

Deploy a Full Stack Java Application With Docker

Steps to do This project

1. Prerequisites

Ensure you have the following installed:

Java Development Kit (JDK)

Docker and Docker Compose

Git (optional, for version control)

2. Application Structure

Your full stack application should typically have:

Backend: Java application using Spring Boot or similar framework.

Frontend: JavaScript framework like React, Angular, or Vue.js.

Database: MySQL, PostgreSQL, or any other database of choice.

3. Dockerize Backend

Create a Dockerfile for your backend. Here's an example for a Spring Boot application

4. Dockerize Frontend

Create a Dockerfile for your frontend application.

5. Docker Compose

Use Docker Compose to manage multi-container applications. Create a docker-compose.yml file:

6. Build and Run

Navigate to the directory containing the docker-compose.yml file and run

7. Access the Application

Backend: <http://localhost:8070>

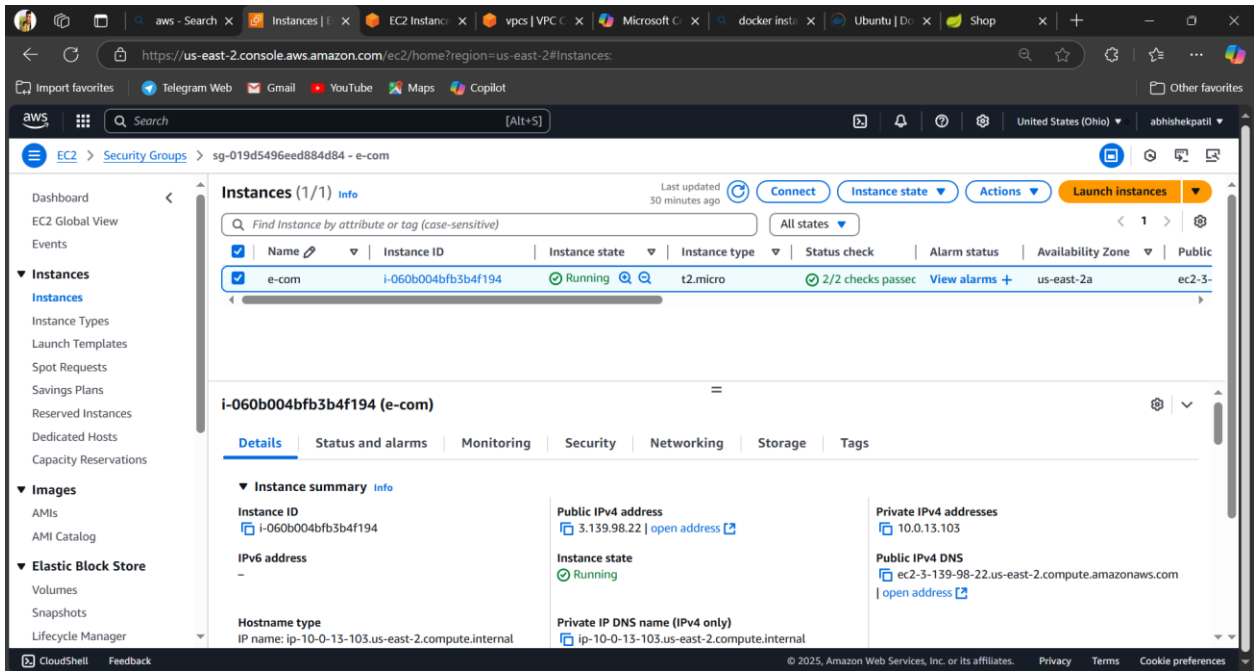
8. Managing Containers

To stop the containers: docker-compose down

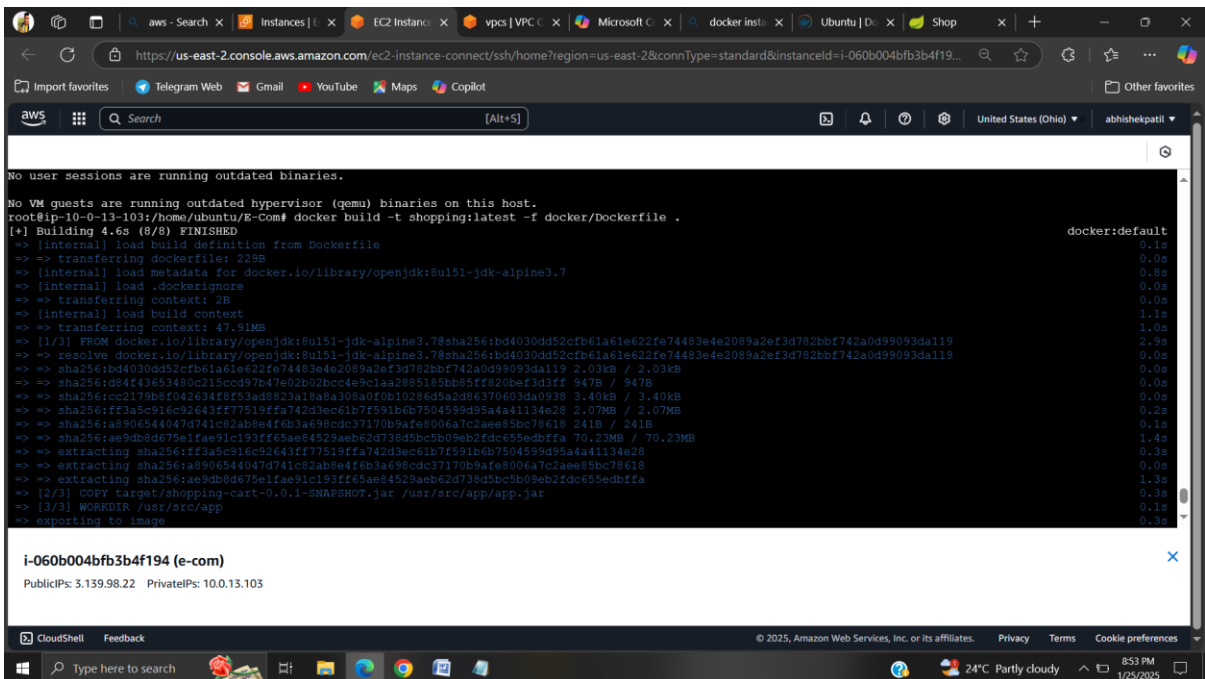
To start the containers: docker-compose up

