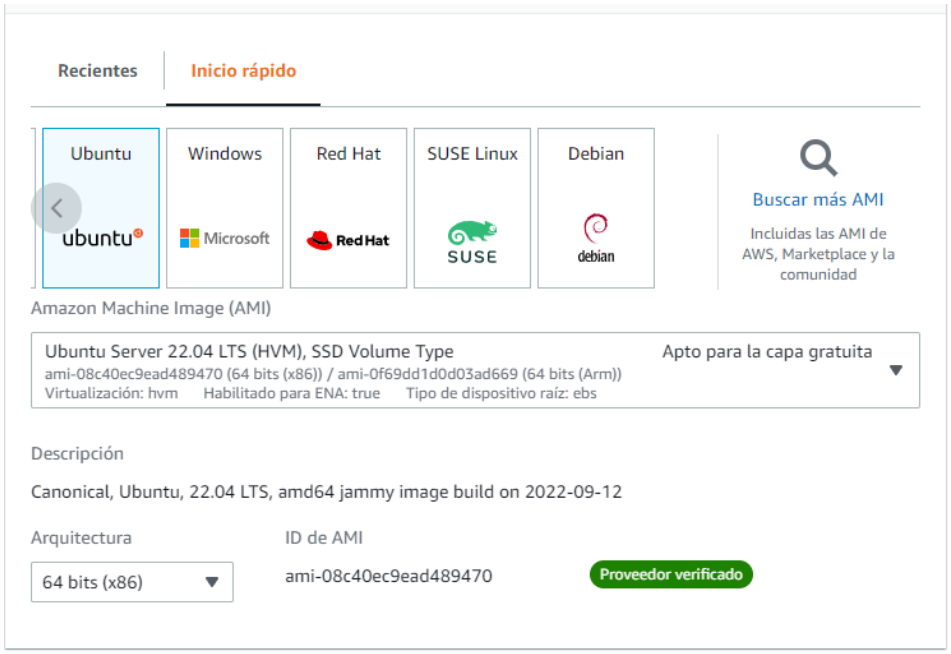
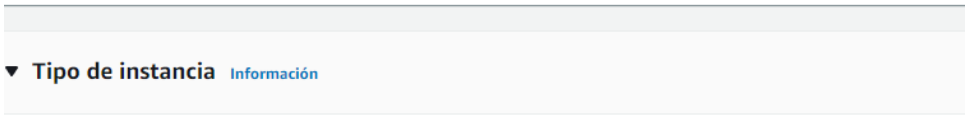
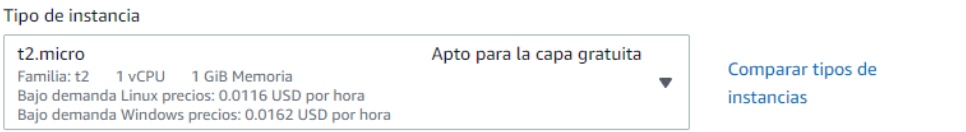
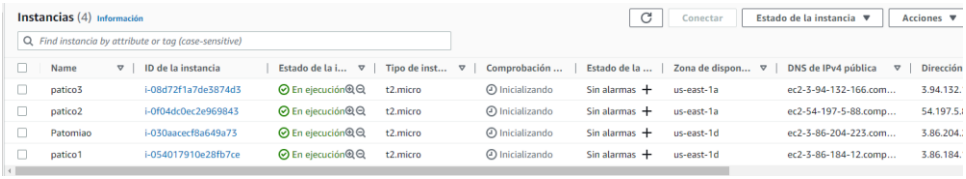
Documentación proyecto 1

1. Cree 4 instancias de aws y la regla para que escuchen por el puerto 8080









2. Una vez creadas las instancias y nombradas para diferenciarlas se procede a crear el codigo del proxy utilizando python
    a. En el habran 4 archivos primero constantes.py el cual esta encargado de mantener las constantes

constants.py

```python

privadas = open('/home/ubuntu/pibl/proxy/privadas.txt')
priv= privadas.read().split(",")
publicas = open('/home/ubuntu/pibl/proxy/publicas.txt')
publ= publicas.read().split(",")

IP_SERVER=str(priv[0])

IP_SERVERS=[publ[1],publ[2],publ[3]]


PORT = 8080
ENCONDING_FORMAT = "utf-8"
RECV_BUFFER_SIZE = 2048
QUIT = 'QUIT'
print(IP_SERVERS)
print(IP_SERVER)
```

