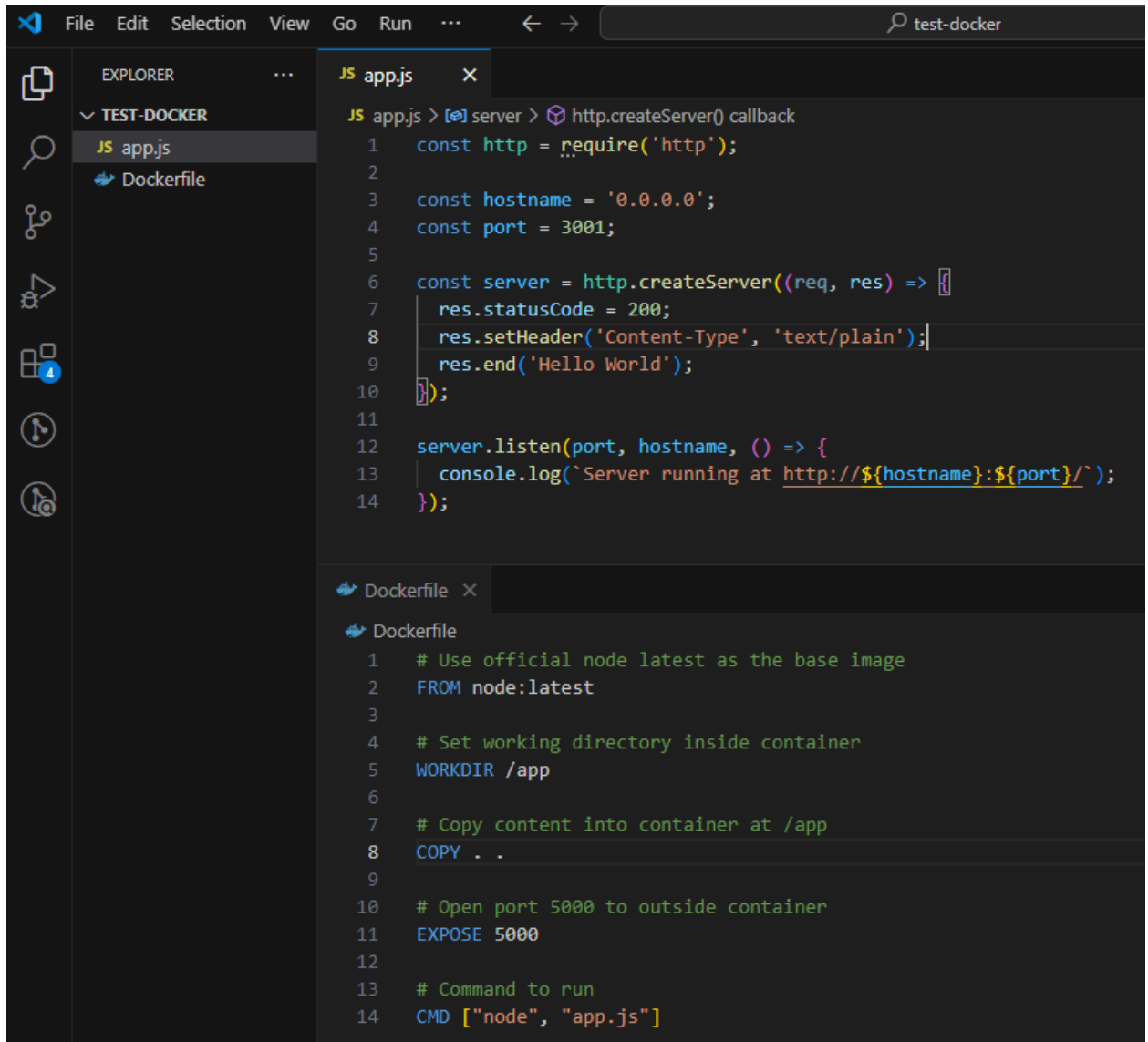


Dockerize

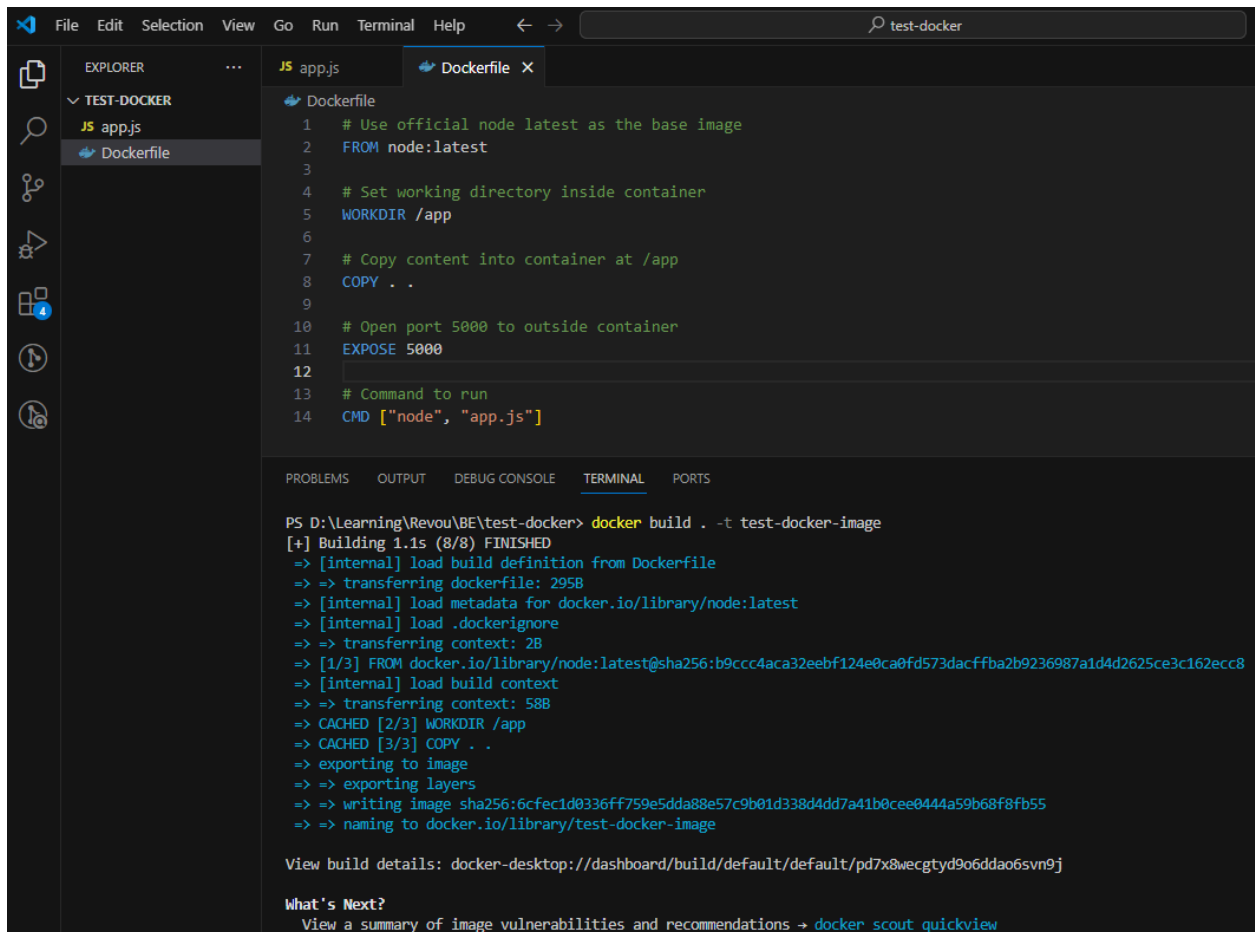
1. Create a Dockerfile

A screenshot of the Visual Studio Code editor interface. The Explorer sidebar on the left shows a project named 'TEST-DOCKER' with two files: 'app.js' and 'Dockerfile'. The 'app.js' file is open in the main editor, displaying a Node.js HTTP server script. The 'Dockerfile' file is also open in a separate tab below 'app.js', showing the Dockerfile instructions for building and running the application. The script in 'app.js' uses 'http.createServer()' and listens on port 3001. The Dockerfile in 'Dockerfile' uses 'node:latest' as the base image, sets the working directory to '/app', copies the current directory's contents, exposes port 5000, and runs the command 'node app.js'.

```
JS app.js > [?] server > http.createServer() callback
1  const http = require('http');
2
3  const hostname = '0.0.0.0';
4  const port = 3001;
5
6  const server = http.createServer((req, res) => {
7    res.statusCode = 200;
8    res.setHeader('Content-Type', 'text/plain');
9    res.end('Hello World');
10 });
11
12 server.listen(port, hostname, () => {
13   console.log(`Server running at http://${hostname}:${port}/`);
14 });
```

```
Dockerfile
Dockerfile
1  # Use official node latest as the base image
2  FROM node:latest
3
