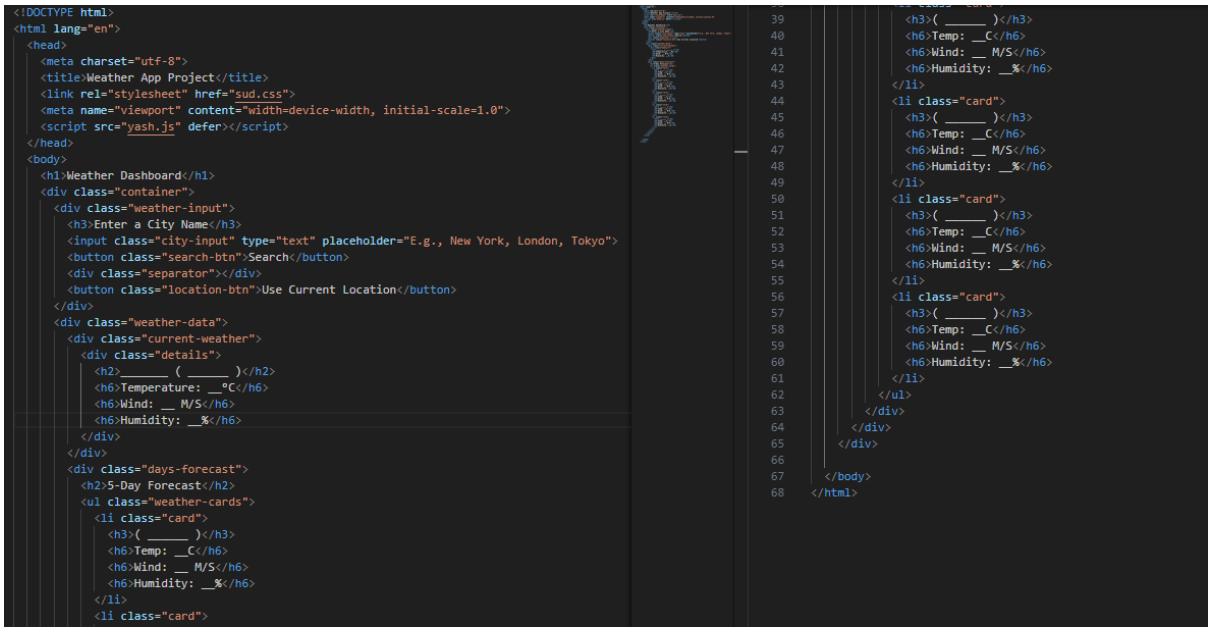
Team: VYQUADS

Team Members-

1.	Yashavanth K R	(4NI22CS254)
2.	Sudarshan B	(4NI22CS182)
3.	Suhas U	(4NI22CS222)
4.	Sujay Mudakappa Matur	(4NI22CS225)
5.	Vishal D	(4NI22CS247)
6.	Kiran K S	(4NI23CS411)
7.	Sanjay M M	(4NI22CS190)

Project Name: Simple Weather Web Application.

Development of the weather app project



```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Weather App Project</title>
    <link rel="stylesheet" href="sud.css">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script src="yash.js" defer></script>
  </head>
  <body>
    <h1>Weather Dashboard</h1>
    <div class="container">
      <div class="weather-input">
        <h3>Enter a City Name</h3>
        <input class="city-input" type="text" placeholder="E.g., New York, London, Tokyo">
        <button class="search-btn">Search</button>
        <div class="separator"></div>
        <button class="location-btn">Use Current Location</button>
      </div>
      <div class="weather-data">
        <div class="current-weather">
          <div class="details">
            <h2>_______ ( ______ )</h2>
            <h6>Temperature: __°C</h6>
            <h6>Wind: __ M/S</h6>
            <h6>Humidity: __%</h6>
          </div>
        </div>
        <div class="days-forecast">
          <h2>5-Day Forecast</h2>
          <ul class="weather-cards">
            <li class="card">
              <h3>( ______ )</h3>
              <h6>Temp: __C</h6>
              <h6>Wind: __ M/S</h6>
              <h6>Humidity: __%</h6>
            </li>
            <li class="card">
              <h3>( ______ )</h3>
              <h6>Temp: __C</h6>
              <h6>Wind: __ M/S</h6>
              <h6>Humidity: __%</h6>
            </li>
            <li class="card">
              <h3>( ______ )</h3>
              <h6>Temp: __C</h6>
              <h6>Wind: __ M/S</h6>
              <h6>Humidity: __%</h6>
            </li>
            <li class="card">
              <h3>( ______ )</h3>
              <h6>Temp: __C</h6>
              <h6>Wind: __ M/S</h6>
              <h6>Humidity: __%</h6>
            </li>
            <li class="card">
              <h3>( ______ )</h3>
              <h6>Temp: __C</h6>
              <h6>Wind: __ M/S</h6>
              <h6>Humidity: __%</h6>
            </li>
          </ul>
        </div>
      </div>
    </div>

  </body>
</html>
```

