Project Objective: Dockerised Web Apps on IBM Cloud

This presentation explores the streamlined process of deploying Dockerised web applications on IBM Cloud, focusing on a practical weather application example.

Agenda

1. ***Project Overview***

   Understand the weather application and its core components.

2. ***Tools & Technologies***

   A deep dive into the essential tools used in this deployment.

3. ***Deployment Workflow and Steps***

   Visualising the journey from code to cloud.

4. ***Docker Benefits***

   Exploring the advantages of containerisation.

5. ***IBM Cloud Features***

   Key capabilities of IBM Cloud for containerised applications.

6. ***Conclusion & Next Steps***

   Summarising key takeaways and future considerations.

Project Overview: Weather Web Application

Our project demonstrates a simple weather web application that fetches real-time weather data. It's built with standard web technologies and designed for easy containerisation.

- **Functionality:** Displays current weather conditions for a user-specified location.
- **Data Source:** Leverages the OpenWeather API for weather data.
- **Goal:** To illustrate an end-to-end deployment of a Dockerised app on IBM Cloud.

Essential Tools & Technologies

HTML, CSS, JavaScript

Frontend development for user interface and interactivity.

OpenWeather API

Provides meteorological data for the application.

GitHub

Version control and collaborative code hosting.

VS Code

Integrated Development Environment for coding and Docker integration.

Docker

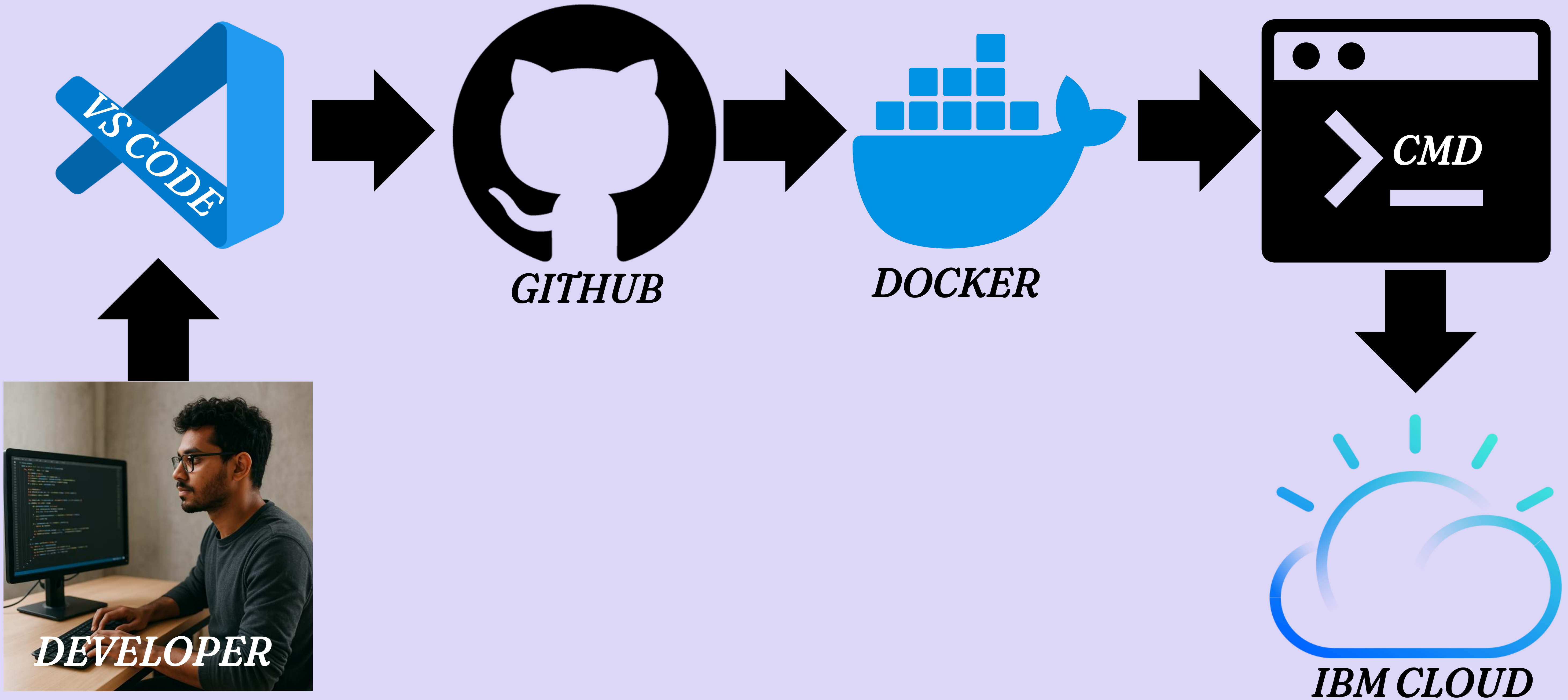
Containerisation platform for packaging the application.

Python

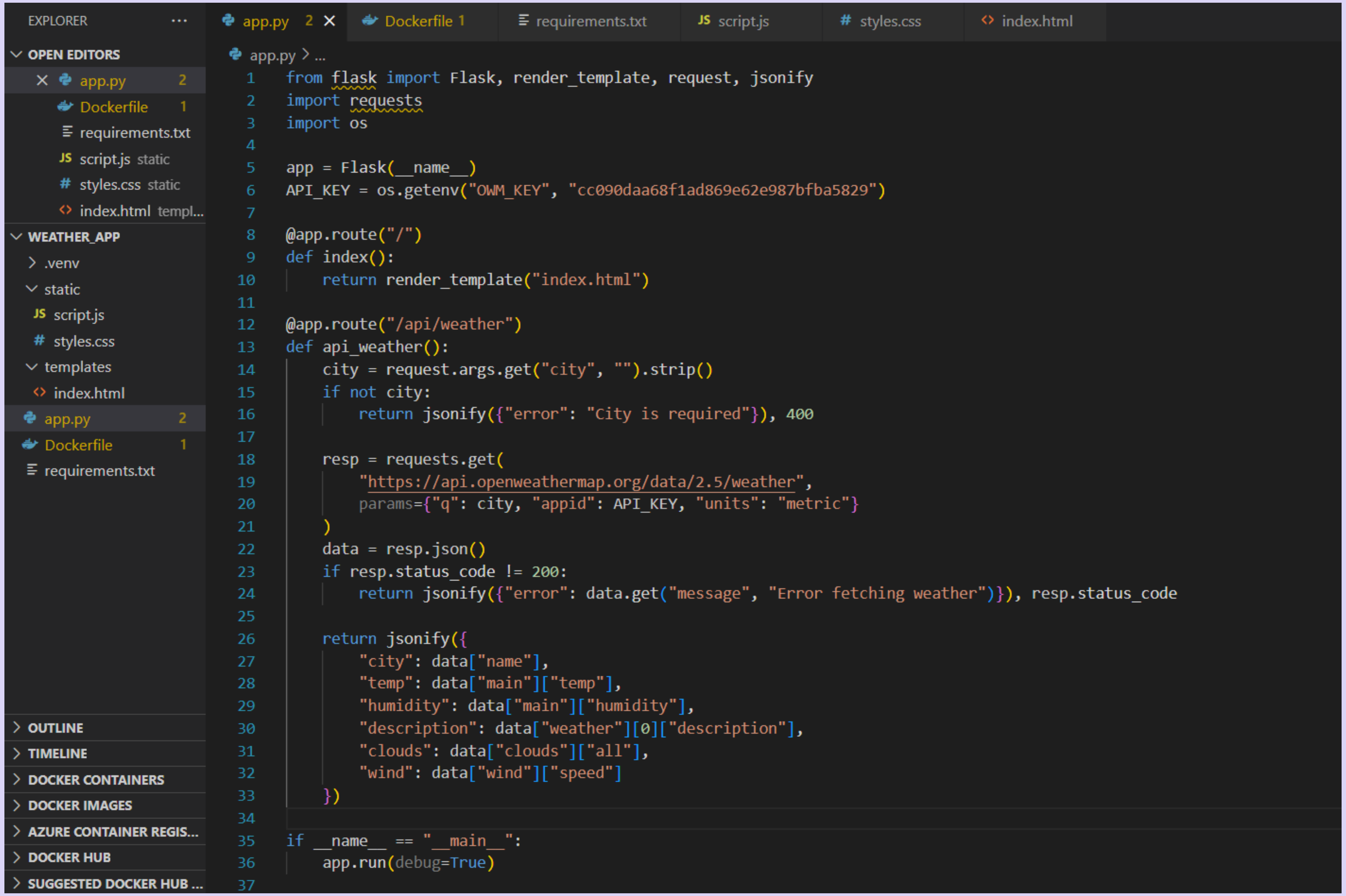
Backend logic and API integration for the web application.

Deployment Workflow

This diagram illustrates the sequential steps involved in deploying our Dockerised web application.



Deployment Steps

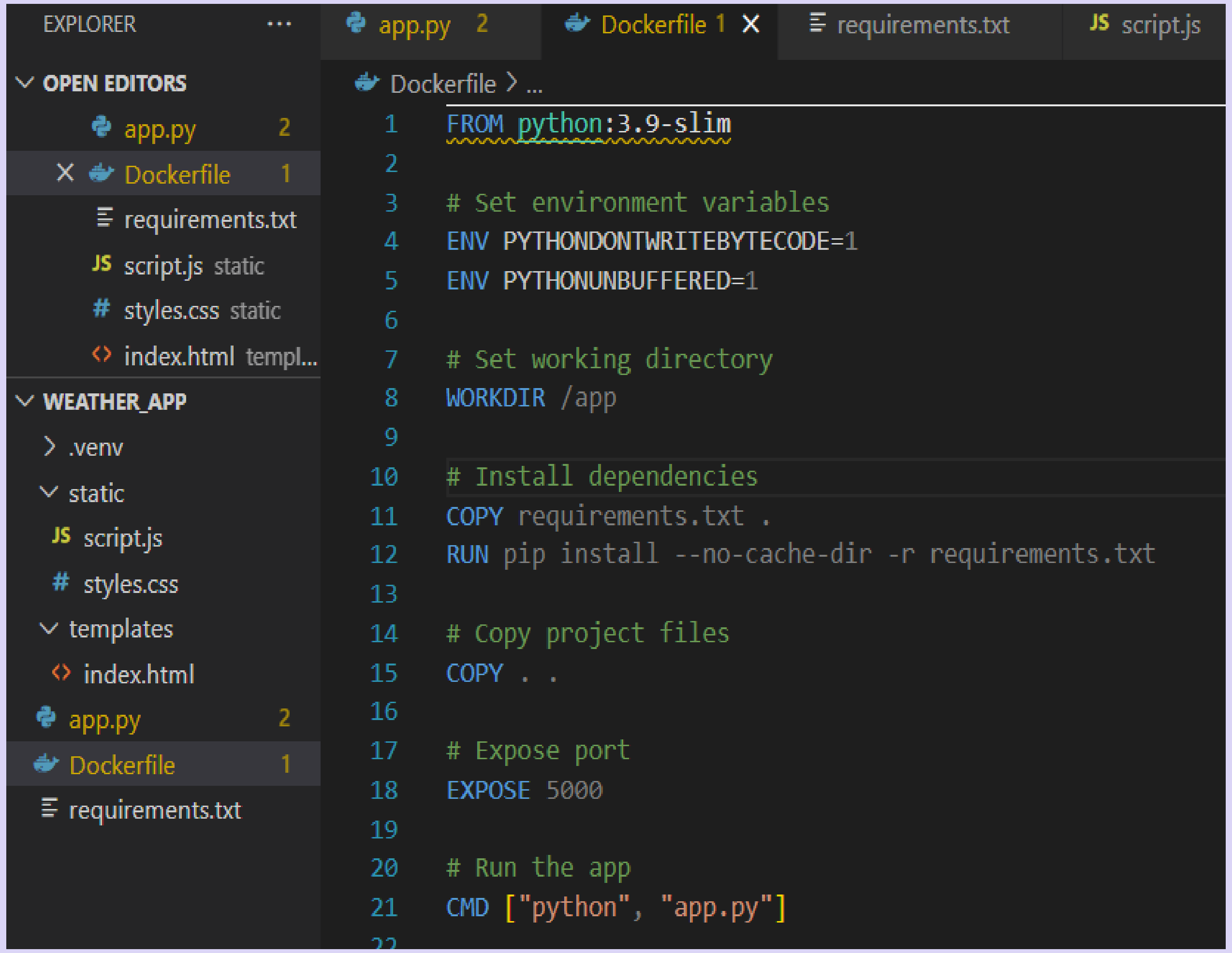


```python
from flask import Flask, render_template, request, jsonify
import requests
import os

app = Flask(__name__)
API_KEY = os.getenv("OWM_KEY", "cc090daa68f1ad869e62e987bfba5829")

@app.route("/")
def index():
    return render_template("index.html")

@app.route("/api/weather")
def api_weather():
    city = request.args.get("city", "").strip()
    if not city:
        return jsonify({"error": "City is required"}), 400

    resp = requests.get(
        "https://api.openweathermap.org/data/2.5/weather",
        params={"q": city, "appid": API_KEY, "units": "metric"}
    )
    data = resp.json()
    if resp.status_code != 200:
        return jsonify({"error": data.get("message", "Error fetching weather")}), resp.status_code

    return jsonify({
        "city": data["name"],
        "temp": data["main"]["temp"],
        "humidity": data["main"]["humidity"],
        "description": data["weather"][0]["description"],
        "clouds": data["clouds"]["all"],
        "wind": data["wind"]["speed"]
    })

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

