



Desafío Técnico: Demuestra Tu Talento y Únete a ELDAR

Challenge DEVOPS

Entorno de Vagrant

Se creará el entorno de Vagrant utilizando el Box Focal64 compuesto por una imagen Linux Ubuntu

Desde la web <https://app.vagrantup.com/ubuntu/boxes/focal64> se obtiene la información para usar el box

Para utilizar Vagrant, es necesario instalarlo en el sistema operativo a utilizar desde la web <https://developer.hashicorp.com/vagrant/downloads>

Primero, creamos una carpeta en el escritorio. Luego abrimos Visual Studio Code (VSC) y abrimos la carpeta recién creada. Por último abrimos una nueva terminal.

Y creamos una subcarpeta Vagrant

Adjunto captura de terminal:

```
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 hacemos un **vagrant init** (con esto creamos el Vagrantfile)

Adjunto captura:

<pre>✓ Vagrant └─ Vagrantfile</pre>	<pre>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> </pre>
-------------------------------------	---

Una vez creado el **Vagrantfile**, lo configuramos agregando las siguientes líneas:

Primera línea configuramos el nombre de la vm

Segunda línea le decimos que box vamos a utilizar

```
config.vm.define "desafio" do |desafio|
  desafio.vm.box = "ubuntu/focal64"
end
```

Modificamos los valores de memoria y cpus

```
#   # Customize the amount of memory on the VM:
#   vb.memory = "6144"
#   vb.cpus = 4
end
```

Modificamos el nombre

```
config.vm.provider "virtualbox" do |vb|
#   # Display the VirtualBox GUI when booting the machine
#   vb.gui = true
  vb.name = "desafio"
```

Descomentamos esta línea para acceder al contenedor desde '**localhost:8080**'

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

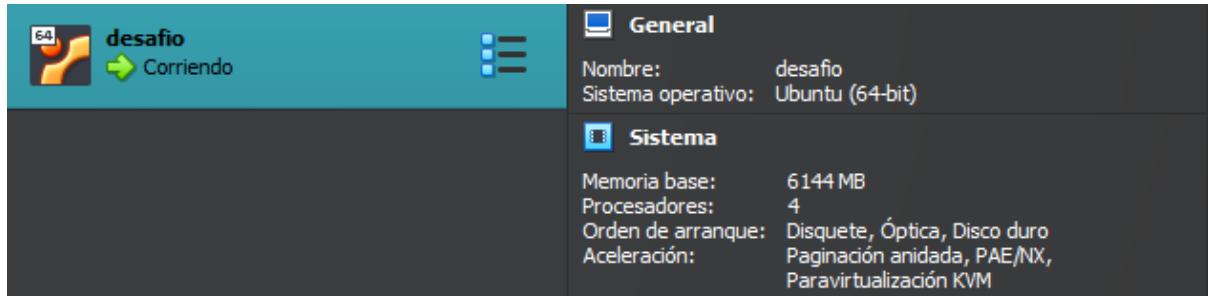
Ejecutamos un **vagrant up**

- 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 el virtual box las características de la vm (6gb y 4 núcleos)



Modificamos en el **Vagrantfile**, el shell para utilizar un script para la instalación de las distintas herramientas que vamos a utilizar a lo largo del desafío.

```
# Enable provisioning with a shell script. Additional provisioners such as
# Ansible, Chef, Docker, Puppet and Salt are also available. Please see the
# documentation for more information about their specific syntax and use.
config.vm.provision "shell", path: "bootstrap.sh"
```

En la misma carpeta creamos un **bootstrap.sh** con los siguientes comandos:

Para visualizar el árbol de directorios instale la herramienta **tree**

Entonces agregamos a nuestro **bootstrap.sh** los siguientes comandos:

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

Para la etapa de **Docker** vamos a instalar dicha herramienta.

Entonces agregamos a nuestro **bootstrap.sh** los siguientes comandos:

```
Vagrant > $ Bootstrap.sh
1 /bin/bash
2
3 √ # Update and Upgrade
4     sudo apt-get update -y
5
6
7 # Docker Install
8 curl -fsSL https://get.docker.com -o get-docker.sh
9 sudo sh ./get-docker.sh
10
11 # Use Docker without sudo
12 sudo groupadd docker
13 sudo usermod -aG docker vagrant
14 newgrp docker
```

Para la etapa de **jenkins** vamos a instalar dicha herramienta

(no olvidar que necesita java)

Entonces agregamos a nuestro **bootstrap.sh** los siguientes comandos:

```
# Install Jenkins and Java
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
https://pkg.jenkins.io/debian-stable/jenkins.io.key
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
sudo apt-get update -y
sudo apt-get install fontconfig openjdk-11-jre -y
sudo apt-get install jenkins -yy
```

- **Verificamos instalación de herramientas**

Verificamos las instalaciones haciendo un **tree --version**

```
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:~$ cd /home/vagrant/Documentos
```

Verificamos las instalaciones haciendo un **docker --version**

```
vagrant@ubuntu-focal:~$ docker --version
Docker version 27.1.1, build 6312585
```

Verificamos las instalaciones haciendo un **jenkins --version**

```
vagrant@ubuntu-focal:~$ jenkins --version
2.452.3
```

- **Linux**

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/frontend/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/archivo.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
```

- **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 voy a crear otro archivo en el directorio donde ya tenemos alojado el archivo.txt

Ejecutamos:

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

Luego

```
echo "Hola" >
```

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

Por último listamos con un **ls**

```
vagrant@ubuntu-focal:~/Documentos/Proyectos/TiendaOnline$ touch /home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/archivo2.txt
vagrant@ubuntu-focal:~/Documentos/Proyectos/TiendaOnline$ echo "Hola" > /home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/archivo2.txt
vagrant@ubuntu-focal:~/Documentos/Proyectos/TiendaOnline$ ls /home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/
archivo.txt archivo2.txt
```