IMAGE 1: .html file of the project

```
# sud.css > ...
/* Import Google font - Open Sans */
@import url('https://fonts.googleapis.com/css2?fa
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family: 'Open Sans', sans-serif;
}
body {
  background: #E3F2FD;
}

h1 {
   background: #5372F0;
  font-size: 1.75rem;
  text-align: center;
  padding: 18px 0;
  color: #fff;
}
.container {
  display: flex;
  gap: 35px;
  padding: 30px;
}
.weather-input {
  width: 550px;
}
.weather-input input {
  height: 46px;
  width: 100%;
  outline: none;
  font-size: 1.07rem;
  padding: 0 17px;
  margin: 10px 0 20px 0;
  border-radius: 4px;
  border: 1px solid #ccc;
}
.weather-input input:focus {
```

```
# sud.css > ...
  padding: 0 16px;
  border: 2px solid #5372F0;
}
.weather-input .separator {
  height: 1px;
  width: 100%;
  margin: 25px 0;
  background: #BBBBBB;
  display: flex;
  align-items: center;
  justify-content: center;
}
.weather-input .separator::before{
  content: "or";
  color: #6C757D;
  font-size: 1.18rem;
  padding: 0 15px;
  margin-top: -4px;
  background: #E3F2FD;
}
.weather-input button {
  width: 100%;
  padding: 10px 0;
  cursor: pointer;
  outline: none;
  border: none;
  border-radius: 4px;
  font-size: 1rem;
  color: #fff;
  background: #5372F0;
  transition: 0.2s ease;
}
.weather-input .search-btn:hover {
  background: #2c52ed;
}
.weather-input .location-btn {
  background: #6C757D;
}
```

```
# sud.css > ...
.weather-input .location-btn:hover
  background: #5c636a;
}
.weather-data {
  width: 100%;
}
.weather-data .current-weather {
  color: #fff;
  background: #5372F0;
  border-radius: 5px;
  padding: 20px 70px 20px 20px;
  display: flex;
  justify-content: space-between;
}
.current-weather h2 {
  font-weight: 700;
  font-size: 1.7rem;
}
.weather-data h6 {
  margin-top: 12px;
  font-size: 1rem;
  font-weight: 500;
}
.current-weather .icon {
  text-align: center;
}
.current-weather .icon img {
  max-width: 120px;
  margin-top: -15px;
}
.current-weather .icon h6 {
  margin-top: -10px;
  text-transform: capitalize;
}
.days-forecast h2 {
  margin: 20px 0;
  font-size: 1.5rem;
}
```

```
# sud.css > ...
.weather-cards .card h3 {
  font-size: 1.3rem;
  font-weight: 600;
}
.weather-cards .card img {
  max-width: 70px;
  margin: 5px 0 -12px 0;
}

@media (max-width: 1400px) {
  .weather-data .current-weather {
    padding: 20px;
  }
  .weather-cards {
    flex-wrap: wrap;
  }
  .weather-cards .card {
    width: calc(100% / 4 - 15px);
  }
}
@media (max-width: 1200px) {
  .weather-cards .card {
    width: calc(100% / 3 - 15px);
  }
}
@media (max-width: 950px) {
  .weather-input {
    width: 450px;
  }
  .weather-cards .card {
    width: calc(100% / 2 - 10px);
  }
}
@media (max-width: 750px) {
  h1 {
    font-size: 1.45rem;
    padding: 16px 0;
  }
```

