

- To run docker images, turn docker images to docker container, two ways, either docker run on each one's terminal or just docker compose up for all.

- First way:

1. Create docker files in directories where codes exist.
2. To create docker images, use docker build

- Frontend:

The screenshot shows a VS Code editor with a Dockerfile being created and built. The Dockerfile content is as follows:

```
1 FROM node:latest
2
3 WORKDIR /frontendtest
4
5 COPY . .
6
7 RUN npm install next \
8     npm install \
9     npm install nodemon \
10    npm install pg
11
12 EXPOSE 3000
13
14 CMD ["npm", "run", "dev"]
```

The terminal output shows the build process:

```
PS C:\Users\ DELL\Downloads\SkillsMatch> docker build -t frontendimage frontend
[+] Building 0.0s (0/0) docker:default
[+] Building 14.5s (9/9) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 226B
=> [internal] load metadata for docker.io/library/node:latest
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/4] FROM docker.io/library/node:latest@sha256:a8ba58f54e770a0f910ec36d25f8a4f1670e741a58c2e6358b2c30b575c84263
=> [internal] load build context
=> => transferring context: 52.70kB
=> CACHED [2/4] WORKDIR /frontendtest
=> CACHED [3/4] COPY . .
=> CACHED [4/4] RUN npm install next     npm install     npm install nodemon     npm install pg
=> exporting to image
=> => exporting layers
=> writing image sha256:2a9847454b39ba3c320e73e2ff780c06e248ef48719d1162a8f79755b3d82f00
=> naming to docker.io/library/frontendimage

What's Next?
1. Sign in to your Docker account + docker login
2. View a summary of image vulnerabilities and recommendations + docker scout quickview
PS C:\Users\ DELL\Downloads\SkillsMatch> docker run --name frontendcontainer -p 3000:3000 frontendimage:latest
> jobi@0.1.0 dev
> next dev

▲ Next.js 14.2.3
- Local:    http://localhost:3000

✓ Starting...
Attention: Next.js now collects completely anonymous telemetry regarding usage.
This information is used to shape Next.js' roadmap and prioritize features.
You can learn more, including how to opt-out if you'd not like to participate in this anonymous program, by visiting the following URL:
https://nextjs.org/telemetry
```

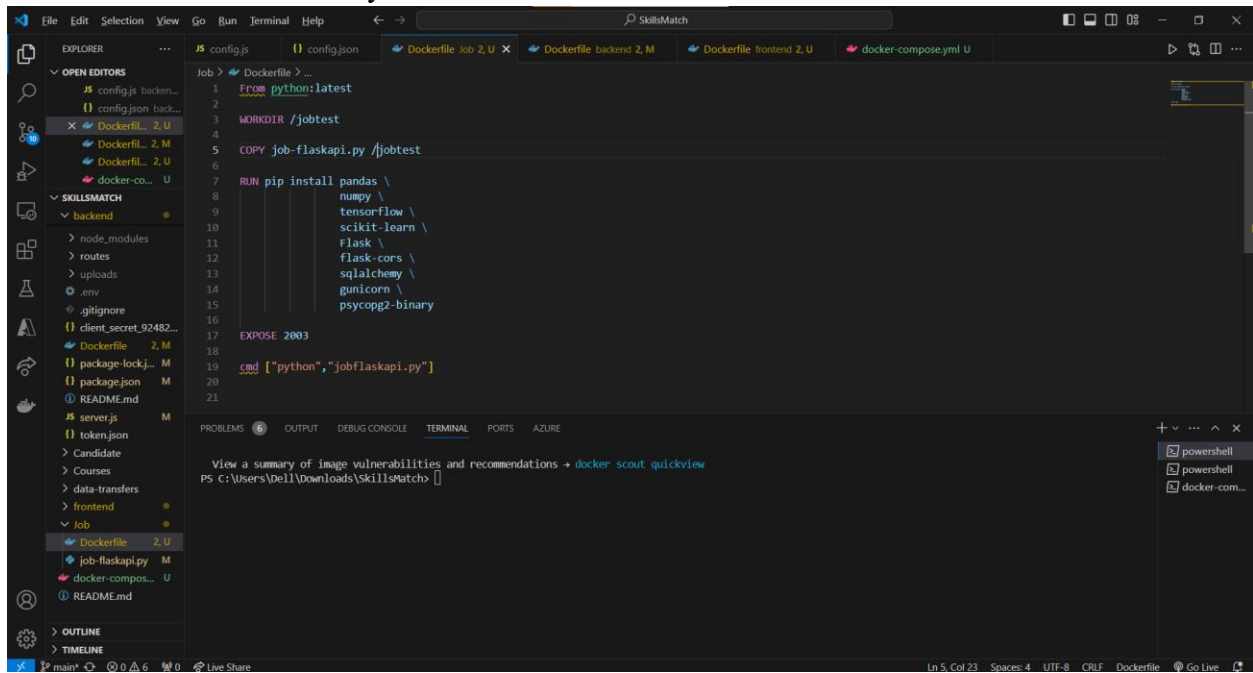
The second screenshot shows the Dockerfile being edited with the following content:

```
1 FROM node:latest
2
3 WORKDIR /frontendtest
4
5 COPY . .
6
7 RUN npm install next \
8     npm install \
9     npm install nodemon \
10    npm install pg
11
12 EXPOSE 3000
13
14 CMD ["npm", "run", "dev"]
```