Luego vamos a **cd /home/vagrant/Documentos/Proyectos/BlogPersonal/artículos**

Creamos el archivo utilizando **nano buscar_palabra.sh**

```
GNU nano 4.8                                     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" .
```

Y guardamos presionando **Ctrl + O**, luego enter y **Ctrl + X** (para salir)

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:~/Documentos/Proyectos/BlogPersonal/artículos$ 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 archivo utilizando `nano 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:"
    read archivo_original
    echo "Ingrese el destino para copiar el archivo:"
    read destino
    cp "$archivo_original" "$destino"
    echo "Archivo copiado exitosamente."
}

# Función para eliminar un archivo
eliminar_archivo() {
    echo "Ingrese el nombre del archivo a eliminar:"
    read archivo
    rm "$archivo"
    echo "Archivo eliminado exitosamente."
}

# Bucle principal del menú
while true; do
    mostrar_menu
    read opcion
    case $opcion in
```

```

1)
    ver_lista_archivos
    ;;
2)
    copiar_archivo
    ;;
3)
    eliminar_archivo
    ;;
4)
    echo "Saliendo..."
    exit 0
    ;;
*)
    echo "Opción inválida. Intente de nuevo."
    ;;
esac
done

```

Y guardamos presionando **Ctrl + O**, luego enter y **Ctrl + X** (para salir)

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

```
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
```

Seleccionamos (1) listar archivos.

```

1
Lista de archivos en el directorio actual:
total 16
-rw-rw-r-- 1 vagrant vagrant 30 Jul 24 23:25 archivo.txt
-rw-rw-r-- 1 vagrant vagrant 5 Jul 24 23:25 archivo2.txt
-rwxrwxr-x 1 vagrant vagrant 332 Jul 24 23:39 buscar_palabra.sh
-rwxrwxr-x 1 vagrant vagrant 1137 Jul 25 00:00 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  
2  
Ingrese el nombre del archivo a copiar:  
archivo.txt  
Ingrese el destino para copiar el archivo:  
/home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/archivo1_copia.txt  
Archivo copiado exitosamente.  
Seleccione una opción:
```

Luego listamos los archivos para ver si se copió correctamente

```
1  
Lista de archivos en el directorio actual:  
total 20  
-rw-rw-r-- 1 vagrant vagrant    30 Jul 24 23:25 archivo.txt  
-rw-rw-r-- 1 vagrant vagrant    30 Jul 25 00:12 archivo1_copia.txt  
-rw-rw-r-- 1 vagrant vagrant     5 Jul 24 23:25 archivo2.txt  
-rwxrwxr-x 1 vagrant vagrant  332 Jul 24 23:39 buscar_palabra.sh  
-rwxrwxr-x 1 vagrant vagrant 1137 Jul 25 00:00 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 y salimos (4).

```
1
Lista de archivos en el directorio actual:
total 16
-rw-rw-r-- 1 vagrant vagrant    30 Jul 24 23:25 archivo.txt
-rw-rw-r-- 1 vagrant vagrant     5 Jul 24 23:25 archivo2.txt
-rwxrwxr-x 1 vagrant vagrant  332 Jul 24 23:39 buscar_palabra.sh
-rwxrwxr-x 1 vagrant vagrant 1137 Jul 25 00:00 menu.sh
Seleccione una opción:
1. Ver la lista de archivos
2. Copiar un archivo
3. Eliminar un archivo
4. Salir
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$
```

Luego creamos el archivo utilizando **nano buscar_reemplazar.sh**

```
GNU nano 4.8                                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'."
```

Y guardamos presionando **Ctrl + O**, luego **enter** y **Ctrl + X** (para salir)

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 “demostracion”)

```
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'.
vagrant@ubuntu-focal:~/Documentos/Proyectos/BlogPersonal/artículos$
```

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 de el mensaje de cuando no encuentra el archivo a buscar.

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

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

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

Y obtenemos que no existe:

```
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.
```

Decidí agregar algunas automatizaciones

Primero, en el **Vagrantfile**, automatizé la creación del árbol de directorios solicitado:

```
# Crear directorios necesarios
mkdir -p /home/vagrant/Documentos/Proyectos/BlogPersonal \
          /home/vagrant/Documentos/Proyectos/BlogPersonal/artículos \
          /home/vagrant/Documentos/Proyectos/BlogPersonal/borradores \
          /home/vagrant/Documentos/Proyectos/BlogPersonal/ímágenes \
          /home/vagrant/Documentos/Proyectos/BlogPersonal/plantillas \
          /home/vagrant/Documentos/Proyectos/ProyectoAppMovil \
          /home/vagrant/Documentos/Proyectos/ProyectoAppMovil/android \
          /home/vagrant/Documentos/Proyectos/ProyectoAppMovil/documentación \
          /home/vagrant/Documentos/Proyectos/ProyectoAppMovil/ios \
          /home/vagrant/Documentos/Proyectos/ProyectoAppMovil/recursos \
          /home/vagrant/Documentos/Proyectos/TiendaOnline \
          /home/vagrant/Documentos/Proyectos/TiendaOnline/código/backend/controllers \
          /home/vagrant/Documentos/Proyectos/TiendaOnline/código/backend/models \
          /home/vagrant/Documentos/Proyectos/TiendaOnline/código/backend/routes \
          /home/vagrant/Documentos/Proyectos/TiendaOnline/código/backend/views \
          /home/vagrant/Documentos/Proyectos/TiendaOnline/código/frontend/assets/css \
          /home/vagrant/Documentos/Proyectos/TiendaOnline/código/frontend/assets/js \
          /home/vagrant/Documentos/Proyectos/TiendaOnline/código/frontend/components \
          /home/vagrant/Documentos/Proyectos/TiendaOnline/código/scripts \
          /home/vagrant/Documentos/Proyectos/TiendaOnline/documentación \
          /home/vagrant/Documentos/Proyectos/TiendaOnline/pruebas
```

