



DESAFIO DEVOPS ELDAR

DOCUMENTACIÓN



DESAFIO DEVOPS ELDAR

ELDAR ACADEMY

Versión: 1.0

Realizado por: Franco Congedo

Fecha: 05 de noviembre de 2024

Prerrequisitos:	2
Entorno de Vagrant	2
Instalacion de aplicaciones utilizando bootstrap.sh	5
Ejercicio 1 - Linux (creación de estructura de directorios)	8
Ejercicio 2 - Bash	15
Automatización de la creación del árbol de directorios:	24
Ejercicio 3 - Docker	26
Jenkins CI/CD	34
Instalacion y configuracion de Jenkins	34
Diagrama CI/CD Docker	41
Creación del pipeline	45
Configuración de ngrok para exponer Jenkins	50
Configuración de Webhook en Github	52
Links	61



Prerrequisitos:

Entorno de Vagrant

Para configurar el entorno de Vagrant con el Box **focal64** de Ubuntu, es necesario realizar los siguientes pasos:

1. **Descargar Vagrant:** Primero, instala Vagrant en tu sistema desde la página oficial de descargas [aquí](#).
2. **Obtener la imagen de Ubuntu:** La imagen **focal64** de Ubuntu se encuentra en el repositorio oficial de Vagrant. Puedes encontrar más detalles sobre esta imagen en [este enlace](#).
3. **Crear el entorno de trabajo:**
 - Crea una carpeta en el escritorio que servirá como directorio del proyecto.
 - Abre Visual Studio Code (VSC) y selecciona la carpeta recién creada.
 - Desde VSC, abre una nueva terminal en el directorio del proyecto y crea una subcarpeta llamada **Vagrant**.

De esta forma, tendrás tu entorno de Vagrant listo para configurarse en el Box de Ubuntu **focal64**.



```
Directorio: E:\Escritorio\Challenge-devops

Mode                LastWriteTime        Length Name
----                -----          ----  -
d-----      24/7/2024     16:30           Vagrant

● PS E:\Escritorio\Challenge-devops> cd Vagrant
○ PS E:\Escritorio\Challenge-devops\Vagrant> 
```

Luego, ejecuta `vagrant init` para crear el archivo `Vagrantfile`, donde se configurará la máquina virtual.

```
└─ Vagrant
    └─ Vagrantfile

Mode                LastWriteTime        Length Name
----                -----          ----  -
d-----      24/7/2024     16:30           Vagrant

● PS E:\Escritorio\Challenge-devops> cd Vagrant
● PS E:\Escritorio\Challenge-devops\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.
○ PS E:\Escritorio\Challenge-devops\Vagrant> 
```

Una vez creado el `Vagrantfile`, configúralo agregando las siguientes líneas:

- Primera línea: configuramos el nombre de la VM.
- Segunda línea: especificamos el box que vamos a utilizar.

```
Vagrant.configure("2") do |config|
  config.vm.define "desafio" do |desafio|
    desafio.vm.box = "ubuntu/focal64"
  end
end
```

Modificamos los valores de memoria y CPUs para asignar los recursos deseados a la máquina virtual.



Configuramos el nombre de la VM, así como los valores de memoria (6144 MB) y el número de CPUs (4).

```
config.vm.provider "virtualbox" do |vb|
    vb.name = "desafio"
    vb.memory = "6144"
    vb.cpus = 4
end
```

Configuramos el reenvío de puertos, de modo que el puerto 80 de la VM se redirija al puerto 8080 del host.

```
config.vm.network "forwarded_port", guest: 80, host: 8080
```

Configuramos un script de aprovisionamiento que se ejecutará al iniciar la VM, especificando el archivo `bootstrap.sh`.

```
config.vm.provision "shell", path: "bootstrap.sh"
end
```

Ejecutamos un `vagrant up`

```
PS E:\Escritorio\Challenge-devops\Vagrant> vagrant up
Bringing machine 'desafio' up with 'virtualbox' provider...
==> desafio: Importing base box 'ubuntu/focal64'...
==> desafio: Matching MAC address for NAT networking...
==> desafio: Checking if box 'ubuntu/focal64' version '20240408.0.0' is up to date...
==> desafio: Setting the name of the VM: desafio
==> desafio: Clearing any previously set network interfaces...
==> desafio: Preparing network interfaces based on configuration...
```

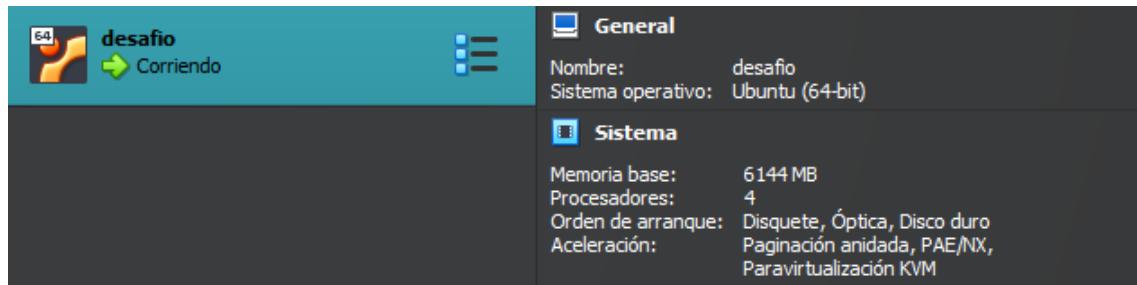


Ejecutamos un `vagrant status`

```
● PS E:\Escritorio\Challenge-devops\Vagrant> vagrant status
Current machine states:

desafio           running (virtualbox)
```

También verificamos en VirtualBox las características de la VM, confirmando que tiene 6 GB de memoria y 4 núcleos de CPU.



Instalacion de aplicaciones utilizando bootstrap.sh

Para la etapa de Docker, vamos a instalar la herramienta. Por lo tanto, agregamos los siguientes comandos a nuestro archivo `bootstrap.sh`:

```
# Docker Install
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh ./get-docker.sh

# Use Docker without sudo
sudo groupadd docker
sudo usermod -aG docker vagrant
newgrp docker
```



Para la etapa de Jenkins, vamos a instalar la herramienta (recuerda que necesita Java). Por lo tanto, agregamos los siguientes comandos a nuestro archivo `bootstrap.sh`:

```
# Install Java 17
sudo apt-get install fontconfig openjdk-17-jre -y

# Descargar la clave GPG de Jenkins
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

# Agregar el repositorio de Jenkins a la lista de fuentes
echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]
  https://pkg.jenkins.io/debian-stable binary/" | sudo tee
/etc/apt/sources.list.d/jenkins.list > /dev/null

# Actualizar la lista de paquetes nuevamente
sudo apt-get update

# Instalar Jenkins
sudo apt-get install -y jenkins
```

Para visualizar el árbol de directorios, instalamos la herramienta `tree`. Por lo tanto, agregamos los siguientes comandos a nuestro archivo `bootstrap.sh`:

```
# Instalar tree
sudo apt-get install -y tree
```



Para instalar `vim`, agrega el siguiente comando a tu archivo `bootstrap.sh`:

```
# Instalar vim
sudo apt install -y vim
```

Para instalar ansible, agrego el siguiente comando al archivo `bootstrap.sh`:

```
# Instalar Ansible
sudo apt-add-repository -y ppa:ansible/ansible
sudo apt-get update
sudo apt-get install -y ansible
```

Para instalar la biblioteca Docker para Python agrego el siguiente comando:

```
# Instalar la biblioteca Docker para Python
sudo apt-get install -y python3-pip
sudo pip3 install docker

sudo ansible-galaxy collection install community.docker
```

Verificacion de instalacion de herramientas:

```
vagrant@ubuntu-focal:~$ tree --version
tree v1.8.0 (c) 1996 - 2018 by Steve Baker, Thomas Moore, Francesc Rocher, Florian Sesser, Kyosuke Tokoro
vagrant@ubuntu-focal:~$ docker --version
Docker version 27.3.1, build ce12230
vagrant@ubuntu-focal:~$ jenkins --version
2.479.1
vagrant@ubuntu-focal:~$ vim --version
VIM - Vi IMproved 8.1 (2018 May 18, compiled Sep 25 2024 05:18:33)
```

Ejercicio 1 - Linux (creación de estructura de directorios)

1. Crea esta estructura de directorios.

```
/home/<nombre_usuario>/Documentos/Proyectos
    └── TiendaOnline
        ├── código
        │   ├── backend
        │   │   ├── controllers
        │   │   ├── models
        │   │   ├── routes
        │   │   └── views
        │   ├── frontend
        │   │   ├── assets
        │   │   │   ├── css
        │   │   │   └── js
        │   │   └── components
        │   ├── scripts
        │   ├── documentación
        │   └── pruebas
    └── BlogPersonal
        ├── artículos
        ├── borradores
        ├── imágenes
        └── plantillas
    └── ProyectoAppMovil
        ├── android
        ├── documentación
        ├── ios
        └── recursos
```

Primero nos conectamos utilizando el comando `vagrant ssh`

```
PS E:\Escritorio\Challenge-devops\Vagrant> vagrant ssh
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.4.0-176-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:      https://landscape.canonical.com
 * Support:         https://ubuntu.com/pro
```



Vagrant nos sitúa por defecto en el directorio '[/home/vagrant](#)' (usuario vagrant)

Entonces procedemos a crear el árbol:

Primero creamos los directorios principales:

Utilizando los siguientes comandos

```
mkdir -p /home/vagrant/Documentos/Proyectos/BlogPersonal  
mkdir -p /home/vagrant/Documentos/Proyectos/ProyectoAppMovil  
mkdir -p /home/vagrant/Documentos/Proyectos/TiendaOnline
```

Luego pasó a crear los subdirectorios de 'BlogPersonal'

Ejecutó los siguientes comandos:

```
mkdir -p  
/home/vagrant/Documentos/Proyectos/BlogPersonal/artículos  
mkdir -p  
/home/vagrant/Documentos/Proyectos/BlogPersonal/borradores  
mkdir -p  
/home/vagrant/Documentos/Proyectos/BlogPersonal/ímágenes  
mkdir -p  
/home/vagrant/Documentos/Proyectos/BlogPersonal/plantillas
```



Luego pasó a crear los subdirectorios de '[ProyectoAppMovil](#)'

Ejecutó los siguientes comandos:

```
mkdir -p  
/home/vagrant/Documentos/Proyectos/ProyectoAppMovil/android  
mkdir -p  
/home/vagrant/Documentos/Proyectos/ProyectoAppMovil/documentación  
mkdir -p  
/home/vagrant/Documentos/Proyectos/ProyectoAppMovil/ios  
mkdir -p  
/home/vagrant/Documentos/Proyectos/ProyectoAppMovil/recursos
```

