CONTAINERIZED APPLICATION DEPLOYMENT FOR DEVELOPMENT TEAMS

1. Introduction

Modern development teams often face deployment failures due to **environment mismatch**, commonly called the **“Works on My Machine”** problem. Applications behave differently across developer systems, testing servers, and production environments.

To solve this issue, **containerization using Docker** provides a standardized, consistent environment where the application runs the same everywhere.

This project demonstrates how a DevOps Engineer introduces container-based deployment using **AWS Cloud UI** without disturbing existing workflows.

2. Problem Statement

The development team frequently encounters:

- Deployment failures
- Dependency conflicts
- Version mismatch between environments

The IT team requires:

- A **standard deployment method**
- Consistent execution across all systems
- Easy rollback and reproducibility

The DevOps intern must implement a **container-based deployment solution**.

3. Project Objective

To containerize a sample application and manage it like a production service using:

- Docker installation and configuration
- Image creation using Dockerfile
- Running containers with port mapping
- Data persistence using volumes
- Image storage in container registry
- Container lifecycle management
- Multi-service setup using Docker Compose

4. Architecture Overview

Workflow

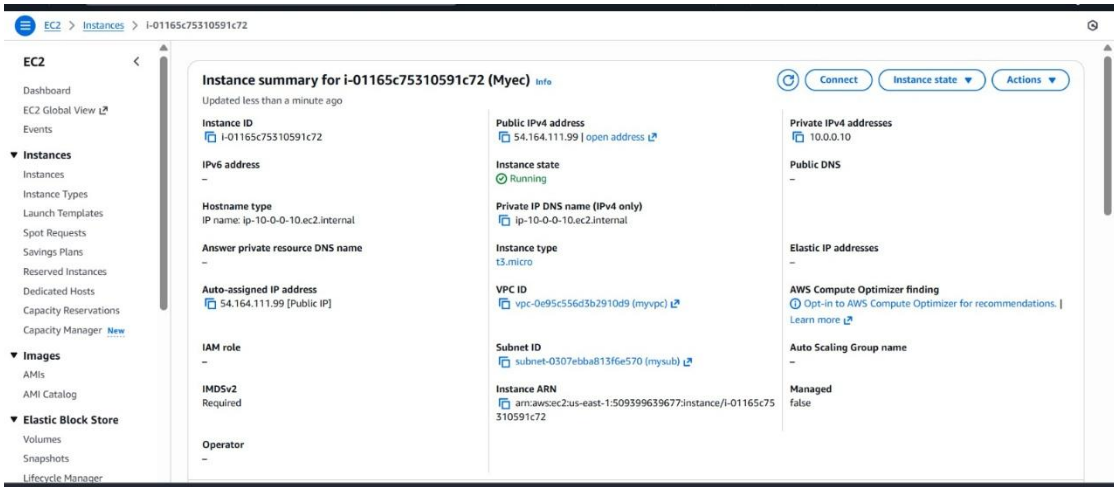
Developer Code → Docker Image → Container → AWS Registry (ECR) → Deployment

This ensures:

- Same environment in Dev, Test, and Production
- Faster deployments
- Easy scaling