```

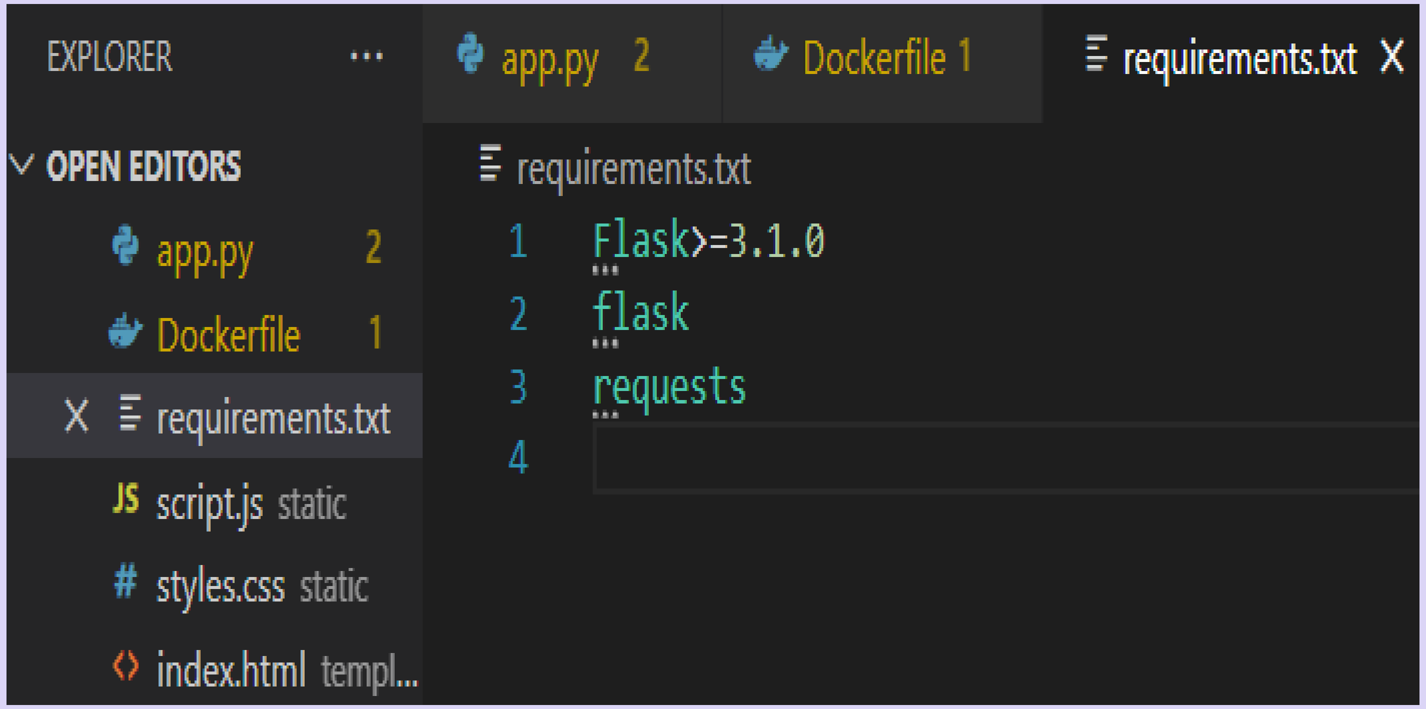
Create a New Folder named WEATHER_APP at Desktop and open folder in VS Code

Create a sub file of project name app.py

Deployment Steps



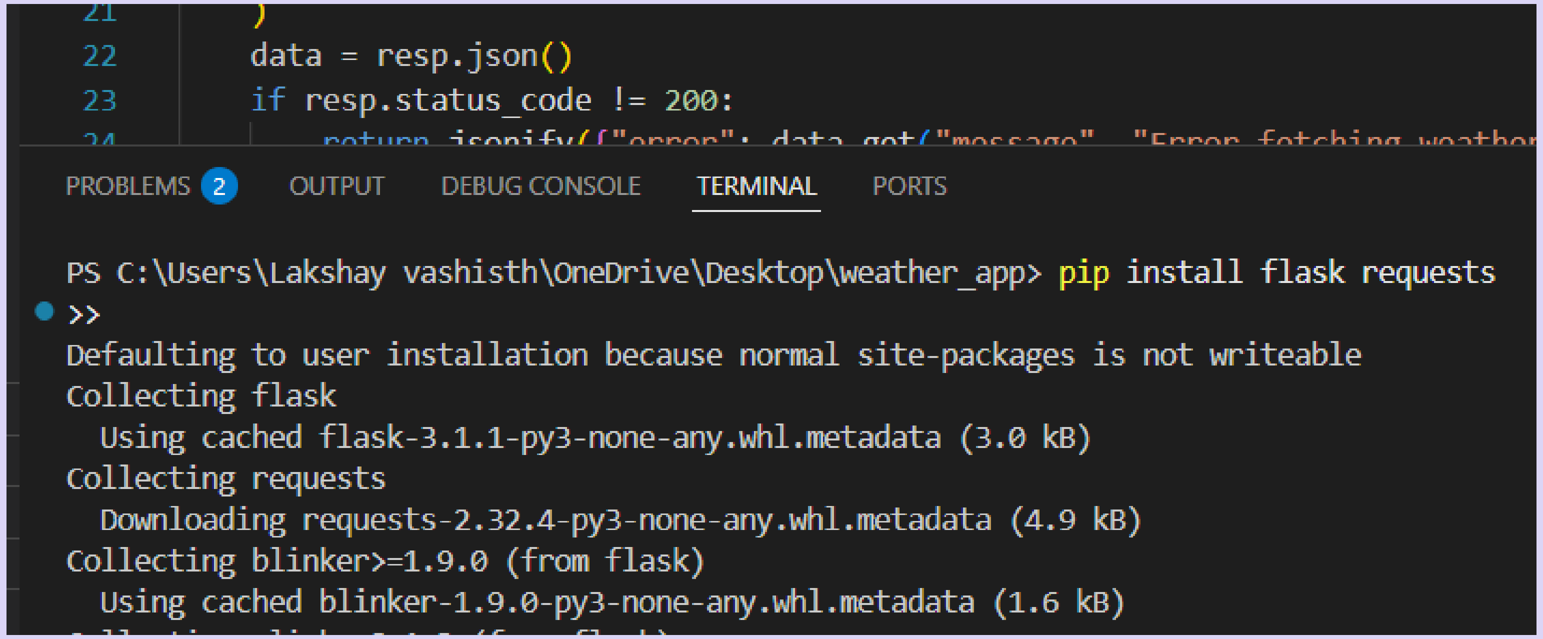
```dockerfile
FROM python:3.9-slim

# Set environment variables
ENV PYTHONDONTWRITEBYTECODE=1
ENV PYTHONUNBUFFERED=1

# Set working directory
WORKDIR /app

# Install dependencies
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# Copy project files
COPY . .

# Expose port
EXPOSE 5000

# Run the app
CMD ["python", "app.py"]
```



```
Flask>=3.1.0
flask
requests
```

Similarly create a sub file named as Dockerfile and requirements.txt with the above code related to the Project needs.

Deployment Steps

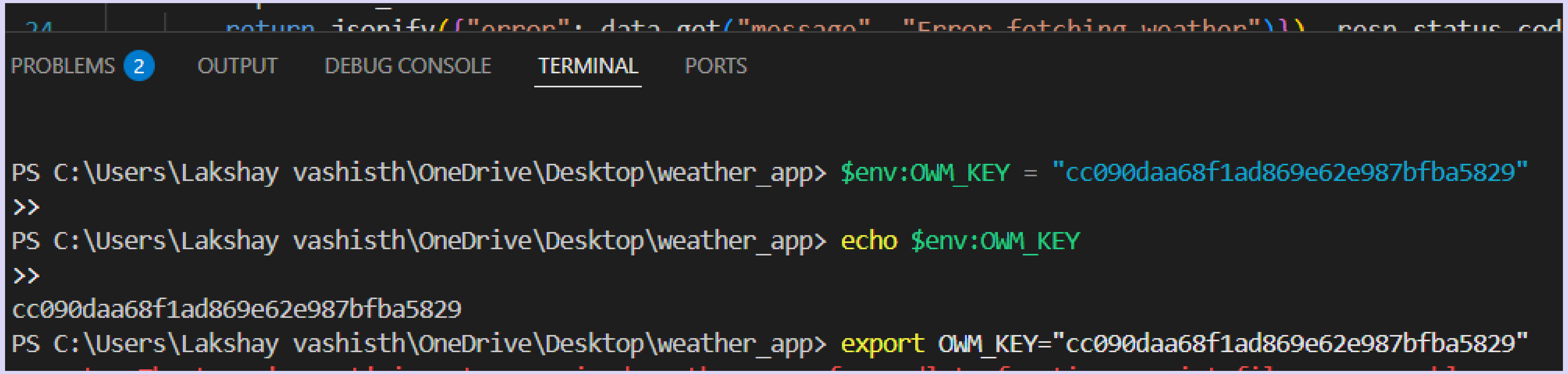


```
        )
        data = resp.json()
        if resp.status_code != 200:
            return jsonify({"error": data.get("message", "Error fetching weather
```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Lakshay vashisth\OneDrive\Desktop\weather_app> pip install flask requests
>>
Defaulting to user installation because normal site-packages is not writeable
Collecting flask
  Using cached flask-3.1.1-py3-none-any.whl.metadata (3.0 kB)
Collecting requests
  Downloading requests-2.32.4-py3-none-any.whl.metadata (4.9 kB)
Collecting blinker>=1.9.0 (from flask)
  Using cached blinker-1.9.0-py3-none-any.whl.metadata (1.6 kB)
```

Then Create a New Terminal in VS Code and Preform a Command for Installing PIP and Flask Modules.



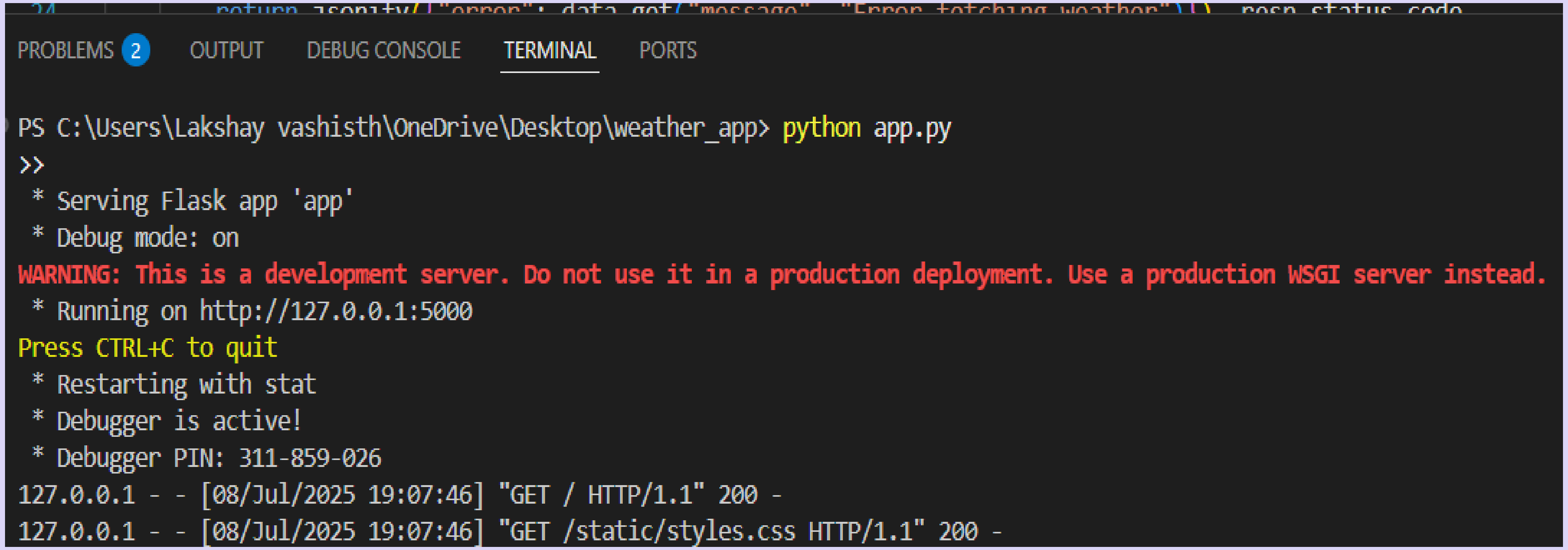
```
    return jsonify({"error": data.get("message", "Error fetching weather")}), resp.status_cod
```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Lakshay vashisth\OneDrive\Desktop\weather_app> $env:OWM_KEY = "cc090daa68f1ad869e62e987bfba5829"
>>
PS C:\Users\Lakshay vashisth\OneDrive\Desktop\weather_app> echo $env:OWM_KEY
>>
cc090daa68f1ad869e62e987bfba5829
PS C:\Users\Lakshay vashisth\OneDrive\Desktop\weather_app> export OWM_KEY="cc090daa68f1ad869e62e987bfba5829"
[illegible]
```

