Name: Meet Brijwani

Batch: T11

Roll no: 14

Experiment 10

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

THEORY:

A **Dockerfile** is a text file that contains a list of instructions for Docker to build an image. It automates the process of creating a Docker image by specifying everything your app needs to run — from base images, dependencies, to startup commands.

Common Dockerfile Instructions:

Instruction	Description
`FROM`	Specifies the base image (e.g., `node:18-alpine`, `python:3.10`)
`WORKDIR`	Sets the working directory inside the container
`COPY`	Copies files from your system to the container
`RUN`	Executes commands to install dependencies or perform setup
`CMD`	Specifies the default command to run when the container starts
`EXPOSE`	Documents the port the container will listen on
`ENV`	Sets environment variables
`ENTRYPOINT`	Like `CMD`, but used when you want the command to always run

---

Practical: Build a Docker Image for a Sample Web App

Let's take a **Node.js Express** web application as an example.

1. Project Structure:

```
sample-app/
├── Dockerfile
├── package.json
├── package-lock.json
└── index.js
```

2. `package.json`

```
json
CopyEdit
{
  "name": "sample-app",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "start": "node index.js"
  },
  "dependencies": {
    "express": "^4.18.2"
  }
}
```

3. `index.js`

```
js

const express = require('express');
const app = express();
const PORT = 3000;

app.get('/', (req, res) => {
  res.send('Hello, Docker!');
});

app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});
```

4. `Dockerfile`

```
dockerfile

# Step 1: Use an official Node.js runtime as a parent image
FROM node:18-alpine

# Step 2: Set working directory
WORKDIR /app

# Step 3: Copy package.json and package-lock.json
COPY package*.json ./

# Step 4: Install dependencies
RUN npm install

# Step 5: Copy source code
COPY . .

# Step 6: Expose the port your app runs on
EXPOSE 3000

# Step 7: Define the command to run the app
CMD ["npm", "start"]
```

Build & Run the Image

Build the Docker Image:

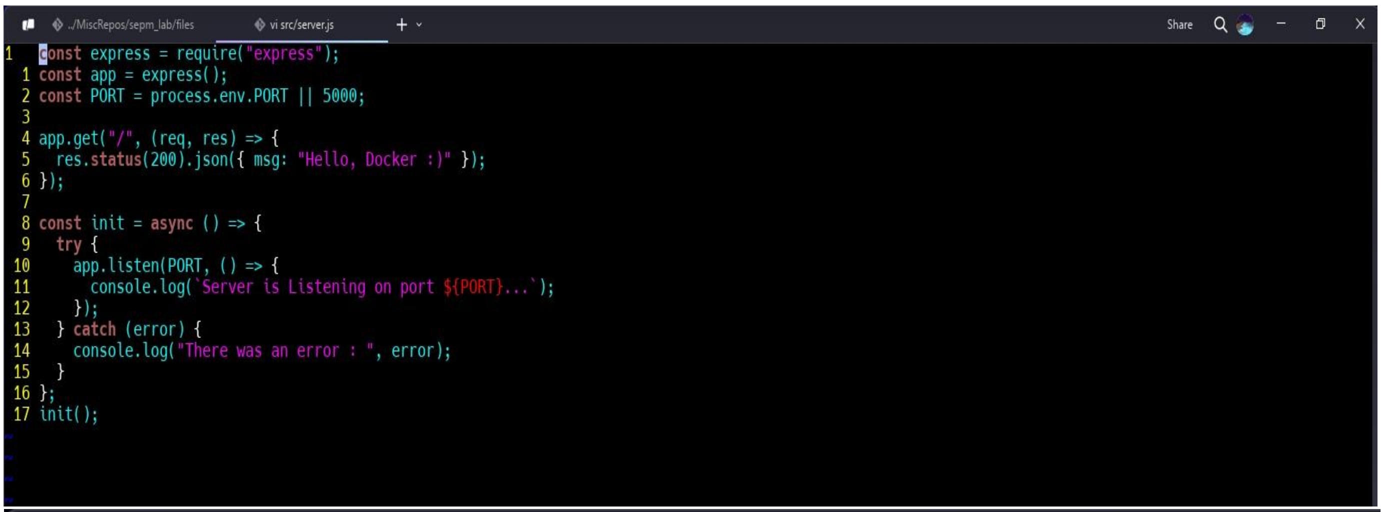
```
bash

docker build -t sample-node-app .
```

Run the Docker Container:

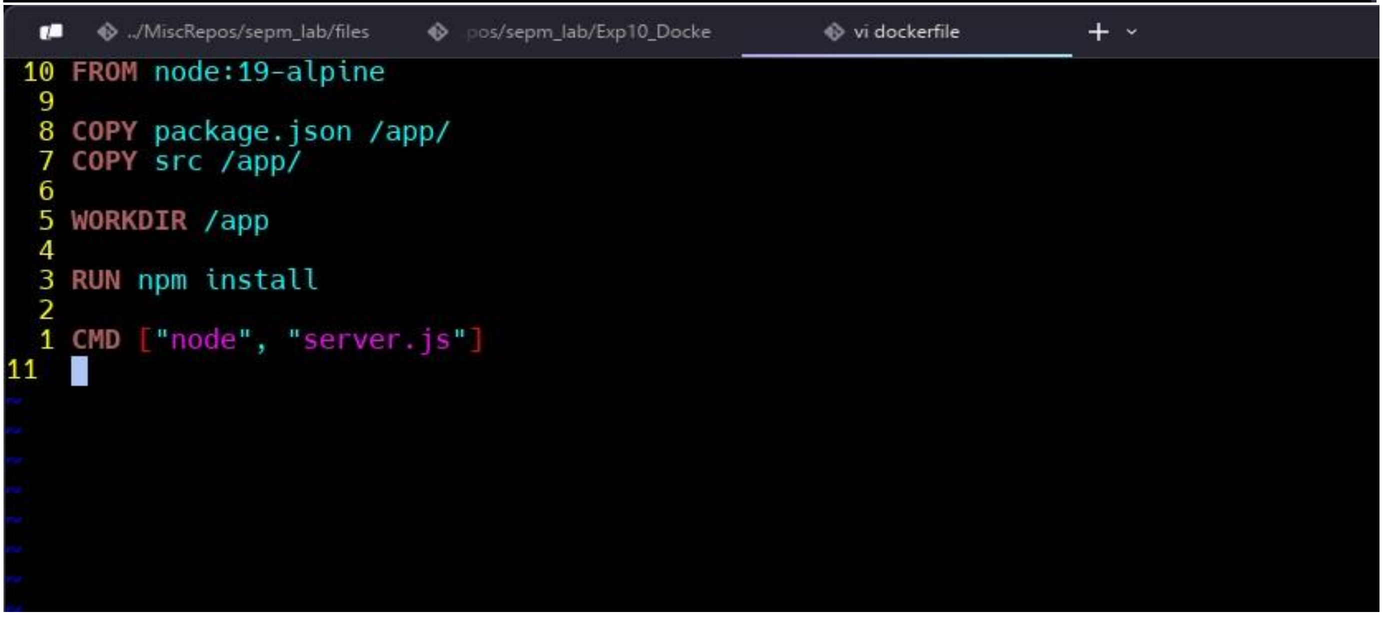
```
bash

docker run -p 3000:3000 sample-node-app
```

Now, visit `http://localhost:3000` and you'll see **"Hello, Docker!"**

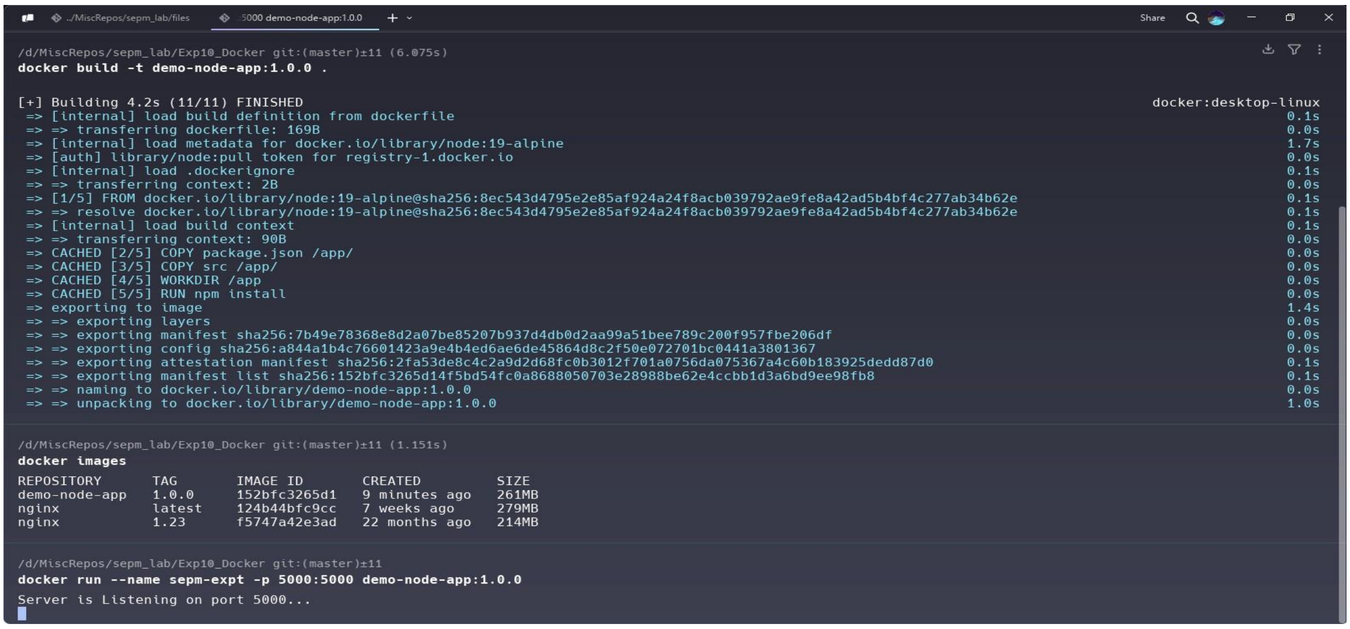
SCREENSHOTS:


../MiscRepos/sepm_lab/files | vi src/server.js

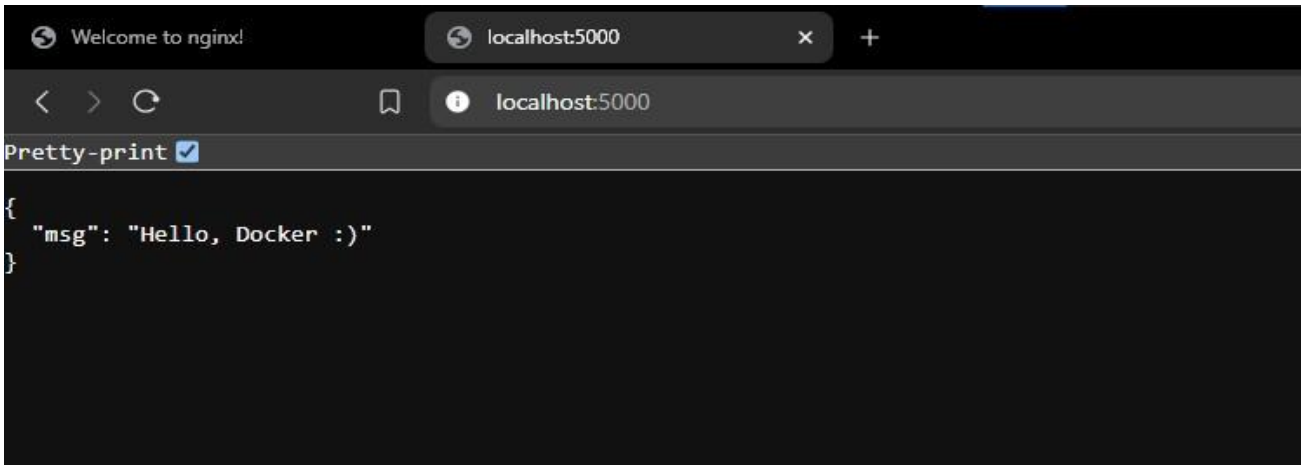
```
const express = require("express");
const app = express();
const PORT = process.env.PORT || 5000;

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();
```


../MiscRepos/sepm_lab/files | pos/sepm_lab/Exp10_Docke | vi dockerfile

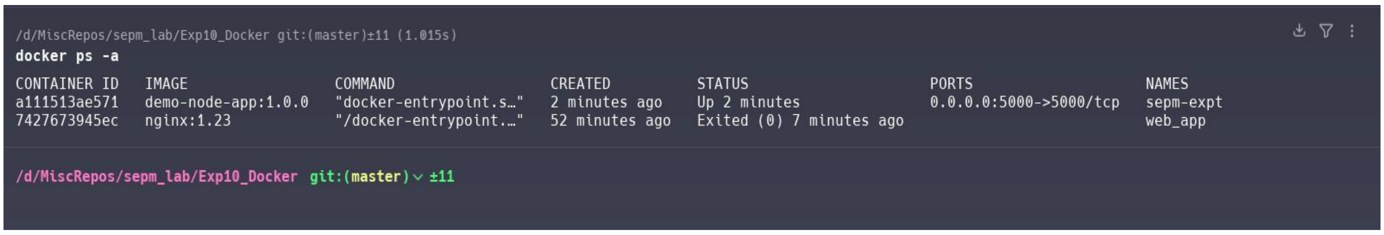
```
FROM node:19-alpine

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

WORKDIR /app

RUN npm install

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



```
/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...
```



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



```
/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
```

CONCLUSION:

Hence, we have learnt Docker file instructions, build an image for a sample web application using DOCKERFILE.