Luego pasó a crear los subdirectorios de '[TiendaOnline](#)'

```
mkdir -p  
/home/vagrant/Documentos/Proyectos/TiendaOnline/código/backend  
/controllers  
mkdir -p  
/home/vagrant/Documentos/Proyectos/TiendaOnline/código/backend  
/models  
mkdir -p  
/home/vagrant/Documentos/Proyectos/TiendaOnline/código/backend  
/routes  
mkdir -p  
/home/vagrant/Documentos/Proyectos/TiendaOnline/código/backend  
/views  
mkdir -p  
/home/vagrant/Documentos/Proyectos/TiendaOnline/código/frontend/assets/css  
mkdir -p  
/home/vagrant/Documentos/Proyectos/TiendaOnline/código/frontend/assets/js
```



```
mkdir -p  
/home/vagrant/Documentos/Proyectos/TiendaOnline/código/fronten  
d/components  
mkdir -p  
/home/vagrant/Documentos/Proyectos/TiendaOnline/código/scripts  
mkdir -p  
/home/vagrant/Documentos/Proyectos/TiendaOnline/documentación  
mkdir -p  
/home/vagrant/Documentos/Proyectos/TiendaOnline/pruebas
```

2. Ubica tu posición actual en los directorios y muévete a la raíz

Me ubique dentro del último directorio creado utilizando

```
cd /home/vagrant/Documentos/Proyectos/TiendaOnline/pruebas
```

Luego ejecuto el comando '`pwd`' (para visualizar la ubicación actual)

```
vagrant@ubuntu-focal:~/Documentos/Proyectos/TiendaOnline/pruebas$ pwd  
/home/vagrant/Documentos/Proyectos/TiendaOnline/pruebas
```

Por último ejecutar el comando '`cd /`' (para moverme a la raíz)

```
vagrant@ubuntu-focal:~/Documentos/Proyectos/TiendaOnline/pruebas$ cd /  
vagrant@ubuntu-focal:/$
```

3/5. Copia los archivos de una carpeta que creaste a otra desde la raíz.

Primero voy a crear un archivo desde la raíz ejecutando el siguiente comando:

```
touch  
/home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/arch  
ivo.txt
```



Luego voy a modificar dicho archivo ejecutando el comando:

```
echo "Este es un archivo de ejemplo" >
/home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/archivo.txt
```

Verificamos que creó el archivo ejecutando el comando:

```
ls /home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/
```

Procedimiento:

```
vagrant@ubuntu-focal:~$ touch /home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/archivo.txt
vagrant@ubuntu-focal:~$ echo "Este es un archivo de ejemplo" > /home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/archivo.txt
vagrant@ubuntu-focal:~$ ls /home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/
archivo.txt
```

Luego verificamos que se escribió en el archivo utilizando el comando '`cat`'

```
cat
/home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/archivo.txt
```

```
vagrant@ubuntu-focal:~$ cat /home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/archivo.txt
Este es un archivo de ejemplo
```

Por último realizamos la copia del archivo desde la raíz.

Voy a copiar desde

```
/home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/archivo.txt
```

Al destino

```
/home/vagrant/Documentos/Proyectos/ProyectoAppMovil/recursos/
```



Para esto utilizamos el comando 'cp'

```
cp  
/home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/archivo.txt  
/home/vagrant/Documentos/Proyectos/ProyectoAppMovil/recursos/
```

Y luego corroboramos que el archivo se copió ejecutando el comando:

```
vagrant@ubuntu-focal:~$ cp /home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/archivo.txt /home/vagrant/Documentos/Proyectos/ProyectoAppMovil/recursos/  
vagrant@ubuntu-focal:~$ ls /home/vagrant/Documentos/Proyectos/ProyectoAppMovil/recursos/  
archivo.txt
```

4. Muestra los comandos que has usado hasta ahora.

```
vagrant@ubuntu-focal:~$ history  
1 mkdir -p /home/vagrant/Documentos/Proyectos/BlogPersonal  
2 mkdir -p /home/vagrant/Documentos/Proyectos/ProyectoAppMovil  
3 mkdir -p /home/vagrant/Documentos/Proyectos/TiendaOnline  
4 mkdir -p /home/vagrant/Documentos/Proyectos/BlogPersonal/artículos  
5 mkdir -p /home/vagrant/Documentos/Proyectos/BlogPersonal/borradores  
6 mkdir -p /home/vagrant/Documentos/Proyectos/BlogPersonal/ímágenes  
7 mkdir -p /home/vagrant/Documentos/Proyectos/BlogPersonal/plantillas  
8 mkdir -p /home/vagrant/Documentos/Proyectos/ProyectoAppMovil/android  
9 mkdir -p /home/vagrant/Documentos/Proyectos/ProyectoAppMovil/documentación  
10 mkdir -p /home/vagrant/Documentos/Proyectos/ProyectoAppMovil/ios  
11 mkdir -p /home/vagrant/Documentos/Proyectos/ProyectoAppMovil/recursos  
12 mkdir -p /home/vagrant/Documentos/Proyectos/TiendaOnline/código/backend/controllers  
13 mkdir -p /home/vagrant/Documentos/Proyectos/TiendaOnline/código/backend/models  
14 mkdir -p /home/vagrant/Documentos/Proyectos/TiendaOnline/código/backend/routes  
15 mkdir -p /home/vagrant/Documentos/Proyectos/TiendaOnline/código/backend/views  
16 mkdir -p /home/vagrant/Documentos/Proyectos/TiendaOnline/código/frontend/assets/css  
17 mkdir -p /home/vagrant/Documentos/Proyectos/TiendaOnline/código/frontend/assets/js  
18 mkdir -p /home/vagrant/Documentos/Proyectos/TiendaOnline/código/frontend/components  
19 mkdir -p /home/vagrant/Documentos/Proyectos/TiendaOnline/código/scripts  
20 mkdir -p /home/vagrant/Documentos/Proyectos/TiendaOnline/documentación  
21 mkdir -p /home/vagrant/Documentos/Proyectos/TiendaOnline/pruebas  
22 pwd  
23 cd /home/vagrant/Documentos/Proyectos/TiendaOnline/pruebas  
24 pwd  
25 cd /  
26 touch /home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/archivo.txt  
27 echo "Este es un archivo de ejemplo" > /home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/archivo.txt  
  
28 ls /home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/  
29 cat /home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/archivo.txt  
30 cp /home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/archivo.txt /home/vagrant/Documentos/Proyectos/  
ProyectoAppMovil/recursos/  
31 ls /home/vagrant/Documentos/Proyectos/ProyectoAppMovil/recursos/  
32 history
```

Por último nos dirigimos al directorio `/home/vagrant`

y ejecutamos `tree Documentos` (para desplegar el árbol de directorios)

```
vagrant@ubuntu-focal:~$ cd /home/vagrant
vagrant@ubuntu-focal:~$ tree Documentos
Documentos
└── Proyectos
    ├── BlogPersonal
    │   ├── artículos
    │   │   └── archivo.txt
    │   ├── borradores
    │   ├── imágenes
    │   └── plantillas
    ├── ProyectoAppMovil
    │   ├── android
    │   ├── documentación
    │   ├── ios
    │   └── recursos
    │       └── archivo.txt
    └── TiendaOnline
        ├── código
        │   ├── backend
        │   │   ├── controllers
        │   │   ├── models
        │   │   ├── routes
        │   │   └── views
        │   ├── frontend
        │   │   ├── assets
        │   │   │   ├── css
        │   │   │   └── js
        │   │   └── components
        │   └── scripts
        ├── documentación
        └── pruebas
```



Ejercicio 2 - Bash

1. Escribir un script que busque la palabra "palabra_a_buscar" en todos los archivos del directorio actual y muestre los nombres de los archivos que la contienen.

Para realizar este ejercicio, primero creamos otro archivo en el directorio donde ya tenemos alojado `archivo.txt`. Ejecutamos:

```
touch /home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/archivo2.txt
```

Luego, añadimos contenido al archivo:

```
echo "Hola" >
/home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/archivo2.txt
```

Por último, listamos los archivos en el directorio con:

```
ls
```

Luego, cambiamos al directorio donde se encuentran los archivos:

```
cd /home/vagrant/Documentos/Proyectos/BlogPersonal/artículos
```

Creamos el archivo sh ejecutando:

```
vim nano buscar_palabra.sh
```



```
#!/bin/bash

# Verifica si se proporcionó la palabra a buscar
if [ -z "$1" ]; then
    echo "Uso: $0 palabra_a_buscar"
    exit 1
fi

palabra_a_buscar="$1"

# Busca la palabra en todos los archivos del directorio actual
echo "Buscando la palabra '$palabra_a_buscar' en archivos del directorio actual..."
grep -rl "$palabra_a_buscar" .
```

Luego hacemos ejecutable el script de bash ejecutando el comando:

```
chmod +x
/home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/buscar_palabra.sh
```

Nos dirigimos al directorio donde esta el script y los archivos para testear:

```
vagrant@ubuntu-focal:~$ cd /home/vagrant/Documentos/Proyectos/BlogPersonal/artículos
vagrant@ubuntu-focal:~/Documentos/Proyectos/BlogPersonal/artículos$ ls
archivo.txt archivo2.txt buscar_palabra.sh
```

Para ejecutar el script `buscar_palabra.sh`, debemos usar el comando:

```
./buscar_palabra.sh Hola ('Hola' es la palabra que el script buscará en los archivos.)
```

```
Buscando la palabra 'hola' en archivos del directorio actual...
vagrant@ubuntu-focal:~/Documentos/Proyectos/BlogPersonal/artículos$ ./buscar_palabra.sh Hola
Buscando la palabra 'Hola' en archivos del directorio actual...
./archivo2.txt
```

2. Escribir un script que cree un menú simple con opciones para realizar diferentes tareas, como ver la lista de archivos, copiar un archivo o eliminar un archivo.
(Utilizó los mismos archivos y directorio del punto anterior)



