Cristóbal Suárez Abad

Optativa-Fundamentos de DevOps para Administradores
2º ASIR
Examen Vagrant

Parte 1: Preguntas Conceptuales

Responde brevemente en un documento de texto.

1. Explica la diferencia entre un **box** y una **máquina virtual** en ejecución.

Un box es un paquete que contiene una plantilla de un sistema operativo, está preconfigurado para poder funcionar en cualquier ambiente.

Una máquina virtual en ejecución es el sistema operativo que se crea cuando ejecutamos un Vagrantfile con nuestras especificaciones. Usamos el box como plantilla para crearlo.

2. En la configuración de redes, ¿cuál es la diferencia principal entre una red **"private_network"** y una red **"public_network" (bridged)**?

La principal diferencia es que en la "bridged" la VM se presenta en la red como si fuese un equipo más: usa una IP de la misma LAN que el host. Mientras que en la privada se crea una subred distinta a la que tiene el Host.

3. ¿Para qué sirve el comando `vagrant provision` y cuándo es necesario ejecutarlo manualmente?

Se usa para aplicar las "provision" que configuramos en el Vagrantfile. Se ejecuta cuando la VM ya está encendida y queremos aplicar la "provision" sin tener que reiniciar la VM. Si apagamos la VM y la volvemos a encender, las "provision" configuradas se aplicarán con el nuevo encendido.

4. Si configuras `config.vm.synced_folder`, ¿qué esperas que ocurra entre tu sistema anfitrión (host) y la máquina virtual (guest)?

Se creará una carpeta sincronizada que permitirá compartir documentos entre el Anfitrión (Host) y el Huésped (VM).

Parte 2: Escenario Práctico

Escenario: Eres el administrador de sistemas y necesitas entregar a los desarrolladores un entorno local que simule producción. Debes crear un **único archivo `Vagrantfile`** que levante dos máquinas virtuales: `ServidorWeb` y `BaseDatos`.

Ejercicio 1: Definición del Entorno (Multi-machine)

Crea un directorio llamado `Examen_Vagrant` e inicializa un entorno.

Generamos el archivo base "Vagrantfile".

```
C:\Users\UsuarioASIR\Documents\examen_vagrant>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.

C:\Users\UsuarioASIR\Documents\examen_vagrant>
```

Configura el `Vagrantfile` para definir dos máquinas:

- **Máquina 1:** Nombre lógico `web01`. Debe usar un box de Ubuntu (ej. `ubuntu/jammy64` o `trusty64`).

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

  # --------------------------
  # MÁQUINA: WEB01
  # --------------------------
  config.vm.define "web01" do |db|
    db.vm.box = "ubuntu/jammy64"
    db.vm.hostname = "web01"
```

- **Máquina 2:** Nombre lógico `db01`. Debe usar un box de Ubuntu.

```
# -----------------------------
# MÁQUINA: DB01
# -----------------------------
config.vm.define "db01" do |db|
  db.vm.box = "ubuntu/jammy64"
  db.vm.hostname = "db01"
```

Ejercicio 2: Gestión de Recursos

Los desarrolladores tienen recursos limitados en sus portátiles. Configura la máquina `db01` para que utilice especificaciones reducidas a través de VirtualBox:

- **RAM:** 512 MB.
- **CPUs:** 1.

```
# CONFIGURACIÓN RECURSOS HARDWARE
config.vm.provider "virtualbox" do |vb|
# # Display the VirtualBox GUI when booting the machine
# vb.gui = true
#
# # Customize the amount of memory on the VM:
vb.memory = "512"
vb.cpus = 1
end
```

Una vez levantada la máquina, podemos comprobarlo con:

CPU: **grep processor /proc/cpuinfo | wc -l**

```
vagrant@db01:~$ grep processor /proc/cpuinfo | wc -l
1
```

RAM: **grep MemTotal /proc/meminfo**

```
vagrant@db01:~$ grep MemTotal /proc/meminfo
MemTotal:         465064 kB
vagrant@db01:~$
```

Ejercicio 3: Configuración de Redes

- **Red Privada:** Ambas máquinas deben estar en una red privada para comunicarse entre ellas de forma segura.
    - Asigna la IP `192.168.50.10` a `web01`.