5. Implementation Using AWS Cloud UI (Step-by-Step)

Step 1: Launch an EC2 Instance

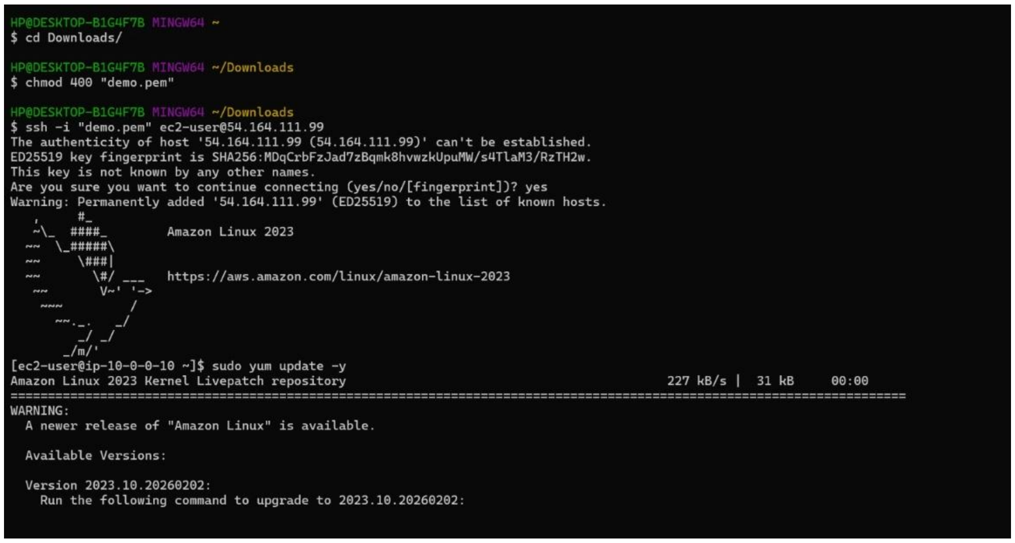


Instance summary for i-01165c75310591c72 (Myec) Info

Updated less than a minute ago

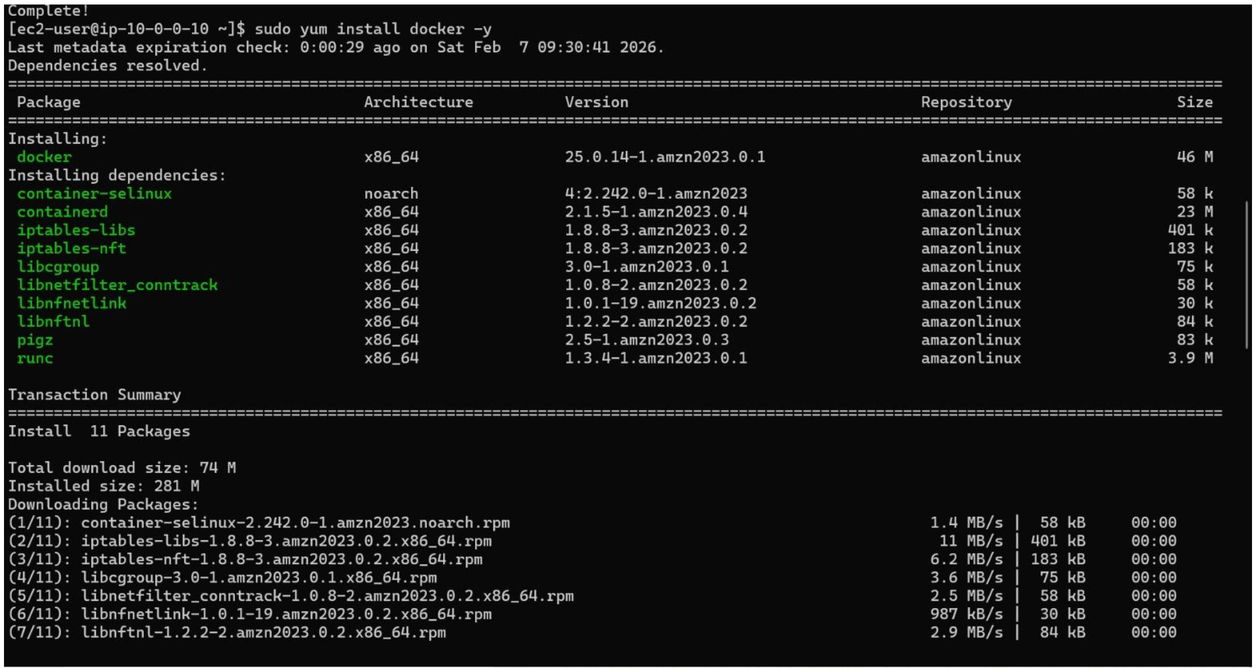
Instance ID i-01165c75310591c72	**Public IPv4 address** 54.164.111.99 | open address	**Private IPv4 addresses** 10.0.0.10
IPv6 address –	**Instance state** Running	**Public DNS** –
Hostname type IP name: ip-10-0-0-10.ec2.internal	**Private IP DNS name (IPv4 only)** ip-10-0-0-10.ec2.internal	
Answer private resource DNS name –	**Instance type** t3.micro	**Elastic IP addresses** –
Auto-assigned IP address 54.164.111.99 [Public IP]	**VPC ID** vpc-0e95c556d3b2910d9 (myvpc)	**AWS Compute Optimizer finding** Opt-in to AWS Compute Optimizer for recommendations. | Learn more
IAM role –	**Subnet ID** subnet-0307ebba813f6e570 (mysub)	**Auto Scaling Group name** –
IMDSv2 Required	**Instance ARN** arn:aws:ec2:us-east-1:509399639677:instance/i-01165c75310591c72	**Managed** false
Operator –		

