

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 VS Code interface with the Dockerfile open in the editor. The terminal at the bottom shows the command being run and its output.

```
#latest ubuntu image
FROM ubuntu:latest
#update and install
RUN apt-get update && apt-get install -y curl
# working directory
WORKDIR /app
ENV APP_ENV=development
RUN echo "Hello Docker world!" > /app/message.txt
RUN cat /app/message.txt
RUN ls -1 /app
#expose the port app will run on
EXPOSE 8080
#label maintainer
LABEL maintainer="IRV24MC014_AKSHATHAPRASHANTH"
```

```
IRV24MC014_AKSHATHAPRASHANTH@DESKTOP-KU275PP:~/first$ sudo docker build -t new .
=> [internal] load build definition from Dockerfile
=> [internal] transfering Dockerfile: 425B
=> [internal] load metadata for docker.io/library/ubuntu:latest
=> [internal] load dockerignore
=> [internal] transfering context: 2B
=> [1/6] FROM docker.io/library/ubuntu:latest@sha256:66460d557b25769b102175144d538d88219c077c678a49af4afca5fb
=> [2/6] RUN apt-get update && apt-get install -y curl
=> [3/6] WORKDIR /app
=> [4/6] RUN echo "Hello Docker world!" > /app/message.txt
=> [5/6] RUN cat /app/message.txt
=> [6/6] RUN ls -1 /app
=> exporting to image
```

```
#!/bin/bash
# Latest ubuntu image
FROM ubuntu:latest
# Update and install
RUN apt-get update && apt-get install -y curl
# Working directory
WORKDIR /app
# Environment variable
ENV APP_ENV=development
RUN echo "Hello Docker world!" > ./app/message.txt
RUN cat ./app/message.txt
RUN ls -l /app
# Expose the port app will run on
EXPOSE 8080
# Label maintainer
LABEL maintainers="IRV24MCB14_AKSHATHAPRASHANTH"
```

Build with agent mode

AI responses may be inaccurate. Generate Agent Instructions to onboard AI onto your codebase.

```
18/24MCB14_AKSHATHAPRASHANTH@DESKTOP-KUZ75PP:~/first$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
<none> <none> b6d9a9057b 8 days ago 143MB
<none> <none> c7878cc73608 39 minutes ago 143MB
bleh latest c714bd8b7662 39 minutes ago 143MB
new latest c714bd8b7662 39 minutes ago 143MB
<none> <none> 994756cd1a49 14 hours ago 137MB
<none> <none> e885915df3 14 hours ago 143MB
<none> <none> e885915df3 14 hours ago 143MB
<none> <none> 95e931fc946c 4 days ago 143MB
<none> <none> f3882f95f8e7 4 days ago 147MB
hello-world latest 1b4405a3e68a 2 months ago 10.1kB

18/24MCB14_AKSHATHAPRASHANTH@DESKTOP-KUZ75PP:~/first$ sudo docker run -it bleh
```

sudo docker run -it bleh

In root: cat /app/message.txt

```
#!/bin/bash
# Latest ubuntu image
FROM ubuntu:latest
# Update and install
RUN apt-get update && apt-get install -y curl
# Working directory
WORKDIR /app
# Environment variable
ENV APP_ENV=development
RUN echo "Hello Docker world!" > ./app/message.txt
RUN cat ./app/message.txt
RUN ls -l /app
# Expose the port app will run on
EXPOSE 8080
# Label maintainer
LABEL maintainers="IRV24MCB14_AKSHATHAPRASHANTH"
```

Build with agent mode

AI responses may be inaccurate. Generate Agent Instructions to onboard AI onto your codebase.

```
18/24MCB14_AKSHATHAPRASHANTH@DESKTOP-KUZ75PP:~/first$ docker images
bleh latest c714bd8b7662 39 minutes ago 143MB
new latest c714bd8b7662 39 minutes ago 143MB
<none> <none> 994756cd1a49 14 hours ago 137MB
<none> <none> e88282d15df3 14 hours ago 143MB
<none> <none> e88282d15df3 14 hours ago 143MB
<none> <none> 95e931fc946c 4 days ago 143MB
<none> <none> f3882f95f8e7 4 days ago 147MB
hello-world latest 1b4405a3e68a 2 months ago 10.1kB

18/24MCB14_AKSHATHAPRASHANTH@DESKTOP-KUZ75PP:~/first$ sudo docker run -it bleh
root@4405a3e68a:~/app# cat ./app/message.txt
Hello Docker world!
```

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](#)