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

Luego en el **Bootstrap.sh** agregue lo siguiente:

En Primer lugar, utilicé el comando sudo cp para copiar los archivos de prueba y los scripts de bash desde la carpeta **archivos_bash** al directorio que había creado anteriormente

```
# Copy files to destination directory
sudo cp /home/vagrant/Challenge-DEVOPS/archivos_bash/archivo2.txt /home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/
sudo cp /home/vagrant/Challenge-DEVOPS/archivos_bash/archivo.txt /home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/
sudo cp /home/vagrant/Challenge-DEVOPS/archivos_bash/buscar_palabra.sh /home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/
sudo cp /home/vagrant/Challenge-DEVOPS/archivos_bash/buscar_reemplazar.sh /home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/
sudo cp /home/vagrant/Challenge-DEVOPS/archivos_bash/menu.sh /home/vagrant/Documentos/Proyectos/BlogPersonal/artículos/
```

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**.

```
# Change permissions to execute for all .sh files in the destination directory
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:~$ tree Documentos
Documentos
└── Proyectos
    ├── BlogPersonal
    │   ├── artículos
    │   │   ├── archivo.txt
    │   │   ├── archivo2.txt
    │   │   ├── buscar_palabra.sh
    │   │   └── buscar_reemplazar.sh
    │   └── menu.sh
    ├── borradores
    ├── imágenes
    └── plantillas
    └── ProyectoAppMovil
        ├── android
        ├── documentación
        ├── ios
        └── recursos
    └── TiendaOnline
        ├── código
        │   ├── backend
        │   │   ├── controllers
        │   │   ├── models
        │   │   ├── routes
        │   │   └── views
        │   └── frontend
        │       ├── assets
        │       │   ├── css
        │       │   └── js
        │       └── components
        └── scripts
        └── documentación
        └── pruebas
```

- **Docker**

Ejecutar un contenedor interactivo:

Primero ejecutamos el comando

docker run -it debian /bin/bash (-it es modo interactivo)

```
vagrant@ubuntu-focal:~$ docker run -it debian /bin/bash
Unable to find image 'debian:latest' locally
latest: Pulling from library/debian
ca4e5d672725: Pull complete
Digest: sha256:45f2e735295654f13e3be10da2a6892c708f71a71be845818f6058982761a6d3
Status: Downloaded newer image for debian:latest
root@b6f7175afa84:/#
```

Luego dentro del contenedor ejecutamos

apt-get update (para actualizar paquetes)

```
root@b6f7175afa84:/# apt-get update
Get:1 http://deb.debian.org/debian bookworm InRelease [151 kB]
Get:2 http://deb.debian.org/debian bookworm-updates InRelease [55.4 kB]
Get:3 http://deb.debian.org/debian-security bookworm-security InRelease [48.0 kB]
Get:4 http://deb.debian.org/debian bookworm/main amd64 Packages [8788 kB]
Get:5 http://deb.debian.org/debian bookworm-updates/main amd64 Packages [13.8 kB]
Get:6 http://deb.debian.org/debian-security bookworm-security/main amd64 Packages [169 kB]
Fetched 9225 kB in 2s (4755 kB/s)
Reading package lists... Done
root@b6f7175afa84:/#
```

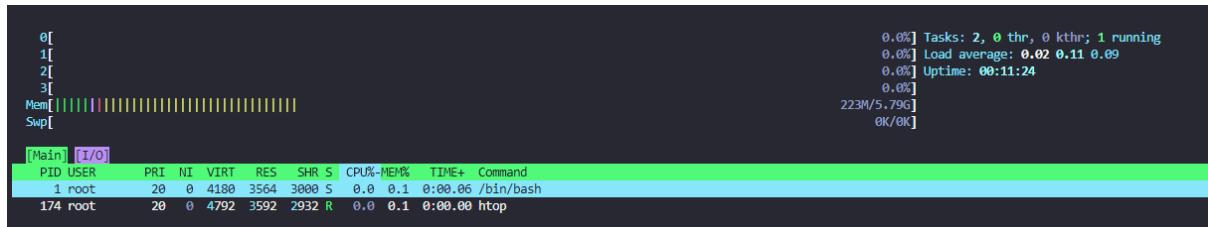
Luego instalamos 'htop' ejecutando **apt-get install -y htop**

```
Reading package lists... Done
root@b6f7175afa84:/# [200]apt-get install -y htop
bash: [200]apt-get: command not found
root@b6f7175afa84:/# apt-get install -y htop
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libgpm2 libncursesw6 libnl-3-200 libnl-genl-3-200
Suggested packages:
  lm-sensors lsof strace gpm
```

Verificamos con **htop --version**

```
root@b6f7175afa84:/# htop --version
htop 3.2.2
```

Por último ejecutamos **htop** para monitorear el sistema



primero busco el nombre id del contenedor a eliminar usando **docker ps -a**

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
b6f7175afa84	debian	"/bin/bash"	10 minutes ago	Exited (0) 3 minutes ago		xenodochial_yallow

Por último voy a eliminar el contenedor ejecutando

docker rm b6f7175afa84

b6f7175afa84	debian	"/bin/bash"	10 minutes ago	Exited (0) 3 minutes ago	xenodochial_yallow	
vagrant@ubuntu-focal:~\$ docker rm b6f7175afa84						
b6f7175afa84						
vagrant@ubuntu-focal:~\$ docker ps -a						
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES

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.

