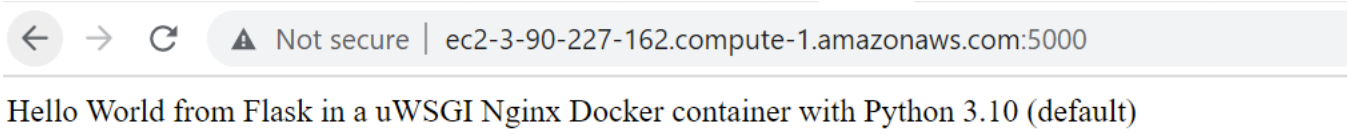
For this assignment, I have used AWS EC2 (Ubuntu 18.04, instance type- t2.micro).

Problem 1:

- I have installed docker and pulled up the image as mentioned in problem.
- I created a container in detached mode and port forwarded on 5000 using command-

  docker run -d -p 5000:5000 tiangolo/uwsgi-nginx-flask
- I was able to connect to this container from my local machine on port 5000(instance public dns/ip:5000)
- The output is as below-
  Hello World from Flask in a uWSGI Nginx Docker container with Python 3.10 (default)

  Screenshot for the same-



Problem 2:

Please find the shell scripts for both problems and their respective output as below-

A. Reverse a number-

root@ip-172-31-41-59:/home/ubuntu/scripts# cat reverse.sh

#read first argument and assign it to num variable

num=$1

sd=0

```
rev=0
while [ $num -gt 0 ]
do
   sd=$(( $num % 10 ))
   rev=$(( $rev * 10 + $sd ))
   num=$(( $num / 10 ))
done
echo "Reverse number of entered digit is $rev"
```

```
root@ip-172-31-41-59:/home/ubuntu/scripts# ./reverse.sh 123456
Reverse number of entered digit is 654321
```

B. Sum of Numbers-
   The script for second problem and the output is as below-

```
root@ip-172-31-41-59:/home/ubuntu/scripts# cat sum_of_number.sh

#/bin/bash

num=$1
sum=0

for((i=1;i<=$num;i++))
do
      sum=$(($sum +$i))
done
echo "The sum of numbers from 1 to $num is $sum"
```

root@ip-172-31-41-59:/home/ubuntu/scripts# ./sum_of_number.sh 10

The sum of numbers from 1 to 10 is 55.

Problem 3-

- For this problem, I have used flask with postgre sql(both are deployed in docker container)which are performing crud operations using rest apis.(I have used docker for this problem as I don't have flask/postgre db setup on my local)
- I am using docker-compose for spinning multi-container environment- one is python flask application and another is postgre db and linked frotend with backend db using docker environment variables.
- I am using 4 main files-
  1. A Dockerfile for python flask application
  2. A requirements.txt for all specifying all dependencies required for this application,
  3. A docker-compose.yml for specifying container details
  4. A file app.py which has view functions/crud apis for accessing
- Have tested all crud apis using postman, all were successfully able to connect to database to perform the required operations.
- I have created one new endpoint for executing weather.py script.
- This script is scraping the mentioned website and dumps its data in weather.csv file.
- I have written one view function for displaying weather(basically this is weather.py file in your repo), I was struggling with docker-compose and executing weather.py in docker container. It created weather.csv inside container, but *I am quite not sure what is the correct way to do this*(tried my best!).
- My docker-compose.yml file screenshot-

```
root@ip-172-31-41-59:/home/ubuntu/scripts/flask-postgresql# cat docker-compose.yml
version: '2.2'

services:
  pythonapp:
    container_name: pythonapp
    image: pythonapp
    build: .
    ports:
      - "8090:8090"
    environment:
      - DATABASE_URL=postgresql://postgres:postgres@db:5432/myguestbook
    depends_on:
      - db

  db:
    container_name: db
    image: postgres:12
    ports:
      - "5432:5432"
    environment:
      - POSTGRES_PASSWORD=postgres
      - POSTGRES_USER=postgres
      - POSTGRES_DB=myguestbook
    volumes:
      - pgdata:/var/lib/postgresql/data


volumes:
  pgdata: {}
```

