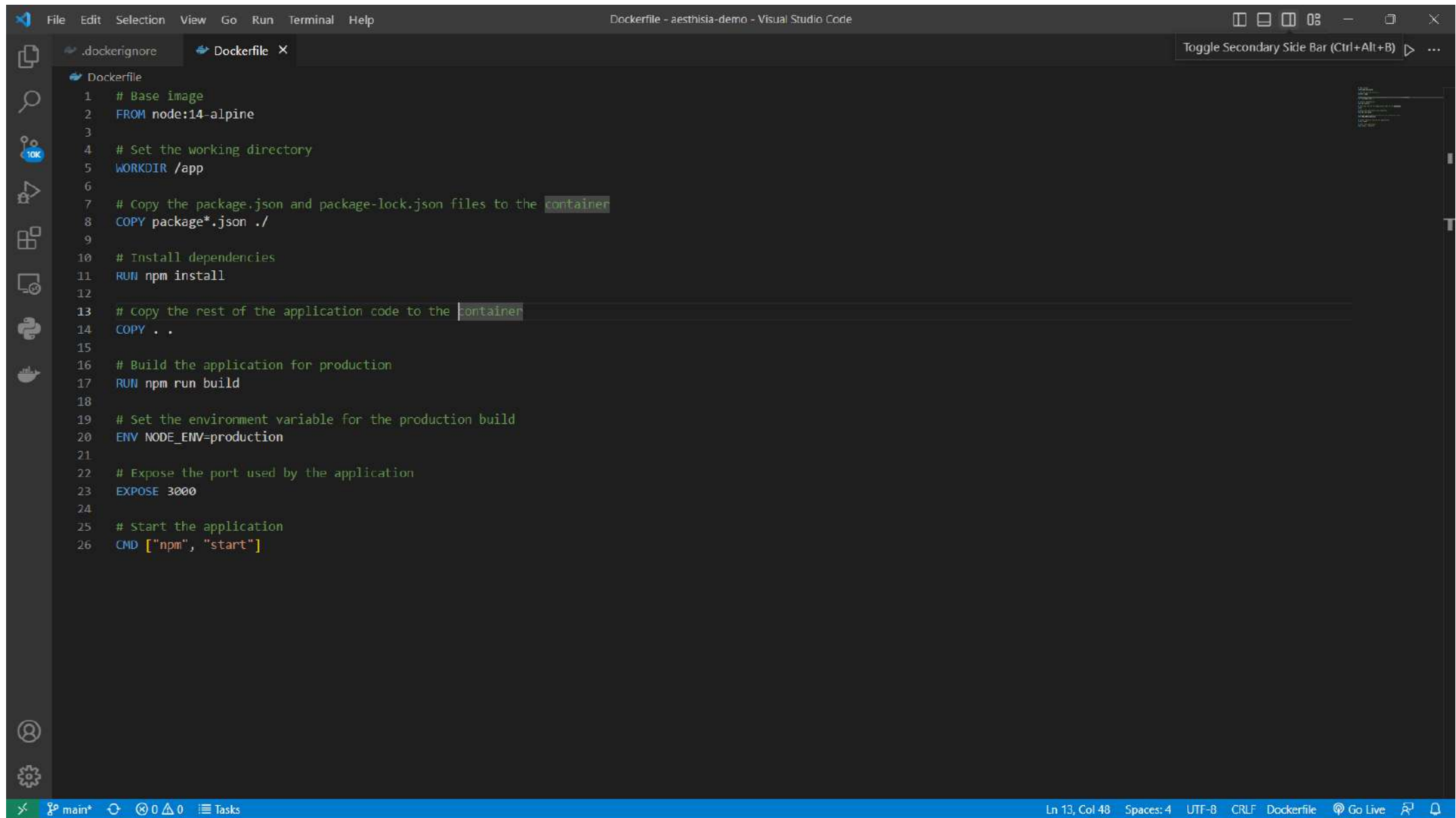


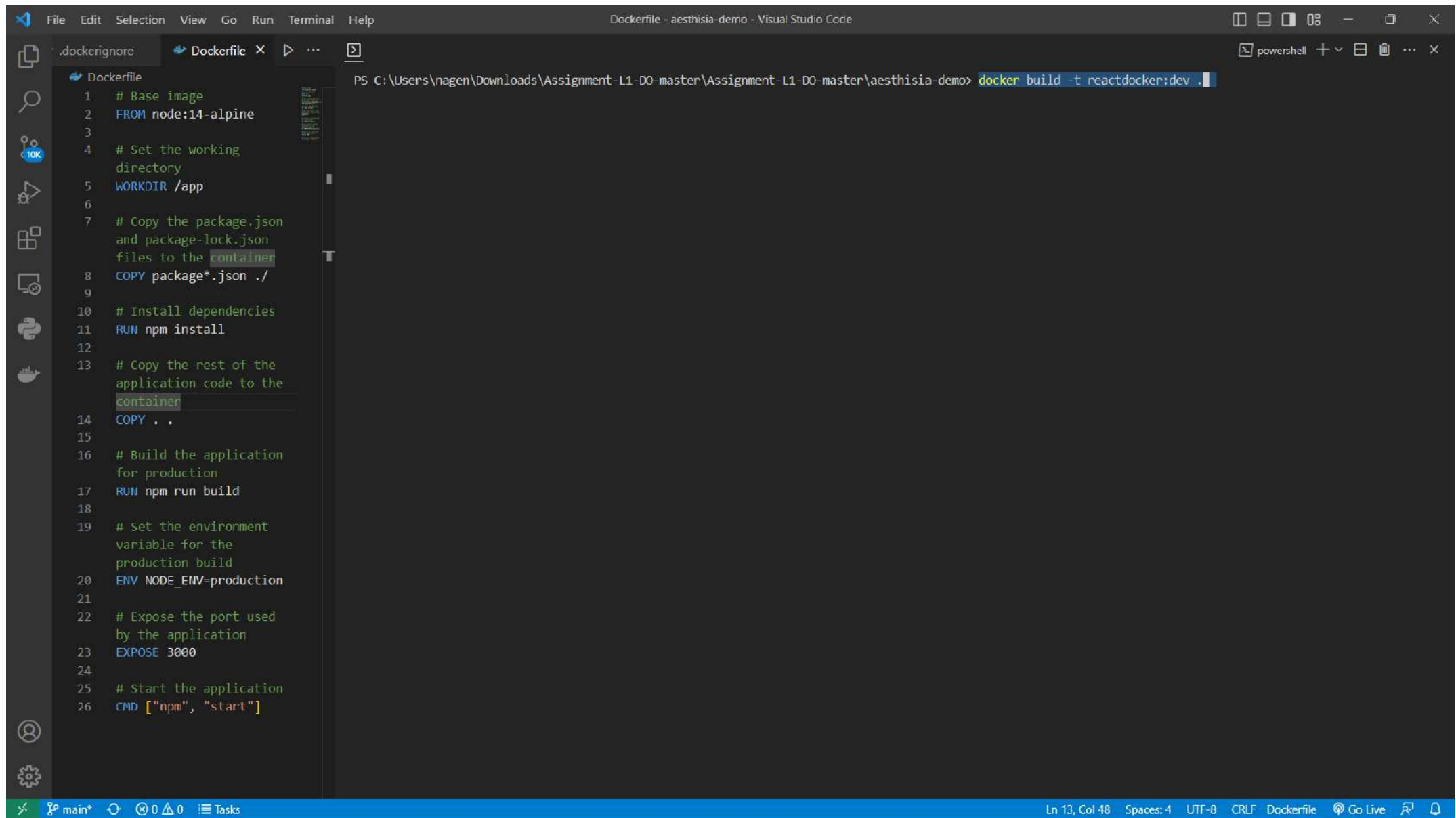
My Docker File



```
File Edit Selection View Go Run Terminal Help
Dockerfile - aesthisia-demo - Visual Studio Code
Toggle Secondary Side Bar (Ctrl+Alt+B)
Dockerfile
1 # Base image
2 FROM node:14-alpine
3
4 # Set the working directory
5 WORKDIR /app
6
7 # Copy the package.json and package-lock.json files to the container
8 COPY package*.json ./
9
10 # Install dependencies
11 RUN npm install
12
13 # Copy the rest of the application code to the container
14 COPY . .
15
16 # Build the application for production
17 RUN npm run build
18
19 # Set the environment variable for the production build
20 ENV NODE_ENV=production
21
22 # Expose the port used by the application
23 EXPOSE 3000
24
25 # Start the application
26 CMD ["npm", "start"]
```

main 0 0 Tasks Ln 13, Col 48 Spaces: 4 UTF-8 CRLF Dockerfile Go Live

Build the Image Cmd: docker build -t "container name":"tag" .