IMAGE 2: .CSS of the project

```
const cityInput = document.querySelector(".city-input");
const searchButton = document.querySelector(".search-btn");
const locationButton = document.querySelector(".location-btn");
const currentWeatherDiv = document.querySelector(".current-weather");
const weatherCardsDiv = document.querySelector(".weather-cards");

const API_KEY = "3da301f156bb5f1e5e5a425af22dcb15"; // API key for OpenWeatherMap API

const createWeatherCard = (cityName, weatherItem, index) => {
    if(index === 0) { // HTML for the main weather card
        return `<div class="details">
                    <h2>${cityName} (${weatherItem.dt_txt.split(" ")[0]})</h2>
                    <h6>Temperature: ${(weatherItem.main.temp - 273.15).toFixed(2)}°C</h6>
                    <h6>Wind: ${weatherItem.wind.speed} M/S</h6>
                    <h6>Humidity: ${weatherItem.main.humidity}%</h6>
                </div>
                <div class="icon">
                    <img src="https://openweathermap.org/img/wn/${weatherItem.weather[0].icon}@4x.png" alt="weather-icon">
                    <h6>${weatherItem.weather[0].description}</h6>
                </div>`;
    } else { // HTML for the other five day forecast card
        return `<li class="card">
                    <h3>${weatherItem.dt_txt.split(" ")[0]}</h3>
                    <img src="https://openweathermap.org/img/wn/${weatherItem.weather[0].icon}@4x.png" alt="weather-icon">
                    <h6>Temp: ${(weatherItem.main.temp - 273.15).toFixed(2)}°C</h6>
                    <h6>Wind: ${weatherItem.wind.speed} M/S</h6>
                    <h6>Humidity: ${weatherItem.main.humidity}%</h6>
                </li>`;
    }
}

const getWeatherDetails = (cityName, latitude, longitude) => {
    const WEATHER_API_URL = `https://api.openweathermap.org/data/2.5/forecast?lat=${latitude}&lon=${longitude}&appid=${API_KEY}`;

    fetch(WEATHER_API_URL).then(response => response.json()).then(data => {
        // Filter the forecasts to get only one forecast per day
        const uniqueForecastDays = [];
        const fiveDaysForecast = data.list.filter(forecast => {
```

```
            const forecastDate = new Date(forecast.dt_txt).getDate();
            if (!uniqueForecastDays.includes(forecastDate)) {
                return uniqueForecastDays.push(forecastDate);
            }
        });

        // Clearing previous weather data
        cityInput.value = "";
        currentWeatherDiv.innerHTML = "";
        weatherCardsDiv.innerHTML = "";

        // Creating weather cards and adding them to the DOM
        fiveDaysForecast.forEach((weatherItem, index) => {
            const html = createWeatherCard(cityName, weatherItem, index);
            if (index === 0) {
                currentWeatherDiv.insertAdjacentHTML("beforeend", html);
            } else {
                weatherCardsDiv.insertAdjacentHTML("beforeend", html);
            }
        });
    }).catch(() => {
        alert("An error occurred while fetching the weather forecast!");
    });
}

const getCityCoordinates = () => {
    const cityName = cityInput.value.trim();
    if (cityName === "") return;
    const API_URL = `https://api.openweathermap.org/geo/1.0/direct?q=${cityName}&limit=1&ap

    // Get entered city coordinates (latitude, longitude, and name) from the API response
    fetch(API_URL).then(response => response.json()).then(data => {
        if (!data.length) return alert(`No coordinates found for ${cityName}`);
        const { lat, lon, name } = data[0];
        getWeatherDetails(name, lat, lon);
    }).catch(() => {
        alert("An error occurred while fetching the coordinates!");
    });
```

```
}

const getUserCoordinates = () => {
    navigator.geolocation.getCurrentPosition(
        position => {
            const { latitude, longitude } = position.coords; // Get coordinates of user lo
            // Get city name from coordinates using reverse geocoding API
            const API_URL = `https://api.openweathermap.org/geo/1.0/reverse?lat=${latitude
            fetch(API_URL).then(response => response.json()).then(data => {
                const { name } = data[0];
                getWeatherDetails(name, latitude, longitude);
            }).catch(() => {
                alert("An error occurred while fetching the city name!");
            });
        },
        error => { // Show alert if user denied the location permission
            if (error.code === error.PERMISSION_DENIED) {
                alert("Geolocation request denied. Please reset location permission to gra
            } else {
                alert("Geolocation request error. Please reset location permission.");
            }
        });
}

locationButton.addEventListener("click", getUserCoordinates);
searchButton.addEventListener("click", getCityCoordinates);
cityInput.addEventListener("keyup", e => e.key === "Enter" && getCityCoordinates());
```

IMAGE3-4: JS of the project

1. Creating project directory

Create html, CSS, JavaScript for your website and a dockerfile in a single folder(weather_app).

Dockerimg2	07-11-2024 10:19	Dockerfile Source ...	1 KB
sud	06-11-2024 15:25	CSS Source File	4 KB
sud	06-11-2024 20:20	Compressed (zipp...	2 KB
vish	07-11-2024 09:34	Microsoft Edge HT...	3 KB
yash.js	06-11-2024 15:25	JSFile	5 KB

Image 5: project directory

Content of Dockerimg2.dockerfile:

```
# Start from the official NGINX image
FROM nginx:latest

# Copy the application files to the default NGINX location
COPY vish.html /usr/share/nginx/html/index.html
COPY sud.css /usr/share/nginx/html/sud.css
COPY yash.js /usr/share/nginx/html/yash.js
```

Image 6: Dockerimg2.dockerfile

It will import the NGINX default image.

Make sure that you will give index.html at the end of first COPY command as NGINX default image will always refer to index.html.

COPY command will copy files into default image.

2. Building docker image

Open command prompt.

Enter to your folder directory By using cd.

Then give the following command:

docker build -f Dockerimg2.dockerfile -t weather_app_image .

Dockerimg2.dockerfile : your docker file name.

weather_app_image : docker image name.

```
C:\Users\yashw\OneDrive\Desktop\weather_app>docker build -f Dockerimg2.dockerfile -t weather_app_image .
[+] Building 0.8s (9/9) FINISHED                                                                  docker:desktop-linux
 => [internal] load build definition from Dockerimg2.dockerfile                                                    0.1s
 => => transferring dockerfile: 216B                                                                               0.1s
 => [internal] load metadata for docker.io/library/nginx:latest                                                    0.0s
 => [internal] load .dockerignore                                                                                  0.0s
 => => transferring context: 2B                                                                                    0.0s
 => [internal] load build context                                                                                  0.0s
 => => transferring context: 2.25kB                                                                                0.0s
 => CACHED [1/4] FROM docker.io/library/nginx:latest                                                               0.0s
 => [2/4] COPY vish.html /usr/share/nginx/html/index.html                                                          0.1s
 => [3/4] COPY sud.css /usr/share/nginx/html/sud.css                                                               0.0s
 => [4/4] COPY yash.js /usr/share/nginx/html/yash.js                                                               0.1s
 => exporting to image                                                                                             0.2s
 => => exporting layers                                                                                            0.1s
 => => writing image sha256:7da5382f06318662f708b5c24512ba51e6a088dec414a0a5cfb116f97a4891b8                       0.0s
 => => naming to docker.io/library/weather_app_image                                                               0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/5pgck21mowieg166wva4scvg7

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
```

Image 7: docker build -f Dockerimg2.dockerfile -t weather_app_image . command

3. Running the docker image

After building the docker image, run the following command.

