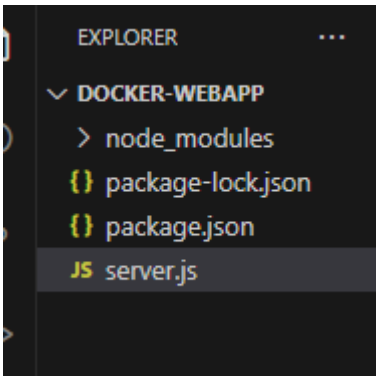
Assignment - 1

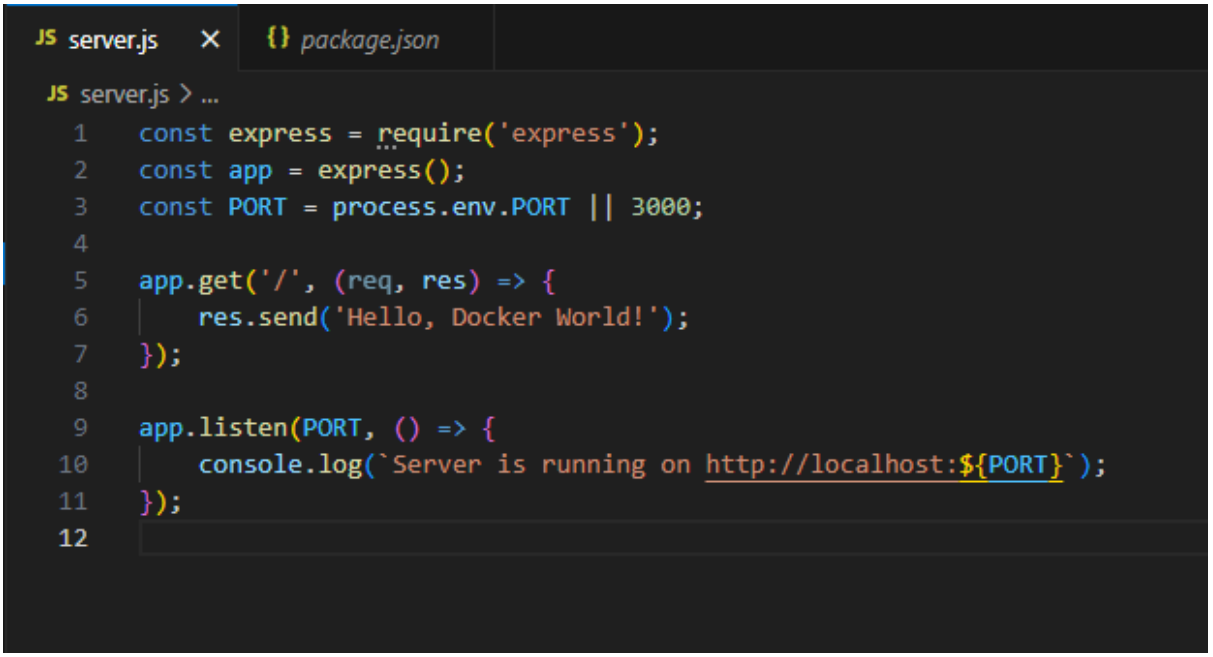
1. Create a web application or use an existing one (don't use the project used in labs).

- Creating a Simple Web Application using Node.js(Express.js).

1.1 Setting up the project Structure:

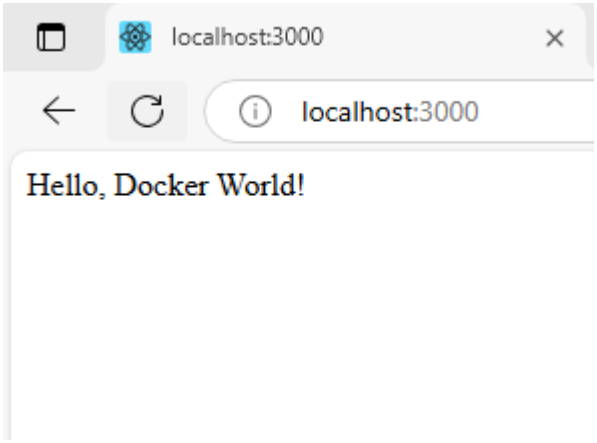


- Used “npm init -y” to initializes a package.json file.
- created server.js using “echo. > server.js” .