Then Give Command in Terminal for venu Environment and API key value in the VS Code.

Deployment Steps

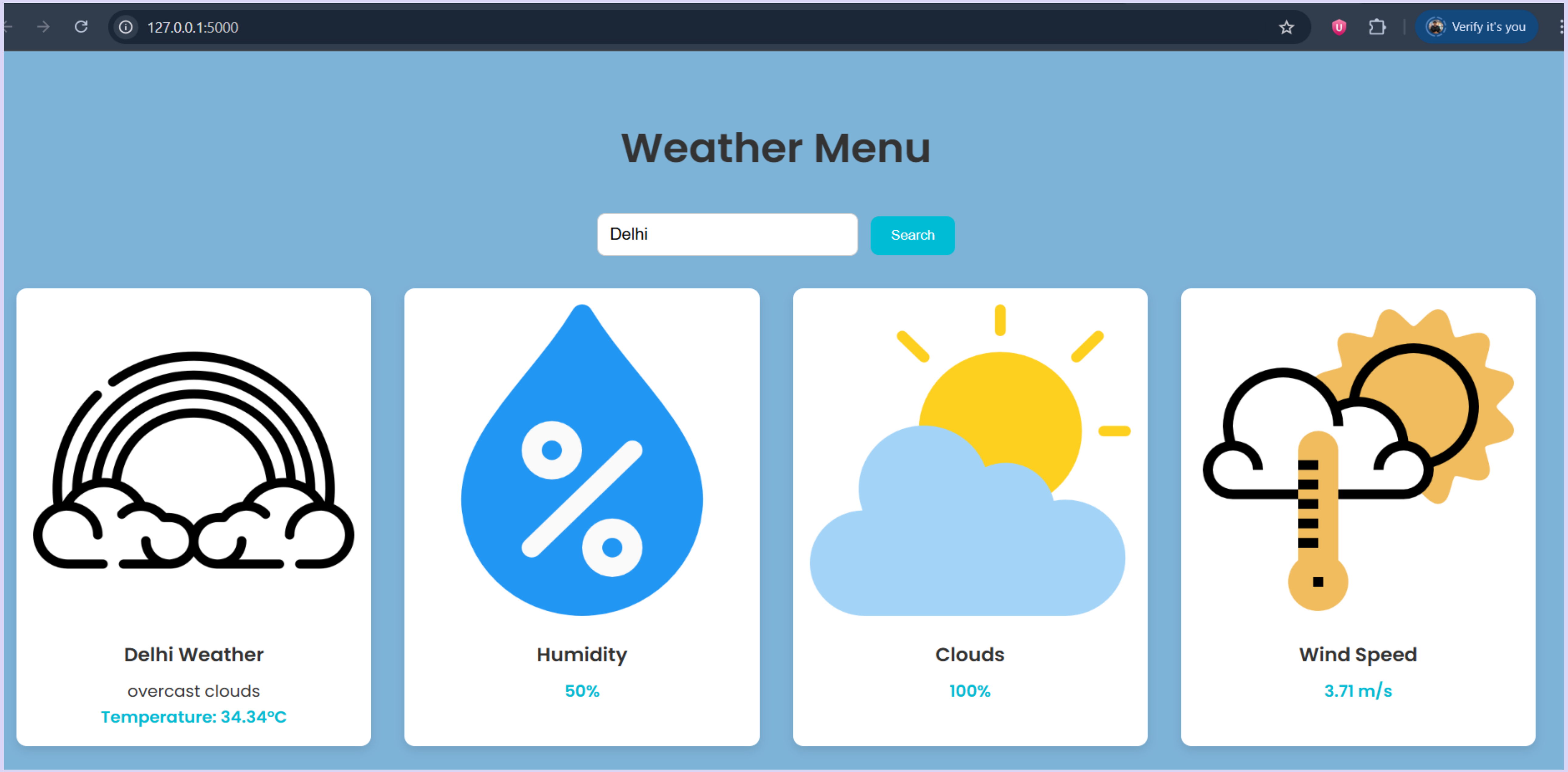


```
PS C:\Users\Lakshay vashisth\OneDrive\Desktop\weather_app> python app.py
>>
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 311-859-026
127.0.0.1 - - [08/Jul/2025 19:07:46] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [08/Jul/2025 19:07:46] "GET /static/styles.css HTTP/1.1" 200 -
```

Then Give Command Python app.py this will help to active the app code in the Localhost.

Deployment Steps

Now the Weather Web Application is running good at Localhost .


