Name: Harshi Lodha **Roll NO: 55** **Batch: T13**

Experiment No. 10

AIM: To learn Dockerfile instructions, build an image for a sample web application using DOCKERFILE.

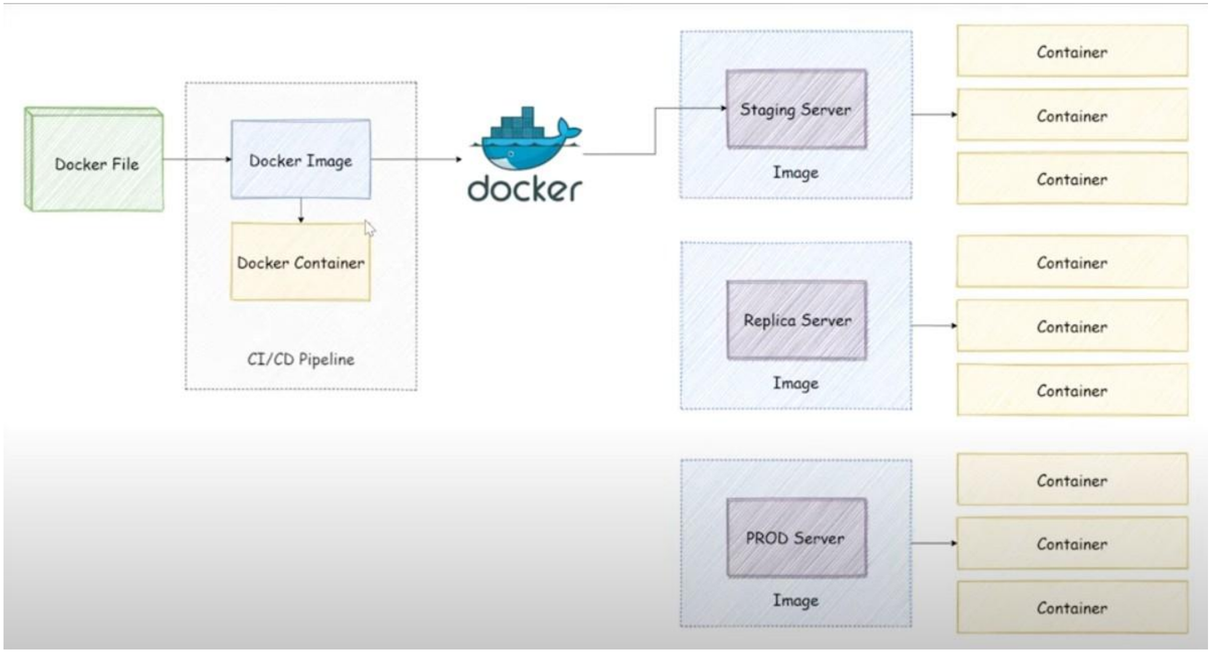
Theory :

Dockerfiles are the cornerstone of creating Docker images. They contain a set of instructions that automate the process of building a Docker image, specifying everything from the base operating system to the application code, dependencies, and configuration settings.

1. What is a Dockerfile?

A Dockerfile is a plain text file that defines the steps required to build a Docker image. It contains a series of commands (or instructions) that specify how the image should be constructed.

- **Purpose**: Automate the creation of Docker images for reproducibility, scalability, and consistency.
- **Format**: Written in a simple scripting language, using instructions like FROM, RUN, COPY, CMD, etc.



2. Basic Structure of a Dockerfile

# Use an official Python runtime as a parent image

FROM python:3.9-slim

```
# Set the working directory inside the container
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

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

# Make port 80 available to the world outside this container
EXPOSE 80

# Define environment variable
ENV NAME World

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

3. Common Dockerfile Instructions

1. FROM (Base Image)

- **Purpose**: Specifies the base image for your Docker image.

Example:

```
 FROM ubuntu:20.04
FROM node:14
FROM python:3.9-slim
```

- **Note**: This is the first instruction and is mandatory in most cases.

2. WORKDIR (Set Working Directory)

- **Purpose**: Defines the directory inside the container where subsequent instructions will be executed.

3. COPY (Copy Files)

- **Purpose**: Copies files or directories from the host system into the container.

Example:

```
COPY . /app
```

-
- **Variants**:
  - COPY <src> <dest>
  - ADD is similar but supports remote URLs and tar file extraction.

4. RUN (Execute Commands)

- **Purpose**: Executes commands inside the container during the image build process.

Example:

```
 RUN apt-get update && apt-get install -y curl
RUN pip install --no-cache-dir -r requirements.txt
```

5. EXPOSE (Expose Ports)

- **Purpose**: Informs Docker that the container will listen on the specified network ports at runtime.

Example: EXPOSE 80

- **Note**: This does not publish the port; it's just for documentation.

6. ENV (Set Environment Variables)

- **Purpose**: Sets environment variables inside the container.

Example:

```
ENV APP_ENV=production
```

7. CMD (Default Command)

- **Purpose**: Specifies the default command to run when the container starts.

Example:

```
CMD ["python", "app.py"]
```

- **Key Points**:
  - Only one CMD is allowed.
  - It can be overridden by passing a command with docker run.

8. ENTRYPOINT (Set Entry Point)

- **Purpose**: Defines a command that will always be executed when the container starts.

Example:

```
ENTRYPOINT ["python"]

CMD ["app.py"]
```

- **Difference from CMD**: ENTRYPOINT is not overridden unless explicitly done with --entrypoint.

4. Building Images from a Dockerfile:

To build an image:

```
docker build -t myapp:latest.
```

- -t myapp:latest: Tags the image.
- .: Refers to the current directory as build context.

Build Options:

- -f <file>: Specify a custom Dockerfile.
- --no-cache: Build without using the cache.
- --build-arg <arg>: Pass build-time arguments.

5. Managing Docker Images List Images:

```
docker images
```

Remove an Image:

```
docker rmi myapp:latest
```

Run a Container: docker run -p 8080:80 myapp:latest

6. Multi-Stage Builds (Advanced)

Multi-stage builds help reduce image size by separating the build environment from runtime:

```
# Stage 1: Build stage
FROM node:14 AS build
WORKDIR /app
COPY package.json ./
RUN npm install
COPY .