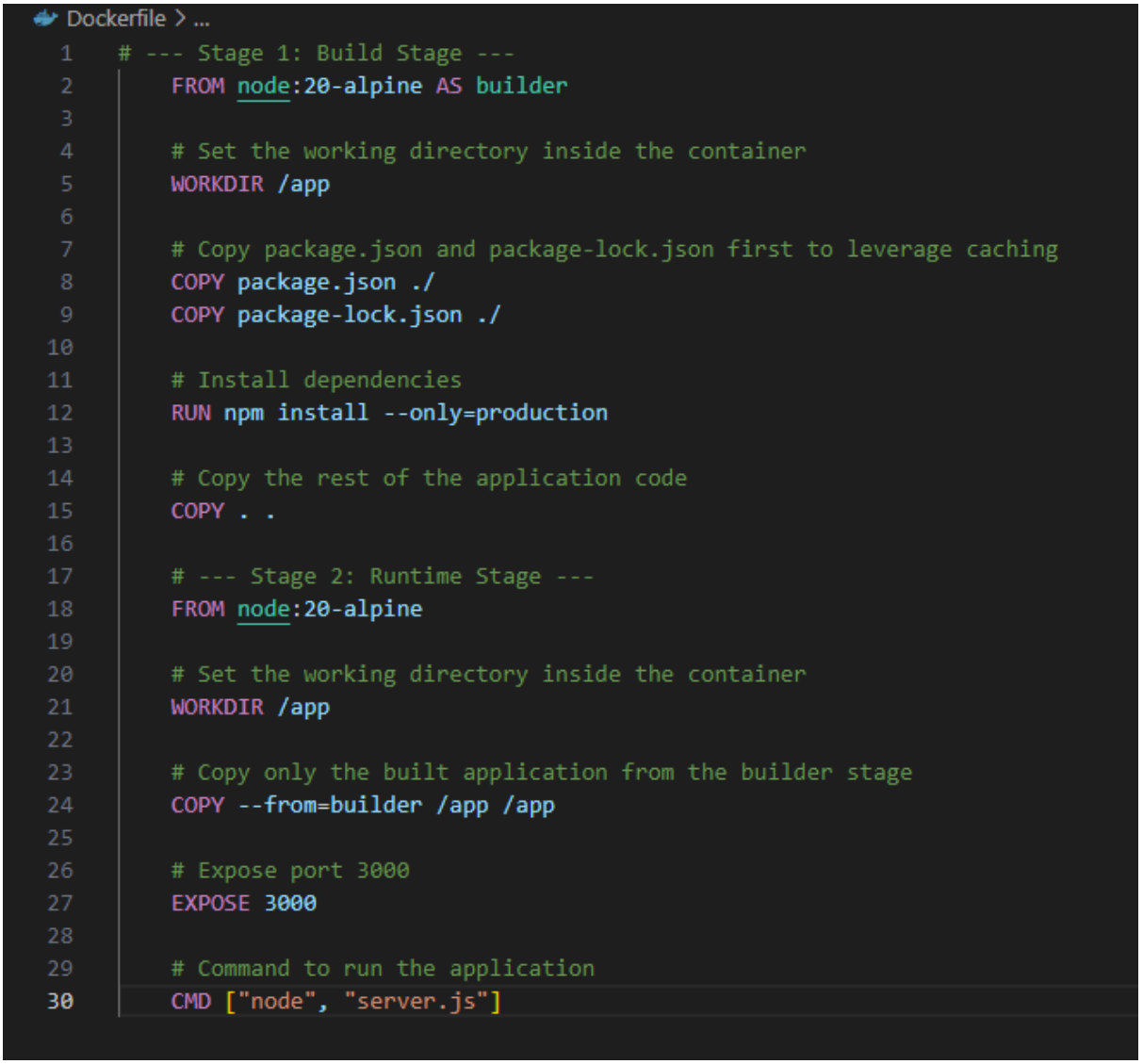
1.2 Tested Application Locally:

- By running “node server.js”
- Opening http://localhost:3000/.