docker run -p 8081:80 weather_app_image

```
C:\Users\yashw\OneDrive\Desktop\weather_app>docker run -p 8081:80 weather_app_image
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2024/11/07 04:51:57 [notice] 1#1: using the "epoll" event method
2024/11/07 04:51:57 [notice] 1#1: nginx/1.27.2
2024/11/07 04:51:57 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2024/11/07 04:51:57 [notice] 1#1: OS: Linux 5.15.153.1-microsoft-standard-WSL2
2024/11/07 04:51:57 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2024/11/07 04:51:57 [notice] 1#1: start worker processes
2024/11/07 04:51:57 [notice] 1#1: start worker process 29
2024/11/07 04:51:57 [notice] 1#1: start worker process 30
2024/11/07 04:51:57 [notice] 1#1: start worker process 31
2024/11/07 04:51:57 [notice] 1#1: start worker process 32
2024/11/07 04:51:57 [notice] 1#1: start worker process 33
2024/11/07 04:51:57 [notice] 1#1: start worker process 34
2024/11/07 04:51:57 [notice] 1#1: start worker process 35
2024/11/07 04:51:57 [notice] 1#1: start worker process 36
2024/11/07 04:51:57 [notice] 1#1: start worker process 37
2024/11/07 04:51:57 [notice] 1#1: start worker process 38
2024/11/07 04:51:57 [notice] 1#1: start worker process 39
2024/11/07 04:51:57 [notice] 1#1: start worker process 40
```

Image 8: docker run -p 8081:80 weather_app_image command

Here, we are using the port 8081 for this application to run.

Access your web application through **http://localhost:8081**

Deploying the project on AWS

Note: In this documentation, we have deployed a weather app project. So use your project name wherever there is " weather_app" mentioned. And we have used Dockerimg.dockerfile as our dockerfile. So replace that also by your dockerfile name.

Part 1: Set Up AWS and Create an EC2 Instance

1. **Create an AWS Account**:
   - Go to AWS and sign up for a new account.
   - Provide your payment details. AWS has a free tier, which should cover basic hosting.

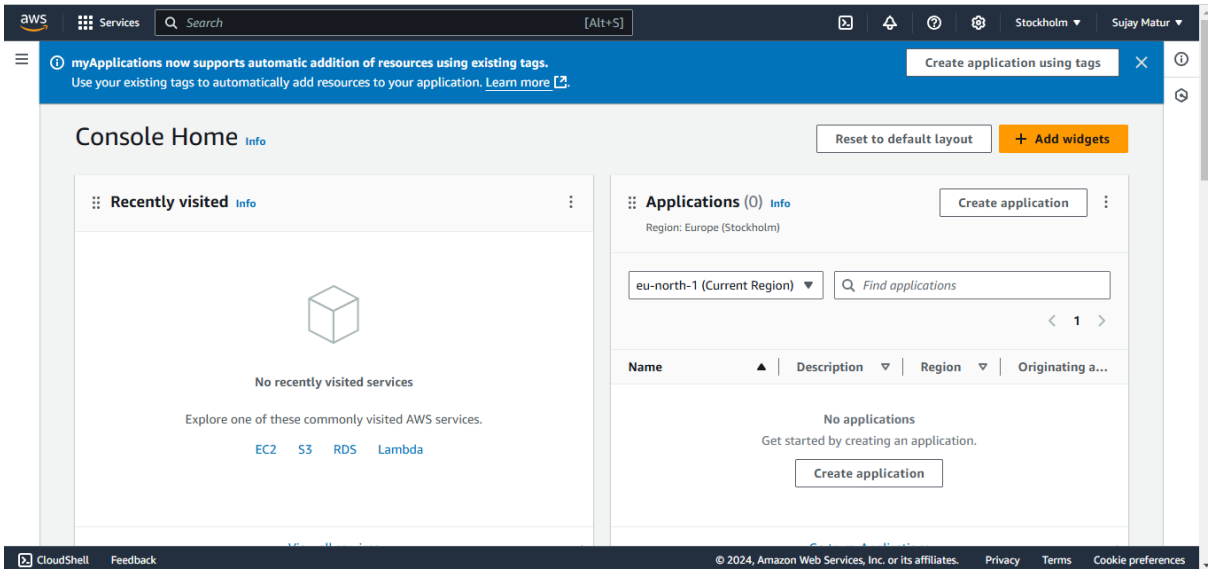


IMAGE 9:AWS account home page

2. **Log In and Launch an EC2 Instance**:

- In the AWS console, go to **Services > EC2** and select **Launch Instance**.
- Choose **Amazon Linux 2** or **Ubuntu** as the operating system.
- For **Instance Type**, select **t2.micro** (free tier eligible).
- Click **Next** until you reach the **Security Group** section. Here, add rules to allow:
  - **HTTP** (port 80) for web access.
  - **SSH** (port 22) for secure shell access.
- Click **Review and Launch** and then **Launch**.
- Download the `.pem` key file when prompted (e.g., `my-key.pem`). This file is used for SSH access.