# Stage 2: Production stage
FROM node:14-slim
WORKDIR /app
COPY --from=build /app /app
CMD ["node", "server.js"]
```

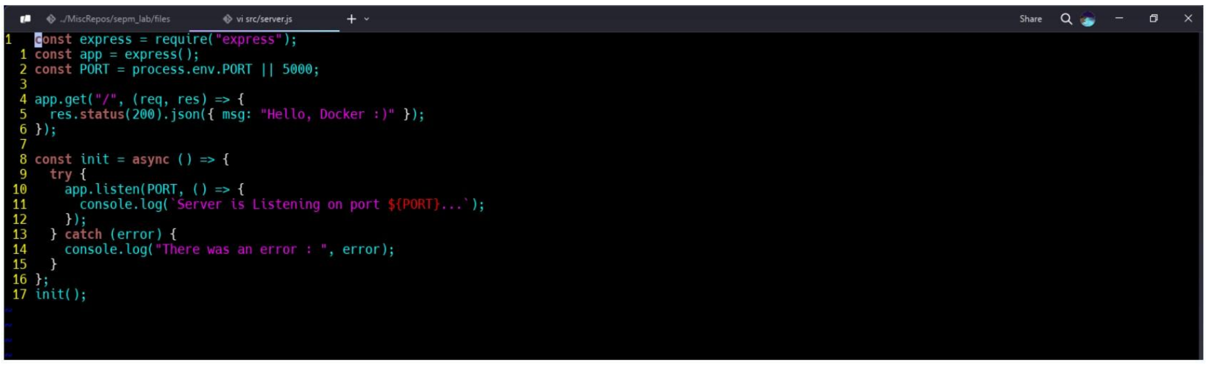
This keeps the final image small and excludes unnecessary build tools.

7. Best Practices for Dockerfiles

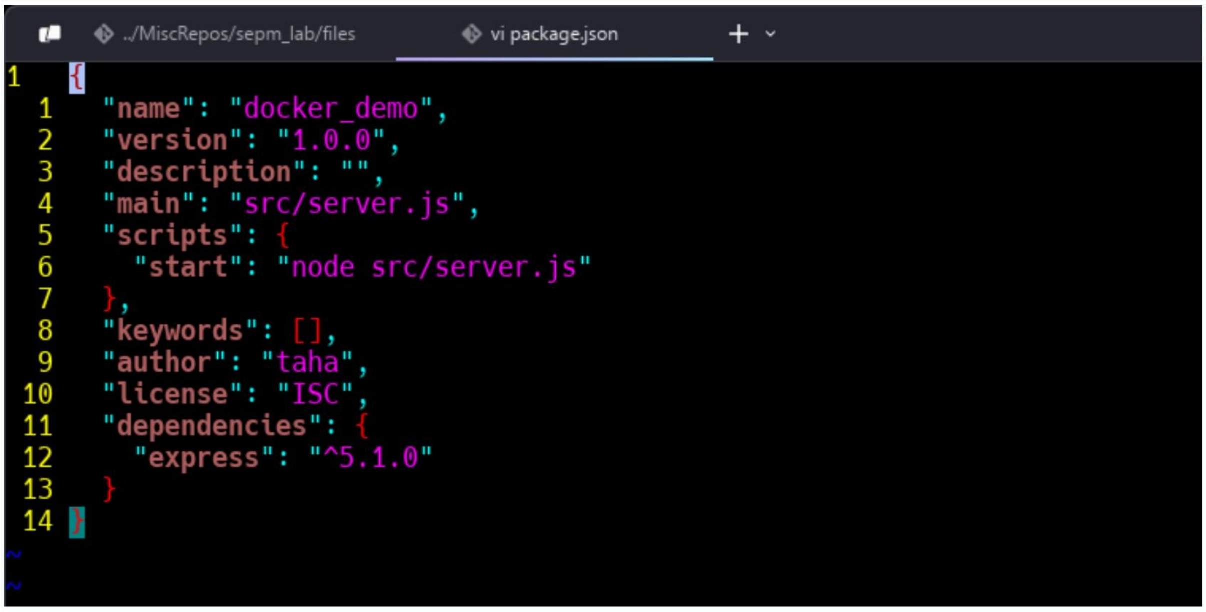
1. Use minimal base images (e.g., alpine).
2. Order instructions from least to most frequently changing to leverage caching.
3. Combine RUN commands with &&.
4. Avoid root – use non-root users.
5. Clean up unnecessary files to reduce image size.

Name: Harshi Lodha **Roll NO: 55** **Batch: T13**

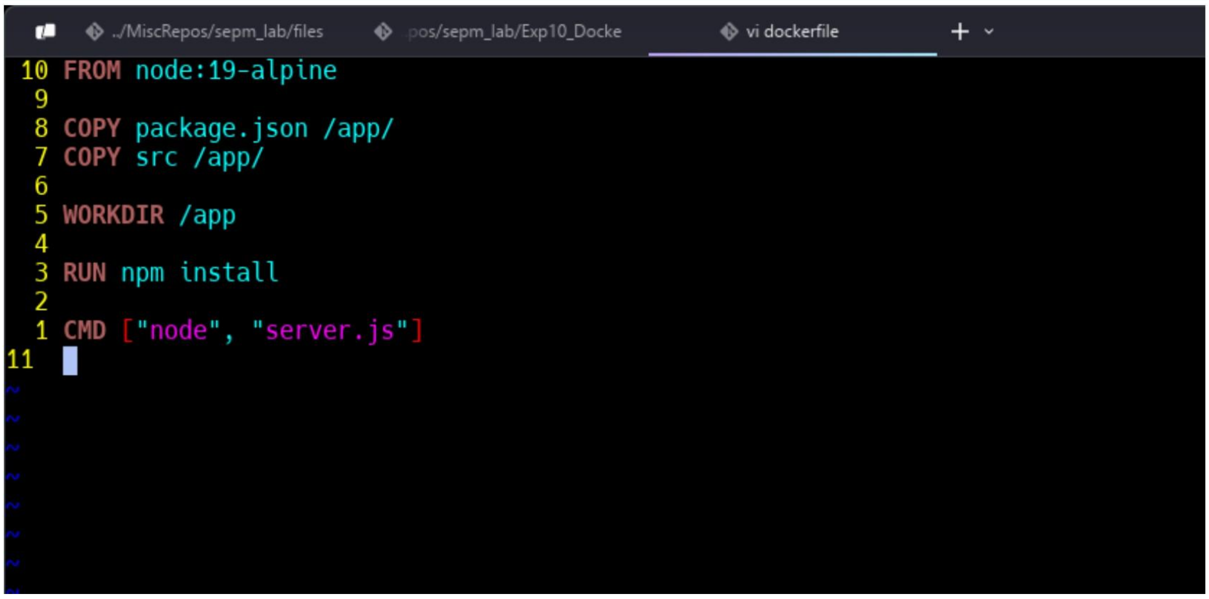
OUTPUT:



```
const express = require("express");
const app = express();
const PORT = process.env.PORT || 5000;

app.get("/", (req, res) => {
  res.status(200).json({ msg: "Hello, Docker :)" });
});

const init = async () => {
  try {
    app.listen(PORT, () => {
      console.log(`Server is Listening on port ${PORT}...`);
    });
  } catch (error) {
    console.log("There was an error : ", error);
  }
};
init();
```

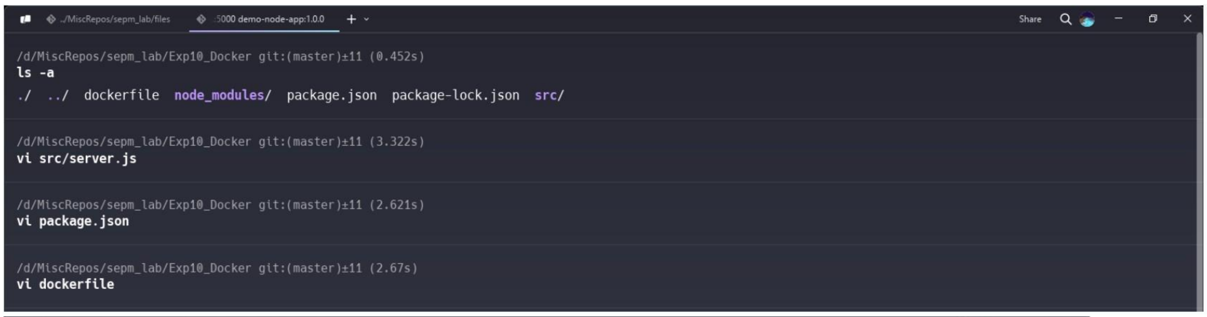


```
{
  "name": "docker_demo",
  "version": "1.0.0",
  "description": "",
  "main": "src/server.js",
  "scripts": {
    "start": "node src/server.js"
  },
  "keywords": [],
  "author": "taha",
  "license": "ISC",
  "dependencies": {
    "express": "^5.1.0"
  }
}
```

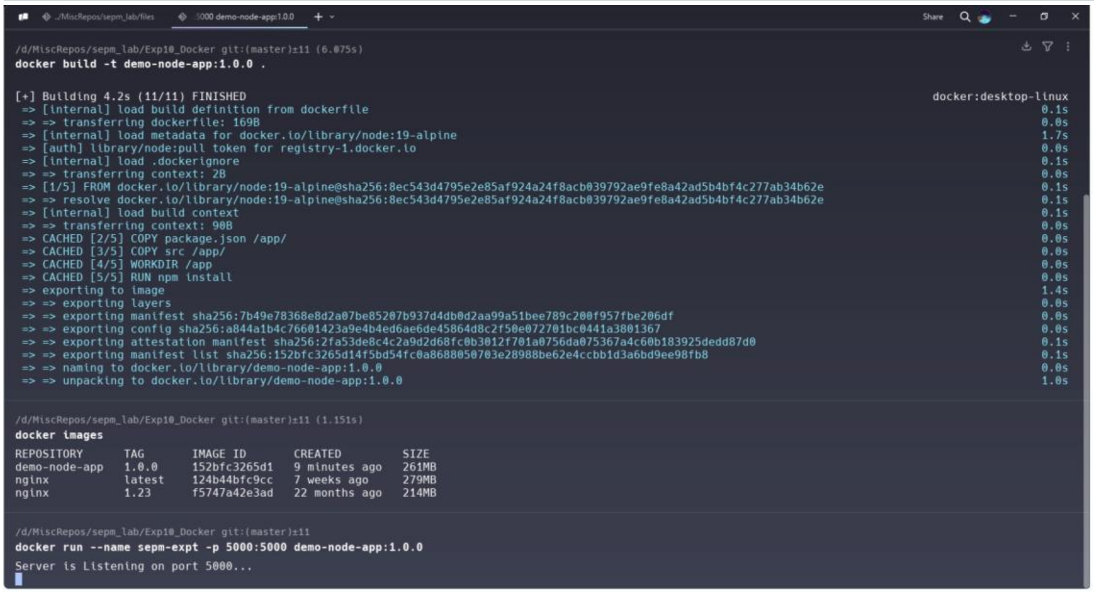