b. Pb.py encargado de hacer el proxy y el balanceador de carga

pb.py

```python
import socket
import constants
import logging
import time
host , port =  constants.IP_SERVER, constants.PORT
serversocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
logging.basicConfig(filename = 'Register.log', format="%(asctime)s %(levelname)s %(message)s",  level = 'INFO')
def main():
    serversocket.setsockopt(socket.SOL_SOCKET , socket.SO_REUSEADDR , 1)
    serversocket.bind((host , port))
    serversocket.listen(1)
    print('servidor en el puerto',port)
    ip_index = 0
    while True:
        connection , address = serversocket.accept()
        request = connection.recv(1024).decode('utf-8')
        print(address)
        print(request)
        logging.info('Consulta: %r\n recibida en el puerto: %r\n',request, port)
        data_received = send(request, ip_index)
        print(data_received)
        response = data_received
        response = str(data_received.decode(constants.ENCONDING_FORMAT))
        connection.send(response.encode(constants.ENCONDING_FORMAT))
        print(response)
        logging.info('respuesta: %r\n enviada por el puerto: %r\n', response, port)
        if ip_index <= len(constants.IP_SERVERS)-1:
            ip_index+=1
        else:
            ip_index = 0
def send(command_to_send, ip_index):
    clientsocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    clientsocket.connect((constants.IP_SERVERS[0],constants.PORT))
    if command_to_send == '':
        print('Please input a valid command...')
    else:
        clientsocket.send(bytes(command_to_send,constants.ENCONDING_FORMAT))
        data_received = clientsocket.recv(constants.RECV_BUFFER_SIZE)
        return data_received
if __name__ == '__main__':
    constants
    main()
```

c. Y por ultimo dos archivos txt llamados privadas.txt y publicas.txt

Siendo el publicas el único que se modifica ya que las ips privadas nunca cambian

3. Procedemos a crear los archivos que irán en cada uno de los servidores
    a. Constantes.py para manejar las que siempre quedan concretas

```python
PORT = 8080
ENCONDING_FORMAT = "utf-8"
RECV_BUFFER_SIZE = 2048
privadas = open('/home/ubuntu/patico2/privadas.txt')
priv= privadas.read().split(",")

IP_SERVER=str(priv[1])
```

    b. Pato.py que será el server el cual correrá el framework en este caso un HTML

```python
import socket
import constantes
import os

host = constantes.IP_SERVER
port = constantes.PORT


def listen():
    serversocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    serversocket.setsockopt(socket.SOL_SOCKET , socket.SO_REUSEADDR , 1)
    serversocket.bind((host , port))
    serversocket.listen(1)
    print('servidor en el puerto',port)
    return serversocket


def temp(serversocket):

    while True:

        connection , address = serversocket.accept()
        request = connection.recv(1024).decode(constantes.ENCONDING_FORMAT)
        print("------x1"+request+"--------")
        string_list = request.split(' ')

        if request == "":
            temp(serversocket)

        requesting_file = string_list[1]

        print('Client request',requesting_file)

        myfile = requesting_file.split('?')[0]
        myfile = myfile.lstrip('/')

        if(myfile == ''):
            myfile = 'index.html'
        if(myfile == 'favicon.ico'):
            temp(serversocket)
        try:
            file = open(myfile , 'rb')
            response = file.read()
            file.close()

            header = 'HTTP/1.1 200 OK\n'

            if(myfile.endswith('.jpg')):
                mimetype = 'image/jpg'
            elif(myfile.endswith('.ico')):
                mimetype = 'image/x-ico'
            else:
                mimetype = 'text/html'

            header += 'Content-Type: '+str(mimetype)+'\n\n'

            final_response = header.encode(constantes.ENCONDING_FORMAT)
            final_response += response
            send(connection,final_response)

        except Exception as e:
            print("-")
            header = 'HTTP/1.1 404 Not Found\n\n'
            response = '<html><body>Error 404: File not found</body></html>'.encode(constantes.ENCONDING_FOR
            send(connection,response)


def send(connection,final_response):
    connection.send(final_response)
    connection.close()


if __name__ == '__main__':
    soc = listen()
    temp(soc)
```

    c. El index.html el cual sera el framework utilizado para la mostrar la pagina

index.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Servidor2</title>
</head>
<body>
    <center><h1>El servidor #2 esta funcionando</h1></center>
</body>
</html>
```

d. Y por último privadas.txt el cual se encargar de almacenar todas las ips privadas

privadas.txt

```
172.31.80.165,172.31.91.208,172.31.19.11,172.31.22.129
```