

Name : Aryan Patel

Email : aaryanpatel03@gmail.com

Domain : Devops

WEEK 4 Assignment (Docker)

1. Introduction to containerization and Docker fundamentals, Basic Commands.

2. Docker installation and basic container operations, Build an image from Dockerfile.

Steps :

1. What is Containerization?

A lightweight alternative to full machine virtualization. Packages the application and its dependencies into a single image that runs consistently in any environment.

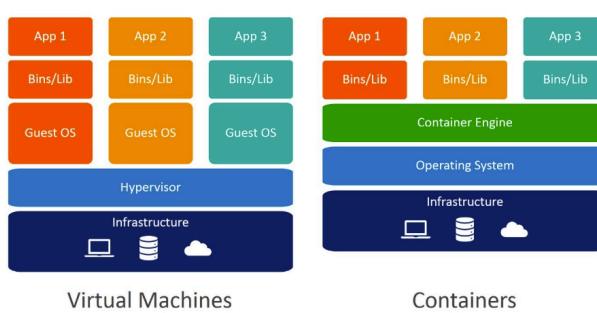
Benefits :

- Portability
- Efficiency
- Scalability
- Isolation



How does it work?

Containerization vs Virtualization



An application on a VM requires a guest OS and thus an underlying hypervisor to run. Hypervisor is used to create multiple machines on a host operating system and it manages virtual machines. These virtual machines have their own operating system and do not use the host's operating system. They have some space allocated. In the software world, containerization is an efficient method for deploying applications. A container encapsulates an application with its own operating environment. It can be placed on any host machine without special configuration, removing the issue of dependencies. VM is hardware virtualization,

[Subscribe](#)



Reference Link : <https://www.techwithkunal.com/blog/getting-started-with-docker>

2. Install Docker Desktop.

The screenshot shows a web browser with the URL docs.docker.com/desktop/setup/install/windows-install/. The page title is "Introducing Docker Hardened Images - Learn More". The main content is titled "Install Docker Desktop on Windows". It includes sections for "Docker Desktop terms" (mentioning commercial use requirements), "System requirements" (listing administrator privileges, WSL verification, and options for WSL via terminal or MSI package), and "Install Docker Desktop on Windows" (with links for x86_64, Microsoft Store, and Arm Early Access). A sidebar on the left contains navigation links for AI, PRODUCTS, and Troubleshoot and support. A right sidebar has "Edit this page" and "Request changes" buttons, and a "Table of contents" section. The bottom of the screen shows a Windows taskbar with various pinned icons like File Explorer, Microsoft Edge, and Docker Desktop.

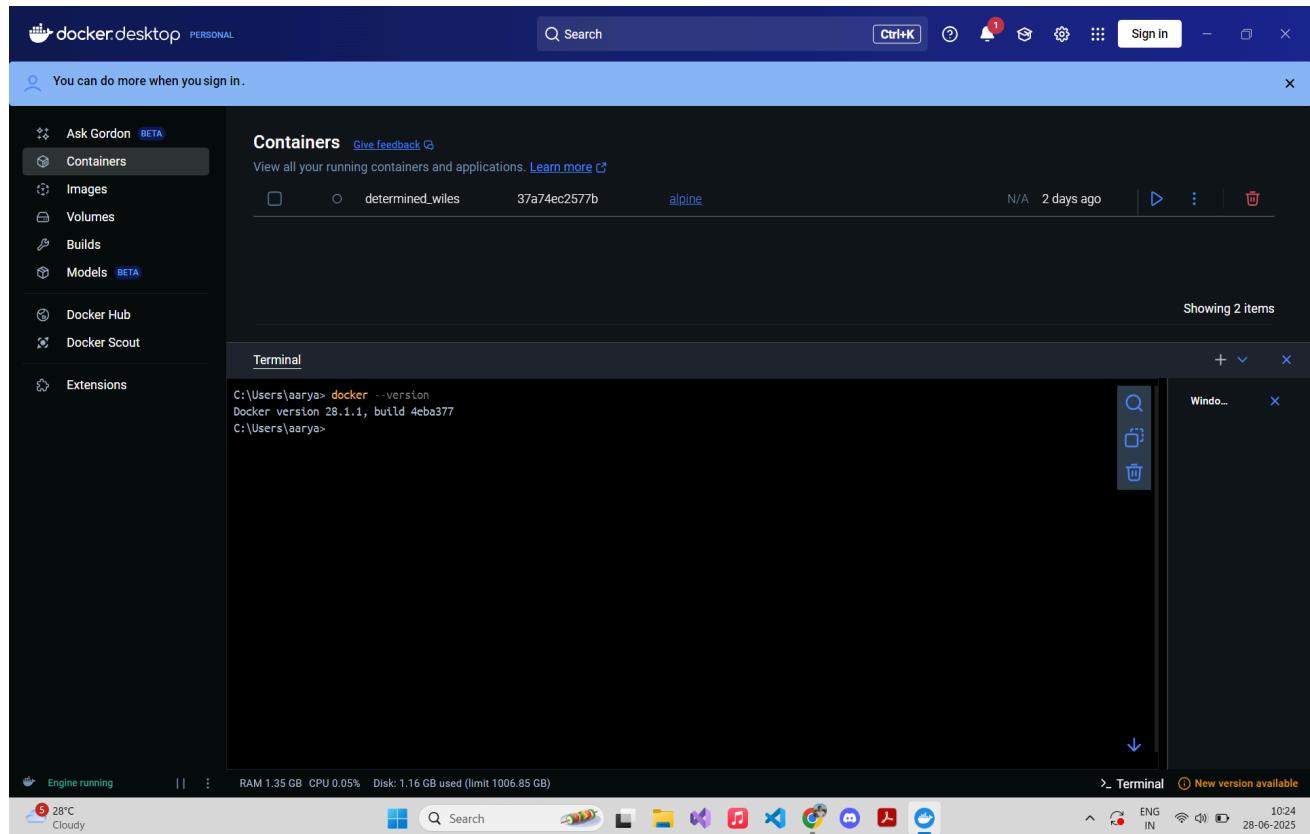
The screenshot shows the Docker Desktop application window. The sidebar on the left lists various running containers: Google Chrome, Clion 2024.3.5, Aaryan - Chrome, demo-app, Cisco Packet Tracer, harshpatel.., Oracle VirtualBox, Assignment 3 - Azure Infra, Notion, Docker Desktop, Microsoft Edge, and MongoDB... The main area is titled "Containers" and shows a table of running containers:

Name	Container ID	Image	Port(s)	CPU (%)	Last st	Actions
naughty_pasteu	53362ff36f7e	alpine		N/A	2 days	
determined_wile	37a74ec2577b	alpine		N/A	2 days	
epic_borg	b99ba545ab1d	hello-world		N/A	2 days	
romantic_buck	58eebe64a0d6	hello-world		N/A	2 days	

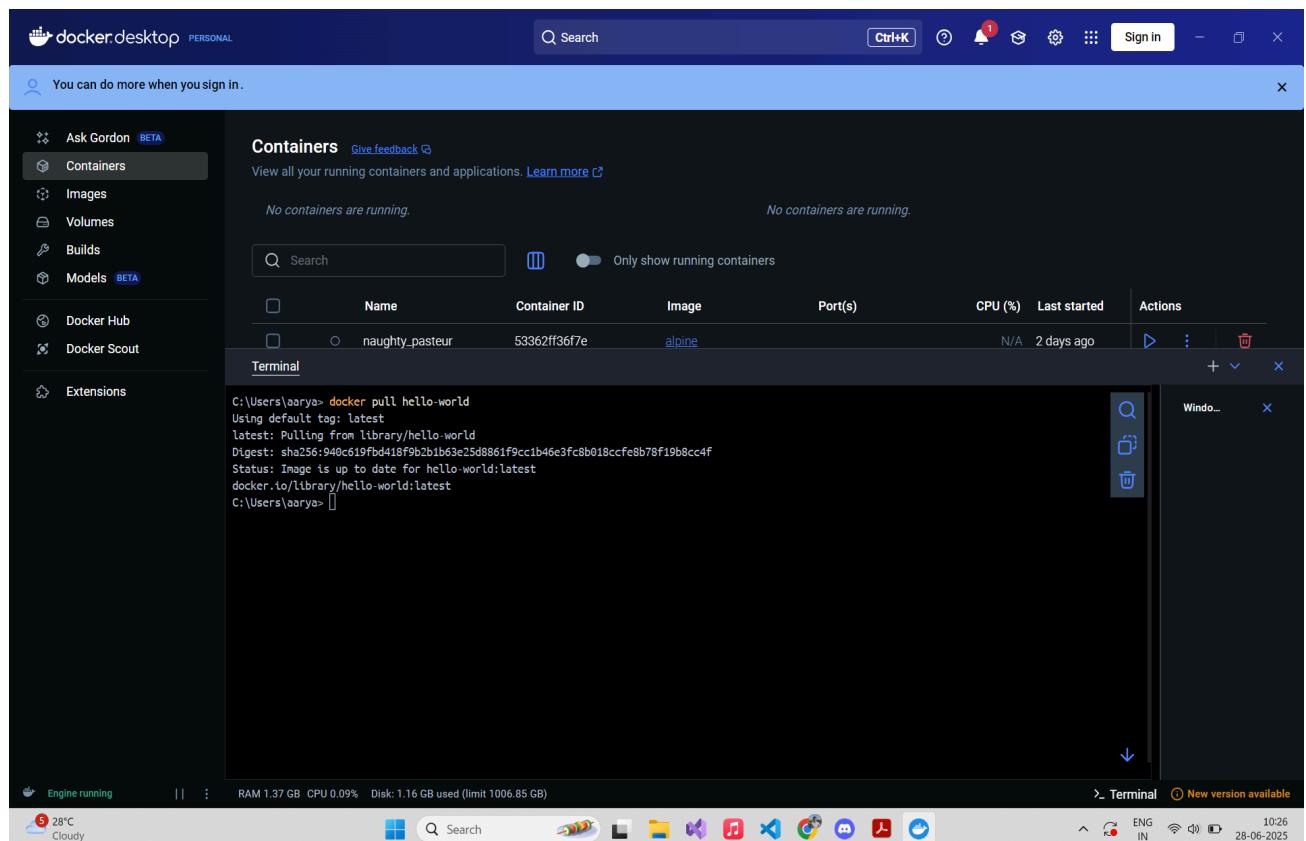
At the bottom, it says "Showing 4 items". Other tabs in the sidebar include "Images", "Volumes", "Builds", "Models", "Docker Hub", "Docker Scout", and "Extensions". The status bar at the bottom shows "Engine running", system resources (RAM 1.35 GB, CPU 0.00%, Disk ~-- GB used / limit ~-- GB), and a "Terminal" button with a "New version available" notification.

3. Docker fundamentals and basic commands.

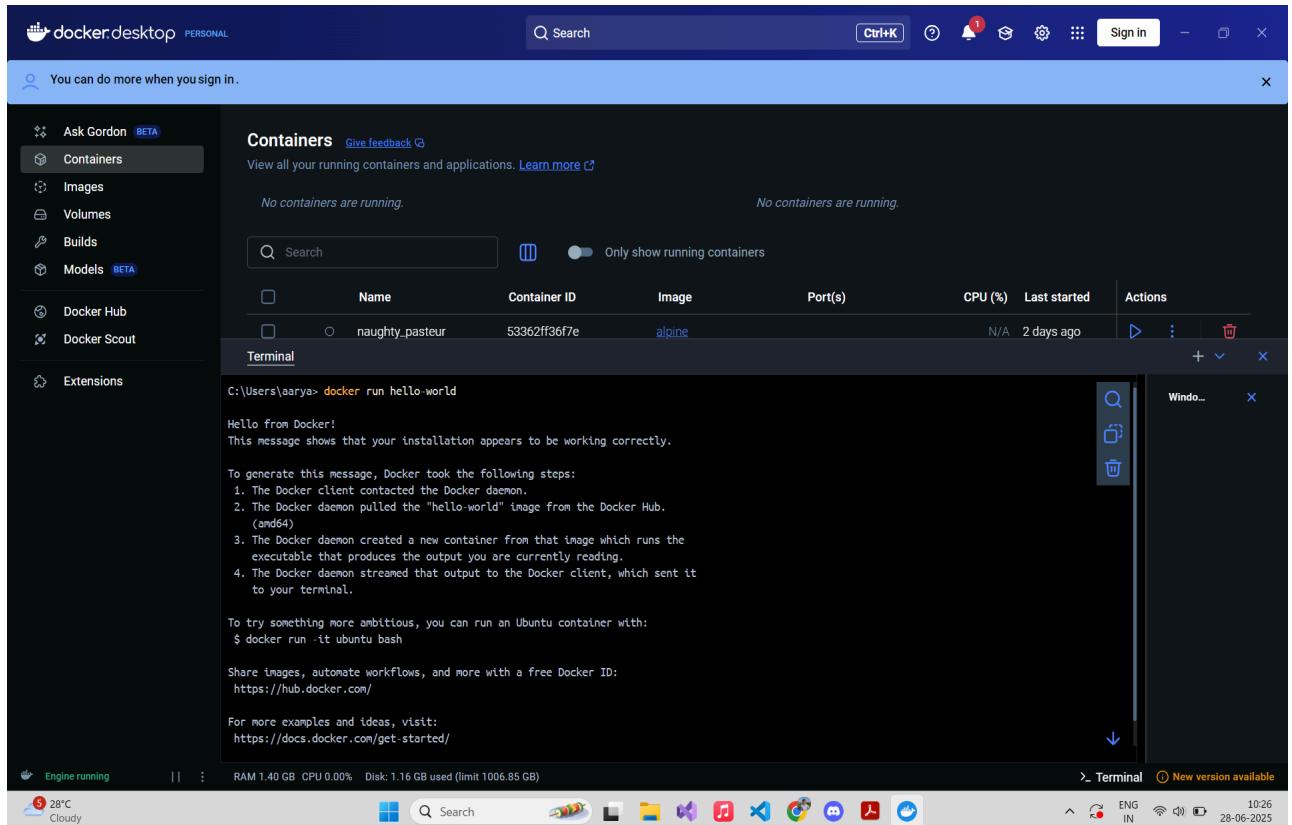
- Check Docker Version



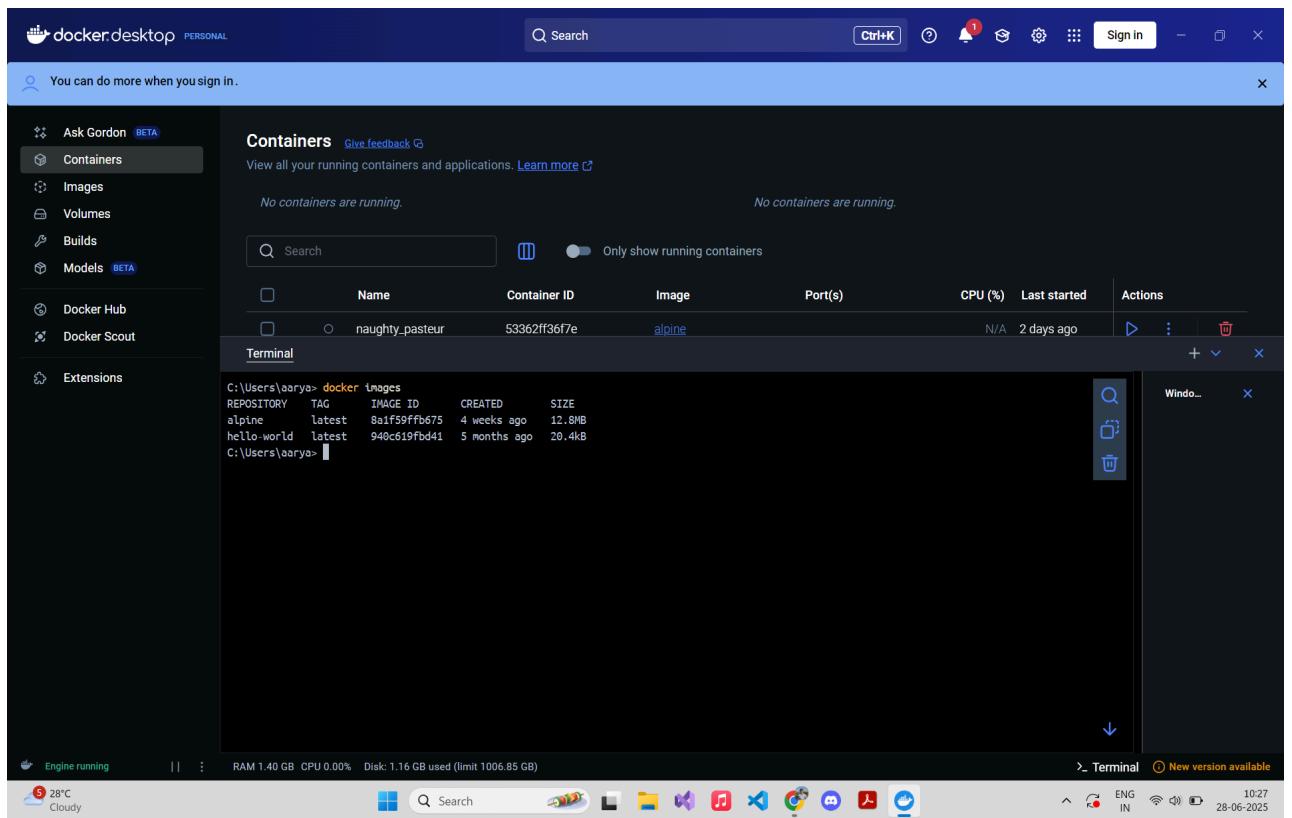
- Pull an Image



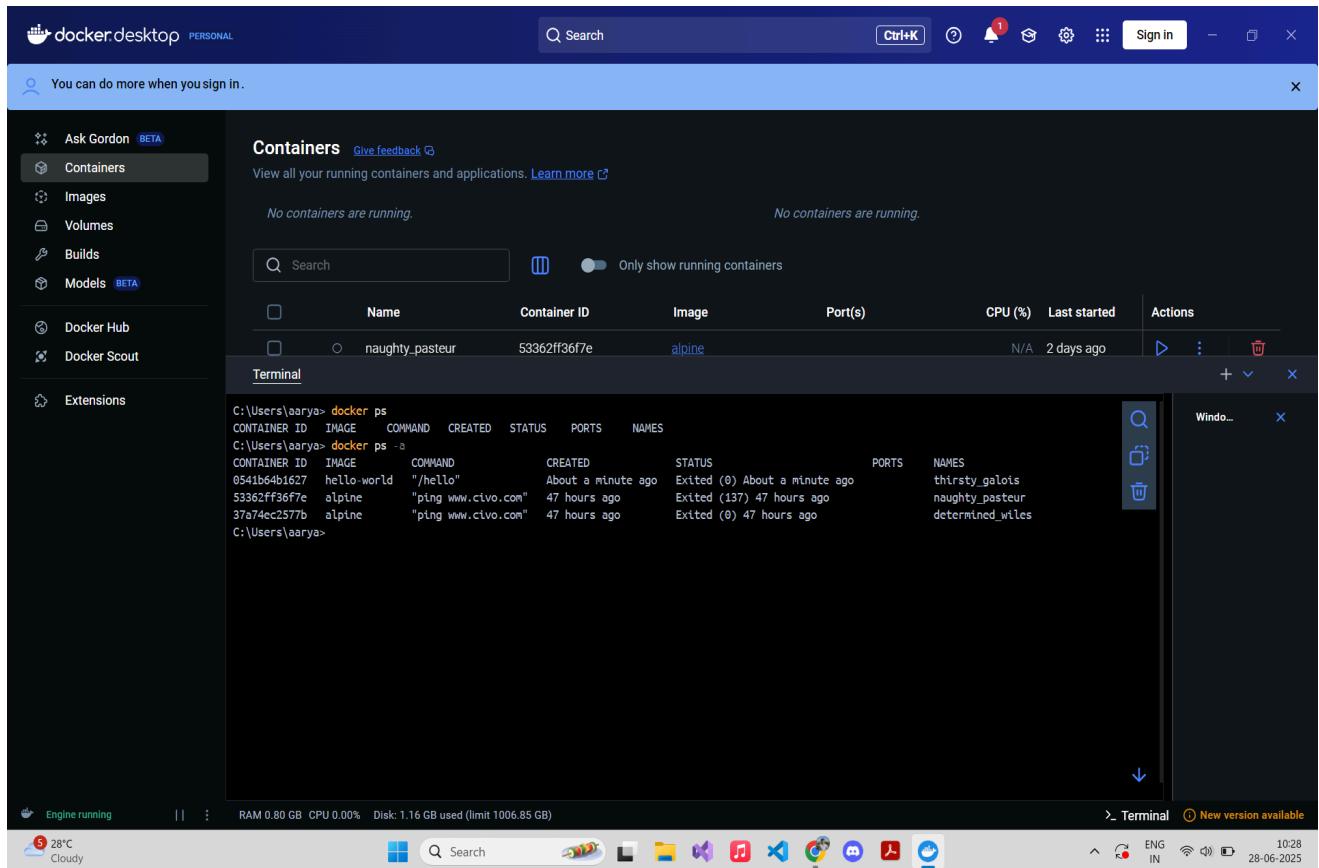
- Run a Container



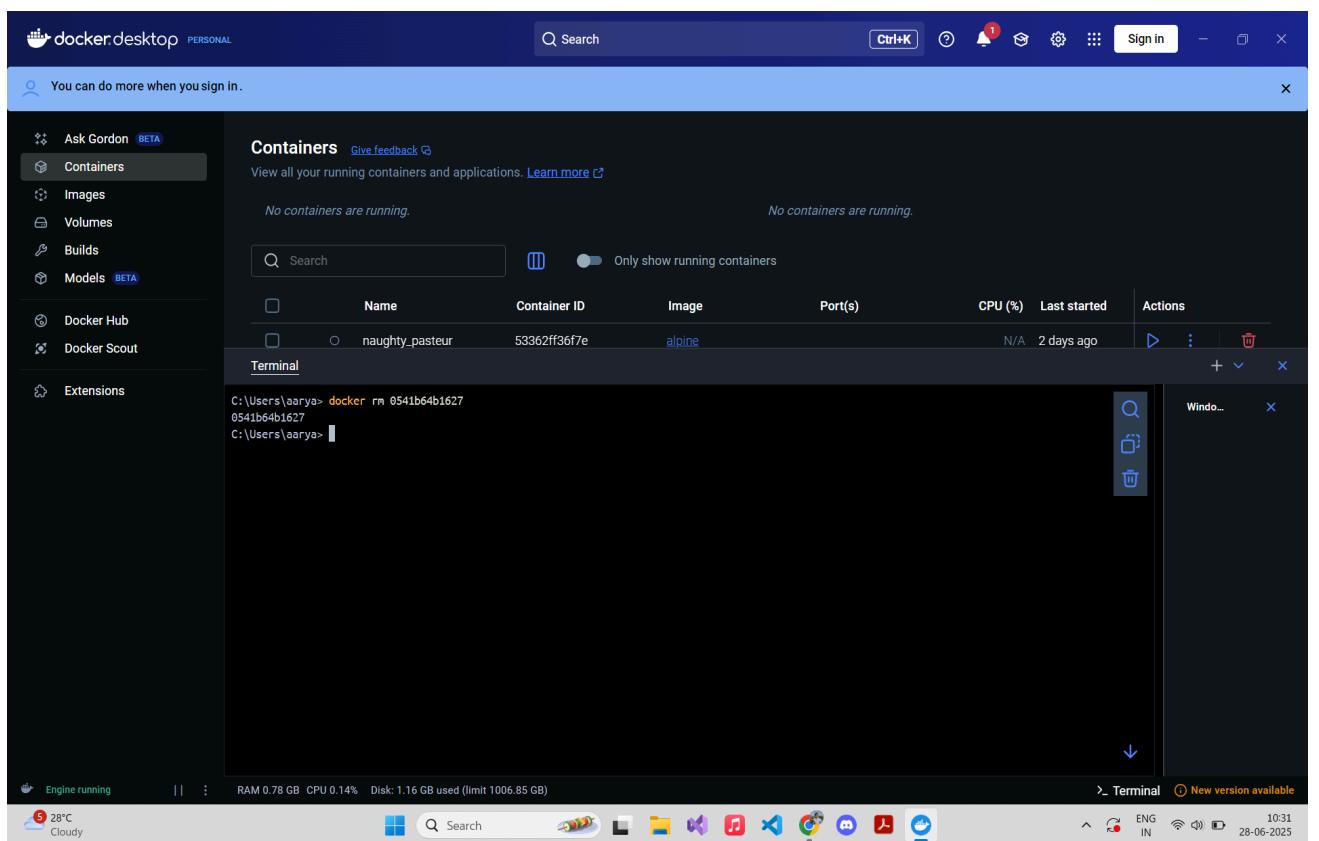
- List Docker Images



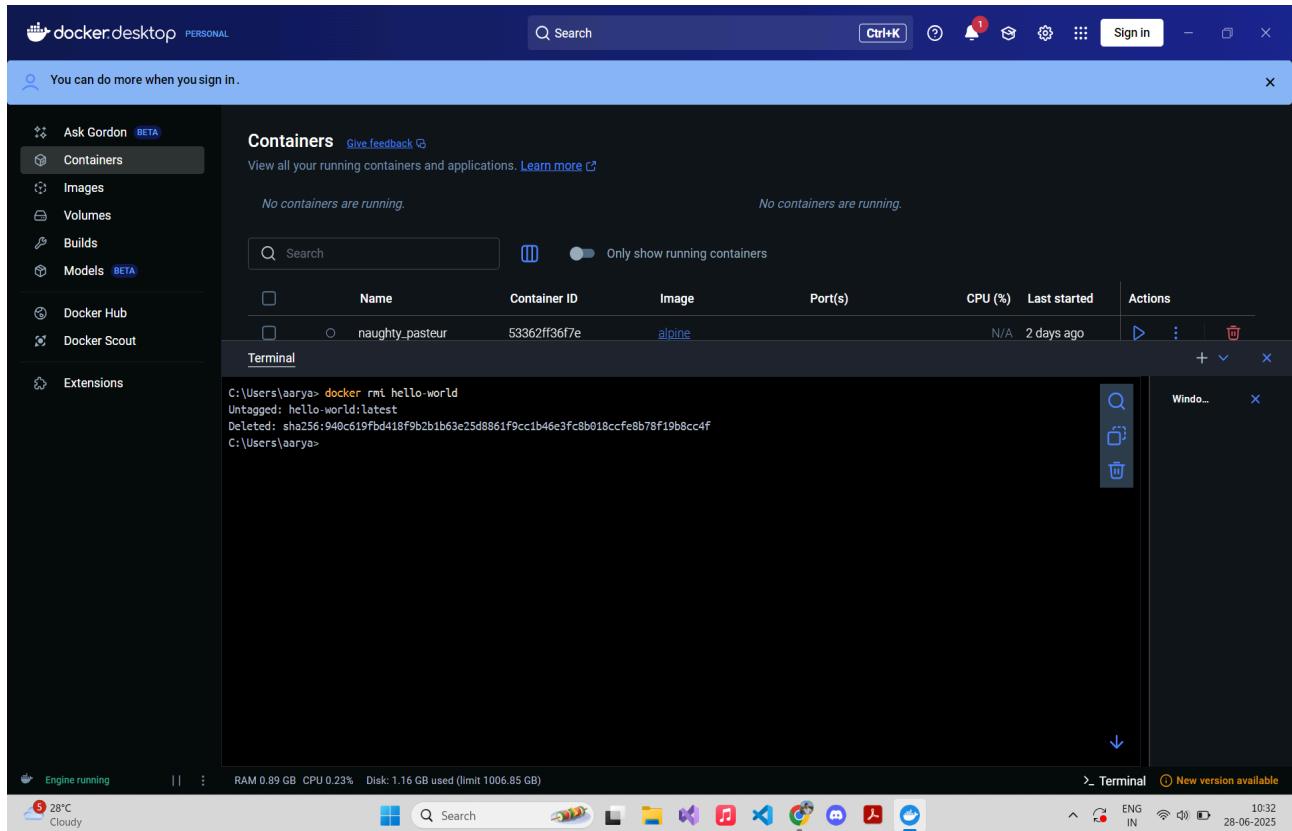
- List Running Containers
- List All Containers (Running + Stopped)