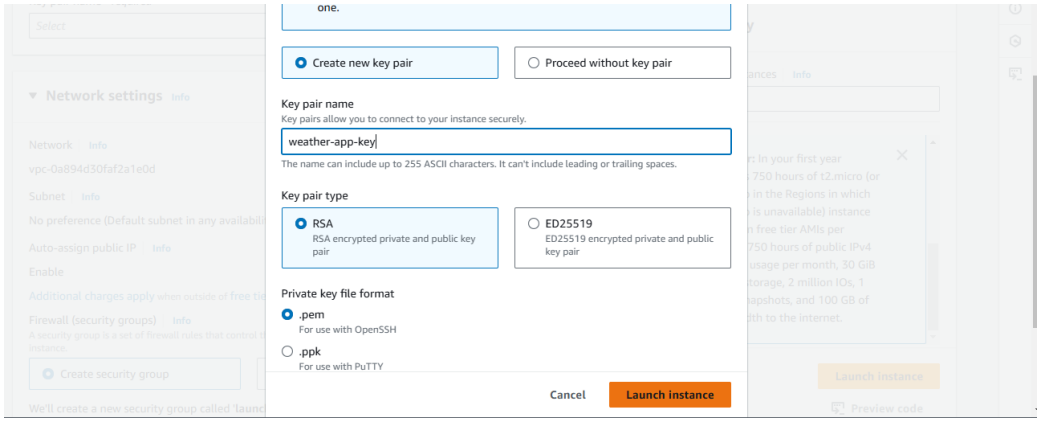


IMAGE 10: Setting .pem filename

Part 2: Set Up SSH and Transfer Files from Windows

3. **Prepare Windows for SSH**:
   - Open **PowerShell** or **Command Prompt** on Windows.
   - Navigate to the folder containing your `.pem` file (replace `path\to\your-key.pem` with your actual path):

   In powershell: cd path\to\your-key.pem

4. **Connect to EC2 via SSH**:

- In **PowerShell** or **Command Prompt**, connect using this command:

```
ssh -i "my-key.pem" ec2-user@your-ec2-public-dns
```

- Replace `my-key.pem` with your key filename and `your-ec2-public-dns` with the public DNS name of your instance (available in your EC2 Dashboard).
- This command will run an instance of EC2 in powershell.

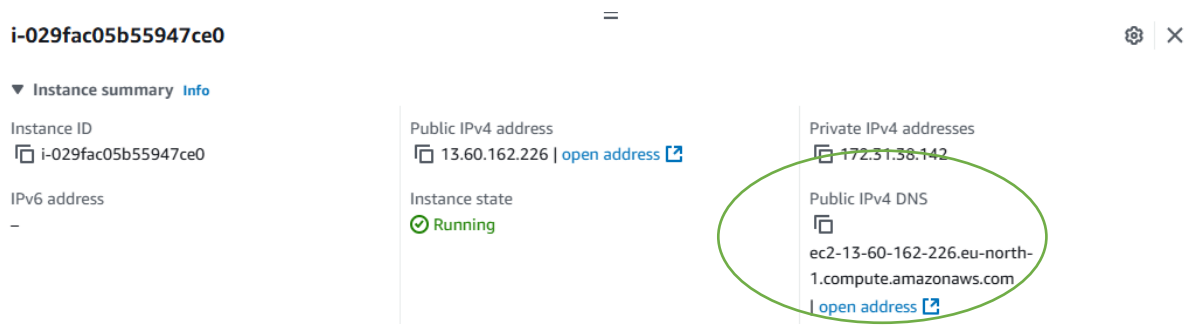


IMAGE 11: `your-ec2-public-dns`

5. **Transfer Project Files**:

- Use `scp` (secure copy protocol) to transfer your project files. In **PowerShell**, run:

```
scp -i "my-key.pem" path\to\project\files\* ec2-user@your-ec2-public-dns:/home/ec2-user/weather_app
```

replace "my-key.pem" and path\to\project\files* and your-ec2-public-dns accordingly.

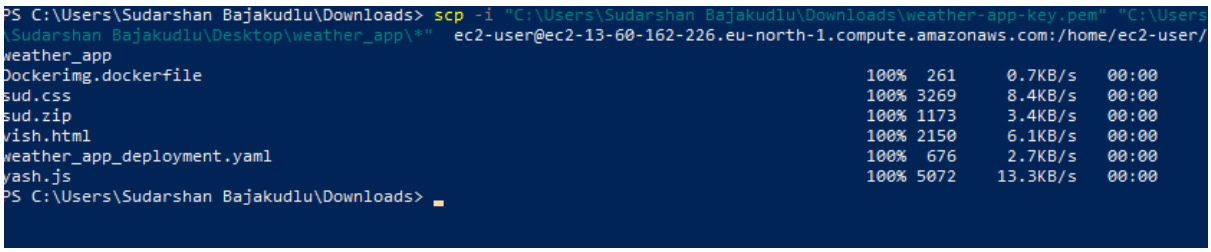


IMAGE 12: Transfering project files

Part 3: Install Docker and Set Up the App on EC2

First connect to your EC2 instance using the command:

ssh -i "path\to\your-key.pem" ec2-user@your-ec2-public-dns

replace path\to\your-key.pem and your-ec2-public-dns accordingly.