The terminal output shows the build process:

```
at Provider (webpack-internal:///((ssr))/.node_modules/react-redux/es/components/Provider.js:16:3)
at Providers (webpack-internal:///((ssr))/.src/redux/provider.tsx:12:22)
at lazy
at body
at html
at RedirectErrorBoundary (webpack-internal:///((ssr))/.node_modules/next/dist/client/components/redirect-boundary.js:73:9)
at RedirectBoundary (webpack-internal:///((ssr))/.node_modules/next/dist/client/components/redirect-boundary.js:81:11)
at ReactDevOverlay (webpack-internal:///((ssr))/.node_modules/next/dist/client/components/react-dev-overlay/app/ReactDevOverlay.js:87:9)
at HotReload (webpack-internal:///((ssr))/.node_modules/next/dist/client/components/react-dev-overlay/app/hot-reloader-client.js:322:11)
at Router (webpack-internal:///((ssr))/.node_modules/next/dist/client/components/app-router.js:202:11)
at ErrorBoundaryHandler (webpack-internal:///((ssr))/.node_modules/next/dist/client/components/error-boundary.js:112:9)
at ErrorBoundary (webpack-internal:///((ssr))/.node_modules/next/dist/client/components/error-boundary.js:158:11)
at AppRouter (webpack-internal:///((ssr))/.node_modules/next/dist/client/components/app-router.js:556:13)
at lazy
at r1 (/frontendtest/node_modules/next/dist/compiled/next-server/app-page.runtime.dev.js:39:18701)
at r1 (/frontendtest/node_modules/next/dist/compiled/next-server/app-page.runtime.dev.js:39:18701)
at ServerInsertedHTMLProvider (/frontendtest/node_modules/next/dist/compiled/next-server/app-page.runtime.dev.js:39:24131)
GET / 200 in 2174ms
GET /favicon.ico 200 in 332ms
```