2. Write a Dockerfile (Multistage if possible) to containerize the application.

- Created a MultiStage Dockerfile.



```
Dockerfile > ...
# --- Stage 1: Build Stage ---
FROM node:20-alpine AS builder

# Set the working directory inside the container
WORKDIR /app

# Copy package.json and package-lock.json first to leverage caching
COPY package.json ./
COPY package-lock.json ./

# Install dependencies
RUN npm install --only=production

# Copy the rest of the application code
COPY . .

# --- Stage 2: Runtime Stage ---
FROM node:20-alpine

# Set the working directory inside the container
WORKDIR /app

# Copy only the built application from the builder stage
COPY --from=builder /app /app

# Expose port 3000
EXPOSE 3000

# Command to run the application
CMD ["node", "server.js"]
```

Explanation of Multi-Stage Dockerfile:

- First Stage (builder)

    Used a lightweight Node.js image (node:20-alpine).

    Installs dependencies.

    Copies all project files.

- Second Stage (runtime)

    Used another lightweight Node.js Alpine image.

    Copies the built app from the previous stage to reduce image size.

    Exposes port 3000 and runs the app.

3. Building the Docker image:

3.1 Build the Docker Image:

- Using “docker build -t my-webapp .”

```
C:\Users\admin\Documents\docker-webapp>docker build -t my-webapp .
[+] Building 21.8s (13/13) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 825B
 => [internal] load metadata for docker.io/library/node:20-alpine
 => [auth] library/node:pull token for registry-1.docker.io
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load build context
 => => transferring context: 2.32MB
```

- -t my-webapp, Assigns the name my-webapp to the image.
- ., Specifies the current directory as the build context.

3.2 Check the Built Image Size:

- Using “docker images” .

```
C:\Users\admin\Documents\docker-webapp>docker images
REPOSITORY   TAG       IMAGE ID       CREATED         SIZE
my-webapp    latest    2986fde21a85   5 minutes ago   195MB

C:\Users\admin\Documents\docker-webapp>
```

- As I have used Alpine-based images (node:20-alpine), the size of the image will be small compared to full Node.js images which will around 500MB.

4. Push Docker Image to Docker Hub with 3 Different Tags:

4.1 Login to Docker Hub using,

- docker login

```
C:\Users\admin\Documents\docker-webapp>docker login
Authenticating with existing credentials...
Login Succeeded

C:\Users\admin\Documents\docker-webapp>
```

4.2 Tagging The Images:

```
C:\Users\admin\Documents\docker-webapp>docker tag my-webapp rishishori51/assignment1:v1.0

C:\Users\admin\Documents\docker-webapp>docker tag my-webapp rishishori51/assignment1:v1.1

C:\Users\admin\Documents\docker-webapp>docker tag my-webapp rishishori51/assignment1:latest

C:\Users\admin\Documents\docker-webapp>
```

```
C:\Users\admin\Documents\docker-webapp>docker images
REPOSITORY                TAG       IMAGE ID       CREATED          SIZE
my-webapp                 latest    2986fde21a85   52 minutes ago   195MB
rishishori51/assignment1  latest    2986fde21a85   52 minutes ago   195MB
rishishori51/assignment1  v1.0      2986fde21a85   52 minutes ago   195MB
rishishori51/assignment1  v1.1      2986fde21a85   52 minutes ago   195MB

C:\Users\admin\Documents\docker-webapp>
```

4.3 Push the Image to Docker Hub:

```
C:\Users\admin\Documents\docker-webapp>docker push rishishori51/assignment1:v1.0
The push refers to repository [docker.io/rishishori51/assignment1]
4102cc97a071: Pushed
6eb7dbb7f2a7: Pushed
905fb610ce71: Pushed
f18232174bc9: Pushed
2a055f0c83be: Pushed
0416551166ae: Pushed
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

C:\Users\admin\Documents\docker-webapp>docker push rishishori51/assignment1:v1.1
The push refers to repository [docker.io/rishishori51/assignment1]
4102cc97a071: Already exists
6eb7dbb7f2a7: Layer already exists
905fb610ce71: Layer already exists
0416551166ae: Layer already exists
2a055f0c83be: Layer already exists
df40448c730c: Layer already exists
f18232174bc9: Layer already exists
v1.1: digest: sha256:2986fde21a858ea6149e7bbbe9278dd3ae5c86b8659be715a6d3b24cf6895d8a size: 856
```

```
C:\Users\admin\Documents\docker-webapp>docker push rishishori51/assignment1:latest
The push refers to repository [docker.io/rishishori51/assignment1]
6eb7dbb7f2a7: Layer already exists
905fb610ce71: Layer already exists
2a055f0c83be: Layer already exists
0416551166ae: Layer already exists
df40448c730c: Layer already exists
f18232174bc9: Layer already exists
4102cc97a071: Already exists
latest: digest: sha256:2986fde21a858ea6149e7bbbe9278dd3ae5c86b8659be715a6d3b24cf6895d8a size: 856
```

rishishori51/assignment1

Last pushed 4 minutes ago · Repository size: 46 MB

Pushing Images related to the assignment in this repo

Add a category

General | Tags | Image Management BETA | Builds | Collaborators | Webhooks | Settings

Tags

This repository contains 3 tag(s).