4  # Set working directory inside container
5  WORKDIR /app
6
7  # Copy content into container at /app
8  COPY . .
9
10 # Open port 5000 to outside container
11 EXPOSE 5000
12
13 # Command to run
14 CMD ["node", "app.js"]
```

2. Create a docker image using "docker build . -t test-docker-image"



The screenshot shows the VS Code interface with a Dockerfile open in the editor. The Dockerfile contains the following instructions:

```
1 # Use official node latest as the base image
2 FROM node:latest
3
4 # Set working directory inside container
5 WORKDIR /app
6
7 # Copy content into container at /app
8 COPY . .
9
10 # Open port 5000 to outside container
11 EXPOSE 5000
12
13 # Command to run
14 CMD ["node", "app.js"]
```

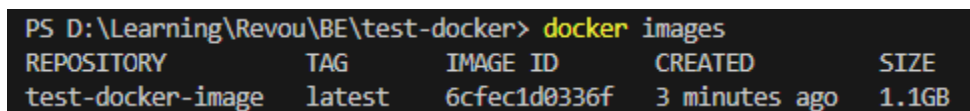
The terminal at the bottom shows the output of the command `docker build . -t test-docker-image`:

```
PS D:\Learning\Revou\BE\test-docker> docker build . -t test-docker-image
[+] Building 1.1s (8/8) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 295B
=> [internal] load metadata for docker.io/library/node:latest
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/3] FROM docker.io/library/node:latest@sha256:b9ccc4aca32eebf124e0ca0fd573dacffba2b9236987a1d4d2625ce3c162ecc8
=> [internal] load build context
=> => transferring context: 58B
=> CACHED [2/3] WORKDIR /app