127.0.0.1:5000

Verify it's you

Weather Menu

Delhi

Search

Delhi Weather

overcast clouds

Temperature: 34.34°C

Humidity

50%

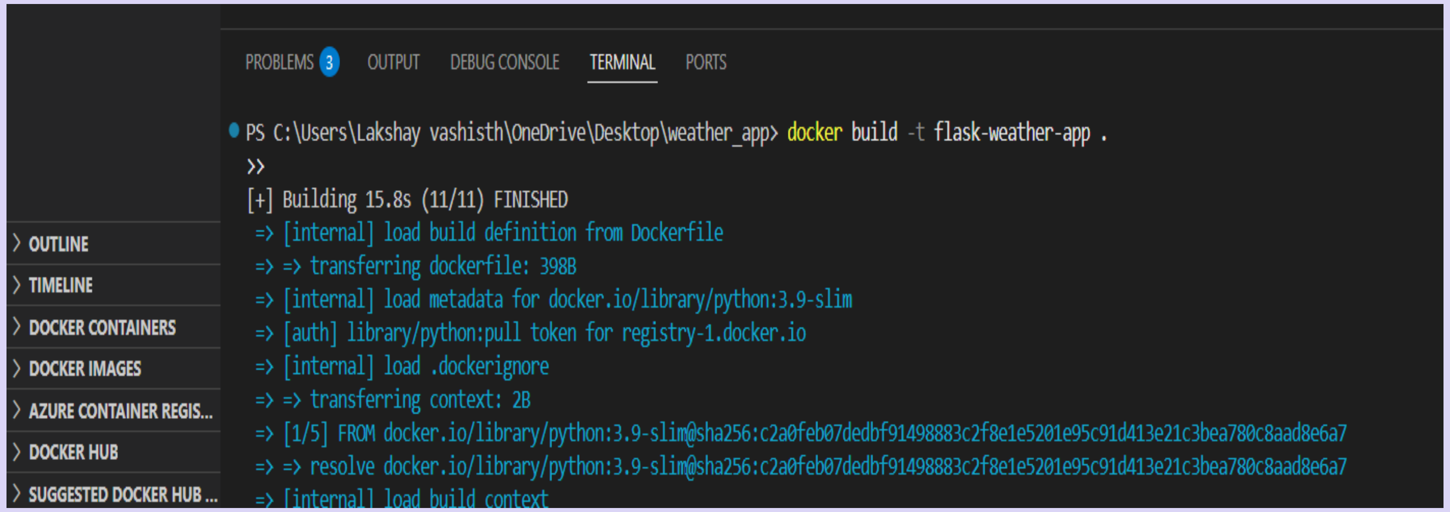
Clouds

100%

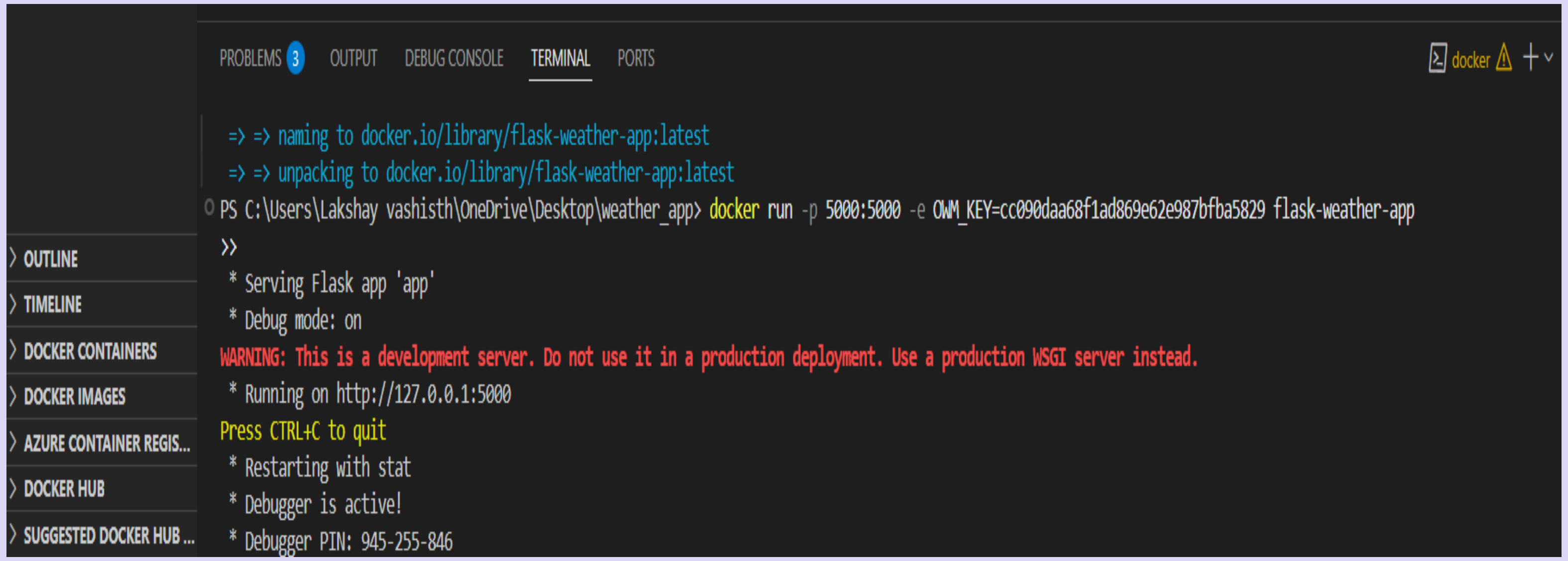
Wind Speed

3.71 m/s

Deployment Steps



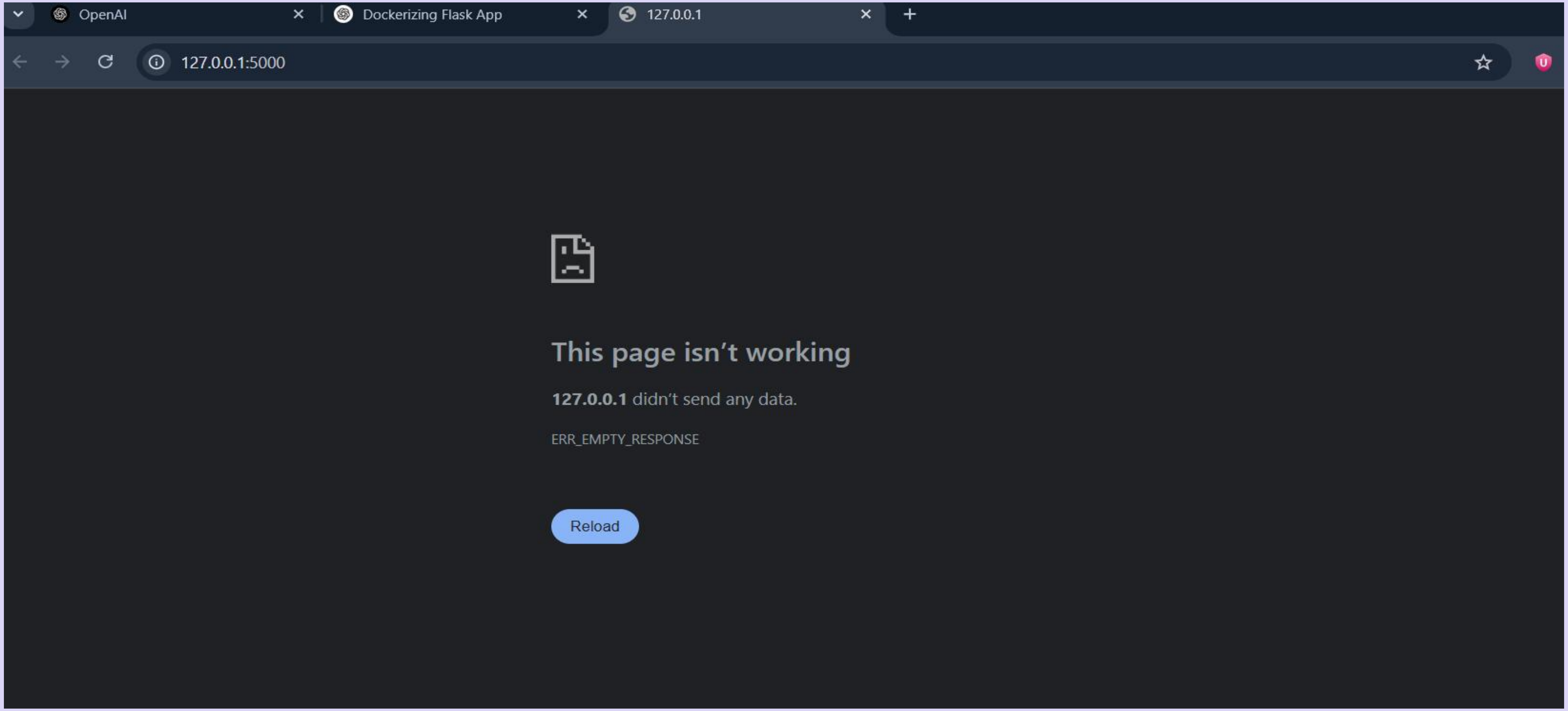
```
PS C:\Users\Lakshay vashisth\OneDrive\Desktop\weather_app> docker build -t flask-weather-app .
>>
[+] Building 15.8s (11/11) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 398B
 => [internal] load metadata for docker.io/library/python:3.9-slim
 => [auth] library/python:pull token for registry-1.docker.io
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/5] FROM docker.io/library/python:3.9-slim@sha256:c2a0feb07dedbf91498883c2f8e1e5201e95c91d413e21c3bea780c8aad8e6a7
 => => resolve docker.io/library/python:3.9-slim@sha256:c2a0feb07dedbf91498883c2f8e1e5201e95c91d413e21c3bea780c8aad8e6a7
 => [internal] load build context
```



```
 => => naming to docker.io/library/flask-weather-app:latest
 => => unpacking to docker.io/library/flask-weather-app:latest
PS C:\Users\Lakshay vashisth\OneDrive\Desktop\weather_app> docker run -p 5000:5000 -e OWM_KEY=cc090daa68f1ad869e62e987bfba5829 flask-weather-app
>>
 * Serving Flask app 'app'
 * 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 http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 945-255-846
```

Here we give "docker build –t flask-weather-app ." and "docker run –p 5000:5000 –e OWN_KEY=YOUR_API_KEY" Command in Terminal Successfully to dockerised.

Deployment Steps



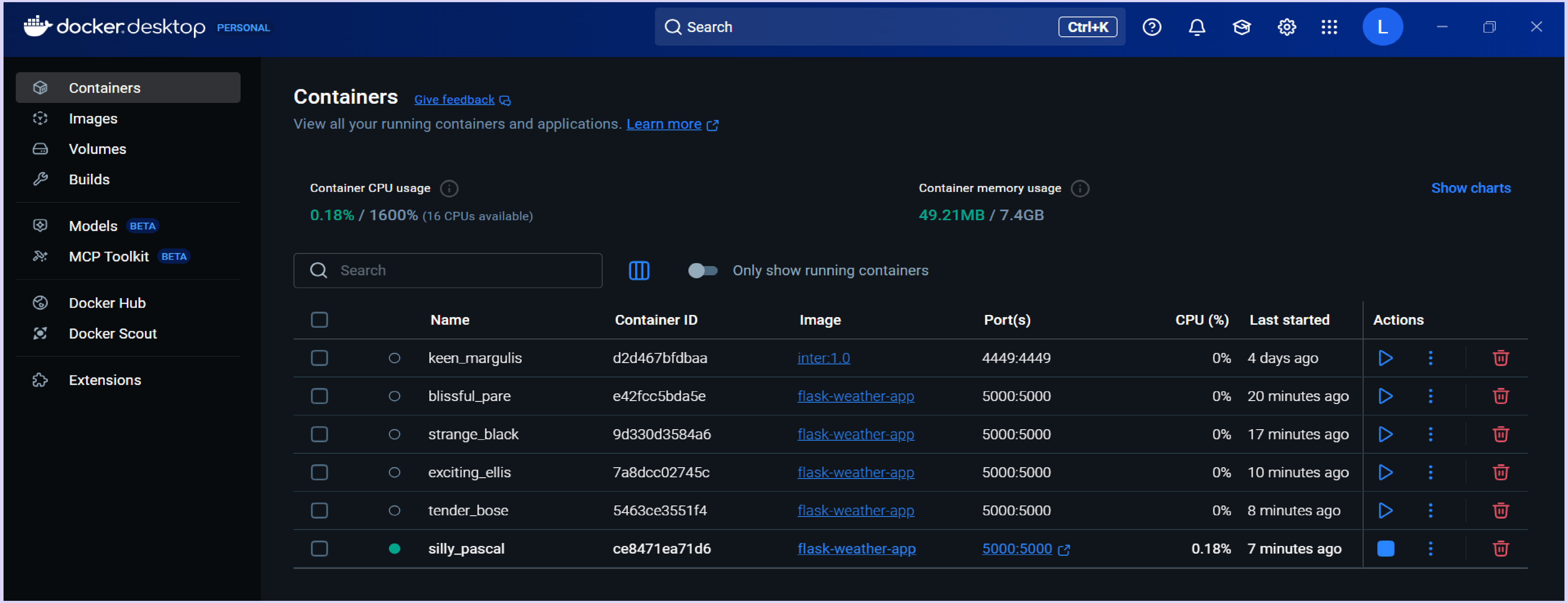
This page isn't working

127.0.0.1 didn't send any data.

ERR_EMPTY_RESPONSE

This is Localhost where we reached by ctrl + click on URL.

Deployment Steps



Containers Give feedback

View all your running containers and applications. Learn more

Container CPU usage
0.18% / 1600% (16 CPUs available)

Container memory usage
49.21MB / 7.4GB

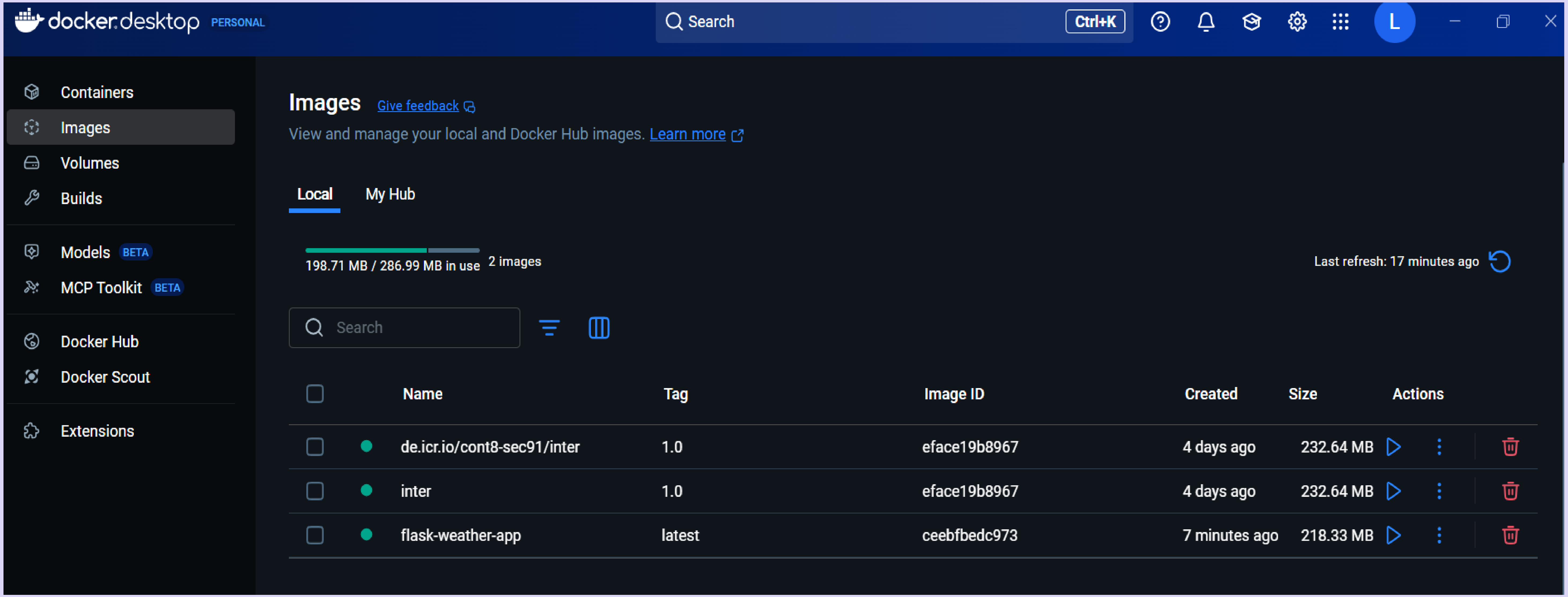
Show charts

Search   Only show running containers

	Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
○	keen_margulis	d2d467bfdbaa	inter:1.0	4449:4449	0%	4 days ago	
○	blissful_pare	e42fcc5bda5e	flask-weather-app	5000:5000	0%	20 minutes ago	
○	strange_black	9d330d3584a6	flask-weather-app	5000:5000	0%	17 minutes ago	
○	exciting_ellis	7a8dcc02745c	flask-weather-app	5000:5000	0%	10 minutes ago	
○	tender_bose	5463ce3551f4	flask-weather-app	5000:5000	0%	8 minutes ago	
●	silly_pascal	ce8471ea71d6	flask-weather-app	5000:5000	0.18%	7 minutes ago	

Here we open Docker Desktop app which was running at the back during the whole process. We can see clearly that our Container is form.

