

PASO A PASO TAREA 1 DOCKER

Este es un documento guía paso a paso para lograr las 3 partes de la tarea Docker, creado por nuestro grupo conformado por Benjamin Lobos y Diego Gonzalez

Tarea 1 y paso a paso realizado por Benjamin Lobos

Consideraciones:

- Descargar previamente DEVASC.ova desde cursos oficiales de netacad
- Descargar e instalar previamente Virtualbox
- Configurar adaptador de red según su adaptador activo con acceso a internet

Parte 1:

Crear una aplicación flask debe mostrar la ip desde donde está conectado y debe correr en el puerto 8000

Entramos al terminal como super usuario y escribimos:

```
# sudo su
```

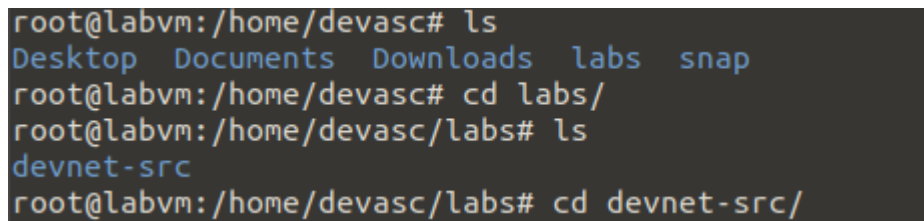
```
# pip3 install flask
```

A terminal window screenshot showing the command 'pip3 install flask' being executed. The prompt is 'root@labvm:~#'. Above the command, there is a message: '0/3 packages can be upgraded. Run 'apt list --upgradable' to see them.' Below the command, it says 'collecting flask'.

Luego dentro de /home/devasc

```
# cd labs/
```

```
# cd devnet-src/
```

A terminal window screenshot showing directory navigation. The prompt is 'root@labvm:/home/devasc#'. The user enters 'ls', and the output is 'Desktop Documents Downloads labs snap'. Then the user enters 'cd labs/', and the prompt changes to 'root@labvm:/home/devasc/labs#'. Then the user enters 'ls', and the output is 'devnet-src'. Finally, the user enters 'cd devnet-src/'.

Dentro de devnet-src/

cd sample-app/

nano sample_app.py

```
root@labvm:/home/devasc/labs/devnet-src# ls
ansible      jenkins      parsing      python       school-library  unittest
coding-basics  mapquest     ptna        sample-app   security        webex-teams
root@labvm:/home/devasc/labs/devnet-src# cd sample-app/
root@labvm:/home/devasc/labs/devnet-src/sample-app# ls
sample_app.py  sample-app.sh  static  templates
root@labvm:/home/devasc/labs/devnet-src/sample-app# nano sample_app.py
```

Y editamos el archivo sample_app.py

```
from flask import Flask
from flask import request

sample = Flask(__name__)
@sample.route("/")
def main():
    return "Tu dirección Ip es : " + request.remote_addr + "\n"
if __name__ == "__main__":
    sample.run(host="0.0.0.0", port=8000)
```

Este código crea una pequeña aplicación web usando Flask instalado previamente y cuando accedes a su página principal (localhost:8000) o (ip:8000), te muestra tu dirección IP.

Dentro de nano

ctrl + o // + enter para guardar el archivo

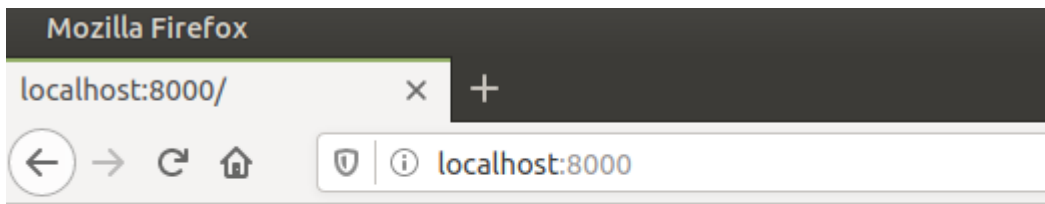
ctrl + x // exit

Para correr el código

python3 sample_app.py

```
root@labvm:/home/devasc/labs/devnet-src/sample-app# python3 sample_app.py
* Serving Flask app 'sample_app'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment
Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:8000
* Running on http://192.168.1.82:8000
Press CTRL+C to quit
```

E ingresamos a localhost:8000 en su navegador de confianza



Tu dirección Ip es : 127.0.0.1

Con esto concluimos la parte 1, completándola a la perfección.

