Computer and Communication Networks
INSTRUCTOR: Dr. Hassaan Khaliq Qureshi
COURSE CODE: EE357
SEMESTER: 6th

PROJECT BY: Umer Abdullah Khan
CMS: 375870

Create a containerized weather application available as a web front end using Minikube.

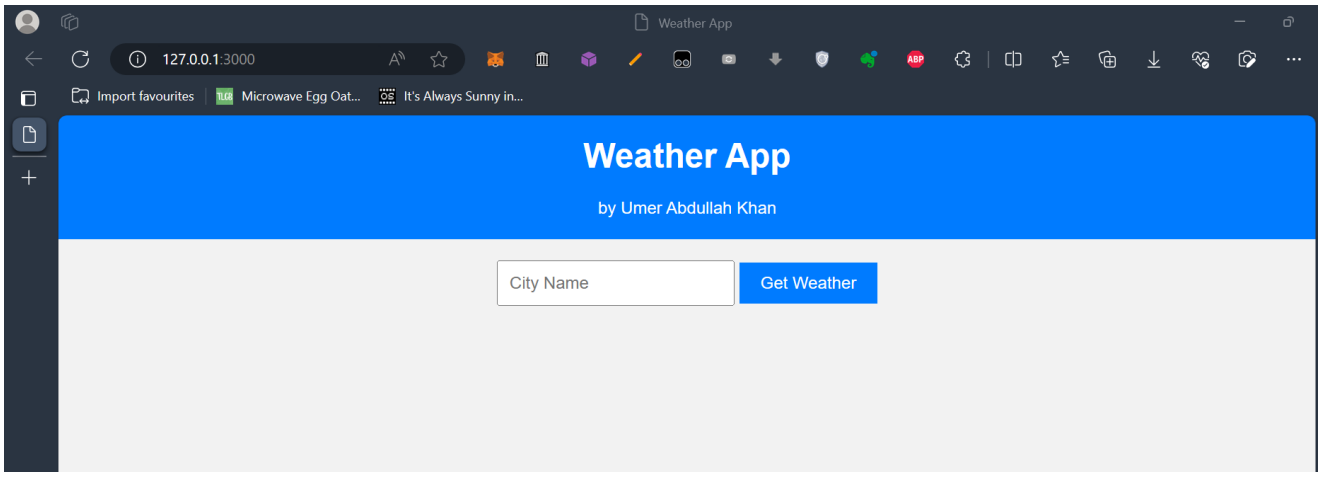
1. Create a python application that uses OpenWeatherAPI to get the weather data.

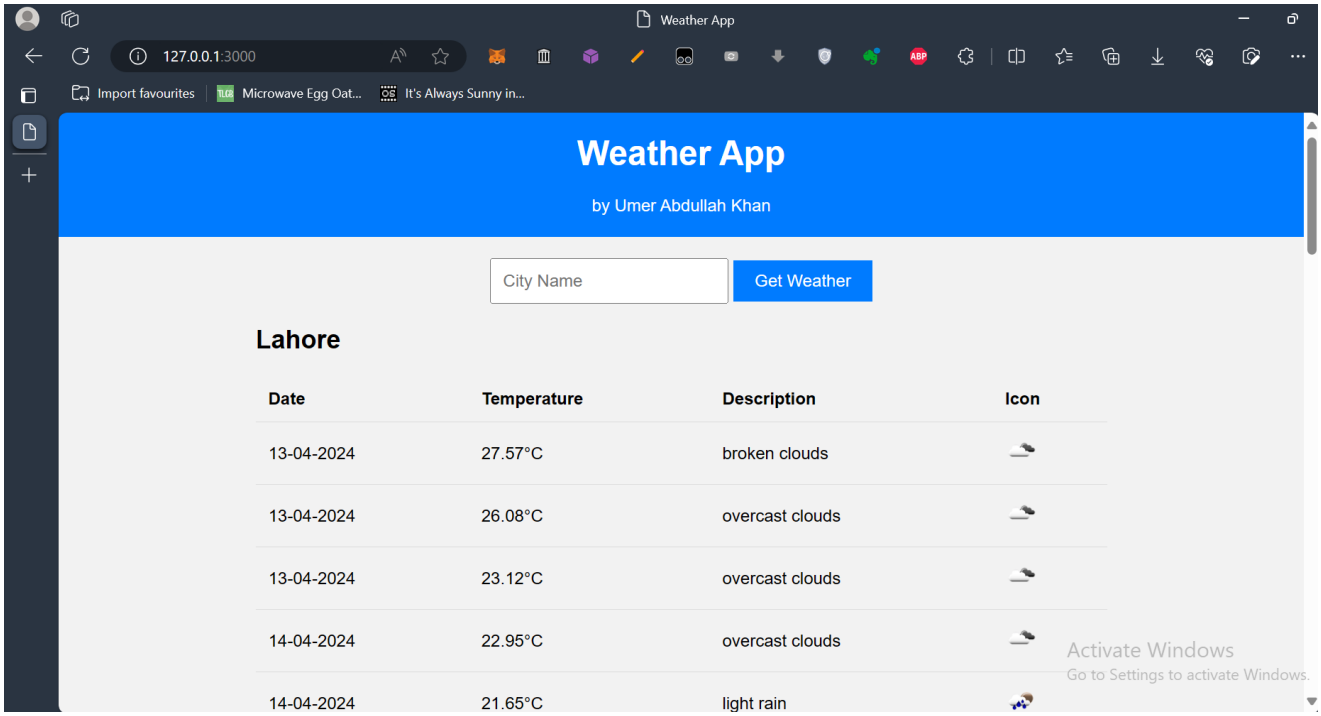
> Note
>
> Codes for python and html in Appendix (7. Appendix)

