EXPERIMENT NO. 07

Aim:

To install Docker and practice essential Docker commands for managing images and interacting with containers.

Theory:

Understanding Docker

Docker is a platform that allows developers to package applications along with all required libraries and configurations into an isolated unit called a *container*.

- Without Docker, applications often face the *“works on my machine”* problem, because environments differ across systems.
- With Docker, the application runs identically on any machine — whether it’s your laptop, a server, or in the cloud.

Core Components of Docker Architecture:

1. **Docker Engine (Daemon)**
   - The background process responsible for creating, running, and managing containers and images.
2. **Docker CLI (Command-Line Interface)**
   - A tool through which users send instructions (docker run, docker build, etc.).
   - Example: typing docker run nginx tells the daemon to fetch the Nginx image and start a container.
3. **Container Runtime (containerd)**
   - Responsible for pulling images, storing them, running containers, and controlling their lifecycle (start, stop, delete).
4. **runc**
   - The lowest-level runtime that interacts with the Linux kernel.
   - Provides features like:
     - Namespaces → isolation of processes.
     - cgroups → resource management (CPU, RAM, I/O).

5. **Images & Registries**
    - Image: A read-only template containing application code + environment setup.
    - Registry: A storage hub for images (public → Docker Hub, or private → company repositories).

Containers vs Virtual Machines

- Containers: share the host OS kernel, making them lightweight and fast.
- VMs: require a separate guest OS on top of a hypervisor, making them heavier and slower.
- Startup: Containers launch in seconds; VMs take minutes.
- Efficiency: Containers consume fewer resources due to shared kernel.

Images, Layers & Filesystem

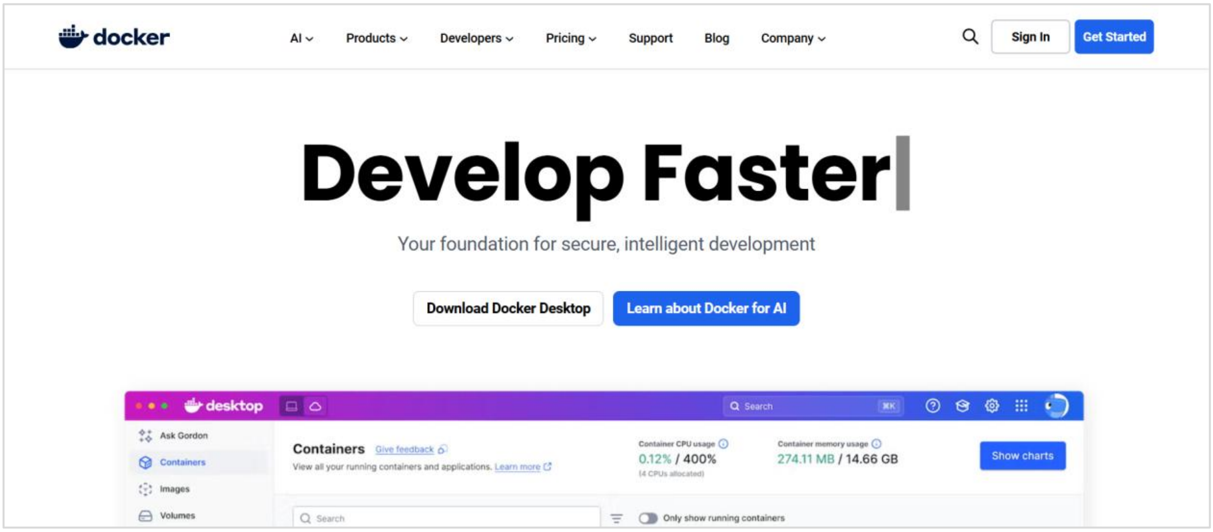
- Layered Images: Every Dockerfile instruction creates a separate cached layer.
- Writable Layer: Containers get a temporary writable layer for runtime changes.
- Union Filesystem: Merges image layers + writable layer into one consistent view.

Dockerfile & Image Building Process

- A Dockerfile contains build instructions (e.g., FROM, RUN, COPY, CMD).
- Each line creates a layer; Docker caches these to speed up future builds.
- Best practices:
    - Use slim base images.
    - Avoid adding unnecessary files.
    - Use .dockerignore for excluding files.
    - Apply multi-stage builds to reduce final image size.

Steps to be Performed:

- ***Install Docker → Download and install from the official Docker site.***


docker
AI
Products
Developers
Pricing
Support
Blog
Company
Sign In
Get Started
Develop Faster
Your foundation for secure, intelligent development
Download Docker Desktop
Learn about Docker for AI
desktop
Ask Gordon
Containers
Images
Volumes
Containers Give feedback
View all your running containers and applications. Learn more
Container CPU usage 0.12% / 400%
Container memory usage 274.11 MB / 14.66 GB
Show charts
Only show running containers

- ***Run Commands in Terminal:***

1. Check Docker version → docker -v

```
C:\Users\humay>docker -v
Docker version 28.3.3, build 980b856

C:\Users\humay>
```

