

Program 1: Build a docker container from a custom dockerfile

=> In the terminal,

- Mkdir first
- Cd first
- Code .

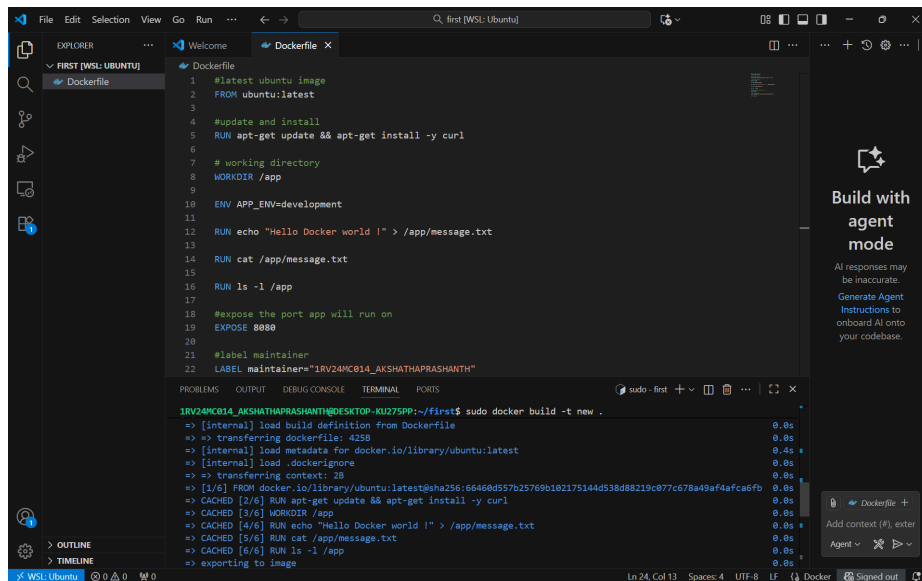
In VS code, create a Dockerfile

- FROM ubuntu:latest
- RUN apt-get update && apt-get install -y curl
- WORKDIR /app
- ENV APP_ENV=development
- RUN echo "hello docker world: > /app/message.txt
- RUN cat /app/message.txt
- COPY ./app
- ENV APP_ENV=development
- EXPOSE 8080
- CMD ["bash"]
- Label maintainer="IRV24MC014_AKSHATHAPRASHANTH"

RUN:

docker build -t new .

docker images



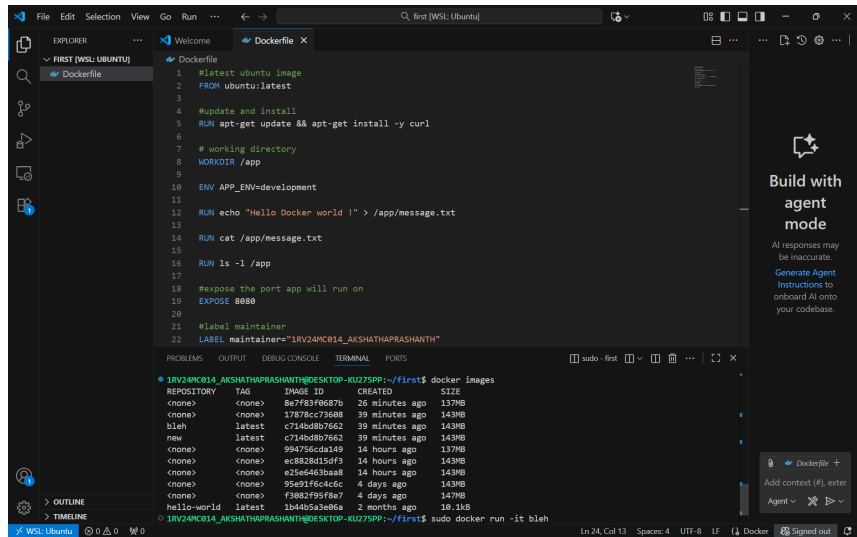
The screenshot shows the Visual Studio Code interface with a Dockerfile open in the editor and its build output in the terminal. The Dockerfile contains the following instructions:

```
1 #latest ubuntu image
2 FROM ubuntu:latest
3
4 #update and install
5 RUN apt-get update && apt-get install -y curl
6
7 # working directory
8 WORKDIR /app
9
10 ENV APP_ENV=development
11
12 RUN echo "Hello Docker world !" > /app/message.txt
13
14 RUN cat /app/message.txt
15
16 RUN ls -l /app
17
18 #expose the port app will run on
19 EXPOSE 8080
20
21 #label maintainer
22 LABEL maintainer="IRV24MC014_AKSHATHAPRASHANTH"
```

