hijo de datacenterweb que apunta a los hostnames y direcciones IPs de los servidores web indexados con Ansible.
    - **loadbalancer** - Sub-Grupo hijo de datacenterweb que apunta a los hostnames y direcciones IPs de los balanceadores de carga web indexados con Ansible.
  - **databases** - Sub-Grupo hijo de datacenter que apunta a los hostnames y direcciones IPs de los servidores SGBD.

### 3.3. Ansible-playbooks

Los Playbooks son archivos de *orquestración* que nos permiten modularizar el aprovisionamiento y ejecución.

La estructura del Playbook empieza con la definición del grupo, o a un hostname concreto (un servidor) al que afecta las tareas ejecutadas y a continuación se define variables o no. Lo importante del Playbook, es que podemos definir un Playbook de tal manera que nos permita realizar un objetivo final concreto que requiere de un sin número de pasos, o tareas, como desplegar una base de datos y asegurarla. Pues en una terminal sería como sigue:

```

1 $ sudo apt-get install mariadb-server -y
2 $ sudo systemctl enable mariadb-server.service
3 $ sudo systemctl start mariadb-server.service
4 $ sudo mysql_secure_installation

```

¿Podemos automatizarlo con un script? Sí, pero si ocurren casos como que, mariadb esté instalado o el servicio no se ejecute bien en un sistema en concreto, el script dejaría de

funcionar o necesitamos destinar recursos para programar una ejecución de errores que sería muy concreto y poco portable a otras plataformas.

Además de que tendríamos que diseñar el script para apuntar a los servidores o máquinas concretas, complicando el diseño ya que necesitaremos una estructura de datos concreta que permita indexar los hostnames, servidores y control de errores. Entonces si tenemos 40 servidores web, nos encontramos que en caso de fallo, esos servidores podrían quedarse inutilizados. O que la instalación prosiga pero en un estado erróneo sin informarnos debidamente.

Todo eso se soluciona con un Playbook que funciona de manera idempotente y que con solo pequeños cambios podemos reutilizarlo para otra distribución Red Hat style o Debian style ...

Veremos el caso de Mariadb en la siguiente subsección

### 3.4. Playbooks usados para el proyecto y para qué sirven.

#### 3.4.1. Playbook de hosts.yml

Este Playbook ha sido diseñado de tal manera que será el padre de todos los playbooks a los que va llamando sucesivamente. Pero podemos comentar las líneas de `import _playbook` para que solo se ejecute simplemente el propósito de *hosts.yml*.

Su propósito es el de a partir de la plantilla localizada en **templates/hosts.j2**. Que contiene lo siguiente:

```
1 # {{ ansible_managed }}
2 127.0.0.1    localhost localhost.localdomain localhost4 localhost4.
   localhost4
3 ::1         localhost localhost.localdomain localhost6 localhost6.
   localhost6
4
5 {% for host in groups['datacenter'] %}
6 {{ hostvars[host]['ansible_host'] }} {{ hostvars[host]['inventory_hostname']
   }}
7 {% endfor %}
```

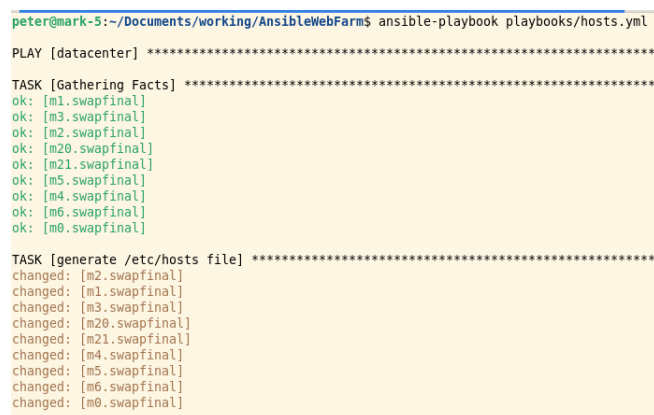
En el template podemos ver que se genera una línea de host por cada variable encontrada (denominado también Facts de Ansible) en el fichero `inventory.yml` de Ansible que ejecuta el Playbook. Lo cual es muy útil para obtener de manera dinámica todos los hostnames asociados a su dirección IP sin tener que realizar el proceso manual de añadir un servidor nuevo a mano a cada fichero `hosts`. Si no tenemos un DNS.

