Deploying a Weather Forecast Application as a Background Service on EC2-Instance

**Introduction:**

This documentation talks about the process and steps to build and deploy a simple weather forecasting application on an EC2-instance as a background running service. The application runs at all times in the background once started and auto-restarts in case of any failures.

**Project Description:**

The project employs flask as its backend service for loading the application and uses an open source API from open-meteo for fetching weather data according to city coordinates. Once build locally, we transfer the application source code to our ec2-instance via git and setup our repository on our remote machine. Finally we create a service file in the system/system folder so that we can run our application as a service on our linux remote machine.

**Features:**

* Search for weather details using city/region name
* Display temperature, wind speed and an overall description of the required city
* Uses a world city database for location cities according to name and coordinates
* Uses open-meteo API to fetch weather details
* Simple web interface made using Flask and HTML

**Code:**

Below I will add the application code used in this assignment and display the project file structure:

/weather-app-assignment

├── app.py , weather.service

├── /templates

│ └── index.html

├── /static

│ └── worldcities.xlsx

App.py

from flask import Flask, render\_template, request

import pandas as pd

import requests

import os

app = Flask(\_\_name\_\_, static\_folder='static')

CITIES\_FILE = os.path.join(app.static\_folder, 'worldcities.xlsx')

def get\_coordinates(city\_name):

    data = pd.read\_excel(CITIES\_FILE)

    city = data[data['city\_ascii'].str.lower() == city\_name.lower()]

    if not city.empty:

        latitude = city.iloc[0]['lat']

        longitude = city.iloc[0]['lng']

        return latitude, longitude

    return None, None

def fetch\_weather\_data(latitude, longitude, params, data\_type):

    if data\_type == 'current':

        url = f"https://api.open-meteo.com/v1/forecast?latitude={latitude}&longitude={longitude}&current\_weather=true&hourly={','.join(params)}"

    elif data\_type == 'historical':

        url = f"https://archive-api.open-meteo.com/v1/era5?latitude={latitude}&longitude={longitude}&start\_date=2021-01-01&end\_date=2021-12-31&hourly={','.join(params)}"

    response = requests.get(url)

    if response.status\_code == 200:

        hourly\_data = response.json().get('hourly', {})

        formatted\_data = []

        for i, timestamp in enumerate(hourly\_data.get('time', [])):

            hour\_info = {'time': timestamp}

            for param in params:

                hour\_info[param] = hourly\_data.get(param, [None])[i]

            formatted\_data.append(hour\_info)

        return formatted\_data

    return []

@app.route('/', methods=['GET', 'POST'])

def index():

    weather\_data = {}

    city = None

    if request.method == 'POST':

        city = request.form['city']

        parameters = request.form.getlist('parameters')

        data\_type = request.form['data\_type']

        latitude, longitude = get\_coordinates(city)

        if latitude and longitude:

            weather\_data = fetch\_weather\_data(latitude, longitude, parameters, data\_type)

        else:

            weather\_data = {'Error': 'City not found in the database'}

    return render\_template('index.html', weather\_data=weather\_data, city=city)

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True, host='0.0.0.0')

index.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Weather Forecast</title>

</head>

<body>

    <h1>Weather Forecast Application</h1>

    <form method="POST">

        <label for="city">City:</label>

        <input type="text" id="city" name="city" required><br><br>

        <label for="data\_type">Data Type:</label>

        <select id="data\_type" name="data\_type">

            <option value="current">Current</option>

            <option value="historical">Historical</option>

        </select><br><br>

        <label>Select Parameters:</label><br>

        <input type="checkbox" name="parameters" value="temperature\_2m"> Temperature<br>

        <input type="checkbox" name="parameters" value="precipitation"> Precipitation<br>

        <input type="checkbox" name="parameters" value="wind\_speed\_10m"> Wind Speed<br>

