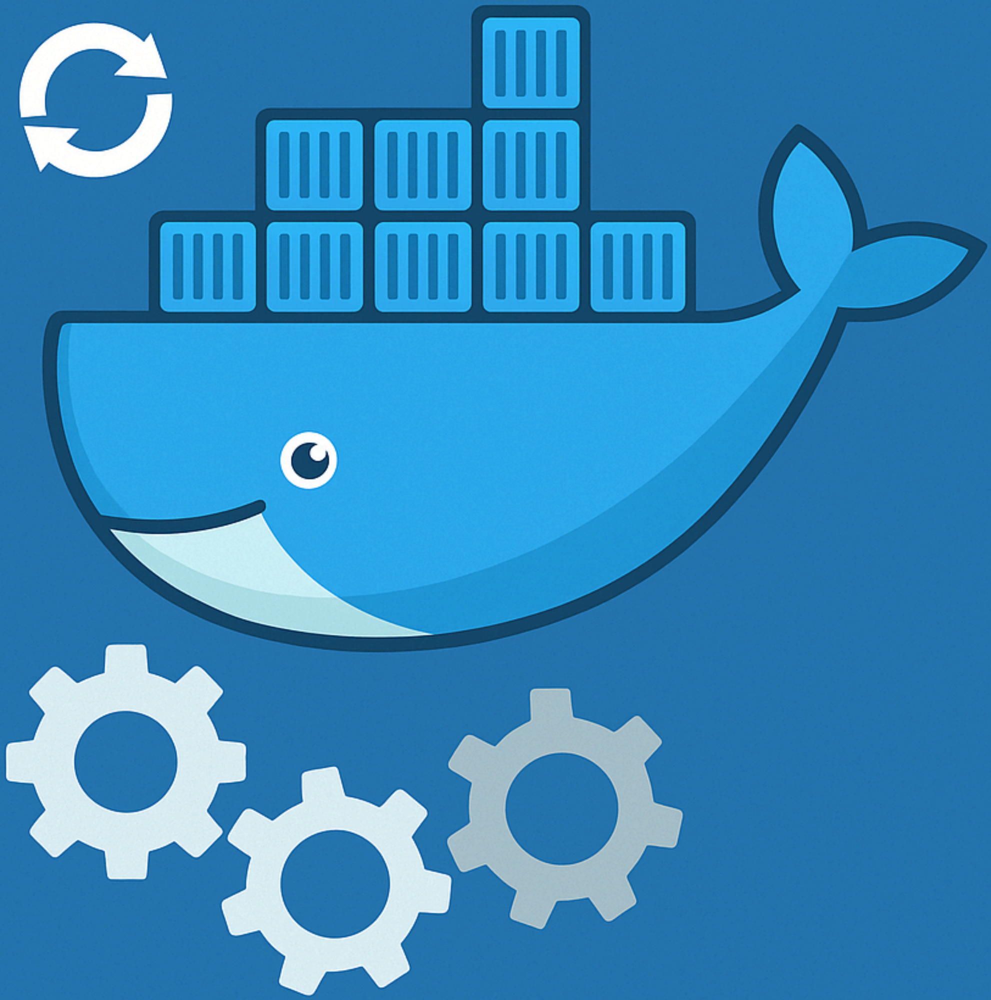
DOCKER MULTI-STAGE BUILD

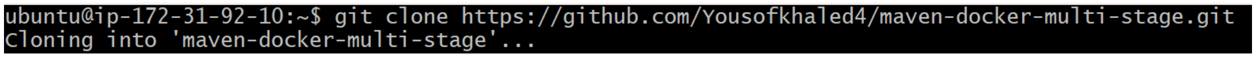


🚀 Dockerizing a Maven Spring Boot App Using Multi-Stage Build

🔧 Step-by-Step Instructions

1. Clone the GitHub Repository

git clone https://github.com/Yousofkhaled4/maven-docker-multi-stage.git
cd maven-docker-multi-stage



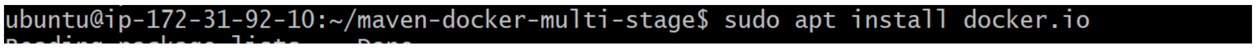
```
ubuntu@ip-172-31-92-10:~$ git clone https://github.com/Yousofkhaled4/maven-docker-multi-stage.git
Cloning into 'maven-docker-multi-stage'...
```

📌 *Clones a public Maven Spring Boot project prepared with a multi-stage Dockerfile.*

---

2. Install Docker (if not already installed)

sudo apt update
sudo apt install docker.io



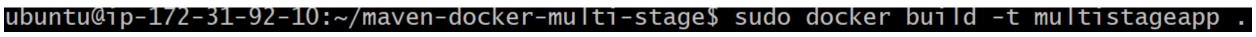
```
ubuntu@ip-172-31-92-10:~/maven-docker-multi-stage$ sudo apt install docker.io
```

📌 *Installs Docker engine on Ubuntu to allow containerization.*

---

3. Build the Docker Image

sudo docker build -t multistageapp .