## ➤ Job Recommender System:

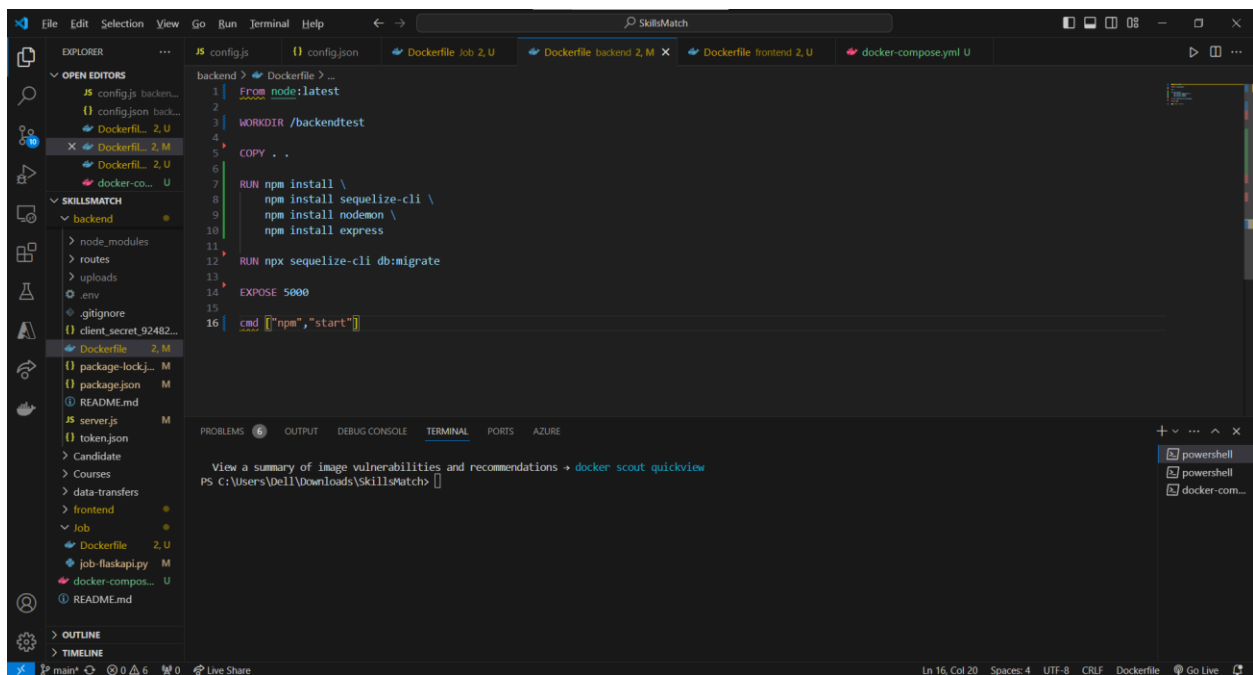


The screenshot shows the VS Code interface with the Dockerfile for the Job Recommender System. The Explorer sidebar on the left shows the project structure, including files like config.js, config.json, Dockerfile, job-flaskapi.py, and docker-compose.yml. The Dockerfile is open in the editor, showing the following content:

```
1 FROM python:latest
2
3 WORKDIR /jobtest
4
5 COPY job-flaskapi.py /jobtest
6
7 RUN pip install pandas \
8     numpy \
9     tensorflow \
10    scikit-learn \
11    flask \
12    flask-cors \
13    sqlalchemy \
14    gunicorn \
15    psycpg2-binary
16
17 EXPOSE 2003
18
19 CMD ["python", "jobflaskapi.py"]
```

The bottom panel shows the TERMINAL with the command `docker scout quickview` and its output, which is a summary of image vulnerabilities and recommendations.

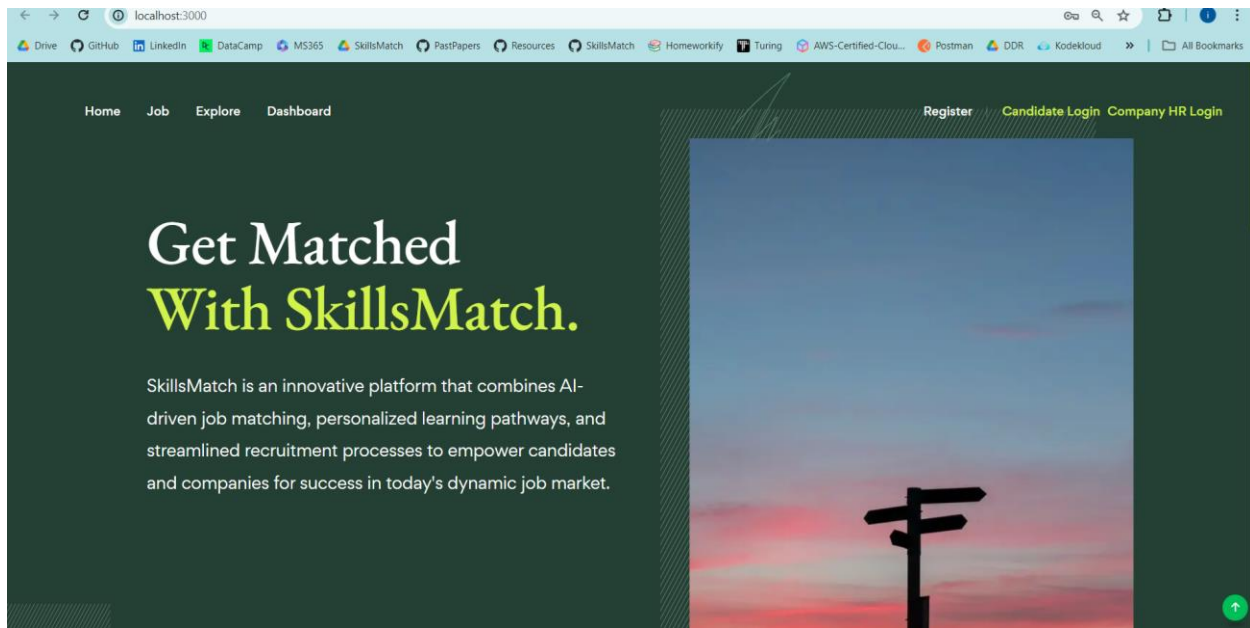
## ➤ Backend:



The screenshot shows the VS Code interface with the Dockerfile for the Backend. The Explorer sidebar on the left shows the project structure, including files like config.js, config.json, Dockerfile, job-flaskapi.py, and docker-compose.yml. The Dockerfile is open in the editor, showing the following content:

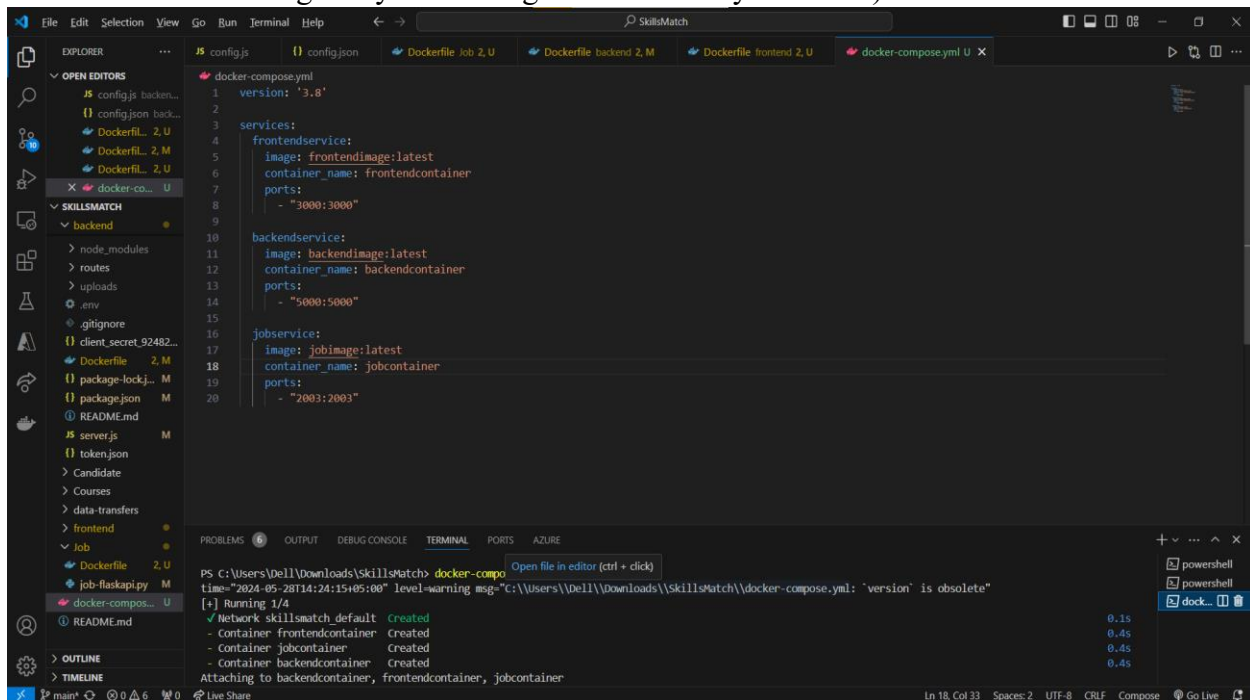
```
1 FROM node:latest
2
3 WORKDIR /backendtest
4
5 COPY . .
6
7 RUN npm install \
8     npm install sequelize-cli \
9     npm install nodemon \
10    npm install express
11
12 RUN npx sequelize-cli db:migrate
13
14 EXPOSE 5000
15
16 CMD ["npm", "start"]
```