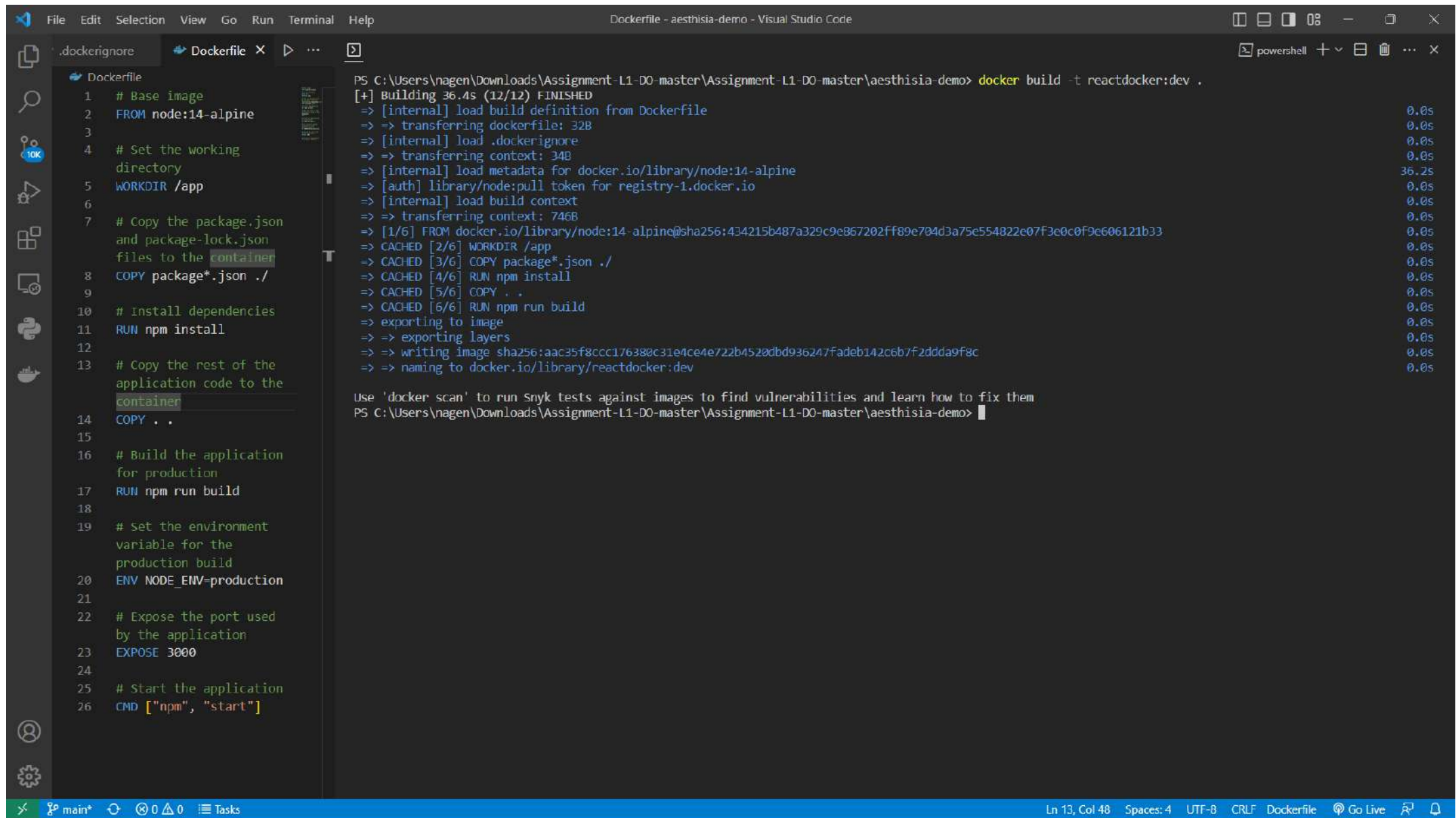
The screenshot shows the Visual Studio Code interface with a Dockerfile open in the editor and a terminal window at the bottom. The Dockerfile contains the following instructions:

```
1 # Base image
2 FROM node:14-alpine
3
4 # Set the working
  directory
5 WORKDIR /app
6
7 # Copy the package.json
  and package-lock.json
  files to the container
8 COPY package*.json ./
9
10 # Install dependencies
11 RUN npm install
12
13 # Copy the rest of the
  application code to the
  container
14 COPY . .
15
16 # Build the application
  for production
17 RUN npm run build
18
19 # Set the environment
  variable for the
  production build
20 ENV NODE_ENV=production
21
22 # Expose the port used
  by the application
23 EXPOSE 3000
24
25 # Start the application
26 CMD ["npm", "start"]
```

The terminal window shows the command `docker build -t reactdocker:dev .` being executed in a PowerShell session at the path `C:\Users\nagen\Downloads\Assignment-L1-D0-master\Assignment-L1-D0-master\ aesthisia-demo`.