Primero vamos a crear el contenedor ejecutando el comando:

docker run -d -p 8080:80 --name nginx-server nginx

-d (ejecuta el contenedor en segundo plano)

-p 8080:80 (mapea el puerto 80 del contenedor al puerto 8080 de la máquina host)

--name nginx-server (nombre del contenedor)

nginx (imagen a usar)

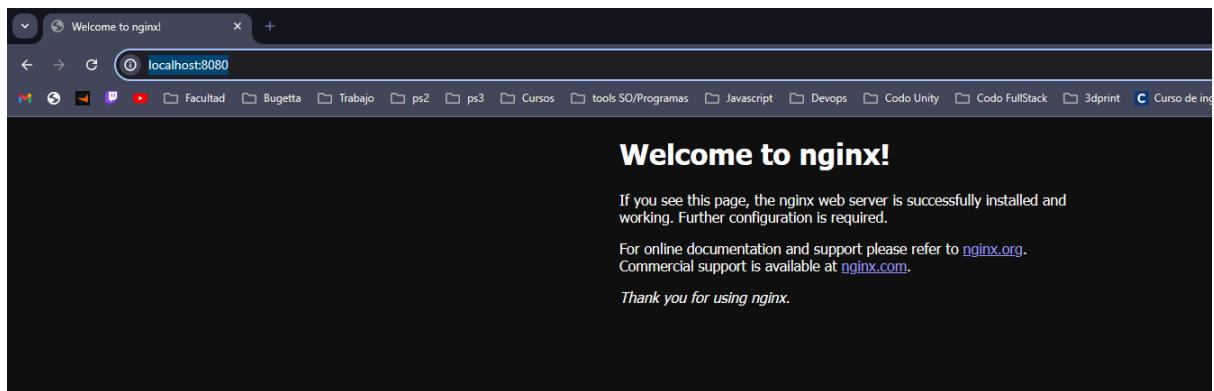
vagrant@ubuntu-focal:~\$ docker run -d -p 8080:80 --name nginx-server nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
efc2b5ad9eec: Pull complete
8fe9a55eb80f: Pull complete
045037a63be8: Pull complete
7111b42b4bfa: Pull complete
3dfc528a4df9: Pull complete
9e891cdb453b: Pull complete
0f11e17345c5: Pull complete
Digest: sha256:6af79ae5de407283dcea8b00d5c37ace95441fd58a8b1d2aa1ed93f5511bb18c
Status: Downloaded newer image for nginx:latest
a305c36954b43aaaf6aa9ddb831818b97cd6b130be946a0245aba90c597e01ae6

En otra terminal ejecutamos

vagrant ssh -- -L 8080:localhost:8080 (genera el túnel entre la máquina host y la vm)

Y por último accedemos mediante el navegador utilizando

<http://localhost:8080/>

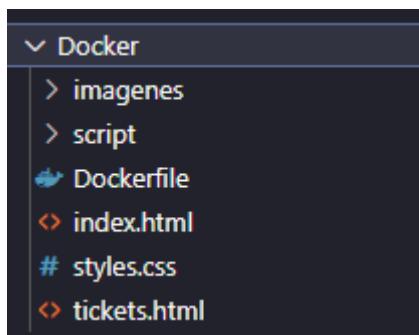


Crear una imagen personalizada:

- 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.

Utilice una web creada previamente por mi (en el curso codo a codo fullstack de java)

Creamos un directorio llamado Docker y luego colocamos los archivos 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 Docker al directorio de documentos de apache2)
- **'EXPOSE 80'** (configuramos donde se va a exponer el contenedor, en este caso puerto 80)

```
1 # Utiliza una imagen base de Apache2
2 FROM httpd:2.4.58
3
4 # Copia solo el contenido de la carpeta "Docker" al directorio de documentos de Apache2 en el contenedor
5 COPY ./ /usr/local/apache2/htdocs
6
7 # Expone el puerto 80 en el que Apache2 escuchará
8 EXPOSE 80
```

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

Para tener los archivos dentro de la maquina de vagrant, realice esta modificacion en el **bootstrap.sh**

```
#Clone repo
git clone https://github.com/fcongedo/Challenge-DEVOPS
```

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

Luego nos situamos en el directorio Docker

```
vagrant@ubuntu-focal:~/Challenge-DEVOPS$ cd /home/vagrant/Challenge-DEVOPS/Docker
```

Ejecutamos la siguiente instrucción para crear la imagen
docker build -t website-apache:1.0 .

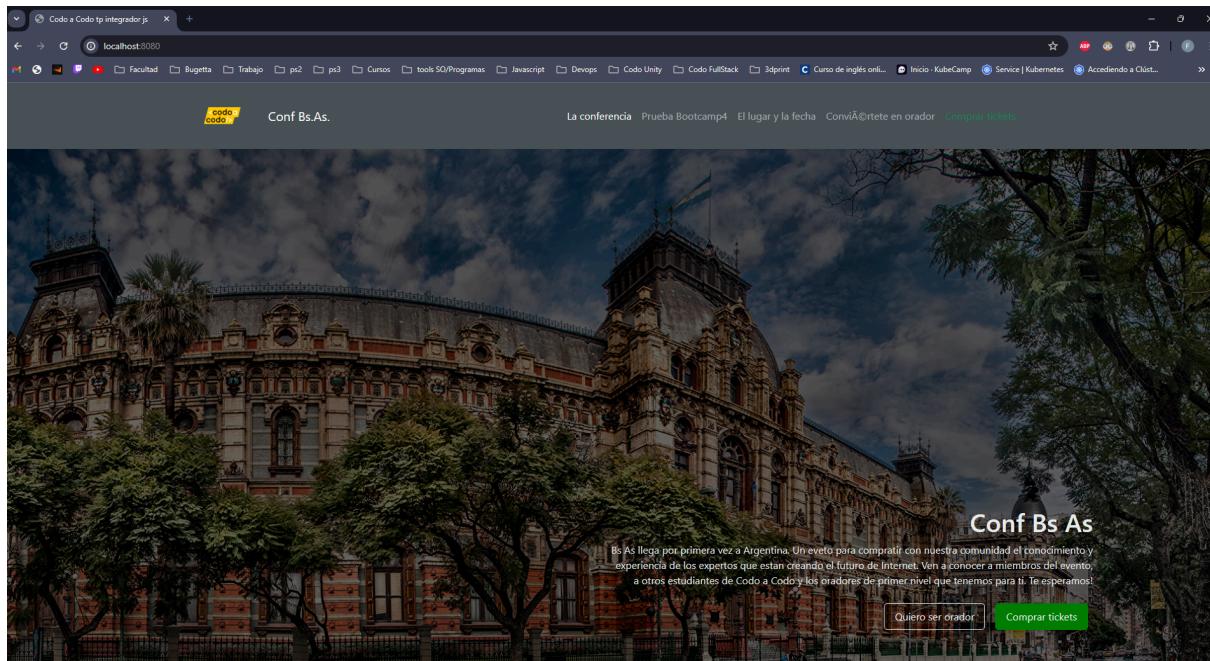
```
vagrant@ubuntu-focal:~/Challenge-DEVOPS/Docker$ docker build -t website-apache:1.0 .
[+] Building 0.6s (7/7) FINISHED
          docker:default
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 331B
=> [internal] load metadata for docker.io/library/httpd:2.4.58
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load build context
=> => transferring context: 450B
=> [1/2] FROM docker.io/library/httpd:2.4.58@sha256:374766f5bc5977c9b72fdb8ae3
=> CACHED [2/2] COPY ./ /usr/local/apache2/htdocs
=> exporting to image
=> => exporting layers
=> => writing image sha256:22734331c538f21dbf7abaad4f19efe50939a3d1a315249b7f2
=> => naming to docker.io/library/website-apache:1.0
vagrant@ubuntu-focal:~/Challenge-DEVOPS/Docker$ docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
website-apache  1.0      22734331c538  2 minutes ago  170MB
debian          latest   2e5b8d3ef33e  2 days ago   117MB
nginx           latest   a72860cb95fd  4 weeks ago   188MB
```