Creamos el menu utilizando `vim menu.sh`

```
#!/bin/bash

# Función para mostrar el menú
mostrar_menu() {
    echo "Seleccione una opción:"
    echo "1. Ver la lista de archivos"
    echo "2. Copiar un archivo"
    echo "3. Eliminar un archivo"
    echo "4. Salir"
}

# Función para ver la lista de archivos
ver_lista_archivos() {
    echo "Lista de archivos en el directorio actual:"
    ls -l
}

# Función para copiar un archivo
copiar_archivo() {
    echo "Ingrese el nombre del archivo a copiar y el destino,
separados por un espacio:"
    read archivo_original destino
    if [ -f "$archivo_original" ]; then
        cp "$archivo_original" "$destino"
        echo "Archivo '$archivo_original' copiado exitosamente a
'$destino'."
    else
        echo "Error: El archivo '$archivo_original' no existe."
    fi
}

# Función para eliminar un archivo
eliminar_archivo() {
    echo "Ingrese el nombre del archivo a eliminar:"
```



```
read archivo
if [ -f "$archivo" ]; then
    rm "$archivo"
    echo "Archivo '$archivo' eliminado exitosamente."
else
    echo "Error: El archivo '$archivo' no existe."
fi
}

# Bucle principal del menú
while true; do
    mostrar_menu
    read -p "Seleccione una opción: " opcion

    # Verificar que la opción esté en el rango de 1 a 4
    # Verificar que la opción esté en el rango de 1 a 4
    if [[ "$opcion" =~ ^[1-4]$ ]]; then
        case $opcion in
            1)
                ver_lista_archivos
                ;;
            2)
                copiar_archivo
                ;;
            3)
                eliminar_archivo
                ;;
            4)
                echo "Saliendo..."
                exit 0
                ;;
        esac
    else
        echo "Opción inválida. Intente de nuevo con una opción
entre 1 y 4."
    fi
```



```
echo # Línea en blanco para mejor legibilidad entre opciones  
done
```

Luego hacemos ejecutable el script de bash ejecutando el comando:

```
chmod +x  
/home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/menu  
.sh
```

Para ejecutar el script menu.sh, debemos usar el comando:

```
./menu.sh
```

Seleccionamos (1) listar archivos.

```
vagrant@ubuntu-focal:~/Documentos/Proyectos/BlogPersonal/artículos$ ./menu.sh  
Seleccione una opción:  
1. Ver la lista de archivos  
2. Copiar un archivo  
3. Eliminar un archivo  
4. Salir  
Seleccione una opción: 1  
Lista de archivos en el directorio actual:  
total 20  
-rw-r--r-- 1 root root 29 Nov 5 05:04 archivo.txt  
-rw-r--r-- 1 root root 4 Nov 5 05:04 archivo2.txt  
-rwxr-xr-x 1 root root 332 Nov 5 06:11 buscar_palabra.sh  
-rwxr-xr-x 1 root root 729 Nov 5 05:04 buscar_reemplazar.sh  
-rwxr-xr-x 1 root root 1613 Nov 5 06:21 menu.sh
```

Seleccionamos (2) copiar archivos.

```
Seleccione una opción:  
1. Ver la lista de archivos  
2. Copiar un archivo  
3. Eliminar un archivo  
4. Salir  
Seleccione una opción: 2  
Ingrese el nombre del archivo a copiar y el destino, separados por un espacio:  
archivo.txt /home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/archivo1_copia.txt  
Archivo 'archivo.txt' copiado exitosamente a '/home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/archivo1_copia.txt'.
```



Luego listamos los archivos para ver si se copió correctamente

```
Seleccione una opción:  
1. Ver la lista de archivos  
2. Copiar un archivo  
3. Eliminar un archivo  
4. Salir  
Seleccione una opción: 1  
Lista de archivos en el directorio actual:  
total 24  
-rw-r--r-- 1 root      root    29 Nov  5 05:04 archivo.txt  
-rw-r--r-- 1 vagrant   vagrant  29 Nov  5 06:23 archivo1_copia.txt  
-rw-r--r-- 1 root      root     4 Nov  5 05:04 archivo2.txt  
-rwxr-xr-x 1 root      root    332 Nov  5 06:11 buscar_palabra.sh  
-rwxr-xr-x 1 root      root    729 Nov  5 05:04 buscar_reemplazar.sh  
-rwxr-xr-x 1 root      root   1613 Nov  5 06:21 menu.sh
```

Por último seleccionamos (3) eliminar archivo.

```
Seleccione una opción:  
1. Ver la lista de archivos  
2. Copiar un archivo  
3. Eliminar un archivo  
4. Salir  
3  
Ingrese el nombre del archivo a eliminar:  
archivo1_copia.txt  
Archivo eliminado exitosamente.  
Seleccione una opción:  
1. Ver la lista de archivos  
2. Copiar un archivo  
3. Eliminar un archivo  
4. Salir
```



Y volvemos a listar los archivos (1) para corroborar

```
Seleccione una opción:  
1. Ver la lista de archivos  
2. Copiar un archivo  
3. Eliminar un archivo  
4. Salir  
Seleccione una opción: 1  
Lista de archivos en el directorio actual:  
total 20  
-rw-r--r-- 1 root root 29 Nov 5 05:04 archivo.txt  
-rw-r--r-- 1 root root 4 Nov 5 05:04 archivo2.txt  
-rwxr-xr-x 1 root root 332 Nov 5 06:11 buscar_palabra.sh  
-rwxr-xr-x 1 root root 729 Nov 5 05:04 buscar_reemplazar.sh  
-rwxr-xr-x 1 root root 1613 Nov 5 06:21 menu.sh
```

Verificamos al ingresar 0 o 5 nos da error de [opción invalida](#)

```
Seleccione una opción:  
1. Ver la lista de archivos  
2. Copiar un archivo  
3. Eliminar un archivo  
4. Salir  
Seleccione una opción: 0  
Opción inválida. Intente de nuevo con una opción entre 1 y 4.  
  
Seleccione una opción:  
1. Ver la lista de archivos  
2. Copiar un archivo  
3. Eliminar un archivo  
4. Salir  
Seleccione una opción: 5  
Opción inválida. Intente de nuevo con una opción entre 1 y 4.
```

Por último salimos del programa ejecutando (4)

```
Seleccione una opción:  
1. Ver la lista de archivos  
2. Copiar un archivo  
3. Eliminar un archivo  
4. Salir  
Seleccione una opción: 4  
Saliendo...
```



3. Escribir un script que utilice expresiones regulares para buscar y reemplazar texto en un archivo

Primero nos situamos en el directorio donde voy a crear el script

```
cd /home/vagrant/Documentos/Proyectos/BlogPersonal/artículos
```

```
vagrant@ubuntu-focal:~$ cd /home/vagrant/Documentos/Proyectos/BlogPersonal/artículos
vagrant@ubuntu-focal:~/Documentos/Proyectos/BlogPersonal/artículos$ ls
archivo.txt  archivo2.txt  buscar_palabra.sh  buscar_palabra.sh.save  menu.sh
vagrant@ubuntu-focal:~/Documentos/Proyectos/BlogPersonal/artículos$
```

Creamos el archivo ejecutando `vim buscar_reemplazar.sh`

```
#!/bin/bash

# Directorio de trabajo
directorio="/home/vagrant/Documentos/Proyectos/BlogPersonal/artículos"

# Archivo en el que se realizará el reemplazo
archivo="$directorio/$1"

# Texto a buscar y reemplazar (pasado como argumentos al script)
texto_buscar=$2
texto_reemplazar=$3

# Verificar si se han pasado los argumentos necesarios
if [ $# -ne 3 ]; then
    echo "Uso: $0 nombre_archivo 'texto_a_buscar'
'texto_a_reemplazar'"
    exit 1
fi

# Verificar si el archivo existe
```



```
if [ ! -f "$archivo" ]; then
    echo "El archivo '$archivo' no existe."
    exit 1
fi

# Realizar el reemplazo en el archivo especificado
sed -i "s/${texto_buscar}/${texto_reemplazar}/g" "$archivo"

echo "Reemplazo completado en el archivo '$archivo'."
```

Luego hacemos ejecutable el script de bash ejecutando el comando:

```
chmod +x
/home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/buscar_reemplazar.sh
```

Para ejecutar el script `buscar_reemplazar.sh`, debemos usar el comando:

```
./buscar_reemplazar.sh archivo.txt "ejemplo" "demostración"
```

(Donde “ejemplo” es la palabra a reemplazar por “demostración”)

```
vagrant@ubuntu-focal:~/Documentos/Proyectos/BlogPersonal/artículos$ ./buscar_reemplazar.sh archivo.txt "ejemplo" "demostración"
Reemplazo completado en el archivo '/home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/archivo.txt'.
```

Ahora verificamos que cambió la palabra “ejemplo” por “demostración”

Ejecutado un `cat archivo.txt`

```
vagrant@ubuntu-focal:~/Documentos/Proyectos/BlogPersonal/artículos$ cat archivo.txt
Este es un archivo de demostración
```

Voy a verificar el funcionamiento del mensaje de cuando no encuentra el archivo a buscar.

Primero listamos los archivos con un `ls`: (vemos que no existe `archivo1.txt`)

```
Ejecutamos ./buscar_reemplazar.sh archivo1.txt "ejemplo"
"demostración"
```



Y obtenemos que no existe:

```
vagrant@ubuntu-focal:~/Documentos/Proyectos/BlogPersonal/artículos$ ls  
archivo.txt  archivo2.txt  buscar_palabra.sh  buscar_reemplazar.sh  menu.sh  
vagrant@ubuntu-focal:~/Documentos/Proyectos/BlogPersonal/artículos$ ./buscar_reemplazar.sh archivo1.txt "ejemplo" "demostración"  
El archivo '/home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/archivo1.txt' no existe.  
vagrant@ubuntu-focal:~/Documentos/Proyectos/BlogPersonal/artículos$
```

Automatización de la creación del árbol de directorios:

En el archivo llamado '`bootstrap.sh`' agregando esta linea automatice la creación del árbol de directorios:

```
mkdir -p  
/home/vagrant/Documentos/Proyectos/BlogPersonal/{artículos,bor  
radores,imágenes,plantillas,ProyectoAppMovil/android,ProyectoA  
ppMovil/documentación,ProyectoAppMovil/ios,ProyectoAppMovil/re  
cursos,TiendaOnline/código/backend/{controllers,models,routes,  
views},TiendaOnline/código/frontend/assets/{css,js},TiendaOnli  
ne/código/frontend/components,TiendaOnline/documentación,prueb  
as}
```

El comando

```
chown -R vagrant:vagrant /home/vagrant/Documentos/Proyectos
```

cambia la propiedad de todos los archivos y directorios en

```
/home/vagrant/Documentos/Proyectos
```

al usuario y grupo `vagrant`

```
# Cambiar la propiedad del directorio para el usuario vagrant  
sudo chown -R vagrant:vagrant /home/vagrant/Documentos/Proyectos
```

Este bloque de código verifica si existe el directorio

```
/home/vagrant/Desafío-DevOps. Si existe, copia los archivos archivo.txt,  
archivo2.txt, y tres scripts (buscar_palabra.sh, buscar_reemplazar.sh, menu.sh) al  
directorío /home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/.
```



```
# Copiar archivos al directorio de destino
if [ -d /home/vagrant/Desafio-DevOps ]; then
    sudo cp /home/vagrant/Desafio-DevOps/archivos_bash/archivo{,2}.txt /home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/
    sudo cp /home/vagrant/Desafio-DevOps/archivos_bash/{buscar_palabra.sh,buscar_reemplazar.sh,menu.sh} /home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/
fi
```

Finalmente, añadí el comando `sudo chmod +x` `/home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/*.sh` para dar permisos de ejecución a todos los archivos `.sh` en el directorio de destino.