```
# --------------------------
# MÁQUINA: WEB01
# --------------------------
config.vm.define "web01" do |db|
  db.vm.box = "ubuntu/jammy64"
  db.vm.hostname = "web01"
  config.vm.network "private_network", ip: "192.168.50.10"
```

    - Asigna la IP `192.168.50.20` a `db01`.

```
# --------------------------
# MÁQUINA: DB01
# --------------------------
config.vm.define "db01" do |db|
  db.vm.box = "ubuntu/jammy64"
  db.vm.hostname = "db01"
  config.vm.network "private_network", ip: "192.168.50.20"
```

- **Port Forwarding:** Necesitamos ver la web desde nuestro navegador. Configura `web01` para reenviar el puerto **80** del invitado (guest) al puerto **8080** de tu máquina anfitriona (host).

```
#PORT FORWARDING
config.vm.network "forwarded_port", guest: 80, host: 8080, auto_correct: true
```

He usado "**auto_correct: true**" porque la primera vez que levanté el Vagrantfile, me dio un error diciendo que entraría en conflicto con otros servicios del Host. Sin embargo, cuando he ido a comprobar si funcionaba, ha usado el puerto 8080.

http://localhost:8080/

Ejercicio 4: Aprovisionamiento (Provisioning)

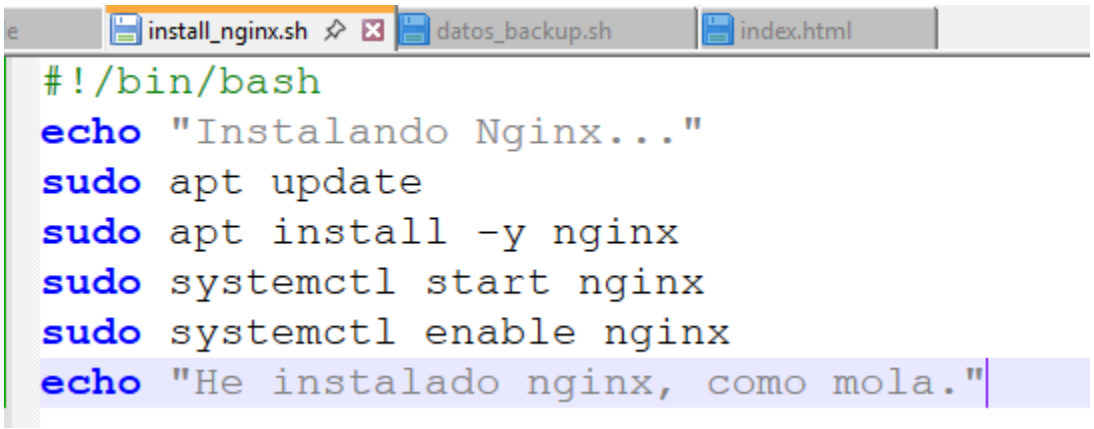
Automatiza la instalación del software necesario para que, al hacer `vagrant up`, el software ya esté listo.

- **En `web01`:** Crea un script de shell (inline o archivo externo) que actualice los repositorios e instale el servidor **Nginx** (o Apache).

La provisión en el Vagrantfile.

```
#Provision para instalar NGINX
config.vm.provision "shell", path: "install_nginx.sh"
```

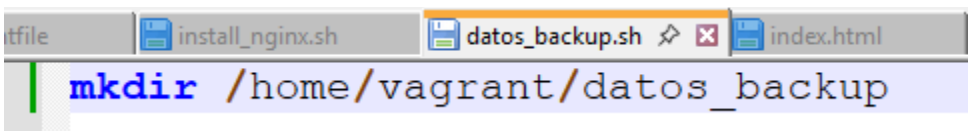
El script para instalar nginx:



- **En `db01`:** Crea un script de provisionamiento que simplemente cree una carpeta llamada `/home/vagrant/datos_backup`.