=> CACHED [3/3] COPY . .
=> exporting to image
=> => exporting layers
=> => writing image sha256:6cfec1d0336ff759e5dda88e57c9b01d338d4dd7a41b0cee0444a59b68f8fb55
=> => naming to docker.io/library/test-docker-image

View build details: docker-desktop://dashboard/build/default/default/pd7x8wecgtyd9o6ddao6svn9j

What's Next?
View a summary of image vulnerabilities and recommendations -> docker scout quickview
```

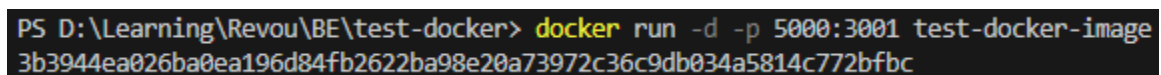
3. Confirm if the image was created successfully using "docker images"



```
PS D:\Learning\Revou\BE\test-docker> docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
test-docker-image	latest	6cfec1d0336f	3 minutes ago	1.1GB

4. Create and run a docker container using "docker run -d -p 5000:3001 test-docker-image". Need to specify the port to be same with the EXPOSE in Dockerfile



```
PS D:\Learning\Revou\BE\test-docker> docker run -d -p 5000:3001 test-docker-image
3b3944ea026ba0ea196d84fb2622ba98e20a73972c36c9db034a5814c772bfbc
```

5. Confirm if the container was created successfully using "docker ps"

```
PS D:\Learning\Revou\BE\test-docker> docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
3b3944ea026b	test-docker-image	"docker-entrypoint.s..."	6 seconds ago	Up 6 seconds	5000/tcp, 0.0.0.0:5000->3001/tcp	mystifying_jackson

6. App is working

