Unknown Title

Actividad 2 - Shell scripts de aprovisionamiento

Requisitos de finalización

Cierre: domingo, 23 de noviembre de 2025, 23:59

Ver video que está en la Guía antes de realizar la actividad

Necesitamos una máquina virtual con una versión de ubuntu. Busca en Vagrant Hub, un box adecuado.

Inicializa y levanta la máquina virtual

Comprueba que tienes acceso.

Aprovisionamiento inline

Crea un aprovisionamiento de tipo shell inline, en el que crees una carpeta nueva.

¿Cómo puedes ejecutar el aprovisionamiento si la máquina ya está levantada? ¿ Y si no? Usa la que te interese

Comprueba que el aprovisionamiento se ha realizado

Aprovisionamiento por documento

Elimina el aprovisionamiento anterior y crea uno nuevo mediante un fichero, que cree una nueva carpeta (que sea distinta)

Recarga el aprovisionamiento

¿Existe la carpeta nueva? ¿Existe la carpeta anterior?

Modifica los recursos que usa la VM

Indica que cuando se use VirtualBox use solo una cpu y 1gb de ram

Aplica los cambios y demuestra que se han llevado a cabo

¡Claro! Aquí tienes la guía paso a paso para realizar la actividad utilizando Vagrant y el aprovisionamiento con Shell Scripts.

1. Configuración Inicial de la Máquina Virtual 💻

1.1. Búsqueda y Inicialización del Box

Un box de Ubuntu adecuado en Vagrant Cloud es, por ejemplo, **`ubuntu/jammy64`** (Ubuntu 22.04 LTS).

Pasos:

1. **Crea una carpeta** para tu proyecto y entra en ella:

```
mkdir vagrant_proyecto
cd vagrant_proyecto
```

2. **Inicializa el box** para crear el `Vagrantfile`:

```
vagrant init ubuntu/jammy64
```

3. **Levanta la máquina virtual:**

```
vagrant up
```

1.2. Comprobación de Acceso

Una vez levantada, puedes comprobar el acceso mediante SSH:

```
vagrant ssh
```

Si la conexión se realiza con éxito, significa que tienes acceso. Sal de la sesión SSH con `exit`.

---

2. Aprovisionamiento Inline (Creación de `carpeta_inline`)

El aprovisionamiento inline se define directamente dentro del `Vagrantfile` utilizando el parámetro `inline`.

2.1. Configuración en `Vagrantfile`

Edita tu `Vagrantfile` y añade la siguiente sección (dentro del bloque `Vagrant.configure("2") do |config|`):

```
  # Aprovisionamiento Inline
  config.vm.provision "shell", inline: <<-SHELL
    echo "Ejecutando aprovisionamiento inline..."
    mkdir /home/vagrant/carpeta_inline
    echo "Carpeta /home/vagrant/carpeta_inline creada."
  SHELL
```

2.2. Ejecución del Aprovisionamiento

- **Si la máquina no está levantada** (o está destruida), el aprovisionamiento se ejecuta automáticamente con `vagrant up`.
- **Si la máquina ya está levantada** (como en este caso, después de `vagrant up`), debes usar el comando **`vagrant provision`** para ejecutar solo los provisioners definidos en el `Vagrantfile`.

Comando de ejecución:

```
vagrant provision
```

2.3. Comprobación

Accede por SSH para verificar la existencia de la carpeta:

```
vagrant ssh
ls -l /home/vagrant/
# Deberías ver 'carpeta_inline' en la lista.
exit
```

---

3. Aprovisionamiento por Documento (Creación de `carpeta_documento`)

Este método utiliza un script de Shell externo (`provision.sh`) al que se hace referencia en el `Vagrantfile` mediante el parámetro `path`.

3.1. Eliminación del Aprovisionamiento Anterior

Elimina o comenta el bloque de aprovisionamiento inline del `Vagrantfile`.

3.2. Creación del Script

Crea un nuevo archivo llamado **`provision.sh`** en el mismo directorio que el `Vagrantfile` con el siguiente contenido:

```
#!/bin/bash
echo "Ejecutando aprovisionamiento por documento..."
mkdir /home/vagrant/carpeta_documento
echo "Carpeta /home/vagrant/carpeta_documento creada."
```

3.3. Configuración en `Vagrantfile`

Añade el nuevo bloque de aprovisionamiento por documento en el `Vagrantfile`:

```
  # Aprovisionamiento por Documento
  config.vm.provision "shell", path: "provision.sh"
```

3.4. Recarga del Aprovisionamiento

Como has modificado la configuración del `Vagrantfile`, debes recargar la máquina para aplicar los cambios y ejecutar el nuevo provisioner.