```
#Carpeta nueva - Provision
config.vm.provision "shell", path: "datos_backup.sh"
```

El script.



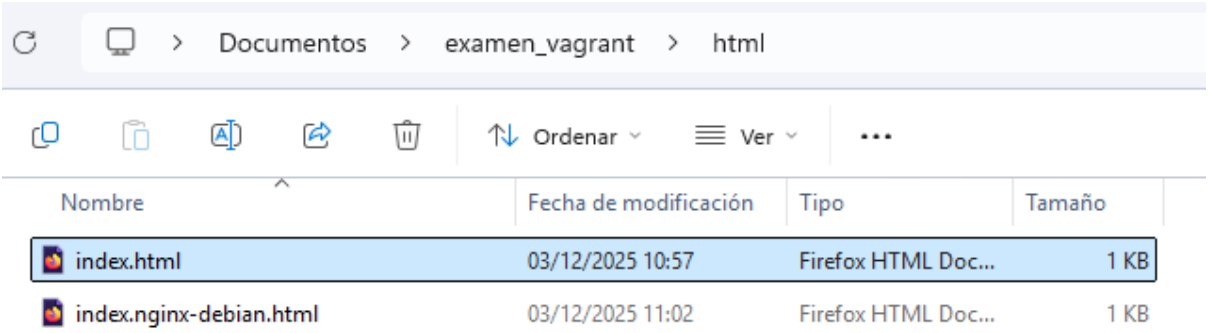
Ejercicio 5: Carpetas Sincronizadas y Validación

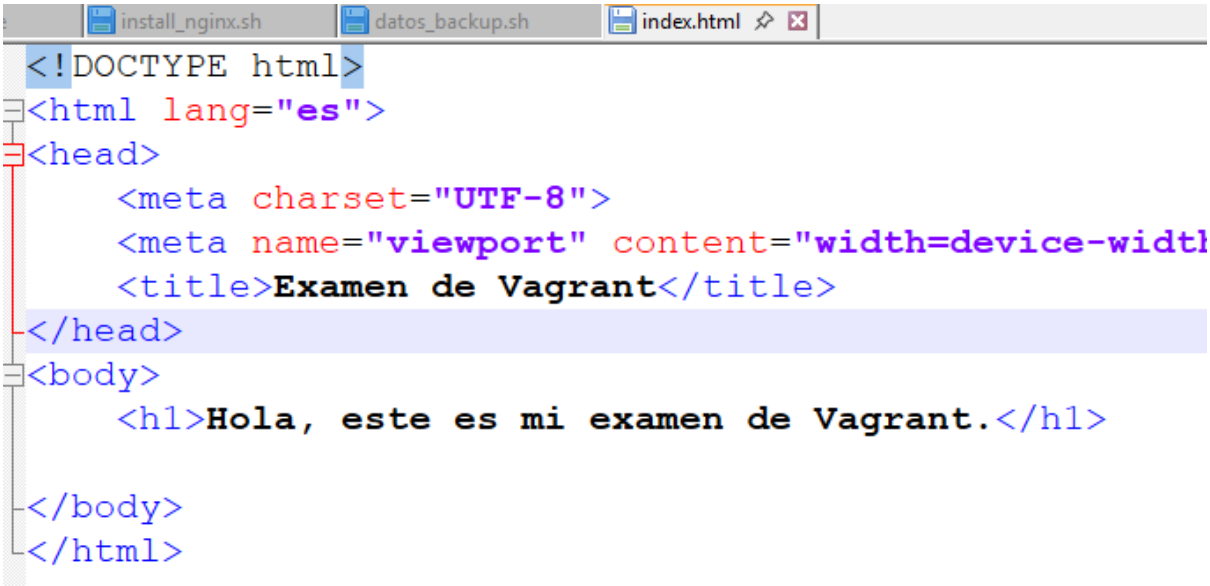
- Configura una **carpeta sincronizada** en `web01`. La carpeta `./html` de tu máquina física debe aparecer en `/var/www/html` (o la ruta por defecto del servidor web elegido) de la máquina virtual.