```
FROM node:19-alpine

COPY package.json /app/
COPY src /app/

WORKDIR /app

RUN npm install

CMD ["node", "server.js"]

```

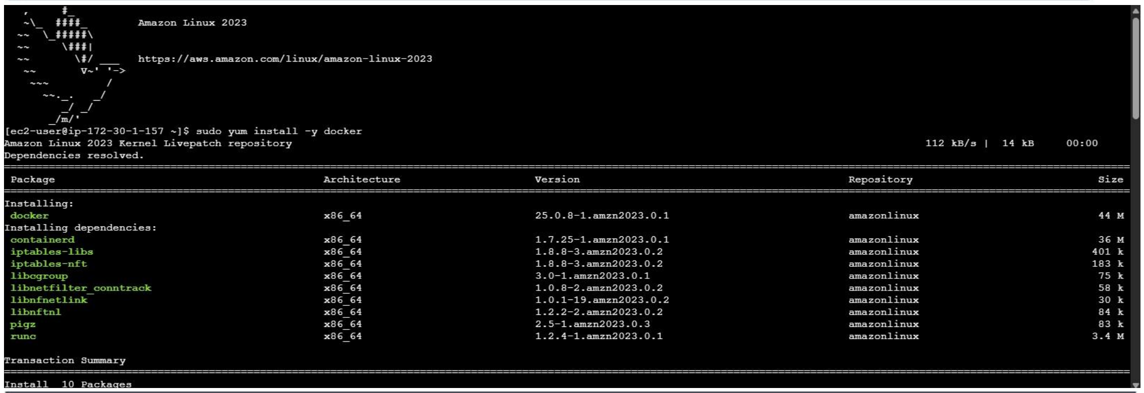
Name: Harshi Lodha **Roll NO: 55** **Batch: T13**



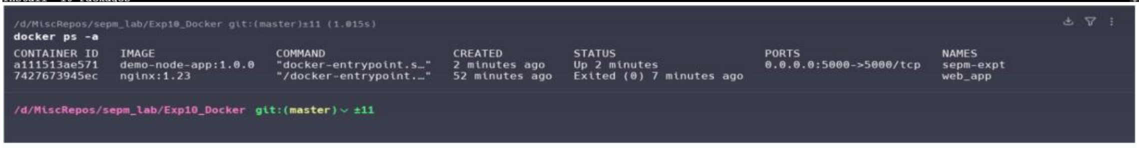
```
/d/MiscRepos/sepm_lab/Exp10_Docker git:(master)±11 (0.452s)
ls -a
./  ../  dockerfile  node_modules/  package.json  package-lock.json  src/

/d/MiscRepos/sepm_lab/Exp10_Docker git:(master)±11 (3.322s)
vi src/server.js

/d/MiscRepos/sepm_lab/Exp10_Docker git:(master)±11 (2.621s)
vi package.json

/d/MiscRepos/sepm_lab/Exp10_Docker git:(master)±11 (2.67s)
vi dockerfile
```



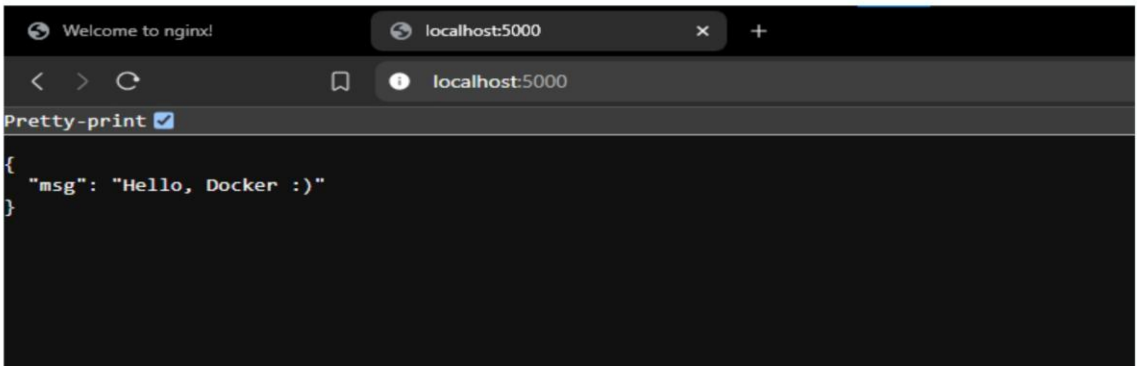
```
/d/MiscRepos/sepm_lab/Exp10_Docker git:(master)±11 (6.075s)
docker build -t demo-node-app:1.0.0 .