The file app.py script-

```
from flask import Flask, request, jsonify
from flask_sqlalchemy import SQLAlchemy
import os
import requests
import bs4
from bs4 import BeautifulSoup
import csv

app = Flask(__name__)

if __name__ == '__main__':
    app.run(debug=True)

app.config['SQLALCHEMY_DATABASE_URI'] = os.environ.get('DATABASE_URL')
db = SQLAlchemy(app)

#postgredb name is- myguestbook and table name is - guest
#defining model which is our postgredb table with attributes name,
address.Here, id is an autoincremented primary key field.
class Guest(db.Model):
    id = db.Column(db.Integer, primary_key=True)
```

```
    name = db.Column(db.String(100), unique=True, nullable=False)
    address = db.Column(db.String(50), unique=True, nullable=False)

    def __init__(self, name, address):
        self.name = name
        self.address = address


# This command will allow SQLAlchemy to synchronize with the Postgres
database, creates table automatically
db.create_all()

#CRUD operations implementation---->
# To retrieve single guest
@app.route('/guests/<id>', methods=['GET'])
def get_guest(id):
    guest = Guest.query.get(id)
    del guest.__dict__['_sa_instance_state']
    return jsonify(guest.__dict__)

#to retrieve all guests from guest book
@app.route('/guests', methods=['GET'])
def retrieve_all_guests():
    guests = []
    for guest in db.session.query(Guest).all():
        del guest.__dict__['_sa_instance_state']
        guests.append(guest.__dict__)
    return jsonify(guests)


#to add a new guest to guest book
@app.route('/guests', methods=['POST'])
def create_item():
  body = request.get_json()
  db.session.add(Guest(body['name'], body['address']))
  db.session.commit()
  return "guest added"

#To update an existing guest
@app.route('/guests/<id>', methods=['PUT'])
def update_item(id):
  body = request.get_json()
  db.session.query(Guest).filter_by(id=id).update(dict(name=body['name'],
address=body['address']))
  db.session.commit()
  return "guest updated"

#To delete guest
@app.route('/guests/<id>', methods=['DELETE'])
def delete_item(id):
  db.session.query(Guest).filter_by(id=id).delete()
  db.session.commit()
  return "guest deleted"

#new endpoint for weather.py script
@app.route('/weather-info', methods=['GET'])
def display_weather():
```

```
    page = requests.get("https://www.bbc.com/weather/1275339")
    soup = BeautifulSoup(page.text, 'html.parser')
    weather = soup.find(class_="wr-day-carousel__scrollable")

    days = weather.find_all('li')
    file_name = "weather.csv"
    f = csv.writer(open(file_name, 'w', newline=''))
    f.writerow(['Day', 'Description', 'Temperature'])
    for weather in days:
        day = weather.find(class_="wr-date").get_text()
        description = weather.find(class_="wr-day__weather-type-description-
container").get_text()
        temp = weather.find(class_="wr-day-temperature").get_text()
        print('day', day)
        print('desc', description)
        print('temp', temp)
        print('Writing rows')
        f.writerow([day, description, temp])
```

Problem 4-

I have setup sftp server on my ubuntu environment using following commands-

1. First, I have created a new user, and added it group sftp_users and changed access permissions to the user's home directly to deny access to it to any other users on the same system using below commands-

sudo groupadd sftp_users

sudo useradd -m sftpuser -g sftp

sudo passwd sftpuser

sudo chmod 700 /home/sftpuser/

2. The chroot is a way of isolating applications from the rest of your system. Basically, the SFTP chroot Jail environment restrict users to their home directories only, not any other directory. For security, the chroot functionality can be enabled in ssh settings. I have modified a few settings in /etc/sshd_config to enable sftp.

SFTP configuration-

sudo nano /etc/ssh/sshd_config

Then edited this file and commented below part-

override default of no subsystems

#Subsystem sftp /usr/lib/openssh/sftp-server

After this, added line below to enable SFTP. This will change the subsystem to internal-sftp only.

Subsystem sftp internal-sftp

3. Also added following lines in the end of file-

Match Group sftp_users

X11Forwarding no

AllowTcpForwarding no

ChrootDirectory /home

ForceCommand internal-sftp

4. Restarted ssh service to apply newly made changes.

5. Then using sftp client such as FileZilla or sftp command, we can connect this remote server securely via sftp protocol and its directories.