```
#Carpeta sincronizada
config.vm.synced_folder "html", "/var/www/html"
```

- Crea un archivo `index.html` en tu máquina física dentro de esa carpeta con el texto: *"Hola, este es mi examen de Vagrant"*.

El segundo archivo se ha creado una vez levantado las máquinas. Es el original de NGINX.





He levantado las máquinas:

```
C:\Users\UsuarioASIR\Documents\examen_vagrant>vagrant status
Current machine states:

web01                     running (virtualbox)
db01                      running (virtualbox)

This environment represents multiple VMs. The VMs are all listed
above with their current state. For more information about a specific
VM, run `vagrant status NAME`.

C:\Users\UsuarioASIR\Documents\examen_vagrant>
```

Y carga el archivo “**index.html**”.

Ejercicio 6: Snapshots

Una vez que todo el entorno esté funcionando:

1. Toma una instantánea (snapshot) de la máquina `db01` llamada `estado_inicial`.

Usamos: **vagrant snapshot save db01 estado_inicial**

```
C:\Users\UsuarioASIR\Documents\examen_vagrant>vagrant snapshot save db01 estado_inicial
==> db01: Snapshotting the machine as 'estado_inicial'...
==> db01: Snapshot saved! You can restore the snapshot at any time by
==> db01: using `vagrant snapshot restore`. You can delete it using
==> db01: `vagrant snapshot delete`.

C:\Users\UsuarioASIR\Documents\examen_vagrant>
```

```
C:\Users\UsuarioASIR\Documents\examen_vagrant>vagrant snapshot list
==> web01: No snapshots have been taken yet!
    web01: You can take a snapshot using `vagrant snapshot save`. Note that
    web01: not all providers support this yet. Once a snapshot is taken, you
    web01: can list them using this command, and use commands such as
    web01: `vagrant snapshot restore` to go back to a certain snapshot.
==> db01:
estado_inicial
```

2. Entra por SSH a `db01` y borra la carpeta `datos_backup` creada en el ejercicio 4.

Usamos: **vagrant ssh db01**

Borrado de datos_backup. Me pide confirmación.

```
vagrant@db01:~$ ls -l
total 4
drwxr-xr-x 2 root root 4096 Dec  3 10:06 datos_backup
vagrant@db01:~$ rm -r datos_backup/
rm: remove write-protected directory 'datos_backup/'? yes
vagrant@db01:~$ ls -l
total 0
vagrant@db01:~$ date
Wed Dec  3 10:50:48 UTC 2025
vagrant@db01:~$
```

3. Demuestra (mediante captura de pantalla o comando) que sabes restaurar el estado anterior para recuperar la carpeta borrada.

Nos salimos de la VM o usamos otro terminal. Desde fuera usamos:

vagrant snapshot restore db01 estado_inicial

```
vagrant@db01:~$ exit
logout

C:\Users\UsuarioASIR\Documents\examen_vagrant>vagrant snapshot restore db01 estado_inicial
==> db01: Forcing shutdown of VM...
==> db01: Restoring the snapshot 'estado_inicial'...
==> db01: Checking if box 'ubuntu/jammy64' version '20241002.0.0' is up to date...
==> db01: Resuming suspended VM...
==> db01: Booting VM...
==> db01: Waiting for machine to boot. This may take a few minutes...
    db01: SSH address: 127.0.0.1:2201
    db01: SSH username: vagrant
    db01: SSH auth method: private key
==> db01: Machine booted and ready!
==> db01: Machine already provisioned. Run `vagrant provision` or use the `--provision`
==> db01: flag to force provisioning. Provisioners marked to run always will still run.

C:\Users\UsuarioASIR\Documents\examen_vagrant>
```

Nos volvemos a conectar a db01: **vagrant ssh db01**

Ahí está otra vez la carpeta **datos_backup**.

```
vagrant@db01:~$ ls -l
total 4
drwxr-xr-x 2 root root 4096 Dec  3 10:06 datos_backup
vagrant@db01:~$ date
Wed Dec  3 10:51:38 UTC 2025
vagrant@db01:~$
```