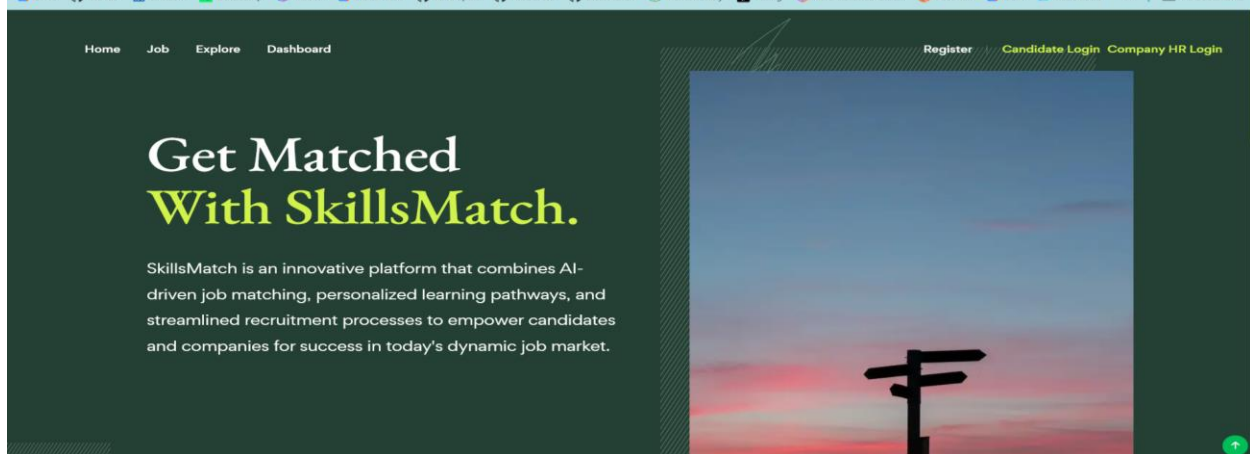
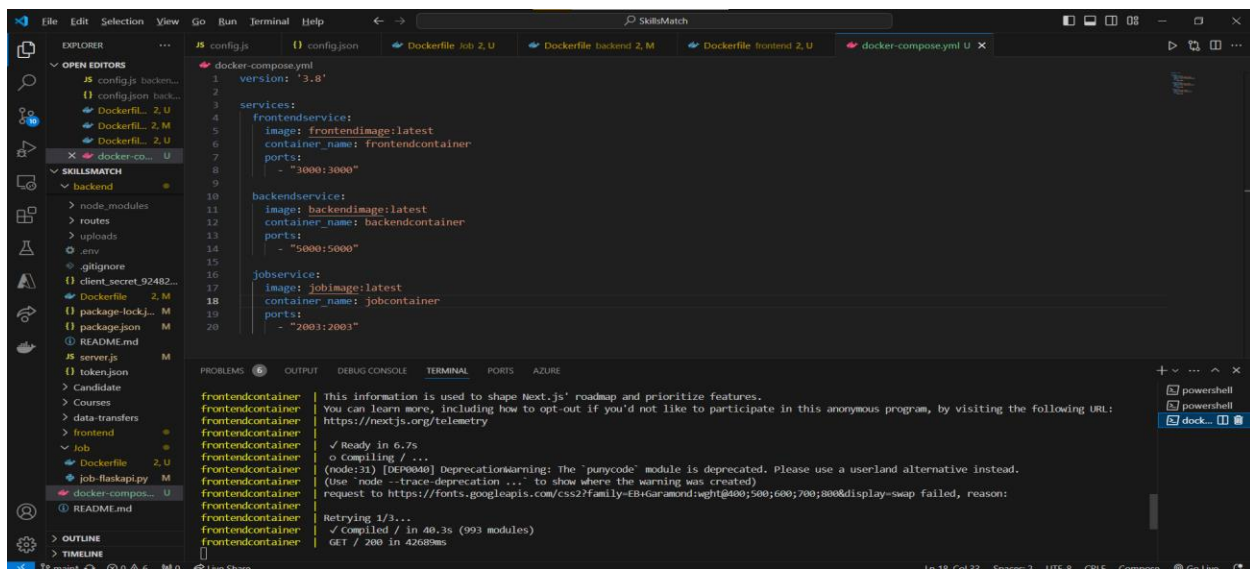
The bottom panel shows the TERMINAL with the command `docker scout quickview` and its output, which is a summary of image vulnerabilities and recommendations.