Por último corremos la imagen ejecutando:

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

Y por último accedemos mediante el navegador utilizando
<http://localhost:8080/>



● Jenkins

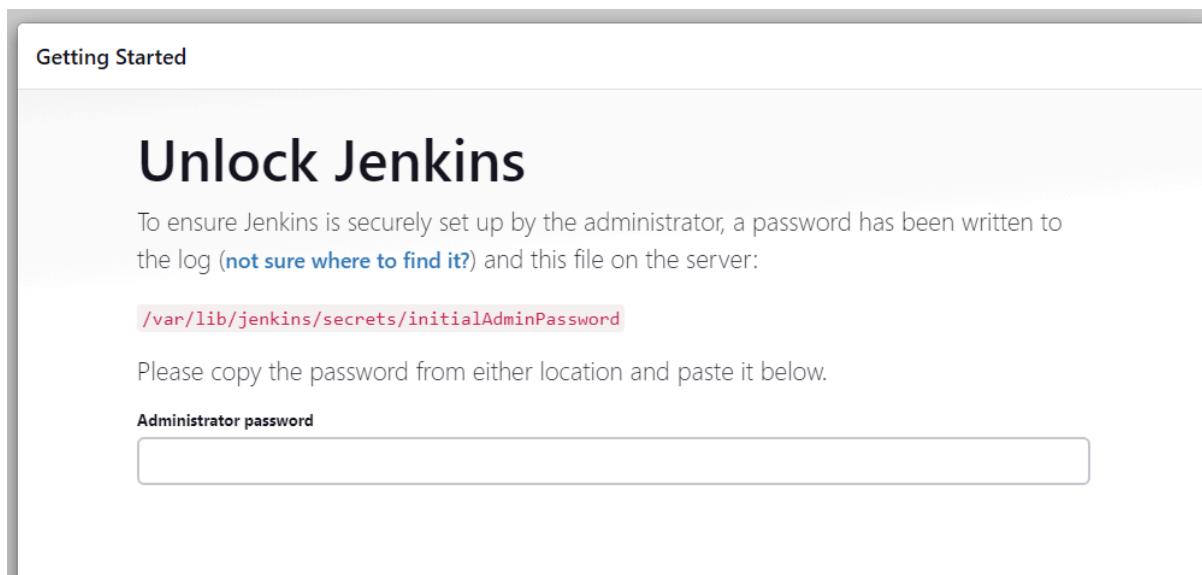
El objetivo de este ejercicio es crear un flujo de trabajo de CI/CD en Jenkins para implementar una aplicación web sencilla utilizando Docker. El flujo de trabajo automatizará las siguientes tareas:

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

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



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

The screenshot shows the Jenkins 'Getting Started' page. At the top, it says 'Getting Started'. Below that is a large 'Getting Started' heading with a progress bar underneath. A table follows, listing various Jenkins features and their status:

✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	✓ Credentials Binding	** 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
⌚ Timestampper	⌚ 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	

Luego creamos el usuario

The screenshot shows the 'Create First Admin User' configuration page. It has several input fields:

- Usuario: fcongado
- Contraseña: (redacted)
- Confirma la contraseña: (redacted)
- Nombre completo: Franco Congedo

At the bottom, it says 'Jenkins 2.414.3' and has two buttons: 'Skip and continue as admin' and '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.