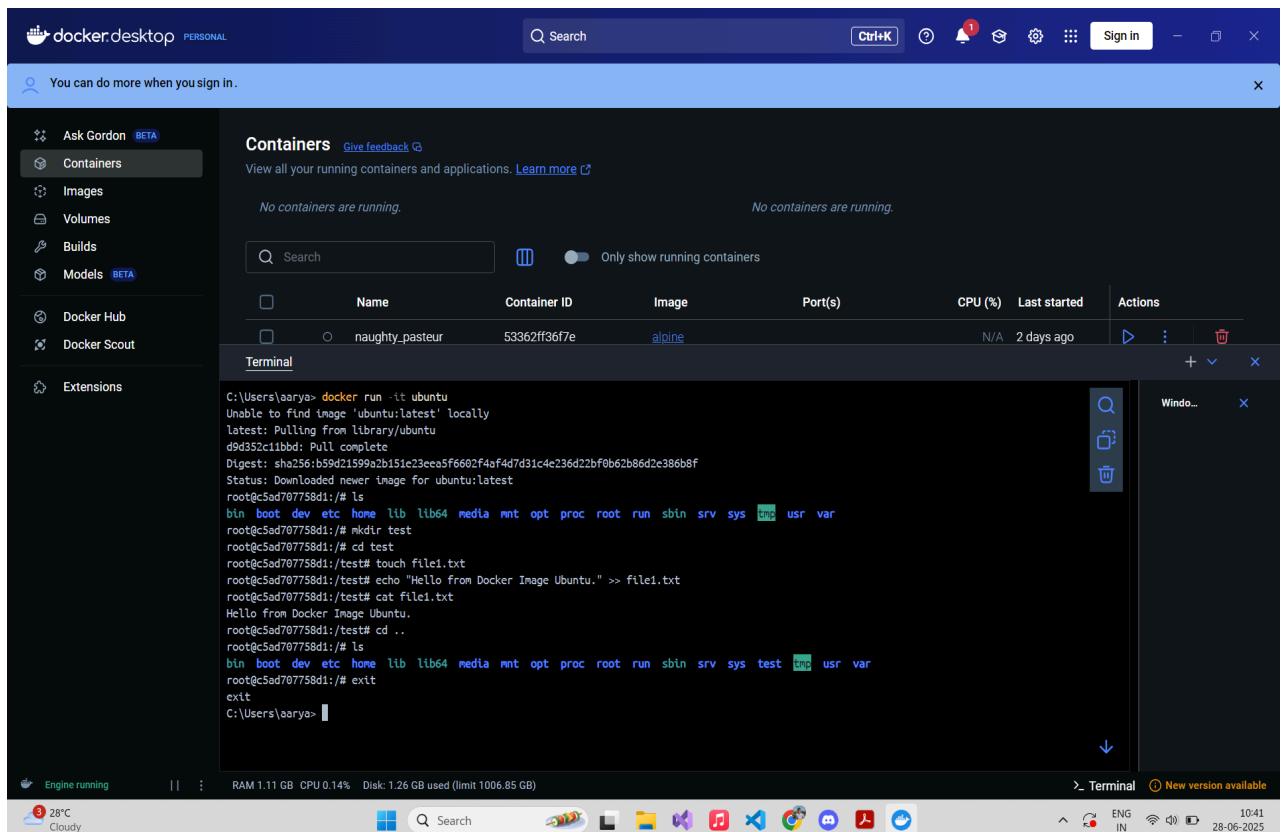
- Remove a Container



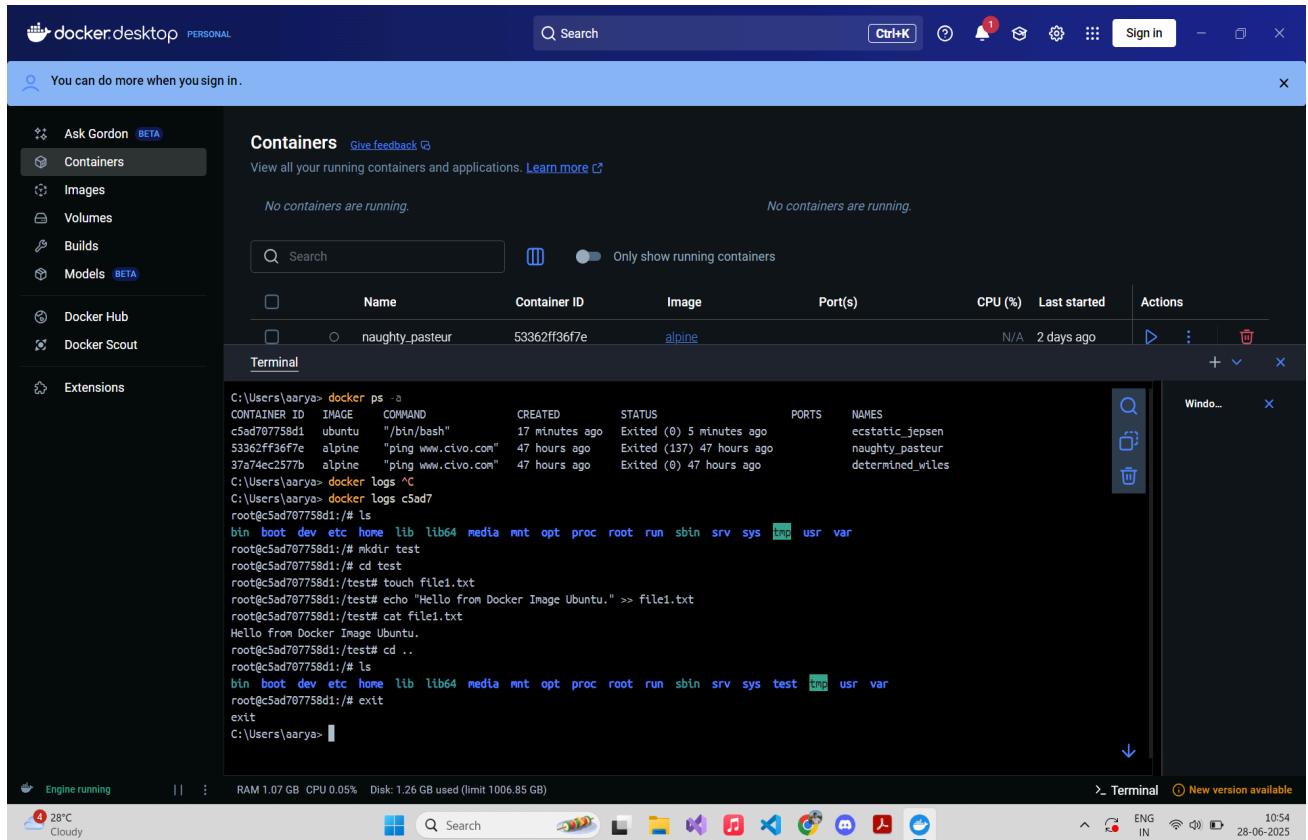
- Remove an Image



- Create and Run Interactive Ubuntu Container

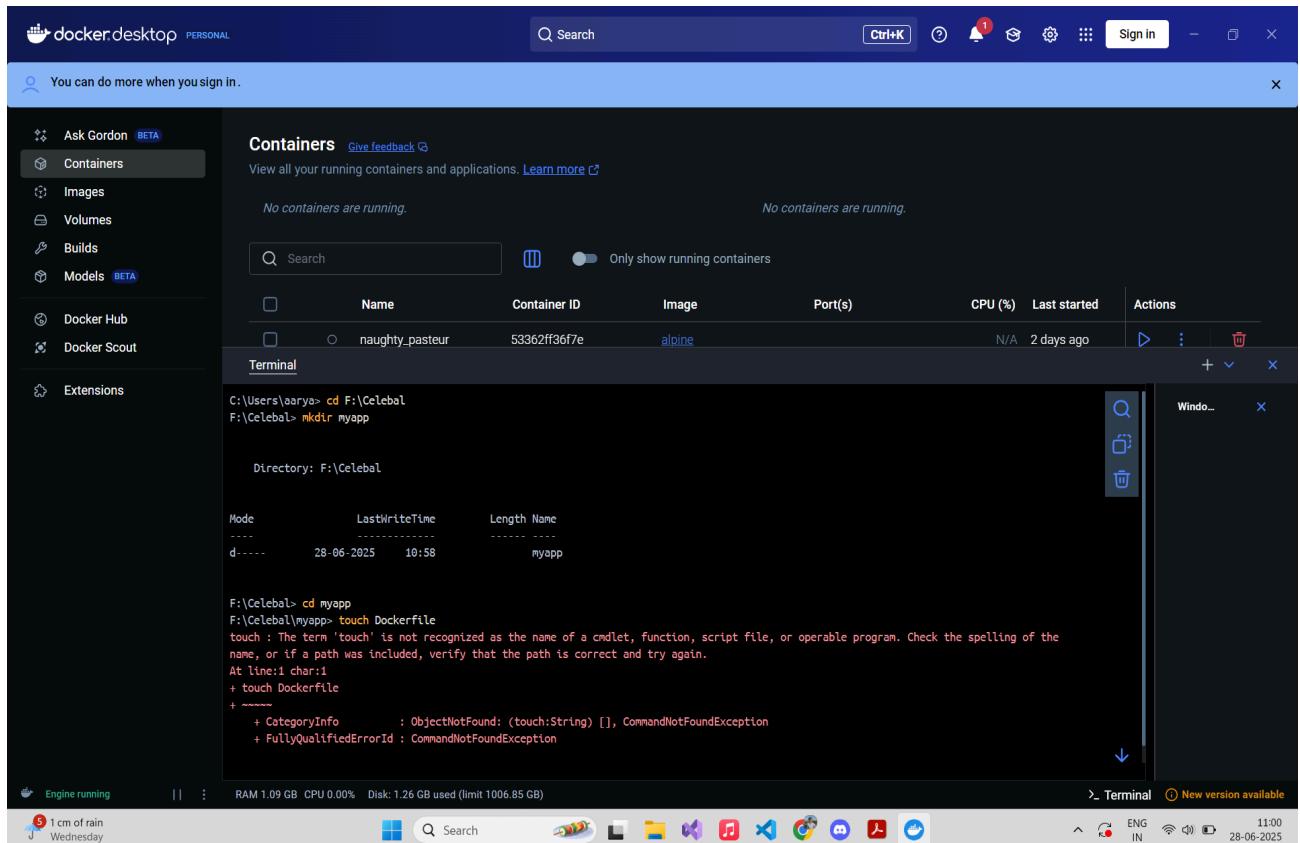


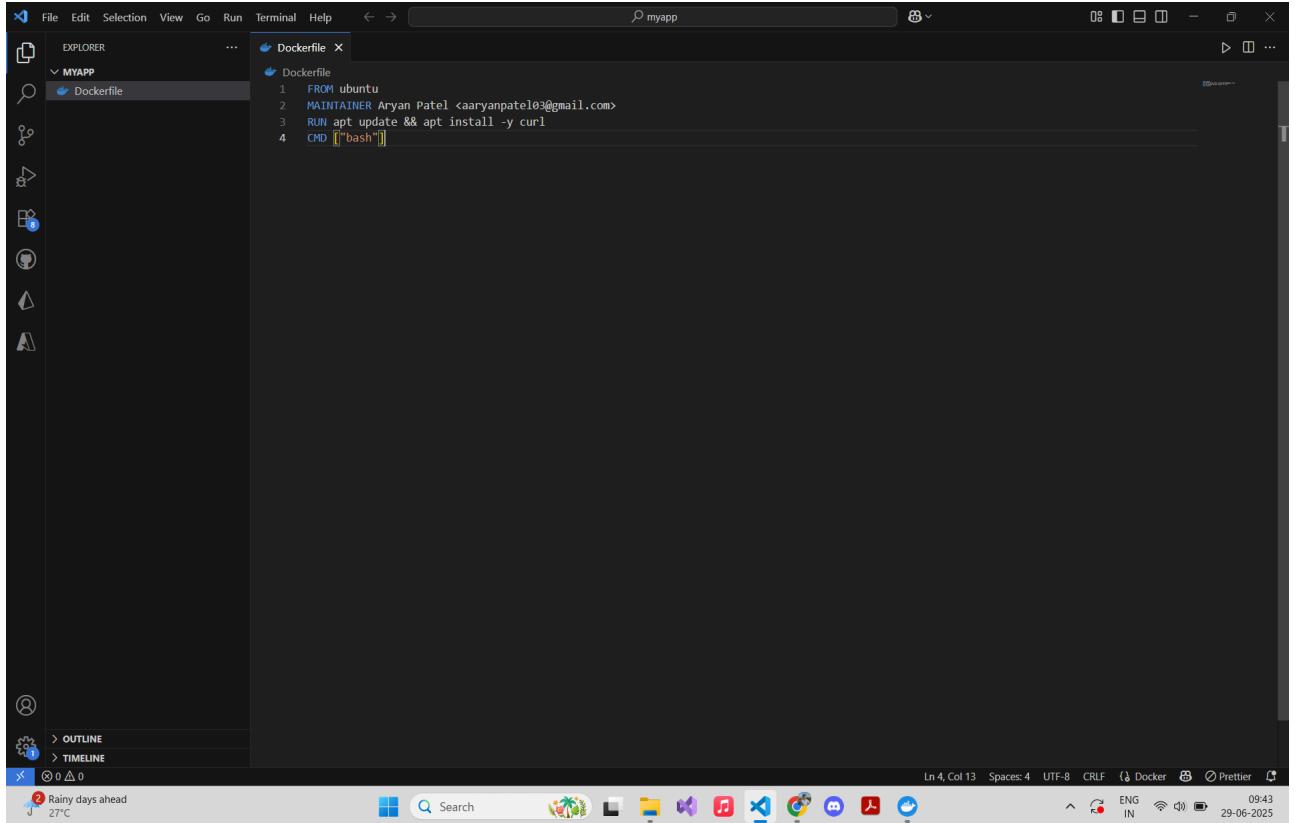
4. Basic Container Operations.



5. Build an Image from a Dockerfile.

- Create a folder "myapp", and inside it, create a file named Dockerfile.

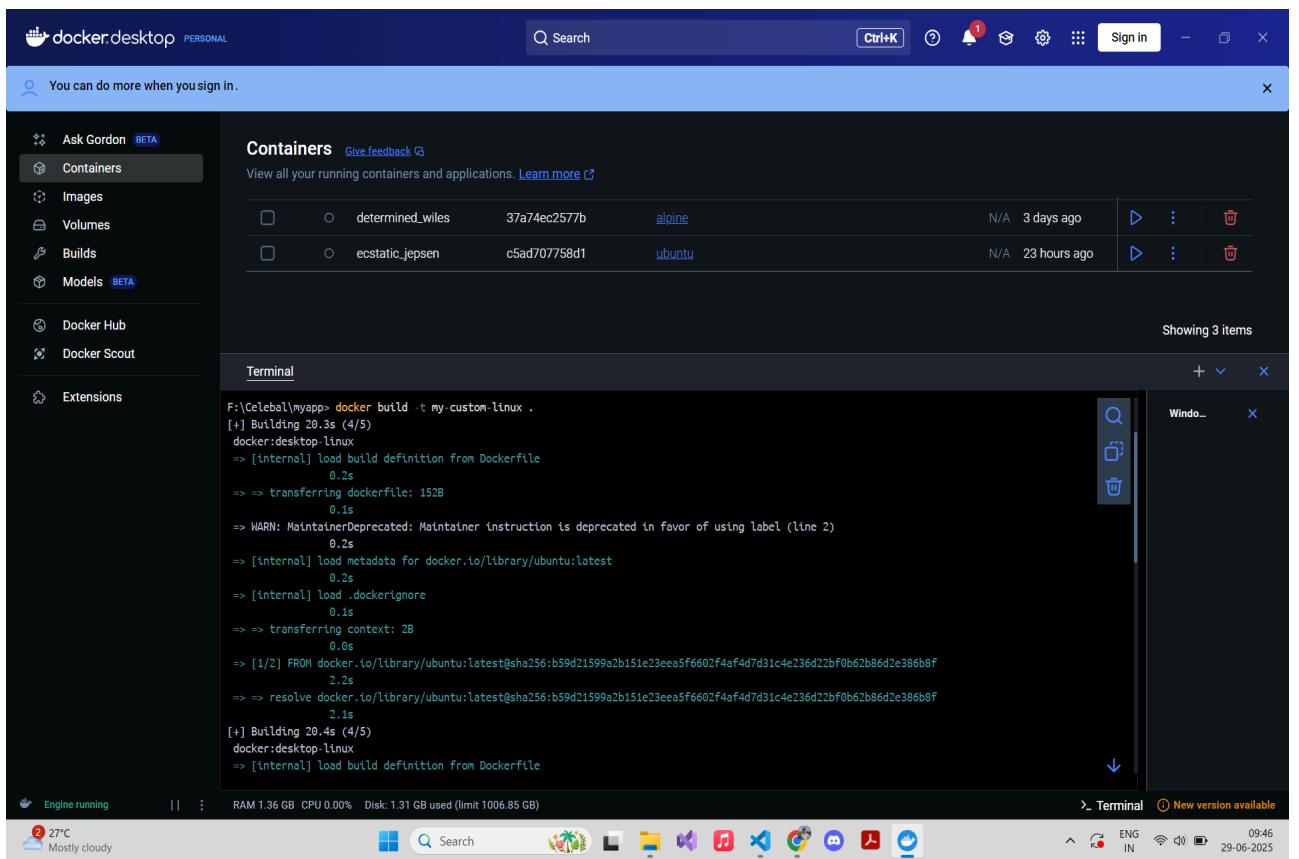




A screenshot of the Visual Studio Code (VS Code) interface. The left sidebar shows a file tree with a folder named 'MYAPP' containing a 'Dockerfile'. The main editor area displays the following Dockerfile:

```
FROM ubuntu
MAINTAINER Aryan Patel <aaryanpatel03@gmail.com>
RUN apt update && apt install -y curl
CMD ["bash"]
```

The status bar at the bottom shows: Ln 4, Col 13 Spaces: 4 UTF-8 CRLF Docker Prettier. The system tray indicates it's 09:43, ENG IN, and the date is 29-06-2025.



A screenshot of the Docker Desktop application. The left sidebar has a 'Containers' tab selected, showing two running containers: 'determined_wiles' (alpine, 3 days ago) and 'ecstatic_jepsen' (ubuntu, 23 hours ago). The main area is a 'Terminal' window displaying the output of a Docker build command:

```
F:\Celebal\myapp> docker build -t my-custom-linux .
[+] Building 20.3s (4/5)
docker:desktop-linux
=> [internal] load build definition from Dockerfile
      0.2s
=> => transferring dockerfile: 152B
      0.1s
=> WARN: MaintainerDeprecated: Maintainer instruction is deprecated in favor of using label (line 2)
      0.2s
=> [internal] load metadata for docker.io/library/ubuntu:latest
      0.2s
=> [internal] load .dockertignore
      0.1s
=> => transferring context: 2B
      0.0s
=> [1/2] FROM docker.io/library/ubuntu:latest@sha256:b59d21599a2b151e23eea5f6602f4af4d7d31c4e236d22bf0b62b86d2e386b8f
      2.2s
=> => resolve docker.io/library/ubuntu:latest@sha256:b59d21599a2b151e23eea5f6602f4af4d7d31c4e236d22bf0b62b86d2e386b8f
      2.1s
[+] Building 20.4s (4/5)
docker:desktop-linux
=> [internal] load build definition from Dockerfile
```

The status bar at the bottom shows: Engine running RAM 1.36 GB CPU 0.00% Disk: 1.31 GB used (limit 1006.85 GB). The system tray indicates it's 09:46, ENG IN, and the date is 29-06-2025.

The screenshot shows the Docker Desktop application interface. On the left, a sidebar contains links like Ask Gordon (BETA), Containers (selected), Images, Volumes, Builds, Models (BETA), Docker Hub, Docker Scout, and Extensions. The main area displays a table of running containers:

		determined_wiles	37a74ec2577b	alpine	N/A	3 days ago	More	⋮	trash
		ecstatic_jepsen	c5ad707758d1	ubuntu	N/A	23 hours ago	More	⋮	trash

Below the table is a terminal window showing the command `docker images` output:

```
F:\Celebal\myapp> docker images
REPOSITORY        TAG      IMAGE ID      CREATED       SIZE
my-custom-linux   latest   761fe38e5d54  2 minutes ago  216MB
alpine            latest   8a1f59ffb675  4 weeks ago   12.8MB
ubuntu             latest   b59d21599a2b  4 weeks ago   117MB
F:\Celebal\myapp>
```

The taskbar at the bottom shows system status: Engine running, RAM 0.88 GB, CPU 0.32%, Disk 1.41 GB used (limit 1006.85 GB). It also includes a weather icon (27°C, Mostly cloudy), a search bar, and various pinned icons.

This screenshot is similar to the first one, showing the Docker Desktop interface. The sidebar and container table are identical. The terminal window now displays the output of the `apt install -y curl` command:

```
F:\Celebal\myapp> docker run -it my-custom-linux
root@4a436440ad03:/# ls
bin boot dev etc home lib lib64 media mnt opt proc root run sbin srv sys tmp usr var
root@4a436440ad03:/# sudo apt install -y curl
bash: sudo: command not found
root@4a436440ad03:/# sudo apt install -y curl
bash: sudo: command not found
root@4a436440ad03:/# apt install -y curl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
curl is already the newest version (8.5.0-2ubuntu10.6).
0 upgraded, 0 newly installed, 0 to remove and 6 not upgraded.
root@4a436440ad03:/# exit
exit
F:\Celebal\myapp>
```

The taskbar at the bottom shows system status: RAM 1.00 GB, CPU 0.86%, Disk 1.41 GB used (limit 1006.85 GB). It includes a weather icon (27°C, Mostly cloudy), a search bar, and various pinned icons.