Se puede comentar las líneas de incluir otros Playbooks. Permitiendo que se ejecute de manera individual la tarea, para generar ficheros de `hosts` en el caso de que se añada al inventario muchos más servidores. Con ello nos aseguramos de que cada servidor va a tener su fichero `hosts` siempre actualizado.

```

1  ---
2
3  - hosts: datacenter
4    become: true
5
6    tasks:
7      - name: generate /etc/hosts file
8        ansible.builtin.template:
9          src: ../templates/hosts.j2
10         dest: /etc/hosts
11         owner: root
12         group: root
13         mode: '0644'
14
15 - name: Running SSL Playbook
16   import_playbook: SSL.yml
17
18 - name: Running install servers NGINX Playbook
19   import_playbook: webservers.yml
20
21 - name: Running install loadbalancers NGINX Playbook
22   import_playbook: loadbalancer.yml
23
24 - name: Running install databases Playbook
25   import_playbook: databases.yml

```



```

peter@mark-5:~/Documents/working/AnsibleWebFarm$ ansible-playbook playbooks/hosts.yml
PLAY [datacenter] *****

TASK [Gathering Facts] *****
ok: [m1.swapfinal]
ok: [m3.swapfinal]
ok: [m2.swapfinal]
ok: [m20.swapfinal]
ok: [m21.swapfinal]
ok: [m5.swapfinal]
ok: [m4.swapfinal]
ok: [m6.swapfinal]
ok: [m0.swapfinal]

TASK [generate /etc/hosts file] *****
changed: [m2.swapfinal]
changed: [m1.swapfinal]
changed: [m3.swapfinal]
changed: [m20.swapfinal]
changed: [m21.swapfinal]
changed: [m4.swapfinal]
changed: [m5.swapfinal]
changed: [m6.swapfinal]
changed: [m0.swapfinal]

```

Figura 11: Generación de los ficheros `/etc/hosts` en el Playbook.

El funcionamiento básico del playbook es:

- Sección **PLAY [datacenter]**: Indica en qué grupos del inventario se está aplicando el Playbook. En este caso se aplica para todo el mundo ya que datacenter es el grupo padre de todos los grupos.
- Sección **TASK [Gathering Facts]**: Es la sección donde Ansible asegura que los hosts referenciados están disponibles, si falla uno no aplica el Playbook. Ya que debe ser idempotente, es decir que se aplica a todo o a nada.

- Sección **TASK [generate /etc/hosts file]**: Es la tarea en la que se especializa el Playbook que genera a partir de la plantilla y lo lleva a todo el grupo llamado datacenter. El estado de changed indica que se ha aplicado los cambios de manera correcta.

```
peter@mark-5:~/Documents/working/AnsibleWebFarm$ ansible-playbook playbooks/hosts.yml
PLAY [datacenter] *****

TASK [Gathering Facts] *****
ok: [m1.swapfinal]
ok: [m3.swapfinal]
ok: [m2.swapfinal]
ok: [m20.swapfinal]
ok: [m21.swapfinal]
ok: [m5.swapfinal]
ok: [m4.swapfinal]
ok: [m6.swapfinal]
ok: [m0.swapfinal]

TASK [generate /etc/hosts file] *****
changed: [m2.swapfinal]
changed: [m1.swapfinal]
changed: [m3.swapfinal]
changed: [m20.swapfinal]
changed: [m21.swapfinal]
changed: [m4.swapfinal]
changed: [m5.swapfinal]
changed: [m6.swapfinal]
changed: [m0.swapfinal]
```