Deployment Steps



Images Give feedback

View and manage your local and Docker Hub images. Learn more

Local My Hub

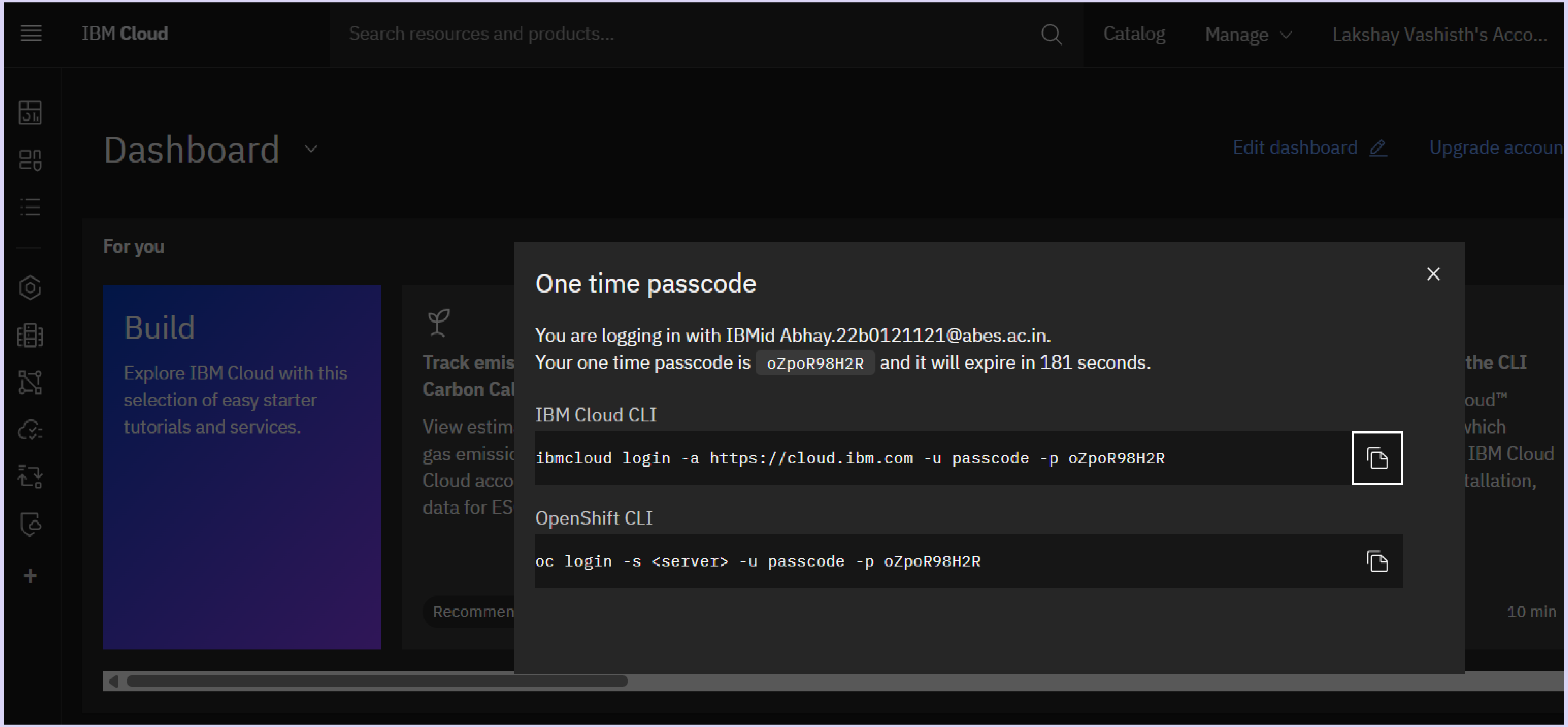
198.71 MB / 286.99 MB in use 2 images

Last refresh: 17 minutes ago

Name	Tag	Image ID	Created	Size	Actions
de.icr.io/cont8-sec91/inter	1.0	eface19b8967	4 days ago	232.64 MB	
inter	1.0	eface19b8967	4 days ago	232.64 MB	
flask-weather-app	latest	ceebfbedc973	7 minutes ago	218.33 MB	

This is Docker Desktop App Images Sections where the images are also formed. Which mean our code is converted into images.

Deployment Steps



One time passcode

You are logging in with IBMid Abhay.22b0121121@abes.ac.in.
Your one time passcode is `oZpoR98H2R` and it will expire in 181 seconds.

IBM Cloud CLI

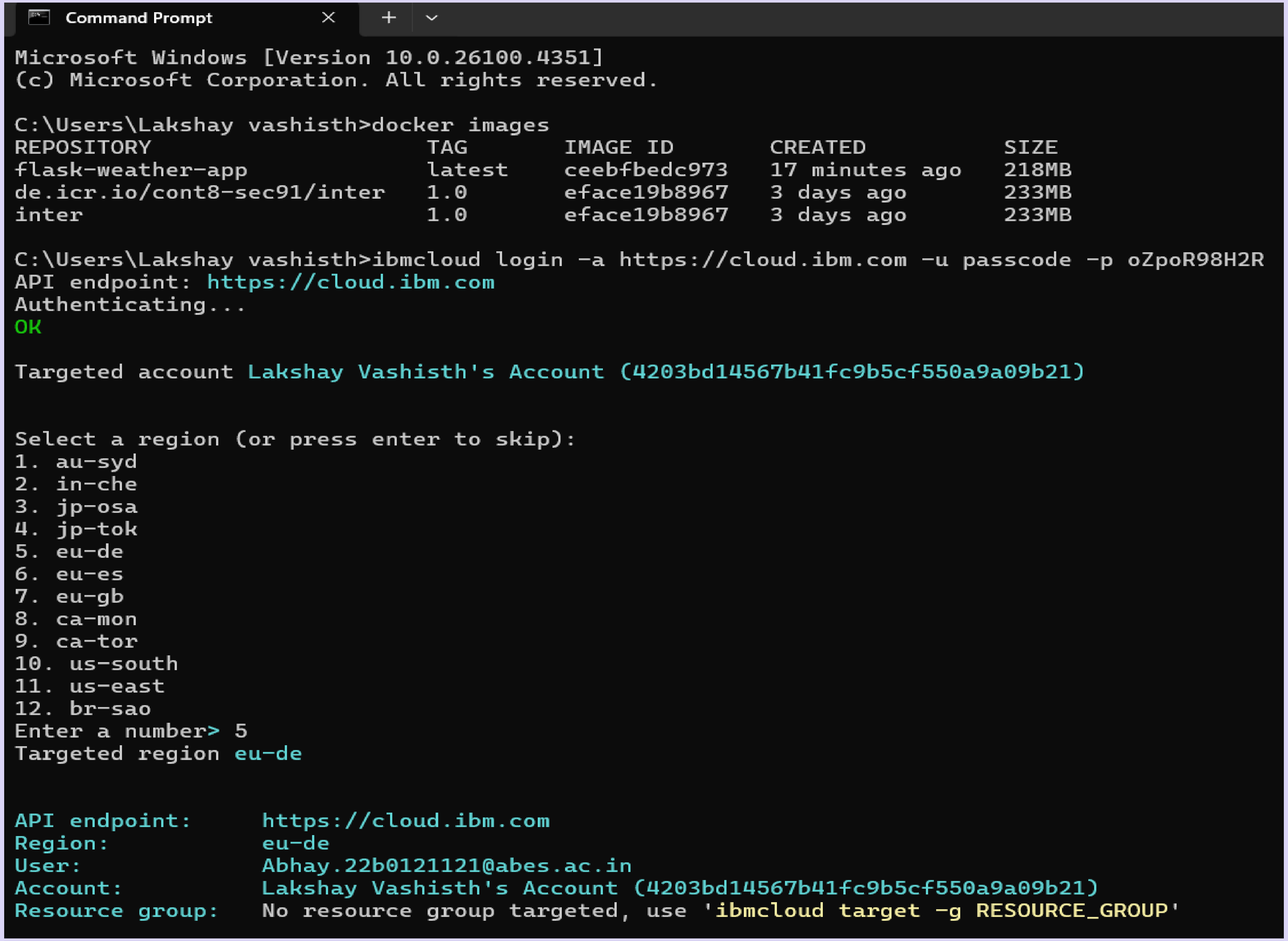
```
ibmcloud login -a https://cloud.ibm.com -u passcode -p oZpoR98H2R
```

OpenShift CLI

```
oc login -s <server> -u passcode -p oZpoR98H2R
```

Now open the IBM CLOUD APP and LOG-IN it . Then go to profile section and copy the IBM Cloud CLI.