2. Get system details → docker info

```
C:\Users\humay>docker info
Client:
 Version:    28.3.3
 Context:    desktop-linux
 Debug Mode: false
 Plugins:
  ai: Docker AI Agent - Ask Gordon (Docker Inc.)
    Version:  v1.9.11
    Path:     C:\Program Files\Docker\cli-plugins\docker-ai.exe
  buildx: Docker Buildx (Docker Inc.)
    Version:  v0.27.0-desktop.1
    Path:     C:\Program Files\Docker\cli-plugins\docker-buildx.exe
  cloud: Docker Cloud (Docker Inc.)
    Version:  v0.4.21
    Path:     C:\Program Files\Docker\cli-plugins\docker-cloud.exe
  compose: Docker Compose (Docker Inc.)
    Version:  v2.39.2-desktop.1
    Path:     C:\Program Files\Docker\cli-plugins\docker-compose.exe
  debug: Get a shell into any image or container (Docker Inc.)
    Version:  0.0.42
    Path:     C:\Program Files\Docker\cli-plugins\docker-debug.exe
  desktop: Docker Desktop commands (Docker Inc.)
    Version:  v0.2.0
    Path:     C:\Program Files\Docker\cli-plugins\docker-desktop.exe
  extension: Manages Docker extensions (Docker Inc.)
```

3. List available images → docker images

```
C:\Users\humay>docker images
REPOSITORY   TAG         IMAGE ID       CREATED       SIZE
postgres     latest      1d288494853e   2 days ago    643MB
postgres     15-alpine   987b24217300   6 weeks ago   391MB

C:\Users\humay>
```

4. Search images → docker search <image_name>

```
C:\Users\humay>docker search postgres
NAME                       DESCRIPTION                                     STARS     OFFICIAL
postgres                   The PostgreSQL object-relational database sy…   14558     [OK]
circleci/postgres          The PostgreSQL object-relational database sy…   34
cimg/postgres                                                              3
elestio/postgres           Postgres, verified and packaged by Elestio      1
kasmweb/postgres           Postgres image maintained by Kasm Technologi…   5
ubuntu/postgres            PostgreSQL is an open source object-relation…   41
mcp/postgres               Connect with read-only access to PostgreSQL …   27
chainguard/postgres        Build, ship and run secure software with Cha…   1
artifacthub/postgres                                                       0
corpusops/postgres         https://github.com/corpusops/docker-images/     0
geokrety/postgres          Postgres with postgis + quantile and amqp ex…   0
rootpublic/postgres                                                        0
dockette/postgres          My PostgreSQL image with tunning and preinst…   1
cleanstart/postgres        Secure by Design, Built for Speed, Hardened …   0
vulhub/postgres                                                            0
wayofdev/postgres                                                          0
pgrouting/postgres          Postgres Docker images with PostGIS and dep…   0
uselagoon/postgres                                                         0
```

5. Pull image → docker pull <image_name>

```
C:\Users\humay>docker pull postgres
Using default tag: latest
latest: Pulling from library/postgres
8d0a13cb166d: Pull complete
6458b7f41c65: Pull complete
5773151508cd: Pull complete
8551209c5a1e: Pull complete
30b4b10fcf1d: Pull complete
0a51ed68fa52: Pull complete
e1b18b5359f0: Pull complete
ef07360e404d: Pull complete
8a9c24e23f88: Pull complete
28b206cbbc14: Pull complete
168b3ade331e: Pull complete
1d5017cf452d: Pull complete
ce1261c6d567: Pull complete
180db792316f: Pull complete
Digest: sha256:1d288494853e244e7a78d87b3526e650e5221c622f9768ecac9313d0874a9c39
Status: Downloaded newer image for postgres:latest
docker.io/library/postgres:latest
```

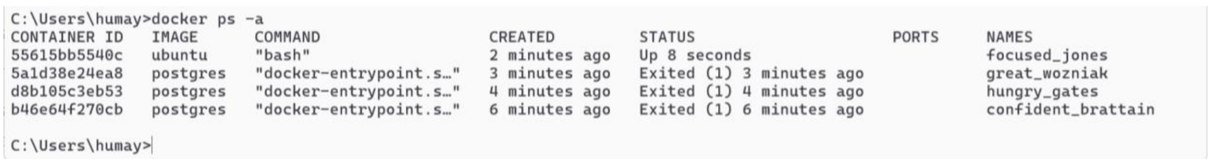
6. Run Ubuntu with shell → docker run -it ubuntu bash

```
C:\Users\humay>docker run -it ubuntu bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
953cdd413371: Pull complete
Digest: sha256:353675e2a41babd526e2b837d7ec780c2a05bca0164f7ea5dbbd433d21d166fc
Status: Downloaded newer image for ubuntu:latest
root@55615bb5540c:/# ls
bin   dev  home  lib64  mnt  proc  run   srv  tmp  var
boot  etc  lib   media  opt  root  sbin  sys  usr
root@55615bb5540c:/#
```

7. View active containers → docker ps

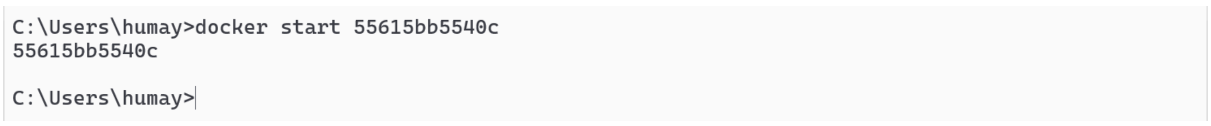
```
C:\Users\humay>docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES

C:\Users\humay>
```