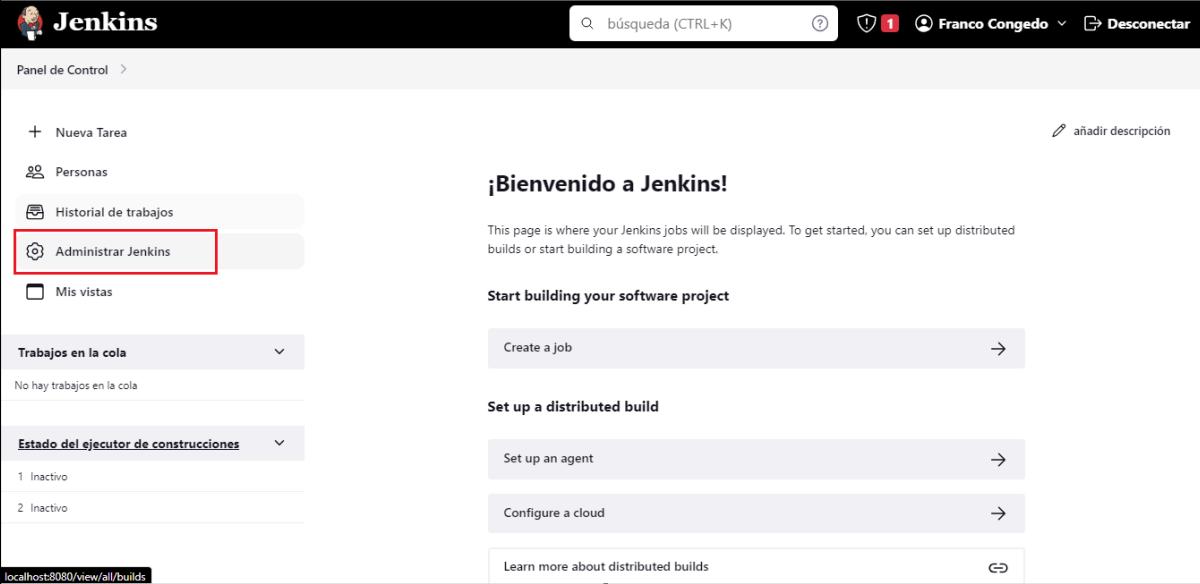
Getting Started

Jenkins is ready!

Your Jenkins setup is complete.

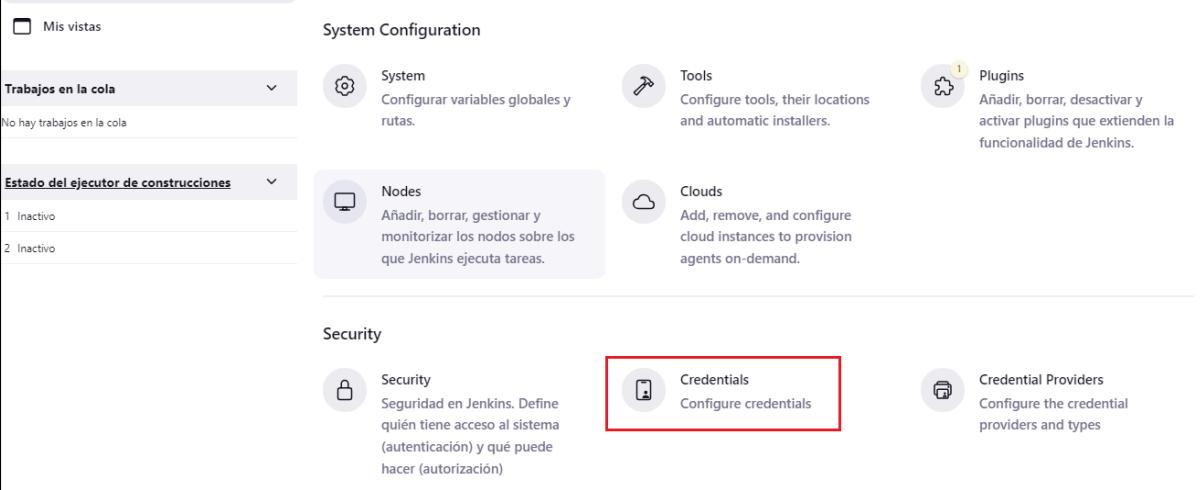
[Start using Jenkins](#)

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



The screenshot shows the Jenkins dashboard with a navigation bar at the top. The 'Administrador Jenkins' link is highlighted with a red box. The main content area displays a welcome message: '¡Bienvenido a Jenkins!' and instructions: 'This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.' Below this, there are sections for creating a job, setting up a distributed build, and managing agents.

Luego a la opción **credentials**



The screenshot shows the Jenkins System Configuration page. The 'Credentials' link under the 'Security' section is highlighted with a red box. Other sections visible include 'System', 'Tools', 'Nodes', 'Clouds', and 'Plugins'.

Hacemos click en **System**

The screenshot shows the Jenkins 'Credentials' page under 'Stores scoped to Jenkins'. A table lists a single entry: 'System' (global). The 'System' row is highlighted with a red box. Below the table are size icons: S, M, L.

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

The screenshot shows the Jenkins 'System' page with a table for global credentials. One entry, 'Global credentials (unrestricted)', is highlighted with a red box. A blue button '+ Add domain' is visible at the top right. Below the table are size icons: S, M, L.

Luego en **add credentials**

The screenshot shows the 'Global credentials (unrestricted)' page. A blue button '+ Add Credentials' is highlighted with a red box. A message at the bottom says: 'This credential domain is empty. How about [adding some credentials?](#)' Below the message are size icons: S, M, L.

Por último configuramos las credenciales ingresando los datos.

Clickeamos 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	Username with password	credenciales para logear en dockerhub

Luego primero vamos a agregar **jenkins** al grupo **docker** (permitiéndole ejecutar comandos de Docker sin necesidad de privilegios de administrador).

Para esto ejecutamos el comando **sudo usermod -aG docker jenkins**

Luego ejecutamos **newgrp docker** (Cambia temporalmente la sesión actual al grupo docker para ejecutar comandos en ese contexto durante la sesión actual, sin necesidad de cerrar y volver a abrir sesión.)

```
vagrant@ubuntu-focal:~$ sudo usermod -aG docker jenkins
vagrant@ubuntu-focal:~$ newgrp docker
```

Importante reiniciar el proceso de jenkins utilizando el comando:
sudo systemctl restart jenkins

Voy a reutilizar la web alojada en la carpeta Docker
Creamos nuestro **Jenkinsfile**.