3. Docker Registry, DockerHub, Create a Multi-Stage Build.

Steps :

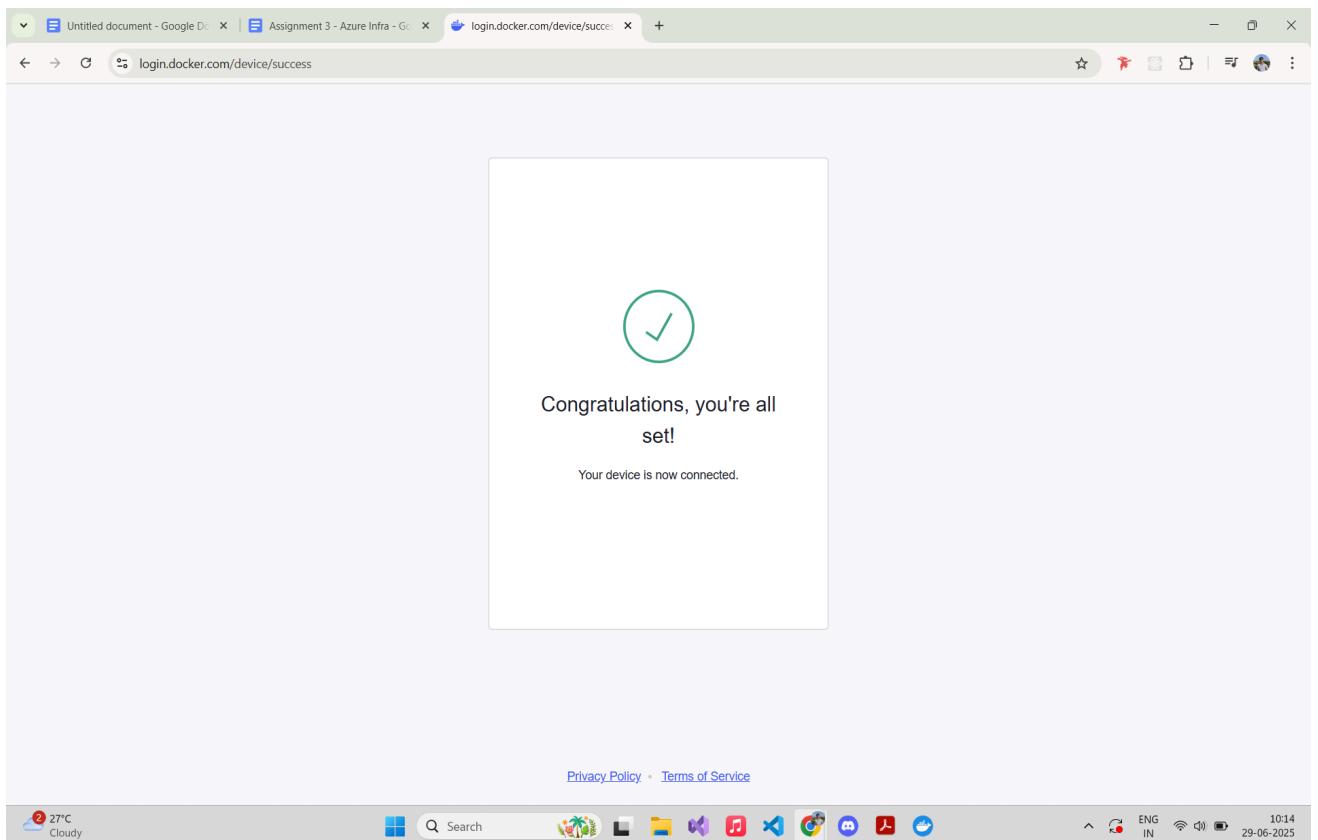
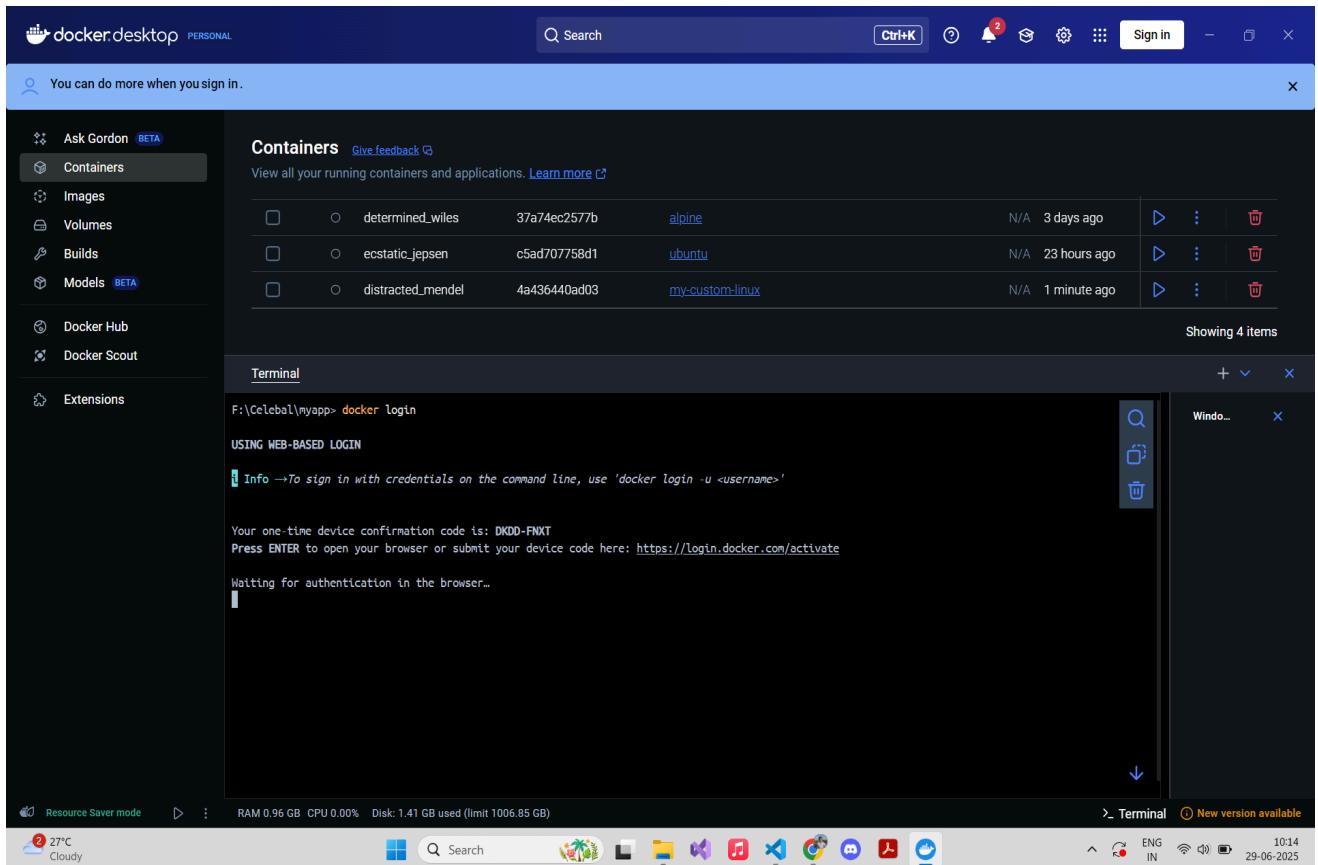
1. Create a Docker Hub account and repository.

- Click "Create Repository", enter name, visibility (public/private).

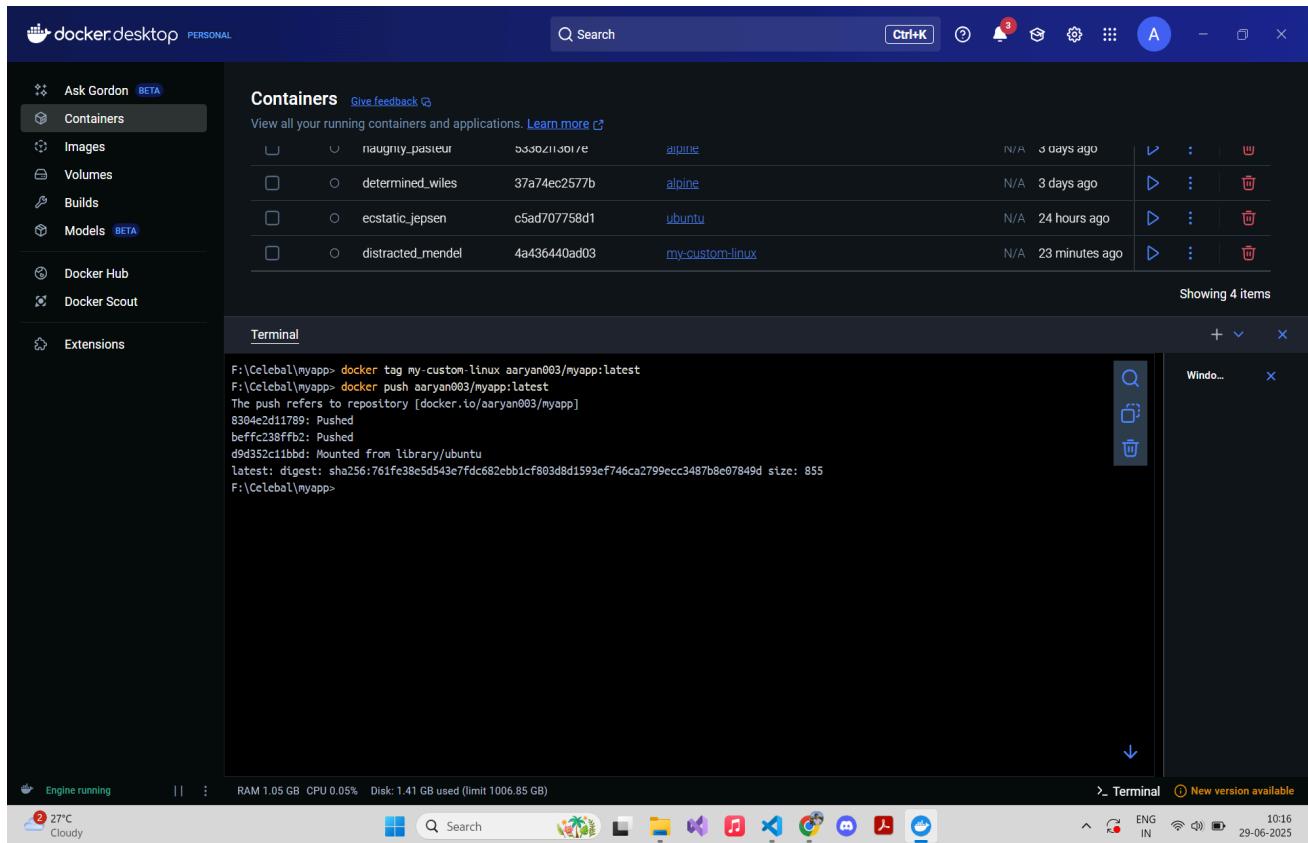
The screenshot shows the Docker Hub homepage. On the left, there's a sidebar with navigation links: 'Repositories' (which is selected), 'Collaborations', 'Settings', 'Default privacy', 'Notifications', 'Billing', 'Usage', 'Pulls', and 'Storage'. The main content area features a large blue banner with the text 'Welcome to Docker' and 'Download the desktop application'. Below the banner are three cards: 'Create a Repository' (Push container images to a repository on Docker Hub.), 'Docker Hub Basics' (Watch the guide on how to create and push your first image into a Docker Hub repository.), and 'Language-Specific Guides' (Learn how to containerize language-specific applications using Docker.). Further down, there's a section titled 'Access the world's largest library of container images' with a 'Spotlight' card. The status bar at the bottom shows weather information ('Heavy rain Today'), system icons, and the date ('29-06-2025').

The screenshot shows the 'Create repository' form on Docker Hub. The left sidebar has the same navigation as the previous screenshot. The main form has a title 'Create repository'. It includes fields for 'Repository Name' (set to 'myapp'), 'Short description' (set to 'celab1 week-4 Docker'), and 'Visibility' (set to 'Private'). To the right, there's a 'Pushing images' section with instructions for using the CLI: 'docker tag local-image:tagname new-repo:tagname' and 'docker push new-repo:tagname'. A note says to replace 'tagname' with the desired image repository tag. At the bottom are 'Cancel' and 'Create' buttons. The status bar at the bottom shows weather information ('Rainy days ahead 27°C'), system icons, and the date ('29-06-2025').

- Login via CLI.



- Tag a local image
- Push image to Docker Hub



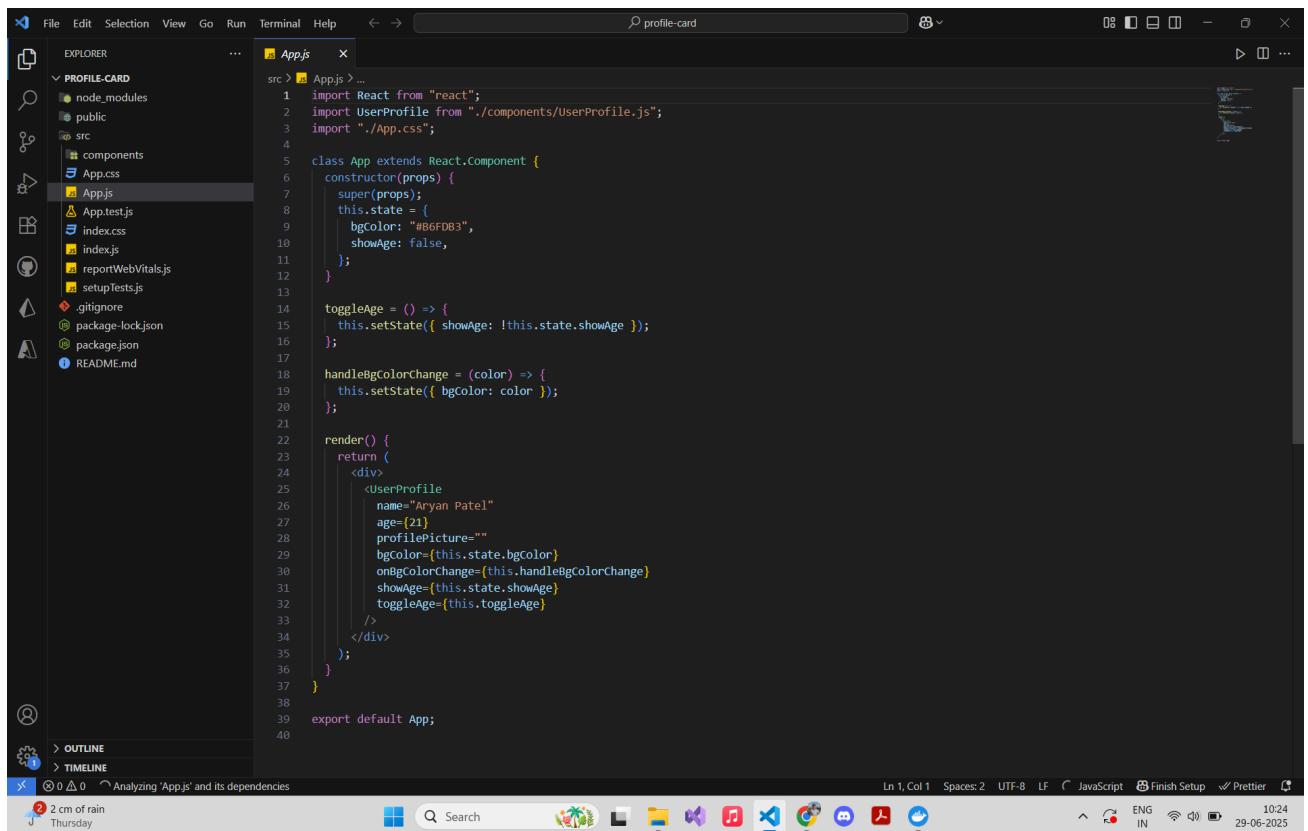
- Verify on Docker Hub

The screenshot shows the Docker Hub repository page for 'aaryan003/myapp'. The left sidebar includes 'Repositories', 'Collaborations', 'Settings', 'Default privacy', 'Notifications', 'Billing', 'Usage', 'Pulls', and 'Storage'. The main content area shows the repository details:

- General** tab: Shows 0 tags. A table lists one tag: 'latest' (OS: Alpine, Type: Image, Pulled: less than 1 day, Pushed: 1 minute ago).
- Docker commands**: A button to 'Push a new tag to this repository' with the command 'docker push aaryan003/myapp:tagname'.
- Tags** section: Shows 0 tags.
- Image Management** (BETA) tab: Shows Docker Scout inactive.
- Collaborators**, **Webhooks**, and **Settings** tabs are also present.
- buildcloud** advertisement: 'Build with Docker Build Cloud' and 'Accelerate image build times with access to cloud-based builders and shared cache.'
- Repository overview**: Status 'INCOMPLETE'.

2. Creating a Multi-Stage build.

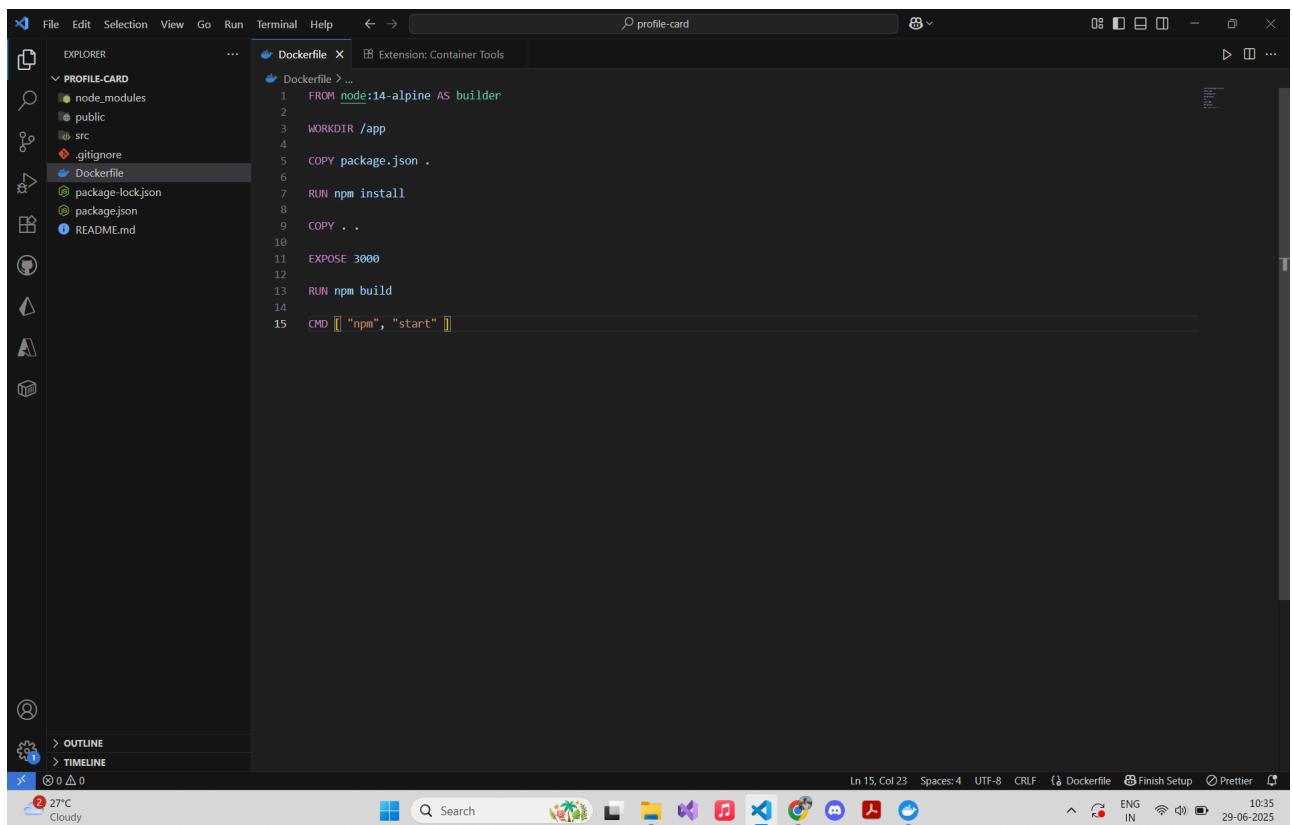
- Create a Sample React App



```
src > App.js > ...
1 import React from "react";
2 import UserProfile from "./components/UserProfile.js";
3 import "./App.css";
4
5 class App extends React.Component {
6   constructor(props) {
7     super(props);
8     this.state = {
9       bgColor: "#B6F0B3",
10      showAge: false,
11    };
12  }
13
14  toggleAge = () => {
15    this.setState({ showAge: !this.state.showAge });
16  };
17
18  handleBgColorChange = (color) => {
19    this.setState({ bgColor: color });
20  };
21
22  render() {
23    return (
24      <div>
25        <UserProfile
26          name="Aryan Patel"
27          age={21}
28          profilePicture=""
29          bgColor={this.state.bgColor}
30          onBgColorChange={this.handleBgColorChange}
31          showAge={this.state.showAge}
32          toggleAge={this.toggleAge}
33        />
34      </div>
35    );
36  }
37}
38
39 export default App;
```

- Before using multi-stage build :

Dockerfile :



```
src > Dockerfile > ...
1 FROM node:14-alpine AS builder
2
3 WORKDIR /app
4
5 COPY package.json .
6
7 RUN npm install
8
9 COPY . .
10
11 EXPOSE 3000
12
13 RUN npm build
14
15 CMD [ "npm", "start" ]
```

```

F:\GUNI\SEM 6\React\Practical_6\profile-card> docker build . -t profile-app:build
[+] Building 357.0s (12/12) FINISHED
-> [internal] load build definition from Dockerfile
-> [internal] load metadata for docker.io/library/node:14-alpine
-> [auth] library/node:pull token for registry-1.docker.io
-> [internal] load .dockerignore
-> [internal] transfer context: 2B
-> FROM docker.io/library/node:14-alpine@sha256:43421515487a37329c9e86720ff89e704d3a75e554822e07f3e0c0f9e606121b33
-> => resolve docker.io/library/node:14-alpine@sha256:43421515487a37329c9e86720ff89e704d3a75e554822e07f3e0c0f9e606121b33
-> sha256:561cb69653d5a69725be56e02128a4e96fb34a8b47decf2bdeb79a225feaf 448B / 448B
-> => sha256:c22e46face39e2c3de3f7fe7c15f8fd7c4e5add29d7f64b87abda09 3.37MB / 3.37MB
-> => sha256:esfcac395a62e277102a9e5283f6ed43b3e4f20f98e3ce7e425be226ba6 2.37MB / 2.37MB
-> => sha256:8f665685b215c7da9164545f1bbd74d800af77d0267d031fe0345c0c8fb88 37.17MB / 37.17MB
-> => extracting sha256:f56be85fc22e46face39e2c3de3f7fe7c15f8fd7c4e5add29d7f64b87abda09 0.39
-> => extracting sha256:8f665685b215c7da9164545f1bbd74d800af77d0267d031fe0345c0c8fb88 2.58
-> => extracting sha256:esfcac395a62e277102a9e5283f6ed43b3e4f20f98e3ce7e425be226ba6 0.15
-> => extracting sha256:561cb69653d56e9725be56e02128a4e96fb34a8b47decf2bdeb79a225feaf 0.09
-> [internal] load build context
-> [internal] transfer context: 288.83MB
-> [2/6] WORKDIR /app
-> [3/6] COPY package.json .
-> [4/6] RUN npm install
-> [5/6] COPY . .
-> [6/6] RUN npm build
-> exporting to image
-> exporting layers
-> exporting manifest sha256:1ebff803d5e8ca58e42074f9dff7bc688e783fa903ff9ebb534a417b2a9279
-> => exporting config sha256:52e66d9d7dfa9c54aa8c7209e7d6b08725398fd7615c3cad65a9781e88d6f90a
-> => exporting attestation manifest sha256:c803c60e87e5d3cbf3ec1d7f9ef075469b464fe6b0b74a19ccc2c556ea41
-> => exporting manifest list sha256:3156ba74f351dd0888703215a726d0e8b0f8b66a31fad9558e138cebbf3c33f5
-> => naming to docker.io/library/profile-app:build
-> => unpacking to docker.io/library/profile-app:build

```

Ln 15, Col 23 Spaces: 4 UTF-8 CRLF Dockerfile Finish Setup Prettier ENG IN 10:43 29-06-2025

- You can see the size of this docker image “profile-app” which is very large before using multi-stage build.

```

F:\GUNI\SEM 6\React\Practical_6\profile-card> docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
profile-app build 3156ba74f35 About a minute ago 1.32GB
aaryan003/myapp latest 761fe38e5d54 57 minutes ago 216MB
my-custom-linux latest 761fe38e5d54 57 minutes ago 216MB
alpine latest 8a1f59ffbd75 4 weeks ago 12.8MB
ubuntu latest b59d21599a2b 4 weeks ago 117MB

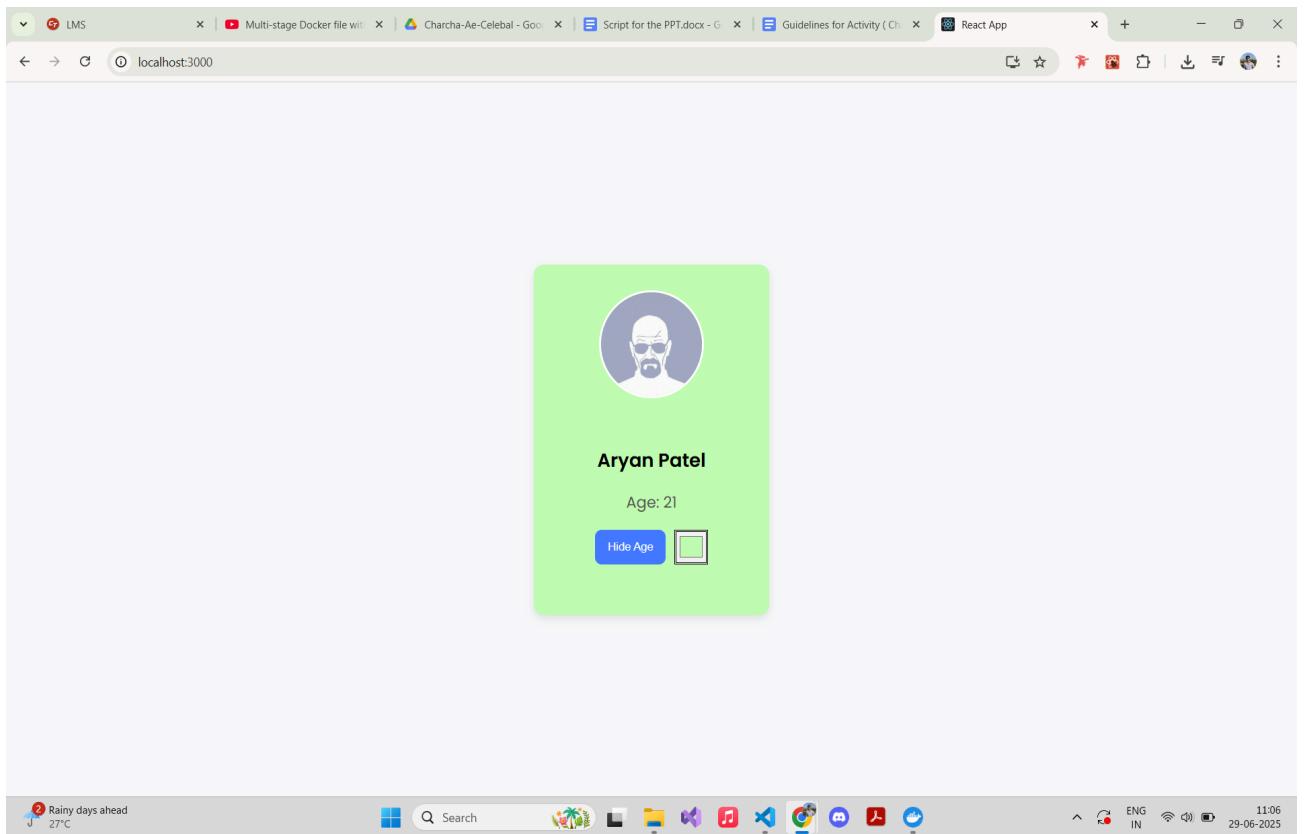
```

Ln 15, Col 23 Spaces: 4 UTF-8 CRLF Dockerfile Finish Setup Prettier ENG IN 10:43 29-06-2025

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** On the left, it shows a project structure for "PROFILE-CARD" containing files like node_modules, public, src, .gitignore, Dockerfile, nginx.conf, package-lock.json, package.json, and README.md.
- Dockerfile:** The file is open in the editor, showing the following Dockerfile content:

```
FROM node:14-alpine AS builder
WORKDIR /app
COPY package.json .
RUN npm install
COPY . .
EXPOSE 3000
RUN npm build
CMD [ "npm", "start" ]
```
- Terminal:** The bottom panel shows the terminal output with two entries:
 - F:\GUNI\SEM 6\React\Practical_6\profile-card> docker run -p 3000:3000 -it -d profile-app:build 1dd17e6fcfafb3f739b3609627bcba0915359bd810091f470a58c8bea55e092e1
 - F:\GUNI\SEM 6\React\Practical_6\profile-card> |
- Bottom Status Bar:** Shows "Ln 17, Col 1" and other system information.
- Taskbar:** At the very bottom, the taskbar includes icons for Search, File Explorer, Task List, Docker, Finish Setup, Prettier, and others.