El `+x` agrega permisos de ejecución,
El `*` selecciona todos los archivos con extensión `.sh`.

```
# Cambiar permisos para ejecutar archivos .sh
sudo chmod +x /home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/*.sh
```

Con estos cambios, al ejecutar la máquina virtual de [Vagrant](#), se crea automáticamente el directorio solicitado y se copian los archivos de script de Bash y los archivos de texto necesarios para las pruebas

Para visualizar el árbol de directorios creado, simplemente ejecuta el siguiente comando desde el directorio home del usuario vagrant (donde te encuentras por defecto): `tree Documentos`

```
vagrant@ubuntu-focal:~/ Documentos/  
Documentos/  
└── Proyectos  
    ├── BlogPersonal  
    │   ├── ProyectoAppMovil  
    │   │   ├── android  
    │   │   ├── documentación  
    │   │   ├── ios  
    │   │   └── recursos  
    │   └── TiendaOnline  
    │       ├── código  
    │       │   ├── backend  
    │       │   │   ├── controllers  
    │       │   │   ├── models  
    │       │   │   └── routes  
    │       │   └── views  
    │       ├── frontend  
    │       │   ├── assets  
    │       │   │   ├── css  
    │       │   │   └── js  
    │       │   └── components  
    │       └── documentación  
    ├── artículos  
    │   ├── archivo.txt  
    │   ├── archivo2.txt  
    │   ├── buscar_palabra.sh  
    │   ├── buscar_reemplazar.sh  
    │   └── menu.sh  
    ├── borradores  
    ├── imágenes  
    ├── plantillas  
    └── pruebas  
  
25 directories, 5 files
```

Ejercicio 3 - Docker

Ejecutar un contenedor interactivo:

- Crea un contenedor interactivo a partir de una imagen de Debian.
- Accede al contenedor usando la consola.
- Instala el paquete "htop" para monitorizar el sistema
- Sal del contenedor.



Crear un contenedor en segundo plano:

- Lanza un contenedor en segundo plano con un servidor Nginx.
- Accede al servidor web desde un navegador web.
- Visualiza los logs del contenedor.

Para estos dos puntos decidí automatizarlo utilizando la herramienta ‘ansible’:

Primero creamos un [inventory.ini](#)

```
[local]
localhost ansible_connection=local
```

Luego creamos un [playbook.yml](#)

```
---
- name: Configurar y ejecutar contenedores Docker
  hosts: localhost
  become: yes
  tasks:
    - name: Verificar si Docker está instalado
      command: docker --version
      register: docker_installed
      ignore_errors: yes

    - name: Instalar Docker
      apt:
        name: docker.io
        state: present
      when: docker_installed.rc != 0

    - name: Iniciar y habilitar el servicio Docker
      service:
        name: docker
        state: started
```



```
enabled: yes

- name: Ejecutar contenedor Nginx en segundo plano
  community.docker.docker_container:
    name: nginx-server
    image: nginx
    state: started
    detach: yes
    ports:
      - "8081:80"

- name: Ejecutar contenedor Debian en modo interactivo
  community.docker.docker_container:
    name: debian-htop
    image: debian
    state: started
    tty: yes
    command: /bin/bash

- name: Actualizar repositorios en el contenedor Debian
  community.docker.docker_container_exec:
    container: debian-htop
    command: apt-get update

- name: Instalar htop en el contenedor Debian
  community.docker.docker_container_exec:
    container: debian-htop
    command: apt-get install -y htop
```



Ejecutamos dicho playbook:

```
vagrant@ubuntu-focal:~/ansible$ ansible-playbook -i inventory playbook1.yml
[WARNING]: Unable to parse /home/vagrant/ansible/inventory as an inventory source
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'
[WARNING]: Collection community.docker does not support Ansible version 2.12.10

PLAY [Configurar y ejecutar contenedores Docker] *****

TASK [Gathering Facts] *****
ok: [localhost]

TASK [Verificar si Docker está instalado] *****
changed: [localhost]

TASK [Instalar Docker] *****
skipping: [localhost]

TASK [Iniciar y habilitar el servicio Docker] *****
ok: [localhost]

TASK [Ejecutar contenedor Nginx en segundo plano] *****
changed: [localhost]

TASK [Ejecutar contenedor Debian en modo interactivo] *****
changed: [localhost]

TASK [Actualizar repositorios en el contenedor Debian] *****
changed: [localhost]

TASK [Instalar htop en el contenedor Debian] *****
changed: [localhost]

PLAY RECAP *****
localhost          : ok=7    changed=5    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
```

Listamos los contenedores ejecutando un docker ps:

vagrant@ubuntu-focal:~/ansible\$ docker ps						
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
940e72f998a4	debian	"/bin/bash"	8 seconds ago	Up 7 seconds		debian-htop
21f478b47530	nginx	"/docker-entrypoint..."	9 seconds ago	Up 8 seconds	0.0.0.0:8081->80/tcp	nginx-server

Ingresamos a el contenedor de debian ejecutado:

```
docker exec -it 940 /bin/bash
```

Luego dentro del contenedor ejecutamos `htop`

The figure shows a terminal window running the htop command. The top section displays system metrics: CPU usage (Tasks: 3, 0 thr, 1 running), load average (0.06 over 10, 0.04 over 60), and uptime (04:36:57). Below these are memory usage (Mem: 831M/5.79G) and swap usage (Swap: 0K/0K). The bottom section is a detailed process list with columns for PID, USER, PRI, NI, VIRT, RES, SHR, S, CPU%, MEM%, TIME+, and Command. It lists three processes: root at PID 1, root at PID 179, and root at PID 185, all running /bin/bash or htop.

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
1	root	20	0	4188	3264	2992	S	0.0	0.1	0:00.01	/bin/bash
179	root	20	0	4188	3472	2968	S	0.0	0.1	0:00.01	/bin/bash
185	root	20	0	4916	3836	2928	R	0.0	0.1	0:00.01	htop

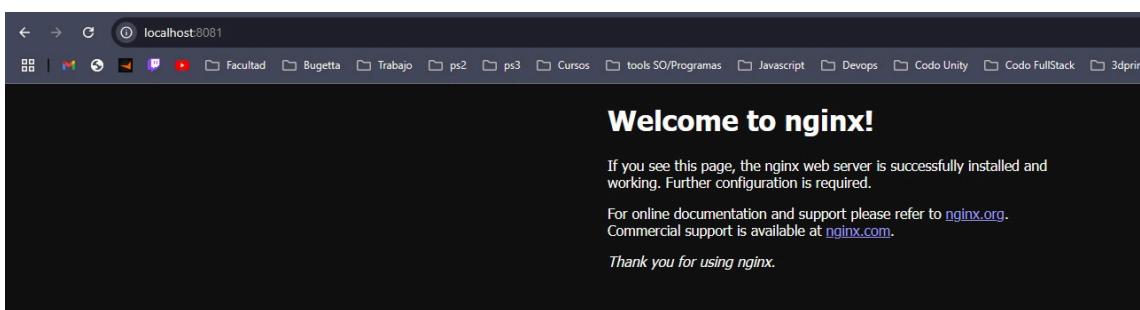


En otra terminal ejecutamos

`vagrant ssh -L 8081:localhost:8081` (genera el túnel entre la máquina host y la vm)

Y por último accedemos mediante el navegador utilizando

<http://localhost:8081/>



Por último para visualizar los logs del contenedor ejecutamos el siguiente comando: `docker logs -f nginx-server`

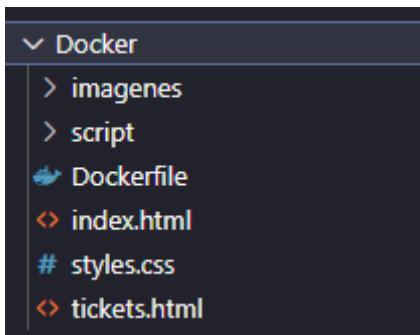
```
vagrant@ubuntu-focal:~$ docker logs -f nginx-server
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: Info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: Info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2024/11/05 12:25:29 [notice] #1: using the "epoll" event method
2024/11/05 12:25:29 [notice] #1: nginx/1.27.2
2024/11/05 12:25:29 [notice] #1: built by gcc 12.2.0 (Debian 12.2.0-14)
2024/11/05 12:25:29 [notice] #1: OS: Linux 5.4.0-176-generic
2024/11/05 12:25:29 [notice] #1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2024/11/05 12:25:29 [notice] #1: start worker processes
2024/11/05 12:25:29 [notice] #1: start worker process 29
2024/11/05 12:25:29 [notice] #1: start worker process 30
2024/11/05 12:25:29 [notice] #1: start worker process 31
2024/11/05 12:25:29 [notice] #1: start worker process 32
172.17.0.1 - [05/Nov/2024:12:33:00 +0000] "GET / HTTP/1.1" 200 615 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/130.0.0.0 Safari/537.36"
2024/11/05 12:33:00 [error] 31#31: *2 open() "/usr/share/nginx/html/favicon.ico" failed (2: No such file or directory), client: 172.17.0.1, server: localhost, request: "GET /favicon.ico HTTP/1.1", host: "localhost:8081", referrer: "http://localhost:8081/"
172.17.0.1 - [05/Nov/2024:12:33:00 +0000] "GET /favicon.ico HTTP/1.1" 404 555 "http://localhost:8081/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/130.0.0.0 Safari/537.36" "-"
```

Crear una imagen personalizada:

- Crea una página web estática sencilla (por ejemplo, un archivo HTML).
- Escribe un Dockerfile que configure un servidor web para servir la página.
- Crea una nueva imagen a partir del Dockerfile.
- Ejecuta un contenedor a partir de la imagen personalizada.



Utiliza una web creada previamente por mí (en el curso Codo a Codo Fullstack de Java). Creamos un directorio llamado **Docker** y luego colocamos los archivos de la web y un archivo llamado **Dockerfile**.