Tag	OS	Type	Pulled	Pushed
latest		Image	less than 1 day	4 minutes
v1.1		Image	less than 1 day	4 minutes
v1.0		Image	less than 1 day	4 minutes

See all

5. Run the Docker container from the image that you've built & access the web application in a web browser in local machine:

5.1 Run the Docker Container:

- using “docker run -d -p 3000:3000 --name my-web-container rishishori51/assignment1:latest”

```
C:\Users\admin\Documents\docker-webapp>docker run -d -p 3000:3000 --name my-web-container rishishori51/assignmen
59e29e4dfae5c7ab9628072bfdd16ed4c560706e2def45b965159f8447b96855

C:\Users\admin\Documents\docker-webapp>
```

- -d: Runs the container in detached mode (in the background).
- -p 3000:3000: Maps port 3000 of the container to port 3000 on your local machine.
- --name my-web-container: Assigns a name to the container.
- rishishori51/assignment1:latest: The image I have built and pushed.

5.2 Checking if the Container is Running:

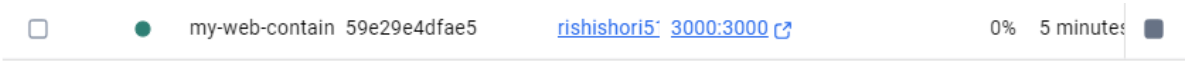
- Using “docker ps”

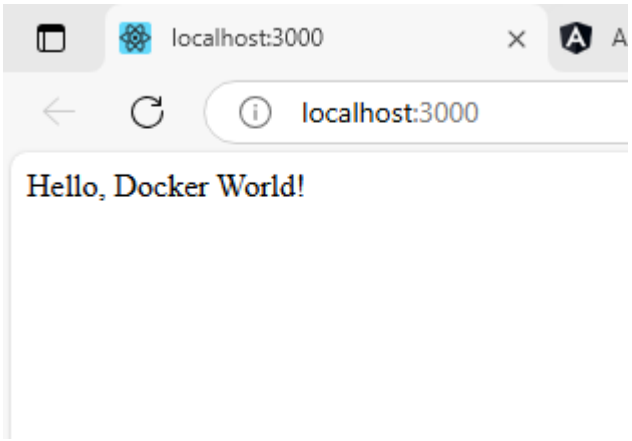
```
C:\Users\admin\Documents\docker-webapp>docker ps
CONTAINER ID   IMAGE                              COMMAND                  CREATED
59e29e4dfae5   rishishori51/assignment1:latest   "docker-entrypoint.s…"   3 minutes ago

C:\Users\admin\Documents\docker-webapp>
```

- Container is Running.

5.3 Accessing the Web Application in Browser:

- Accessing on the http://localhost:3000





6. Start Another Docker Container with the Latest 'nginx' Image:

6.1 Pull the Latest nginx Image:

- Using “docker pull nginx:latest”

```
C:\Users\admin\Documents\docker-webapp>docker pull nginx:latest
latest: Pulling from library/nginx
103f50cb3e9f: Download complete
9dd21ad5a4a6: Download complete
943ea0f0c2e4: Download complete
d014f92d532d: Download complete
513c3649bb14: Download complete
7cf63256a31a: Download complete
bf9acace214a: Download complete
Digest: sha256:9d6b58feebd2dbd3c56ab5853333d627cc6e281011cfd6050fa4bcf2072c9496
Status: Downloaded newer image for nginx:latest
```

6.2 Run the nginx Container:

- Using, “docker run -d --name my-nginx-container nginx:latest”

```
C:\Users\admin\Documents\docker-webapp>docker run -d --name my-nginx-container nginx:latest
7b474124c95f092464bb54de2de6c407f89205b5df79bcfd5a3a3e0c16e7f84b

C:\Users\admin\Documents\docker-webapp>
```

Explanation:

-d: Runs the container in detached mode (in the background).