Comando de recarga:

```
vagrant reload --provision
```

El flag `--provision` asegura que se ejecuten los provisioners durante la recarga.

3.5. Comprobación de las Carpetas

Accede por SSH y verifica:

```
vagrant ssh
ls -l /home/vagrant/
# Deberías ver 'carpeta_documento' (nueva) y 'carpeta_inline' (anterior).
exit
```

- **¿Existe la carpeta nueva?** Sí, **`carpeta_documento`** debe existir.
- **¿Existe la carpeta anterior?** Sí, **`carpeta_inline`** también debe existir, ya que el aprovisionamiento anterior la creó y el comando `vagrant reload --provision` no la elimina.

---

4. Modificación de Recursos de la VM (VirtualBox)

Los recursos de hardware se modifican dentro del bloque del proveedor (`virtualbox`) en el `Vagrantfile`.

4.1. Configuración en `Vagrantfile`

Añade o modifica el bloque del proveedor en tu `Vagrantfile` para configurar 1 CPU y 1024 MB (1 GB) de RAM:

```
  # Modificación de recursos para VirtualBox
  config.vm.provider "virtualbox" do |v|
    v.memory = 1024 # 1 GB de RAM
    v.cpus = 1      # 1 CPU
  end
```

4.2. Aplicación y Demostración de los Cambios

Para aplicar cambios de hardware, debes **reiniciar** la máquina virtual.

Comando de aplicación:

```
vagrant reload
```

Demostración de los cambios (Opcional): Puedes verificar los cambios en la interfaz gráfica de **VirtualBox**. Al seleccionar la máquina virtual, en la configuración del sistema, verás que la memoria base está configurada a **1024 MB** y en la pestaña de procesador, el número de CPUs será **1**.

También, desde la propia VM, puedes ver la memoria y CPU asignadas:

```
vagrant ssh
grep MemTotal /proc/meminfo
# Debería mostrar un valor cercano a 1024 MB (e.g., 1017364 kB).
grep processor /proc/cpuinfo | wc -l
# Debería mostrar '1'.
exit
```

Siguiente Paso: ¿Te gustaría que te ayudara a configurar un aprovisionamiento adicional, como por ejemplo, instalar un servidor web Apache en esta VM?

Actividad 3 - Redes

Requisitos de finalización

Cierre: domingo, 23 de noviembre de 2025, 23:59

Vamos a probar las distintas configuración de redes en Vagrant

Parte 1 - Proyecto base (NAT por defecto)

Crea una máquina virtual a partir del box ubuntu/jammy64

Visualiza las interfaces de red y las ips usadas.

¿Tienes acceso a internet?

Parte 2 - Red privada (Host-only)

Configura red privada con ip fija

Recarga la máquina virtual.

Visualiza las interfaces de red y las ips usadas.

¿Tienes acceso a internet?

¿Puedes acceder a esta máquina virtual desde otro equipo?

Parte 3 - Reenvío de puertos

Crea un aprovisionamiento instalando nginx

Mapea el puerto host y guest

Desde el host accede a la web

Provocad una colisión cambiando host: 8080 a host: 80. ¿Qué avisa Vagrant? Activad auto_correct y anotad el nuevo puerto.

Parte 4 - Red pública (bridged)

Añadir adaptador puente para que la VM sea un “equipo más” en la LAN del aula

Recarga la máquina virtual.

Visualiza las interfaces de red y las ips usadas.

¿Tienes acceso a internet?

¿Puedes acceder a esta máquina virtual desde otro equipo?

¡Excelente! Esta actividad se centra en comprender cómo Vagrant maneja las diferentes configuraciones de red. Aquí tienes la guía completa para realizar cada parte.

Parte 1 - Proyecto Base (NAT por defecto) 🌍

Vagrant por defecto configura la red de la máquina virtual (VM) usando **NAT (Network Address Translation)**. Esto permite que la VM acceda a Internet, pero el Host no puede acceder a los servicios de la VM directamente (solo a través de reenvío de puertos).