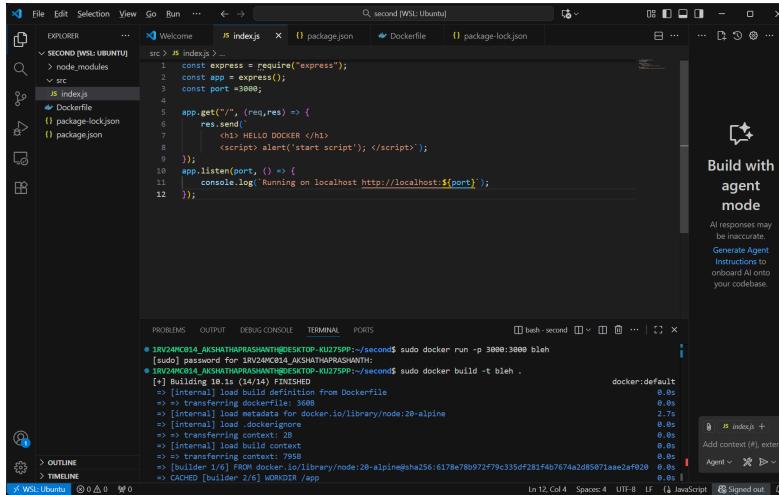
```
const express = require('express');
const app = express();
const port = 3000;
```

```
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

The screenshot shows the VS Code interface with the following details:

- Explorer View:** Shows a project structure for a folder named "SECOND [WSL: UBUNTU]" containing "node_modules", "src", "index.js", "Dockerfile", and "package-lock.json".
- Code Editor:** Displays a Dockerfile with the following content:

```
1 FROM node:20-alpine AS builder
2 WORKDIR /app
3 COPY package.json .
4 RUN npm install
5 COPY src ./src
6 RUN npm run build
7 #stage2
8 FROM node:20-alpine
9 WORKDIR /app
10 COPY --from=builder /app/package.json .
11 COPY --from=builder /app/node_modules ./node_modules
12 COPY --from=builder /app/dist ./dist
13 EXPOSE 3000
14 CMD ["npm", "start"]
```
- Terminal:** Shows the command history and output of the terminal session:

```
IRV24MCB14_AKSHATHAPRASHANTI@DESKTOP-KU27ZPP:~/second$ sudo docker run -p 3000:3000 bleh
[sudo] password for IRV24MCB14_AKSHATHAPRASHANTI:
IRV24MCB14_AKSHATHAPRASHANTI@DESKTOP-KU27ZPP:~/second$ sudo docker build -t bleh .
[4] Building docker://bleh
[4] 1/1 [internal] load build definition from Dockerfile
=> transferring dockerfile: 340B
[4] 1/1 [internal] load metadata for docker.io/library/node:20-alpine
=> [internal] load .dockerrigignore
=> [internal] transfer context: 2B
=> [internal] load build context
=> transferring context: 795B
=> [builder 1/6] FROM docker.io/library/node:20-alpine@sha256:6178e78b972f79c335df281f4b7674a2d85071aae2af029 0.0s
=> CACHED [builder 2/6] WORKDIR /app
0.0s
```
- Bottom Status Bar:** Shows file counts (0), workspace status (0), and other system information.

Next, run node src/index.js

The screenshot shows the Visual Studio Code interface running in a Windows Subsystem for Linux (WSL) Ubuntu environment. The Explorer sidebar on the left lists files in a folder named 'SECOND [WSL: UBUNTU]'. The current file is 'Dockerfile', which contains the following code:

```
1  FROM node:10-alpine
2  WORKDIR /app
3  COPY . .
4  RUN npm install
5  EXPOSE 3000
6  CMD ["node", "index.js"]
```

The terminal tab at the bottom shows the command `sudo docker build -t bleh .` being run, followed by the output of the build process:

```
1  [stage-1 4/5] COPY --from=builder /app/node_modules ./node_modules
2  [stage-1 5/5] COPY --from=builder /app/dist ./dist
3  [stage-1] 0.1s
4  [stage-1] 0.1s
5  [stage-1] 0.1s
6  [stage-1] 0.1s
7  [stage-1] 0.1s
8  [stage-1] 0.1s
9  [stage-1] 0.1s
10 [stage-1] 0.1s
11 [stage-1] 0.1s
12 [stage-1] 0.1s
13 [stage-1] 0.1s
14 [stage-1] 0.1s
15 [stage-1] 0.1s
16 [stage-1] 0.1s
17 [stage-1] 0.1s
```

The status bar at the bottom indicates the file is in 'Edit' mode, has 34 lines, 2 spaces, and is in UTF-8 encoding.