8. View all containers → docker ps -a



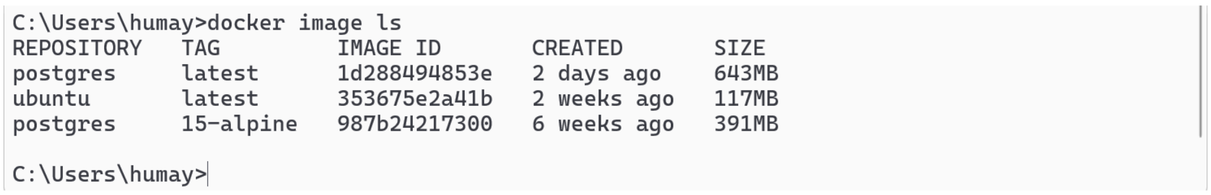
```
C:\Users\humay>docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED         STATUS                     PORTS     NAMES
55615bb5540c   ubuntu     "bash"                   2 minutes ago   Up 8 seconds                         focused_jones
5a1d38e24ea8   postgres   "docker-entrypoint.s…"   3 minutes ago   Exited (1) 3 minutes ago             great_wozniak
d8b105c3eb53   postgres   "docker-entrypoint.s…"   4 minutes ago   Exited (1) 4 minutes ago             hungry_gates
b46e64f270cb   postgres   "docker-entrypoint.s…"   6 minutes ago   Exited (1) 6 minutes ago             confident_brattain

C:\Users\humay>
```

9. Start stopped container → docker start <container_id>



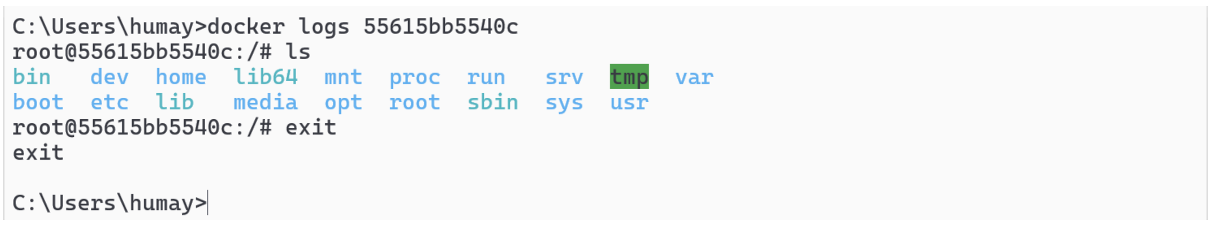
```
C:\Users\humay>docker start 55615bb5540c
55615bb5540c

C:\Users\humay>
```

10. List all images → docker image ls



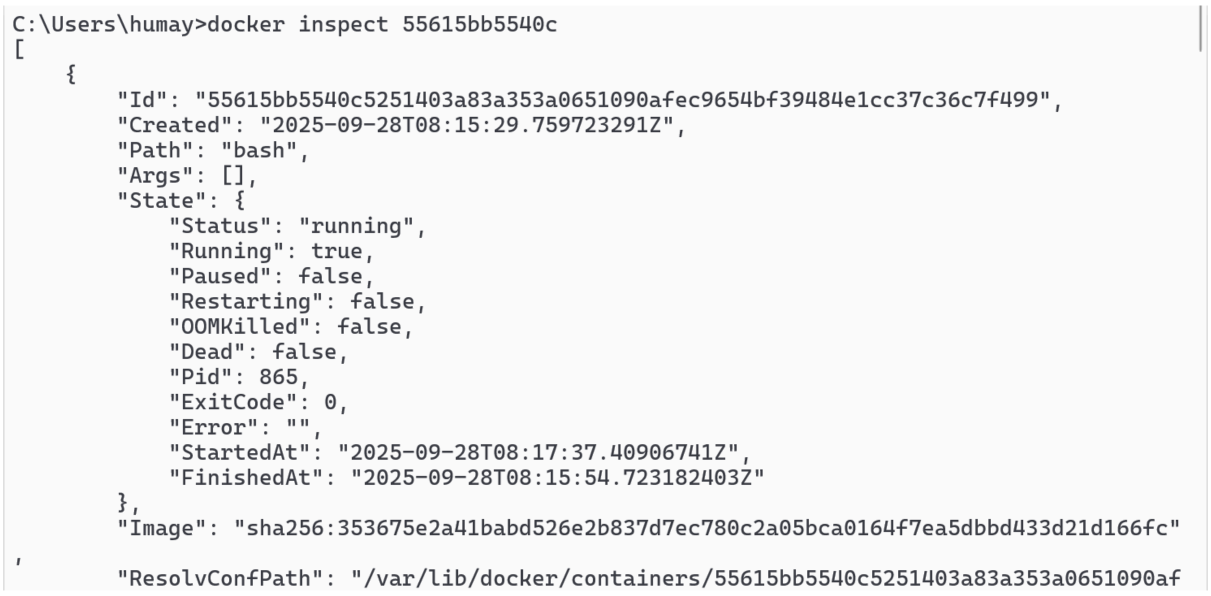
```
C:\Users\humay>docker image ls
REPOSITORY   TAG         IMAGE ID       CREATED        SIZE
postgres     latest      1d288494853e   2 days ago     643MB
ubuntu       latest      353675e2a41b   2 weeks ago    117MB
postgres     15-alpine   987b24217300   6 weeks ago    391MB

C:\Users\humay>
```