6. **Install Docker on the EC2 Instance**:
   - Once connected to your EC2 instance via SSH, update and install Docker:

     ```
     sudo yum update -y      # For Amazon Linux 2
     sudo yum install docker -y
     ```

   - Start Docker and add your user to the Docker group:

     ```
     sudo service docker start
     sudo usermod -aG docker ec2-user
     ```

   - Log out and log back in to apply Docker permissions.
   - To log out, use ‘exit’ command.
   - To log in back, use ssh -i "path\to\your-key.pem" ec2-user@your-ec2-public-dns command.

```
PS C:\Users\Sudarshan Bajakudlu\Downloads> ssh -i "C:\Users\Sudarshan Bajakudlu\Downloads\weather-app-key.pem" ec2-user@ec2-13-60-162-226.eu-north-1.compute.amazonaws.com
       ,     #_
       ~\_  ####_        Amazon Linux 2023
      ~~  \_#####\
      ~~     \###|
      ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
       ~~       V~' '->
        ~~~         /
          ~~._.   _/
             _/ _/
           _/m/'
[ec2-user@ip-172-31-38-142 ~]$ sudo yum update -y
Last metadata expiration check: 0:38:32 ago on Fri Nov  8 17:32:29 2024.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-38-142 ~]$ sudo yum install docker -y
Last metadata expiration check: 0:39:08 ago on Fri Nov  8 17:32:29 2024.
Dependencies resolved.
================================================================================================================================== Package                          Architecture       Version
                    Repository              Size
================================================================================================================================Installing:
 docker                                 x86_64            25.0.6-1.amzn2023.0.2             amazonlinux             44 M
Installing dependencies:
 containerd                             x86_64            1.7.22-1.amzn2023.0.2             amazonlinux             36 M
 iptables-libs                          x86_64            1.8.8-3.amzn2023.0.2              amazonlinux            401 k
 iptables-nft                           x86_64            1.8.8-3.amzn2023.0.2              amazonlinux            183 k
 libcgroup                              x86_64            3.0-1.amzn2023.0.1                amazonlinux             75 k
 libnetfilter_conntrack                 x86_64            1.0.8-2.amzn2023.0.2              amazonlinux             58 k
 libnfnetlink                           x86_64            1.0.1-19.amzn2023.0.2             amazonlinux             30 k
 libnftnl                               x86_64            1.2.2-2.amzn2023.0.2              amazonlinux             84 k
 pigz                                   x86_64            2.5-1.amzn2023.0.3                amazonlinux             83 k
 runc                                   x86_64            1.1.14-1.amzn2023.0.1             amazonlinux            3.2 M

Transaction Summary
================================================================================================================================Install  10 Packages

Total download size: 84 M
Installed size: 317 M
Downloading Packages:
(1/10): iptables-libs-1.8.8-3.amzn2023.0.2.x86_64.rpm                                5.2 MB/s | 401 kB     00:00
(2/10): iptables-nft-1.8.8-3.amzn2023.0.2.x86_64.rpm                                 6.7 MB/s | 183 kB     00:00
(3/10): libcgroup-3.0-1.amzn2023.0.1.x86_64.rpm                                      3.8 MB/s |  75 kB     00:00
(4/10): libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64.rpm                       3.0 MB/s |  58 kB     00:00
(5/10): libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64.rpm                                1.6 MB/s |  30 kB     00:00
(6/10): libnftnl-1.2.2-2.amzn2023.0.2.x86_64.rpm                                     3.0 MB/s |  84 kB     00:00
(7/10): pigz-2.5-1.amzn2023.0.3.x86_64.rpm                                           3.6 MB/s |  83 kB     00:00
(8/10): runc-1.1.14-1.amzn2023.0.1.x86_64.rpm                                         57 MB/s | 3.2 MB     00:00
(9/10): containerd-1.7.22-1.amzn2023.0.2.x86_64.rpm                                   58 MB/s |  36 MB     00:00
```

IMAGE 13: Docker installation in EC2 instance

```
[ec2-user@ip-172-31-38-142 ~]$ sudo service docker start
Redirecting to /bin/systemctl start docker.service
[ec2-user@ip-172-31-38-142 ~]$ sudo usermod -aG docker ec2-user
[ec2-user@ip-172-31-38-142 ~]$ exit
logout
Connection to ec2-13-60-162-226.eu-north-1.compute.amazonaws.com closed.
PS C:\Users\Sudarshan Bajakudlu\Downloads>
```

IMAGE 14: Starting docker engine, log out and log back in to apply docker permissions

7. **Navigate to Project Directory**:

- On the EC2 instance, navigate to the folder containing your files:

```
cd /home/ec2-user/weather_app
```

8. **Build and Run Docker Container**:

- Build the Docker image using your Dockerfile:

```
docker build -t weather_app_image -f Dockerimg.dockerfile .
```

- Run the container, mapping it to port 80:

```
docker run -d -p 80:80 weather_app_image
```

(-d refers to detached container which will run independently even if you log out of your EC2 instance.

Part 4: Access Your Weather App

9. **Access Your App**:
    - Open your browser and go to `http://your-ec2-public-dns`. Replace `your-ec2-public-dns` with the actual public DNS of your instance.
    - Your weather app should now be live!

This guide provides a step-by-step setup for hosting your weather app on AWS EC2 using Docker from a Windows system.

For detailed information, checkout:

https://chatgpt.com/share/672e6a7f-d760-8009-8955-b0ebabd0e873

Our final project link:

http://ec2-13-60-162-226.eu-north-1.compute.amazonaws.com/

Deployment using Github

1. Login to your GitHub Account and Create a new repository for your project.

2. upload the project files in the repository.