        <button type="submit">Get Weather</button>

    </form>

    {% if weather\_data %}

    <h2>Weather Data for {{ city }}</h2>

    <table border="1" cellpadding="5">

        <thead>

            <tr>

                <th>Time</th>

                {% for param in weather\_data[0].keys() if param != 'time' %}

                <th>{{ param.replace('\_', ' ').title() }}</th>

                {% endfor %}

            </tr>

        </thead>

        <tbody>

            {% for entry in weather\_data %}

            <tr>

                <td>{{ entry.time }}</td>

                {% for param, value in entry.items() if param != 'time' %}

                <td>{{ value }}</td>

                {% endfor %}

            </tr>

            {% endfor %}

        </tbody>

    </table>

    {% endif %}

</body>

</html>

Weather.service

**[Unit]**

**Description=Flask Application Service**

**After=network.target**

**[Service]**

**User=ec2-user**

**WorkingDirectory=/home/ec2-user/weather-app-assignment**

**ExecStart=/usr/bin/python3 /home/ec2-user/weather-app-assignment/app.py**

**Restart=always**

**[Install]**

**WantedBy=multi-user.target**

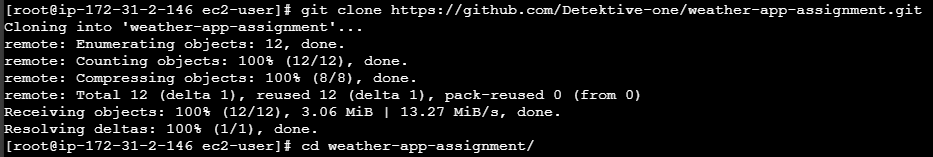
**Installation and Setup:**

**Prerequisites:**

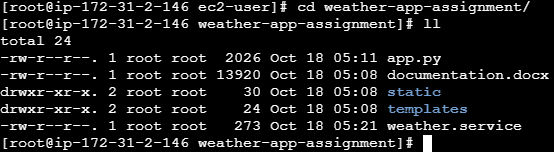
* EC2 – instance
* Python , python-pip , git installed
  + *Sudo yum install python python-pip git -y*
* Flask , openpyxl, pandas installed
  + *Pip install flask openpyxl pandas*

**Steps:**

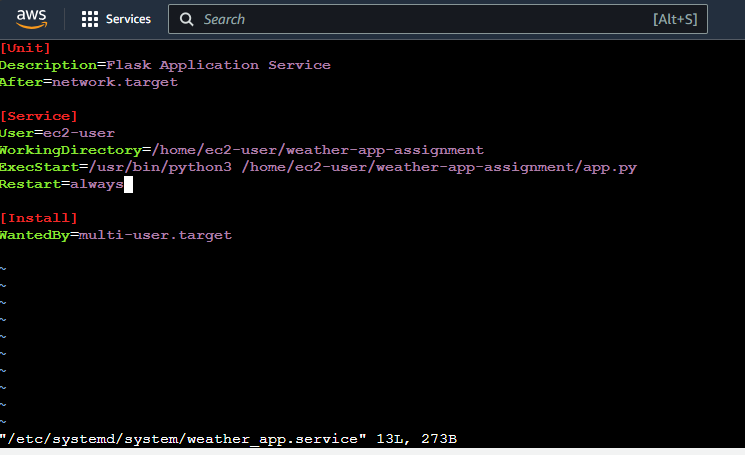
* Git clone <https://github.com/Detektive-one/weather-app-assignment.git>