11. View logs → docker logs <container_id>



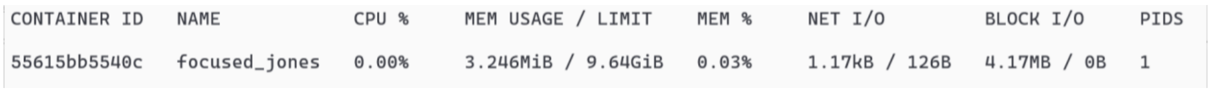
```
C:\Users\humay>docker logs 55615bb5540c
root@55615bb5540c:/# ls
bin   dev  home  lib64  mnt  proc  run   srv  tmp  var
boot  etc  lib   media  opt  root  sbin  sys  usr
root@55615bb5540c:/# exit
exit

C:\Users\humay>
```

12. Inspect details → docker inspect <container_id>



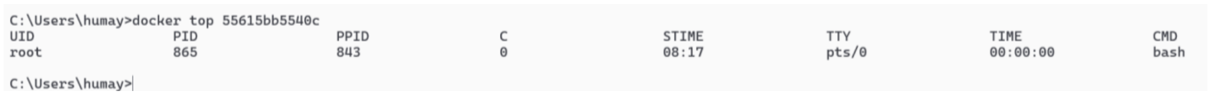
```
C:\Users\humay>docker inspect 55615bb5540c
[
    {
        "Id": "55615bb5540c5251403a83a353a0651090afec9654bf39484e1cc37c36c7f499",
        "Created": "2025-09-28T08:15:29.759723291Z",
        "Path": "bash",
        "Args": [],
        "State": {
            "Status": "running",
            "Running": true,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
            "Dead": false,
            "Pid": 865,
            "ExitCode": 0,
            "Error": "",
            "StartedAt": "2025-09-28T08:17:37.40906741Z",
            "FinishedAt": "2025-09-28T08:15:54.723182403Z"
        },
        "Image": "sha256:353675e2a41babd526e2b837d7ec780c2a05bca0164f7ea5dbbd433d21d166fc"
,
        "ResolvConfPath": "/var/lib/docker/containers/55615bb5540c5251403a83a353a0651090af
```

13. Monitor usage → docker stats



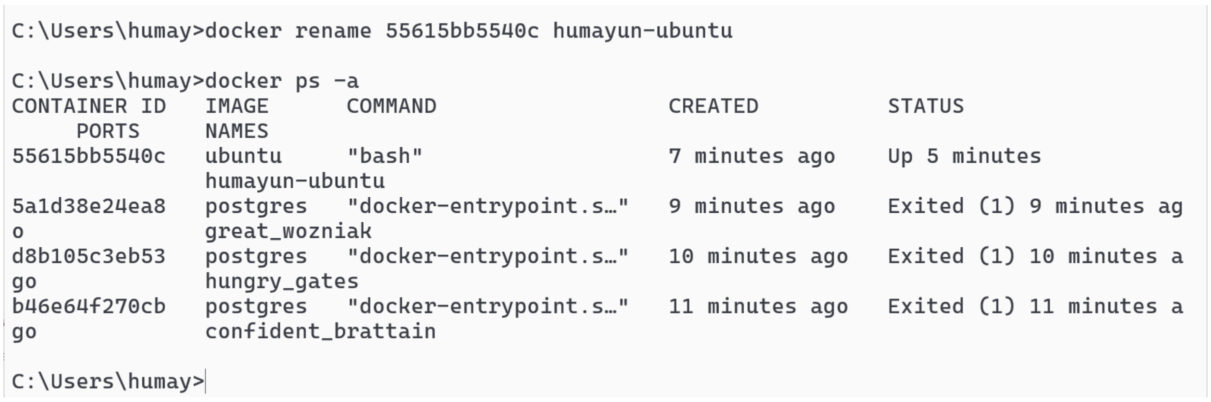
```
CONTAINER ID   NAME           CPU %    MEM USAGE / LIMIT     MEM %    NET I/O          BLOCK I/O     PIDS
55615bb5540c   focused_jones  0.00%    3.246MiB / 9.64GiB    0.03%    1.17kB / 126B    4.17MB / 0B   1
```

14. Show processes inside container → docker top <container_id>



```
C:\Users\humay>docker top 55615bb5540c
UID                 PID                 PPID                C                   STIME               TTY                 TIME                CMD
root                865                 843                 0                   08:17               pts/0               00:00:00            bash

C:\Users\humay>
```