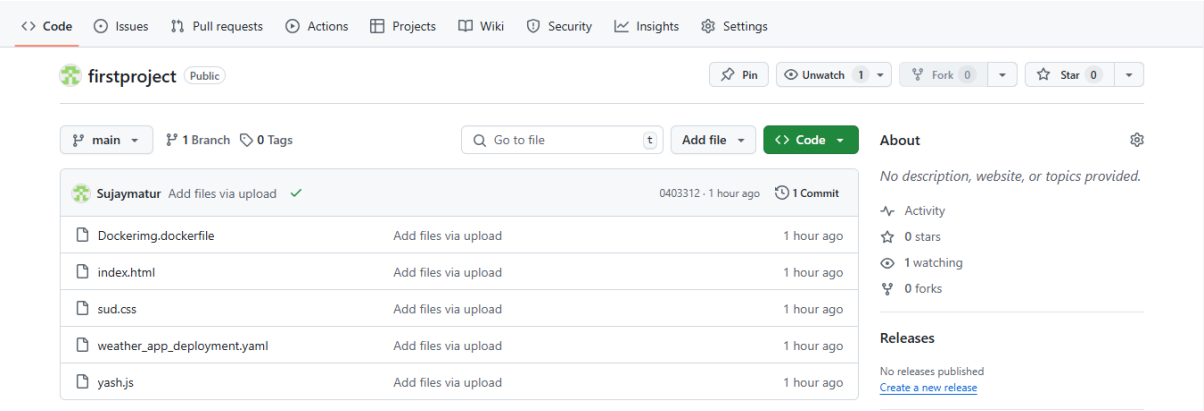


IMAGE 15: Github repository

3.Select the settings and select pages in that. In Source: Select Deploy from the branch. And in branchselect main and root. And click on save.

4.After some time you will get the link where you project is deployed.

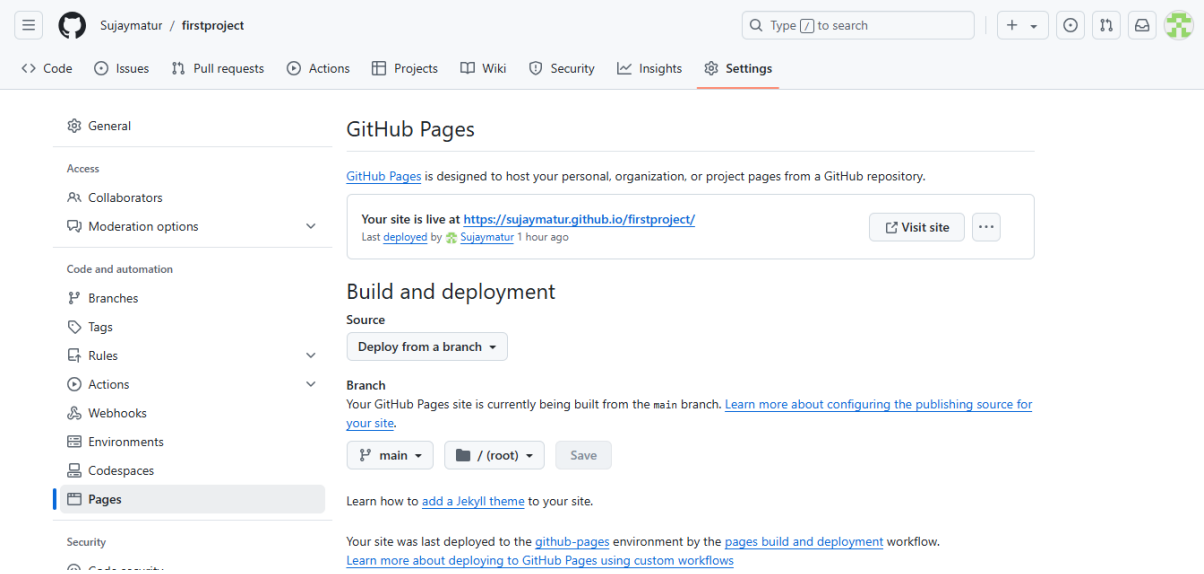


IMAGE 16: Link generated in Settings/pages

Project Link: https://sujaymatur.github.io/firstproject/

Deployment using Netlify

Create an account in Netlify

Then upload your project **folder** into Netlify. It will generate the peoject link.