- Now, let's create a Multi-Stage Dockerfile using React-Nginx.

The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Search Bar:** profile-card
- Left Sidebar (Explorer):** PROFILE-CARD folder containing node_modules, public, src, .gitignore, Dockerfile, nginx.conf, package-lock.json, package.json, and README.md.
- Central Area (Editor):** The Dockerfile content is displayed:

```
1 # ----- Build Stage -----
2 FROM node:14-alpine AS builder
3
4 WORKDIR /app
5
6 COPY package.json .
7 RUN npm install
8
9 COPY .
10 ENV NODE_ENV production
11
12 RUN npm build
13
14 # ----- Production Stage -----
15 FROM nginx:stable-alpine AS production
16 ENV NODE_ENV production
17
18 COPY --from=builder /app/build /usr/share/nginx/html
19 RUN ls -latr /usr/share/nginx/html
20
21 COPY nginx.conf /etc/nginx/conf.d/default.conf
22
23 EXPOSE 80
24
25 CMD ["nginx", "-g", "daemon off;"]
```
- Bottom Status Bar:** Ln 19, Col 35, Spaces: 4, UTF-8, CRLF, Dockerfile, Finish Setup, Prettier.
- Bottom Icons:** Weather (27°C, Cloudy), Search, Home, File, Find, Replace, Undo, Redo, ENG IN, Network, Battery, 11:16, Date (29-06-2023).

- Build the Docker Image.

The screenshot shows the VS Code interface with a Dockerfile open in the Explorer. The terminal tab is active, displaying the output of a docker build command. The build process is shown step-by-step, from loading the Dockerfile to transferring files and exporting layers. The status bar at the bottom indicates the file is 112 lines long, has 18 spaces, and is using UTF-8 encoding.

```
1 # ----- Build Stage -----
2 FROM node:14-alpine AS builder
3
4 WORKDIR /app
5
6 COPY package.json .
7 RUN npm install

F:\GUNI\SEM 6\React\Practical_6\profile-card$ docker build --no-cache . -t profile-app:latest
[+] Building 102.2s (16/16) FINISHED
=> [internal] load build definition from Dockerfile
=> [internal] load 30B
=> [internal] load metadata for docker.io/library/node:14-alpine
=> [internal] load metadata for docker.io/library/nginx:stable-alpine
=> [internal] load .dockerignore
=> [internal] transfer context: 2B
=> [builder 1/6] FROM docker.io/library/node:14-alpine@sha256:434215b487a329c9e867202ff89e704d3a7e554822e07f3e0c0f9e606121b33
=> [internal] resolve docker.io/library/node:14-alpine@sha256:434215b487a329c9e867202ff89e704d3a7e554822e07f3e0c0f9e606121b33
=> CACHED [production 1/4] FROM docker.io/library/nginx:stable-alpine@sha256:aed997342488e51764ff1f2146835ecad42b5f994081fa6631cc5d79240891ec9
=> [internal] resolve docker.io/library/nginx:stable-alpine@sha256:aed997342488e51764ff1f2146835ecad42b5f994081fa6631cc5d79240891ec9
=> [internal] load build context
=> [internal] transfer context: 3.05B
=> CACHED [builder 2/6] WORKDIR /app
=> [builder 3/6] COPY package.json .
=> [builder 4/6] RUN npm install
=> [builder 5/6] COPY .
=> [builder 6/6] RUN npm run build
=> [production 2/4] COPY --from=builder /app/build /usr/share/nginx/html
=> [production 3/4] RUN ls -latr /usr/share/nginx/html
=> [production 4/4] COPY nginx.conf /etc/nginx/conf.d/default.conf
=> exporting to image
=> => exporting layers
=> => exporting manifest sha256:2b1d65e4cc573e1afdb72c552c3189e017cc3e722bbae0af1fc3ff6e9e9c98e
=> => exporting config sha256:d7f707906880043dd60f778383d6f9978bfe50ab38174eb79c2b663c4b683b
=> => exporting attestation manifest sha256:d28b138bc479defcfa60b0393b6dad27a64a6a0862f5fc8d8c498a3b4fa5b
=> => exporting manifest list sha256:6235b0417a8f76ae6b3c9533fa7884c6df7a8d2cca70d7fe94509b0dcfbfd
=> => naming to docker.io/library/profile-app:latest
=> => unpacking to docker.io/library/profile-app:latest
○ F:\GUNI\SEM 6\React\Practical_6\profile-card$
```

- Now, check the size of the image recently created “profile-app:latest” using ‘docker images’ command.

The screenshot shows the VS Code interface with the Docker extension. The Explorer sidebar shows files like node_modules, public, src, .gitignore, Dockerfile, nginx.conf, package-lock.json, package.json, and README.md. The Dockerfile tab is active, displaying:

```

1 # ----- Build Stage -----
2 FROM node:14-alpine AS builder
3
4 WORKDIR /app
5
6 COPY package.json .
7 RUN npm install

```

The terminal tab shows the command `docker images` output:

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
profile-app	latest	6235b04177a8	About a minute ago	76.6MB
profile-app	build	3150baef4f35	43 minutes ago	1.32GB
aaryan003/myapp	latest	761fe38e5d54	2 hours ago	216MB
my-custom-linux	latest	761fe38e5d54	2 hours ago	216MB
nginx	latest	dc53cf825a10	4 days ago	279MB
alpine	latest	8af15fffb675	4 weeks ago	12.8MB
ubuntu	latest	b59d21599a2b	4 weeks ago	117MB

The status bar at the bottom indicates the terminal has 112 lines, 18 columns, and is in UTF-8 mode. It also shows icons for Dockerfile, Finish Setup, and Prettier.

We can see the image size was reduced to 77MB previously from 1.32GB with the help of **Multi-Stage Build**.

The screenshot shows the VS Code interface with the Docker extension. The Explorer sidebar shows files like node_modules, public, src, .gitignore, Dockerfile, nginx.conf, package-lock.json, package.json, and README.md. The Dockerfile tab is active, displaying:

```

1 # ----- Build Stage -----
2 FROM node:14-alpine AS builder
3
4 WORKDIR /app
5
6 COPY package.json .
7 RUN npm install

```

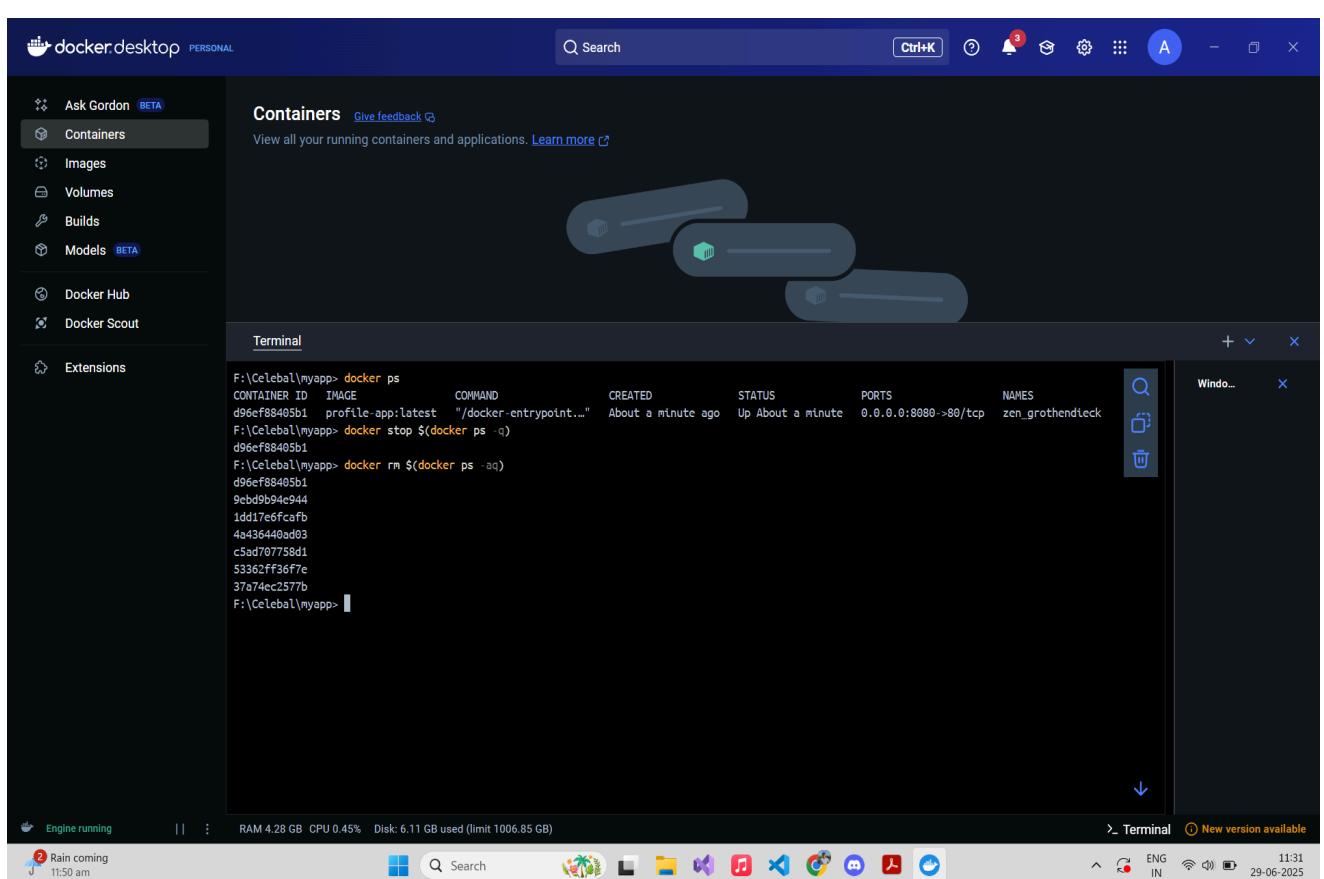
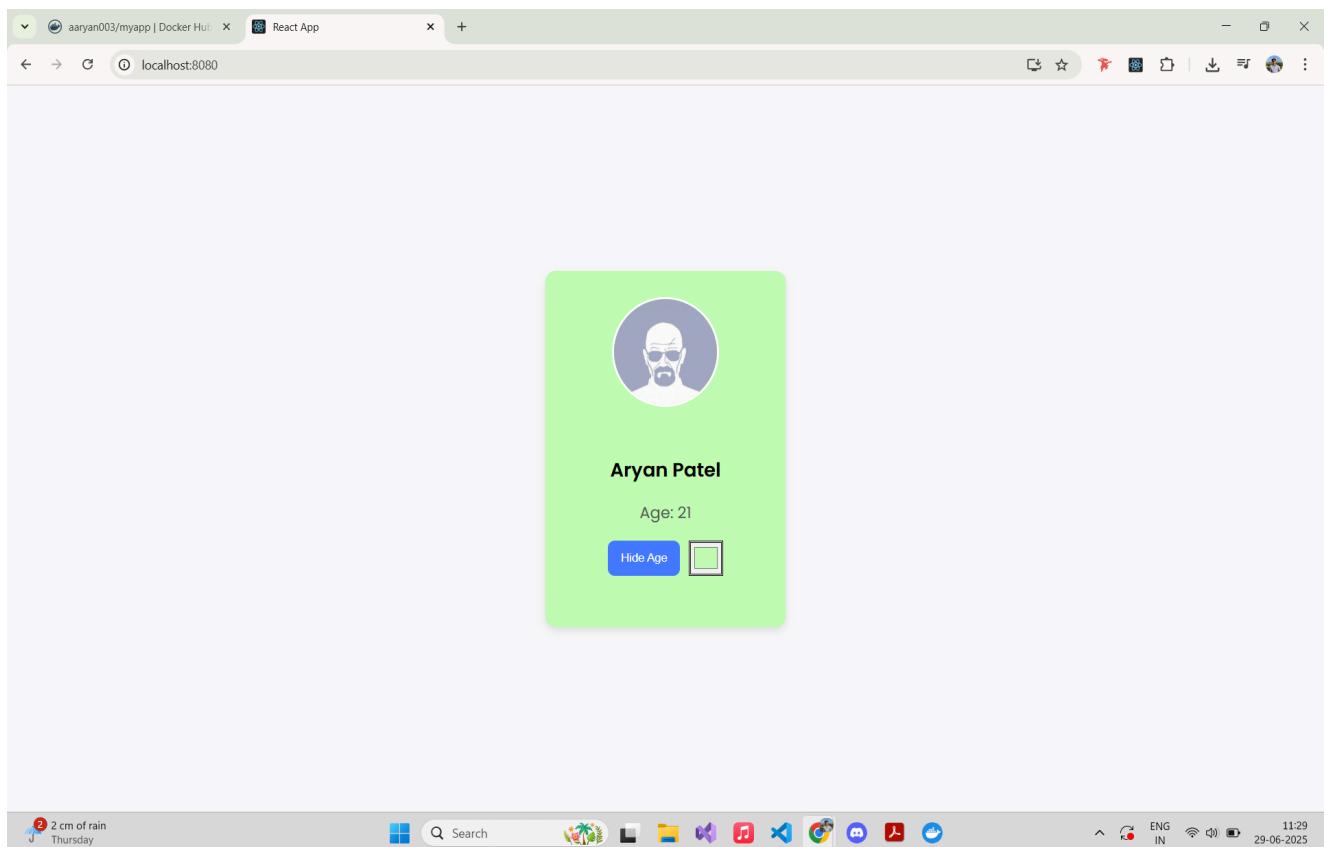
The terminal tab shows the command `docker run -p 8000:80 profile-app:latest` output:

```

$ docker run -p 8000:80 profile-app:latest
/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: /etc/nginx/conf.d/default.conf differs from the packaged version
/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
2025/06/29 05:59:03 [notice] 1#1: using the "epoll" event method
2025/06/29 05:59:03 [notice] 1#1: nginx/1.28.0
2025/06/29 05:59:03 [notice] 1#1: built by gcc 14.2.0 (Alpine 14.2.0)
2025/06/29 05:59:03 [notice] 1#1: OS: Linux 6.6.87-1-microsoft-standard-wsl2
2025/06/29 05:59:03 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2025/06/29 05:59:03 [notice] 1#1: start worker processes
2025/06/29 05:59:03 [notice] 1#1: start worker process 29
2025/06/29 05:59:03 [notice] 1#1: start worker process 30
2025/06/29 05:59:03 [notice] 1#1: start worker process 31
2025/06/29 05:59:03 [notice] 1#1: start worker process 32
2025/06/29 05:59:03 [notice] 1#1: start worker process 33
2025/06/29 05:59:03 [notice] 1#1: start worker process 34
2025/06/29 05:59:03 [notice] 1#1: start worker process 35
2025/06/29 05:59:03 [notice] 1#1: start worker process 36
2025/06/29 05:59:03 [notice] 1#1: start worker process 37
2025/06/29 05:59:03 [notice] 1#1: start worker process 38
2025/06/29 05:59:03 [notice] 1#1: start worker process 39
2025/06/29 05:59:03 [notice] 1#1: start worker process 40
2025/06/29 05:59:03 [notice] 1#1: start worker process 41
2025/06/29 05:59:03 [notice] 1#1: start worker process 42
2025/06/29 05:59:03 [notice] 1#1: start worker process 43
2025/06/29 05:59:03 [notice] 1#1: start worker process 44
2025/06/29 05:59:03 [notice] 1#1: start worker process 45
2025/06/29 05:59:03 [notice] 1#1: start worker process 46
2025/06/29 05:59:03 [notice] 1#1: start worker process 47
2025/06/29 05:59:03 [notice] 1#1: start worker process 48

```

The status bar at the bottom indicates the terminal has 112 lines, 18 columns, and is in UTF-8 mode. It also shows icons for Dockerfile, Finish Setup, and Prettier.



4. Create a docker image from multiple methods like Dockerfile, running containers.

Method 1: Create Image from Dockerfile

The screenshot shows the Visual Studio Code interface with the following details:

- Explorer:** Shows a folder named "MYAPP" containing "Dockerfile" and "index.html".
- Dockerfile:** Content:

```
1 FROM nginx:alpine
2 COPY index.html /usr/share/nginx/html/index.html
```
- Terminal:** Output of the command `docker build -t nginx-app .`, showing the build process for "nginx-app":

```
[+] Building 9.1s (8/8) FINISHED
[+] Loading build definition from Dockerfile
[+] Transferring dockerfile: 10B
[+] Loading metadata for docker.io/library/nginx:alpine
[+] Auth library/nginx:pull token for registry-1.docker.io
[+] Loading .dockerignore
[+] Transferring context: 2B
[+] Loading build context
[+] Transferring context: 23B
[+] Resolving docker.io/library/nginx:alpine@sha256:b2e814d28359e77hd0aa5fed1939620075e4ffapeb20423cc557b375bd5c14ad
[+] sha256:b3b7062d09e028c442f2b05e7628654199b23a69a2b9e0cd26911a5200bc86 1.81MB / 1.81MB
[+] sha256:63dd02ad85b92ed611dalb1943243996afer5e007b59a724ea6362a0552 1.21kB / 1.21kB
[+] sha256:b55ed707b2d0ef46f44870cabb5743ff31daef10a58ff7d978939788789c62943 1.40kB / 1.40kB
[+] sha256:92971aeab101e5e1df5c7962c2399dc7e0c34c09f356f428f4256e04039cef2 16.78MB / 16.78MB
[+] Extracting sha256:3b7062d09e028c442f2b05e76286341d923a69a429sec3d226911a5200bc86
[+] sha256:a9ff9ba1741d1f2a4a5df1b4dpf0993118ch1a9201f23a01cb295e0274768 955B / 955B
[+] sha256:2c127093dfc748d5aa28bdf3d6936de033ee9cb643eab75d6070e4333fc4 404B / 404B
[+] sha256:fb746e72516f99f0bf4f5204c47f8c2aa325a49d075769800c42548fe722b25d 627B / 627B
[+] Extracting sha256:fb746e72516f99f0bf4f5204c47f8c2aa325a49d075769800c42548fe722b25d
[+] Extracting sha256:a9ff9ba1741d1f2a4a5df1b4dpf0993118ch1a9201f23a01cb295e0274768
[+] Extracting sha256:2c127093dfc748d5aa28bdf3d6936de033ee9cb643eab75d6070e4333fc4
[+] Extracting sha256:63dd02ad85b92ed611dalb1943243996afer5e007b59a724ea6362a0552
[+] Extracting sha256:b55ed707b2d0ef46f44870cabb5743ff31daef10a58ff7d978939788789c62943
[+] [2/2] Copy index.html /usr/share/nginx/html/index.html
[+] Exporting to image
```
- Bottom Status Bar:** Shows the date (29-06-2025), time (11:40), and weather (Cloudy, 28°C).

- Verify the Image Exists using 'docker images'.

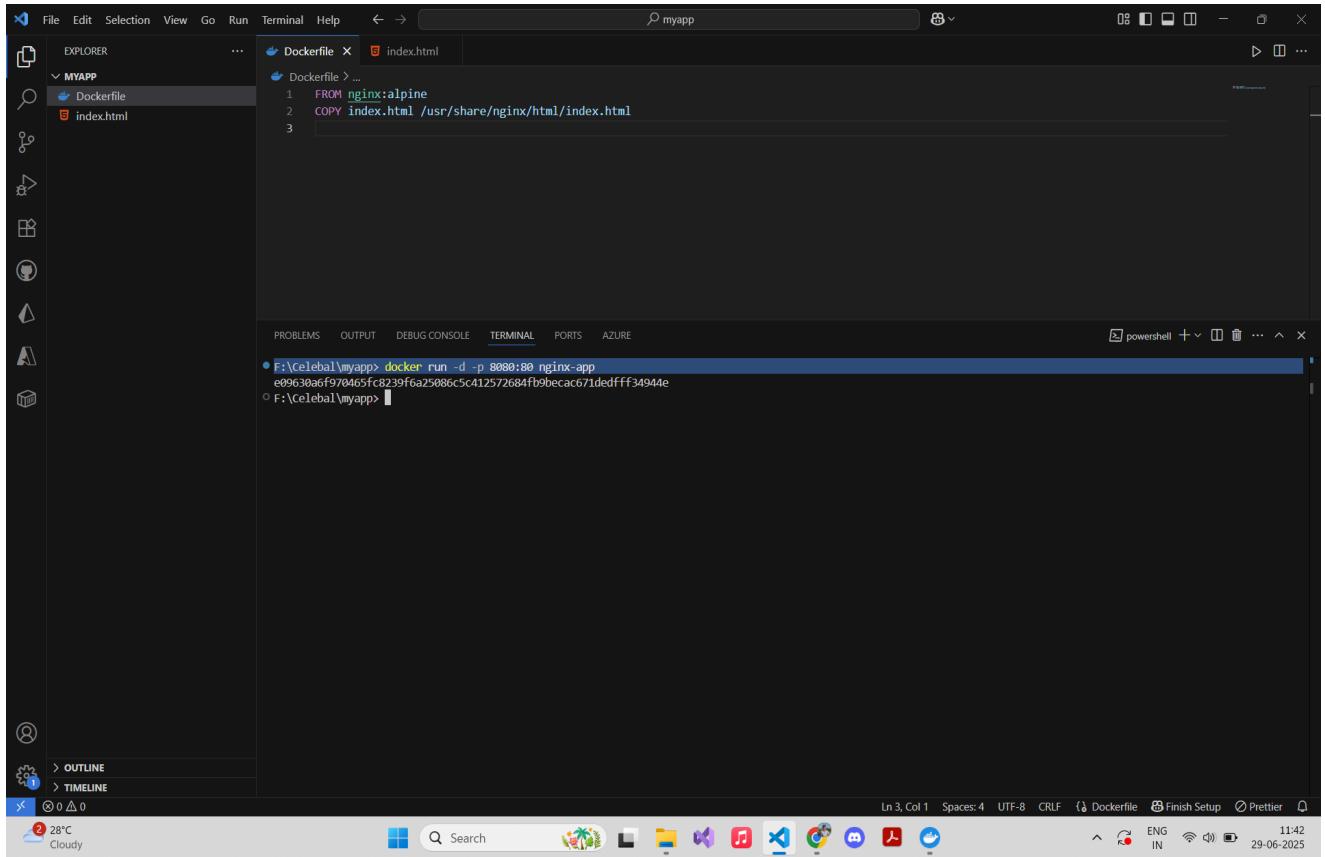
The screenshot shows the Visual Studio Code interface with the following details:

- Explorer:** Shows a folder named "MYAPP" containing "Dockerfile" and "index.html".
- Dockerfile:** Content:

```
1 FROM nginx:alpine
2 COPY index.html /usr/share/nginx/html/index.html
```
- Terminal:** Output of the command `docker images`, listing available Docker images:

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
nginx-app	latest	69ff1ed2b227	41 seconds ago	79.3MB
profile-app	latest	6235b0a177a8	17 minutes ago	76.6MB
profile-app	build	3156ba74f35	59 minutes ago	1.32GB
aaryan03/myapp	latest	761fe39e5d4	2 hours ago	216MB
my-custom-linux	latest	761fe39e5d4	2 hours ago	216MB
nginx	latest	dc53c812a10	4 days ago	279MB
alpine	latest	8af159ff675	4 weeks ago	12.8MB
ubuntu	latest	b59d21599a2b	4 weeks ago	117MB
- Bottom Status Bar:** Shows the date (29-06-2025), time (11:42), and weather (Cloudy, 28°C).