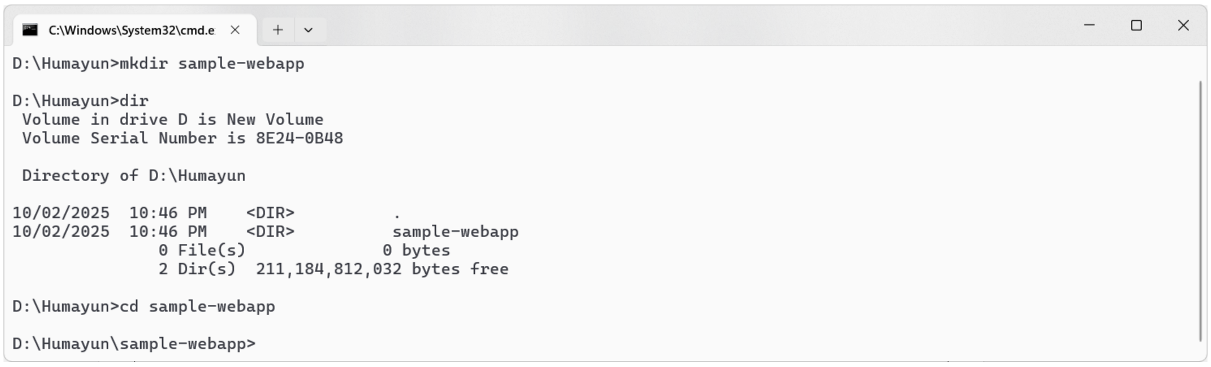
15. Rename container → docker rename <old_name/id> <new_name>



```
C:\Users\humay>docker rename 55615bb5540c humayun-ubuntu

C:\Users\humay>docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED          STATUS
     PORTS     NAMES
55615bb5540c   ubuntu     "bash"                   7 minutes ago    Up 5 minutes
               humayun-ubuntu
5a1d38e24ea8   postgres   "docker-entrypoint.s…"   9 minutes ago    Exited (1) 9 minutes ag
o              great_wozniak
d8b105c3eb53   postgres   "docker-entrypoint.s…"   10 minutes ago   Exited (1) 10 minutes a
go             hungry_gates
b46e64f270cb   postgres   "docker-entrypoint.s…"   11 minutes ago   Exited (1) 11 minutes a
go             confident_brattain

C:\Users\humay>
```

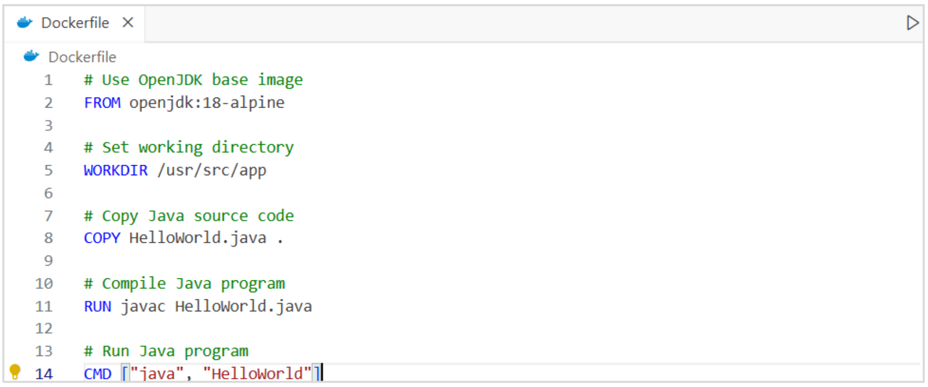
---

1. **Create a project folder:**



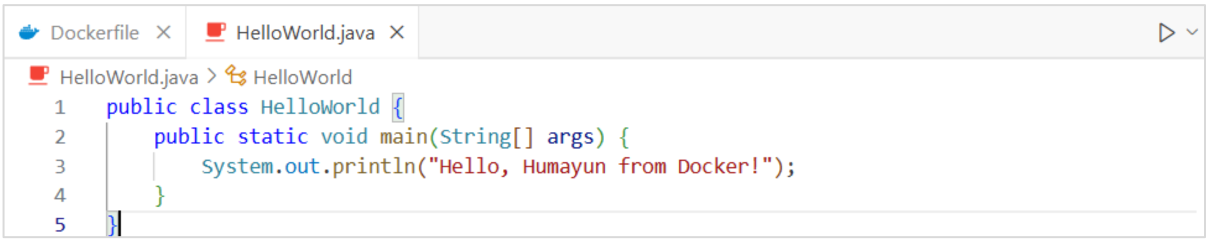
```
D:\Humayun>mkdir sample-webapp

D:\Humayun>dir
 Volume in drive D is New Volume
 Volume Serial Number is 8E24-0B48

 Directory of D:\Humayun

10/02/2025  10:46 PM    <DIR>          .
10/02/2025  10:46 PM    <DIR>          sample-webapp
               0 File(s)              0 bytes
               2 Dir(s)  211,184,812,032 bytes free

D:\Humayun>cd sample-webapp

D:\Humayun\sample-webapp>
```

2. **Create a Docker file:** Inside the project folder, create a file named Dockerfile & Add the following instructions



```
# Use OpenJDK base image
FROM openjdk:18-alpine

# Set working directory
WORKDIR /usr/src/app

# Copy Java source code
COPY HelloWorld.java .

# Compile Java program
RUN javac HelloWorld.java

# Run Java program
CMD ["java", "HelloWorld"]
```