Dentro del Dockerfile, vamos a realizar la siguiente configuración:

- **FROM httpd:2.4**: utilizamos como imagen base Apache2 versión 2.4.
- **COPY ./ /usr/local/apache2/htdocs**: copiamos el contenido del directorio Docker al directorio de documentos de Apache2.
- **EXPOSE 80**: configuramos dónde se va a exponer el contenedor, en este caso, el puerto 80.

A esta altura, ya estoy utilizando un repositorio de GitHub donde subí el contenido de la carpeta **Vagrant** y el contenido de la carpeta **Docker**.

Para tener los archivos dentro de la máquina de Vagrant, realicé esta modificación en el **bootstrap.sh**.

```
# Clonar el repositorio
git clone https://github.com/fcongedo/Desafio-DevOps
```

Vuelvo a ejecutar el **vagrant reload -provision** (para clonar el repo)



Luego nos situamos en el directorio Docker

```
vagrant@ubuntu-focal:~/Desafio-DevOps/Docker$ ls
Dockerfile  imagenes  index.html  script  styles.css  tickets.html
```

Ejecutamos el siguiente comando para crear la imagen:

```
docker build -t website-apache:1.0
```

```
vagrant@ubuntu-focal:~/Desafio-DevOps/Docker$ docker build -t website-apache:1.0 .
[+] Building 26.8s (7/7) FINISHED
   => [internal] Load build definition from Dockerfile
   => => transferring dockerfile: 295B
   => [internal] Load metadata for docker.io/library/httpd:2.4.58
   => [internal] Load .dockerignore
   => => transferring context: 2B
   => [internal] Load build context
   => => transferring context: 2.34MB
[1/2] FROM docker.io/library/httpd:2.4.58@sha256:374766f5bc5977c9b72fd8ae3ed05b7fc89060e7edc88fcfb142d6988e58eeb
--> => resolve docker.io/library/httpd:2.4.58@sha256:374766f5bc5977c9b72fd8ae3ed05b7fc89060e7edc88fcfb142d6988e58eeb
--> => sha256:1a3d41a9f66b29d48cafe57e9f5ed8d539b12cafb9062488be377f5c86ab 2.10kB / 2.10kB
--> => sha256:a45b24b92cc8527cafa600679d0701f688a6d4214cf5cf9a147f28127d9685e 8.02kB / 8.02kB
--> => sha256:4f4fb700efc54461cfa2571ae0db9ad01e0cb557484a6d75e8d38e8ac1 32B / 32B
--> => sha256:374766f5bc5977c9b72fd8ae3ed05b7fc89060e7edc88fcfb142d6988e58eeb 9.74kB / 9.74kB
--> => sha256:8a1e25ce7c4f75e372e0884f8f7b1bedcfe4a7a7d452e4b9a0tc7477c9a90345 29.12MB / 29.12MB
--> => sha256:8ba0a7c8478f88543c0fc3c85342abb736016bc8fc281049eaa1deff897fcf854 145B / 145B
--> => sha256:91e4b2f2b52aca9f963647ae6408e999a528bcfec377d6377e2f96bbdf5a26 31.20MB / 31.20MB
--> => sha256:7787bba042e02e065725f2e4aedfc168e8c5e37d5e70488daca73ed0499678 4.20MB / 4.20MB
--> => extracting sha256:8a1e25ce7c4f75e372e0884f8f7b1bedcfe4a7a7d452e4b9a0tc7477c9a90345
--> => sha256:c78cdbf9617df5d6de4d728691c40987ef7d7e0d1ae651034d76879e53e862 293B / 293B
--> => extracting sha256:8ba0a7c8478f88543c0fc3c85342abb736016bc8fc281049eaa1deff897fcf854
--> => extracting sha256:4f4fb700efc54461cfa2571ae0db9ad01e0cb557484a6d75e8d38e8ac1
--> => extracting sha256:7fffbba042e02e065725f2e4aedfc168e8c5e37d5e70488daca73ed0499678
--> => extracting sha256:91e4b2f2b52aca9f963647ae6408e999a528bcfec377d6377e2f96bbdf5a26
--> => extracting sha256:c78cdbf9617df5d6de4d728691c4d9d87ef7e0d1ae651034d76879e53e862
--> [2/2] COPY ./ /usr/local/apache2/htdocs
--> exporting to image
--> => exporting layers
--> => writing image sha256:7b22fd2b750968004f9d78430da48df58db4417d1dfdd8fdfdd292e963bfaa
--> => naming to docker.io/library/website-apache:1.0
```

Por último corremos la imagen ejecutando el siguiente comando:

```
docker run -d -p 80:80 --name mi-contenedor-website-apache
website-apache:1.0
```

-d significa que ejecutamos en segundo plano,

-p 80:80 túnel entre puerto 80 host y puerto 80 contenedor,

--name mi-contenedor-website-apache nombre del contenedor,

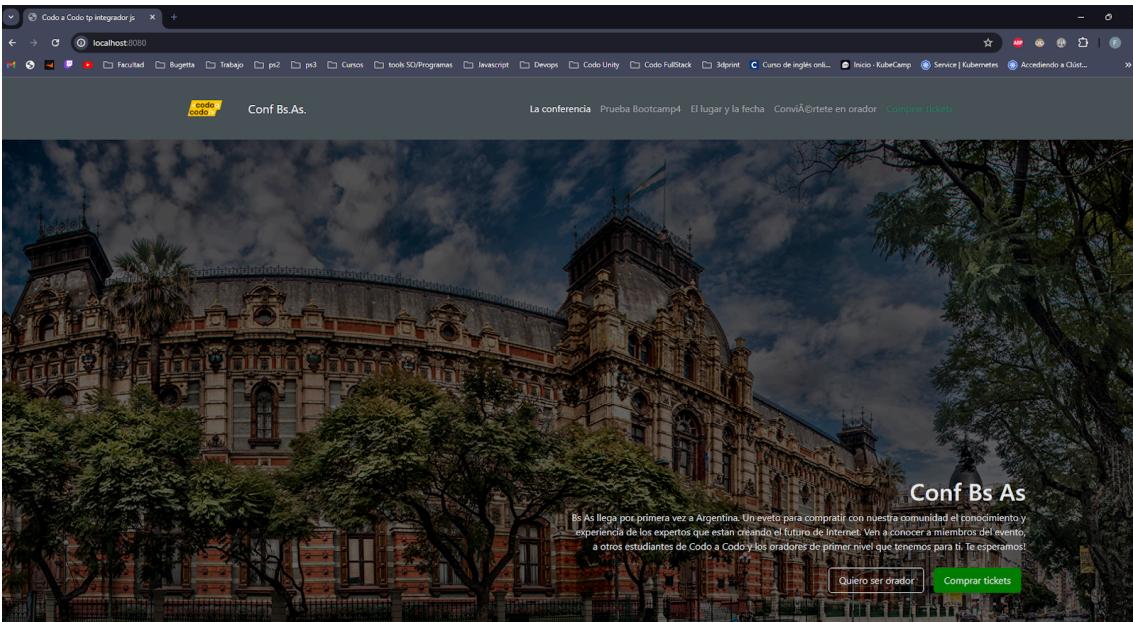
website-apache:1.0 contenedor y versión



```
vagrant@ubuntu-focal:~/Desafio-DevOps/Dockers$ docker run -d -p 80:80 --name mi-contenedor-website-apache website-apache:1.0
cfa282444f83e17e766fa95317551237230a72d9275104dc53c947db9b85574
vagrant@ubuntu-focal:~/Desafio-DevOps/Dockers$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
cfa282444f83 website-apache:1.0 "httpd-foreground" 14 minutes ago Up 14 minutes 0.0.0.0:80->80/tcp, ::80->80/tcp mi-contenedor-website-apache
```

Y por último accedemos mediante el navegador utilizando

<http://localhost:8080/>





Jenkins CI/CD

El objetivo de este ejercicio es crear un flujo de trabajo de CI/CD en Jenkins para implementar una aplicación web sencilla utilizando Docker. El flujo de trabajo automatizará las siguientes tareas:

1. Obtener el código fuente: Jenkins clonará el código fuente de la aplicación desde un repositorio Git.
2. Compilación: Instalar las dependencias y compilar el código fuente de la aplicación.
3. Desplegar la aplicación: Se implementará la aplicación en un servidor de destino, utilizando la imagen Docker subida

Instalacion y configuracion de Jenkins

Ahora accedemos a la interfaz de jenkins mediante el puente de ssh:

```
vagrant ssh -- -L 8080:localhost:8080
```

Luego nos conectamos por el navegador usando <http://localhost:8080/> y accedemos a la interfaz.



Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

`/var/lib/jenkins/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

Copiamos la ruta marcada en **rojo** y le hacemos un cat para poder acceder a la password.

```
vagrant@ubuntu-focal:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword  
0d76a0a1b3584129a15dc812aba4ad04
```

Pegamos dicha contraseña y empezamos la instalación



Getting Started

Getting Started

A progress bar at the bottom of the page shows approximately 25% completion.

✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	✓ Credentials Binding
⌚ Timestamper	⌚ Workspace Cleanup	⌚ Ant	⌚ Gradle
⌚ Pipeline	⌚ GitHub Branch Source	⌚ Pipeline: GitHub Groovy Libraries	⌚ Pipeline Graph View
⌚ Git	⌚ SSH Build Agents	⌚ Matrix Authorization Strategy	⌚ PAM Authentication
⌚ LDAP	⌚ Email Extension	⌚ Mailer	⌚ Dark Theme

On the right side of the table, there is a vertical sidebar with the following menu items:

- Ionicons API
- Folders
- OWASP Markup Formatter
 - ASM API
 - JSON Path API
 - Structs
 - Pipeline: Step API
 - Token Macro
- Build Timeout
 - bouncycastle API
 - Credentials
 - Plain Credentials
 - Variant
 - SSH Credentials
- Credentials Binding
 - SCM API
 - Pipeline: API

Luego creamos el usuario

Getting Started

Create First Admin User

Usuario

Contraseña

Confirma la contraseña

Nombre completo

Jenkins 2.414.3

Skip and continue as admin

Save and Continue



Getting Started

Instance Configuration

Jenkins URL:

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the `BUILD_URL` environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Getting Started

Jenkins is ready!

Your Jenkins setup is complete.