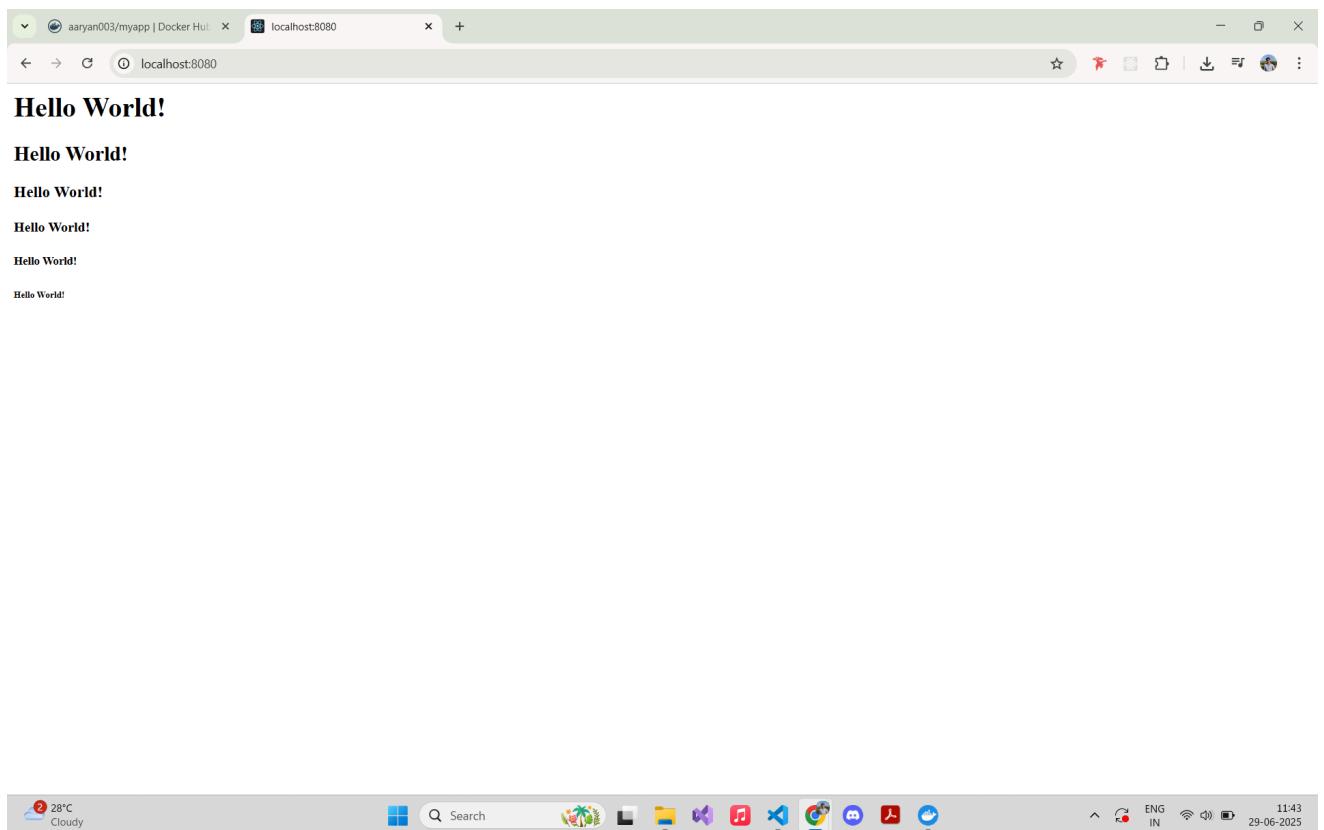
- Run and Test the Image.



The screenshot shows the Visual Studio Code interface. In the Explorer sidebar, there is a folder named 'MYAPP' containing a 'Dockerfile' and an 'index.html' file. The 'Dockerfile' content is:

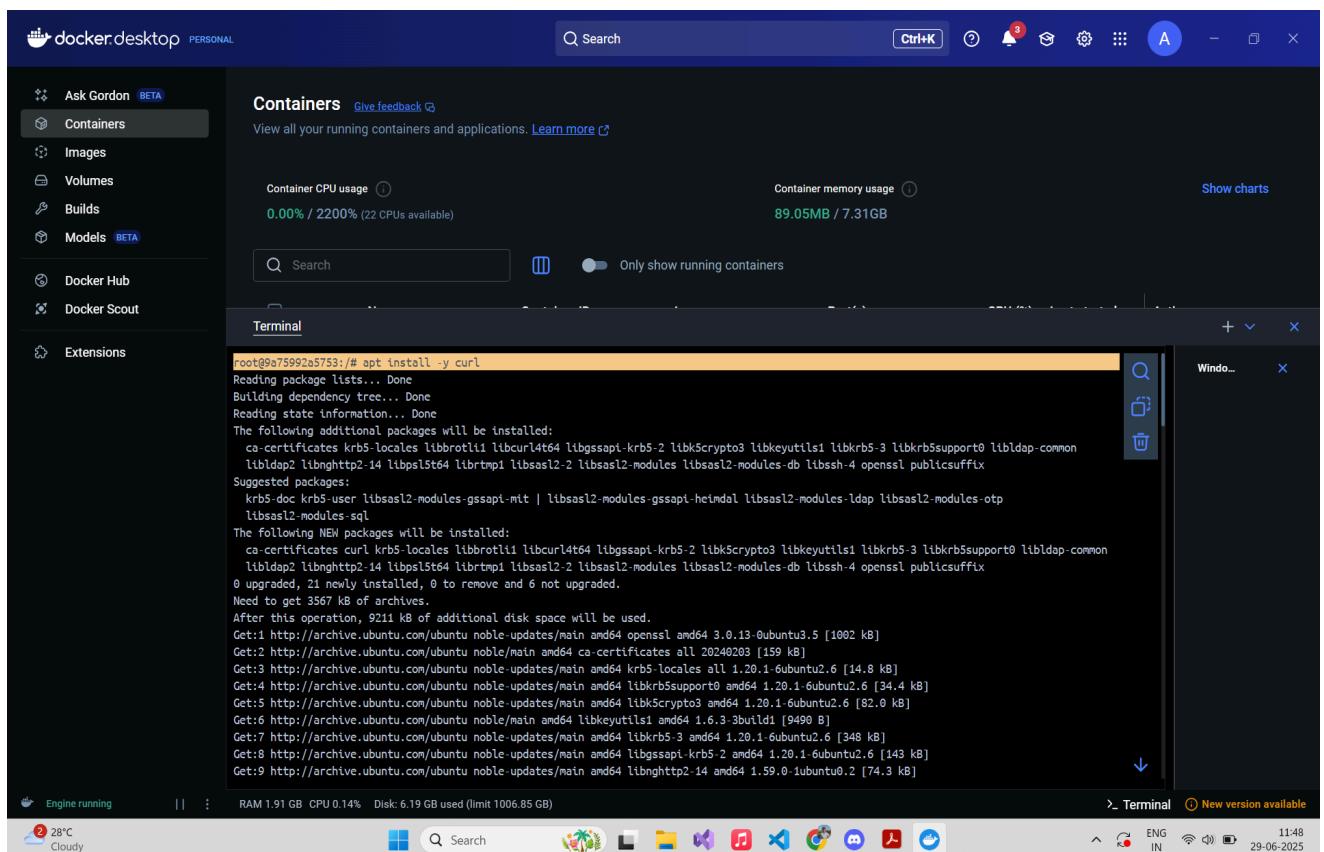
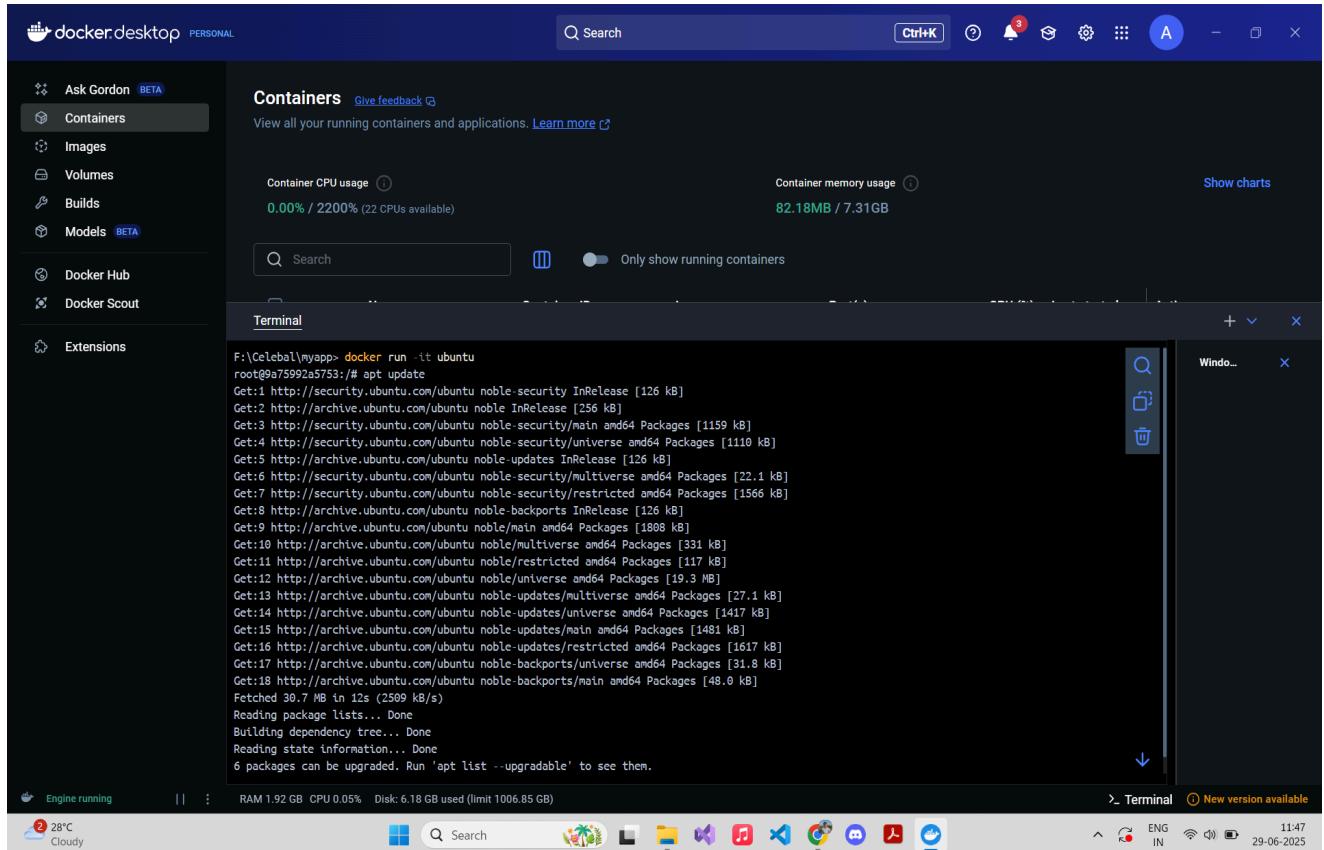
```
FROM nginx:alpine
COPY index.html /usr/share/nginx/html/index.html
```

In the Terminal tab, the command `docker run -d -p 8080:80 nginx-app` is run, resulting in the container ID `e99630a6f970465fc8239f6a25886c5c412572684fb9becac671dedffff34944e`. The status bar at the bottom indicates the file is 3 lines long, 4 spaces wide, using UTF-8 encoding, and includes icons for Dockerfile, Finish Setup, and Prettier.

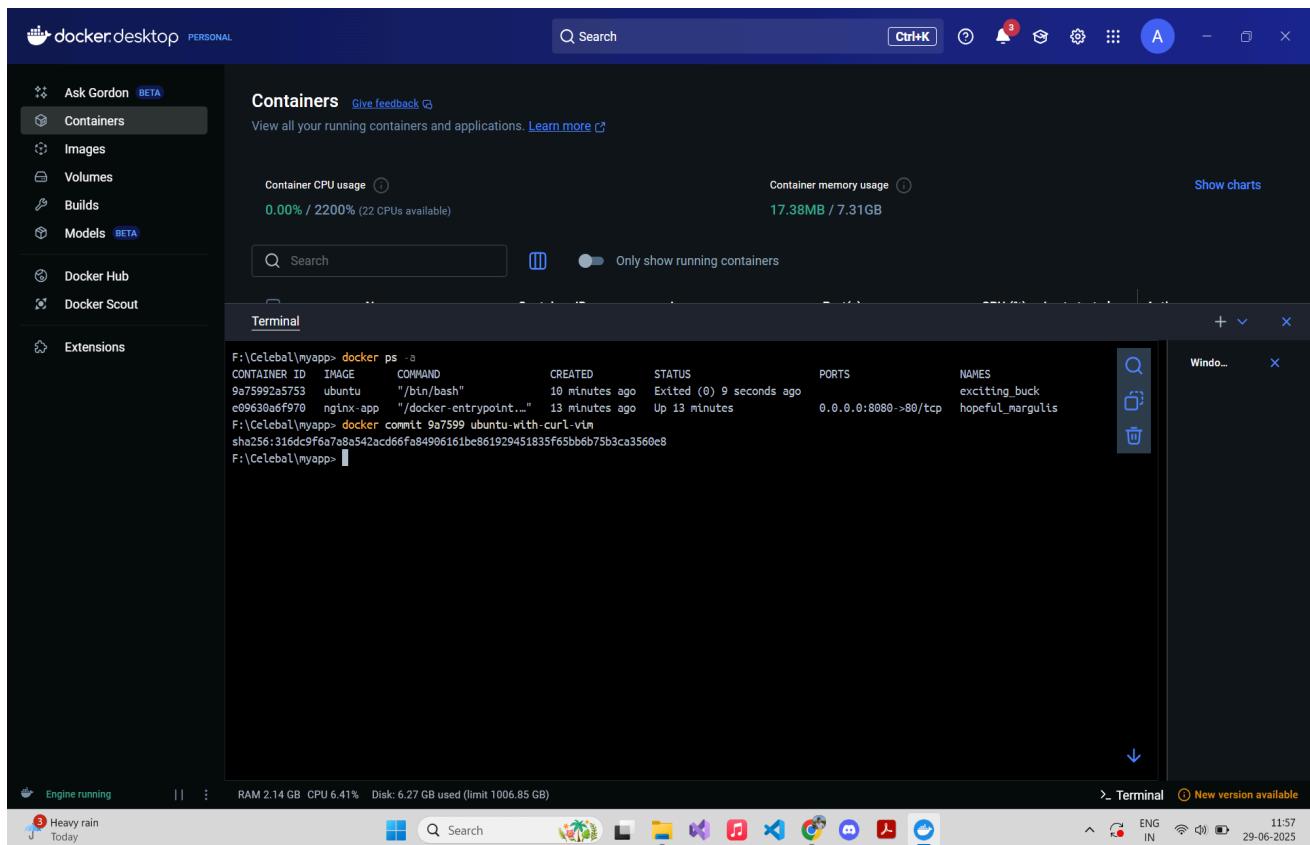


Method 2: Create Image from a Running Container

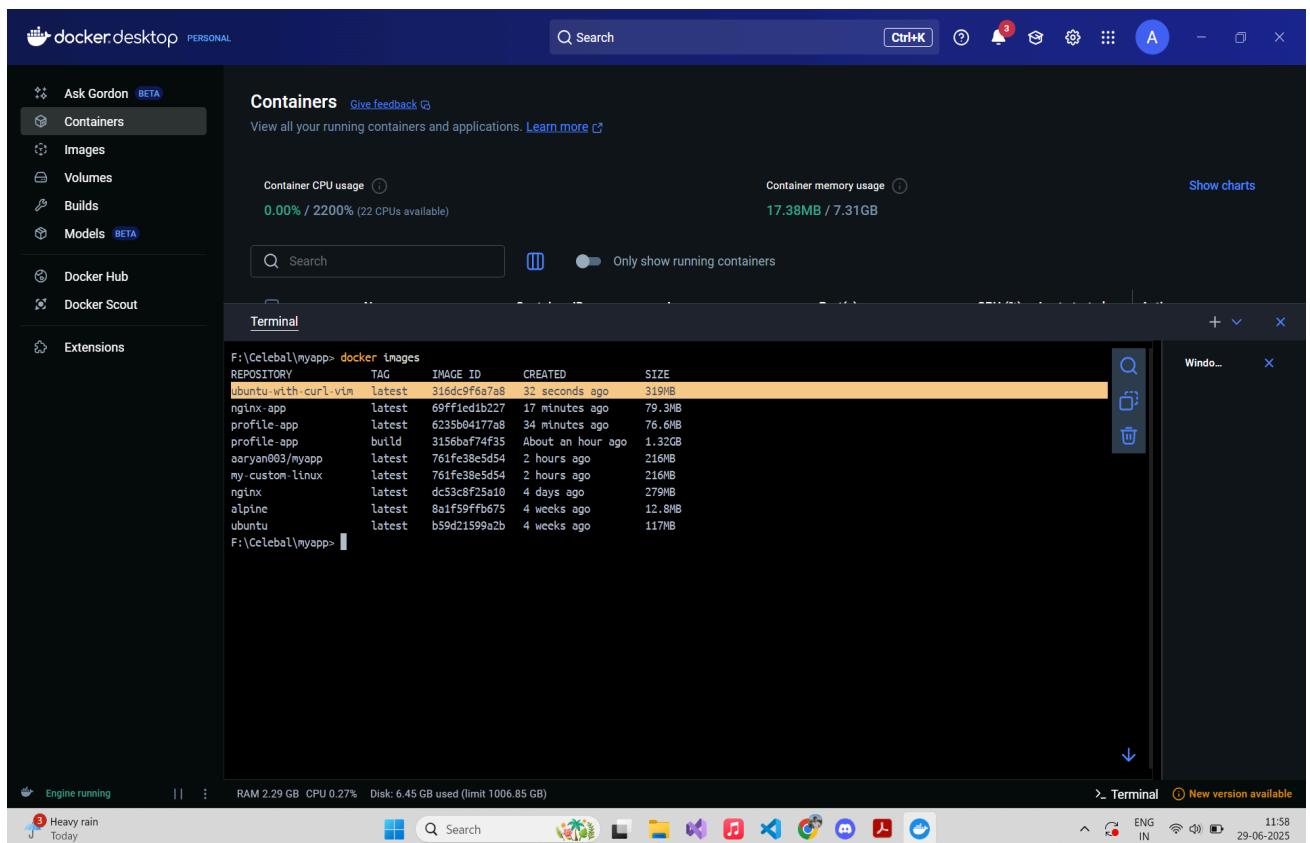
- Run a Base Container and Modify It.



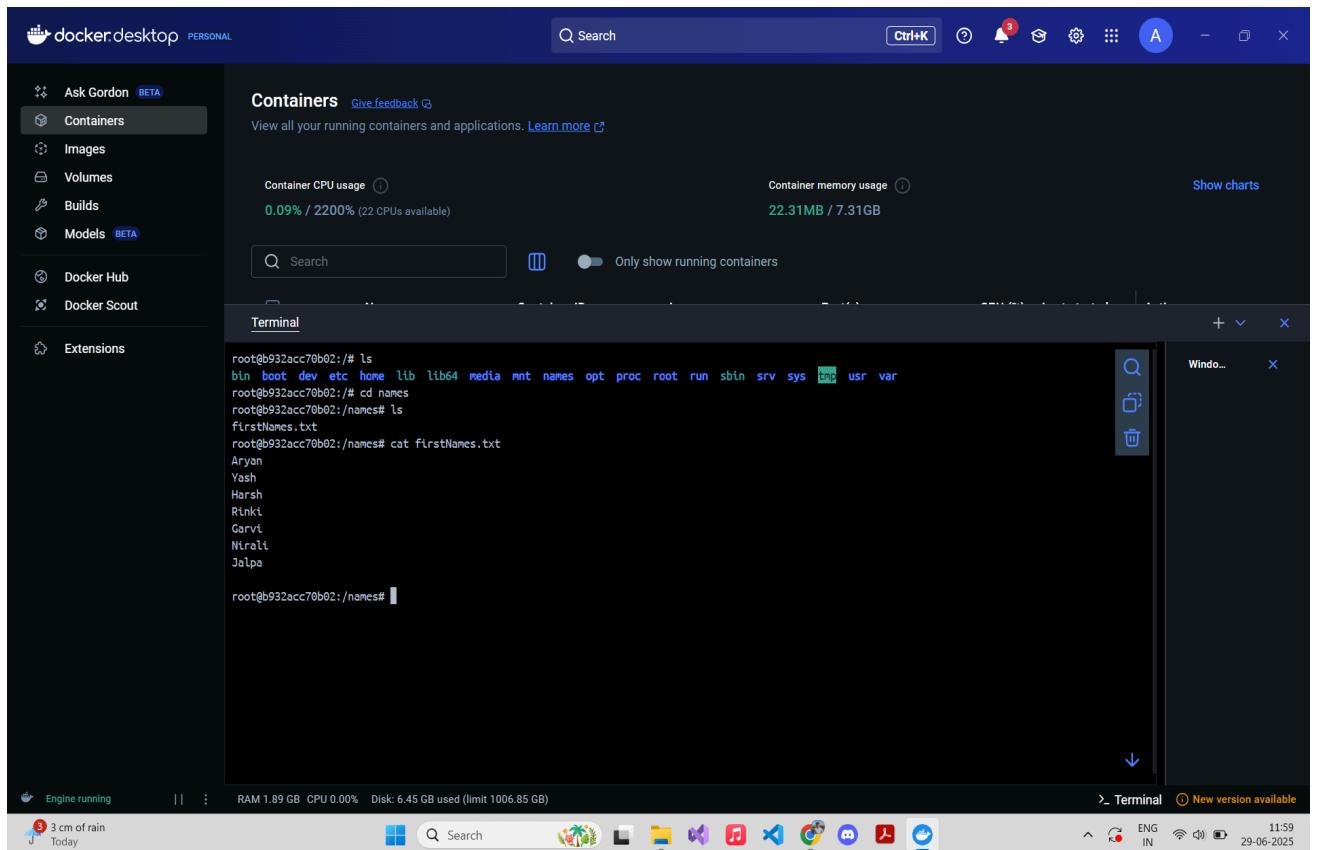
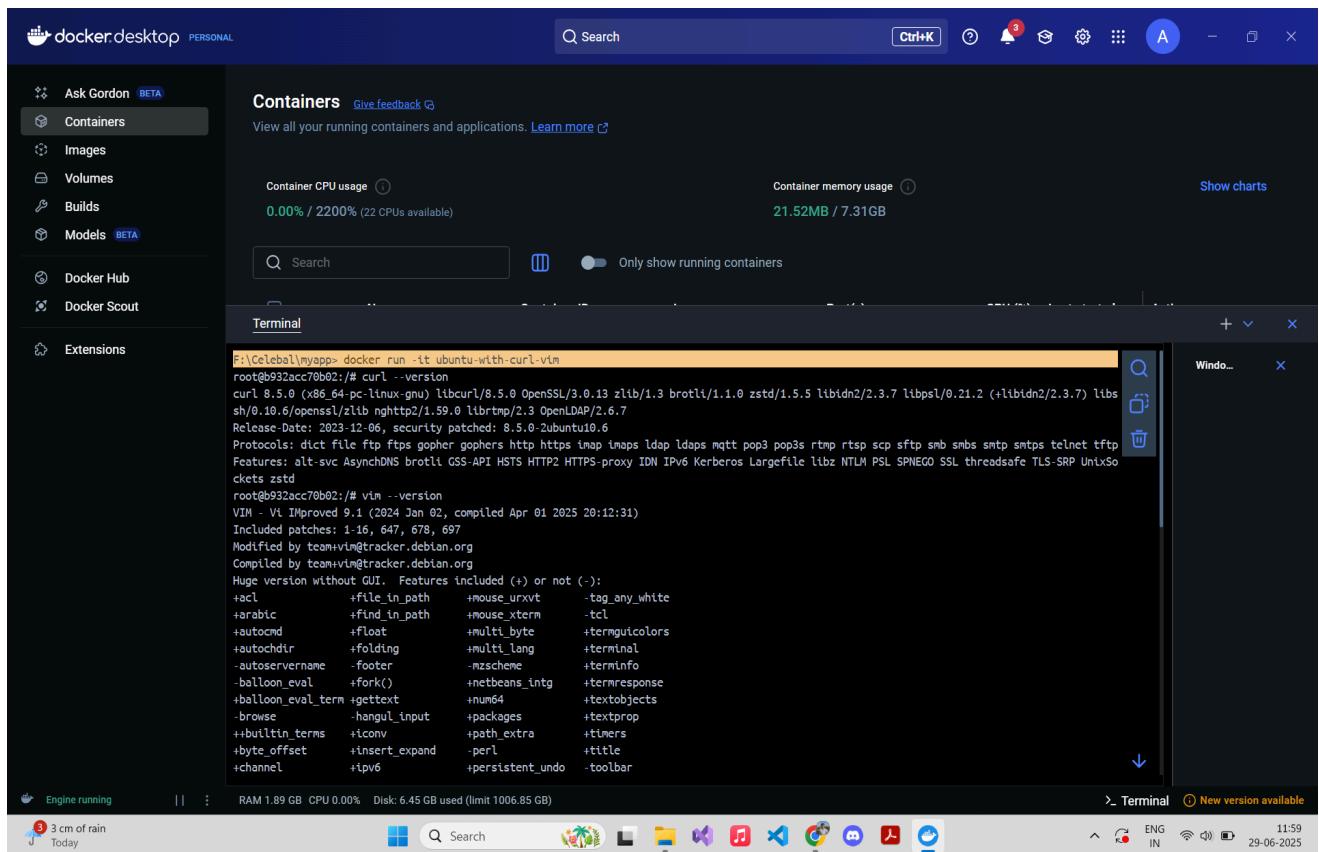
- Commit the Container to a New Image.