[+] Building 4.2s (11/11) FINISHED                                                                               docker:desktop-linux
 => [internal] load build definition from dockerfile                                                                              0.1s
 => => transferring dockerfile: 169B                                                                                              0.0s
 => [internal] load metadata for docker.io/library/node:19-alpine                                                                 1.7s
 => [auth] library/node:pull token for registry-1.docker.io                                                                       0.0s
 => [internal] load .dockerignore                                                                                                 0.1s
 => => transferring context: 2B                                                                                                   0.0s
 => [1/5] FROM docker.io/library/node:19-alpine@sha256:8ec543d4795e2e85af924a24f8acb039792ae9fe8a42ad5b4bf4c277ab34b62e           0.1s
 => => resolve docker.io/library/node:19-alpine@sha256:8ec543d4795e2e85af924a24f8acb039792ae9fe8a42ad5b4bf4c277ab34b62e           0.1s
 => [internal] load build context                                                                                                 0.1s
 => => transferring context: 90B                                                                                                  0.0s
 => CACHED [2/5] COPY package.json /app/                                                                                          0.0s
 => CACHED [3/5] COPY src /app/                                                                                                   0.0s
 => CACHED [4/5] WORKDIR /app                                                                                                     0.0s
 => CACHED [5/5] RUN npm install                                                                                                  0.0s
 => exporting to image                                                                                                            1.4s
 => => exporting layers                                                                                                           0.0s
 => => exporting manifest sha256:7b49e78368e8d2a07be85207b937d4db0d2aa99a51bee789c200f957fbe206df                                 0.0s
 => => exporting config sha256:a844a1b4c76601423a9e4b4ed6ae6de45864d8c2f50e072701bc0441a3801367                                   0.0s
 => => exporting attestation manifest sha256:2fa53de8c4c2a9d2d68fc0b3012f701a0756da075367a4c60b183925dedd87d0                     0.1s
 => => exporting manifest list sha256:152bfc3265d14f5bd54fc0a8688050703e28988be62e4ccbb1d3a6bd9ee98fb8                            0.1s
 => => naming to docker.io/library/demo-node-app:1.0.0                                                                            0.0s
 => => unpacking to docker.io/library/demo-node-app:1.0.0                                                                         1.0s

/d/MiscRepos/sepm_lab/Exp10_Docker git:(master)±11 (1.151s)
docker images
REPOSITORY      TAG       IMAGE ID       CREATED          SIZE
demo-node-app   1.0.0     152bfc3265d1   9 minutes ago    261MB
nginx           latest    124b44bfc9cc   7 weeks ago      279MB
nginx           1.23      f5747a42e3ad   22 months ago    214MB