Ln 13, Col 48 Spaces: 4 UTF-8 CRLF Dockerfile Go Live

Image Created successfully



The image shows a Visual Studio Code editor window with a Dockerfile open on the left and its build output in a terminal on the right. The Dockerfile contains instructions for building a Docker image, including setting the base image, working directory, copying files, installing dependencies, and exposing a port. The terminal shows the successful execution of the `docker build` command, with a progress bar indicating the build is complete.

```
.dockerignore
Dockerfile
Dockerfile
1 # Base image
2 FROM node:14-alpine
3
4 # Set the working
  directory
5 WORKDIR /app
6
7 # Copy the package.json
  and package-lock.json
  files to the container
8 COPY package*.json ./
9
10 # Install dependencies
11 RUN npm install
12
13 # Copy the rest of the
  application code to the
  container
14 COPY . .
15
16 # Build the application
  for production
17 RUN npm run build
18
19 # Set the environment
  variable for the
  production build
20 ENV NODE_ENV=production
21
22 # Expose the port used
  by the application
23 EXPOSE 3000
24
25 # Start the application
26 CMD ["npm", "start"]
```

```
PS C:\Users\nagen\Downloads\Assignment-L1-DO-master\Assignment-L1-DO-master\ aesthisia-demo> docker build -t reactdocker:dev .
[+] Building 36.4s (12/12) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 32B
=> [internal] load .dockerignore
=> => transferring context: 34B
=> [internal] load metadata for docker.io/library/node:14-alpine
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load build context
=> => transferring context: 746B
=> [1/6] FROM docker.io/library/node:14-alpine@sha256:434215b487a329c9e867202ff89e704d3a75e554822e07f3e0c0f9e606121b33
=> CACHED [2/6] WORKDIR /app
=> CACHED [3/6] COPY package*.json ./
=> CACHED [4/6] RUN npm install
=> CACHED [5/6] COPY . .
=> CACHED [6/6] RUN npm run build
=> exporting to image
=> => exporting layers
=> => writing image sha256:aac35f8ccc176380c31e4ce4e722b4520dbd936247fadeb142c6b7f2ddda9f8c
=> => naming to docker.io/library/reactdocker:dev

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
PS C:\Users\nagen\Downloads\Assignment-L1-DO-master\Assignment-L1-DO-master\ aesthisia-demo>
```

Ln 13, Col 48 Spaces: 4 UTF-8 CRLF Dockerfile Go Live

Check whether the image is created or not in Docker desktop

Docker Desktop

Update to latest

Search

Ctrl+K

chvna...

Containers

Images

Volumes

Dev Environments BETA

Extensions

Add Extensions

Images [Give feedback](#)

An image is a read-only template with instructions for creating a Docker container. [Learn more](#)

Local

Hub

0 Bytes / 206.45 MB in use 3 images

Last refresh: about 3 hours ago

Search

<input type="checkbox"/>	Name	Tag	Status	Created	Size	Actions
<input type="checkbox"/>	reactdocker aac35f8ccc17	dev	Unused	about 2 hours ago	306.31 MB	
<input type="checkbox"/>	nginx 3f8a00f137a0	latest	Unused	3 months ago	141.83 MB	
<input type="checkbox"/>	httpd 3a4ea134cf8e	latest	Unused	3 months ago	145.12 MB	

Showing 3 items

RAM 1.85 GB CPU 0.11% Connected to Hub

v4.16.3

The status of the image is not in use

Docker Desktop

Update to latest

Search

Ctrl+K

chvna...

Containers

Images

Volumes

Dev Environments BETA

Extensions

Add Extensions

< reactdocker:dev

IMAGE ID
aac35f8ecc17

CREATED
about 2 hours ago

SIZE
306.31 MB

More actions

Run

We're trialing a new image details page which provides more information about the image and gives insight into packages and vulnerabilities. You can turn it off in your [settings](#).

Image hierarchy

FROM alpine:3, 3.17, 3.17.3, latest

FROM node:14-alpine, 14-alpine3.17, 14.21-alpine, 14.21-alpine...

ALL reactdocker:dev

Layers (17)

0

ADD file:9a4f77dfaba7fd2aa78186e4ef0e7486ad5...

7.05 MB

1

CMD ["/bin/sh"]

0 B

2

ENV NODE_VERSION=14.21.3

0 B

3

addgroup -g 1000 node && adduser -u 1000 -G nod...

104 MB

4

ENV YARN_VERSION=1.22.19

0 B

5

apk add --no-cache --virtual .build-deps-yarn curl g...

7.85 MB

6

COPY file:4d192565a7220e135cab6c77fbc1c732...