- Verify Image Was Created using “docker images”.



- Test New Image using “docker run” command.

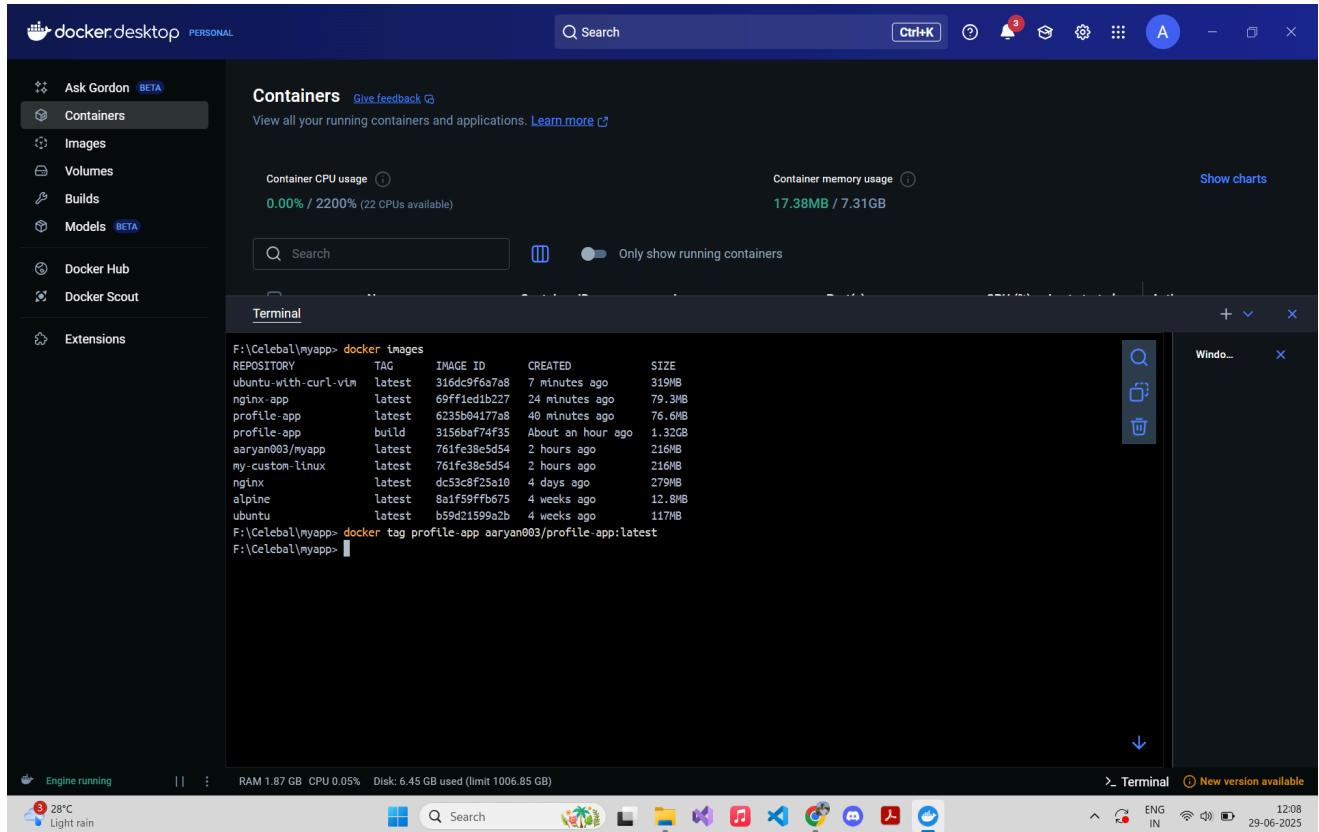


5. Push and pull image to Docker hub and ACR

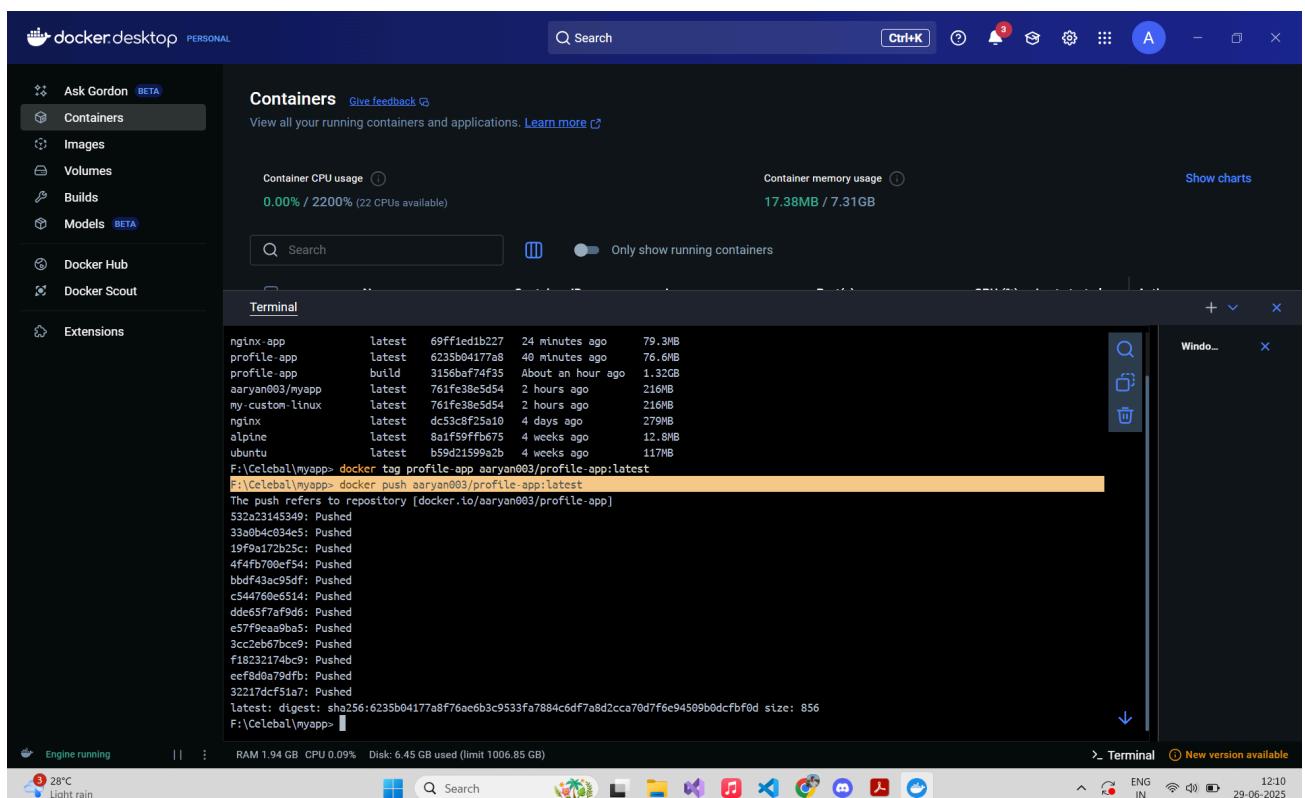
Steps :

1. Push & Pull Docker Image to/from Docker Hub

- Log in to Docker Hub using the “docker login” command and then tag any of your local images (eg : profile-app:latest).



- Push Image to Docker Hub.



- Verify on Docker Hub Website.

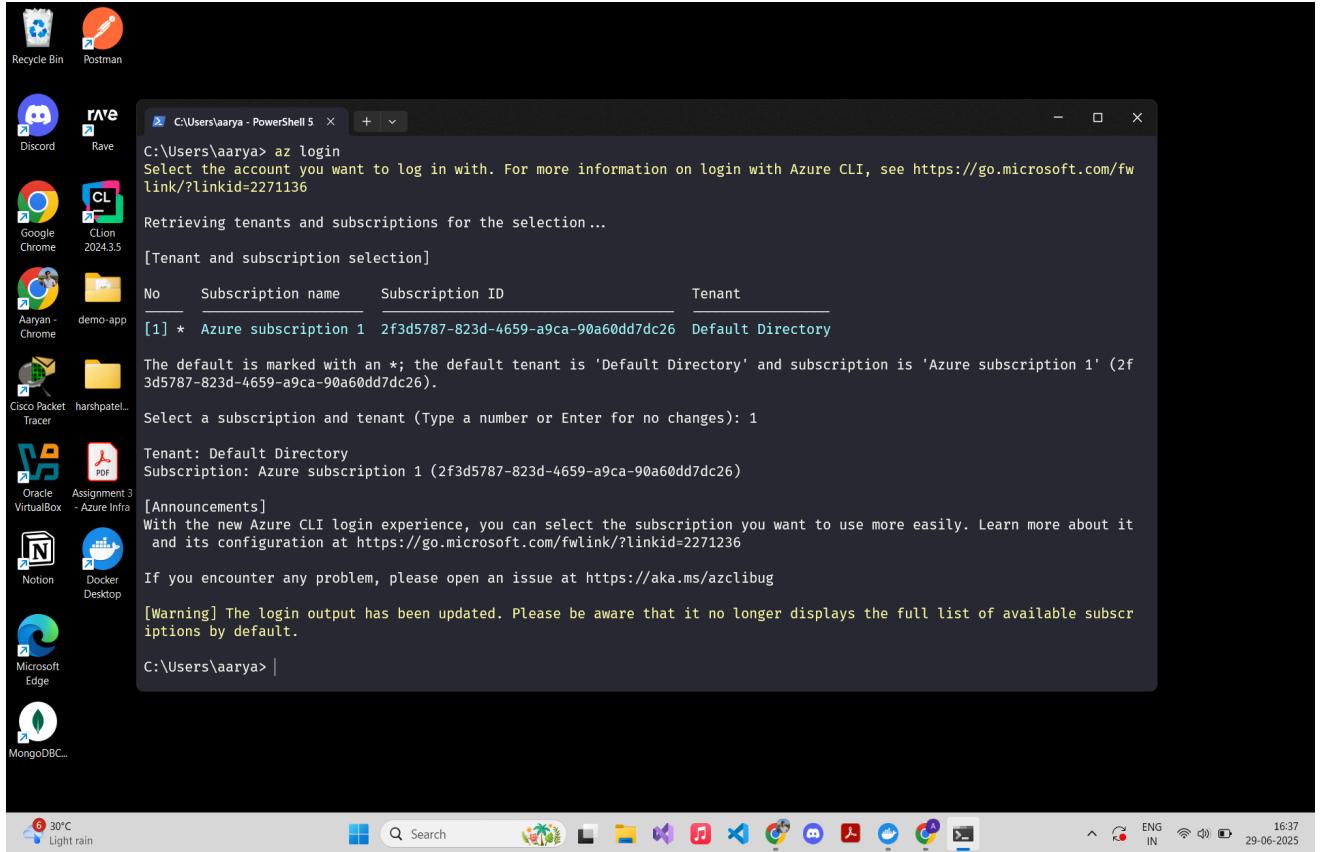
The screenshot shows the Docker Hub website interface. On the left, there's a sidebar with options like 'Repositories', 'Collaborations', 'Settings', 'Default privacy', 'Notifications', 'Billing', 'Usage', 'Pulls', and 'Storage'. The main area displays the 'profile-app' repository under 'aaryan003'. It shows the 'latest' tag with details: INDEX DIGEST (sha256:6235b04177a8f76ae6b3c9533fa7884c6df7a8d2cca70d7f6e94509b0dcfb0d), OS/ARCH (linux/amd64), COMPRESSED SIZE (21.01 MB), LAST PUSHED (1 minute ago by aaryan003), TYPE (Image), and MANIFEST DIGEST (sha256:2b1d65e4cc573e1af4b72c552c31806017cc3e722bbbae0a1fc3ff6e9ec98e). Below this, the 'Image Layers' tab is selected, showing a list of layers with their commands and sizes. The first layer is 'ADD alpine-minirootfs-3.21.3-x86_64.tar.gz / # buildkit 3.47 MB'. The terminal tab at the bottom shows the command 'docker pull aaryan003/profile-app:latest' being run.

- Pull from Docker Hub on Any Machine.

The screenshot shows the Docker Desktop application window. The left sidebar includes 'Containers', 'Images', 'Volumes', 'Builds', 'Models', 'Docker Hub', 'Docker Scout', and 'Extensions'. The main area has a 'Containers' section with CPU and memory usage stats, and a 'Terminal' section where the command 'docker pull aaryan003/profile-app:latest' is being executed. The terminal output shows the image is up-to-date and provides the digest. The status bar at the bottom indicates 'Engine running' and system information like RAM, CPU, and Disk usage.

2. Push & Pull Docker Image to/from Azure Container Registry (ACR).

- Create Azure Container Registry.



A screenshot of a Windows desktop environment. In the center is a PowerShell window titled 'C:\Users\aaarya - PowerShell 5'. The command 'az login' is being run, prompting the user to select an account. The window shows a table of subscriptions and their details, including the default subscription and tenant. The desktop background is dark, and the taskbar at the bottom shows various pinned icons like Recycle Bin, Postman, Discord, Rave, Google Chrome, Clion, Aayan - Chrome, demo-app, Cisco Packet Tracer, Oracle VirtualBox, Assignment 3 - Azure Infra, Notion, Docker Desktop, Microsoft Edge, and MongoDB... The system tray indicates it's 30°C and light rain outside, and the date is 29-06-2025.

```
C:\Users\aaarya> az login
Select the account you want to log in with. For more information on login with Azure CLI, see https://go.microsoft.com/fwlink/?linkid=2271136
Retrieving tenants and subscriptions for the selection ...
[Tenant and subscription selection]
No Subscription name Subscription ID Tenant
[1] * Azure subscription 1 2f3d5787-823d-4659-a9ca-90a60dd7dc26 Default Directory

The default is marked with an *; the default tenant is 'Default Directory' and subscription is 'Azure subscription 1' (2f3d5787-823d-4659-a9ca-90a60dd7dc26).

Select a subscription and tenant (Type a number or Enter for no changes): 1

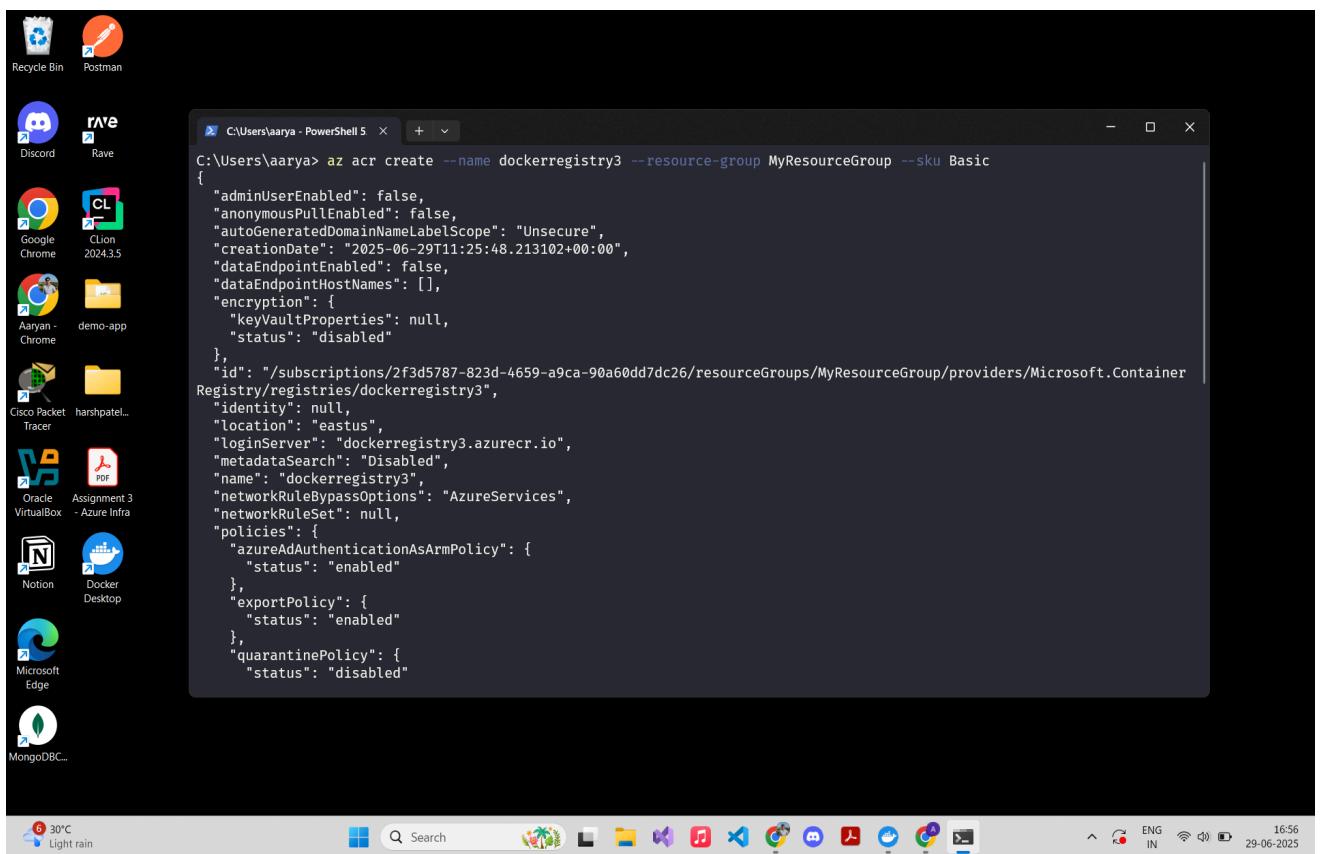
Tenant: Default Directory
Subscription: Azure subscription 1 (2f3d5787-823d-4659-a9ca-90a60dd7dc26)

[Announcements]
With the new Azure CLI login experience, you can select the subscription you want to use more easily. Learn more about it and its configuration at https://go.microsoft.com/fwlink/?linkid=2271236

If you encounter any problem, please open an issue at https://aka.ms/azclibug

[Warning] The login output has been updated. Please be aware that it no longer displays the full list of available subscriptions by default.

C:\Users\aaarya> |
```

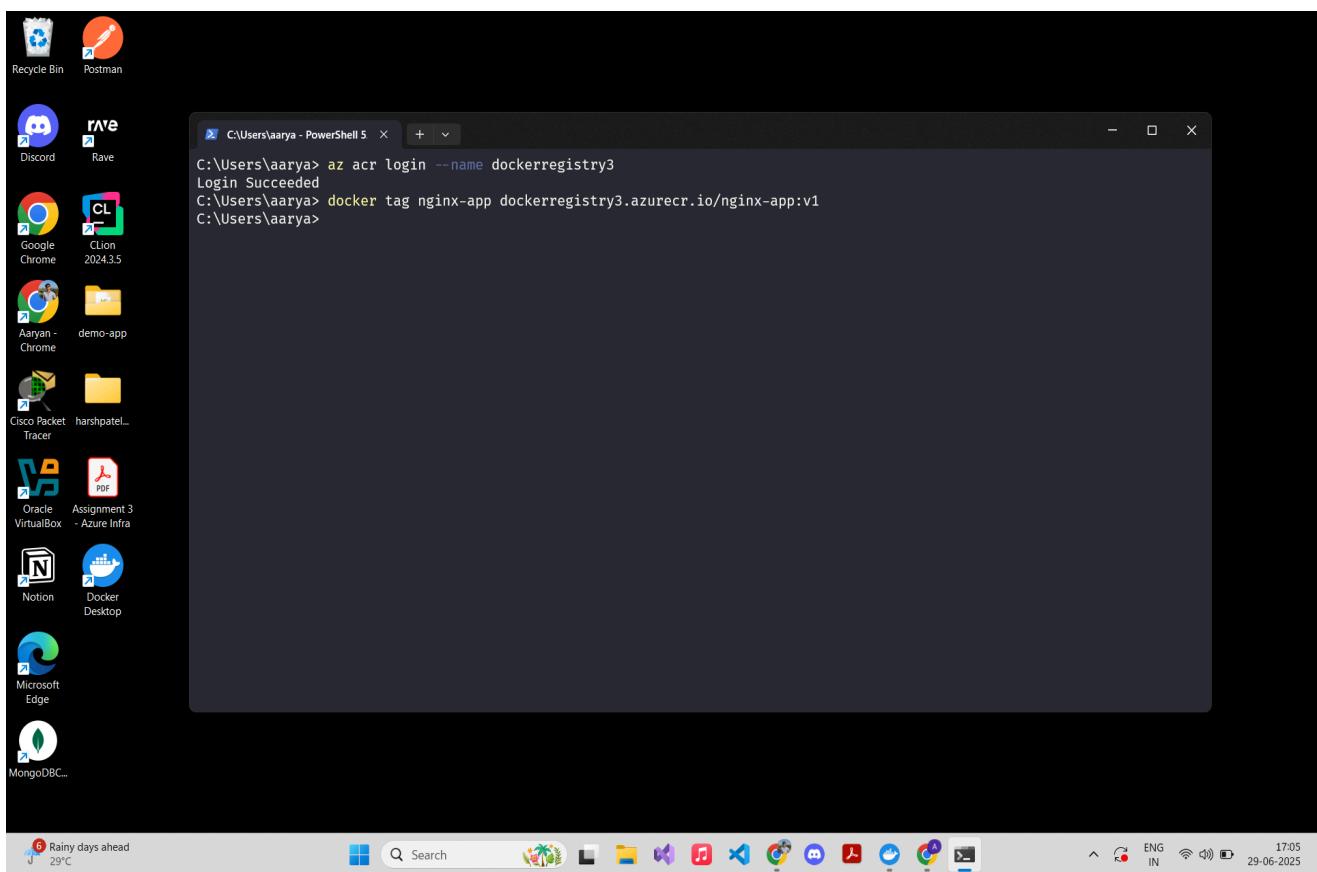


A screenshot of a Windows desktop environment, identical to the one above, showing a PowerShell window titled 'C:\Users\aaarya - PowerShell 5'. This time, the command 'az acr create --name dockerregistry3 --resource-group MyResourceGroup --sku Basic' is being run. The window displays the JSON response from the Azure Container Registry creation, detailing the registry's properties such as location, login server, and policies. The desktop background, taskbar, and system tray are consistent with the previous screenshot.

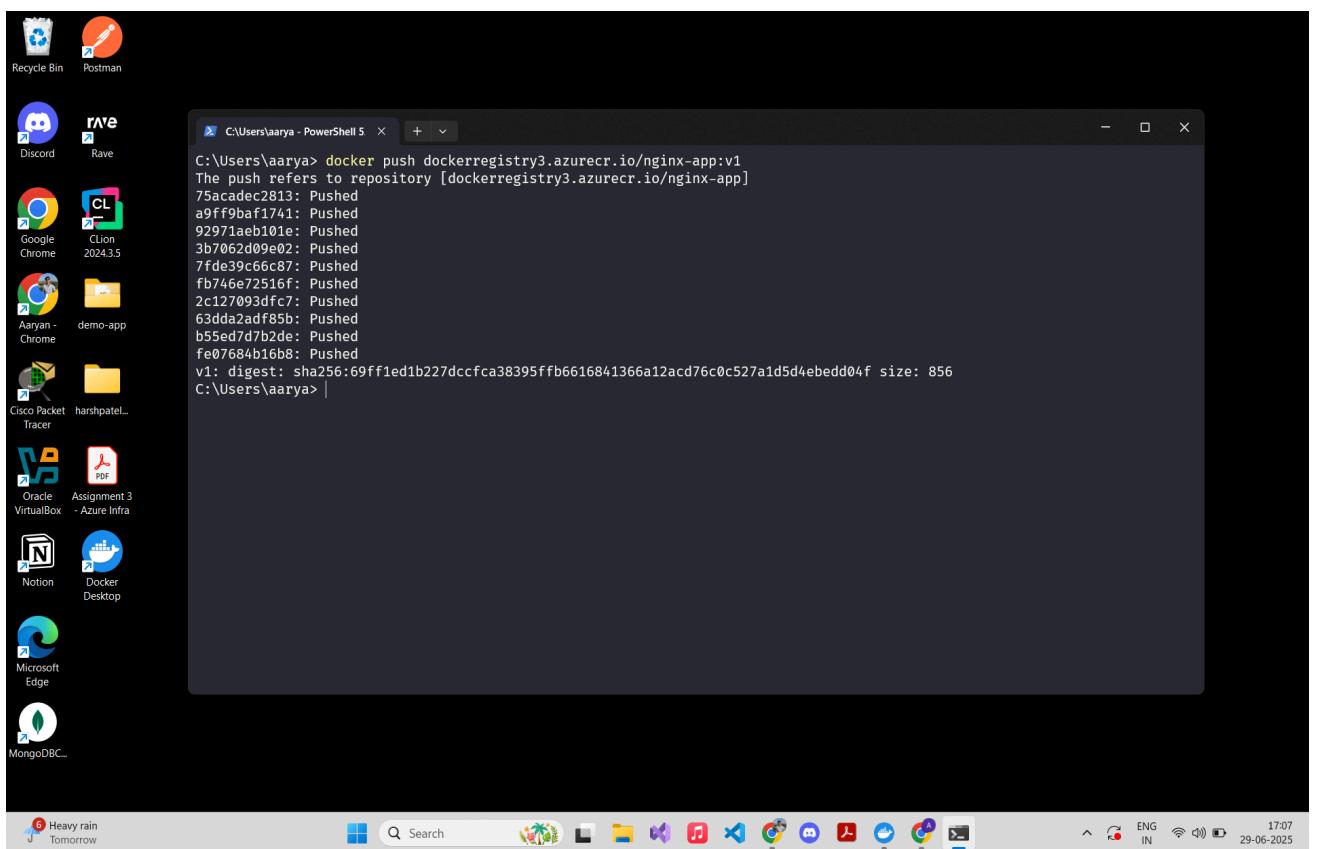
```
C:\Users\aaarya> az acr create --name dockerregistry3 --resource-group MyResourceGroup --sku Basic
{
  "adminUserEnabled": false,
  "anonymousPullEnabled": false,
  "autoGeneratedDomainNameLabelScope": "Unsecure",
  "creationDate": "2025-06-29T11:25:48.213102+00:00",
  "dataEndpointEnabled": false,
  "dataEndpointHostNames": [],
  "encryption": {
    "keyVaultProperties": null,
    "status": "disabled"
  },
  "id": "/subscriptions/2f3d5787-823d-4659-a9ca-90a60dd7dc26/resourceGroups/MyResourceGroup/providers/Microsoft.ContainerRegistry/registries/dockerregistry3",
  "identity": null,
  "location": "eastus",
  "loginServer": "dockerregistry3.azurecr.io",
  "metadataSearch": "Disabled",
  "name": "dockerregistry3",
  "networkRuleBypassOptions": "AzureServices",
  "networkRuleSet": null,
  "policies": {
    "azureAdAuthenticationAsArmPolicy": {
      "status": "enabled"
    },
    "exportPolicy": {
      "status": "enabled"
    },
    "quarantinePolicy": {
      "status": "disabled"
    }
  }
}

C:\Users\aaarya> |
```

- Log in to ACR
- Tag Image with ACR Format



- Push to ACR



- Verify in Azure Portal

The screenshot shows the Microsoft Azure portal's Docker registry interface. The left sidebar navigation includes 'Overview', 'Activity log', 'Access control (IAM)', 'Tags', 'Quick start', 'Resource visualizer', 'Events', 'Settings', and 'Services'. Under 'Services', 'Repositories' is selected. The main content area shows the 'nginx-app' repository details. The 'Essentials' section lists the repository name, tag count (1), manifest count (3), last updated date (6/29/2025, 5:07 PM GMT+5:30), and the first tag (v1) with its digest and last modified time. The status bar at the bottom indicates 'Rain to stop In about 1 hour', the date '29-06-2025', and the time '17:09'.