.
.
.
.

```
PS F:\clone ccn proj> & "f:/clone ccn proj/venv/Scripts/python.exe" "f:/clone ccn proj/weather.py"
 * Serving Flask app 'weather'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:3000
 * Running on http://192.168.18.52:3000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 484-903-257
```

(weather forecasted for next 5 days with 3 hour step)

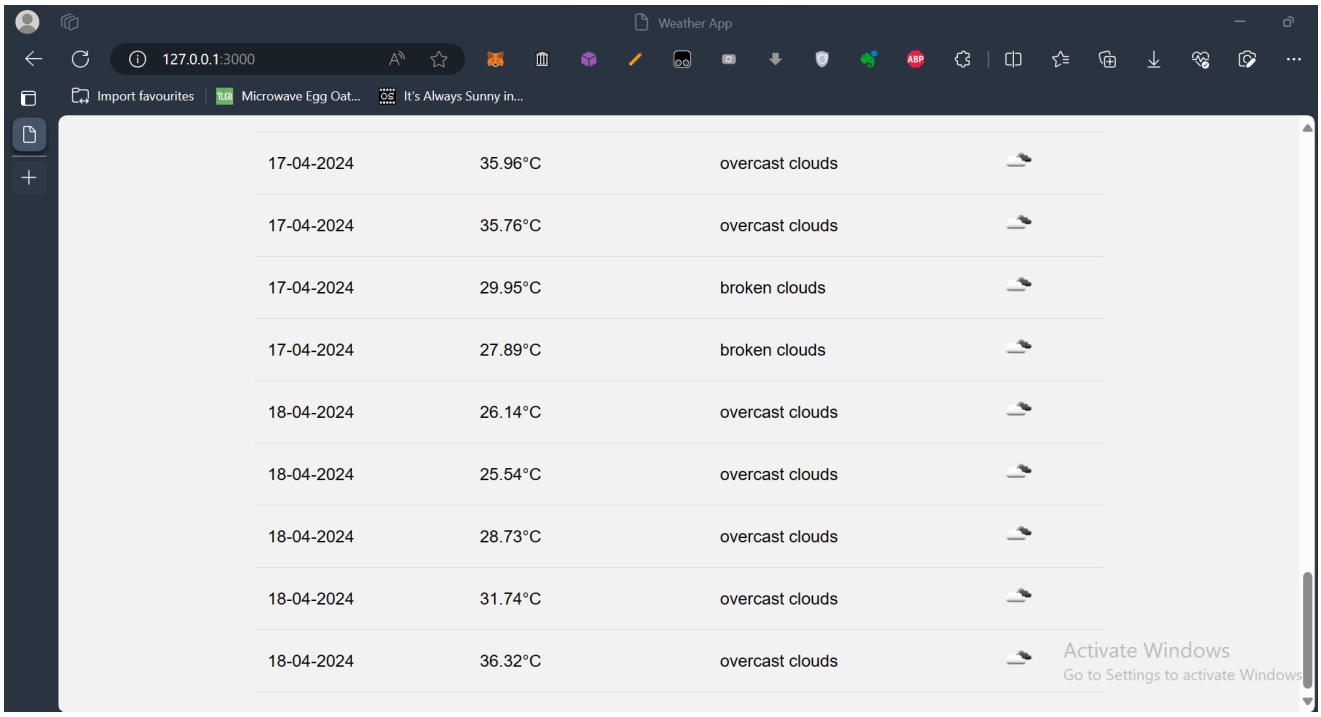

Weather App
127.0.0.1:3000
Import favourites
Microwave Egg Oat...
It's Always Sunny in...
Weather App
by Umer Abdullah Khan
City Name
Get Weather


Weather App
127.0.0.1:3000
Import favourites
Microwave Egg Oat...
It's Always Sunny in...
Weather App
by Umer Abdullah Khan
City Name
Get Weather
Lahore
Date
Temperature
Description
Icon
13-04-2024
27.57°C
broken clouds
13-04-2024
26.08°C
overcast clouds
13-04-2024
23.12°C
overcast clouds
14-04-2024
22.95°C
overcast clouds
14-04-2024
21.65°C
light rain
Activate Windows
Go to Settings to activate Windows.