The terminal output shows the successful execution of the `docker build -t new .` command, with the following steps and durations:

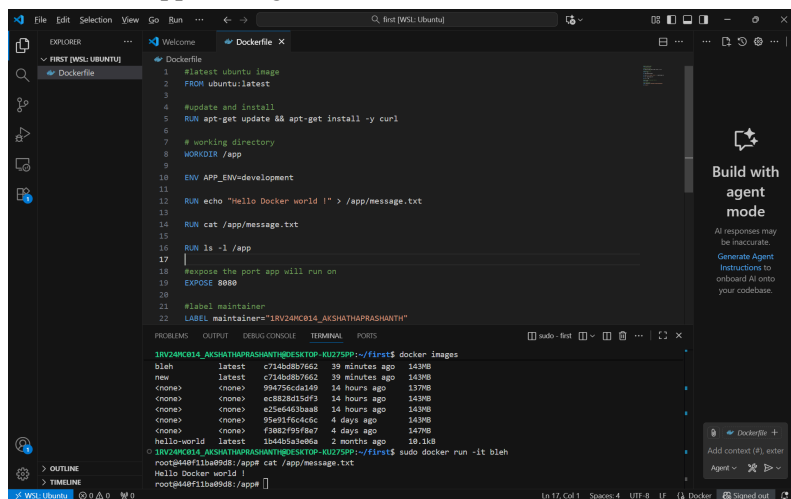
```
IRV24MC014_AKSHATHAPRASHANTH@DESKTOP-KU275PP:~/first$ sudo docker build -t new .
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 425B
=> [internal] load metadata for docker.io/library/ubuntu:latest
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/6] FROM docker.io/library/ubuntu:latest@sha256:66409d3576257696102175144d538d88219c077c678a49af4efca6fb
=> CACHED [2/6] RUN apt-get update && apt-get install -y curl
=> CACHED [3/6] WORKDIR /app
=> CACHED [4/6] RUN echo "Hello Docker world !" > /app/message.txt
=> CACHED [5/6] RUN cat /app/message.txt
=> CACHED [6/6] RUN ls -l /app
=> exporting to image
```

The terminal output also shows the build progress bar and the final image name `new`.



sudo docker run -it bleh

In root: cat /app/message.txt



Program 2: Develop a multistage Dockerfile for container orchestration

=> In the terminal,

- Mkdir second
- cd second
- code .

In VS code, create a folder src, in that create

[index.js](#)

```
const express= require('express');
```

```
const app = express();
```

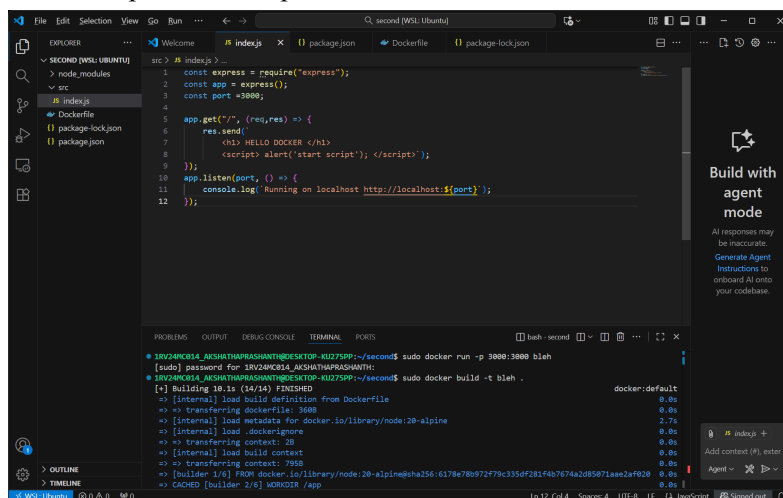
```
const port =3000;
```

```
app.get('/', (req,res) => {  
  res.send(  
    <script> alert ('start script') ; </script>  
    <h1> Hello Docker </h1> `);  
});
```

```
app.listen(port, () => {  
  console.log(`server running on localhost http://localhost:\${port}`);  
});
```