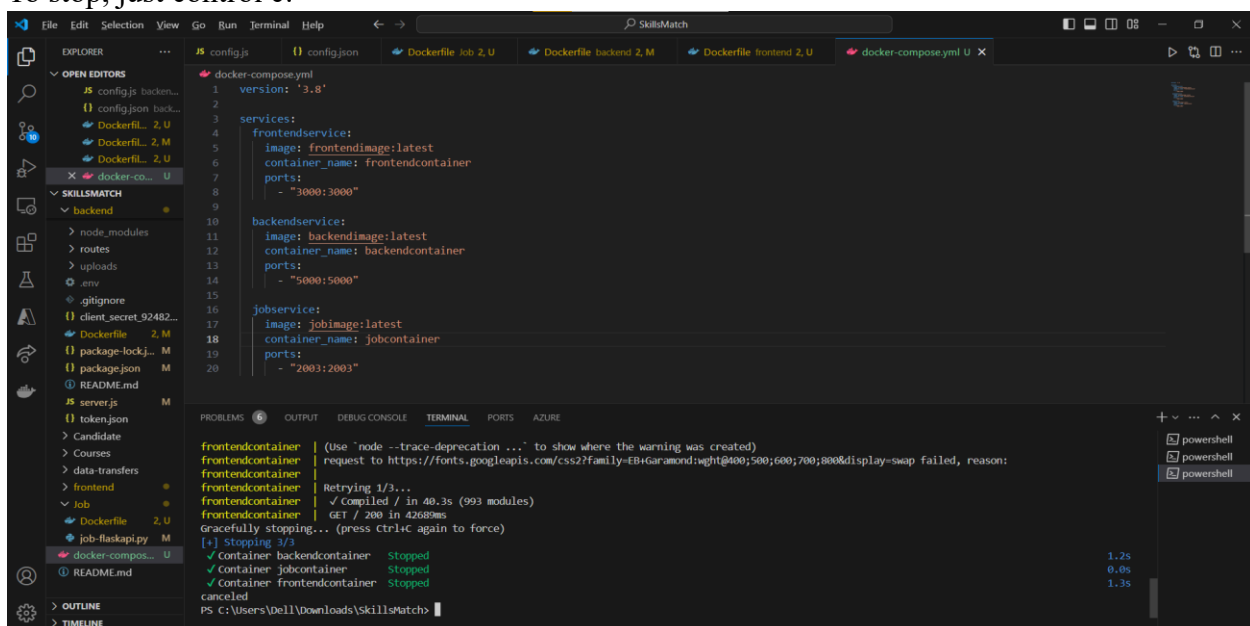
## ➤ Second way: Docker compose

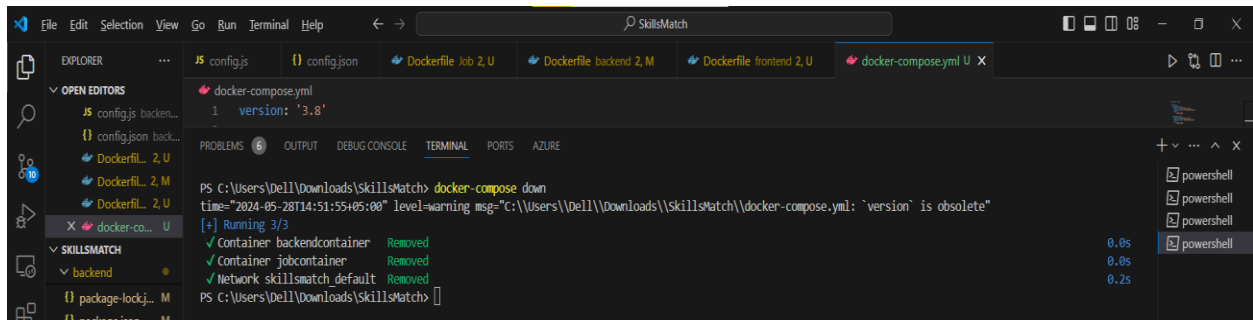
In this scenario, we are using prebuilt images. (if we want to create image, we can build stage in docker-compose.yml or put both, build and image, stages and docker will create new image only if the images aren't already available)





To stop, just control c:





For manual container and image removal:

