Batch: CC-9 Roll No.: 16010122323

Experiment No. 7

Grade: AA / AB / BB / BC / CC / CD /DD

Title: Implementation of Containers and Dockers

Objective: To implement and understand containerization by creating and managing containers using Docker.

Expected Outcome of Experiment:

CO	Outcome
CO5	Configure and experiment with advanced cloud technologies

Books/ Journals/ Websites referred:
Official Docker Documentation: https://docs.docker.com/

Docker Hub (Public Docker Image Registry): https://hub.docker.com/

"Docker Deep Dive" by Nigel Poulton – A comprehensive book for understanding Docker concepts.

"Kubernetes Up & Running" by Kelsey Hightower, Brendan Burns, and Joe Beda – For further reading on container orchestration.

Medium Articles and Tutorials on Docker: https://medium.com/tag/docker

YouTube – Docker Crash Course for Beginners: https://www.youtube.com/results?search_query=docker+for+beginners

Abstract:-

This experiment focuses on understanding and implementing containerization using **Docker**, a leading containerization platform. Containers offer a lightweight, efficient, and consistent environment for deploying applications across different systems. Docker allows developers to package applications with all their dependencies into standardized units, ensuring that the software runs reliably in any computing environment. The goal of this experiment is to gain practical experience in creating, managing, and deploying Docker containers. Through hands-on implementation, students learn how containerization enhances application portability, scalability, and resource efficiency in modern software development workflows.

Related Theory: -

Containerization is a form of virtualization that allows applications to run in isolated environments called containers. Unlike traditional virtual machines, containers share the host system's kernel and isolate the application processes, resulting in lower overhead and faster startup times.

Docker is an open-source platform designed to automate the deployment of applications inside containers. It provides tools for building, running, and managing containers and includes:

- Docker Engine: The core runtime for building and executing containers.
- Dockerfile: A script that contains instructions to build a Docker image.
- Docker Image: A lightweight, standalone, and executable package that includes everything needed to run an application (code, libraries, environment variables).

- Docker Container: A runtime instance of a Docker image.

- Docker Hub: A cloud-based registry where users can share and download Docker images.

Key Docker commands include:

- docker build: Builds an image from a Dockerfile.

- docker run: Runs a container from an image.

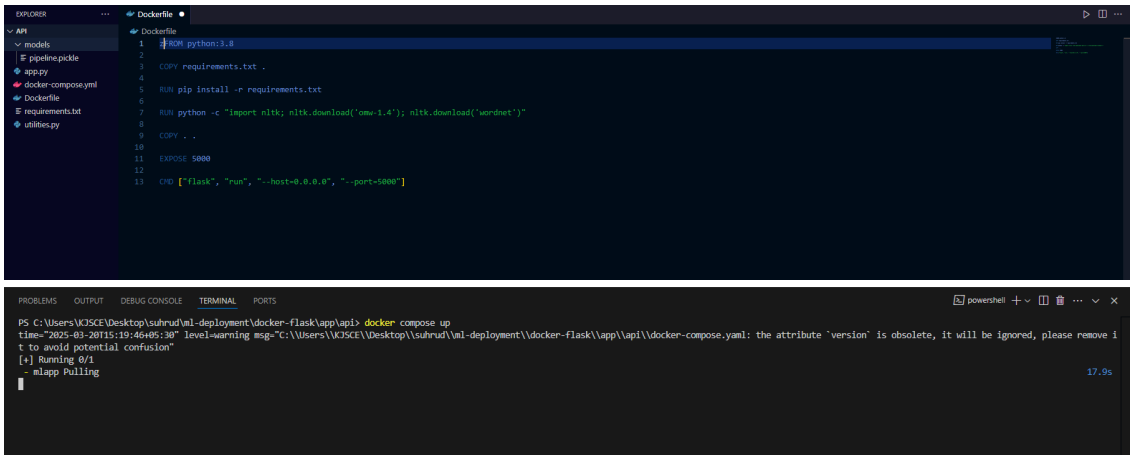
- docker ps: Lists running containers.