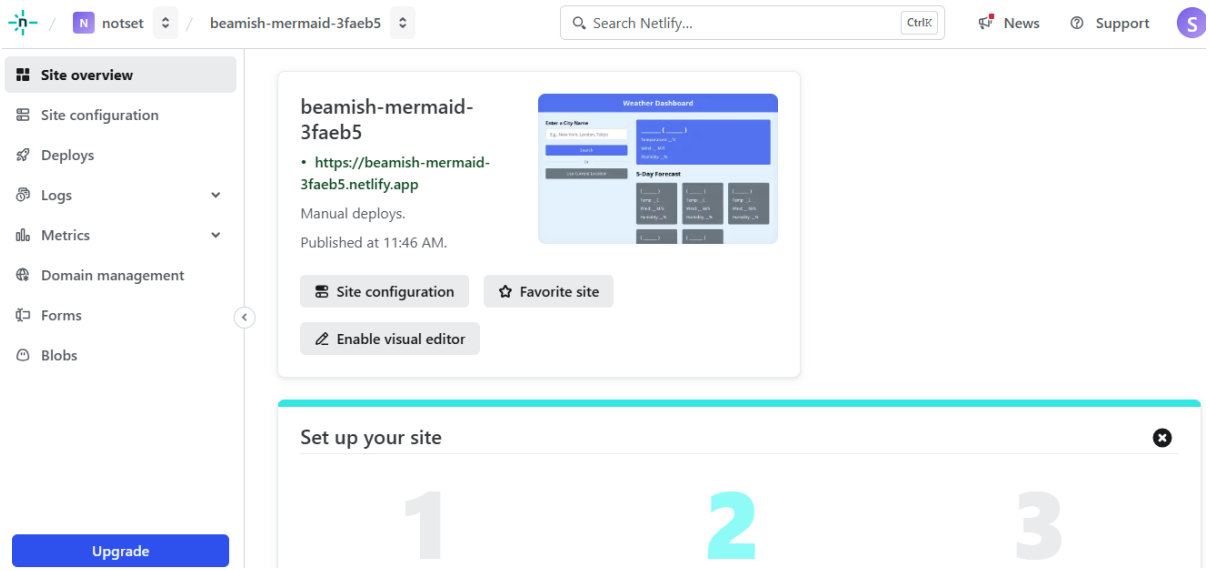


IMAGE 17: Project deployed in Netlify

Project Link:https://beamish-mermaid-3faeb5.netlify.app/

Deploy the Weather app on Vercel:

This guide explains how to deploy the Weather app project on Vercel from the setup to deployment steps.

Step 1: Prepare the Project Files

1. Clone or Download the Project:

• Clone the repository from GitHub: Weather App GitHub Repository.

2. Install Dependencies:

• In the project directory, install the required dependencies using the requirements.txt file. pip install -r requirements.txt

Step 2: Deploy the Project to Vercel

1. Login to Vercel:

• Go to vercel.com and log in or create an account.

2. Create a New Project:

- Click New Project in the Vercel dashboard.
- Select Import Git Repository and link the Weather app repository.

Step 3: Deploy the Application

1. Click Deploy to start the deployment process.
2. Vercel will automatically build and deploy your project.
3. Once deployment is complete, Vercel will provide a URL where your Weather App is live and accessible.

Step 4: Test and Monitor the Deployment

1. Test the Web App:

• Open the provided Vercel URL and test the Weather App's functionality.

2. Monitor Performance:

• In the Vercel dashboard, use the Analytics and Logs sections to monitor the deployment's performance and troubleshoot if needed.

This guide outlines deploying the Gemini Chatbot on Vercel. With Vercel's seamless deployment pipeline, you can also enjoy automatic updates when you push changes to the linked repository.

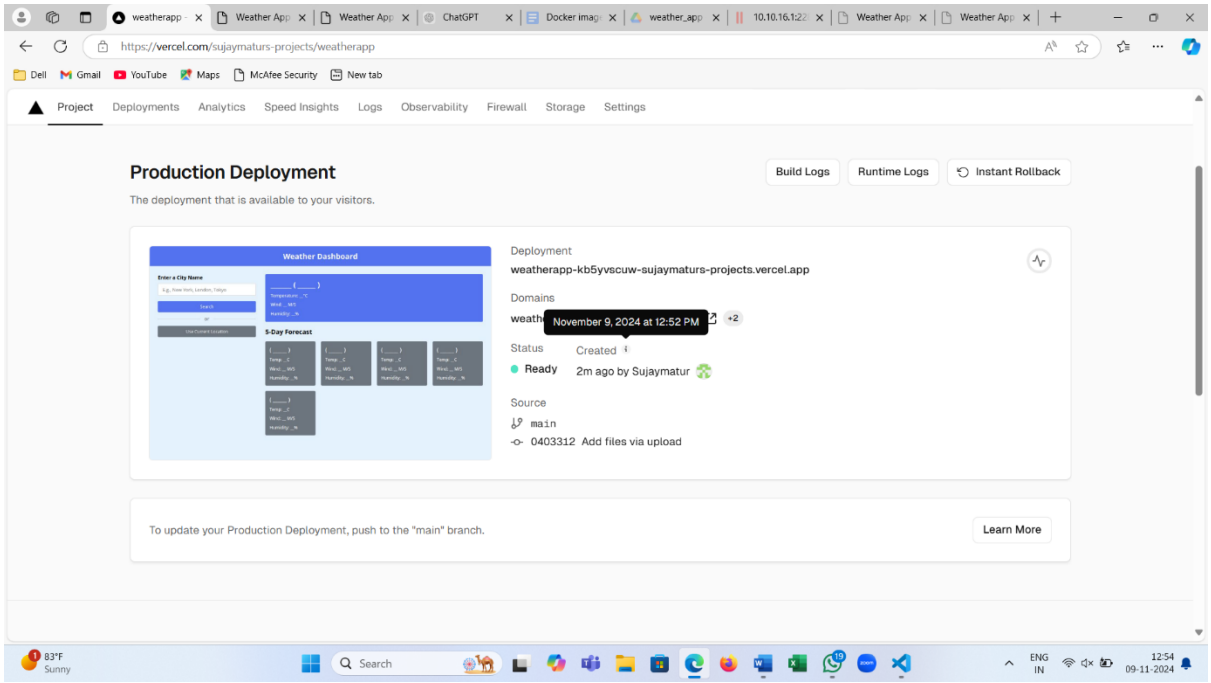


Image 18: Project deployed in vercel

Project link: https://weatherapp-seven-henna.vercel.app/