/d/MiscRepos/sepm_lab/Exp10_Docker git:(master)±11
docker run --name sepm-expt -p 5000:5000 demo-node-app:1.0.0
Server is Listening on port 5000...
```



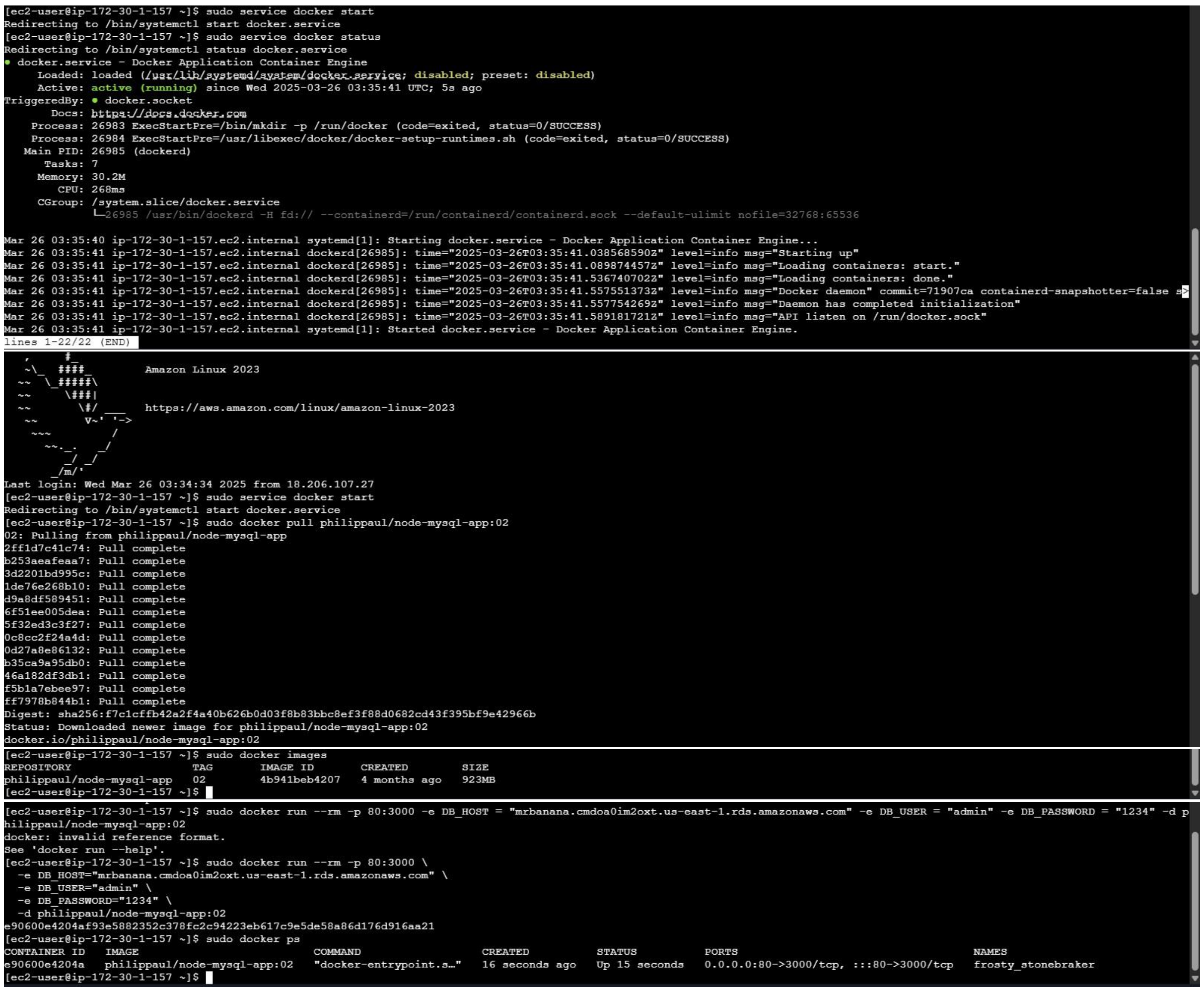
```
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-30-1-157 ~]$ sudo yum install -y docker
Amazon Linux 2023 Kernel Livepatch repository                                              112 kB/s |  14 kB     00:00
Dependencies resolved.
==========================================================================================================================
 Package                      Architecture          Version                          Repository                      Size
==========================================================================================================================
Installing:
 docker                       x86_64                25.0.8-1.amzn2023.0.1            amazonlinux                     44 M
Installing dependencies:
 containerd                   x86_64                1.7.25-1.amzn2023.0.1            amazonlinux                     36 M
 iptables-libs                x86_64                1.8.8-3.amzn2023.0.2             amazonlinux                    401 k
 iptables-nft                 x86_64                1.8.8-3.amzn2023.0.2             amazonlinux                    183 k
 libcgroup                    x86_64                3.0-1.amzn2023.0.1               amazonlinux                     75 k
 libnetfilter_conntrack       x86_64                1.0.8-2.amzn2023.0.2             amazonlinux                     58 k
 libnfnetlink                 x86_64                1.0.1-19.amzn2023.0.2            amazonlinux                     30 k
 libnftnl                     x86_64                1.2.2-2.amzn2023.0.2             amazonlinux                     84 k
 pigz                         x86_64                2.5-1.amzn2023.0.3               amazonlinux                     83 k
 runc                         x86_64                1.2.4-1.amzn2023.0.1             amazonlinux                    3.4 M

Transaction Summary
==========================================================================================================================
Install  10 Packages
```



```
/d/MiscRepos/sepm_lab/Exp10_Docker git:(master)±11 (1.015s)
docker ps -a
CONTAINER ID   IMAGE                 COMMAND                  CREATED          STATUS                     PORTS                    NAMES
a111513ae571   demo-node-app:1.0.0   "docker-entrypoint.s…"   2 minutes ago    Up 2 minutes               0.0.0.0:5000->5000/tcp   sepm-expt
7427673945ec   nginx:1.23            "/docker-entrypoint.…"   52 minutes ago   Exited (0) 7 minutes ago                            web_app

/d/MiscRepos/sepm_lab/Exp10_Docker git:(master)✓ ±11
```



```
{
  "msg": "Hello, Docker :)"
}
```



```
[ec2-user@ip-172-30-1-157 ~]$ sudo service docker start
Redirecting to /bin/systemctl start docker.service
[ec2-user@ip-172-30-1-157 ~]$ sudo service docker status
Redirecting to /bin/systemctl status docker.service
● docker.service - Docker Application Container Engine
     Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; preset: disabled)
     Active: active (running) since Wed 2025-03-26 03:35:41 UTC; 5s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
    Process: 26983 ExecStartPre=/bin/mkdir -p /run/docker (code=exited, status=0/SUCCESS)
    Process: 26984 ExecStartPre=/usr/libexec/docker/docker-setup-runtimes.sh (code=exited, status=0/SUCCESS)
   Main PID: 26985 (dockerd)
      Tasks: 7
     Memory: 30.2M
        CPU: 268ms
     CGroup: /system.slice/docker.service
             └─26985 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-ulimit nofile=32768:65536