388 B

7

ENTRYPOINT ["docker-entrypoint.sh"]

0 B

8

CMD ["node"]

0 B

9

WORKDIR /app

0 B

Images (3)

Vulnerabilities (21)

Packages (1534)

[Give feedback](#)

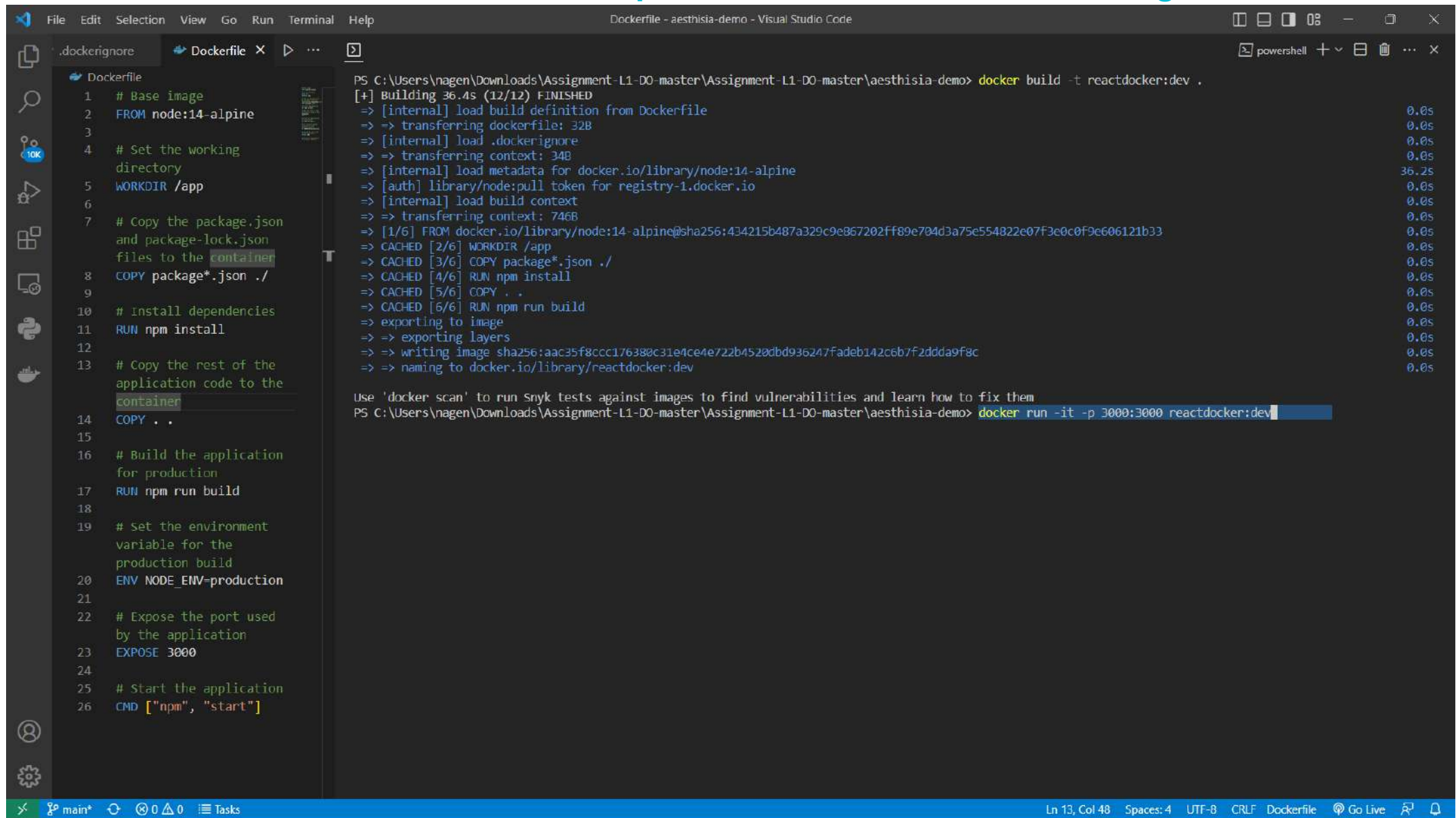
Package	Vulnerabilities
> loader-utils 2.0.2	1 C 2 H 0 M 0 L
> execa 0.7.0	1 C 0 H 0 M 0 L
> webpack 5.73.0	1 C 0 H 0 M 0 L
> loader-utils 3.2.0	2 H 0 M 0 L
> decode-uri-component 0.2.0	1 H 1 M 0 L
> ansi-regex 3.0.0	1 H 0 M 0 L
> ansi-regex 4.1.0	1 H 0 M 0 L
> http-cache-semantics 3.8.1	1 H 0 M 0 L
> json5 1.0.1	1 H 0 M 0 L
> json5 2.2.1	1 H 0 M 0 L
> minimatch 3.0.4	1 H 0 M 0 L
> nth-check 1.0.2	1 H 0 M 0 L
> qs 6.5.3	1 H 0 M 0 L