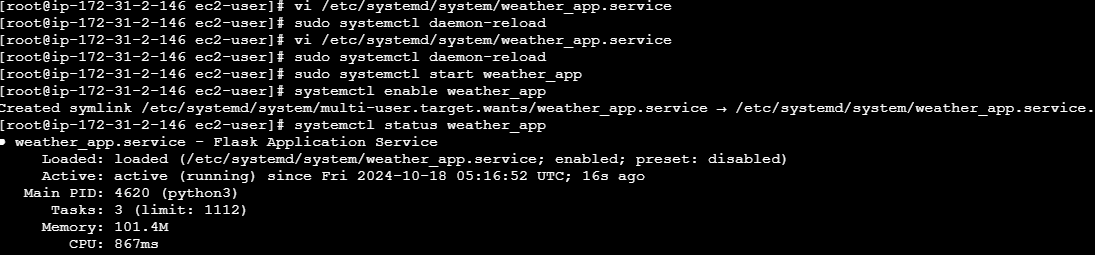
* Cd weather-app-assignment



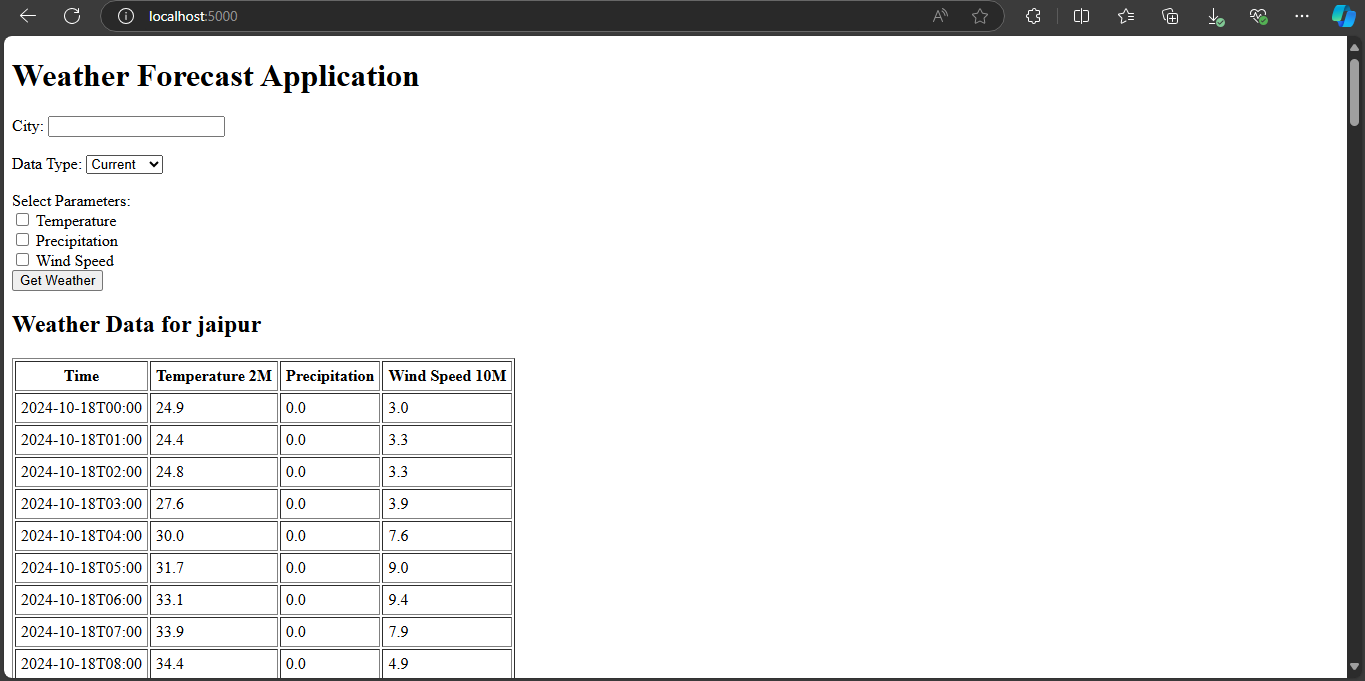
* vi weather.service [copy the contents]
* vi etc/system/system/weather-app.service