Pasos

1. **Inicializa y levanta la máquina:**

```
mkdir vagrant_redes
cd vagrant_redes
```

```
vagrant init ubuntu/jammy64
vagrant up
```

2. **Visualiza las interfaces y las IPs:** Accede a la VM y usa el comando `ip a` (o `ifconfig` si está instalado).

```
vagrant ssh
ip a
```

Resultado esperado:

- Verás la interfaz **`eth0`** (o similar) con una IP privada, generalmente del rango **10.0.2.x**, que es la IP por defecto que VirtualBox asigna para el adaptador NAT.

3. **¿Tienes acceso a internet?** Sí. Puedes comprobarlo haciendo un `ping` a una dirección externa:

```
ping google.com -c 3
# Deberías ver respuesta.
exit
```

La configuración NAT permite que la VM salga a Internet, compartiendo la conexión del Host.

Parte 2 - Red Privada (Host-only) 🤫

La red privada (Host-only) crea una red aislada **solo entre el Host y la VM**, permitiendo una comunicación interna directa. La VM no tiene acceso directo a la LAN o Internet a través de esta interfaz.

Pasos

1. **Configura red privada con IP fija:** Modifica tu `Vagrantfile` para añadir la siguiente línea (dentro del bloque `config.vm.define`):

```
  # Red privada con IP fija
  config.vm.network "private_network", ip: "192.168.50.10"
```

2. **Recarga la máquina virtual:**

```
vagrant reload
```

3. **Visualiza las interfaces y las IPs:** Accede de nuevo por SSH.

```
vagrant ssh
ip a
```

Resultado esperado:

- Verás dos interfaces: **`eth0`** (NAT, IP 10.0.2.x) y **`eth1`** (Host-only, IP **192.168.50.10**).

4. **¿Tienes acceso a internet?** Sí, la VM seguirá teniendo acceso a Internet a través de la interfaz **`eth0`** (NAT), ya que esa configuración sigue activa por defecto.

```
ping google.com -c 3 # Debería funcionar.
exit
```

5. **¿Puedes acceder a esta máquina virtual desde otro equipo? No**, el Host-only (192.168.50.10) solo permite comunicación **entre el Host y la VM**. Un equipo externo en la LAN del aula no puede acceder directamente a esta IP porque la red 192.168.50.x es interna y no es visible en la red física del aula.

Parte 3 - Reenvío de Puertos (Port Forwarding) 🚪

El reenvío de puertos permite que el tráfico dirigido a un puerto específico del **Host** se redirija a un puerto de la **VM** (trabajando sobre la red NAT).

Pasos

1. **Crea un aprovisionamiento instalando Nginx:** Crea un archivo `install_nginx.sh` en el directorio:

```
#!/bin/bash
echo "Instalando Nginx..."
sudo apt update
sudo apt install -y nginx
sudo systemctl start nginx
sudo systemctl enable nginx
echo "Nginx instalado y funcionando."
```

   Añade el provisioner al `Vagrantfile`:

```
  # Aprovisionamiento para instalar Nginx
  config.vm.provision "shell", path: "install_nginx.sh"
```

2. **Mapea el puerto Host (8080) y Guest (80):** Añade la configuración de reenvío de puertos en el `Vagrantfile`:

```
  # Reenvío de puertos: Host:8080 -> Guest:80
  config.vm.network "forwarded_port", guest: 80, host: 8080
```

   Aplica y ejecuta el aprovisionamiento:

```
vagrant reload --provision
```

3. **Desde el Host accede a la web:** Abre tu navegador en la máquina Host y navega a: **`http://localhost:8080`**. Deberías ver la página por defecto de bienvenida de Nginx.

4. **Provocad una colisión cambiando host: 8080 a host: 80.** Cambia la línea en el `Vagrantfile`:

```
  config.vm.network "forwarded_port", guest: 80, host: 80 # Nuevo
puerto del host
```

   Intenta aplicar los cambios:

```
vagrant reload
```

   ¿Qué avisa Vagrant? Vagrant avisará de que el puerto 80 del Host probablemente está en uso por otro servicio (como el servidor web del Host o permisos de administrador en algunos SO). El mensaje será similar a:

   > "The forwarded port to **80** is **already in use** on the host machine. You may need to disable the service listening on this port..."