To view logs: docker-compose logs -f

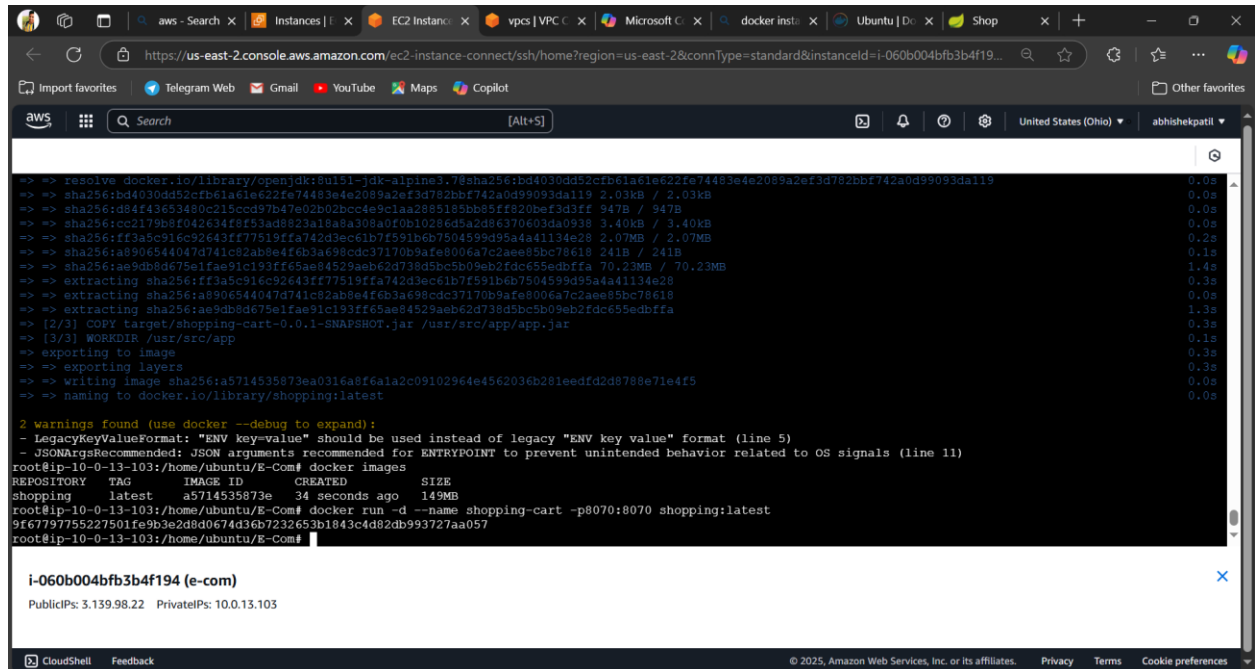
- Create an EC2 instance with Name, Ami, keyPair, Network Settings



- Connect with SSH Client
- Install openjdk-11-jre -y
- Install maven -y
- Docker Installation
- <https://docs.docker.com/engine/install/ubuntu/>
- Repo <https://github.com/AbhishekPatil06/E-Com.git>



After Installation JDK, Maven, Docker & Package Installation , Docker Build Of Shopping Cart