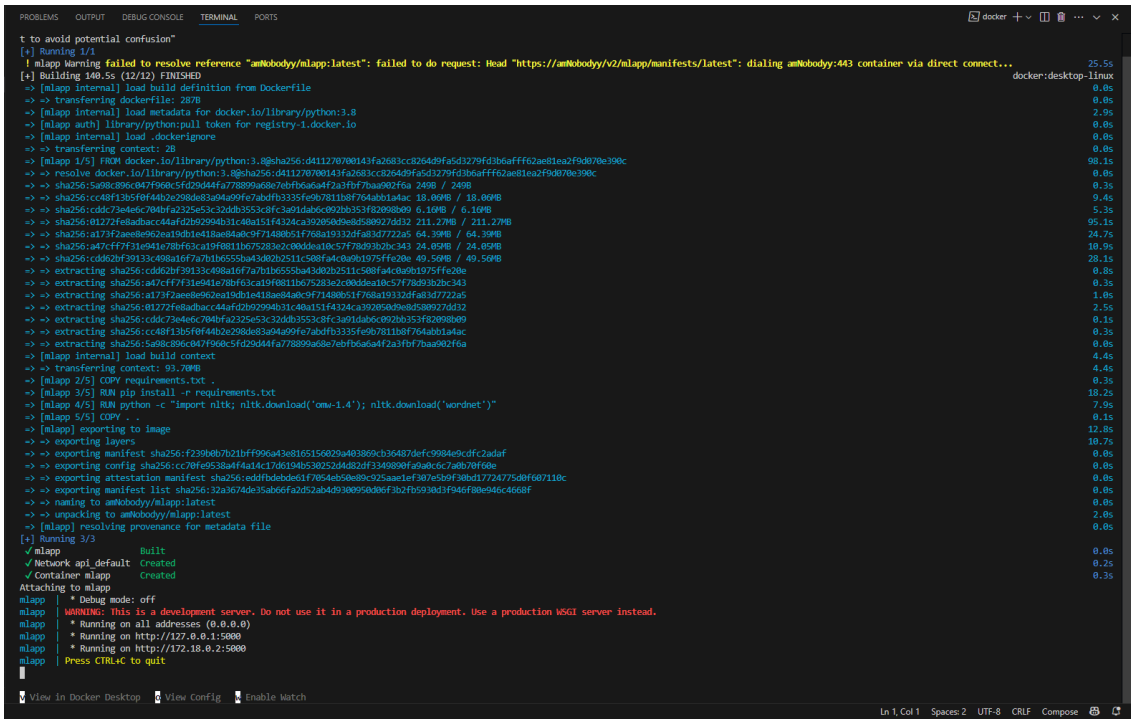
- docker stop / docker rm: Stops or removes containers.

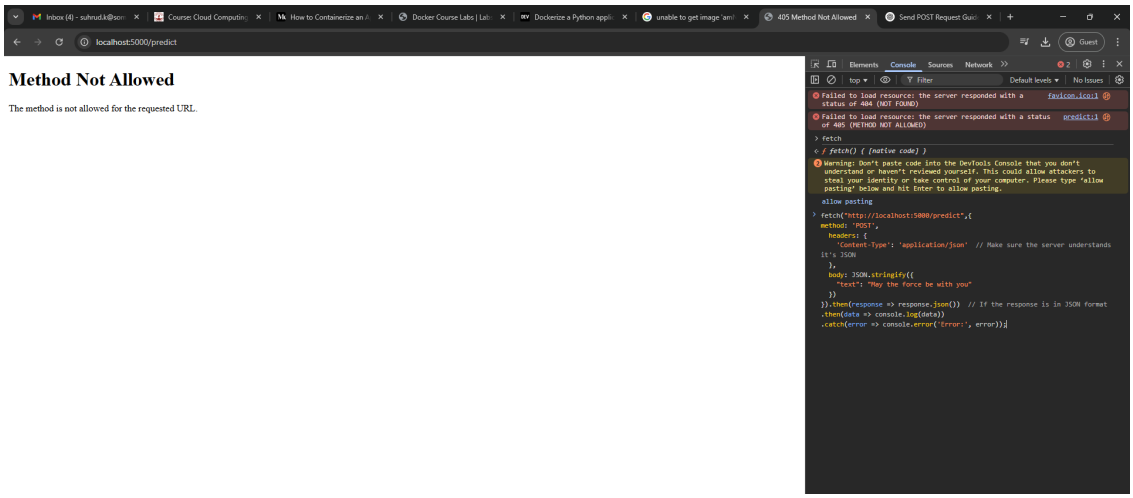
Containers are highly portable and consistent across different environments, making them ideal for DevOps, CI/CD pipelines, microservices, and cloud-native applications.