   Activad `auto_correct` y anotad el nuevo puerto. Modifica la línea de reenvío para activar la corrección automática:

```
  config.vm.network "forwarded_port", guest: 80, host: 80,
auto_correct: true
```

   Vuelve a recargar:

```
vagrant reload
```

   Vagrant intentará encontrar automáticamente el siguiente puerto libre. Al finalizar, te informará en la consola:

   > "The guest port **80** is forwarded to this host port: **2200** (using auto-correct)" (El puerto puede variar, por ejemplo, **2200, 2201,** etc.).

   Nuevo puerto (ejemplo): 2200.

Parte 4 - Red Pública (Bridged) 🏢

La red pública (Bridged) hace que la VM se conecte directamente a la red física del Host (la LAN del aula). La VM actuará como un equipo independiente más y obtendrá una IP dentro del rango de la red del aula (por DHCP, si está activo).

Pasos

1. **Añadir adaptador puente:** Elimina o comenta el bloque de `private_network` del `Vagrantfile` para evitar conflictos y añade la configuración de red puente:

```
  # Red pública (Bridged)
  config.vm.network "public_network"
  # Opcional: Si tienes varios adaptadores en el host y quieres
especificar uno:
  # config.vm.network "public_network", use_dhcp_assigned_host_if:
true, bridge: "eth0" # Cambia "eth0" por el nombre de tu adaptador
host
```

2. **Recarga la máquina virtual:**

```
vagrant reload
```

Nota: Vagrant te pedirá que selecciones el adaptador de red del Host a usar como puente.

3. **Visualiza las interfaces y las IPs:** Accede y comprueba las IPs.

```
vagrant ssh
ip a
```

Resultado esperado:

- La nueva interfaz (`eth1` o similar) tendrá una IP dentro del rango de la LAN del aula (ej. **192.168.1.x**, si ese es el rango del aula).

4. **¿Tienes acceso a internet? Sí.** Al estar directamente en la LAN, la VM puede acceder a Internet como cualquier otro equipo de la red.

```
ping google.com -c 3 # Debería funcionar.
exit
```

5. **¿Puedes acceder a esta máquina virtual desde otro equipo? Sí.** Dado que la VM tiene una IP válida dentro del rango de la LAN del aula, cualquier otro equipo conectado a esa misma LAN puede acceder a los servicios de la VM (como el servidor Nginx en el puerto 80) usando su IP asignada (por ejemplo, `http://192.168.1.15`).

Siguiente Paso: ¿Te gustaría ahora crear un script de aprovisionamiento que automatice la instalación de un stack completo como LAMP (Linux, Apache, MySQL, PHP)?

Actividad 4 - Carpetas sincronizadas y snapshots

Requisitos de finalización

Cierre: domingo, 23 de noviembre de 2025, 23:59

Entrega documento con explicaciones y la carpeta del proyecto Vagrant

Crea un nuevo proyecto con Vagrant con un entorno de Ubuntu:

Carpeta sincronizada

Añade una carpeta sincronizada

Comprueba que la carpeta se sincroniza al iniciar la VM

Dentro de la máquina virtual, crea un archivo dentro de la carpeta sincronizada

Comprueba desde el host que se ha sincronizado la carpeta.

Ahora crea un archivo desde el host y comprueba que se sincroniza en la VM

Snapshots

Crea un snapshot llamado estado1

Crea un archivo nuevo dentro de la máquina virtual

Restaura el snapshot anterior y comprueba que se ha realizado correctamente

¡Claro! Aquí tienes la guía para completar la actividad sobre carpetas sincronizadas y *snapshots* en tu proyecto de Vagrant.

1. Creación del Proyecto Base

Crea un nuevo directorio para este proyecto e inicialízalo con el box de Ubuntu:

```
mkdir vagrant_sync_snap
cd vagrant_sync_snap
vagrant init ubuntu/jammy64
```

2. Carpetas Sincronizadas (Synced Folders) 🔄

Las carpetas sincronizadas permiten compartir archivos entre el sistema operativo **Host** (tu máquina) y el **Guest** (la máquina virtual). Por defecto, Vagrant ya sincroniza la carpeta del proyecto del Host (`./`) con la carpeta `/vagrant` en el Guest. Sin embargo, para esta actividad, añadiremos una carpeta explícita.