RAM 1.85 GB CPU 0.11% Connected to Hub

v4.16.3

Run the image and assign the port number:3000

cmd: docker run -it -p 3000:3000 "container name":"tag"



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

Dockerfile:

```
1 # Base image
2 FROM node:14-alpine
3
4 # Set the working
  directory
5 WORKDIR /app
6
7 # Copy the package.json
  and package-lock.json
  files to the container
8 COPY package*.json ./
9
10 # Install dependencies
11 RUN npm install
12
13 # Copy the rest of the
  application code to the
  container
14 COPY . .
15
16 # Build the application
  for production
17 RUN npm run build
18
19 # Set the environment
  variable for the
  production build
20 ENV NODE_ENV=production
21
22 # Expose the port used
  by the application
23 EXPOSE 3000
24
25 # Start the application
26 CMD ["npm", "start"]
```

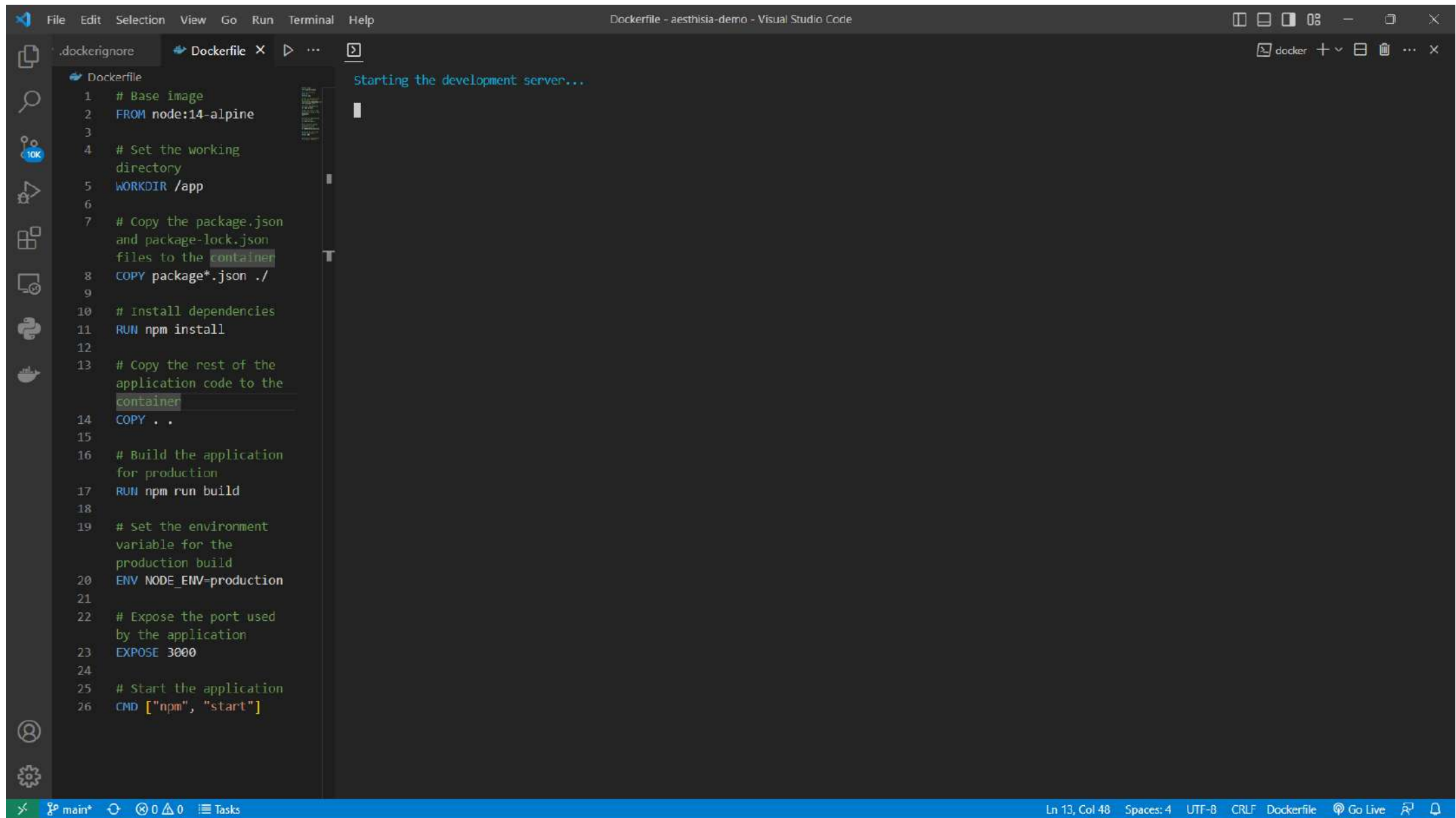
Terminal Output:

```
PS C:\Users\nagen\Downloads\Assignment-L1-DO-master\Assignment-L1-DO-master\ aesthisia-demo> docker build -t reactdocker:dev .
[+] Building 36.4s (12/12) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 32B
=> [internal] load .dockerignore
=> => transferring context: 34B
=> [internal] load metadata for docker.io/library/node:14-alpine
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load build context
=> => transferring context: 746B
=> [1/6] FROM docker.io/library/node:14-alpine@sha256:434215b487a329c9e867202ff89e704d3a75e554822e07f3e0c0f9e606121b33
=> CACHED [2/6] WORKDIR /app
=> CACHED [3/6] COPY package*.json ./
=> CACHED [4/6] RUN npm install
=> CACHED [5/6] COPY . .
=> CACHED [6/6] RUN npm run build
=> exporting to image
=> => exporting layers
=> => writing image sha256:aac35f8ccc176380c31e4ce4e722b4520dbd936247fadeb142c6b7f2ddda9f8c
=> => naming to docker.io/library/reactdocker:dev

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
PS C:\Users\nagen\Downloads\Assignment-L1-DO-master\Assignment-L1-DO-master\ aesthisia-demo> docker run -it -p 3000:3000 reactdocker:dev
```