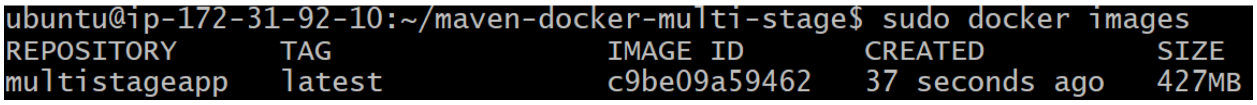
```
ubuntu@ip-172-31-92-10:~/maven-docker-multi-stage$ sudo docker build -t multistageapp .
```

📌 *Builds the Docker image using the Dockerfile present in the project. The `-t` flag tags the image with the name `multistageapp`.*

---

4. Verify the Docker Image

sudo docker images



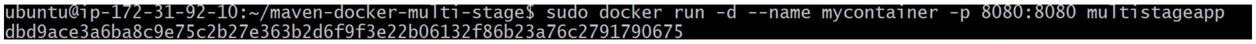
```
ubuntu@ip-172-31-92-10:~/maven-docker-multi-stage$ sudo docker images
REPOSITORY       TAG                 IMAGE ID        CREATED             SIZE
multistageapp    latest              c9be09a59462    37 seconds ago      427MB
```

📌 *Lists all Docker images available locally. You should see* `multistageapp` *with the appropriate size (e.g., ~427MB).*

---

5. Run the Docker Container

sudo docker run -d --name mycontainer -p 8080:8080 multistageapp



```
ubuntu@ip-172-31-92-10:~/maven-docker-multi-stage$ sudo docker run -d --name mycontainer -p 8080:8080 multistageapp
dbd9ace3a6ba8c9e75c2b27e363b2d6f9f3e22b06132f86b23a76c2791790675
```

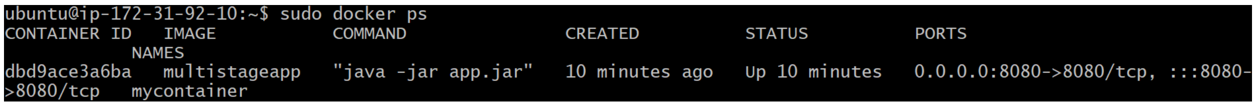
📌 *Runs the container in detached mode `(-d)` with:

- Port forwarding from host `8080` to container `8080`
- Custom container name `mycontainer`*

---

6. Verify Running Container

sudo docker ps



```
ubuntu@ip-172-31-92-10:~$ sudo docker ps
CONTAINER ID   IMAGE           COMMAND                CREATED          STATUS          PORTS
            NAMES
dbd9ace3a6ba   multistageapp   "java -jar app.jar"    10 minutes ago   Up 10 minutes   0.0.0.0:8080->8080/tcp, :::8080-
>8080/tcp   mycontainer
```

📌 *Confirms that the container is up and running. Look for:

- IMAGE name `multistageapp`
- COMMAND `java -jar app.jar`
- STATUS `Up`*

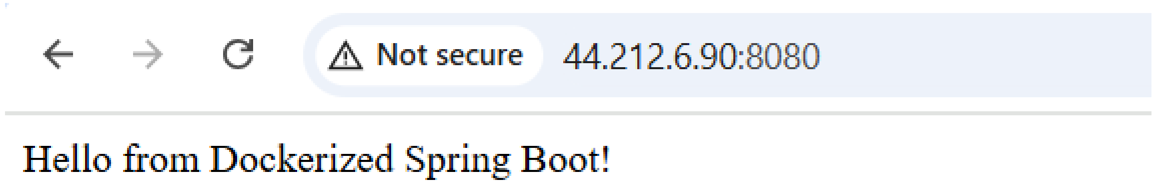
---

7. Access the Application in Browser

Open:
`http://44.212.6.90:8080/`

Output:

```
Hello from Dockerized Spring Boot!
```



Hello from Dockerized Spring Boot!

✅ You've successfully containerized a Spring Boot app with a multi-stage Dockerfile.

🧱 Benefits of Docker Multi-Stage Builds

1. **Smaller Image Size**
   Build dependencies are excluded from the final image.
2. **Better Caching**
   Separate stages help reuse layers for faster rebuilds.
3. **Security**
   Reduces attack surface by removing tools used during build.
4. **Clean Separation**
   Keeps build and runtime environments isolated.

🛠️ Sample Multi-Stage Dockerfile Explained

```
# Stage 1: Build
FROM maven:3.8.5-openjdk-17 as build
WORKDIR /app
COPY . .
RUN mvn clean package -DskipTests

# Stage 2: Runtime
FROM openjdk:17
WORKDIR /app
COPY --from=build /app/target/app.jar app.jar
CMD ["java", "-jar", "app.jar"]
```

🧩 *Explanation*:

- First stage builds the app using Maven.
- Second stage uses only the built `jar` file — cleaner and smaller image.

✅ Dockerfile Best Practices (from Docker Docs)

- Use **multi-stage builds** to minimize image size.
- Use `.dockerignore` to avoid copying unnecessary files.
- Prefer **official base images** (like `openjdk`, `alpine`).
- Set a **WORKDIR** instead of using absolute paths everywhere.
- Combine `RUN` commands to reduce layers.
- Use **COPY** over **ADD** unless you need archive extraction or remote URLs.

🚀 Exploring `docker init` (New Feature)

> `docker init` is a new command to **bootstrap** a Docker project quickly.

How to Use:

```
docker init
```

This command:

- Interactively asks questions about your tech stack (e.g., Java, Python).
- Generates a starter `Dockerfile`, `.dockerignore`, and optional Compose file.

📌 *Use this to scaffold Docker support in new projects fast and consistently.*