[Start using Jenkins](#)



Primero vamos a agregar las credenciales de Docker Hub
Para eso vamos a **administrar jenkins**

The screenshot shows the Jenkins dashboard. On the left, there's a sidebar with links: 'Nueva Tarea', 'Personas', 'Historial de trabajos', 'Administrar Jenkins' (which is highlighted with a red box), and 'Mis vistas'. The main area has a heading '¡Bienvenido a Jenkins!' and a sub-heading 'Start building your software project'. It includes buttons for 'Create a job', 'Set up a distributed build', 'Set up an agent', 'Configure a cloud', and 'Learn more about distributed builds'. At the bottom left, it says 'localhost:8080/view/all/builds'.

Luego a la opción **credentials**

The screenshot shows the 'System Configuration' page. It has sections for 'System', 'Tools', 'Plugins', 'Nodes', 'Clouds', 'Security', and 'Credential Providers'. The 'Credentials' link under the 'Security' section is highlighted with a red box.



Hacemos click en **System**

A screenshot of the Jenkins 'Credentials' page under 'Panel de Control > Administrar Jenkins > Credentials'. The title 'Credentials' is at the top. Below it is a table header with columns 'T', 'P', 'Store', 'Domain', 'ID', and 'Name'. A sub-section titled 'Stores scoped to Jenkins' shows a table with one row: a 'System' store with domain '(global)'. The 'System' icon in this row is highlighted with a red box. At the bottom of the table are icons for 'S', 'M', and 'L'.

Luego hacemos click en **Global credentials (unrestricted)**

A screenshot of the Jenkins 'System' page under 'Panel de Control > Administrar Jenkins > System'. The title 'System' is at the top. Below it is a table with a single row for 'Global credentials (unrestricted)'. The 'Global credentials (unrestricted)' icon is highlighted with a red box. The table has columns 'Domain' and 'Description'. The 'Description' column contains the text 'Credentials that should be available irrespective of domain specification to requirements matching.' At the bottom of the table are icons for 'S', 'M', and 'L'. A blue button labeled '+ Add domain' is visible in the top right corner.

Luego en **add credentials**

A screenshot of the Jenkins 'Global credentials (unrestricted)' page under 'Panel de Control > Administrar Jenkins > System'. The title 'Global credentials (unrestricted)' is at the top. Below it is a table with columns 'ID', 'Name', 'Kind', and 'Description'. A message at the top says 'Credentials that should be available irrespective of domain specification to requirements matching.' A red box highlights the blue '+ Add Credentials' button in the top right corner. The table body is currently empty, showing the message 'This credential domain is empty. How about adding some credentials?' At the bottom of the table are icons for 'S', 'M', and 'L'.



Por último configuramos las credenciales ingresando los datos.

Clíckeamos en **create**

New credentials

Kind: Username with password

Scope: Global (Jenkins, nodes, items, all child items, etc)

Username: fcongedo

Treat username as secret

Password:

ID: dockerhub

Description: credenciales de logear en dockerhub

Create

Credenciales creadas.

Credentials that should be available irrespective of domain specification to requirements matching.				
ID	Name	Kind	Description	
dockerhub	fcongedo/******** (credenciales para logear en dockerhub)	Username with password	credenciales para logear en dockerhub	

Voy a reutilizar la web alojada en la carpeta Docker

Creamos nuestro **Jenkinsfile**.

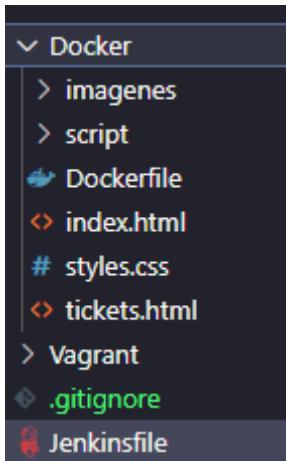
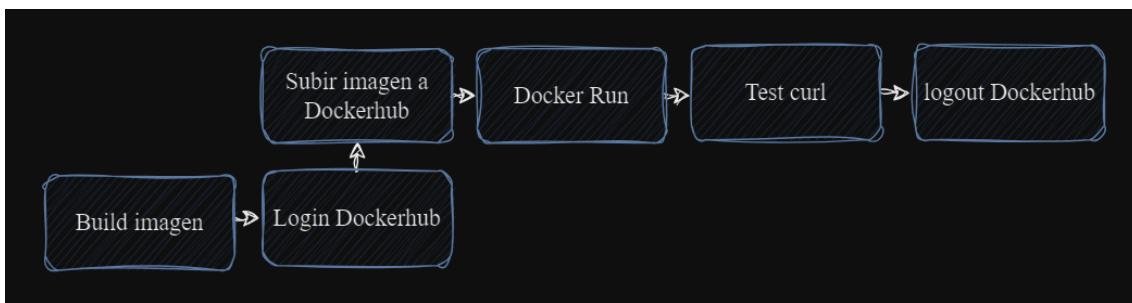


Diagrama CI/CD Docker



Explicación código Jenkinsfile

Primero asignamos que pueda usar cualquier agente de jenkins

Luego creamos variables de entorno para (credencial de dockerhub, nombre del repositorio de dockerhub y nombre del contenedor)

```
pipeline {  
    agent any  
    environment {  
        DOCKERHUB_CREDENTIALS = credentials('dockerhub')  
        RepositoryDockerHub = 'fcongedo'  
        NameContainer = "website-desafio"  
    }  
}
```



Luego Buildeamos la imagen

```
stages {
    stage('Build') {
        steps {
            dir('Docker') {
                sh "docker build -t
${env.RepositoryDockerHub}/${env.NameContainer}:${env.BUILD_NUMBER} ."
            }
        }
    }
}
```

Nos logueamos a Docker hub

```
stage('Login to Dockerhub') {
    steps {

        withCredentials([usernamePassword(credentialsId: 'dockerhub',
usernameVariable: 'DOCKERHUB_CREDENTIALS_USR',
passwordVariable: 'DOCKERHUB_CREDENTIALS_PSW')]) {
            sh "echo \$DOCKERHUB_CREDENTIALS_PSW |
docker login -u \$DOCKERHUB_CREDENTIALS_USR --password-stdin"
        }
    }
}
```

Pushear la imagen a Dockerhub

```
stage('Push image to Dockerhub') {
    steps {
        sh "docker push ${env.RepositoryDockerHub}/${env.NameContainer}:${env.BUILD_NUMBER}"
    }
}
```

Creamos el contenedor con la imagen (Detenemos la ejecución si ya hay una imagen en funcionamiento en el contenedor, eliminamos el contenedor existente y creamos uno nuevo para ejecutar una nueva imagen)

```
stage('Deploy container') {
```



```
steps {
    sh "docker stop ${env.NameContainer} || true"
    sh "docker rm -f ${env.NameContainer} || true"
    sh "docker run -d -p 8082:80 --name
${env.NameContainer}
${env.RepositoryDockerHub}/${env.NameContainer}:${env.BUILD_NUMBER}
"
}
}
```

Testeamos el funcionamiento de la imagen (ejecutando un curl)

```
stage('Test') {
    steps {
        script {
            sleep(time: 5, unit: 'SECONDS')
            def curlOutput = sh(script: "curl -s -o /dev/null
-w '%{http_code}' http://localhost:8082/", returnStdout: true).trim()
            echo "curlOutput: ${curlOutput}"

            if (curlOutput == '200') {
                currentBuild.result = 'SUCCESS'
            } else {
                currentBuild.result = 'FAILURE'
                error("El código de respuesta no es 200, en su
lugar es ${curlOutput}")
            }
        }
    }
}
```



Nos deslogueamos de Docker hub

```
stage('Docker logout') {  
    steps {  
        sh "docker logout"  
    }  
}
```

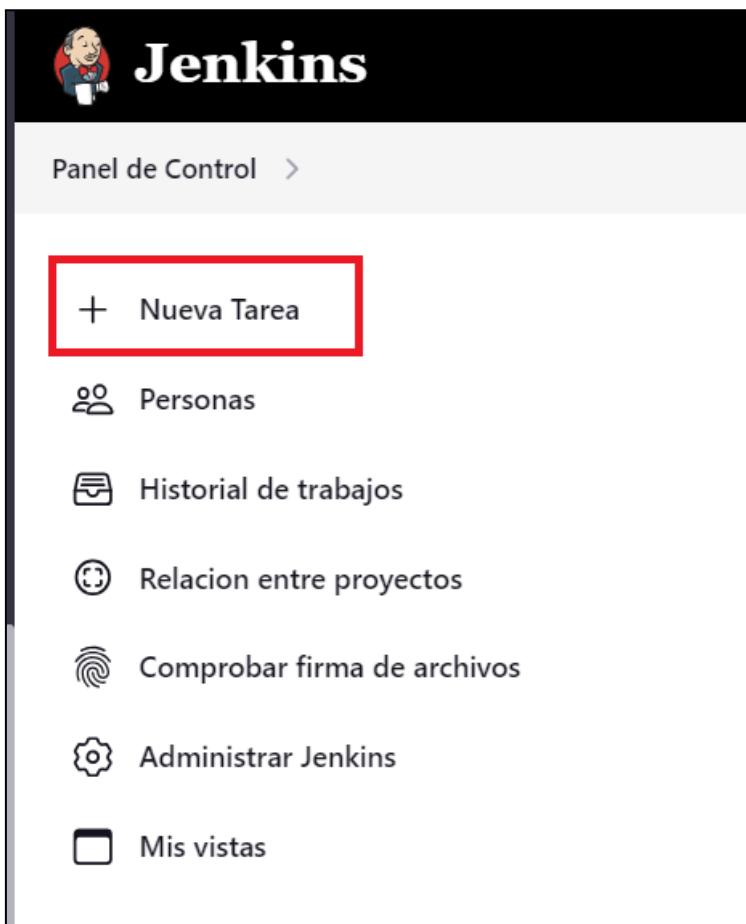
Luego de esto, tenemos que subir todo este contenido a un repositorio en github
(para asi despues usarlo en nuestro [pipeline](#))

fcongedo Agrego archivo Jenkinsfile		
	bf665e7 · 9 minutes ago	10 Commits
📁 Ansible	Agrego playbook y inventory de Ansible	1 hour ago
📁 Docker	Agrego archivos de Docker	1 hour ago
📁 Vagrant	Update bootstrap.sh	9 hours ago
📁 archivos_bash	Agrego archivos de bash	10 hours ago
📄 .gitignore	Update bootstrap.sh	10 hours ago
📄 Jenkinsfile	Agrego archivo Jenkinsfile	9 minutes ago
📄 README.md	first commit	15 hours ago



Creación del pipeline

Nos dirigimos a la parte izquierda y seleccionamos **nueva tarea**





Agregamos el nombre de nuestro pipeline (pipeline-website)

Y seleccionamos la opción **Pipeline**

Luego hacemos click en Ok

Nuevo Tarea

Enter an item name
pipeline-website

Select an item type

 **Crear un proyecto de estilo libre**
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

 **Pipeline**
Gestiona actividades de larga duración que pueden abarcar varios agentes de construcción. Apropiado para construir pipelines (conocidas anteriormente como workflows) y/o para la organización de actividades complejas que no se pueden articular fácilmente con tareas de tipo freestyle.