By understanding Docker, developers can efficiently manage application dependencies, minimize resource usage, and achieve seamless deployment across diverse platforms.

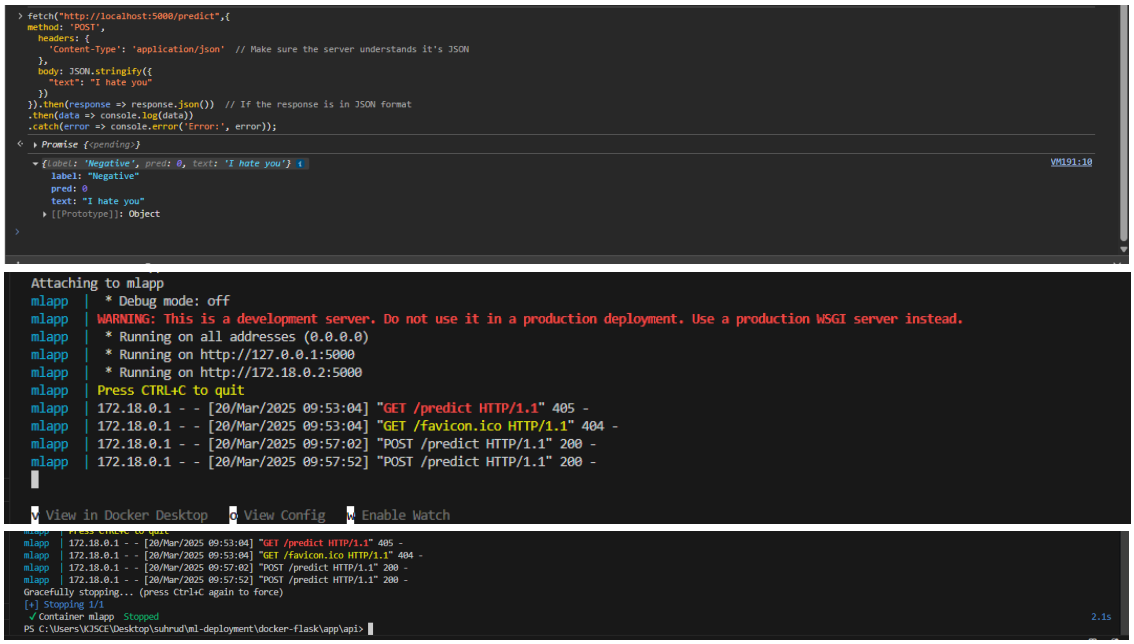
Implementation Details:




PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
t to avoid potential confusion"
[+] Running 1/1
! mlapp Warning failed to resolve reference "amNobodyy/mlapp:latest": failed to do request: Head "https://amNobodyy/v2/mlapp/manifests/latest": dialing amNobodyy:443 container via direct connect...
[+] Building 140.5s (12/12) FINISHED
=> [mlapp internal] load build definition from Dockerfile
=> => transferring dockerfile: 287B
=> [mlapp internal] load metadata for docker.io/library/python:3.8
=> [mlapp auth] library/python:pull token for registry-1.docker.io
=> [mlapp internal] load .dockerignore
=> => transferring context: 2B
=> [mlapp 1/5] FROM docker.io/library/python:3.8@sha256:d411270700143fa2683cc8264d9fa5d3279fd3b6afff62ae81ea2f9d070e390c
=> => resolve docker.io/library/python:3.8@sha256:d411270700143fa2683cc8264d9fa5d3279fd3b6afff62ae81ea2f9d070e390c
=> => sha256:5a98c896c047f960c5fd29d44fa778899a68e7ebfb6a6a4f2a3fbf7baa902f6a 249B / 249B
=> => sha256:cc48f13b5f0f44b2e298de83a94a99fe7abdfb3335fe9b7811b8f764abb1a4ac 18.00MB / 18.00MB
=> => sha256:cddc73e4e6c704bfa2325e53c32ddb3553c8fc3a91dab6c092bb353f82098b09 6.16MB / 6.16MB
=> => sha256:01272fe8adbacc44afd2b92994b31c40a151f4324ca39205d0e8d5080927dd32 211.27MB / 211.27MB
=> => sha256:a173f2aee8e962ea19db1e418ae84a0c9f71480b51f768a19312dfa83d7722a5 64.39MB / 64.39MB
=> => sha256:a47cff7f31e941e78bf63ca19f0811b675283e2c00ddea10c57f78d93b2bc343 24.05MB / 24.05MB
=> => sha256:cdd62bf39133c498a16f7a7b1b6555ba43d02b2511c508fa4c0a9b1975ffe20e 49.56MB / 49.56MB
=> => extracting sha256:cdd62bf39133c498a16f7a7b1b6555ba43d02b2511c508fa4c0a9b1975ffe20e
=> => extracting sha256:a47cff7f31e941e78bf63ca19f0811b675283e2c00ddea10c57f78d93b2bc343
=> => extracting sha256:a173f2aee8e962ea19db1e418ae84a0c9f71480b51f768a19312dfa83d7722a5
=> => extracting sha256:01272fe8adbacc44afd2b92994b31c40a151f4324ca39205d0e8d5080927dd32
=> => extracting sha256:cddc73e4e6c704bfa2325e53c32ddb3553c8fc3a91dab6c092bb353f82098b09
=> => extracting sha256:cc48f13b5f0f44b2e298de83a94a99fe7abdfb3335fe9b7811b8f764abb1a4ac
=> => extracting sha256:5a98c896c047f960c5fd29d44fa778899a68e7ebfb6a6a4f2a3fbf7baa902f6a
=> [mlapp internal] load build context
=> => transferring context: 93.70MB
=> [mlapp 2/5] COPY requirements.txt .
=> [mlapp 3/5] RUN pip install -r requirements.txt
=> [mlapp 4/5] RUN python -c "import nltk; nltk.download('omw-1.4'); nltk.download('wordnet')"
=> [mlapp 5/5] COPY . .
=> [mlapp] exporting to image
=> => exporting layers
=> => exporting manifest sha256:f239b0b7b21bff996a43e8165156029a403869cb36487defc9984e9cdfc2adaf
=> => exporting config sha256:cc70fe9538a4f4a14c17d6194b530252d4d82df3349890fa9a0c6c7a0b70f60e
=> => exporting attestation manifest sha256:eddfbdebde61f7054eb50e89c925aae1ef307e5b9f30bd17724775d0f607110c
=> => exporting manifest list sha256:32a3674de35ab66fa2d52ab4d9300950d06f3b2fb5930d3f946f80e946c4668f
=> => naming to amNobodyy/mlapp:latest
=> => unpacking to amNobodyy/mlapp:latest
=> [mlapp] resolving provenance for metadata file
[+] Running 3/3
✓ mlapp Built
✓ Network api_default Created
✓ Container mlapp Created
Attaching to mlapp
mlapp | * Debug mode: off
mlapp | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
mlapp | * Running on all addresses (0.0.0.0)
mlapp | * Running on http://127.0.0.1:5000
mlapp | * Running on http://172.18.0.2:5000
mlapp | Press CTRL+C to quit
View in Docker Desktop
View Config
Enable Watch
Ln 1, Col 1 Spaces: 2 UTF-8 CRLF Compose


localhost:5000/predict
Method Not Allowed
The method is not allowed for the requested URL.
Elements Console Sources Network
Failed to load resource: the server responded with a status of 404 (NOT FOUND)
favicon.ico:1
Failed to load resource: the server responded with a status of 405 (METHOD NOT ALLOWED)
predict:1
fetch
ƒ fetch() { [native code] }
Warning: Don't paste code into the DevTools Console that you don't understand or haven't reviewed yourself. This could allow attackers to steal your identity or take control of your computer. Please type 'allow pasting' below and hit Enter to allow pasting.
allow pasting
fetch("http://localhost:5000/predict",{
method: 'POST',
headers: {
'Content-Type': 'application/json' // Make sure the server understands it's JSON
},
body: JSON.stringify({
"text": "May the force be with you"
})
}).then(response => response.json()) // If the response is in JSON format
.then(data => console.log(data))
.catch(error => console.error('Error:', error));


fetch("http://localhost:5000/predict",{
method: 'POST',
headers: {
'Content-Type': 'application/json' // Make sure the server understands it's JSON
},
body: JSON.stringify({
"text": "May the force be with you"
})
}).then(response => response.json()) // If the response is in JSON format
.then(data => console.log(data))
.catch(error => console.error('Error:', error));
Promise {<pending>}
{label: 'Positive', pred: 1, text: 'May the force be with you'}
VM158:10
label: "Positive"
pred: 1
text: "May the force be with you"
[[Prototype]]: Object



Conclusion:- Understood and implemented containerization in docker desktop.