.

.

.


Weather App
127.0.0.1:3000
Import favourites
Microwave Egg Oat...
It's Always Sunny in...
17-04-2024
35.96°C
overcast clouds
17-04-2024
35.76°C
overcast clouds
17-04-2024
29.95°C
broken clouds
17-04-2024
27.89°C
broken clouds
18-04-2024
26.14°C
overcast clouds
18-04-2024
25.54°C
overcast clouds
18-04-2024
28.73°C
overcast clouds
18-04-2024
31.74°C
overcast clouds
18-04-2024
36.32°C
overcast clouds
Activate Windows
Go to Settings to activate Windows.

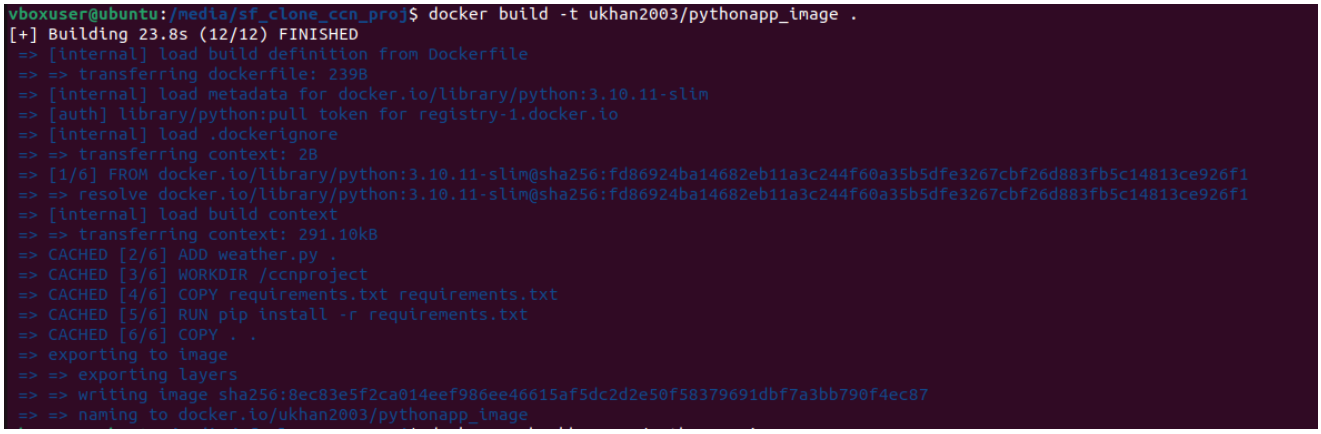
2. Using Docker to containerize this application.

Dockerfile

```
FROM python:3.10.11-slim

ADD weather.py .

WORKDIR /ccnproject

COPY requirements.txt requirements.txt

RUN pip install -r requirements.txt

COPY . .

ENTRYPOINT [ "python3"]

CMD [ "./weather.py" ]
```

Building Docker Image

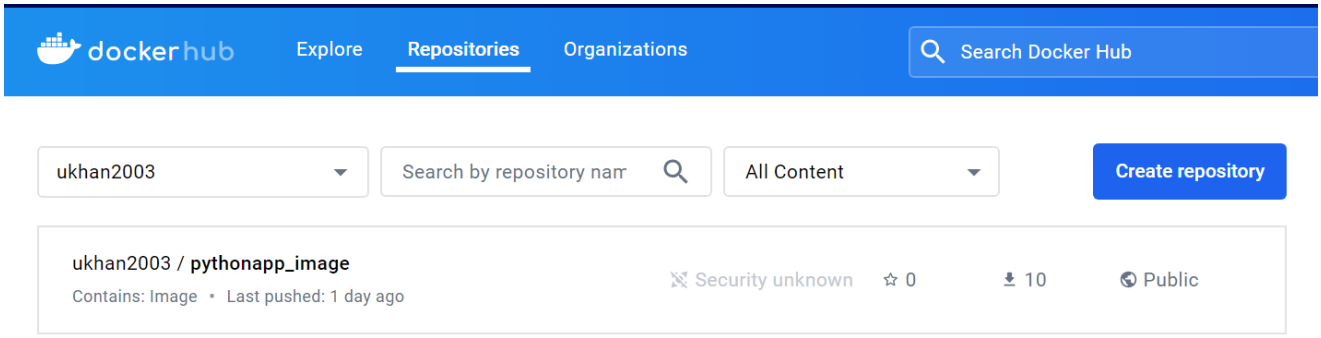
```
docker build -t ukahn2003/pythonapp_image .
```