Deployment Steps

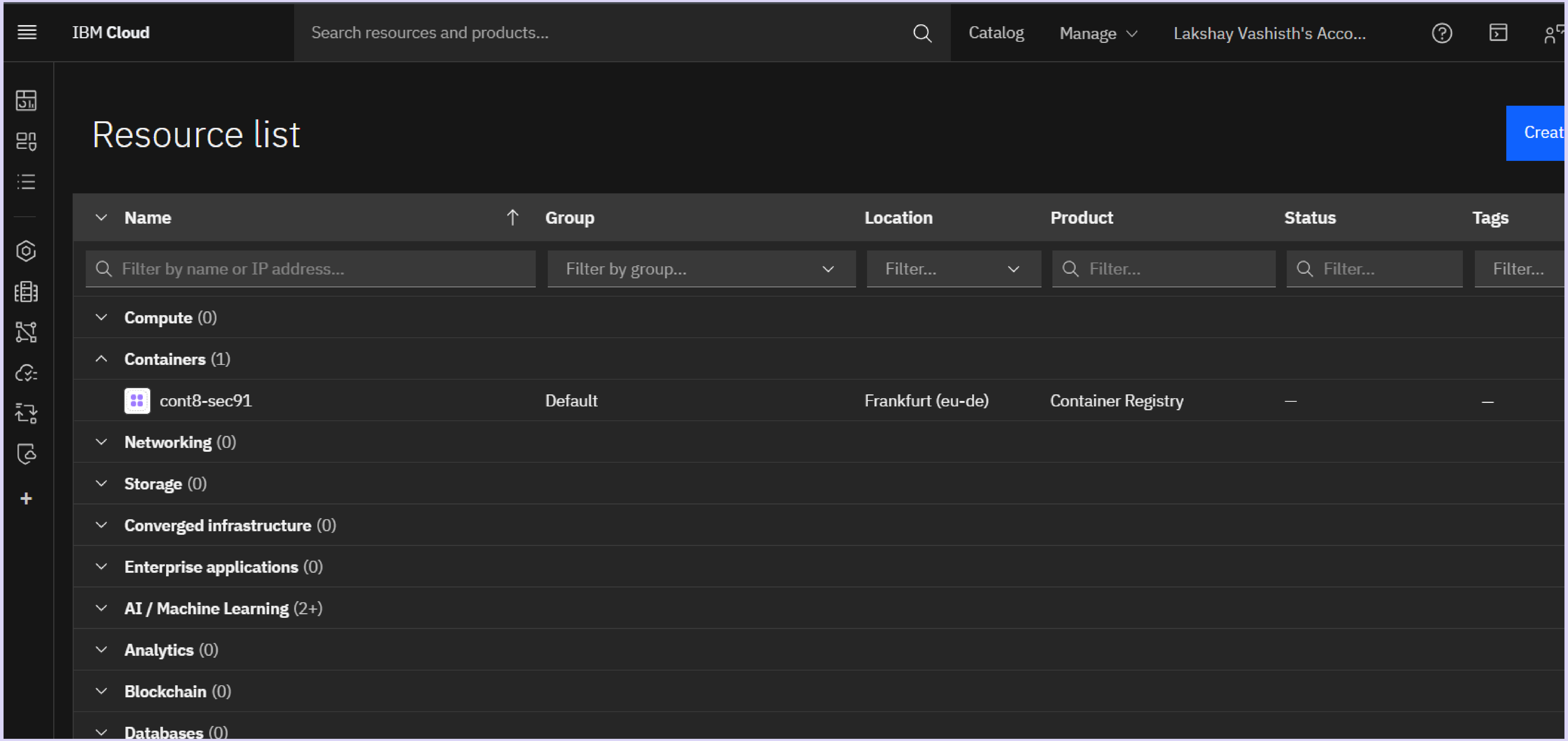


```
Microsoft Windows [Version 10.0.26100.4351]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Lakshay vashisth>docker images
REPOSITORY                    TAG       IMAGE ID       CREATED          SIZE
flask-weather-app             latest    ceebfbedc973   17 minutes ago   218MB
de.icr.io/cont8-sec91/inter   1.0       eface19b8967   3 days ago       233MB
inter                         1.0       eface19b8967   3 days ago       233MB

C:\Users\Lakshay vashisth>ibmcloud login -a https://cloud.ibm.com -u passcode -p oZpoR98H2R
API endpoint: https://cloud.ibm.com
Authenticating...
OK

Targeted account Lakshay Vashisth's Account (4203bd14567b41fc9b5cf550a9a09b21)


Select a region (or press enter to skip):
1. au-syd
2. in-che
3. jp-osa
4. jp-tok
5. eu-de
6. eu-es
7. eu-gb
8. ca-mon
9. ca-tor
10. us-south
11. us-east
12. br-sao
Enter a number> 5
Targeted region eu-de


API endpoint:      https://cloud.ibm.com
Region:            eu-de
User:              Abhay.22b0121121@abes.ac.in
Account:           Lakshay Vashisth's Account (4203bd14567b41fc9b5cf550a9a09b21)
Resource group:    No resource group targeted, use 'ibmcloud target -g RESOURCE_GROUP'
```

Now open the Command Prompt in Desktop and give the command as mention in the following image.

Like docker images, the IBM Cloud CLI to set the region.

Deployment Steps



IBM Cloud

Search resources and products...

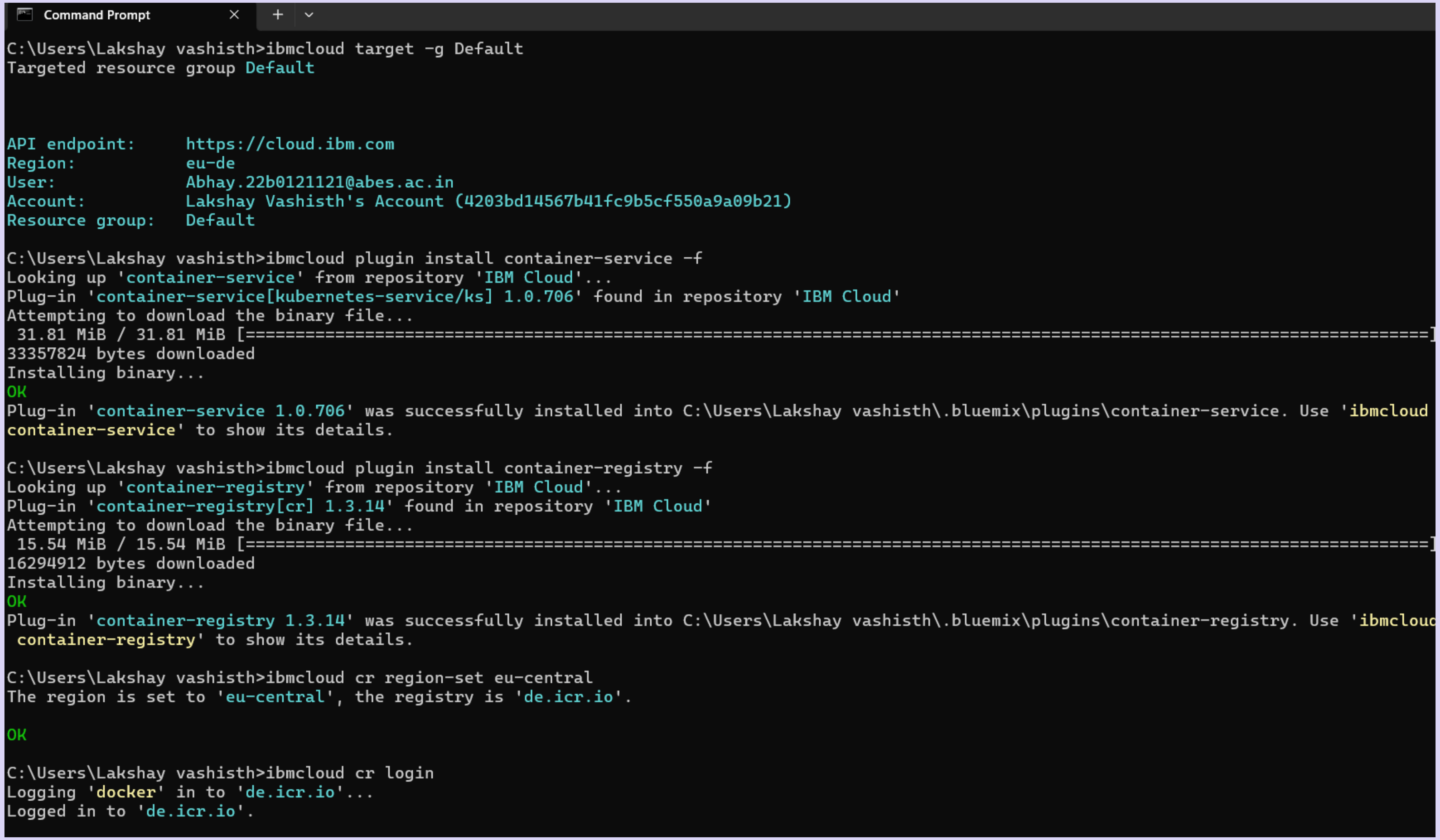
Catalog Manage Lakshay Vashisth's Acco...

Resource list

Creat

Name	Group	Location	Product	Status	Tags
Filter by name or IP address...	Filter by group...	Filter...	Filter...	Filter...	Filter...
Compute (0)					
Containers (1)					
cont8-sec91	Default	Frankfurt (eu-de)	Container Registry	–	–
Networking (0)					
Storage (0)					
Converged infrastructure (0)					
Enterprise applications (0)					
AI / Machine Learning (2+)					
Analytics (0)					
Blockchain (0)					
Databases (0)					

Deployment Steps

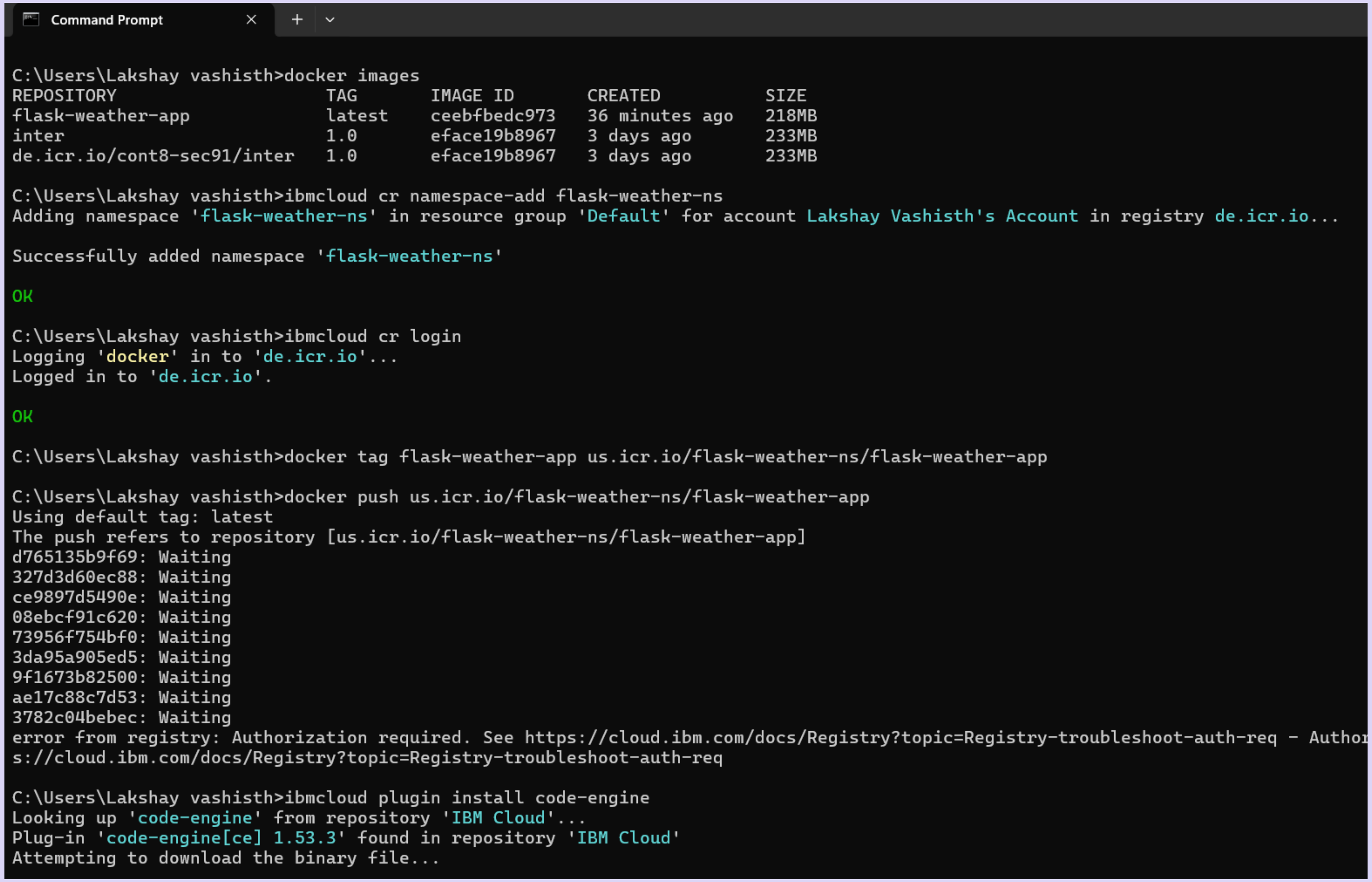


```
C:\Users\Lakshay vashisth>ibmcloud target -g Default
Targeted resource group Default



API endpoint:     https://cloud.ibm.com
Region:           eu-de
User:             Abhay.22b0121121@abes.ac.in
Account:          Lakshay Vashisth's Account (4203bd14567b41fc9b5cf550a9a09b21)
Resource group:   Default

C:\Users\Lakshay vashisth>ibmcloud plugin install container-service -f
Looking up 'container-service' from repository 'IBM Cloud'...
Plug-in 'container-service[kubernetes-service/ks] 1.0.706' found in repository 'IBM Cloud'
Attempting to download the binary file...
 31.81 MiB / 31.81 MiB [===================================================================================================]
33357824 bytes downloaded
Installing binary...
OK
Plug-in 'container-service 1.0.706' was successfully installed into C:\Users\Lakshay vashisth\.bluemix\plugins\container-service. Use 'ibmcloud
container-service' to show its details.

C:\Users\Lakshay vashisth>ibmcloud plugin install container-registry -f
Looking up 'container-registry' from repository 'IBM Cloud'...
Plug-in 'container-registry[cr] 1.3.14' found in repository 'IBM Cloud'
Attempting to download the binary file...
 15.54 MiB / 15.54 MiB [===================================================================================================]
16294912 bytes downloaded
Installing binary...
OK
Plug-in 'container-registry 1.3.14' was successfully installed into C:\Users\Lakshay vashisth\.bluemix\plugins\container-registry. Use 'ibmclouc
 container-registry' to show its details.

C:\Users\Lakshay vashisth>ibmcloud cr region-set eu-central
The region is set to 'eu-central', the registry is 'de.icr.io'.

OK

C:\Users\Lakshay vashisth>ibmcloud cr login
Logging 'docker' in to 'de.icr.io'...
Logged in to 'de.icr.io'.
```