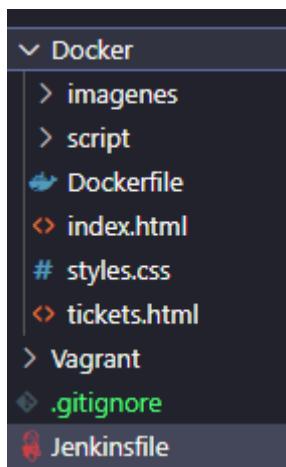
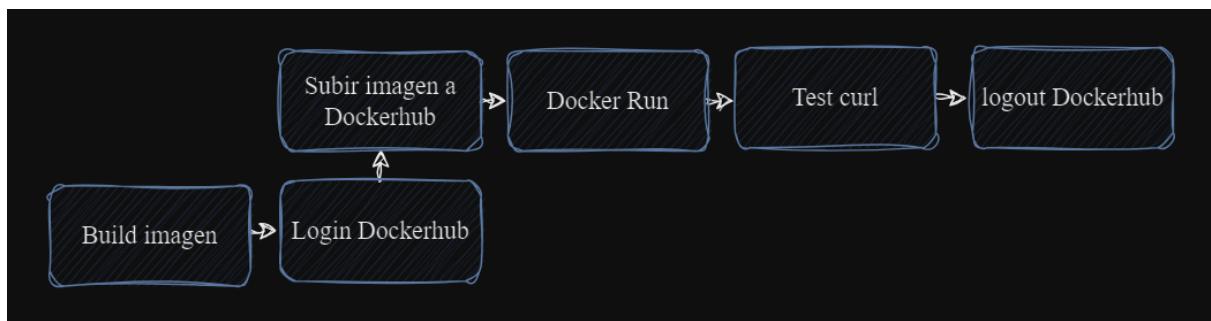


Diagrama CI/CD Docker



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"
        }
    }
}
```

Pusheamos 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)

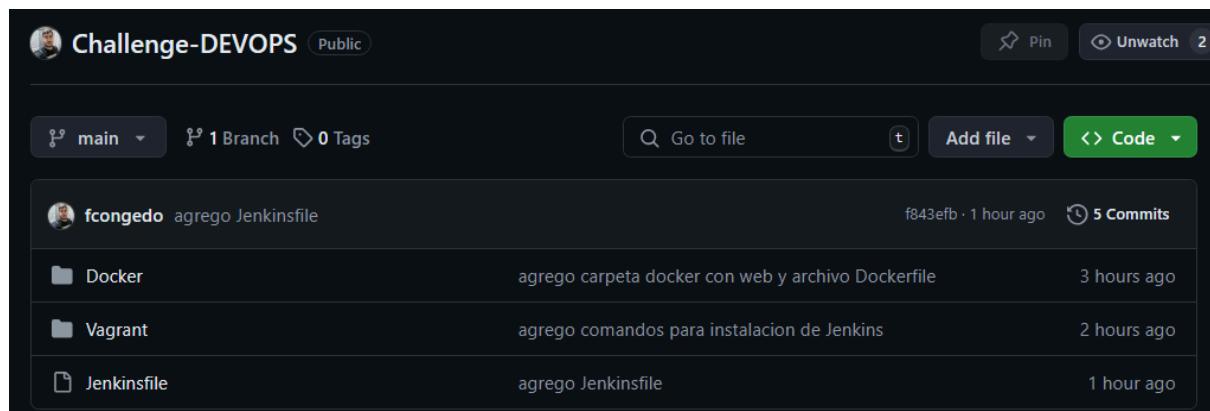
```
stage('Test') {
    steps {
        script {
            sleep(time: 5, unit: 'SECONDS')
            def curlOutput = sh(script: "curl -s -o /dev/null -w '%{http_code}" http://localhost:8082/", returnStdout: true).trim()
            echo "curlOutput: ${curlOutput}"

            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

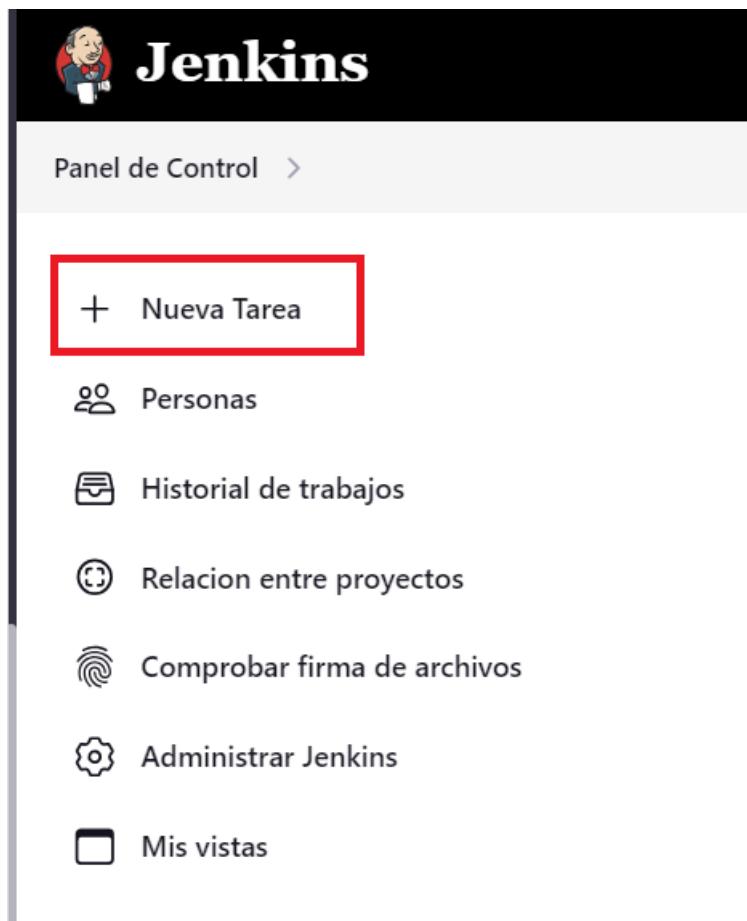
```
stage('Docker logout') {
    steps {
        sh "docker logout"
    }
}
```