Pushing image to Docker Hub Registry

```
docker push ukhan2003/pythonapp_image
```

```
vboxuser@ubuntu:/media/sf_clone_ccn_proj$ docker push ukhan2003/pythonapp_image
Using default tag: latest
The push refers to repository [docker.io/ukhan2003/pythonapp_image]
edd3a5ce96a9: Pushed
968b54d7ef94: Pushed
eb4fc2be92b7: Pushed
4c6e29a854e3: Pushed
7a82130634a1: Pushed
fc0712ddcc40: Mounted from library/python
bf5aece7e593: Mounted from library/python
a8fd1ddfcbf8: Mounted from library/python
6669ab6e06c6: Mounted from library/python
8cbe4b54fa88: Mounted from library/python
latest: digest: sha256:32e624fade8697cc7fef59f3205e43b3a6033339d7d9906c18d07425d9eced92 size: 2415
```



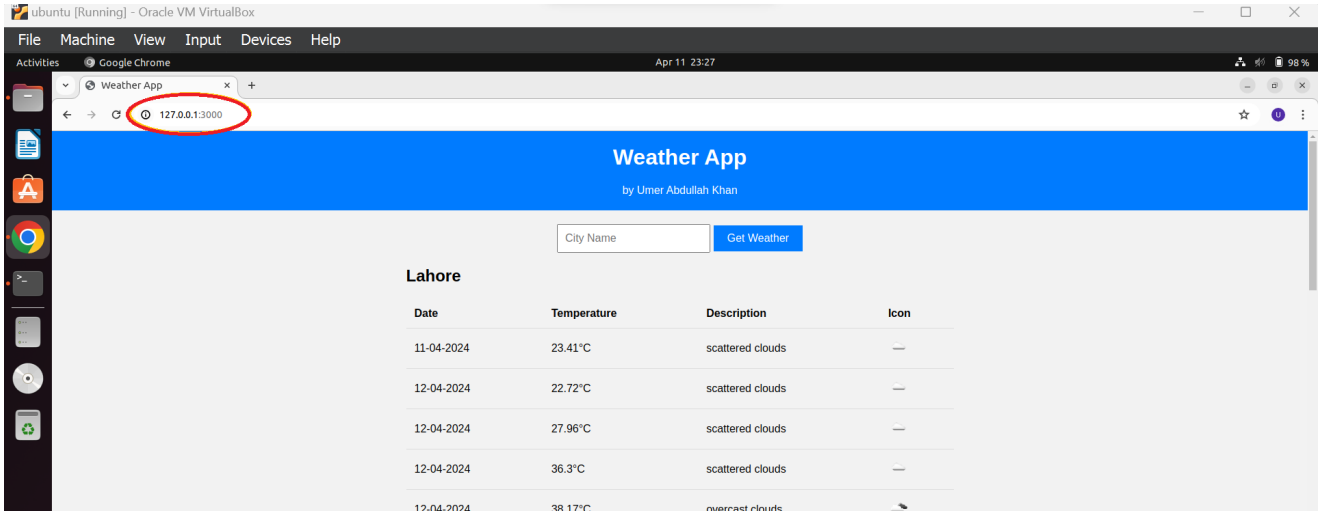
Running the container

```
docker run -p 3000:3000 ukhan2003/pythonapp_image
```

```
vboxuser@ubuntu:/media/sf_clone_ccn_proj$ docker run -p 3000:3000 ukhan2003/pythonapp_image
 * Serving Flask app 'weather'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:3000
 * Running on http://172.17.0.2:3000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 687-190-489
172.17.0.1 - - [11/Apr/2024 18:21:25] "GET / HTTP/1.1" 200 -
```

Running the containerized python app

(it works 👍)



3. Creating a pod in Minikube using our image

Check that Minikube is properly installed, by running the *minikube version* command:

```
vboxuser@ubuntu:~$ minikube version
minikube version: v1.32.0
commit: 8220a6eb95f0a4d75f7f2d7b14cef975f050512d
```

Now we can start the cluster by running the *minikube start* command.
View the cluster details by running ;

```
kubectl cluster-info
```

```
vboxuser@ubuntu:/media/sf_clone_ccn_proj$ kubectl cluster-info
Kubernetes control plane is running at https://192.168.49.2:8443
CoreDNS is running at https://192.168.49.2:8443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

> Note
>
> Cluster IP = 192.168.49.2

To view the nodes in the cluster, run;

```
kubectl get nodes
```

```
vboxuser@ubuntu:/media/sf_clone_ccn_proj$ k get nodes
NAME       STATUS   ROLES           AGE    VERSION
minikube   Ready    control-plane   2m46s   v1.28.3
```

Let's deploy our first app on Kubernetes.

```
nano deployment.yaml
```

This will create a YAML file for us where we can add the definitions for deployments as well as service which will be needed in the next task. The deployment and service definitions are separated using "---".

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: pythonapp-service
spec:
  selector:
    matchLabels:
      app: pythonapp
  replicas: 1
  template:
    metadata:
```

```
      labels:
        app: pythonapp
    spec:
      containers:
        - name: flask-test-app
          image: docker.io/ukhan2003/pythonapp_image
          imagePullPolicy: IfNotPresent
          ports:
            - containerPort: 3000
---

apiVersion: v1
kind: Service
metadata:
  name: pythonapp-deploy
spec:
  selector:
    app: pythonapp
  ports:
    - protocol: "TCP"
      port: 6000
      targetPort: 3000
  type: LoadBalancer
```

File details and explanations:

.