The terminal shows the successful build of the Docker image 'reactdocker:dev' and the start of the container. The status bar at the bottom indicates the current file is 'main' and the cursor is at line 13, column 48.

Starting the development Server



The screenshot shows the Visual Studio Code interface with a Dockerfile open in the editor. The Dockerfile contains instructions for building a Node.js application container. A terminal window is open on the right, showing the command to start the development server.

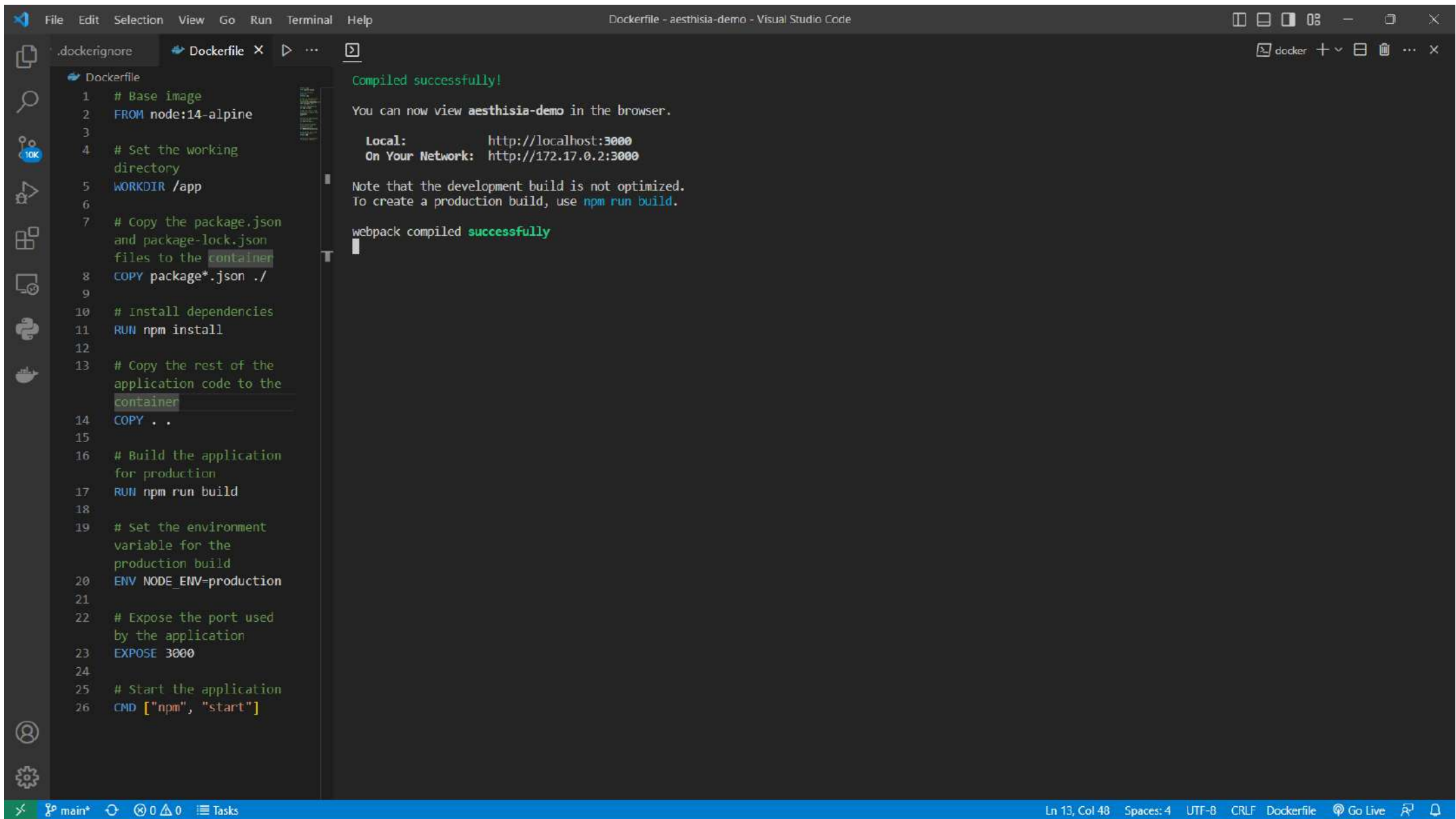
```
File Edit Selection View Go Run Terminal Help
Dockerfile - aesthisia-demo - Visual Studio Code

Dockerfile
1 # Base image
2 FROM node:14-alpine
3
4 # Set the working
  directory
5 WORKDIR /app
6
7 # Copy the package.json
  and package-lock.json
  files to the container
8 COPY package*.json ./
9
10 # Install dependencies
11 RUN npm install
12
13 # Copy the rest of the
  application code to the
  container
14 COPY . .
15
16 # Build the application
  for production
17 RUN npm run build
18
19 # Set the environment
  variable for the
  production build
20 ENV NODE_ENV=production
21
22 # Expose the port used
  by the application
23 EXPOSE 3000
24
25 # Start the application
26 CMD ["npm", "start"]

Starting the development server...
```

