Experiment -2.1

Student Name: Chayan Gope
UID: 22BDO10036
Branch: AIT-CSE-DevOps
Section/Group: 22BCD-1(A)
Semester: 5
Date of Performance: 16/09/24
Subject Name: Docker and Kubernetes
Subject Code: 22CSH-343

1. Aim/Overview of the practical:

To run a *node.js* application using Docker and manage the Docker volume.

2. Apparatus: PC, Docker Engine, DockerHub, Ubuntu Linux

3. Steps for experiment/practical:

- **Dockerfile**:
  1. A Dockerfile is a script that contains instructions for building a customized docker image.
  2. Each instruction in a Dockerfile creates a new layer in the image, and the final image is composed of all the layers stacked on top of each other.
  3. Dockerfile uses a simple, easy-to-read syntax that can be created and edited with any text editor.
  4. Once a Dockerfile has been created, it can be used to build an image using the ***docker build*** command.
  5. Dockerfiles enable faster and more efficient deployment of applications.
  6. Dockerfiles can be used in automation testing to ***build*** and ***run*** test environments for different applications and services.
  7. We can easily integrate your Dockerfile with continuous integration and continuous deployment (CI/CD) pipelines.

- **Some Dockerfile instructions:**
  1. **FROM:** specify the base image we want to start from.
  2. **RUN:** used to run commands during the image build process.
  3. **COPY:** used to copy a file or folder from the host system into the docker image.
  4. **EXPOSE:** specify the port you want the docker image to listen to at runtime.
  5. **WORKDIR:** used to set the current working directory.
  6. **VOLUME:** used to create or mount the volume to the Docker container
  7. **CMD:** Executes a command within a running container. Only one CMD instruction is allowed, and if multiple are present, only the last one takes effect.

- **Node project and docker image creation**
  1. Create a folder for the project and create a file named “hello.js” in that folder.

```
var http = require('http');

http.createServer(function (req, res) {
   res.writeHead(200, {'Content-Type': 'text/html'});
   res.end('Hello World!');
}).listen(8080);

console.log("Server running on port 8080!!");
console.log("Code given by Kamaljit Sir");
```

  2. Initialize the folder as a node project using the **npm init -y** command.

  3. Install the **http** package using the **npm install http** command.

4. Create the **Dockerfile** and write the following code in it.

```
FROM node:latest AS builder
WORKDIR /app
COPY package*.json .
RUN npm install
COPY . .

FROM node:slim
WORKDIR /app
COPY --from=builder /app .
CMD ["node","hello.js"]
EXPOSE 8080
```

5. Build the docker image using the "**docker build -t <image_name> .**" command and then run the container using the **docker run** command.

```
chayan@chayan-virtual-machine:~/Desktop/exp5$ vi hello.js
chayan@chayan-virtual-machine:~/Desktop/exp5$ nano hello.js
chayan@chayan-virtual-machine:~/Desktop/exp5$ npm init -y
Wrote to /home/chayan/Desktop/exp5/package.json:

{
  "name": "exp5",
  "version": "1.0.0",
  "description": "",
  "main": "hello.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}


chayan@chayan-virtual-machine:~/Desktop/exp5$ ls
hello.js  package.json
chayan@chayan-virtual-machine:~/Desktop/exp5$ npm i http

added 1 package, and audited 2 packages in 2s

found 0 vulnerabilities
chayan@chayan-virtual-machine:~/Desktop/exp5$ nano Dockerfile
```

```
chayan@chayan-virtual-machine:~/Desktop/exp5$ docker build -t exp5 .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  10.24kB
Step 1/10 : FROM node:latest AS builder
 ---> 2ef13a9c33b0
Step 2/10 : WORKDIR /app
 ---> Using cache
 ---> f06cb9c76d02
Step 3/10 : COPY package*.json ./
 ---> 109b91e9a276
Step 4/10 : RUN npm install
 ---> Running in 2bc7668fde72

added 1 package, and audited 2 packages in 3s

found 0 vulnerabilities
Removing intermediate container 2bc7668fde72
 ---> 4fe004da73a5
Step 5/10 : COPY . .
 ---> 4ea57cfd2f0b
Step 6/10 : FROM node:slim
slim: Pulling from library/node
a2318d6c47ec: Already exists
```

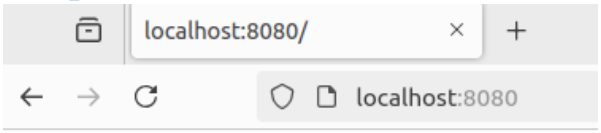
```
chayan@chayan-virtual-machine:~/Desktop/exp5$ docker run -d -p 8080:8080 exp5
8285e5943e6a14939c8c7b6c8e356e8f567d8957129cb5c30984ff94df33ccad
chayan@chayan-virtual-machine:~/Desktop/exp5$ docker ps -a
CONTAINER ID   IMAGE         COMMAND                  CREATED              STATUS
  PORTS                                       NAMES
8285e5943e6a   exp5          "docker-entrypoint.s…"   About a minute ago   Up About a minute
  0.0.0.0:8080->8080/tcp, :::8080->8080/tcp   fervent_burnell
```

```
chayan@chayan-virtual-machine:~/Desktop/exp5$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
exp5          latest    a389097acea5   2 minutes ago    215MB
```

```
chayan@chayan-virtual-machine:~/Desktop/exp5$ docker tag exp5:latest chayan12/exp5:latest
chayan@chayan-virtual-machine:~/Desktop/exp5$ docker push chayan12/exp5:latest
The push refers to repository [docker.io/chayan12/exp5]
1b1e33aad4c0: Pushed
```

4.Result/Output/Writing Summary:

We can access the node application in our host machine by entering "http://localhost:8080/" in the browser.




```
chayan@chayan-virtual-machine:~/Desktop/exp5$ node hello.js
Server running on port 8080!!
Code Given by Kamaljit Sir
```

Learning outcomes (What I have learnt):

1. I have learnt the concept of containerization.
2. I have learnt to configure Docker to work with different environments.
3. I have learnt how to build docker images using Dockerfile.
4. I have learnt the purpose of Dockerfile and its advantages.
5. I have learnt how Dockerfile can help create CI/CD pipelines.

Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.			
2.			
3.			