In VS Code, your folder should look like this:

```
folder_name/
|── server.js
|── package. Json
|── Dockerfile
```

Open **`VS Code → Open Folder → folder_name`**

Step 1: Create `package.json` (if not already created)

In the VS Code terminal:

```
npm init -y
```

It will open `package.json` by default in the terminal

Now install Express:

```
npm install express
```

This creates:

- `package.json`
- `package-lock.json`
- `node_modules/`

check `package.json` again to confirm installation

```
cat package.json
```

Step 2: Your `server.js`

General format

```
//define ineraction mode
'use strict';//please be careful and strict.

// 2. Load a library
const someLibrary = require('some-library');

// 3. Create the server
const app = someLibrary();

// 4. Define server behavior
app.whenSomethingHappens((request, response) => {
  response.sendSomething();
});

// 5. Start the server
app.listen(PORT_NUMBER, () => {
  console.log('Server started');
});
```

Our `server.js`

```
'use strict';
// Tell JavaScript to be strict and catch mistakes early

const PORT = 8080;
const express = require('express');
// Load the Express library so we can create a server

const app = express();
// Create the server and store it in the variable "app"

app.get('/', (req, res) => {
// When a user visits the homepage "/"
  res.send('hello world');
  // Send text back to the browser
});
// Finished defining what happens for "/"

app.listen(PORT, () => {
   // Start the server and listen on port 8080
  console.log('Server listening on port 8080');
  // Print a message in the terminal to show the server is running
});
// Server setup is complete
```

Step 3: Create the Dockerfile

In VS Code:

- Right-click → **New File**
- Name it exactly: **Dockerfile**
  (⚠️ no `.txt`, no extension)

Learn 4 core building blocks.

Block 1: Base Image (FROM)

Every Dockerfile starts with:

```
FROM something
```

This answers:

"What already exists?"

Examples:

```
FROM python:3.10
FROM node:18
FROM ubuntu:22.04
```

Think like

"I am standing on someone else's shoulders."

Block 2: Copy Code (COPY)

```
COPY source destination
```

Examples:

```
COPY . /app
COPY hello.py /app
```

Think like

"Put my files inside the container."

Block 3: Install Things (RUN)

`RUN` some-command

Examples:

```
RUN pip install Flask
RUN npm install
RUN apt update && apt install -y curl
```

Think like

“Set up the environment.”

Block 4: Start Command (CMD or command:)

Two ways to run

Way 1: Start automatically. When the docker run, it will start the app

```
CMD ["python", "hello.py"]
```

Way 2: For interactive mode. Use the terminal to start the app

```
CMD ["/bin/bash"]
```

```
FROM node:latest
# Create app directory
WORKDIR /app
# Install app dependencies
# Copy app files
COPY *.json ./
COPY server.js .
RUN npm install
#Expose port
EXPOSE 8080
#Start the app
#CMD [ "node", "server.js" ]
#Use terminal to start app
CMD ["/bin/bash"]
```

Step 4: Create docker-compose.yml

In VS Code:

- Right-click → **New File**
- Name it exactly: **docker-compose.yml**

```
version: "3.9"
services:
  web:
    build: .
    container_name: node1
    ports:
      - "8080:8080"
    volumes:
      - ./:/app
    stdin_open: true
    tty: true
```

Compose line	Manual Docker equivalent
`build: .`	`docker build -t node1 .`
`container_name: node1`	`docker run --name node1 node1`
`ports: "8080:8080"`	`docker run -p 8080:8080`
`volumes: ./:/app`	`docker run -v $(pwd):/app`
`stdin_open: true`	`docker run -i`
`tty: true`	`docker run -t`

Step 5: Run (Compose way)

From VS Code terminal:

`docker compose up`

You should see:

Server listening on port 8080

Step 6: Test in Browser

http://localhost