3. Create a Java File:

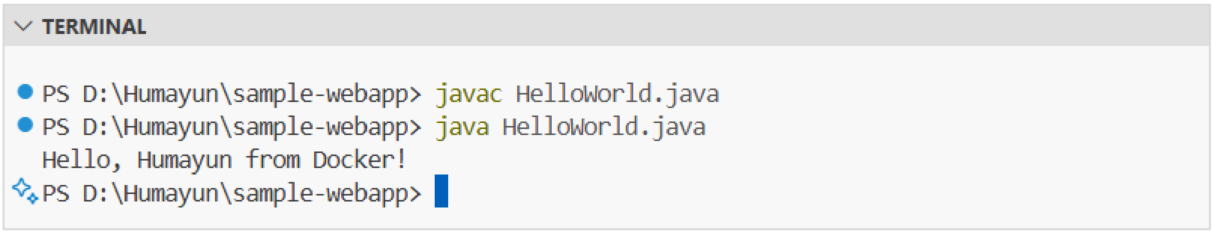


Dockerfile × HelloWorld.java ×

HelloWorld.java > HelloWorld

```java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, Humayun from Docker!");
    }
}
```

4. Compile and run Java project:

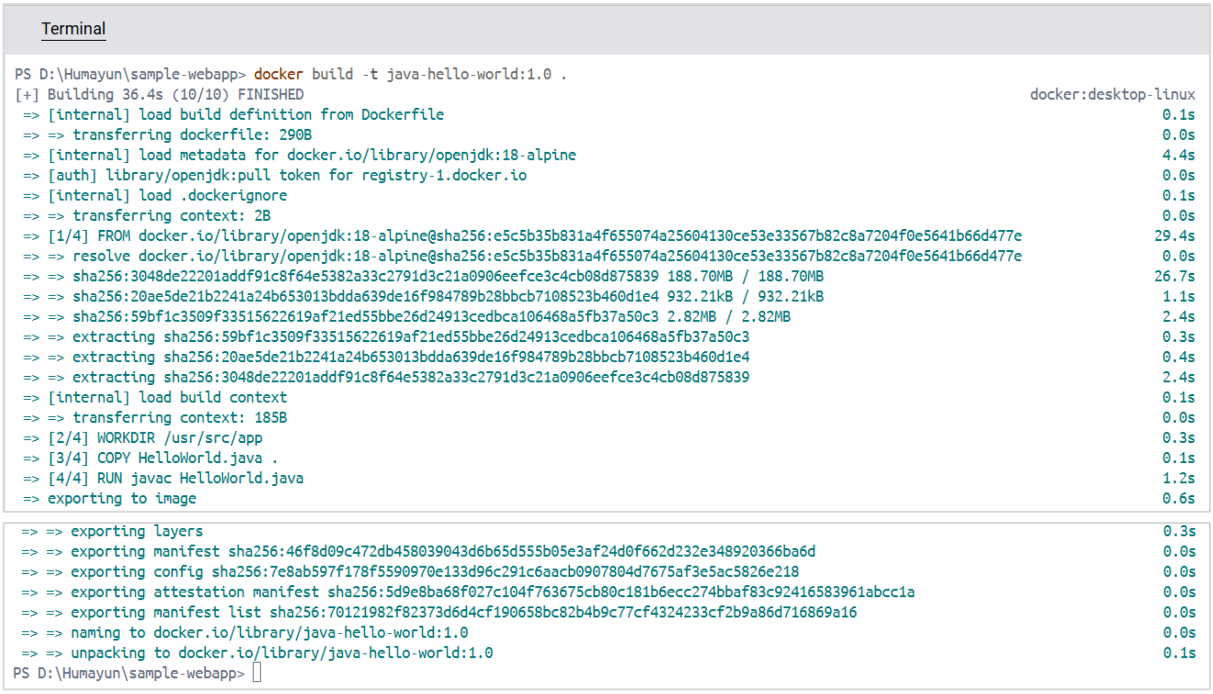


TERMINAL

```
PS D:\Humayun\sample-webapp> javac HelloWorld.java
PS D:\Humayun\sample-webapp> java HelloWorld.java
Hello, Humayun from Docker!
PS D:\Humayun\sample-webapp>
```

5. Build the Docker Image:

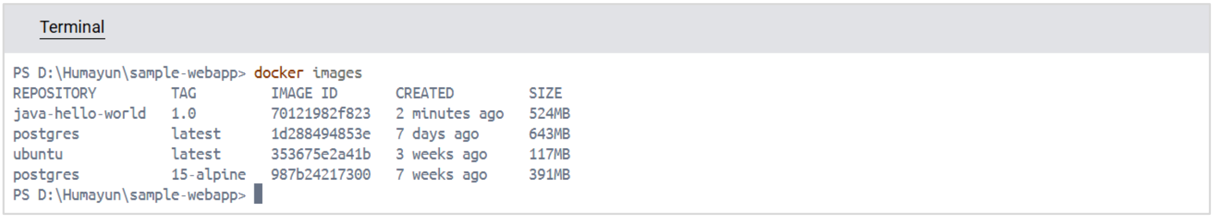
```
docker build -t java-hello-world:1.0 .
```