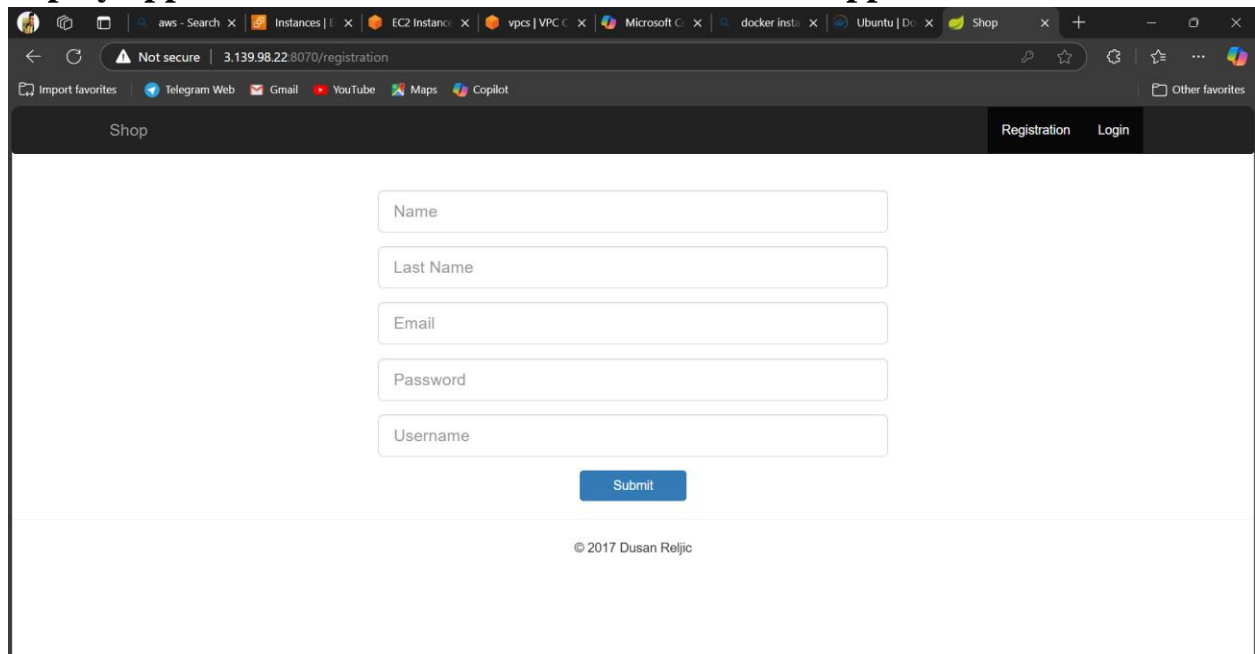


```
>> resolve docker.io/library/openjdk:8u151-jdk-alpine3.78sha256:bd4030dd52cfb61a61e622fe74483e4e2089a2ef3d782bbf742a0d99093da119 0.0s
>> sha256:bd4030dd52cfb61a61e622fe74483e4e2089a2ef3d782bbf742a0d99093da119 2.03kB / 2.03kB 0.0s
>> sha256:d84443e53490218ccdc7e47e02b02cc4e9e1a208618b0b55ffa20be43d3ff 947B / 947B 0.0s
>> sha256:cc217988f042c34f8f53ad8823a18a8a308a0f0b10286d5a23863706c3da0938 3.40kB / 3.40kB 0.0s
>> sha256:ff3a5c916c92643ff77519ffa742d3ec61b7f591b6b7504599d95a4a41134e28 2.07MB / 2.07MB 0.2s
>> sha256:ae9db8d675e1fae91c193ff65ae84529aeb62d738d5bc5b09eb2fdc655edbffa 241B / 241B 0.1s
>> sha256:ae9db8d675e1fae91c193ff65ae84529aeb62d738d5bc5b09eb2fdc655edbffa 70.23MB / 70.23MB 1.4s
>> extracting sha256:ff3a5c916c92643ff77519ffa742d3ec61b7f591b6b7504599d95a4a41134e28 0.3s
>> extracting sha256:ae9db8d675e1fae91c193ff65ae84529aeb62d738d5bc5b09eb2fdc655edbffa 0.0s
>> extracting sha256:ae9db8d675e1fae91c193ff65ae84529aeb62d738d5bc5b09eb2fdc655edbffa 1.3s
>> [2/3] COPY target/shopping-cart-0.0.1-SNAPSHOT.jar /usr/src/app/app.jar 0.3s
>> [3/3] WORKDIR /usr/src/app 0.1s
>> exporting to image 0.3s
>> exporting layers 0.3s
>> writing image sha256:a5714535873ea0316a8f6a1a2c09102964e4562036b281eedfd2d8788e71e4f5 0.0s
>> naming to docker.io/library/shopping:latest 0.0s

2 warnings found (use docker --debug to expand):
- LegacyKeyValueFormat: "ENV key=value" should be used instead of legacy "ENV key value" format (line 5)
- JSONArgsRecommended: JSON arguments recommended for ENTRYPOINT to prevent unintended behavior related to OS signals (line 11)
root@ip-10-0-13-103:/home/ubuntu/E-Com# docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
shopping latest a5714535873e 34 seconds ago 149MB
root@ip-10-0-13-103:/home/ubuntu/E-Com# docker run -d --name shopping-cart -p8070:8070 shopping:latest
9f67797755227501fe9b3e2d8d0674d36b7232653b1843c4d82db993727aa057
root@ip-10-0-13-103:/home/ubuntu/E-Com#
```

i-060b004bfb3b4f194 (e-com)
PublicIPs: 3.139.98.22 PrivateIPs: 10.0.13.103

Deploy application in a docker container access the application



Shop

Registration Login

Name

Last Name

Email

Password

Username

Submit

© 2017 Dusan Reljic