2.1. Configuración en `Vagrantfile`

1. **Crea la carpeta en el Host:**

   ```
   mkdir sync_data
   ```

2. **Configura la sincronización** en el `Vagrantfile`. Asegúrate de añadir la siguiente línea de configuración dentro del bloque `Vagrant.configure("2") do |config|`:

   ```
     # Sintaxis: config.vm.synced_folder "ruta/en/host", "ruta/en/guest"
     config.vm.synced_folder "sync_data", "/home/vagrant/data_sync_vm"
   ```

3. **Inicia la máquina virtual:**

   ```
   vagrant up
   ```

2.2. Comprobación de la Sincronización (Host -> Guest)

1. **Comprueba que la carpeta se sincroniza al iniciar la VM:** Accede a la VM y verifica si existe el directorio creado en la ruta especificada:

   ```
   vagrant ssh
   ls -l /home/vagrant/
   # Deberías ver 'data_sync_vm'
   ```

2. **Crea un archivo desde el Host y comprueba en la VM:** Desde tu **Host** (sal de la sesión SSH con `exit` si estás dentro):

   ```
   echo "Contenido desde el Host" > sync_data/archivo_host.txt
   ```

   Regresa a la **VM** (o usa `ls` si aún estás dentro) y verifica el archivo:

   ```
   vagrant ssh
   cat /home/vagrant/data_sync_vm/archivo_host.txt
   # Debería mostrar: Contenido desde el Host
   ```

   Sal de la sesión SSH (`exit`).

2.3. Comprobación de la Sincronización (Guest -> Host)

1. **Crea un archivo desde la VM y comprueba en el Host:** Accede a la **VM**:

```
vagrant ssh
echo "Contenido desde el Guest" | sudo tee
/home/vagrant/data_sync_vm/archivo_guest.txt
exit # Sal de la sesión SSH
```

2. **Comprueba en el Host:** Desde tu **Host**, verifica que el archivo aparece en la carpeta `sync_data`:

```
cat sync_data/archivo_guest.txt
# Debería mostrar: Contenido desde el Guest
```

---

3. Snapshots (Instantáneas) 📸

Los *snapshots* te permiten guardar el estado exacto de tu máquina virtual en un momento dado, incluyendo el estado de la memoria y del disco. Esto es útil para volver a un punto de configuración anterior.

3.1. Creación de Snapshot

1. **Crea un snapshot llamado `estado1`:** Asegúrate de que la VM esté en ejecución (`vagrant status` debe decir `running`).

```
vagrant snapshot save estado1
```

   Vagrant guardará el estado de la VM, incluyendo las carpetas sincronizadas y los archivos creados hasta ahora.

3.2. Creación y Modificación de Archivo

1. **Crea un archivo nuevo dentro de la máquina virtual:** Entra en la VM y crea un archivo que debe desaparecer al restaurar el snapshot.

```
vagrant ssh
touch ~/archivo_post_snapshot.txt
echo "Archivo creado después de estado1" | sudo tee
/home/vagrant/archivo_post_snapshot.txt
exit
```

2. **Verificación del cambio (opcional):**

```
vagrant ssh
ls -l ~/
```

```
# Deberías ver 'archivo_post_snapshot.txt'
exit
```

3.3. Restauración de Snapshot

1. **Restaura el snapshot anterior (`estado1`):**

   ```
   vagrant snapshot restore estado1
   ```

   Vagrant detendrá la máquina, revertirá su estado al momento en que se tomó el snapshot y la reiniciará.

2. **Comprueba que se ha realizado correctamente:** Vuelve a entrar a la VM y verifica si el archivo creado después del snapshot ha desaparecido:

   ```
   vagrant ssh
   ls -l ~/
   # El archivo 'archivo_post_snapshot.txt' NO debería estar presente.
   # Los archivos creados en las carpetas sincronizadas SÍ estarán, ya
   que la sincronización es una característica del disco del Host.
   exit
   ```

   Verificación clave: La restauración es correcta si `archivo_post_snapshot.txt` no existe.

Archivos a entregar:

1. **Documento con explicaciones:** Este documento que acabas de leer, explicando los comandos y los resultados esperados.
2. **Carpeta del proyecto Vagrant (`vagrant_sync_snap`):** Contiene el `Vagrantfile`, el directorio `sync_data` con los archivos de prueba, y la carpeta `.vagrant` (que contiene los *snapshots*).

C