Terminal

```
PS D:\Humayun\sample-webapp> docker build -t java-hello-world:1.0 .
[+] Building 36.4s (10/10) FINISHED                                                                                docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                               0.1s
 => => transferring dockerfile: 290B                                                                                               0.0s
 => [internal] load metadata for docker.io/library/openjdk:18-alpine                                                               4.4s
 => [auth] library/openjdk:pull token for registry-1.docker.io                                                                     0.0s
 => [internal] load .dockerignore                                                                                                  0.1s
 => => transferring context: 2B                                                                                                    0.0s
 => [1/4] FROM docker.io/library/openjdk:18-alpine@sha256:e5c5b35b831a4f655074a25604130ce53e33567b82c8a7204f0e5641b66d477e         29.4s
 => => resolve docker.io/library/openjdk:18-alpine@sha256:e5c5b35b831a4f655074a25604130ce53e33567b82c8a7204f0e5641b66d477e          0.0s
 => => sha256:3048de22201addf91c8f64e5382a33c2791d3c21a0906eefce3c4cb08d875839 188.70MB / 188.70MB                                 26.7s
 => => sha256:20ae5de21b2241a24b653013bdda639de16f984789b28bbcb7108523b460d1e4 932.21kB / 932.21kB                                  1.1s
 => => sha256:59bf1c3509f33515622619af21ed55bbe26d24913cedbca106468a5fb37a50c3 2.82MB / 2.82MB                                      2.4s
 => => extracting sha256:59bf1c3509f33515622619af21ed55bbe26d24913cedbca106468a5fb37a50c3                                          0.3s
 => => extracting sha256:20ae5de21b2241a24b653013bdda639de16f984789b28bbcb7108523b460d1e4                                          0.4s
 => => extracting sha256:3048de22201addf91c8f64e5382a33c2791d3c21a0906eefce3c4cb08d875839                                          2.4s
 => [internal] load build context                                                                                                  0.1s
 => => transferring context: 185B                                                                                                  0.0s
 => [2/4] WORKDIR /usr/src/app                                                                                                     0.3s
 => [3/4] COPY HelloWorld.java .                                                                                                   0.1s
 => [4/4] RUN javac HelloWorld.java                                                                                                1.2s
 => exporting to image                                                                                                             0.6s
 => => exporting layers                                                                                                            0.3s
 => => exporting manifest sha256:46f8d09c472db458039043d6b65d555b05e3af24d0f662d232e348920366ba6d                                  0.0s
 => => exporting config sha256:7e8ab597f178f5590970e133d96c291c6aacb0907804d7675af3e5ac5826e218                                    0.0s
 => => exporting attestation manifest sha256:5d9e8ba68f027c104f763675cb80c181b6ecc274bbaf83c92416583961abcc1a                      0.0s
 => => exporting manifest list sha256:70121982f82373d6d4cf190658bc82b4b9c77cf4324233cf2b9a86d716869a16                             0.0s
 => => naming to docker.io/library/java-hello-world:1.0                                                                            0.0s
 => => unpacking to docker.io/library/java-hello-world:1.0                                                                         0.1s
PS D:\Humayun\sample-webapp>
```

6. Docker image has been created successfully:

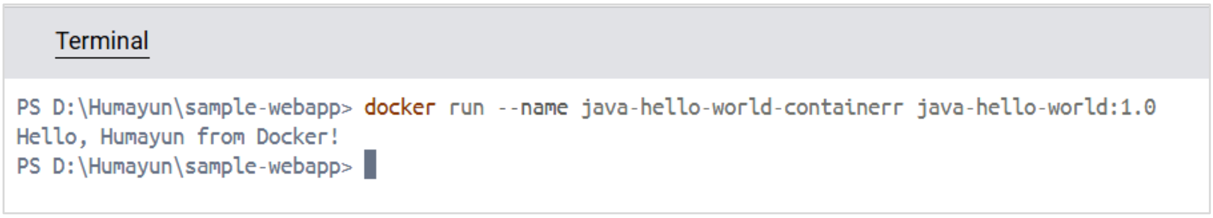


Terminal

```
PS D:\Humayun\sample-webapp> docker images
REPOSITORY         TAG         IMAGE ID       CREATED          SIZE
java-hello-world   1.0         70121982f823   2 minutes ago    524MB
postgres           latest      1d288494853e   7 days ago       643MB
ubuntu             latest      353675e2a41b   3 weeks ago      117MB
postgres           15-alpine   987b24217300   7 weeks ago      391MB
PS D:\Humayun\sample-webapp>
```

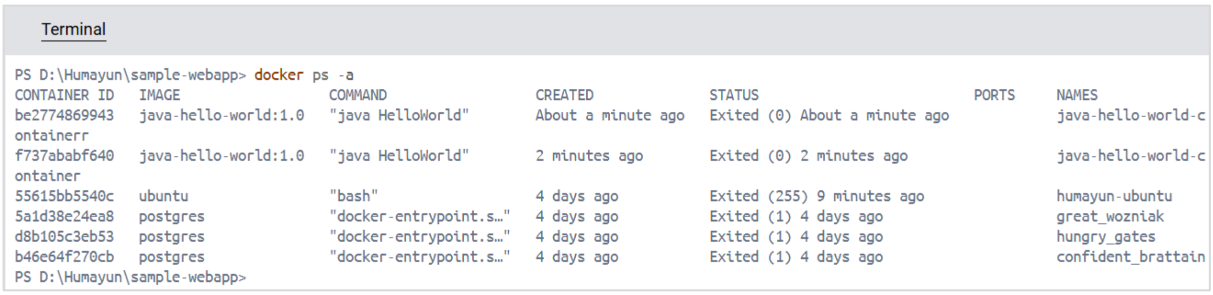
7. Run the docker Container:

```
docker run -d --name java-hello-world-container java-hello-world:1.0
```