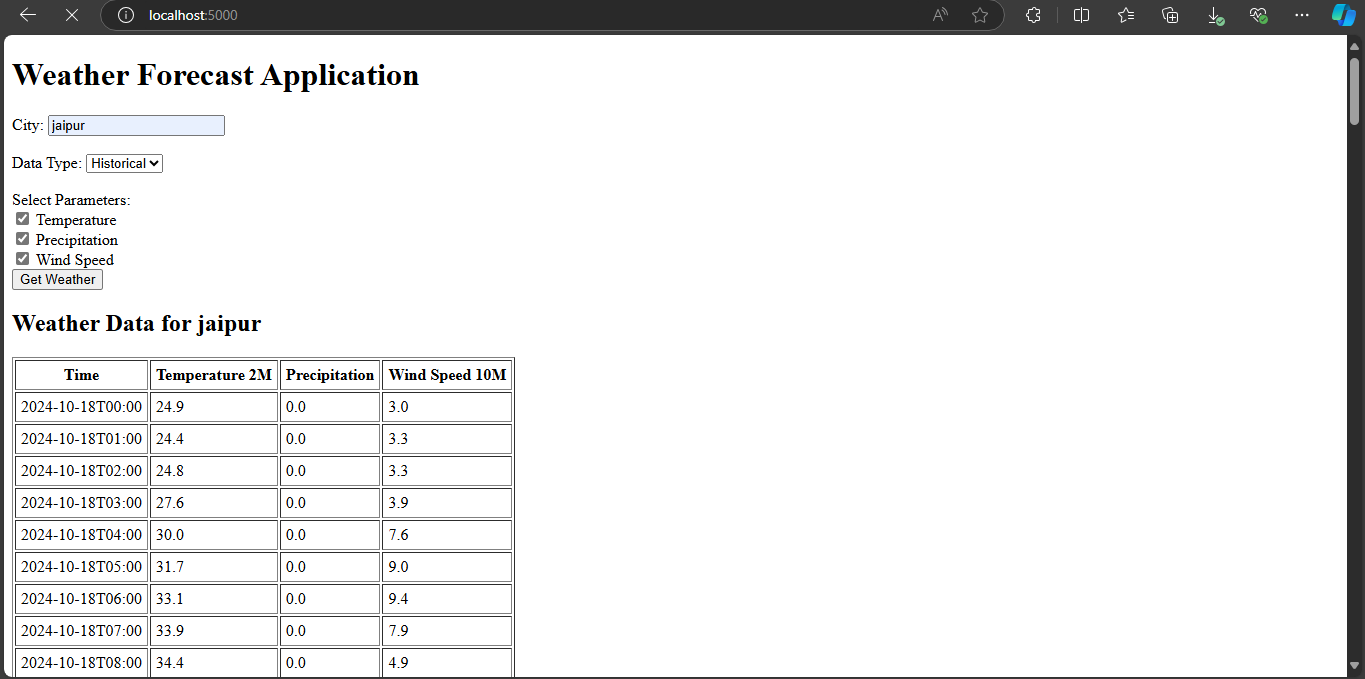


* paste the contents here.
* Sudo systemctl daemon-reload
* Sudo systemctl start weather-app
* Sudo systemctl enable weather-app
* To check the service status run sudo systemctl status weather-app [It should show active]



* Access the application at http://[public-ip]:5000





**Usage:**

* Enter a city name, choose if you wish to display current data or historical data
* Choose the weather criteria you wish to view. [temperature, humidity, wind\_speed]
* The city name is searched through the worlcities.xlxs data that stores each cities coordinates
* The fetched coordinates are input into the open-meteo API that fetches the weather details
* The fetch details are output to the application

**API Reference:**

[https://api.open-meteo.com/v1/forecast?latitude={latitude}&longitude={longitude}&current=temperature\_2m,wind\_speed\_10m](https://api.open-meteo.com/v1/forecast?latitude=%7blatitude%7d&longitude=%7blongitude%7d&current=temperature_2m,wind_speed_10m)

We used this specific API to fetch weather details. The parameters to fetch are also specified in the API. Temperature, wind speed etc.

[https://archive-api.open-meteo.com/v1/era5?latitude={latitude}&longitude={longitude}&start\_date=2021-01-01&end\_date=2021-12-31&hourly={%27,%27.join(params)}](https://archive-api.open-meteo.com/v1/era5?latitude=%7blatitude%7d&longitude=%7blongitude%7d&start_date=2021-01-01&end_date=2021-12-31&hourly=%7b%27,%27.join(params)%7d)

we used this API to fetch historical weather details.

**References:**

* [Detektive-one/weather-app-assignment: a simple weather forecast application, to deploy on ec2 as a background service. (github.com)](https://github.com/Detektive-one/weather-app-assignment)
* [🌤️ Free Open-Source Weather API | Open-Meteo.com](https://open-meteo.com/)
* [World Cities Database | Simplemaps.com](https://simplemaps.com/data/world-cities)