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



Ahora vamos a crear nuestro **pipeline multibranch** en **Jenkins**

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



Agregamos el nombre de nuestro pipeline (pipeline-website)

Y seleccionamos la opción **Multibranch Pipeline**

Luego hacemos click en Ok

Enter an item name

pipeline-website
» Required field

 **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.

Luego asignamos una descripción y en **add source** (seleccionamos github)

General Enabled

Display Name

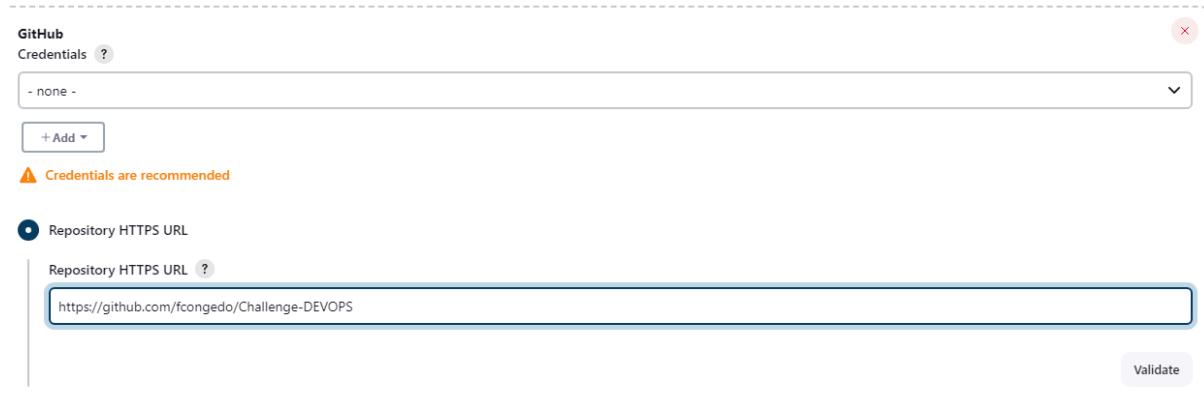
Description
CI/CD website Dockerhub

Plain text [Visualizar](#)

Branch Sources

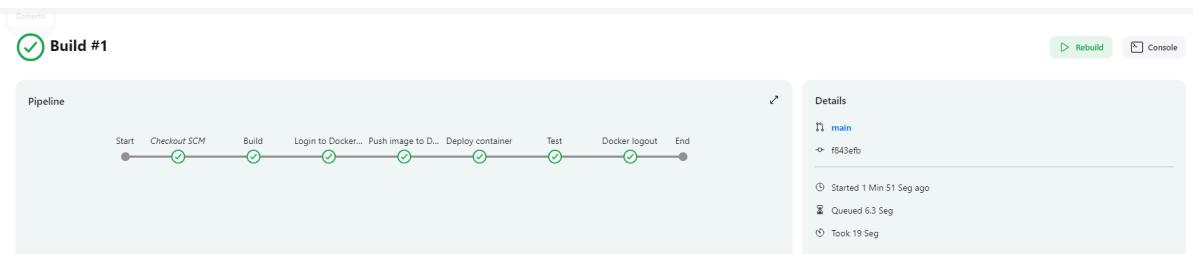
 GitHub
Credentials (none) + Add ▼
⚠ Credentials are recommended

Luego agregamos nuestro link de repositorio



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

El proceso se va a correr automáticamente y vemos si funciona o no

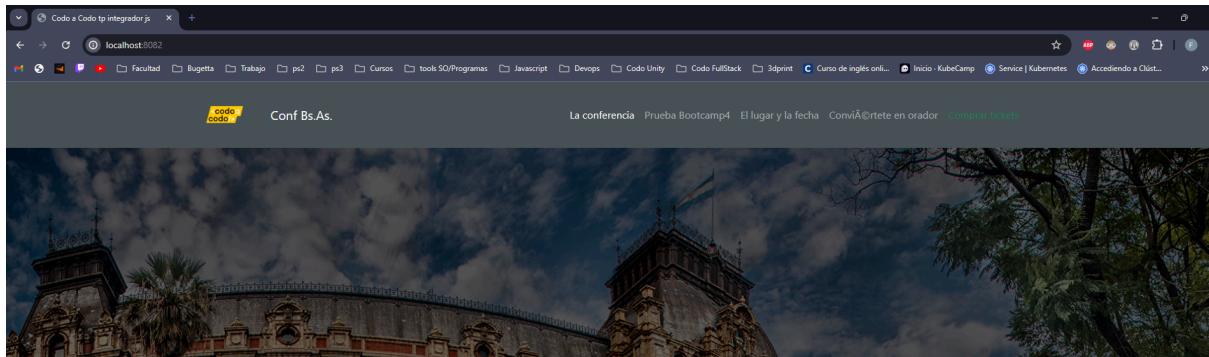


Verificamos en la máquina con vagrant
Ejecutando un docker images y docker ps

```
vagrant@ubuntu-focal:~$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
86c0df3af6a9 fccongedo/website-desafio:1 "httpd-foreground" 2 minutes ago Up 2 minutes 0.0.0.0:8082->80/tcp, :::8082->80/tcp website-desafio
vagrant@ubuntu-focal:~$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
fccongedo/website-desafio 1 22734331c538 3 hours ago 179MB
```

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-F3w7mX05PdgyTmZZMEAngseQB83DfGTowi0IMj1laeVhAn4FJkqJByhZMI3AhU" crossorigin="anonymous">
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/js/bootstrap.bundle.min.js"
    integrity="sha384-/bQdsTh/da6pkI1MST/rwKFNjaCP5gBSY4sEBT38Q/9RBh9AH40zE0g7Hlq2THRZ"
    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.

Tag	OS	Type	Pulled	Pushed
1		Image	an hour ago	7 minutes ago

- ***LINKS***

- [Repositorio de código Github](#)
- [Imagen Docker Hub - Website con CI/CD Jenkins](#)