Then we preform the following command in Command prompt likes :-

- *Ibmcloud target –g Default*
- *Ibmcloud plugin install container-service –f*
- *Ibmcloud plugin install container-registry –f*
- *Ibmcloud cr region-set eu-central*
- *Ibmcloud cr login*

Deployment Steps



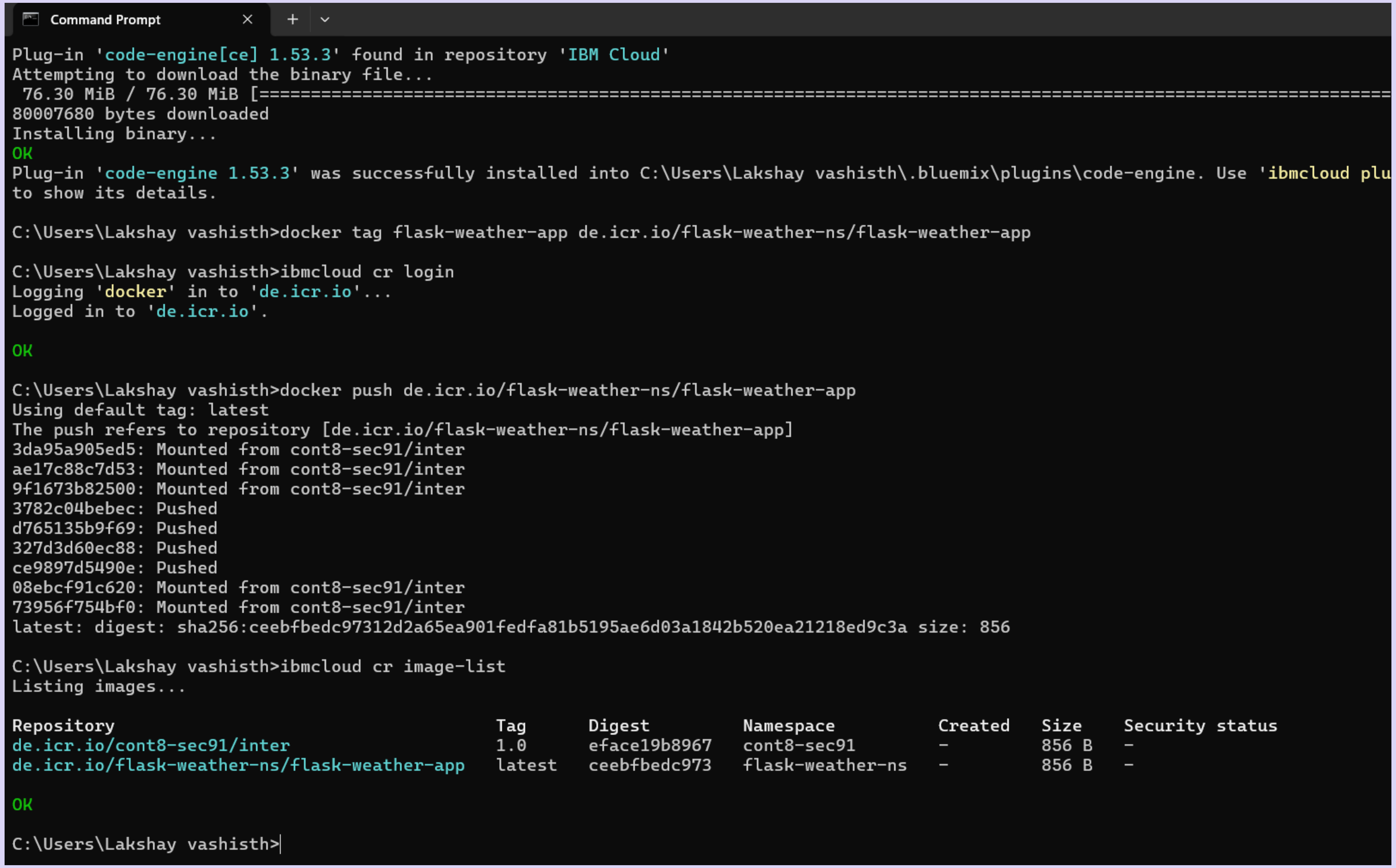
Command Prompt

```
C:\Users\Lakshay vashisth>docker images
REPOSITORY                    TAG       IMAGE ID       CREATED          SIZE
flask-weather-app             latest    ceebfbedc973   36 minutes ago   218MB
inter                         1.0       eface19b8967   3 days ago       233MB
de.icr.io/cont8-sec91/inter   1.0       eface19b8967   3 days ago       233MB

C:\Users\Lakshay vashisth>ibmcloud cr namespace-add flask-weather-ns
Adding namespace 'flask-weather-ns' in resource group 'Default' for account Lakshay Vashisth's Account in registry de.icr.io...

Successfully added namespace 'flask-weather-ns'

OK

C:\Users\Lakshay vashisth>ibmcloud cr login
Logging 'docker' in to 'de.icr.io'...
Logged in to 'de.icr.io'.

OK

C:\Users\Lakshay vashisth>docker tag flask-weather-app us.icr.io/flask-weather-ns/flask-weather-app

C:\Users\Lakshay vashisth>docker push us.icr.io/flask-weather-ns/flask-weather-app
Using default tag: latest
The push refers to repository [us.icr.io/flask-weather-ns/flask-weather-app]
d765135b9f69: Waiting
327d3d60ec88: Waiting
ce9897d5490e: Waiting
08ebcf91c620: Waiting
73956f754bf0: Waiting
3da95a905ed5: Waiting
9f1673b82500: Waiting
ae17c88c7d53: Waiting
3782c04bebec: Waiting
error from registry: Authorization required. See https://cloud.ibm.com/docs/Registry?topic=Registry-troubleshoot-auth-req - Author
s://cloud.ibm.com/docs/Registry?topic=Registry-troubleshoot-auth-req

C:\Users\Lakshay vashisth>ibmcloud plugin install code-engine
Looking up 'code-engine' from repository 'IBM Cloud'...
Plug-in 'code-engine[ce] 1.53.3' found in repository 'IBM Cloud'
Attempting to download the binary file...
```

Then we preform the following command in Command prompt likes :-

- ❖ Docker images
- ❖ Ibmcloud cr namespace-add flask-weather-ns
- ❖ Ibmcloud cr login
- ❖ Docker tag flask-weather-app us.icr.io/flask-weather-ns/flask-weather-app

Deployment Steps

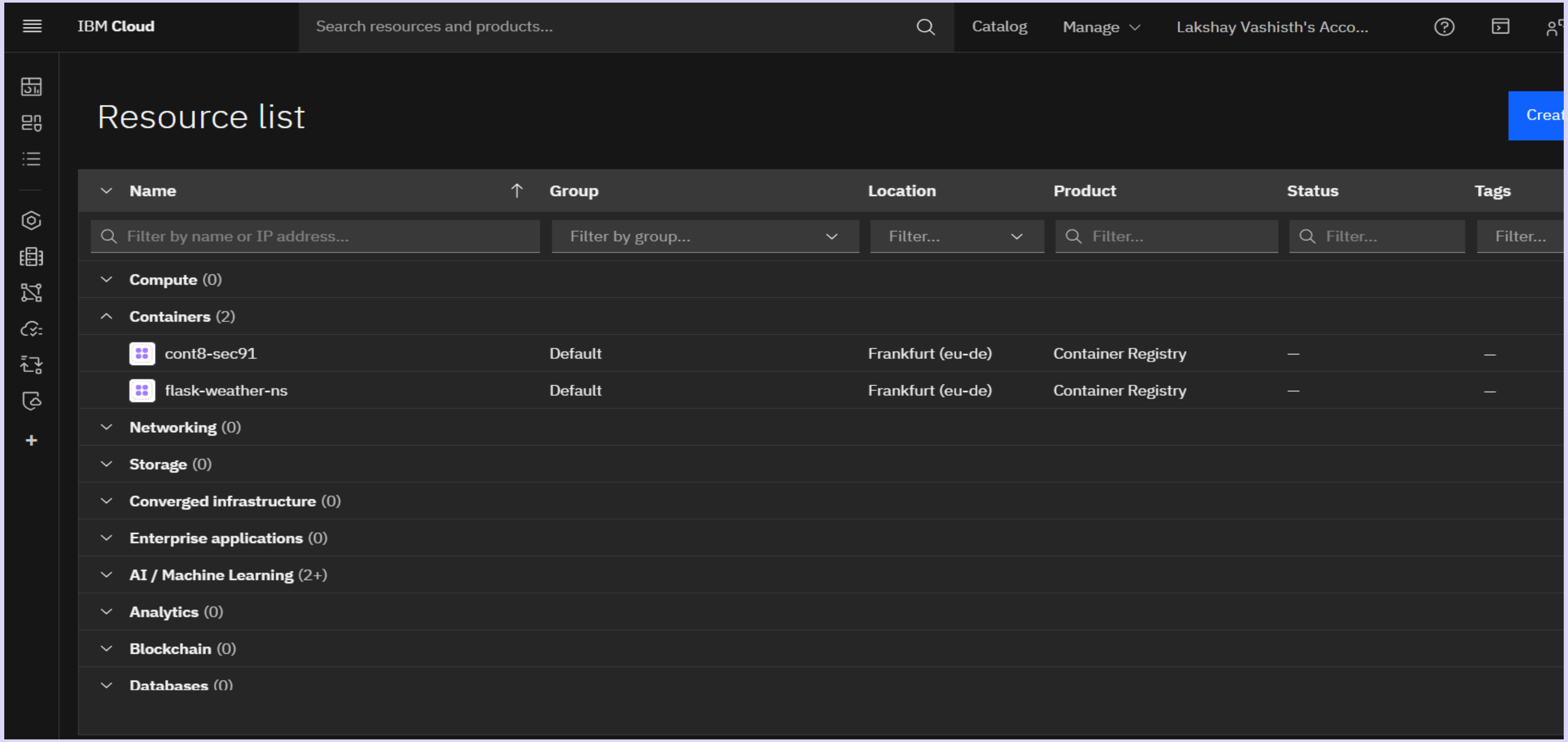


```
Plug-in 'code-engine[ce] 1.53.3' found in repository 'IBM Cloud'
Attempting to download the binary file...
 76.30 MiB / 76.30 MiB [==========================================================================================
80007680 bytes downloaded
Installing binary...
OK
Plug-in 'code-engine 1.53.3' was successfully installed into C:\Users\Lakshay vashisth\.bluemix\plugins\code-engine. Use 'ibmcloud plu
to show its details.

C:\Users\Lakshay vashisth>docker tag flask-weather-app de.icr.io/flask-weather-ns/flask-weather-app

C:\Users\Lakshay vashisth>ibmcloud cr login
Logging 'docker' in to 'de.icr.io'...
Logged in to 'de.icr.io'.

OK

C:\Users\Lakshay vashisth>docker push de.icr.io/flask-weather-ns/flask-weather-app
Using default tag: latest
The push refers to repository [de.icr.io/flask-weather-ns/flask-weather-app]
3da95a905ed5: Mounted from cont8-sec91/inter
ae17c88c7d53: Mounted from cont8-sec91/inter
9f1673b82500: Mounted from cont8-sec91/inter
3782c04bebec: Pushed
d765135b9f69: Pushed
327d3d60ec88: Pushed
ce9897d5490e: Pushed
08ebcf91c620: Mounted from cont8-sec91/inter
73956f754bf0: Mounted from cont8-sec91/inter
latest: digest: sha256:ceebfbedc97312d2a65ea901fedfa81b5195ae6d03a1842b520ea21218ed9c3a size: 856

C:\Users\Lakshay vashisth>ibmcloud cr image-list
Listing images...

Repository                                     Tag      Digest         Namespace         Created   Size    Security status
de.icr.io/cont8-sec91/inter                    1.0      eface19b8967   cont8-sec91       -         856 B   -
de.icr.io/flask-weather-ns/flask-weather-app   latest   ceebfbedc973   flask-weather-ns  -         856 B   -

OK

C:\Users\Lakshay vashisth>
```