// Comandos de vista

netstat

netstat -tnlp

netstat -tnap

Parte 2:

Crear una aplicación Python, y usando plantilla html y css, que levante un sitio web que verifique la ip de conexión y que corra en el puerto 8181.

Con lo previamente configurado simplemente vamos a la ruta:

```
# cd /home/devasc/labs/devnet-src/sample-app/
```

```
root@labvm: /home/devasc/labs/devnet-src/sample-app# cd /home/devasc/labs/devnet-src/sample-app/
```

Y reeditamos el archivo sample_app.py

```
# nano sample_app.py
```

Y pegamos este texto

```
from flask import Flask
from flask import request
from flask import render_template
sample = Flask(__name__)

@sample.route("/")
def main():
    return render_template("index.html")

if __name__ == "__main__":
    sample.run(host="0.0.0.0", port=8181)
```

Este código crea una aplicación web Flask que sirve el contenido de un archivo HTML llamado index.html cuando se accede a la ruta (localhost:8181) o (ip:8181)

Dentro de nano

```
# ctrl + o // + enter para guardar el archivo
```

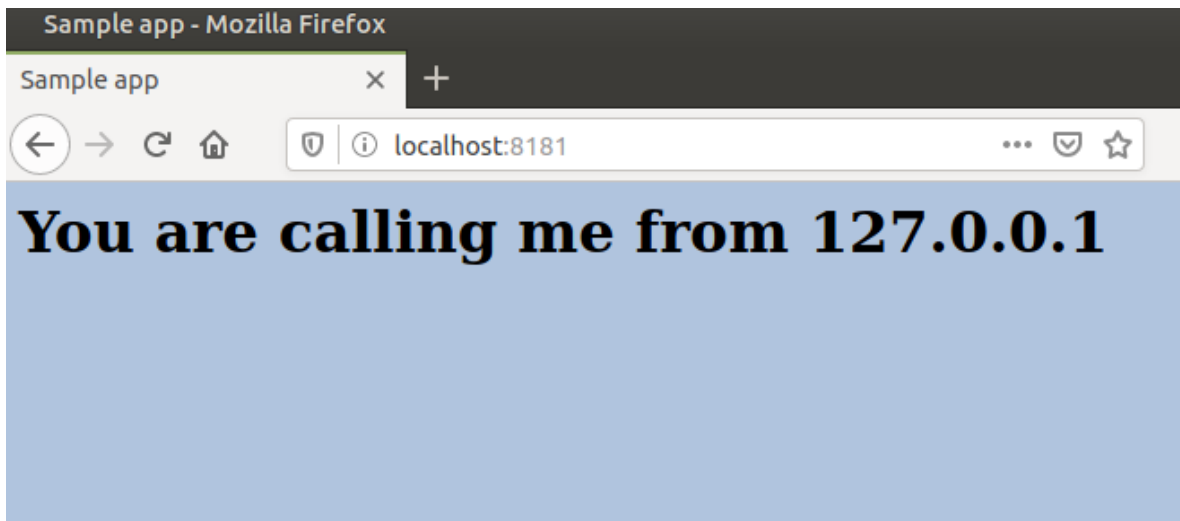
```
# ctrl + x // exit
```

Para correr el código lo mismo:

```
# python3 sample_app.py
```

```
root@labvm: /home/devasc/labs/devnet-src/sample-app# python3 sample_app.py
* Serving Flask app 'sample_app'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment
Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:8181
* Running on http://192.168.1.82:8181
Press CTRL+C to quit
```

E ingresamos a localhost:8181 en su navegador de confianza



Con esto concluimos la parte 2, completándola a la perfección.

// Comandos de vista

netstat

netstat -tnlp

netstat -tnap

Parte 3:

Crear un script en bash que cree y corra un contenedor Docker a partir de un archivo Dockerfile, que levante un sitio web y que exponga el puerto 8888.