Ln 13, Col 48 Spaces: 4 UTF-8 CRLF Dockerfile Go Live

Compiled successfully



The screenshot shows the Visual Studio Code interface with a Dockerfile open in the editor. The Dockerfile contains instructions for building a Node.js application. The output pane on the right shows the successful compilation of the application.

```
File Edit Selection View Go Run Terminal Help
Dockerfile - aesthisia-demo - Visual Studio Code

Dockerfile
1 # Base image
2 FROM node:14-alpine
3
4 # Set the working
  directory
5 WORKDIR /app
6
7 # Copy the package.json
  and package-lock.json
  files to the container
8 COPY package*.json ./
9
10 # Install dependencies
11 RUN npm install
12
13 # Copy the rest of the
  application code to the
  container
14 COPY . .
15
16 # Build the application
  for production
17 RUN npm run build
18
19 # Set the environment
  variable for the
  production build
20 ENV NODE_ENV=production
21
22 # Expose the port used
  by the application
23 EXPOSE 3000
24
25 # Start the application
26 CMD ["npm", "start"]

Compiled successfully!
You can now view aesthisia-demo in the browser.

Local:      http://localhost:3000
On Your Network: http://172.17.0.2:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

Ln 13, Col 48 Spaces: 4 UTF-8 CRLF Dockerfile Go Live

Now the status of the image is in use

Docker Desktop

Update to latest

Search

Ctrl+K

chvna...

Containers

Images

Volumes

Dev Environments BETA

Extensions

Add Extensions

Images

[Give feedback](#)

An image is a read-only template with instructions for creating a Docker container. [Learn more](#)

Local

Hub

306.31 MB / 206.45 MB in use

3 images

Last refresh: about 3 hours ago

Search

	Name	Tag	Status	Created	Size	Actions
<input type="checkbox"/>	reactdocker aac35f8ccc17	dev	In use	about 2 hours ago	306.31 MB	<div></div> <div></div> <div></div>
<input type="checkbox"/>	nginx 3f8a00f137a0	latest	Unused	3 months ago	141.83 MB	<div></div> <div></div> <div></div>
<input type="checkbox"/>	httpd 3a4ea134cf8e	latest	Unused	3 months ago	145.12 MB	<div></div> <div></div> <div></div>

Showing 3 items

RAM 2.37 GB CPU 0.22% Connected to Hub

v4.16.3

Container is running on the port number:3000

Docker Desktop

Update to latest

Search

Ctrl+K

chvna...

Containers

Images

Volumes

Dev Environments BETA

Extensions

Add Extensions

Containers

Give feedback

A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. [Learn more](#)

Only show running containers

Search: aac35f8ccc17

	Name	Image	Status	Port(s)	Started	Actions
<input type="checkbox"/>	<div>affectionate_pascal</div> <div>f6ebfa7bf892</div>	reactdocker:dev	Running	3000:3000	55 seconds ago	<div></div> <div></div> <div></div>

Showing 1 items

RAM 2.37 GB CPU 0.34% Connected to Hub

v4.16.3

http://localhost:3000

Here is the output :