In the terminal, install npm, it will also create the package.json file

- \$npm init -y
- \$npm install express



In the package.json file, under “scripts”, add

```
“scripts” : {
```

```
  “build”: “echo ‘HI’ && mkdir -p dist && cp src/index.js dist”,
```

```

"start": "node dist/index.js",
}

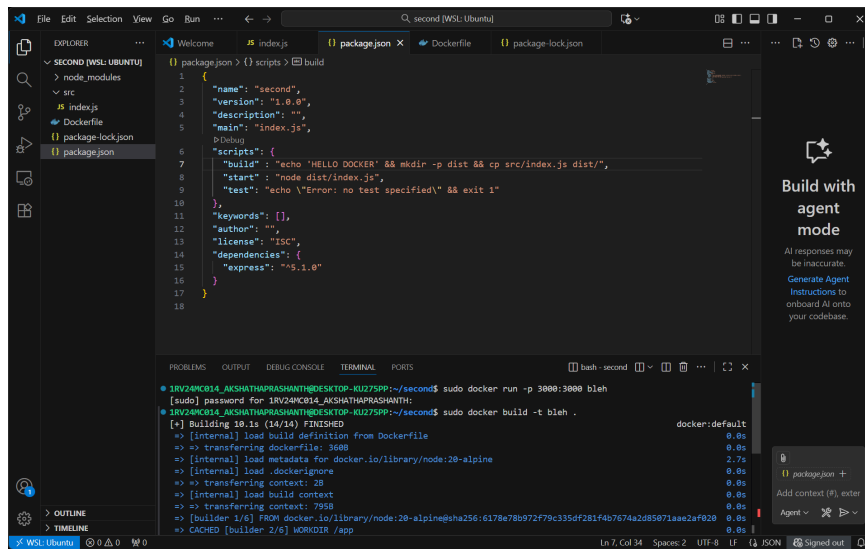
```

Create a Dockerfile

- ```
- FROM node:20-alpine AS builder
- WORKDIR /app
- COPY package.json ./
- RUN npm install
- COPY src ./src
- RUN npm run build
#stage2
- FROM node:20-alpine
- WORKDIR /app
- COPY --from=builder /app/package.json ./
- COPY --from=builder /app/node_modules ./node_modules
- COPY --from=builder /app/dist ./dist
- EXPOSE 3000
- CMD ["npm", "start"]
```

In the terminal, run

- Sudo docker run -p 3000:3000 bleh
- sudo docker build -t bleh



Next, run `node src/index.js`

The screenshot shows the VS Code editor interface with a file explorer on the left. The active file is `package.json`, which contains the following content:

```
{
 "name": "second",
 "version": "1.0.0",
 "description": "",
 "main": "index.js",
 "scripts": {
 "build": "echo 'HELLO DOCKER' && mkdir -p dist && cp src/index.js dist/",
 "start": "node dist/index.js",
 "test": "echo 'Error: no test specified' && exit 1"
 },
 "keywords": [],
 "author": "",
 "license": "ISC",
 "dependencies": {
 "express": "^5.1.0"
 }
}
```

The terminal window at the bottom shows the following commands and output:

```
IRV24K014_AKSHATHAPRASHANTH@DESKTOP-KU275PP:~/second$ sudo docker build -t bleh .
=> [stage-1 4/5] COPY --from=builder /app/node_modules ./node_modules 0.1s
=> [stage-1 5/5] COPY --from=builder /app/dist ./dist 0.1s
=> exporting to image 0.2s
=> >> exporting layers 0.1s
=> writing image sha256:b0f89377f5db233da7297b84b1ac8c0bdc0cd0b25d975beab4994c5f663cf1f9 0.0s
=> naming to docker.io/library/bleh
IRV24K014_AKSHATHAPRASHANTH@DESKTOP-KU275PP:~/second$ src/index.js
Running on localhost http://localhost:3000
^C
IRV24K014_AKSHATHAPRASHANTH@DESKTOP-KU275PP:~/second$ node src/index.js
Running on localhost http://localhost:3000
```

A notification bubble in the bottom right corner states: "Your application running on port 3000 is available. See all forwarded ports" with buttons for "Open in Browser" and "Preview in Editor".