Primero modificamos nuevamente el archivo sample_app.py para mostrar el puerto 8888

```
root@labvm: /home/devasc/labs/devnet-src/sample-app# cd /home/devasc/labs/devnet-src/sample-app/
```

```
# cd /home/devasc/labs/devnet-src/sample-app/
```

```
# nano sample_app.py
```

Y pegamos

```
# sample_app.py
from flask import Flask, render_template, request

sample = Flask(__name__)

@sample.route("/")
def main():
    return render_template("index.html", remote_ip=request.remote_addr)

if __name__ == "__main__":
    sample.run(host="0.0.0.0", port=8888, debug=False, threaded=False,
processes=1)
```

Dentro de nano

```
# ctrl + o // + enter para guardar el archivo
```

```
# ctrl + x // exit
```

Creamos un bash modificando el archivo sample-app.sh

```
root@labvm:/home/devasc/labs/devnet-src/sample-app# cd /home/devasc/labs/devnet-src/sample-app/
```

```
# cd /home/devasc/labs/devnet-src/sample-app/
```

```
# nano sample-app.sh
```

Y pegamos:

```
#!/bin/bash

rm -rf tmpdir

mkdir tmpdir
mkdir tmpdir/templates
mkdir tmpdir/static
cp sample_app.py tmpdir/
cp -r templates/* tmpdir/templates/
cp -r static/* tmpdir/static/

echo "FROM python:3.11-slim" > tmpdir/Dockerfile
echo "RUN pip install --no-cache-dir --progress-bar off flask" >> tmpdir/Dockerfile
echo "COPY ./static /home/myapp/static/" >> tmpdir/Dockerfile
echo "COPY ./templates /home/myapp/templates/" >> tmpdir/Dockerfile
echo "COPY sample_app.py /home/myapp/" >> tmpdir/Dockerfile
echo "EXPOSE 8888" >> tmpdir/Dockerfile
echo "CMD python3 /home/myapp/sample_app.py" >> tmpdir/Dockerfile
```

Dentro de nano

```
# ctrl + o // + enter para guardar el archivo
```

```
# ctrl + x // exit
```

Y ejecutamos

```
# chmod +x sample-app.sh
```

```
# ./sample_app.sh
```

Con esto se crea y edita el Dockerfile y se mueven las carpetas /templates y /static hacia la nueva carpeta tmpdir además del archivo sample_app.py previamente modificado

Editamos archivo style.css dentro de tmpdir/static/

```
# cd tmpdir/
```

```
# cd static/
```

```
# nano style.css
```

```
root@labvm:/home/devasc/labs/devnet-src/sample-app# cd tmpdir/  
root@labvm:/home/devasc/labs/devnet-src/sample-app/tmpdir# cd static/  
root@labvm:/home/devasc/labs/devnet-src/sample-app/tmpdir/static# nano style.css
```

Y pegamos el texto:

```
body {  
  font-family: Arial, sans-  
serif;  
  display: flex;  
  justify-content: center;  
  align-items: center;  
  min-height: 100vh;  
  margin: 0;  
  background-color: #f0f0f0;  
}  
  
p {  
  font-size: 2em;  
  font-weight: bold;  
  color: #333;  
  text-align: center;  
}
```

Dentro de nano

```
# ctrl + o // + enter para guardar el archivo
```

```
# ctrl + x // exit
```

Ahora editamos el archivo index.html dentro de tmpdir/templates/

```
# cd tmpdir/
```

```
# cd templates/
```

```
# nano index.html
```

```
root@labvm:/home/devasc/labs/devnet-src/sample-app# cd tmpdir/  
root@labvm:/home/devasc/labs/devnet-src/sample-app/tmpdir# cd templates/  
root@labvm:/home/devasc/labs/devnet-src/sample-app/tmpdir/templates# nano index.html
```


Y pegamos el texto:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Conexión IP</title>
  <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
  <p>La conexión se esta realizando desde {{ remote_ip }}</p>
</body>
</html>
```

ctrl + o // + enter para guardar el archivo

ctrl + x // exit

Estupendo ya casi terminamos; ahora creamos la imagen Docker con nombre personalizado, dentro de tempdir:

docker build -t benjaydiego .