 **Crear un proyecto multi-configuración**
Adecuado para proyectos que requieran un gran número de configuraciones diferentes, como testear en múltiples entornos, ejecutar sobre plataformas concretas, etc.

 **Folder**
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

 **Multibranch Pipeline**
Creates a set of Pipeline projects according to detected branches in one SCM repository.

 **Organization Folder**
Creates a set of multibranch project subfolders by scanning for repositories.



Agregamos una descripción al proceso.

General

Enabled

Descripción

CI/CD website Dockerhub

Plain text [Visualizar](#)

Desechar ejecuciones antiguas [?](#)

Do not allow concurrent builds

Do not allow the pipeline to resume if the controller restarts

Esta ejecución debe parametrizarse [?](#)

GitHub project

Pipeline speed/durability override [?](#)

Preserve stashes from completed builds [?](#)

Throttle builds [?](#)

Build Triggers

Configura los desencadenadores para que el pipeline se ejecute automáticamente cuando haya cambios en el repositorio (por ejemplo, al hacer un push)

Build Triggers

Construir tras otros proyectos [?](#)

Ejecutar periódicamente [?](#)

GitHub hook trigger for GITScm polling [?](#)

Consultar repositorio (SCM) [?](#)

Periodo de espera [?](#)

Lanzar ejecuciones remotas (ejem: desde 'scripts') [?](#)



En la sección '**Pipeline**', elegimos la opción '**Pipeline script from SCM**'

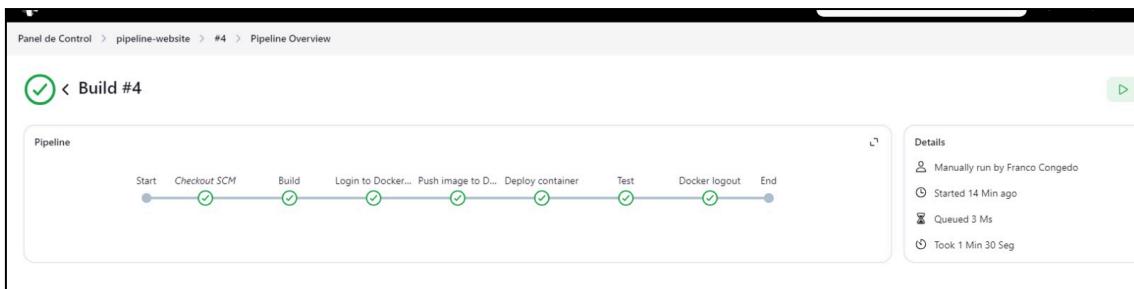
En **SCM** seleccionamos '**Git**'

En la sección '**Repositories**' agregamos el **URL** de nuestro repositorio.

The screenshot shows the Jenkins Pipeline configuration interface. In the 'Definition' section, 'Pipeline script from SCM' is selected. Under 'SCM', 'Git' is chosen. In the 'Repositories' section, a repository URL is set to 'https://github.com/fcongado/Desafio-DevOps'. The 'Branches to build' section has 'main' specified as the branch specifier. Other sections like 'Credentials' and 'Avanzado' are also visible.

Por último vamos a abajo de todo y le damos a save.

El proceso va a correr automáticamente y veremos si funciona o no.



Verificamos en la máquina con vagrant

Ejecutando un **docker images** y **docker ps**

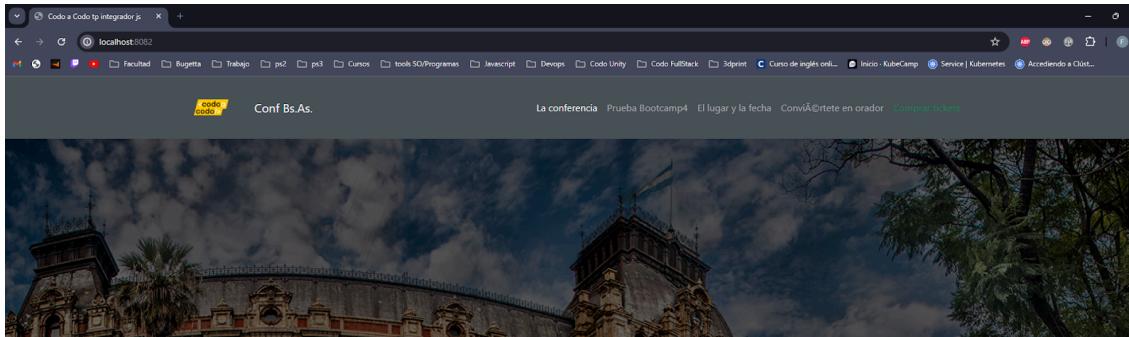


```
vagrant@ubuntu-focal:~$ sudo docker images
REPOSITORY          TAG      IMAGE ID      CREATED        SIZE
fcconedo/website-desafio  3       bdc5845f1695  18 minutes ago  170MB
fcconedo/website-desafio  4       bdc5845f1695  18 minutes ago  170MB
vagrant@ubuntu-focal:~$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
73176f0ca4a4        fcconedo/website-desafio:4   "httpd-foreground"   16 minutes ago   Up 16 minutes   0.0.0.0:8082->80/tcp, [::]:8082->80/tcp   website-desafio
```

Luego de esto nos conectamos a contenedor del website utilizando el túnel

```
vagrant ssh -- -L 8082:localhost:8082
```

Adjunto captura de imagen accediendo de localhost:8082



También haciendo un curl localhost:8082

```
vagrant@ubuntu-focal:~$ curl localhost:8082
<!DOCTYPE html>
<html lang="es">
<head>
  <!-- Meta información del documento -->
  <title>Codo a Codo tp integrador js</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/css/bootstrap.min.css" rel="stylesheet"
    integrity="sha384-F3w7mx95PdgymZMECAngseQB83DFGTowi0iMjIwaeVhAn4FJkqJByhZMI3AhU" crossorigin="anonymous">
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/js/bootstrap.bundle.min.js"
    integrity="sha384-/DQsTh/da6pk11MjJn5+XoJyQZ8/8UUQZl0Jd9KzvHIDuqinlDcD9eQ" crossorigin="anonymous"></script>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <!-- Barra de navegación -->
  <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
    <!-- Contenedor de la barra de navegación -->
    <div class="container">
      <!-- Logo de la conferencia -->
      <a class="navbar-brand" href="#"></a>
      <a class="navbar-brand" href="#">Conf Bs.As.</a>
      <!-- Botón para alternar la navegación en pantallas pequeñas -->
      <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNavAltMarkup"
        aria-controls="navbarNavAltMarkup" aria-expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
```



Verificamos en dockerhub si la imagen está subida.

A screenshot of the DockerHub interface for the repository 'fcongedo/website-desafio'. The 'Tags' tab is selected. It shows two tags: '4' (pushed 20 minutes ago) and '1' (pushed 3 months ago). Each tag entry includes a 'Digest' link, 'OS/ARCH' (linux/amd64), 'Last pull' information, and a 'Copy' button for the tag name. A 'Manage Repository' button and a 'Pulls 12' badge are visible at the top right.

Configuración de ngrok para exponer Jenkins

1. Debemos registrarnos en <https://ngrok.com/>
2. Instalamos **ngrok** en la VM

Para eso ejecutamos los siguientes comandos:

```
curl -sSL https://ngrok-agent.s3.amazonaws.com/ngrok.asc |  
sudo tee /etc/apt/trusted.gpg.d/ngrok.asc >/dev/null && echo  
"deb https://ngrok-agent.s3.amazonaws.com buster main" | sudo  
tee /etc/apt/sources.list.d/ngrok.list && sudo apt update &&  
sudo apt install ngrok
```

Al ingresar a la página nos van a brindar un token único, el cual luego de finalizar la instalación debemos utilizar. (el token es falso)

```
ngrok config add-authtoken 213213121392121301
```



Luego ejecutamos el siguiente comando para exponer [jenkins](#)

```
ngrok http 8080
```

Nos sale esta información (donde la el link forwarding es el que expone jenkins a internet, así puede conectarse con github)

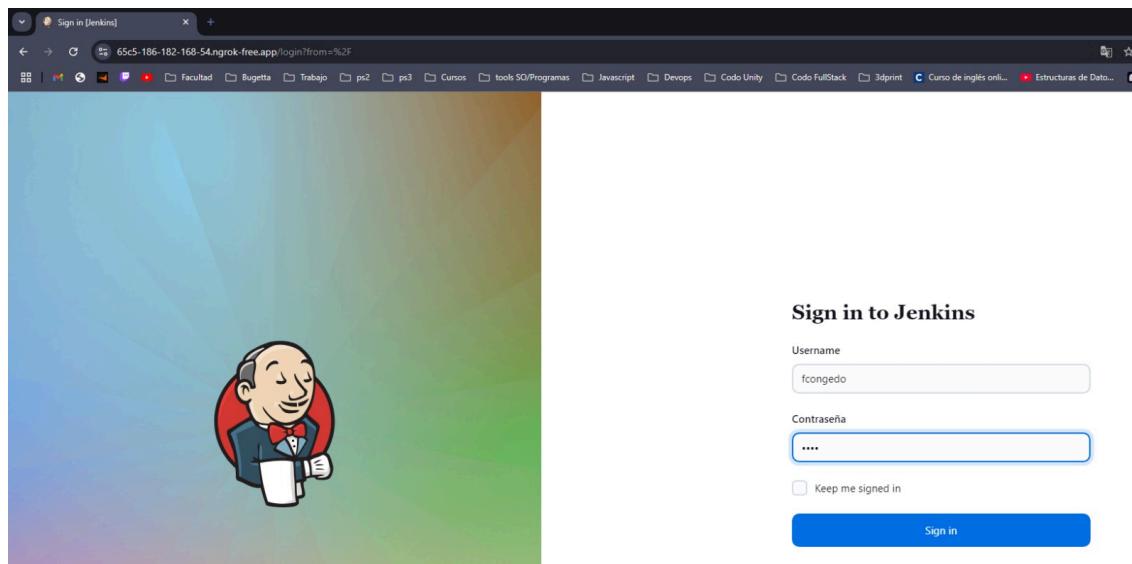
```
ngrok
Sign up to try new private endpoints https://ngrok.com/new-features-update?ref=private

Session Status          online
Account                 Franco (Plan: Free)
Version                3.18.2
Region                 South America (sa)
Latency                40ms
Web Interface          http://127.0.0.1:4040
Forwarding             https://65c5-186-182-168-54.ngrok-free.app -> http://localhost:8080

Connections            ttl     opn      rt1      rt5      p50      p90
                        3        0       0.01    0.01    30.01    30.85

HTTP Requests
-----
18:05:29.108 UTC GET /adjuncts/864bc701/io/jenkins/plugins/themanager/header/main.js 200 OK
18:05:29.948 UTC GET /static/864bc701/favicon.ico                               200 OK
18:05:29.108 UTC GET /static/864bc701/images/svg/logo.svg                      200 OK
18:05:29.106 UTC GET /static/864bc701/jsbundles/simple-page.css                  200 OK
18:05:28.763 UTC GET /                                         403 Forbidden
```

(si presionamos **ctrl+c** terminamos el proceso)





Configuración de Webhook en Github

Dentro del repositorio en **Github**.