Mar 26 03:35:40 ip-172-30-1-157.ec2.internal systemd[1]: Starting docker.service - Docker Application Container Engine...
Mar 26 03:35:41 ip-172-30-1-157.ec2.internal dockerd[26985]: time="2025-03-26T03:35:41.038568590Z" level=info msg="Starting up"
Mar 26 03:35:41 ip-172-30-1-157.ec2.internal dockerd[26985]: time="2025-03-26T03:35:41.089874457Z" level=info msg="Loading containers: start."
Mar 26 03:35:41 ip-172-30-1-157.ec2.internal dockerd[26985]: time="2025-03-26T03:35:41.536740702Z" level=info msg="Loading containers: done."
Mar 26 03:35:41 ip-172-30-1-157.ec2.internal dockerd[26985]: time="2025-03-26T03:35:41.557551373Z" level=info msg="Docker daemon" commit=71907ca containerd-snapshotter=false s>
Mar 26 03:35:41 ip-172-30-1-157.ec2.internal dockerd[26985]: time="2025-03-26T03:35:41.557754269Z" level=info msg="Daemon has completed initialization"
Mar 26 03:35:41 ip-172-30-1-157.ec2.internal dockerd[26985]: time="2025-03-26T03:35:41.589181721Z" level=info msg="API listen on /run/docker.sock"
Mar 26 03:35:41 ip-172-30-1-157.ec2.internal systemd[1]: Started docker.service - Docker Application Container Engine.
lines 1-22/22 (END)
```

```
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Wed Mar 26 03:34:34 2025 from 18.206.107.27
[ec2-user@ip-172-30-1-157 ~]$ sudo service docker start
Redirecting to /bin/systemctl start docker.service
[ec2-user@ip-172-30-1-157 ~]$ sudo docker pull philippaul/node-mysql-app:02
02: Pulling from philippaul/node-mysql-app
2ff1d7c41c74: Pull complete
b253aeafeaa7: Pull complete
3d2201bd995c: Pull complete
1de76e268b10: Pull complete
d9a8df589451: Pull complete
6f51ee005dea: Pull complete
5f32ed3c3f27: Pull complete
0c8cc2f24a4d: Pull complete
0d27a8e86132: Pull complete
b35ca9a95db0: Pull complete
46a182df3db1: Pull complete
f5b1a7ebee97: Pull complete
ff7978b844b1: Pull complete
Digest: sha256:f7c1cffb42a2f4a40b626b0d03f8b83bbc8ef3f88d0682cd43f395bf9e42966b
Status: Downloaded newer image for philippaul/node-mysql-app:02
docker.io/philippaul/node-mysql-app:02
```

```
[ec2-user@ip-172-30-1-157 ~]$ sudo docker images
REPOSITORY                  TAG       IMAGE ID       CREATED        SIZE
philippaul/node-mysql-app   02        4b941beb4207   4 months ago   923MB
[ec2-user@ip-172-30-1-157 ~]$
```

```
[ec2-user@ip-172-30-1-157 ~]$ sudo docker run --rm -p 80:3000 -e DB_HOST = "mrbanana.cmdoa0im2oxt.us-east-1.rds.amazonaws.com" -e DB_USER = "admin" -e DB_PASSWORD = "1234" -d p
hilippaul/node-mysql-app:02
docker: invalid reference format.
See 'docker run --help'.
[ec2-user@ip-172-30-1-157 ~]$ sudo docker run --rm -p 80:3000 \
  -e DB_HOST="mrbanana.cmdoa0im2oxt.us-east-1.rds.amazonaws.com" \
  -e DB_USER="admin" \
  -e DB_PASSWORD="1234" \
  -d philippaul/node-mysql-app:02
e90600e4204af93e5882352c378fc2c94223eb617c9e5de58a86d176d916aa21
[ec2-user@ip-172-30-1-157 ~]$ sudo docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED          STATUS          PORTS                                   NAMES
e90600e4204a   philippaul/node-mysql-app:02   "docker-entrypoint.s…"   16 seconds ago   Up 15 seconds   0.0.0.0:80->3000/tcp, :::80->3000/tcp   frosty_stonebraker
[ec2-user@ip-172-30-1-157 ~]$
```

Conclusion:

We have learnt Dockerfile instructions, built an image for a sample web application using DOCKERFILE.