- Pull from ACR on Any Machine

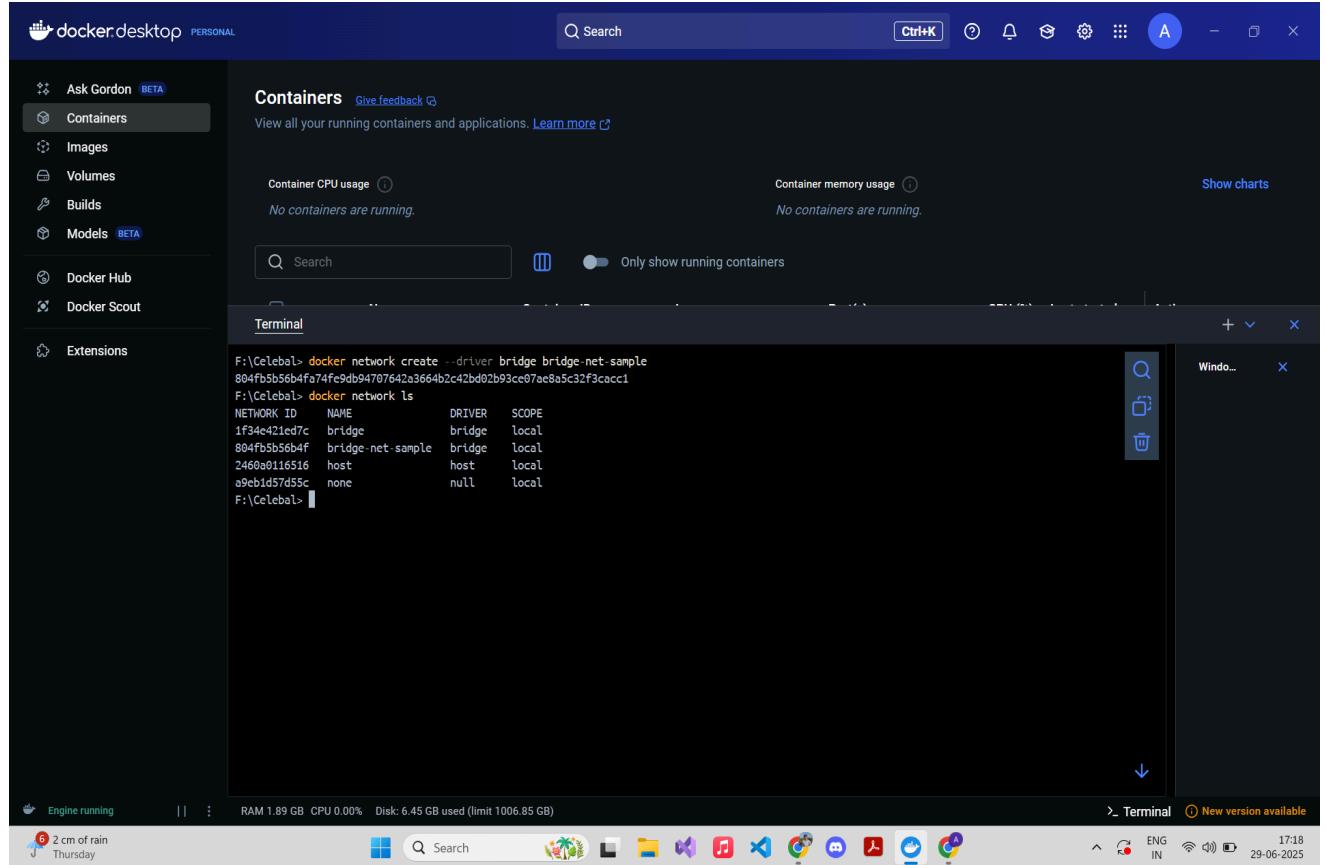
The screenshot shows a Windows desktop environment. A PowerShell window is open in the foreground, displaying the command 'docker pull dockerregistry3.azurecr.io/nginx-app:v1' and its output, which indicates the image is up-to-date. The desktop background is black, and the taskbar contains icons for various applications like Postman, Discord, Rave, Google Chrome, Clion, Notion, Docker Desktop, Microsoft Edge, and MongoDB. The status bar at the bottom shows the weather as '29°C Light rain' and the date as '29-06-2025'.

6. Create a Custom Docker Bridge Network

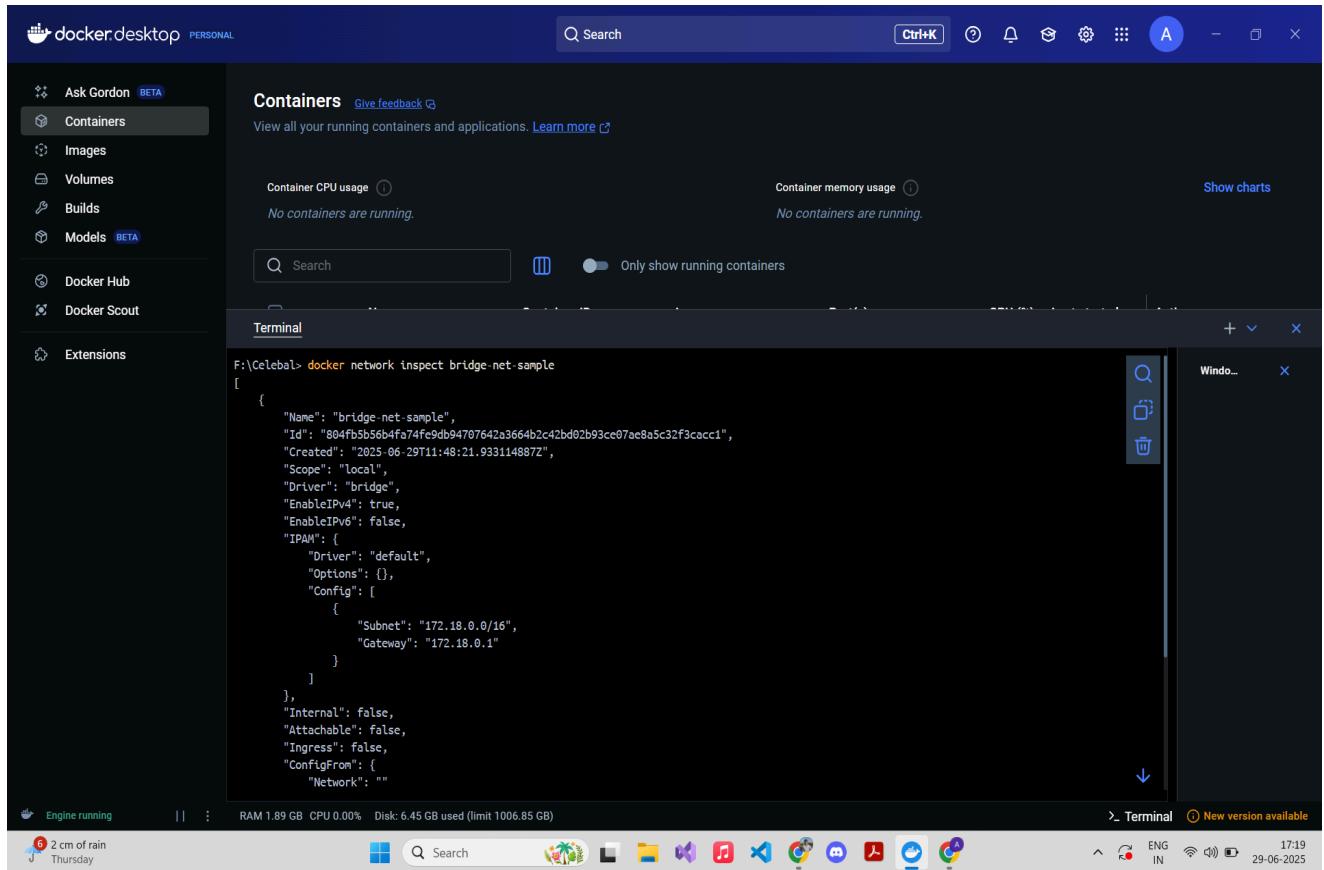
Steps :

1. Create and Use a Custom Docker Bridge Network.

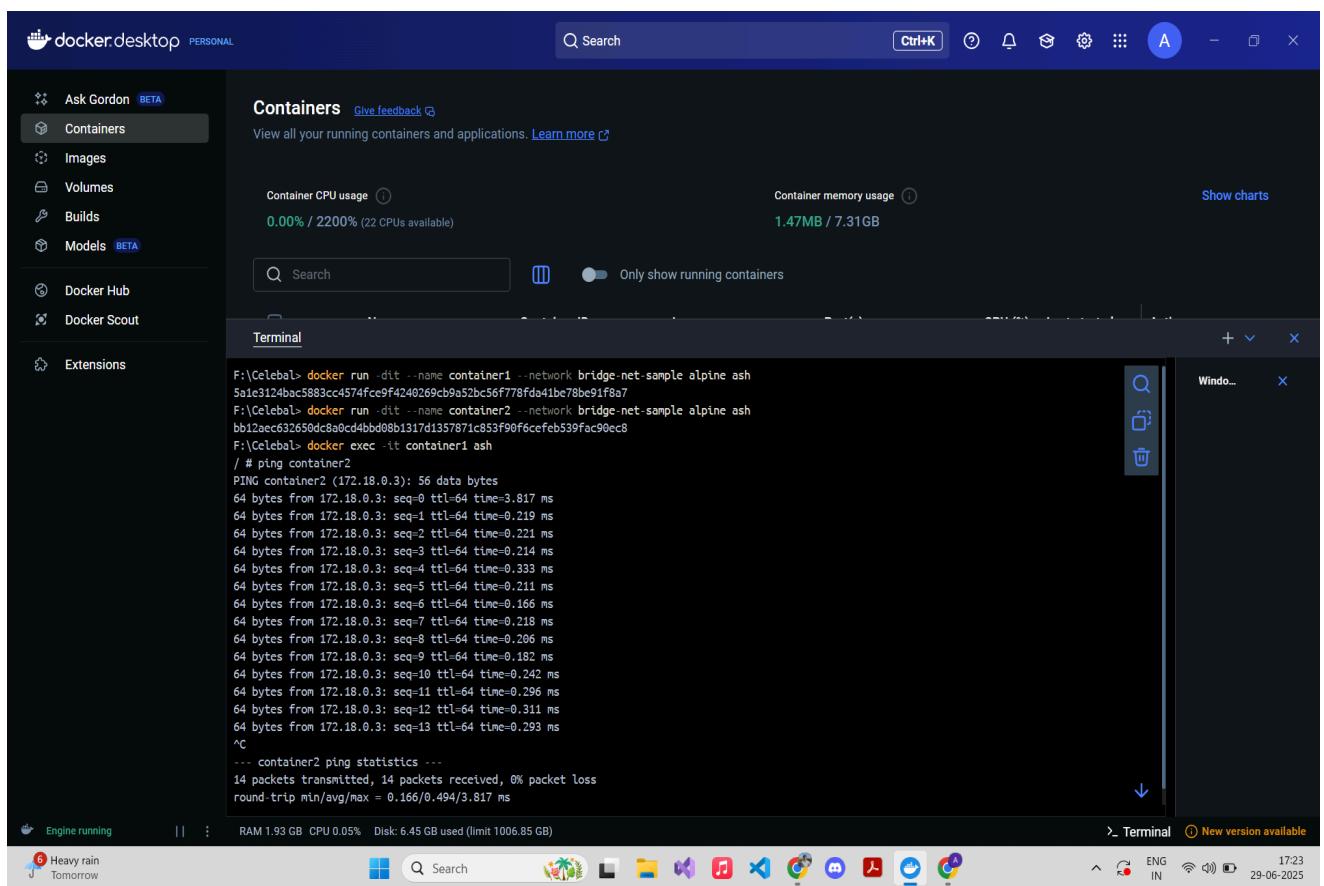
- Create a Custom Bridge Network.
- Verify the Network Exists.

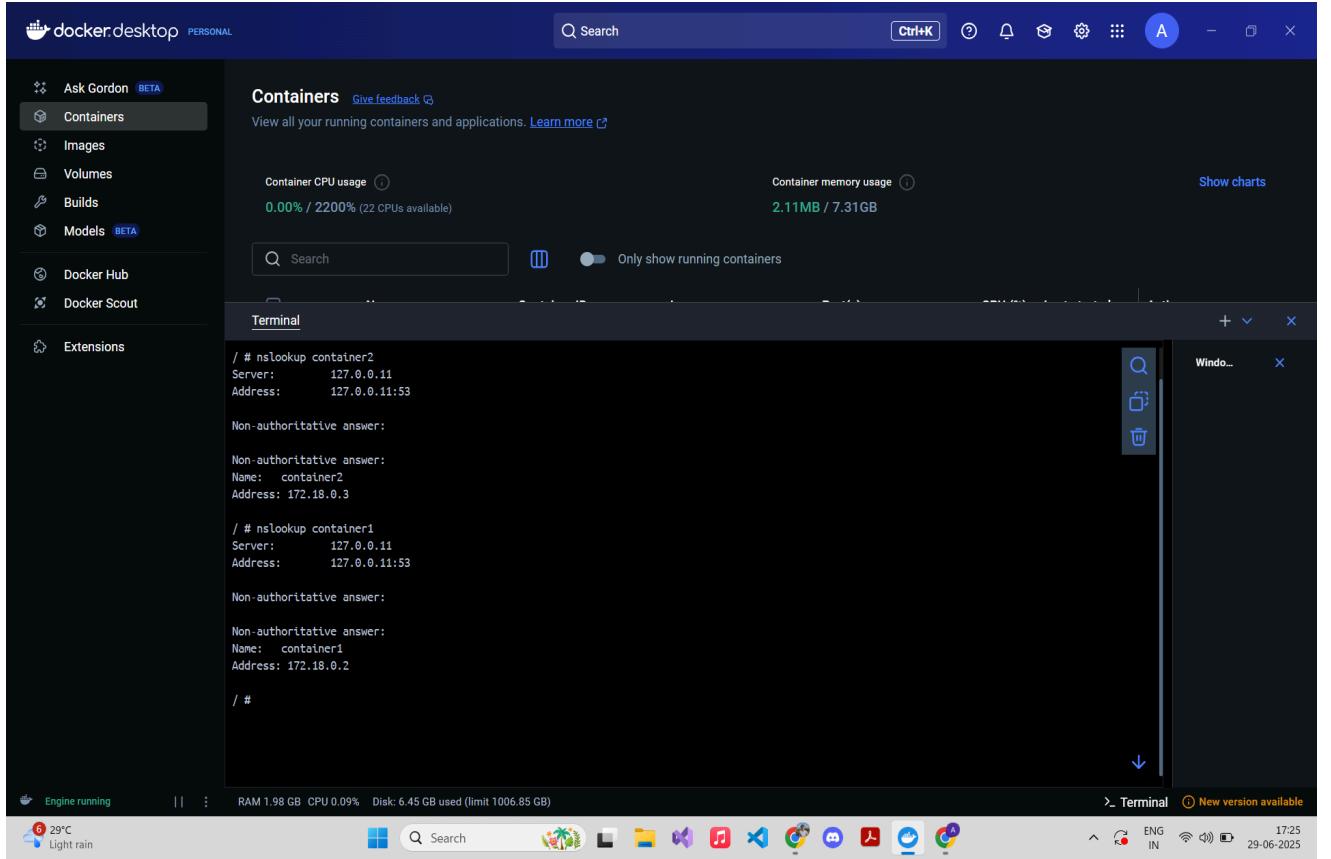


- Inspect the Network



- Run Two Containers in the Same Network.
- Test Inter-Container Communication.



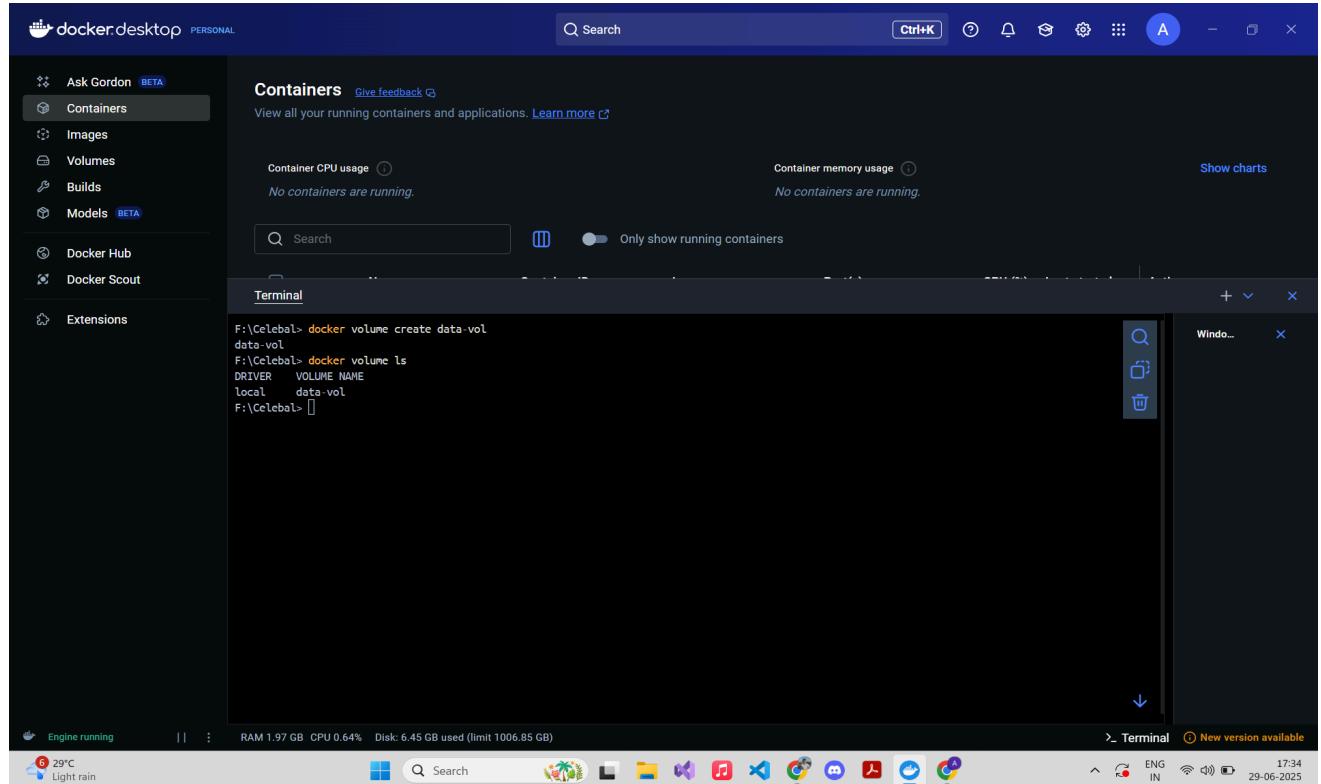


7. Create a Docker volume and mount it to a container.

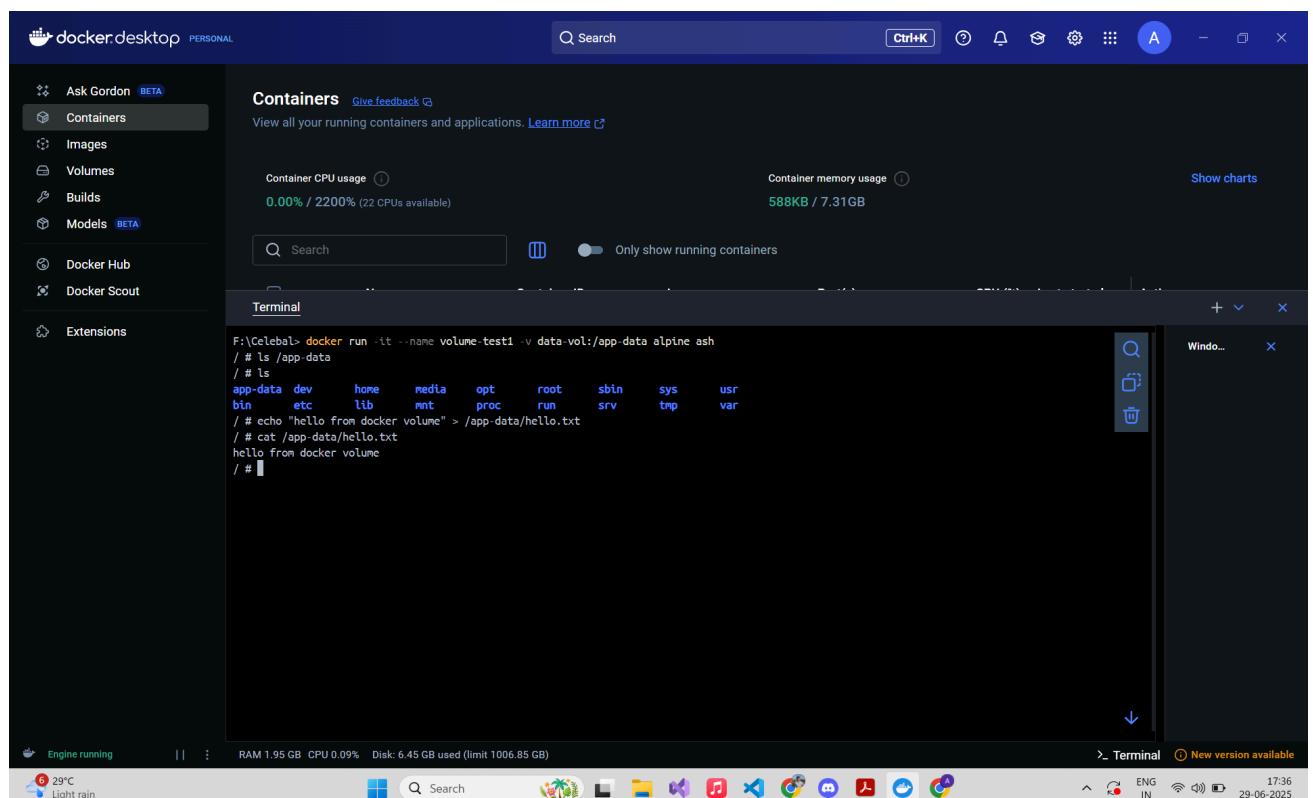
Steps :

1. Create and Use a Docker Volume.

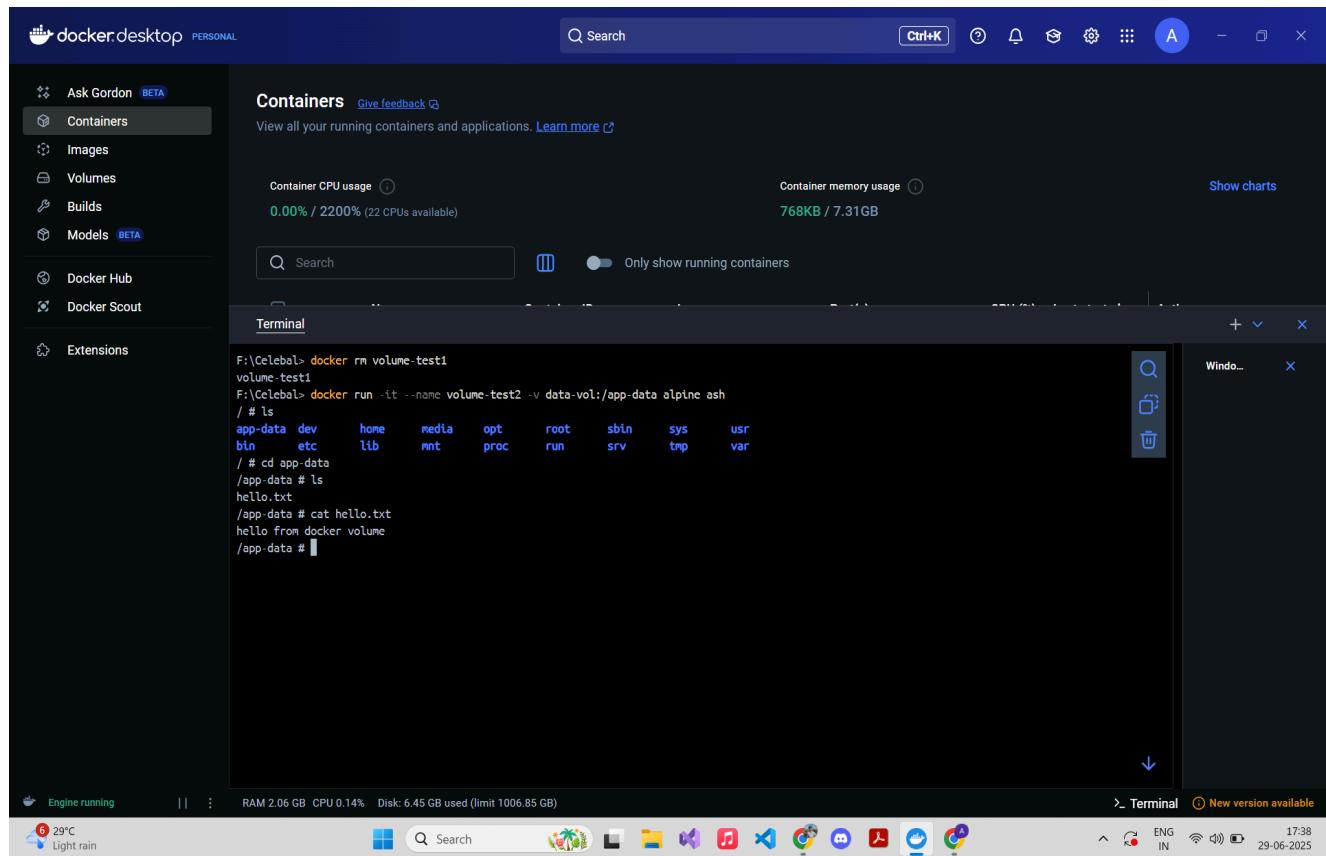
- Create a Docker Volume.
- Verify Volume Creation.