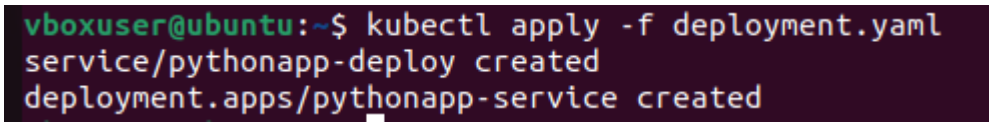
.

.

.

.

```
kubectl apply -f deployment.yaml
```




```
kubectl get all
```

```
vboxuser@ubuntu:~$ kubectl get all
NAME                                      READY   STATUS    RESTARTS   AGE
pod/pythonapp-service-598fb87c7c-n765d    1/1     Running   0          2m41s

NAME                        TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
service/kubernetes          ClusterIP      10.96.0.1       <none>        443/TCP          4m32s
service/pythonapp-deploy    LoadBalancer   10.96.102.207   <pending>     6000:31411/TCP   2m41s

NAME                                   READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/pythonapp-service      1/1     1            1           2m41s

NAME                                              DESIRED   CURRENT   READY   AGE
replicaset.apps/pythonapp-service-598fb87c7c      1         1         1       2m41s
```

4. Expose this pod on a port (using service)

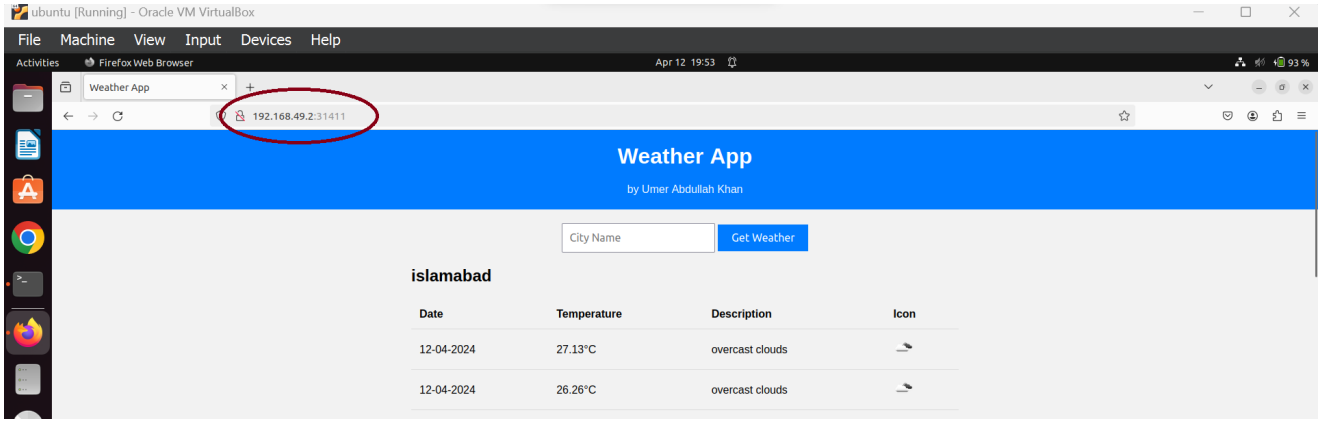
We can use the command below combined with the name of our service

```
minikube service pythonapp-deploy
```

This automatically open the application on the browser. It should display as shown below;

```
vboxuser@ubuntu:~$ minikube service pythonapp-deploy
|-----------|-------------------|-------------|---------------------------|
| NAMESPACE |       NAME        | TARGET PORT |            URL            |
|-----------|-------------------|-------------|---------------------------|
| default   | pythonapp-deploy  |        6000 | http://192.168.49.2:31411 |
|-----------|-------------------|-------------|---------------------------|
🎉  Opening service default/pythonapp-deploy in default browser...
```

(Works 👍)



5 .Observe and interact with this application on your host machine

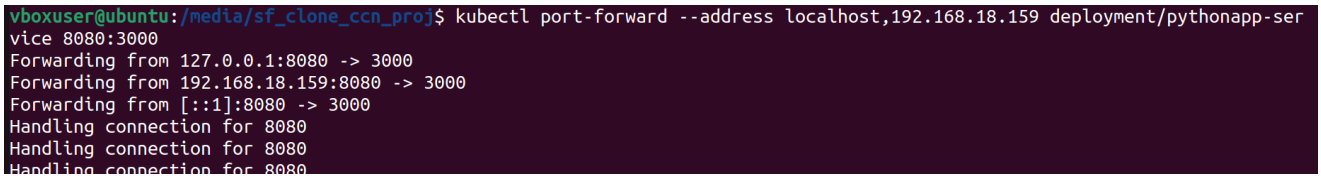
To make out host access the application, we can use port-forwarding.

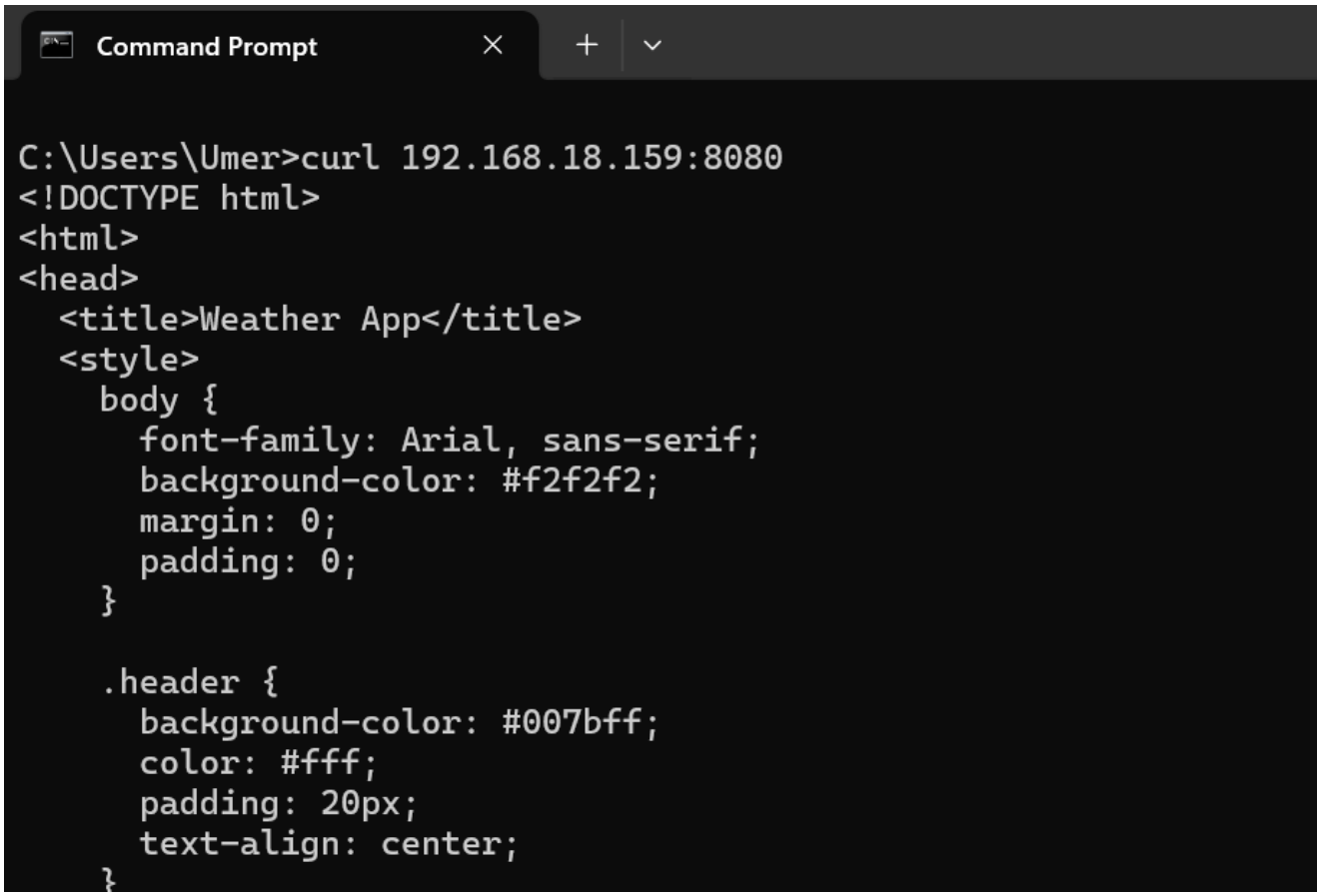
.

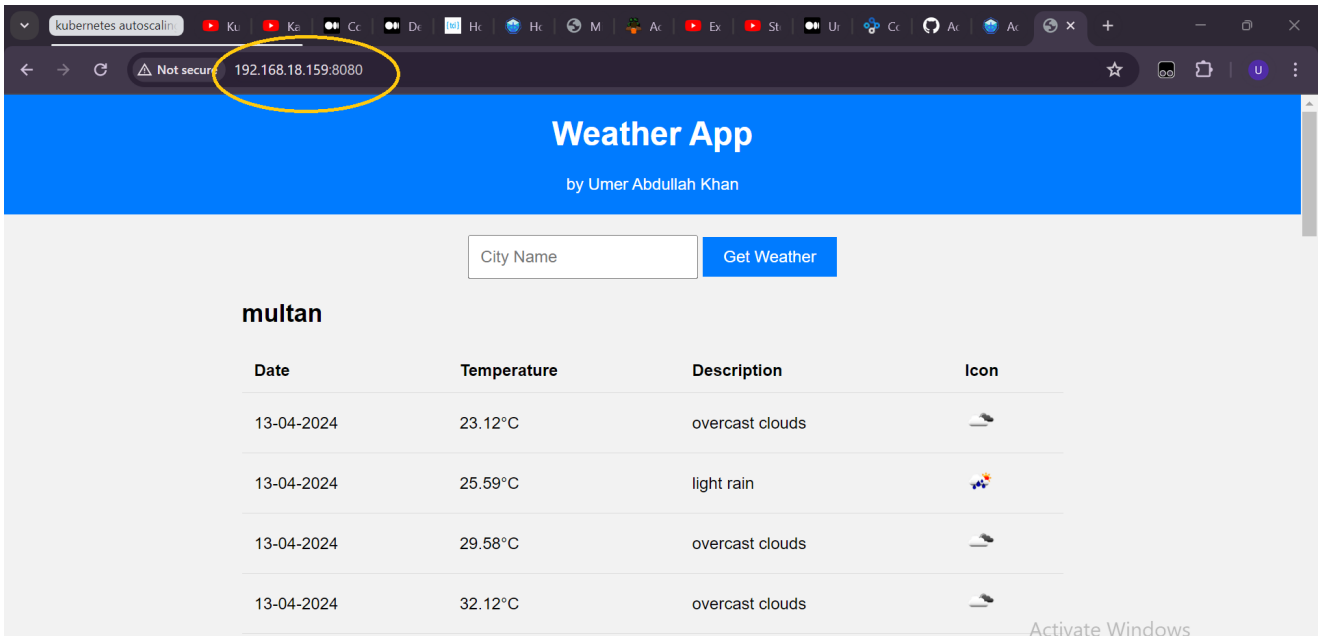
INFO

HOST : WINDOWS 11 OS
VM : UBUNTU 22.04

```
kubectl port-forward --address localhost,192.168.18.159
deployment/pythonapp-service 8080:3000
```

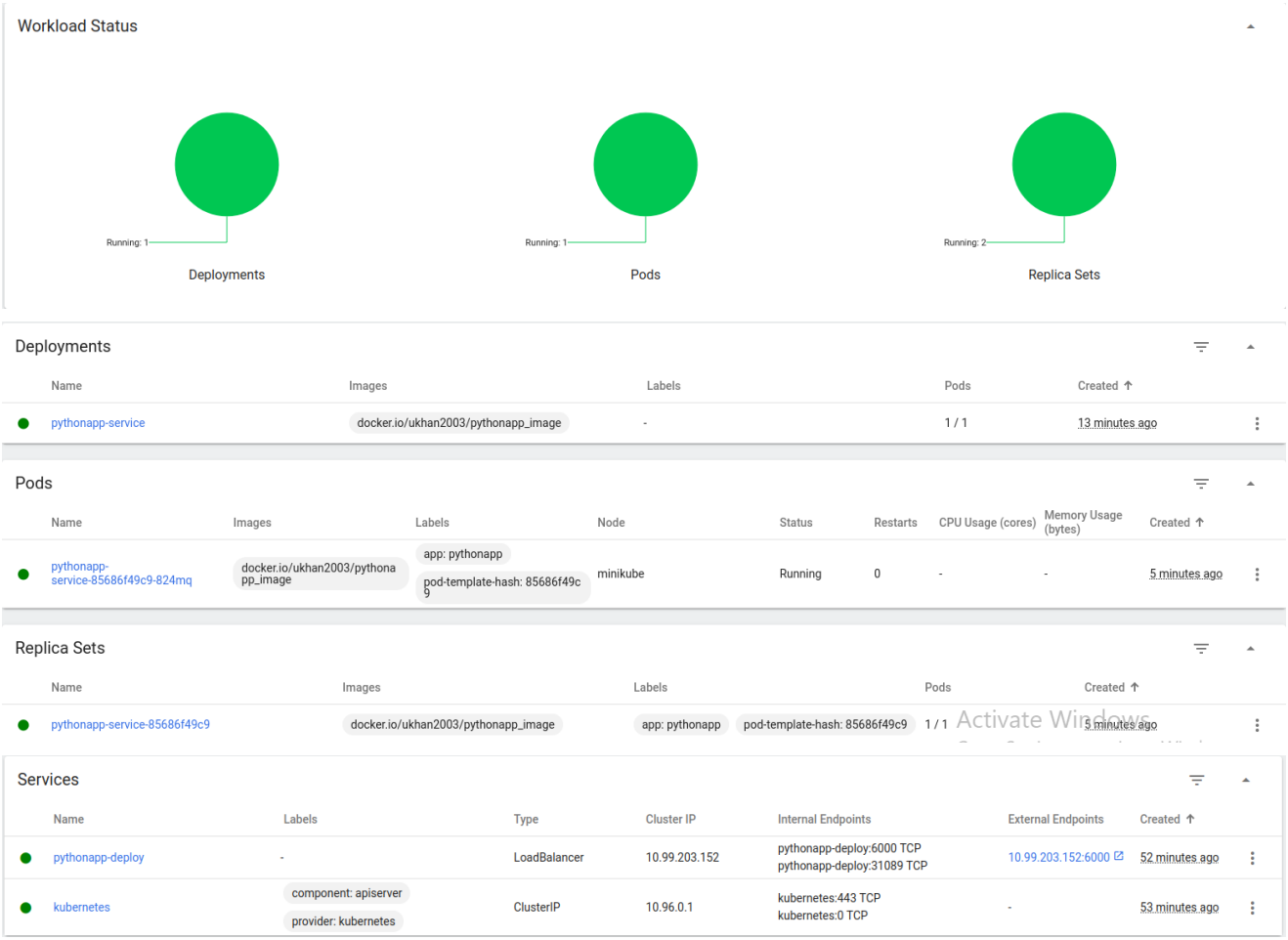






6. Deploy the Minikube dashboard to observe statistics related to your cluster.

Viewing the Minikube Dashboard



7. Appendix

A: Python Code (1. Create a python application that uses OpenWeatherAPI to get the weather data.)

"weather.py"

```
from flask import Flask, request, render_template
import requests
import datetime as dt

app = Flask(__name__)

API_KEY = '007d68128ef4ec788f611c7c0cc7f68a'

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

def search_city():
    if request.method == "POST":
        city = request.form.get("city")
        if city == "" or len(city) <= 1:
            error_message = "City is required."
            return render_template("weather.html",
error_message=error_message)
```

```
        units = 'Metric'
        url = f'http://api.openweathermap.org/data/2.5/forecast?q=
{city}&APPID={API_KEY}&units={units}'
        response = requests.get(url).json()
        if response["cod"] != "200":
            error_message = "City not found."
            return render_template("weather.html",
error_message=error_message)
        forecast_data = response["list"]
        forecast = []
        for item in forecast_data:
            forecast_date =
dt.datetime.fromtimestamp(item["dt"]).strftime('%d-%m-%Y')
            forecast_temp = item["main"]["temp"]
            forecast_desc = item["weather"][0]["description"]
            forecast_icon = item["weather"][0]["icon"]
            forecast.append({"date": forecast_date, "temp": forecast_temp,
"desc": forecast_desc, "icon": forecast_icon})
        return render_template("weather.html", city=city, forecast=forecast)
    return render_template("weather.html")
if __name__ == '__main__':
    app.run(host="0.0.0.0", port=3000, debug=True)
```

B: HTML code (1. Create a python application that uses OpenWeatherAPI to get the weather data.)

```
<!DOCTYPE html>

<html>
<head>
  <title>Weather App</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f2f2f2;
      margin: 0;
      padding: 0;
    }

    .header {
      background-color: #007bff;
      color: #fff;
      padding: 20px;
      text-align: center;
    }

    .container {
      max-width: 800px;
```

```
  margin: 0 auto;
  padding: 20px;
}


h1 {
  margin-top: 0;
  font-size: 32px;
}

p {
  margin: 0;
  font-size: 16px;
}

form {
  margin-bottom: 20px;
  text-align: center;
}

input[type="text"] {
  padding: 10px;
  font-size: 16px;
  width: 200px;
}

button {
  padding: 10px 20px;
  font-size: 16px;
  background-color: #007bff;
  color: #fff;
  border: none;
  cursor: pointer;
}

table {
  width: 100%;
  border-collapse: collapse;
  margin-top: 20px;
}

th, td {
  padding: 12px;
  text-align: left;
  border-bottom: 1px solid #ddd;
}


.weather-icon {
```

```
      width: 30px;
      height: 30px;
    }

  </style>
</head>
<body>
  <div class="header">

    <h1>Weather App</h1>

    <p>by Umer Abdullah Khan</p>

  </div>

  <div class="container">

    <form method="POST">

      <input name="city" type="text" placeholder="City Name">

      <button>Get Weather</button>

    </form>



    {% if error_message %}

      <p>{{ error_message }}</p>

    {% endif %}



    {% if forecast %}

      <h2>{{ city }}</h2>

      <table>

        <tr>

          <th>Date</th>

          <th>Temperature</th>

          <th>Description</th>
```

```
          <th>Icon</th>
        </tr>
        {% for entry in forecast %}
          <tr>
            <td>{{ entry.date }}</td>
            <td>{{ entry.temp }}°C</td>
            <td>{{ entry.desc }}</td>
            <td><img class="weather-icon"
src="http://openweathermap.org/img/w/{{ entry.icon }}.png" alt="Weather
Icon"></td>
          </tr>
        {% endfor %}
      </table>
    {% endif %}
  </div>
</body>
</html>
```