

Manual Despliegue Modelo Heart Attack (Empaquetamiento y API)

Introducción

Este documento detalla los pasos necesarios para empaquetar el modelo de Heart Attack desarrollado en Python. Se cubre desde la configuración de un ambiente virtual en una máquina virtual hasta la creación, prueba y despliegue del paquete, asegurando que el proceso sea escalable y replicable.

Parte 1: Preparación del Ambiente

1. Lanzar una máquina virtual en AWS EC2:
2. Entra a la consola de AWS y selecciona la opción de instancias EC2.
3. Crea una instancia con las siguientes configuraciones:
4. Tipo de instancia: t2.small
5. Sistema operativo: Ubuntu Server
6. Almacenamiento: al menos 20 GB
7. Genera y descarga un archivo .pem para la clave de acceso. Este archivo será necesario para conectar tu máquina local a la instancia.

8. Subir archivos necesarios. Usa el siguiente comando para transferir el archivo package-src.zip a la máquina virtual:

```
scp -i "ruta_a_llave.pem" "ruta_a_package-src.zip" ubuntu@IP:/home/ubuntu
```

9. Conexión a la máquina virtual. Conéctate usando el siguiente comando:

```
ssh -i "ruta_a_llave.pem" ubuntu@IP
```

10. Actualizar paquetes e instalar herramientas. Ejecuta los siguientes comandos para actualizar el sistema operativo e instalar herramientas necesarias:

```
sudo apt update
```

```
sudo apt install python3-pip zip unzip python3.12-venv
```

11. Crear y activar ambiente virtual. Crea y activa un ambiente virtual llamado env-tox:

```
python3 -m venv /home/ubuntu/env-tox
```

```
source /home/ubuntu/env-tox/bin/activate
```

12. Descomprimir el paquete. Usa el siguiente comando para descomprimir el paquete:

```
unzip package-src.zip
```

13. Instalar herramientas necesarias. Instala tox y agrega su ruta al PATH:

```
pip install tox
sudo apt-get install tox
export PATH=$PATH:/home/ubuntu/.local/bin
```

Parte 2: Ejecución del Empaquetamiento

14. Entrar a la carpeta del paquete.

```
cd package-src
```

15. Ejecutar pruebas de entrenamiento. puede generar un warning de unas variables que no se usan pero se puede ignorar:

```
tox run -e train
```

16. Ejecutar pruebas del paquete:

```
tox run -e test_package
```

17. Construir el paquete:

```
python3 -m pip install --upgrade build
python3 -m build
```

18. Transferir el archivo generado:

```
mkdir /home/ubuntu/test
cp dist/model_heart-0.0.1-py3-none-any.whl /home/ubuntu/test
```

19. Instalar el paquete:

```
pip install model_heart-0.0.1-py3-none-any.whl
```

Parte 3: Pruebas con Datos Test

20. Subir archivos de prueba:

```
scp -i "ruta_a_llave.pem" "ruta_a_archivo" ubuntu@IP:/home/ubuntu/test
```

21. Ejecutar pruebas:

```
python3 test-package.py
```

22. Verificar paquetes instalados:

```
pip freeze
```

Parte 4: API

```
#scp -i "C:\Users\DELL\Downloads\test_keys.pem" "C:\Users\DELL\OneDrive -  
Universidad de los andes\Escritorio\docker-api.zip"  
ubuntu@3.208.8.141:/home/ubuntu
```

```
ssh -i "C:\Users\DELL\Downloads\test_keys.pem" ubuntu@3.208.8.141
```

```
sudo apt-get remove docker docker-engine docker.io containerd runc
```

```
sudo apt-get update
```

```
sudo apt-get install ca-certificates curl gnupg
```

```
sudo install -m 0755 -d /etc/apt/keyrings
```

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o  
/etc/apt/keyrings/docker.gpg
```

```
echo \
```

```
"deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg]  
https://download.docker.com/linux/ubuntu \
```

```
"${. /etc/os-release} && echo "$VERSION_CODENAME)" stable" | \
```

```
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

```
sudo apt-get update
```

**sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin**

sudo docker run hello-world

sudo apt install unzip

unzip docker-api.zip

cd docker-api

sudo docker build -t heart-analytics:latest .

sudo docker images

sudo docker run -p 8001:8001 -it -e PORT=8001 heart-analytics

=====
===== Dash

**scp -i "C:\Users\DELL\Downloads\test_keys.pem" "C:\Users\DELL\OneDrive -
Universidad de los andes\Escritorio\docker-dash.zip"
ubuntu@3.89.38.247:/home/ubuntu**

ssh -i "C:\Users\DELL\Downloads\test_keys.pem" ubuntu@3.89.38.247

sudo apt-get remove docker docker-engine docker.io containerd runc

sudo apt-get update

sudo apt-get install ca-certificates curl gnupg

sudo install -m 0755 -d /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

sudo chmod a+r /etc/apt/keyrings/docker.gpg

**echo **

**"deb [arch="\$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu **

**"\${. /etc/os-release} && echo "\$VERSION_CODENAME)" stable" | **

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

**sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin**

sudo docker run hello-world

sudo apt install unzip

unzip docker-dash.zip

```
cd docker-dash
```

```
sudo docker build -t app:latest .
```

```
sudo docker run -p 8050:8050 -it -e PORT=8050 -e API_URL=3.208.8.141 app
```