1. Vamos al apartado '**Settings**'

A screenshot of a GitHub repository page titled 'Desafio-DevOps'. The 'Settings' tab is highlighted with a red box. The main area shows a list of commits from user 'fcongedo'. On the right side, there are sections for 'About', 'Releases', and 'Packages'. The 'About' section shows the repository name 'Desafio-DevOps' and its activity metrics: 1 star, 1 watching, and 0 forks.

2. Seleccionamos '**Webhooks**'

A screenshot of the GitHub repository settings page for 'Desafio-DevOps'. The left sidebar has a red box around the 'Webhooks' option under the 'General' heading. The main content area shows the 'General' settings with sections for 'Repository name' (set to 'Desafio-DevOps'), 'Template repository', 'Require contributors to sign off on web-based commits', and 'Default branch' (set to 'main'). Below that is the 'Social preview' section, which is currently empty. At the bottom is the 'Features' section.



3. Hacemos click en '**Add webhook**' (confirmamos acceso colocando contraseña de tu usuario de GitHub)

A screenshot of the GitHub 'Webhooks' settings page. The title 'Webhooks' is at the top left. On the right, there's a red box around the 'Add webhook' button. Below the title, a descriptive text explains what webhooks are and points to a 'Webhooks Guide'. A table lists one webhook entry: a green checkmark next to a URL, followed by '(push)'. To the right of the URL are 'Edit' and 'Delete' buttons. A note below the table says 'Last delivery was successful.'.

1. En el campo '**Payload URL**', ingresamos la url que creamos en ngrok seguido de **/github-webhook/**.
2. Configuramos el **type** como '**application/json**'.
3. Seleccionamos '**Just the push event**'.
4. Guardamos haciendo click en '**Add webhook**'.



Webhooks / Add webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

4. Payload URL *

5. Content type *

Secret

SSL verification
 By default, we verify SSL certificates when delivering payloads.
 Enable SSL verification Disable (not recommended)

Which events would you like to trigger this webhook?

6. Just the push event. Send me everything. Let me select individual events.

7. Active
We will deliver event details when this hook is triggered.

Add webhook

Ahora por último vamos a hacer un **push** de prueba para comprobar el funcionamiento de **Webhook**, y ver si ejecuta el pipeline en Jenkins.

- PS E:\Escritorio\Desafio-DevOps> git commit --allow-empty -m "Mensaje del commit vacío"
[main a8e0388] Mensaje del commit vacío
- PS E:\Escritorio\Desafio-DevOps> git push



Webhook funcionando

A screenshot of a webhooks configuration page. At the top, there's a header with the title "Webhooks" and a "Add webhook" button. Below the header, a descriptive text explains what webhooks are and points to a "Webhooks Guide". A list shows one active webhook entry: "https://65c5-186-182-168-54.ngrok... (push)". To the right of this entry are "Edit" and "Delete" buttons. Below the list, a message states "Last delivery was successful.".

Captura de la build #5 en estado correcto

A screenshot of a CI/CD pipeline status page for a pipeline named "pipeline-website". The main header shows a green checkmark icon and the pipeline name. Below the header, there are several navigation links: "Status", "</> Changes", "Construir ahora", "Configurar", "Borrar Pipeline", "Stages", "Rename", "Pipeline Syntax", and "GitHub Hook Log". On the right side, under the heading "Enlaces permanentes", there is a list of links related to the pipeline's history. At the bottom, there is a section titled "Builds" with a "Builds" button, a "Filter" input field, and a "Today" section listing two builds: "#5 18:16" and "#4 17:37", both of which have green checkmarks next to them.



Y Como vemos fue ejecutado con el commit "**Mensaje del commit vacío**"

```
Started by GitHub push by fcongedo
Obtained Jenkinsfile from git https://github.com/fcongedo/Desafio-DevOps
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/pipeline-website
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/pipeline-website/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/fcongedo/Desafio-DevOps # timeout=10
Fetching upstream changes from https://github.com/fcongedo/Desafio-DevOps
> git --version # timeout=10
> git --version # 'git version 2.25.1'
> git fetch --tags --force --progress -- https://github.com/fcongedo/Desafio-DevOps +refs/heads/*:refs/remotes/origin/*
> git rev-parse origin/main^{commit} # timeout=10
Checking out Revision a8e038855fc10f99d7dd115fb7391234d7495b21 (origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f a8e038855fc10f99d7dd115fb7391234d7495b21 # timeout=10
Commit message: "Mensaje del commit vacío"
> git rev-list --no-walk bf665e7881e77b0c8969dd71a5322f4f8ac0df5a # timeout=10
[Pipeline]
```

Salida completa de consola:

```
Started by Github push by fcongedo
Obtained Jenkinsfile from git
https://github.com/fcongedo/Desafio-DevOps
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in
/var/lib/jenkins/workspace/pipeline-website
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir
```



```
/var/lib/jenkins/workspace/pipeline-website/.git # timeout=10
Fetching changes from the remote Git repository
  > git config remote.origin.url
https://github.com/fcongedo/Desafio-DevOps # timeout=10
Fetching upstream changes from
https://github.com/fcongedo/Desafio-DevOps
  > git --version # timeout=10
  > git --version # 'git version 2.25.1'
  > git fetch --tags --force --progress --
https://github.com/fcongedo/Desafio-DevOps
+refs/heads/:refs/remotes/origin/ # timeout=10
  > git rev-parse origin/main^{commit} # timeout=10
Checking out Revision a8e038855fc10f99d7dd115fb7391234d7495b21
(origin/main)
  > git config core.sparsecheckout # timeout=10
  > git checkout -f a8e038855fc10f99d7dd115fb7391234d7495b21 # 
timeout=10
Commit message: "Mensaje del commit vacío"
  > git rev-list --no-walk
bf665e7881e77b0c8969dd71a5322f4f8ac0df5a # timeout=10
[Pipeline] }
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
[Pipeline] withCredentials
Masking supported pattern matches of $DOCKERHUB_CREDENTIALS or
$DOCKERHUB_CREDENTIALS_PSW
[Pipeline] {
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] dir
Running in /var/lib/jenkins/workspace/pipeline-website/Docker
[Pipeline] {
[Pipeline] sh
```



```
+ docker build -t fcongedo/website-desafio:5 .
#0 building with "default" instance using docker driver

#1 [internal] load build definition from Dockerfile
#1 transferring dockerfile: 295B done
#1 DONE 0.0s

#2 [internal] load metadata for docker.io/library/httpd:2.4.58
#2 DONE 1.3s

#3 [internal] load .dockerignore
#3 transferring context: 2B done
#3 DONE 0.0s

#4 [1/2] FROM
docker.io/library/httpd:2.4.58@sha256:374766f5bc5977c9b72fdb8a
e3ed05b7fc89060e7edc88fcbf142d6988e58eeb
#4 DONE 0.0s

#5 [internal] load build context
#5 transferring context: 450B done
#5 DONE 0.0s

#6 [2/2] COPY ./ /usr/local/apache2/htdocs
#6 CACHED

#7 exporting to image
#7 exporting layers done
#7 writing image
sha256:bdc5845f1695bbb537e94ce7327753623bf235f442896fa86ec4c95
540173fd0 done
#7 naming to docker.io/fcongado/website-desafio:5 done
#7 DONE 0.0s
[Pipeline] }
[Pipeline] // dir
[Pipeline] }
```



```
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Login to Dockerhub)
[Pipeline] withCredentials
Masking supported pattern matches of
$DOCKERHUB_CREDENTIALS_PSW
[Pipeline] {
[Pipeline] sh
+ docker login -u fcongedo --password-stdin
+ echo **
WARNING! Your password will be stored unencrypted in
/var/lib/jenkins/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credential-stores

Login Succeeded
[Pipeline] }
[Pipeline] // withCredentials
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Push image to Dockerhub)
[Pipeline] sh
+ docker push fcongedo/website-desafio:5
The push refers to repository
[docker.io/fcongedo/website-desafio]
653592ccd6aa: Preparing
d91409980a4e: Preparing
2ef15bc08e25: Preparing
1f9ac7ca16f1: Preparing
5f70bf18a086: Preparing
3ee8143dd880: Preparing
a483da8ab3e9: Preparing
3ee8143dd880: Waiting
a483da8ab3e9: Waiting
```



```
2ef15bc08e25: Layer already exists
1f9ac7ca16f1: Layer already exists
653592ccd6aa: Layer already exists
d91409980a4e: Layer already exists
5f70bf18a086: Layer already exists
3ee8143dd880: Layer already exists
a483da8ab3e9: Layer already exists
5: digest:
sha256:439553a2602864eb39a33d02fab47623090b07159adff9b25095154
26bbe9ec3 size: 1783
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Deploy container)
[Pipeline] sh
+ docker stop website-desafio
website-desafio
[Pipeline] sh
+ docker rm -f website-desafio
website-desafio
[Pipeline] sh
+ docker run -d -p 8082:80 --name website-desafio
fcongedo/website-desafio:5
236d2a0c6e7697f725920c478989ab66e7745888cea273630cc1a5ee27c7d8
4a
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Test)
[Pipeline] script
[Pipeline] {
[Pipeline] sleep
Sleeping for 5 sec
[Pipeline] sh
+ curl -s -o /dev/null -w %{http_code} http://localhost:8082/
[Pipeline] echo
```



```
curlOutput: 200
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Docker logout)
[Pipeline] sh
+ docker logout
Removing login credentials for https://index.docker.io/v1/
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withCredentials
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Links

- [Link repositorio GitHub Desafío-DevOps](#)
- [Imagen Docker Hub - Website con CI/CD Jenkins](#)