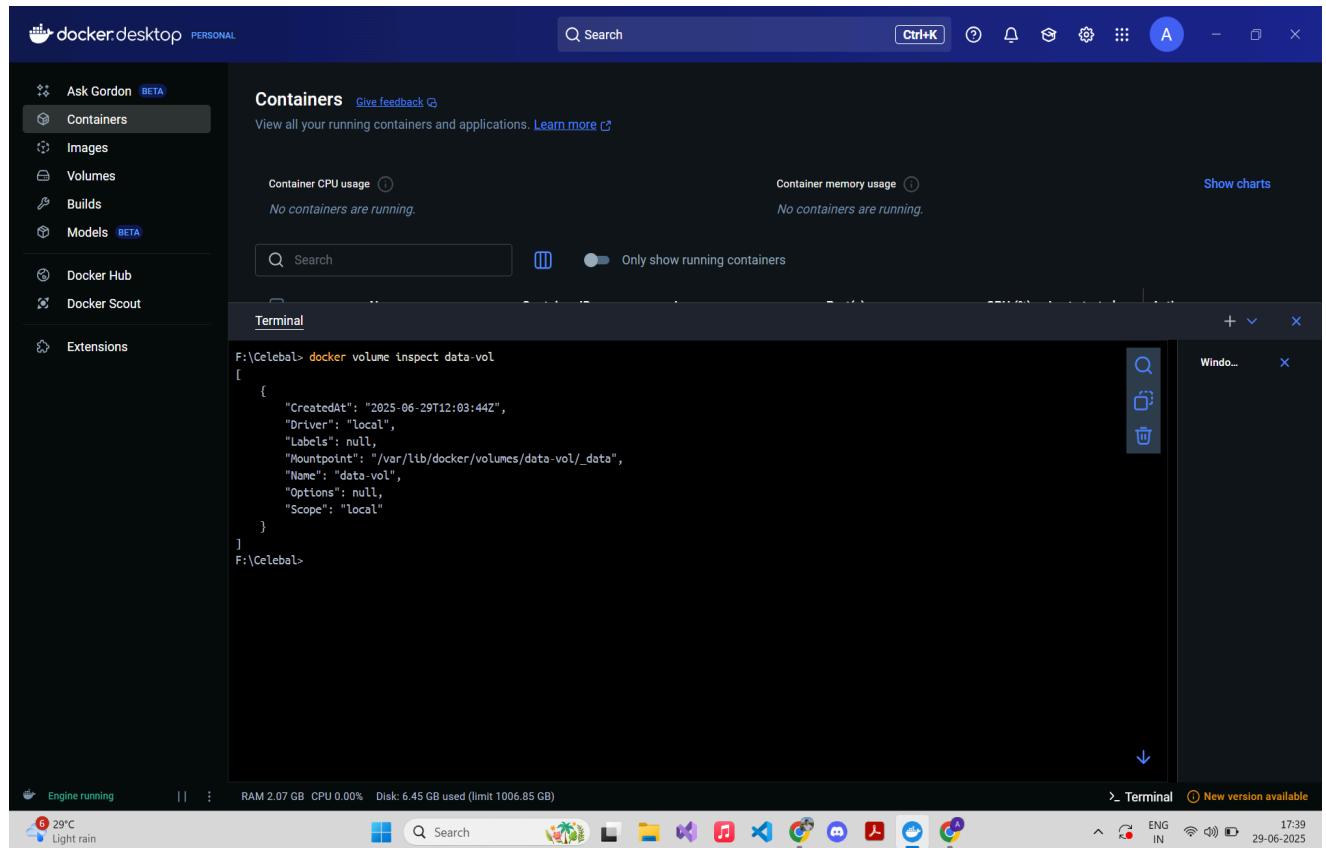
- Run a Container and Mount the Volume.
- Write Some Data into the Volume.



- Remove the Container.
- Run a New Container with the Same Volume.



- Inspect Volume Path on Host.

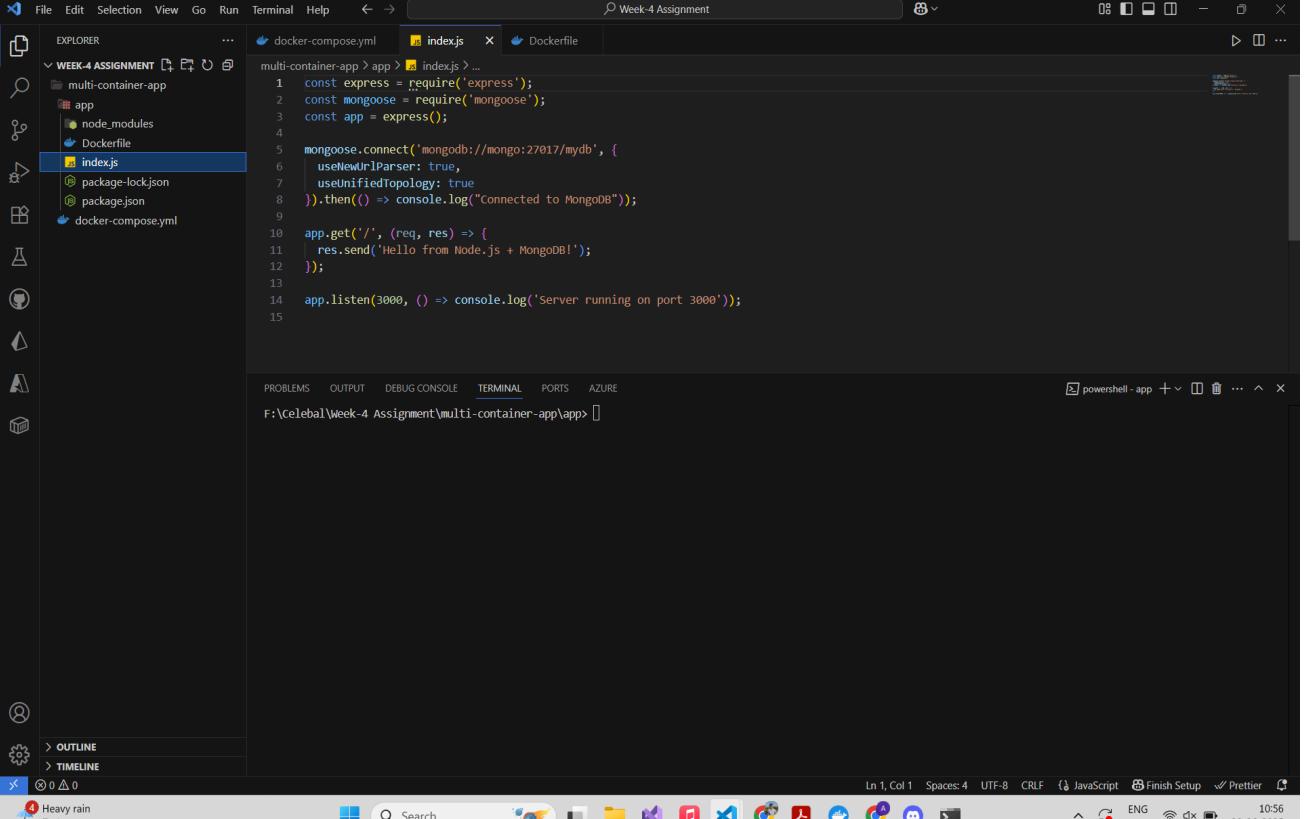


8. Docker Compose for multi-container applications, Docker security best practices

Steps :

Part 1: Docker Compose – Multi-Container Applications

- Create a Simple [Node.js](#) + MongoDB App.



The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows a project structure under "WEEK-4 ASSIGNMENT" named "multi-container-app". It includes files like "app", "node_modules", "Dockerfile", and "index.js".
- Editor:** The "index.js" file is open, displaying the following code:

```
const express = require('express');
const mongoose = require('mongoose');
const app = express();

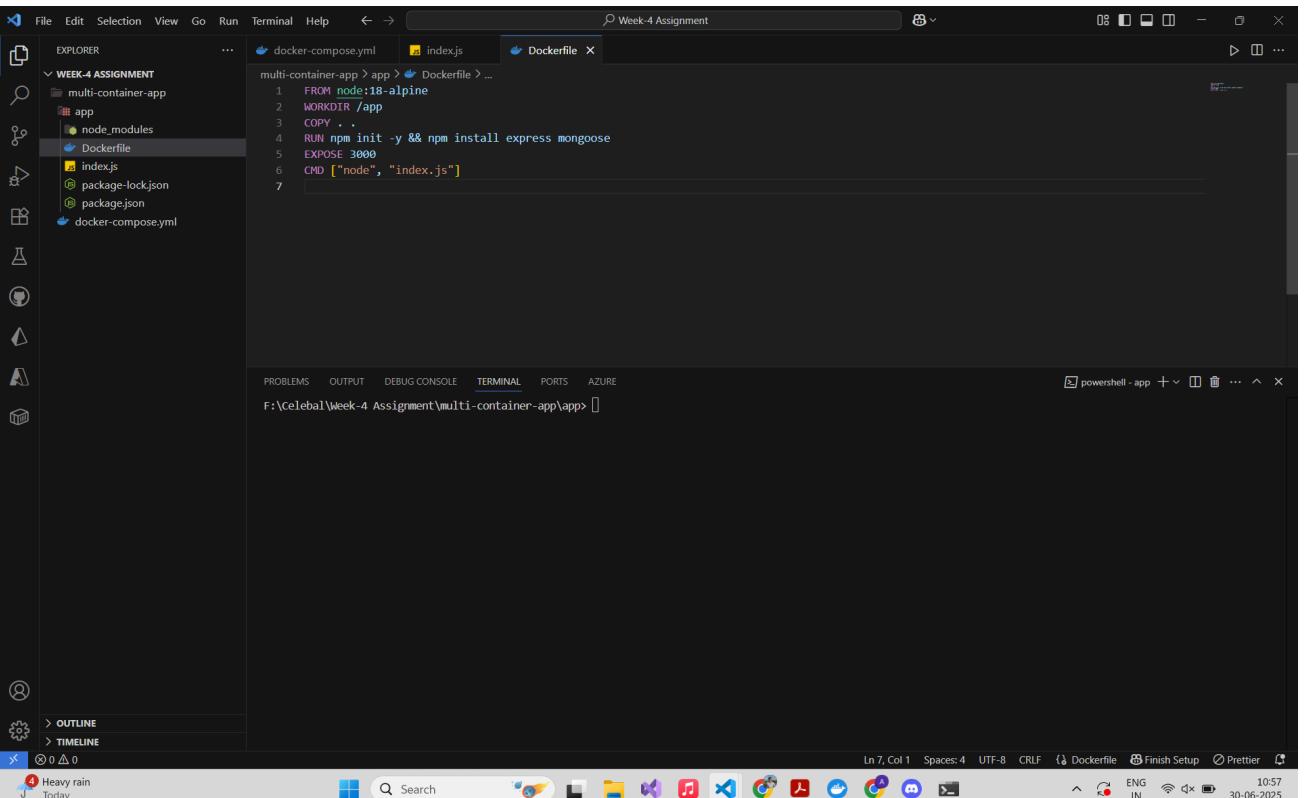
mongoose.connect('mongodb://mongo:27017/mydb', {
  useNewUrlParser: true,
  useUnifiedTopology: true
}).then(() => console.log("Connected to MongoDB"));

app.get('/', (req, res) => {
  res.send('Hello from Node.js + MongoDB!');
});

app.listen(3000, () => console.log('Server running on port 3000'));
```

- Terminal:** The terminal tab shows the path "F:\Celebal\week-4 Assignment\multi-container-app\app>".
- Bottom Status Bar:** Shows system information like "Heavy rain Today", battery level "ENG IN", and date "30-06-2025".

- Create Dockerfile for [Node.js](#)



The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows the same project structure as the previous screenshot.
- Editor:** The "Dockerfile" file is open, displaying the following code:

```
FROM node:18-alpine
WORKDIR /app
COPY .
RUN npm init -y && npm install express mongoose
EXPOSE 3000
CMD ["node", "index.js"]
```

- Terminal:** The terminal tab shows the path "F:\Celebal\week-4 Assignment\multi-container-app\app>".
- Bottom Status Bar:** Shows system information like "Heavy rain Today", battery level "ENG IN", and date "30-06-2025".

- Create docker-compose.yml

```

version: '3.8'
services:
  web:
    build: ./app
    ports:
      - "3000:3000"
    depends_on:
      - mongo
  mongo:
    image: mongo:5
    ports:
      - "27017:27017"
    volumes:
      - mongo-data:/data/db
volumes:
  mongo-data:

```

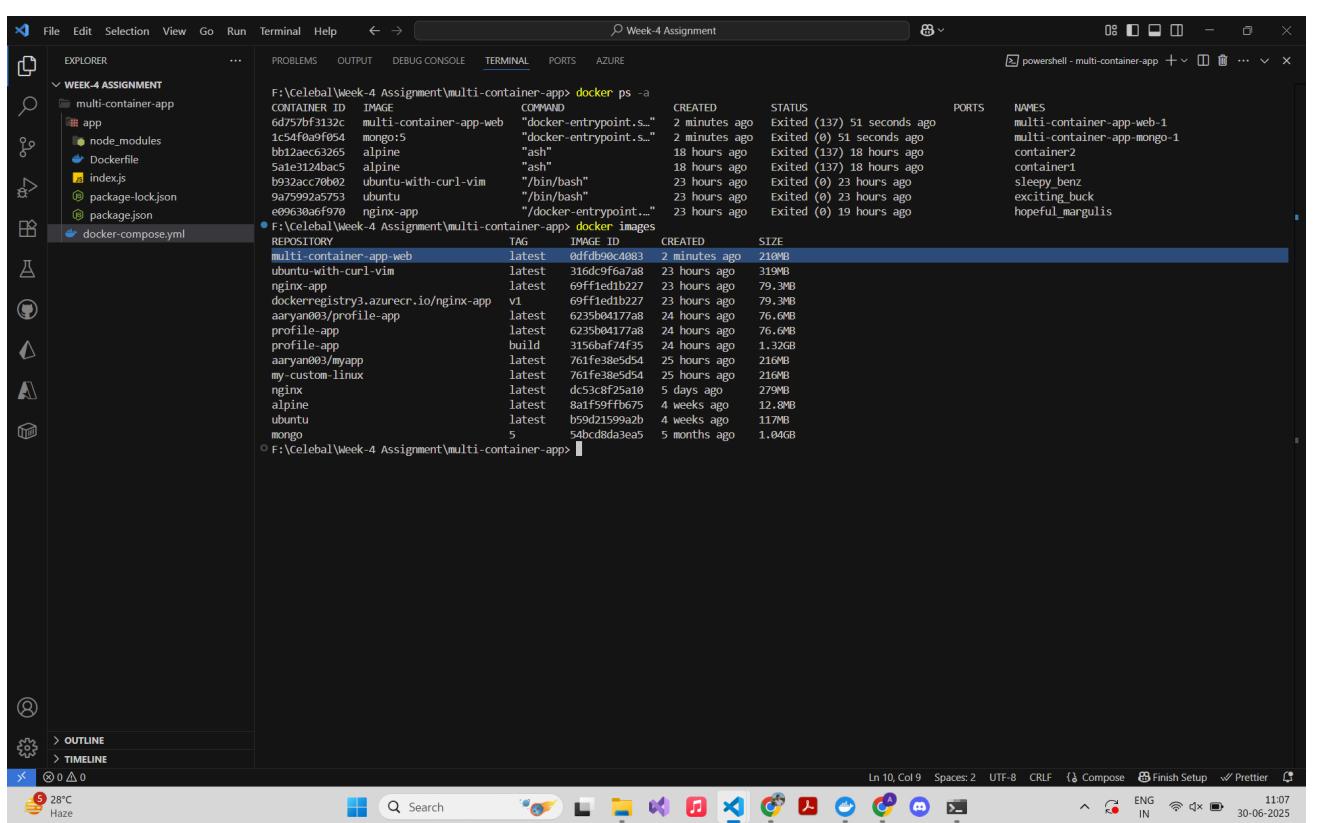
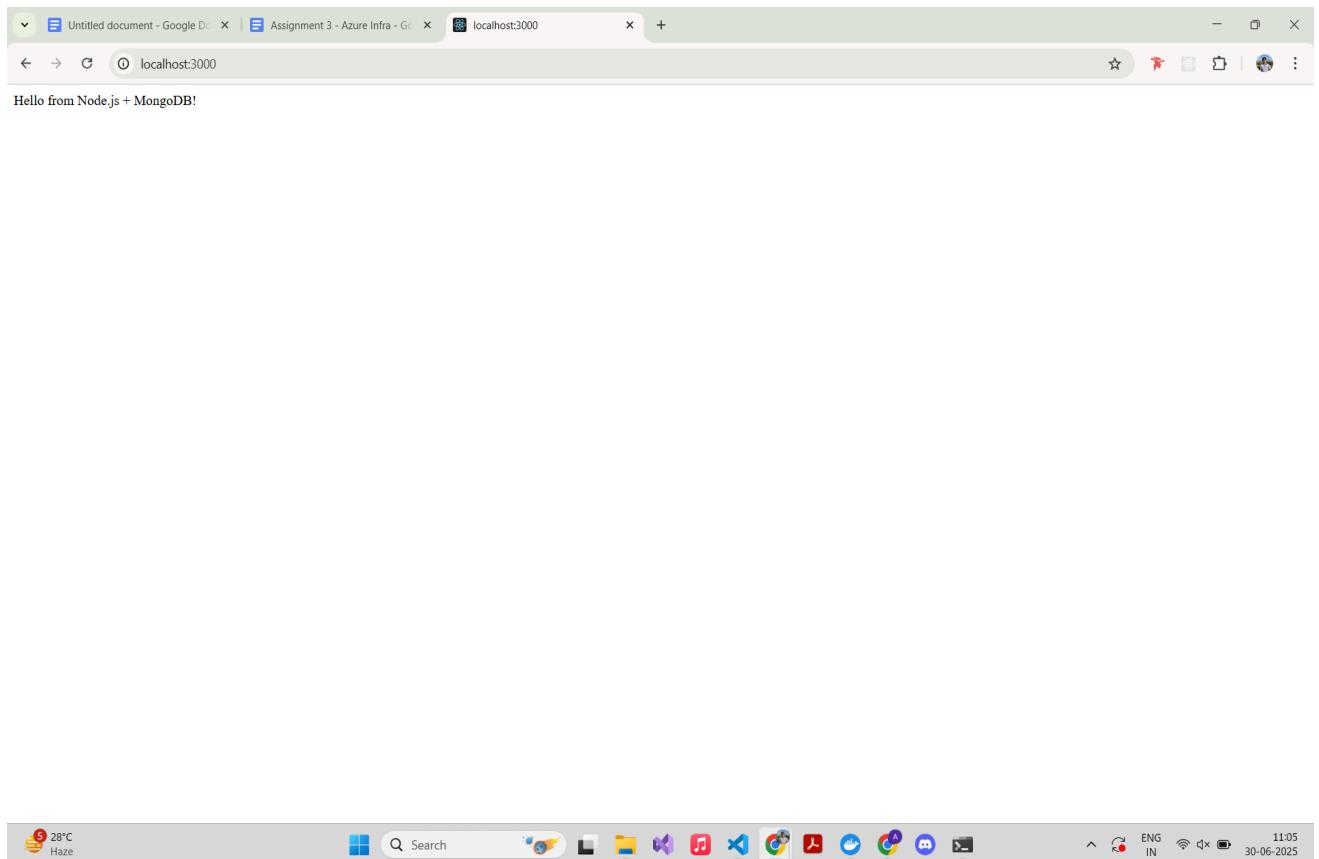
- Run the Compose Application

```

F:\celebal\Week-4 Assignment\multi-container-app\app> cd ..
F:\celebal\Week-4 Assignment\multi-container-app> docker-compose up --build
time=2025-06-30T11:01:26+05:30" level=warning msg=":\\"/celebal\\Week-4 Assignment\\multi-container-app\\docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 9/9
  ✓ mongo Pulled
  ✓ aeddebb0bf1c Pull complete
  ✓ d9802f93d67 Pull complete
  ✓ 3d7e6a7004dc Pull complete
  ✓ 16e9e7e335103 Pull complete
  ✓ b48e1c442bc1 Pull complete
  ✓ a7a1a92a9eac5 Pull complete
  ✓ b076be86645a Pull complete
  ✓ 0456147d1430 Pull complete
Compose can now delegate builds to bake for better performance.
To do so, set COMPOSE_BAKE=true.
[+] Building 47.3s (11/11) FINISHED
  => [web internal] load build definition from Dockerfile
  => => transferring dockerfile: 172B
  => [web internal] load metadata for docker.io/library/node:18-alpine
  => [web auth] library/node:pull token for registry-1.docker.io
  => [web internal] load .dockerrigone
  => => transferring context: 2B
  => [web 1/4] FROM docker.io/library/node:18-alpine@sha256:8d6421d663b4c28fd3ebc498332f249011d118945588d0a35cb9bc4b8ca09d9e
  => => resolve docker.io/library/node:18-alpine@sha256:8d6421d663b4c28fd3ebc498332f249011d118945588d0a35cb9bc4b8ca09d9e
  => sha256:25ff2da83641908f65c3a74d80409d6b1b62ccfaab2209e7a70bd0f5a2e0549 446B / 446B
  => sha256:1e5a4c89ceec5c0826c540a006d4b6491c96eda01837f43bd47f0d26702d6e3 1.26MB / 1.26MB
  => sha256:dd71dde834b5c203d1629026eb994cb2399ae049a0eabcf4fe161b25a3d0e 40.01MB / 40.01MB
  => => extracting sha256:dd71dde834b5c203d1629026eb994cb2399ae049a0eabcf4fe161b25a3d0e
  => => extracting sha256:1e5a4c89ceec5c0826c540a006d4b6491c96eda01837f43bd47f0d26702d6e3
  => => extracting sha256:25ff2da83641908f65c3a74d80409d6b1b62ccfaab2209e7a70bd0f5a2e0549
  => [web internal] load build context
  => => transferring context: 10.1MB
  => [web 2/4] WORKDIR /app
  => [web 3/4] COPY .
  => [web 4/4] RUN npm init -y && npm install express mongoose
  => [web] exporting to image
  => => exporting layers
  => => exporting manifest sha256:e5d1420e83f0cea79c8cb03b7e11e60145321e1a80576507a947c6945ac2d206
  => => exporting config sha256:e413b5756fc62d35a8c55a1a6cb18b5d21cd475d3780c91e49ac1a7a5acf
  => => exporting attestation manifest sha256:fde0bfadabfa0a7d147d560a426a72b215728b593101fff8f501c08e6c9
  => => exporting manifest list sha256:0dfdfb90c40839e05a62a2af9dfdf8b333175ac127064a70d4a52d2272899cac
  => => naming to docker.io/library/multi-container-app-web:latest
  => => unpacking to docker.io/library/multi-container-app-web:latest
  => [web] resolving provenance for metadata file
[+] Running 5/5

```

- Verify the output in the browser.



2. Docker Security Best Practices.

- Use Minimal Base Images
- Scan Images for Vulnerabilities