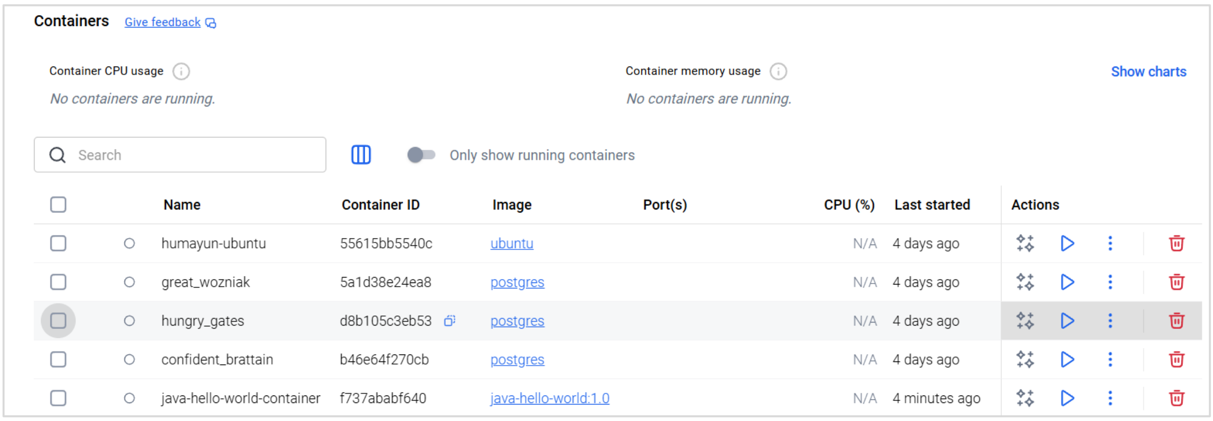
```
PS D:\Humayun\sample-webapp> docker run --name java-hello-world-containerr java-hello-world:1.0
Hello, Humayun from Docker!
PS D:\Humayun\sample-webapp>
```

8. Docker Containers and Images:

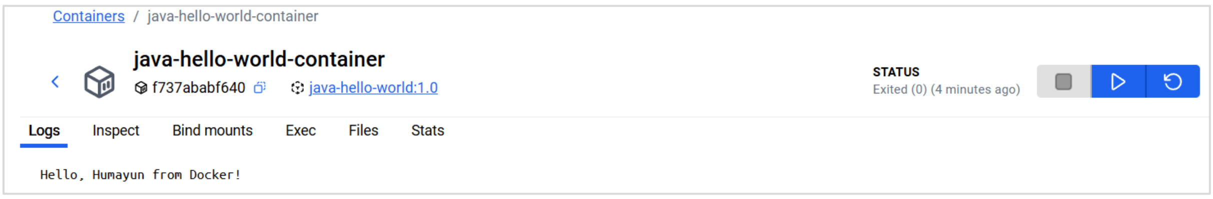


```
PS D:\Humayun\sample-webapp> docker ps -a
CONTAINER ID   IMAGE                  COMMAND                  CREATED              STATUS                          PORTS     NAMES
be2774869943   java-hello-world:1.0   "java HelloWorld"        About a minute ago   Exited (0) About a minute ago             java-hello-world-c
ontainerr
f737ababf640   java-hello-world:1.0   "java HelloWorld"        2 minutes ago        Exited (0) 2 minutes ago                  java-hello-world-c
ontainer
55615bb5540c   ubuntu                 "bash"                   4 days ago           Exited (255) 9 minutes ago                humayun-ubuntu
5a1d38e24ea8   postgres               "docker-entrypoint.s…"   4 days ago           Exited (1) 4 days ago                     great_wozniak
d8b105c3eb53   postgres               "docker-entrypoint.s…"   4 days ago           Exited (1) 4 days ago                     hungry_gates
b46e64f270cb   postgres               "docker-entrypoint.s…"   4 days ago           Exited (1) 4 days ago                     confident_brattain
PS D:\Humayun\sample-webapp>
```

9. Inspect Logs:



Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
humayun-ubuntu	55615bb5540c	ubuntu		N/A	4 days ago	
great_wozniak	5a1d38e24ea8	postgres		N/A	4 days ago	
hungry_gates	d8b105c3eb53	postgres		N/A	4 days ago	
confident_brattain	b46e64f270cb	postgres		N/A	4 days ago	
java-hello-world-container	f737ababf640	java-hello-world:1.0		N/A	4 minutes ago	


Containers / java-hello-world-container
java-hello-world-container
f737ababf640
java-hello-world:1.0
STATUS
Exited (0) (4 minutes ago)
Logs
Inspect
Bind mounts
Exec
Files
Stats

```
Hello, Humayun from Docker!
```

Conclusion:

This experiment introduced us to Docker installation and practical commands for container management. We explored pulling images, running containers, checking logs, monitoring resource usage, and inspecting system information. By working hands-on with Docker, it became clear how containers simplify deployment, ensure consistent environments, and make applications portable across platforms.