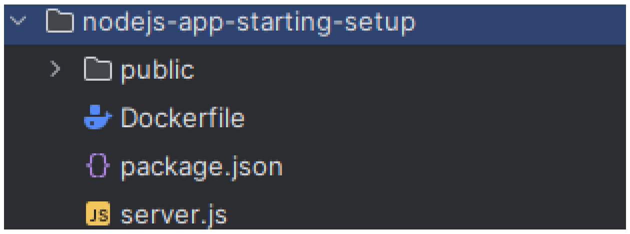
```
FROM eclipse-temurin:17

LABEL mentainer="bhuvandosapati517@gmail.com"

WORKDIR /app

COPY target/docker-learning-demo-0.0.1-SNAPSHOT.jar
/app/springboot-docker-demo.jar

ENTRYPOINT ["java","-jar","springboot-docker-demo.jar"]
```

Node js docker file :



```
nodejs-app-starting-setup
    public
    Dockerfile
    package.json
    server.js
```

```
FROM node:14

WORKDIR /app

COPY package.json .

RUN npm install

COPY . .

EXPOSE 3000

CMD [ "node", "app.mjs" ]
```

Based on your PowerShell history, you've been working extensively with a Spring Boot and MySQL microservices setup, moving from local Docker to Docker Compose and eventually into Kubernetes (Minikube).

Here are the **unique Docker commands** extracted from your history, categorized by their function.

🏗️ Image Management

These commands deal with building, tagging, and moving images to a registry (Docker Hub).

Command	Purpose
`docker build .`	Builds an image from a Dockerfile in the current directory.
`docker build -t name:tag .`	Builds an image and assigns it a specific **name** and **tag**.
`docker images`	Lists all locally available images.
`docker tag source:tag target:tag`	Creates a new alias (tag) for an existing image (used before pushing).
`docker login`	Authenticates your CLI with Docker Hub.
`docker push user/repo:tag`	Uploads your image to a remote registry.
`docker pull image:tag`	Downloads an image from a registry.
`docker rmi image`	Removes a specific image from your local machine.

🚀 Container Lifecycle

These are the "verb" commands that handle the actual running processes.

Command	Purpose
docker run	Creates and starts a new container.
docker start	Starts an existing, stopped container.
docker stop	Gracefully shuts down a running container.
docker kill	Forces a container to stop immediately (hard shutdown).
docker rm	Deletes a stopped container.
docker container prune	Deletes all stopped containers at once.

🔍 Inspection & Debugging

Essential for seeing what's happening inside your containers.

Command	Purpose
docker ps	Lists currently **running** containers.
docker ps -a	Lists **all** containers (running and stopped).
docker logs -f name	Follows/streams the live output logs of a container.

docker exec -it name bash	Opens an interactive terminal inside a running container.
docker info	Displays system-wide information about the Docker installation.
docker inspect	Returns detailed low-level info on a Docker object (JSON).

🌐 Networking

You used these to help your Spring Boot app talk to your MySQL database.

Command	**Purpose**
docker network create name	Creates a virtual network for containers to communicate.
docker network ls	Lists all available Docker networks.
docker network inspect name	Shows which containers are connected to a specific network.

🛠️ Docker Compose

Commands used to manage multi-container applications (defined in your `docker-compose.yml`).

Command	**Purpose**

`docker-compose up`	Starts all services defined in the YAML file.
`docker-compose up -d`	Starts services in **detached** mode (runs in background).
`docker-compose up --build`	Forces a rebuild of images before starting containers.
`docker-compose down`	Stops and removes containers, networks, and images created by `up`.

💡 Pro-Tip from your history:

I noticed you used `docker run -p 8080:8080 -e SPRING_DATASOURCE_URL=...`. Since you have moved to **Kubernetes**, remember that these environment variables are now better handled inside your `deployment.yaml` or via `ConfigMaps` (which I see you started exploring with `kubectl apply -f environment.yaml`).

Would you like me to summarize the unique Kubernetes (`kubectl`) commands from your history as well?