Figura 12: Generación de los ficheros /etc/hosts en el Playbook.

```
peter@mark-5:~/Documents/working/AnsibleWebFarm$ ssh vagrant@192.168.100.21
Last login: Fri Jun 17 10:49:35 2022 from 192.168.100.1
[vagrant@m21 ~]$ cat /etc/host
cat: /etc/host: No such file or directory
[vagrant@m21 ~]$ cat /etc/hosts
# Ansible managed
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6

192.168.100.20 m20.swapfinal
192.168.100.21 m21.swapfinal
192.168.100.4 m1.swapfinal
192.168.100.5 m2.swapfinal
192.168.100.6 m3.swapfinal
192.168.100.7 m4.swapfinal
192.168.100.8 m5.swapfinal
192.168.100.9 m6.swapfinal
192.168.100.2 m0.swapfinal
[vagrant@m21 ~]$
```

Figura 13: Ficheros de /etc/hosts generados en los servidores de base de datos como ejemplo, en este caso es m20-swapfinal.

### 3.4.2. Playbook de SSL.yml

Este Playbook puede parecer muy largo pero es sencillo. Necesita la colección de **community.crypto**. Que es una serie de colecciones que nos permite realizar determinadas acciones que no se hacen de manera nativa en Ansible. Si no que son extendidas por la propia comunidad de usuarios.

Esta colección nos permite realizar muchas opciones desde generar una CA hasta un certificado X.509, con Ansible. Pudiendo como en mi caso hacer un Playbook que especificando una serie de tareas, podemos replicar todo el proceso necesario para generar una clave, generar un CSR Certificate Signed Request, generar el Certificado Firmado. A su vez incluir la firma de la CA y incluir su Certificado para validar el certificado del dominio.

Para instalar la colección necesitamos ejecutar lo siguiente:

```
1 # Instalacion de la coleccion de crypto para crear CA
2 $ ansible-galaxy collection install community.crypto
```

**Nota:** Todo este proceso lo realizamos en un balanceador en concreto, en mi caso es m0-swapfinal. Esto es debido a que no necesitamos generar N claves y certificados, si lo ejecutamos en un grupo en concreto como loadbalancer o webservers. Entonces lo que hacemos es especificar el hostname de un balanceador de carga que va a asumir el proceso entero.

- La primera parte es que instala las dependencias necesarias para que Ansible pueda trabajar con la colección ya que usa paquetes de Python. En concreto el paquete cryptography.
- La segunda parte son todas las tareas que se requieren para generar una CA y sus certificados.
- La tercera parte son todas las tareas que se requieren para generar el certificado de swapfinal y firmarlo con la CA.
- La cuarta parte es el proceso de copia de todos los ficheros CRT y KEY menos el de la CA.key. Al ordenador del administrador.
- La última parte es la ejecución de otro Playbook que permite la copia de los certificados y claves copiados a cada uno de los servidores web. Que son necesarios para poder configurar su HTTPs.

```

PLAY [m0.swapfinal] *****

TASK [Gathering Facts] *****
ok: [m0.swapfinal]

TASK [Installing pip3 package] *****
changed: [m0.swapfinal]

TASK [Installing required pip3 package cryptography] *****
changed: [m0.swapfinal]

TASK [Creando certificados SSL - CA File Private Key] *****
changed: [m0.swapfinal]

TASK [Creando certificados SSL - CSR Certificate Signed Request CA] *****
changed: [m0.swapfinal]

TASK [Creando certificados CA desde el CSR] *****
changed: [m0.swapfinal]

TASK [Creando certificados SSL - mswap-swapfinal Key] *****
changed: [m0.swapfinal]

TASK [Creando certificados SSL - mswap-swapfinal Cert] *****
changed: [m0.swapfinal]

TASK [Creando Certificado SSL Firmando certificado SSL con CA mswap-swapfinal] *****
changed: [m0.swapfinal]

TASK [Fetch and Pull Cert CA to local machine] *****
changed: [m0.swapfinal]

TASK [Fetch and Pull Key Servers to local machine] *****
changed: [m0.swapfinal]

TASK [Fetch and Pull Cert Servers to local machine] *****
changed: [m0.swapfinal]

```

Figura 14: Ejecución de la parte de creación de los certificados SSL y claves de CA y swapfinal.

Ahora podemos ver el proceso de copiado de las claves y certificados necesarios para los servidores web. Se puede ver que la sección **PLAY [webserver]**. Indica que se está ejecutando en el grupo perteneciente a los servidores web.

```

PLAY [webserver] *****

TASK [Gathering Facts] *****
ok: [m3.swapfinal]
ok: [m4.swapfinal]
ok: [m1.swapfinal]
ok: [m2.swapfinal]
ok: [m5.swapfinal]
ok: [m6.swapfinal]

TASK [Push Cert CA to local machine] *****
changed: [m2.swapfinal]
changed: [m1.swapfinal]
changed: [m4.swapfinal]
changed: [m3.swapfinal]
changed: [m5.swapfinal]
changed: [m6.swapfinal]

TASK [Push Key Servers to local machine] **
changed: [m3.swapfinal]
changed: [m2.swapfinal]
changed: [m1.swapfinal]
changed: [m4.swapfinal]
changed: [m5.swapfinal]
changed: [m6.swapfinal]

TASK [Push Cert Servers to local machine] *
changed: [m4.swapfinal]
changed: [m2.swapfinal]
changed: [m1.swapfinal]
changed: [m3.swapfinal]
changed: [m5.swapfinal]
changed: [m6.swapfinal]

```

Figura 15: Ejecución de la parte de copiado de los certificados y claves necesarias para los servidores web.

[illegible]

Figura 16: Ficheros correctamente copiados.

### 3.4.3. Playbook de webservers.yml

Este playbook tiene varios objetivos:

- La primera parte es instalar Nginx, PHP y PHP-fpm necesario para ofrecer una web más dinámica que ofrezca una vista de lo que ocurre cuando solicites el index.php al balanceador.
- La segunda parte es el proceso de copia del fichero de configuración del sitio por defecto. Que debemos definir como plantilla, para exportarla y aplicarla a todos los servidores.
- La tercera parte es el proceso de copia del index.php base de plantilla, que debe ser llevado a todos los servidores. A /var/www/html/.
- La cuarta parte una vez terminado el proceso de copia anterior, reiniciamos, el servicio para que aplique los cambios del sitio por defecto y que arranque PHP. Además de fijarlo como si ejecutamos **sudo systemctl enable nginx.service**. Para que en cada reinicio siempre esté activado.

El playbook en cuestión es el siguiente:

```
1 ---
2 - hosts: webservers
3   become: yes
4
5   tasks:
6     - name: Install Nginx
7       ansible.builtin.apt:
8         name: nginx
9         state: present
10        update_cache: yes
11
12     - name: Install php
13       ansible.builtin.apt:
14         name: php
15         state: present
16
17     - name: Install php7.4-fpm
18       ansible.builtin.apt:
19         name: php-fpm
20         state: present
21
22     - name: Obtener el estado de Nginx y fijando su demonio en autoarranque
23       ansible.builtin.systemd:
24         name: nginx.service
25         state: started
26         enabled: yes
27
28     - name: Push site configuration of NGINX
29       copy:
30         src: ../templates/defaultsite.conf
```



```

31     dest: /etc/nginx/sites-available/default
32     owner: root
33     group: root
34     mode: '0644'
35     backup: yes
36
37 - name: Push index.php of NGINX
38   copy:
39     src: ../templates/index.php
40     dest: /var/www/html/index.php
41     owner: root
42     group: root
43     mode: '0644'
44
45 - name: Restarting nginx webserver
46   ansible.builtin.systemd:
47     name: nginx
48     state: restarted

```

Para los ficheros de index.php tenemos una lectura de la variable de `$_SERVER['SERVER_ADDR']`. Que como se ha especificado antes nos permite identificar que se hagan peticiones en Round Robin con weight 1 a los servidores web.

```

1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <title><?php echo $_SERVER['SERVER_ADDR'] ?></title>
6   </head>
7   <body>
8     <h1>Estamos en el servidor con IP <?php echo $_SERVER['SERVER_ADDR'] ?></h1>
9   </body>
10 </html>

```

```

PLAY [webservers] *****

TASK [Gathering Facts] *****
ok: [m2.swapfinal]
ok: [m4.swapfinal]
ok: [m1.swapfinal]
ok: [m5.swapfinal]
ok: [m3.swapfinal]
ok: [m6.swapfinal]

TASK [Install Nginx] *****
changed: [m4.swapfinal]
changed: [m2.swapfinal]
changed: [m1.swapfinal]
changed: [m3.swapfinal]
changed: [m5.swapfinal]
changed: [m6.swapfinal]

TASK [Install php] *****
changed: [m4.swapfinal]
changed: [m2.swapfinal]
changed: [m3.swapfinal]
changed: [m1.swapfinal]
changed: [m5.swapfinal]
changed: [m6.swapfinal]

TASK [Install php7.4-fpm] *****
changed: [m4.swapfinal]
changed: [m1.swapfinal]
changed: [m2.swapfinal]
changed: [m5.swapfinal]
changed: [m6.swapfinal]
changed: [m3.swapfinal]

TASK [Obtener el estado de Nginx y fijando su demonio en autoarranque]
ok: [m1.swapfinal]
ok: [m4.swapfinal]
ok: [m3.swapfinal]

```

Figura 17: Proceso de ejecución del playbook webservers.yml

```

TASK [Obtener el estado de Nginx y fijando su demonio en autoarranque]
ok: [m1.swapfinal]
ok: [m4.swapfinal]
ok: [m3.swapfinal]
ok: [m5.swapfinal]
ok: [m2.swapfinal]
ok: [m6.swapfinal]

TASK [Push site configuration of NGINX] *****
changed: [m5.swapfinal]
changed: [m3.swapfinal]
changed: [m2.swapfinal]
changed: [m1.swapfinal]
changed: [m4.swapfinal]
changed: [m6.swapfinal]

TASK [Push index.php of NGINX] *****
changed: [m2.swapfinal]
changed: [m3.swapfinal]
changed: [m1.swapfinal]
changed: [m4.swapfinal]
changed: [m5.swapfinal]
changed: [m6.swapfinal]

TASK [Restarting nginx web servers] *****
changed: [m1.swapfinal]
changed: [m2.swapfinal]
changed: [m4.swapfinal]
changed: [m3.swapfinal]
changed: [m5.swapfinal]
changed: [m6.swapfinal]

```

Figura 18: Proceso de ejecución del playbook webservers.yml, segunda parte consistente en fijar el demonio, configurarlo y fijar sus .

```
vagrant@ml:~$ cat /var/www/html/index.php
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title><?php echo $_SERVER['SERVER_ADDR'] ?></title>
  </head>
  <body>
    <h1>Estamos en el servidor con IP <?php echo $_SERVER['SERVER_ADDR'] ?></h1>
  </body>
</html>
```

Figura 19: Fichero index.php correctamente copiado.

### 3.4.4. Playbook de loadbalancer.yml

**Recordemos:** Se está utilizando NGINX como balanceador de carga. Este playbook tiene varios objetivos:

- La primera parte es instalar Nginx y preparando el autoarranque del demonio de Nginx.
- La segunda parte la desactivación de la web por defecto que incluye. Usando un comando de bash sencillo a través de Ansible y que además nos asegura la correcta ejecución de este.
- La tercera parte es la generación del template del fichero de configuración del balanceador de carga. Esto es debido a que hay un número variable de servidores web, nos encontramos con que necesitamos el template .j2 que nos permita indicar en la parte de upstream los servidores web a los que debe hacer de balanceador de carga con el indicador del grupo de webserver.
- La cuarta parte una vez terminado el proceso de copia anterior, reiniciamos, el servicio para que aplique los cambios al balanceador.

El playbook del balanceador es el siguiente:

```
1  ---
2  - hosts: loadbalancer
3    become: yes
4
5    tasks:
6      - name: Install Nginx as Load Balancer
7        ansible.builtin.apt:
8          name: nginx
9          state: present
10         update_cache: yes
11
12      - name: Obtener el estado de Nginx y fijando su demonio en autoarranque
13        ansible.builtin.systemd:
14          name: nginx
15          state: started
16          enabled: yes
17
18      - name: Desactivando la web por defecto
19        command:
20          cmd: unlink /etc/nginx/sites-enabled/default
21          ignore_errors: yes
22
23      - name: generate dynamic /etc/nginx/nginx.conf
24        ansible.builtin.template:
25          src: ../templates/lbnginx.j2
26          dest: /etc/nginx/nginx.conf
27          owner: root
```

```

28     group: root
29     mode: '0644'
30
31 - name: Restarting nginx loadbalancer
32   ansible.builtin.systemd:
33     name: nginx
34     state: restarted

```

El template del fichero de configuración del balanceador es el siguiente:

```

1  # {{ ansible_managed }}
2
3  user www-data;
4  worker_processes auto;
5  pid /run/nginx.pid;
6  include /etc/nginx/modules-enabled/*.conf;
7
8  events {
9      worker_connections 768;
10 }
11
12 http {
13     log_format upstreamlog 'Client_addr: $remote_addr - Server_addr:
        $upstream_addr - Response_time_latency: $upstream_response_time -
        Response_status: $upstream_status';
14
15     upstream upstreamswapfinal {
16         {% for host in groups['webservers'] %}
17         server {{ hostvars[host]['ansible_host'] }}:443;
18         {% endfor %}
19     }
20
21     server {
22         listen 80;
23         server_name swapfinal;
24
25         return 301 https://swapfinal;
26     }
27
28     server {
29         listen 443 ssl;
30
31         ssl_certificate /etc/ssl/certs/mswap.crt;
32         ssl_certificate_key /etc/ssl/private/mswap.key;
33
34         ssl_protocols TLSv1.2 TLSv1.3;
35         ssl_prefer_server_ciphers on;
36
37         location / {
38             proxy_pass https://upstreamswapfinal;
39         }
40

```

```

41     access_log /var/log/nginx/upstream.log upstreamlog;
42 }
43
44 }

```

La parte variable es un bucle sencillo similar al de la plantilla de hosts. Pero con la particularidad de que apuntamos a un grupo distinto al que ejecuta el playbook (webservers).

```

PLAY [loadbalancer] *****
TASK [Gathering Facts] *****
ok: [m0.swapfinal]

TASK [Install Nginx as Load Balancer] *****
changed: [m0.swapfinal]

TASK [Obtener el estado de Nginx y fijando su demonio en autoarranque] *
ok: [m0.swapfinal]

TASK [Desactivando la web por defecto] *****
changed: [m0.swapfinal]

TASK [generate /etc/nginx/nginx.conf dinámico] *****
changed: [m0.swapfinal]

TASK [Restarting nginx loadbalancer] *****
changed: [m0.swapfinal]

```

Figura 20: Ejecución del PlayBook de loadbalancer.

```

vagrant@m0:~$ cat /etc/nginx/nginx.conf
# Ansible managed

user www-data;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;

events {
    worker_connections 768;
}

http {
    log_format upstreamlog 'Client_addr: $remote_addr - Server_addr: $server_addr, $status, $request_time upstream: $upstream_addr';
    log_format upstreamlog 'Client_addr: $remote_addr - Server_addr: $server_addr, $status, $request_time upstream: $upstream_addr';
    upstream upstreamswapfinal {
        server 192.168.100.4:443;
        server 192.168.100.5:443;
        server 192.168.100.6:443;
        server 192.168.100.7:443;
        server 192.168.100.8:443;
        server 192.168.100.9:443;
    }

    server {
        listen 80;
        server_name swapfinal;

        return 301 https://swapfinal;
    }

    server {
        listen 443 ssl;

        ssl_certificate /etc/ssl/certs/mswap.crt;
        ssl_certificate_key /etc/ssl/private/mswap.key;

        ssl_protocols TLSv1.2 TLSv1.3;
        ssl_prefer_server_ciphers on;

        location / {
            proxy_pass https://upstreamswapfinal;
        }

        access_log /var/log/nginx/upstream.log upstreamlog;
    }
}

```

Figura 21: Fichero de configuración generado solo con los webservers.

### 3.4.5. Playbook de databases.yml

Este playbook tiene varios objetivos:

- La primera parte es instalar Mariadb como SGBD.
- La segunda parte es introducir un .my.cnf que incluye una clave por defecto para que permita en el usuario vagrant que es el que se usa para conectarse con Ansible y Vagrant. El acceso root de mysql.
- La tercera parte es añadir PyMySQL que es el driver que permite las conexiones con el SGBD a través de Python que es necesario para ejecutar la collection de community.MySQL.
- La cuarta parte es el proceso de mysql\_secure\_installation imitado en un Ansible Playbook, porque el original es un script de bash interactivo. Entonces aquí hacemos todo el proceso integrado con Ansible para que se ejecute en los N servidores SGBD.

Para instalar la colección de Mysql necesaria para realizar el proceso.

```
1  # Instalacion de la coleccion de crypto para crear CA
2  $ ansible-galaxy collection install community.mysql
```

El playbook de los SGBD es el siguiente:

```
1  ---
2  - hosts: loadbalancer
3    become: yes
4
5    tasks:
6      - name: Install Nginx as Load Balancer
7        ansible.builtin.apt:
8          name: nginx
9          state: present
10         update_cache: yes
11
12      - name: Obtener el estado de Nginx y fijando su demonio en autoarranque
13        ansible.builtin.systemd:
14          name: nginx
15          state: started
16          enabled: yes
17
18      - name: Desactivando la web por defecto
19        command:
20          cmd: unlink /etc/nginx/sites-enabled/default
21          ignore_errors: yes
22
23      - name: generate dynamic /etc/nginx/nginx.conf
24        ansible.builtin.template:
25          src: ../templates/lbnginx.j2
26          dest: /etc/nginx/nginx.conf
27          owner: root
```

```

28     group: root
29     mode: '0644'
30
31 - name: Restarting nginx loadbalancer
32   ansible.builtin.systemd:
33     name: nginx
34     state: restarted

```

El template del fichero my.cnf:

```

1 [ client ]
2 user=root
3 password="pass "
4
5 [ mysql ]
6 user=root
7 password="pass "
8
9 [ mysqldump ]
10 user=root
11 password="pass "
12
13 [ mysqldiff ]
14 user=root
15 password="pass "

```

```

PLAY [databases] *****

TASK [Gathering Facts] *****
ok: [m21.swapfinal]
ok: [m20.swapfinal]

TASK [Install MariaDB server] *****
changed: [m21.swapfinal]
changed: [m20.swapfinal]

TASK [Push .my.cnf with root password] *****
changed: [m21.swapfinal]
changed: [m20.swapfinal]

TASK [Obtener el estado de MariaDB y fijando su demonio en autoarranque] *****
changed: [m20.swapfinal]
changed: [m21.swapfinal]

TASK [Adds Python MySQL support on RedHat Platforms - mysql_secure_installation] **
changed: [m21.swapfinal]
changed: [m20.swapfinal]

TASK [update mysql root password for root account] *****
changed: [m21.swapfinal]
changed: [m20.swapfinal]

TASK [Deletes anonymous MySQL server user for ansible_fqdn] *****
ok: [m20.swapfinal]
ok: [m21.swapfinal]

TASK [Deletes anonymous MySQL server user for localhost] *****
ok: [m20.swapfinal]
ok: [m21.swapfinal]

TASK [Secures the MySQL root user for IPV6 localhost (::1)] *****
changed: [m20.swapfinal]
changed: [m21.swapfinal]

TASK [Secures the MySQL root user for IPV4 localhost (127.0.0.1)] *****
changed: [m21.swapfinal]
changed: [m20.swapfinal]

TASK [Secures the MySQL root user for localhost domain (localhost)] *****
changed: [m20.swapfinal]
changed: [m21.swapfinal]

```

Figura 22: Proceso de instalación del SGBD y mysql\_secure\_installation.



## 4. Conclusiones

Como conclusión es que Ansible es una herramienta que ha sido tan potente que nos ha simplificado el proceso de despliegue de un par de días a minutos. La única parte a tener en consideración es que no te libera de tener que montar y investigar manualmente una instalación previa. Para ir probando los ajustes que quieras tener de cara a un despliegue dinámico de la infraestructura.

Esto es con NGINX como ejemplo. La instalación es nativa con Ansible, pero la configuración y las plantillas relacionadas con NGINX requieren un conocimiento previo necesario. Porque de lo contrario Ansible más que ayudar solo estorbaría al no conocer el background que tiene el despliegue.

Ansible es una navaja suiza, pero antes debes conocer que hace cada parte de la navaja suiza y cómo aplicarla en según qué casos. En el caso de NGINX si hacemos la parte manual sabemos qué ficheros de configuración tocar y qué partes son variables a ajustar con plantillas con los hosts de Ansible.

En el futuro se seguirá usándola y mejorándose. En mi repositorio se publicarán las actualizaciones con Ansible. Se mejorará con Roles propios y mejor escalabilidad.



Figura 23: Balanceador en ejecución.

Firefox

about:certificate?cert=MlIFQjCCAyqgAwIBAgIUcLT6OE4iAesYsPjuoGvQ2Oo6o%2FwwDQYJKo

80%

☆

SWAP CA	
<b>Subject Name</b>	
Country	ES
Organization	SWAP
Common Name	SWAP CA
<b>Issuer Name</b>	
Country	ES
Organization	SWAP
Common Name	SWAP CA
<b>Validity</b>	
Not Before	Thu, 16 Jun 2022 10:45:34 GMT
Not After	Sat, 17 Jun 2023 10:45:34 GMT
<b>Subject Alt Names</b>	
DNS Name	swapfinal
<b>Public Key Info</b>	
Algorithm	RSA
Key Size	4096
Exponent	65537
Modulus	B4:CE:25:DB:72:7B:0C:2F:BD:FF:9F:71:B2:56:D4:DC:F3:D5:28:71:32:30:7B:4F...

Figura 24: Certificado de swapfinal.

## 5. Bibliografía

- <sup>1</sup> <https://www.vagrantup.com/downloads>
- <sup>2</sup> <https://github.com/vagrant-libvirt/vagrant-libvirt>
- <sup>3</sup> [https://docs.ansible.com/ansible/latest/installation\\_guide/intro\\_installation.html#installing-ansible-on-debian](https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html#installing-ansible-on-debian)
- <sup>4</sup> <https://docs.ansible.com/ansible/latest/collections/community/crypto/index.html>
- <sup>5</sup> <https://docs.ansible.com/ansible/latest/collections/community/mysql/index.html>
- <sup>6</sup> <https://github.com/lovelace9981/MemoriasInvestigacionSWAP>
- <sup>7</sup> <https://github.com/lovelace9981/AnsibleWebFarm>

---

<sup>1</sup>Instalación de Vagrant

<sup>2</sup>Vagrant Libvirt provider

<sup>3</sup>Instalación de Ansible

<sup>4</sup>Referencia y instalación de community.Crypto para crear certificados CA y para HTTPs

<sup>5</sup>Referencia y instalación de community.Mysql para crear gestionar bases de datos con Ansible

<sup>6</sup>Enlace al repositorio de las memorias necesarias para comprender el background completo

<sup>7</sup>Enlace al repositorio del proyecto completo (se pondrá a público en cuanto se sepa la nota y termine la convocatoria)