--name my-nginx-container: Assigns a name to the container.

nginx:latest: Uses the latest version of nginx.

6.3 Verify Running Containers:

- Using,” docker ps”

```
C:\Users\admin\Documents\docker-webapp>docker ps
CONTAINER ID   IMAGE                              COMMAND                  CREATED          STATUS          PORTS                    NAMES
7b474124c95f   nginx:latest                       "/docker-entrypoint.…"   10 minutes ago   Up 10 minutes   80/tcp                   my-nginx-container
59e29e4dfae5   rishishori51/assignment1:latest    "docker-entrypoint.s…"   21 minutes ago   Up 21 minutes   0.0.0.0:3000->3000/tcp   my-web-container

C:\Users\admin\Documents\docker-webapp>
```

```
C:\Users\admin\Documents\docker-webapp>docker ps
CONTAINER ID   IMAGE                                COMMAND
7b474124c95f   nginx:latest                         "/docker-entrypoint.…"
59e29e4dfae5   rishishori51/assignment1:latest      "docker-entrypoint.s…"

C:\Users\admin\Documents\docker-webapp>
```

7. Go inside the application container, inside its terminal, and try to access/ping the nginx container to check the connectivity:

- verifying if the web application container can communicate with the nginx container.

7.1 Get the Network Name:

```
C:\Users\admin\Documents\docker-webapp>docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
1436a44905eb   bridge    bridge    local
5b8faede3cf4   host      host      local
0886d1d553f0   none      null      local
```

7.2 Creating a custom network:

- Using, “docker network create my-custom-network”

```
C:\Users\admin\Documents\docker-webapp>docker network create my-custom-network
98b6d1647ebc1c032b81d858fb13ae5562efb8e48288a1f22b5475f87bc36446
```

Restarting both containers with this network:

- Using,

“docker run -d --name my-webapp --network my-custom-network rishishori51/assignment1:latest”

“docker run -d --name my-nginx --network my-custom-network nginx:latest”

```
C:\Users\admin\Documents\docker-webapp>docker run -d --name my-webapp --network my-custom-network rishishori51/assignment1:latest
2090d34999840878b52bdb516eefd86e1fd84a4f02ddb23580b1af4dd8a7fb2c

C:\Users\admin\Documents\docker-webapp>docker run -d --name my-nginx --network my-custom-network nginx:latest
ca972cc5786b04b381aeebde786c40a8f9b0af5ac54995d1dbcd034e66f75217
```

7.3 Access the application container's terminal:

```
C:\Users\admin\Documents\docker-webapp>docker exec -it my-webapp sh
/app #
```

7.4 Ping the nginx container from inside the application container:

- Using, “ping my-nginx”

```
C:\Users\admin\Documents\docker-webapp>docker exec -it my-webapp sh
/app # ping my-nginx
PING my-nginx (172.18.0.3): 56 data bytes
64 bytes from 172.18.0.3: seq=0 ttl=64 time=0.126 ms
64 bytes from 172.18.0.3: seq=1 ttl=64 time=0.152 ms
64 bytes from 172.18.0.3: seq=2 ttl=64 time=0.053 ms
64 bytes from 172.18.0.3: seq=3 ttl=64 time=0.060 ms
64 bytes from 172.18.0.3: seq=4 ttl=64 time=0.088 ms
64 bytes from 172.18.0.3: seq=5 ttl=64 time=0.056 ms
```

```
64 bytes from 172.18.0.3: seq=72 ttl=64 time=0.084 ms
64 bytes from 172.18.0.3: seq=73 ttl=64 time=0.120 ms
64 bytes from 172.18.0.3: seq=74 ttl=64 time=0.099 ms
64 bytes from 172.18.0.3: seq=75 ttl=64 time=0.089 ms
64 bytes from 172.18.0.3: seq=76 ttl=64 time=0.081 ms
64 bytes from 172.18.0.3: seq=77 ttl=64 time=0.074 ms
^C
--- my-nginx ping statistics ---
78 packets transmitted, 78 packets received, 0% packet loss
round-trip min/avg/max = 0.052/0.095/0.188 ms
/app # exit
```

- The application container (my-webapp) can successfully reach the nginx container (my-nginx).

- The containers are on the same Docker network, meaning they can communicate with each other.