Then we preform the following command in Command prompt likes :-

- ❖ *Docker tag flask-weather-app us.icr.io/flask-weather-ns/flask-weather-app*
- ❖ *Ibmcloud cr login*
- ❖ *Docker push de.icr.io/flask-weather-ns/flask-weather-app*
- ❖ *Ibmcloud cr image-list*

Deployment Steps

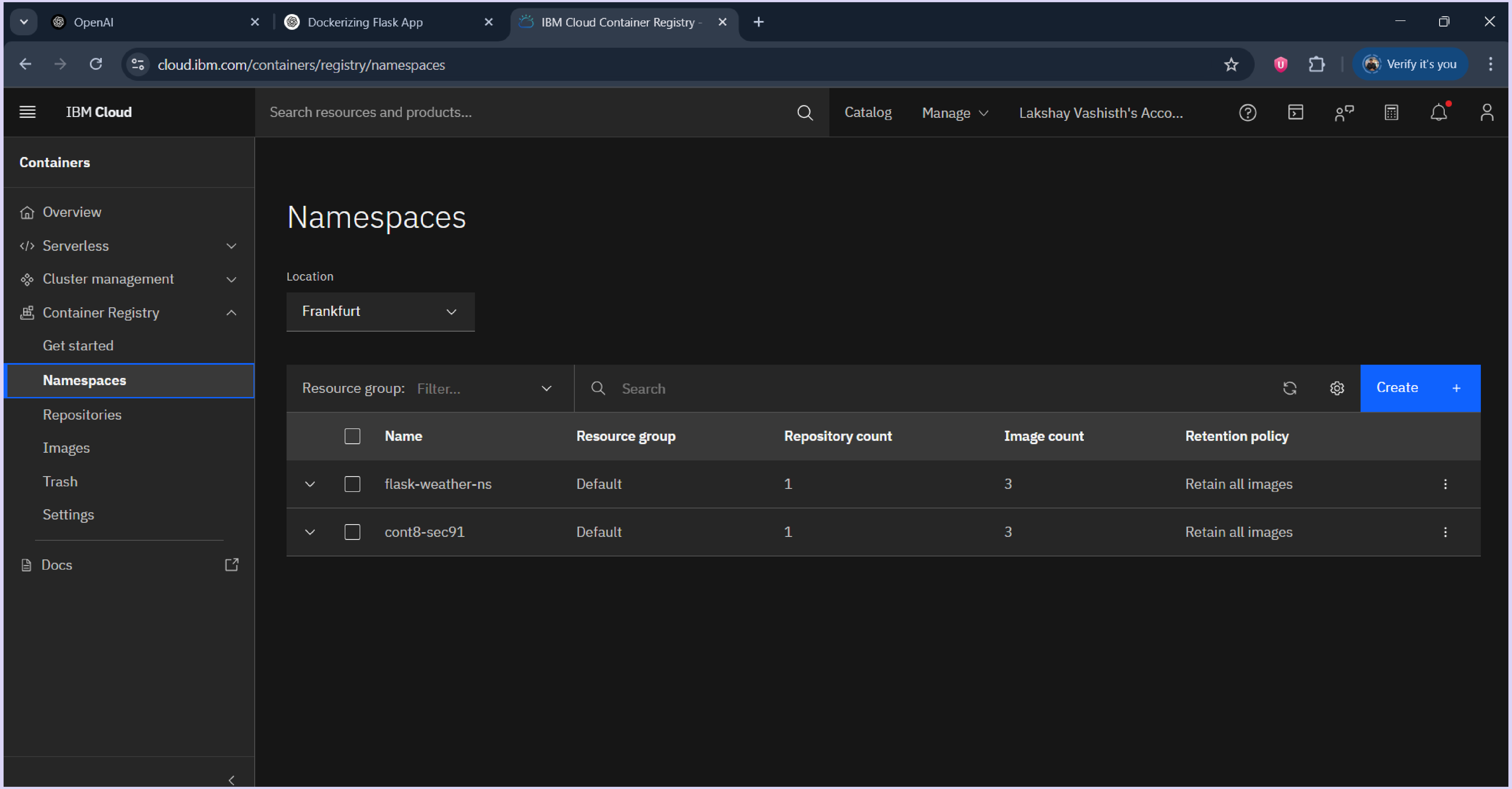


Resource list

Name	Group	Location	Product	Status	Tags
Filter by name or IP address...	Filter by group...	Filter...	Filter...	Filter...	Filter...
Compute (0)					
Containers (2)					
cont8-sec91	Default	Frankfurt (eu-de)	Container Registry	–	–
flask-weather-ns	Default	Frankfurt (eu-de)	Container Registry	–	–
Networking (0)					
Storage (0)					
Converged infrastructure (0)					
Enterprise applications (0)					
AI / Machine Learning (2+)					
Analytics (0)					
Blockchain (0)					
Databases (0)					

Now we came back to the IBM CLOUD APP → Resource list → Container . This clearly shows that our container is deployed in the IBM Cloud.

Deployment Steps



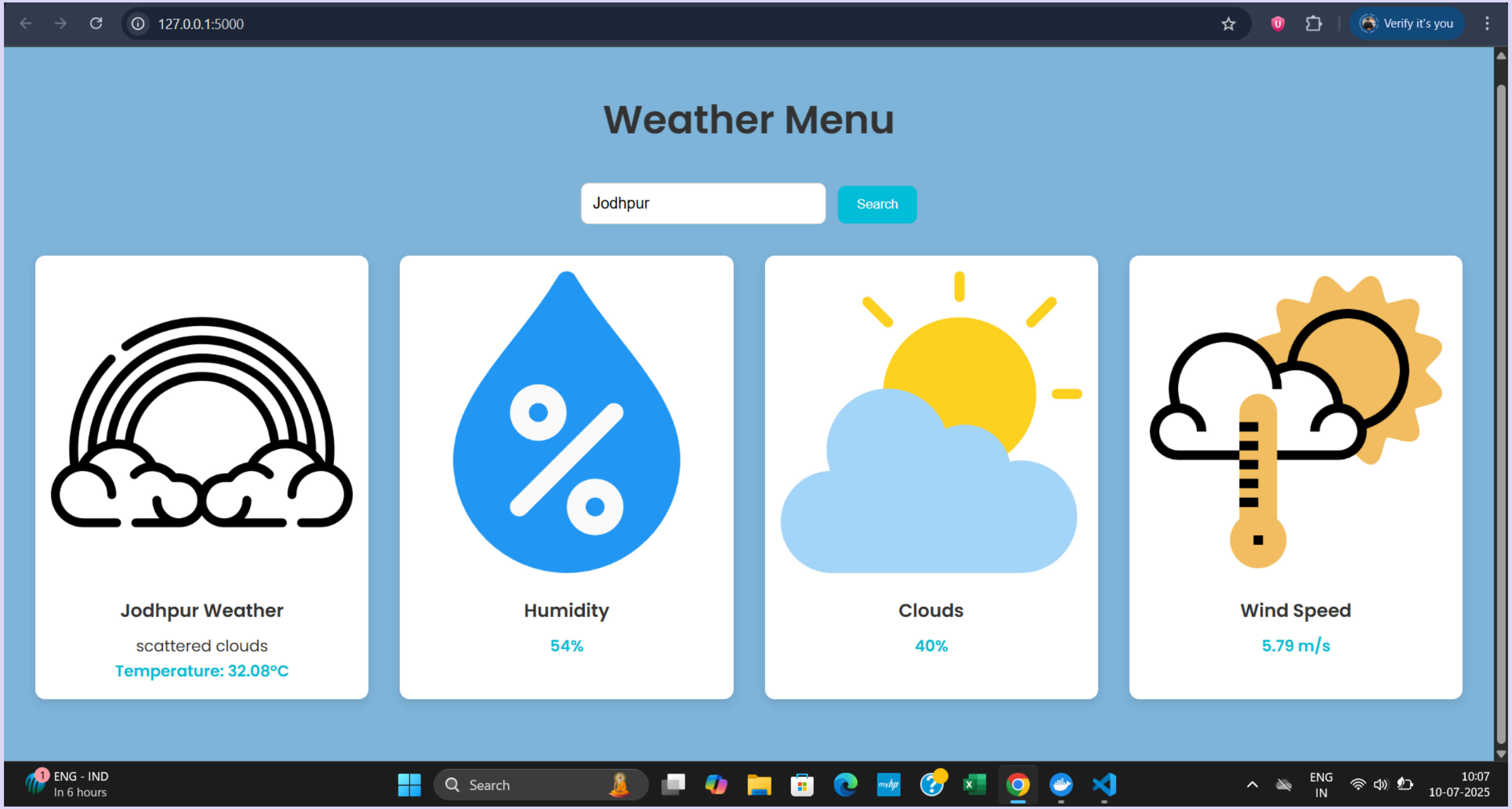
Namespaces

Location: Frankfurt

Resource group: Filter...

Name	Resource group	Repository count	Image count	Retention policy
flask-weather-ns	Default	1	3	Retain all images
cont8-sec91	Default	1	3	Retain all images

Deployment Steps



127.0.0.1:5000

Weather Menu

Jodhpur

Jodhpur Weather
scattered clouds
Temperature: 32.08°C

Humidity
54%

Clouds
40%

Wind Speed
5.79 m/s

The Final Output Display of the Project at Losthost.

Advantages of Dockerisation

Portability

Run consistently across any environment (development, test, production).

Isolation

Applications and dependencies are bundled, preventing conflicts.

Efficiency

Faster startup times and reduced resource consumption.

Scalability

Easily scale applications by spinning up more containers.

Version Control

Container images can be versioned and managed like code.

Key Features of IBM Cloud

Scalable Infrastructure

- Dynamically adjust resources based on demand.
- Supports a wide range of computing options.

Container Services

- IBM Cloud Kubernetes Service for orchestration.
- Container Registry for secure image storage.

Integrated DevOps

- Tools for continuous integration and delivery (CI/CD).
- Streamlined pipelines from development to deployment.

Security & Compliance

- Robust security features and compliance certifications.
- Data encryption and access management.

Conclusion

In summary, deploying Dockerised web applications on IBM Cloud offers a powerful, scalable, and efficient solution for modern software delivery. Leveraging tools like Docker, VS Code, and GitHub streamlines the development and deployment lifecycle.

Key Takeaways:

- Containerisation ensures consistent application behaviour across environments.
- IBM Cloud provides a robust platform with extensive services for container management.
- An integrated toolchain (VS Code, GitHub, Docker) enhances developer productivity.