```
root@labvm:/home/devasc/labs/devnet-src/sample-app/tempdir# docker build -t benjaydiego .
Sending build context to Docker daemon  6.144kB
```

Empezara a descargar dividido en 7 pasos, usted simplemente espere que se complete todo.

```
Step 7/7 : CMD python3 /home/myapp/sample_app.py
--> Running in 14010d24f413
Removing intermediate container 14010d24f413
--> f741d505cb72
Successfully built f741d505cb72
Successfully tagged benjaydiego:latest
root@labvm:/home/devasc/labs/devnet-src/sample-app/tempdir#
```

Una vez todo concluido el siguiente paso es ejecutar el contenedor
docker run -d -p 8888:8888 --name flask-container benjaydiego

```
root@labvm:/home/devasc/labs/devnet-src/sample-app/tempdir# docker run -d -p 8888:8888 --name flask-container benjaydiego
215f93e2fdcfbae526999d6d124206b926e9cfa3c2f827386c9195d2e283dda5
root@labvm:/home/devasc/labs/devnet-src/sample-app/tempdir# docker logs flask-container
* Serving Flask app 'sample_app'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:8888
* Running on http://172.17.0.2:8888
Press CTRL+C to quit
root@labvm:/home/devasc/labs/devnet-src/sample-app/tempdir#
```

Y entrando a (172.17.0.1:8888) o (http://localhost:8888/)



Con esto concluimos la parte 3, completándola a la perfección.

// Comandos de vista

netstat

netstat -tnlp

netstat -tnap

PASO A PASO TAREA 2 RESPALDO DRIVE AUTOMATIZADO

Tarea 2 y paso a paso realizado por Diego Gonzalez

Consideraciones:

- Descargar previamente ubuntu server
- Descargar e instalar previamente Virtualbox
- Configurar adaptador de red según su adaptador activo con acceso a internet

1- Instalar rclone en una máquina virtual con los siguientes comandos:

- `sudo apt update`
- `sudo apt install rclone -y`

2- Configurar rclone con drive

n) New remote

name> drive

3- tener en cuenta que acá va el usuario creado en Google cloud dándole acceso

Google Application Client Id

client_id>

Google Application Client Secret

client_secret>

4-luego elegir configuracion manual por que estamos en un servidor virtual

Use auto config? (y/n)> respuesta =n

5- autenticar en el navegador

Acá se va generar un link con el cual accedemos desde nuestro navegador y posteriormente iniciaremos sesión con nuestra cuenta de Google , después de verificar la cuenta , usaremos el código que nos da Google y lo pondremos en rclone

6-Creacion de scrip de respaldo

nano /home/diegoaether/respaldo.sh

pegaremos dentro esto

```
#!/bin/bash
```

```
# Ruta y nombre del respaldo
```

```
FECHA=$(date +"%Y-%m-%d_%H-%M")
```

```
ARCHIVO="/home/diegoaether/respaldo_${FECHA}.tar.gz"
```

```
# Crear respaldo de /etc (puedes cambiar esta ruta)
```

```
tar -czf "$ARCHIVO" /etc
```

```
# Subir a Google Drive
```

```
/usr/bin/rclone copy "$ARCHIVO" miDrive:respaldos
```

```
# Guardar log
```

```
echo "$(date) - Respaldo generado y subido: $ARCHIVO" >>  
/home/diegoaether/respaldo.log
```

7- hacer el ejecutable con el siguiente comando

```
chmod +x /home/diegoaether/respaldo.sh
```

8- opciones para probar el script

```
/home/diegoaether/respaldo.sh
```

Verificación con

```
rclone ls miDrive:respaldos
```

9-programar ejecución automática

```
crontab -e
```

agregar en la ultima línea

```
30 8 * * * /home/diegoaether/respaldo.sh // esto va cambiando dependiendo de la hora que se quiera hacer el respaldo
```

10- verificar respaldo

```
rclone ls miDrive:respaldos
```

visualizar logs del scrip

```
cat /home/diegoaether/respaldo.log
```

Con esto concluimos la tarea 2, completándola a la perfección.