Step 2: Connect to EC2



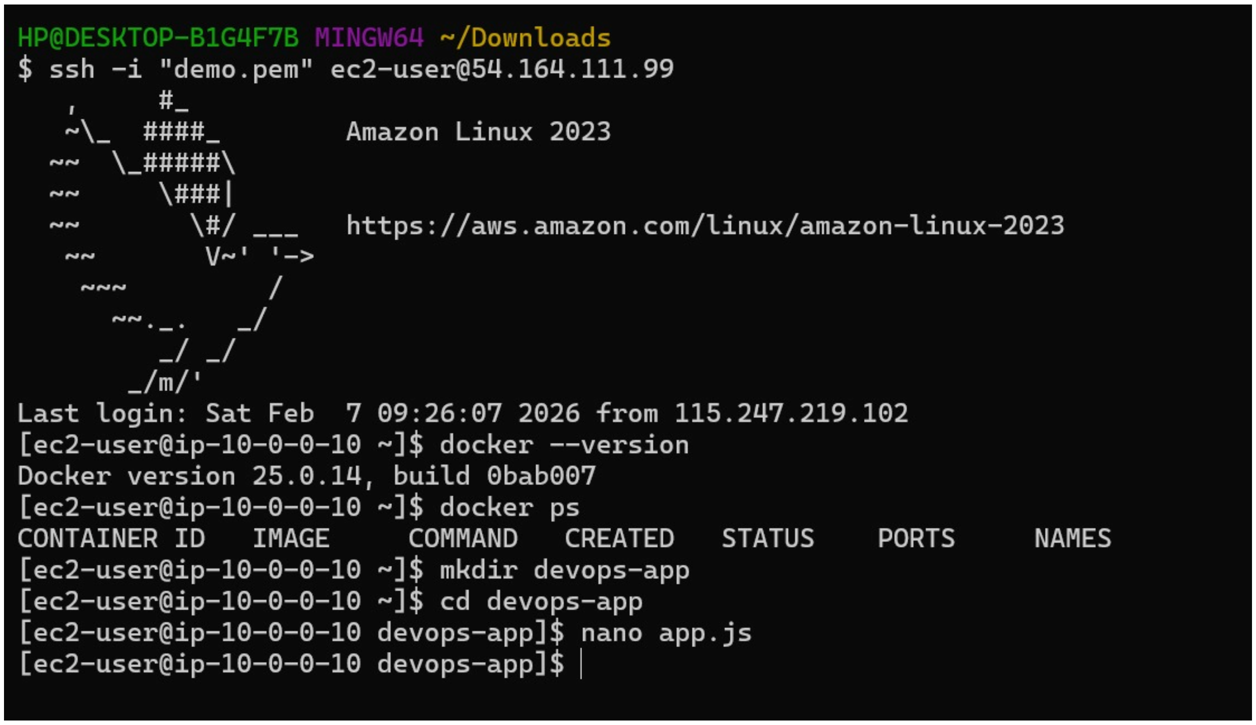
```
HP@DESKTOP-B1G4F7B MINGW64 ~
$ cd Downloads/

HP@DESKTOP-B1G4F7B MINGW64 ~/Downloads
$ chmod 400 "demo.pem"

HP@DESKTOP-B1G4F7B MINGW64 ~/Downloads
$ ssh -i "demo.pem" ec2-user@54.164.111.99
The authenticity of host '54.164.111.99 (54.164.111.99)' can't be established.
ED25519 key fingerprint is SHA256:MDqCrbFzJad7zBqmk8hvwzkUpuMW/s4TlaM3/RzTH2w.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '54.164.111.99' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-10-0-0-10 ~]$ sudo yum update -y
Amazon Linux 2023 Kernel Livepatch repository                                      227 kB/s |  31 kB     00:00
==================================================================================================================
WARNING:
  A newer release of "Amazon Linux" is available.

  Available Versions:

  Version 2023.10.20260202:
    Run the following command to upgrade to 2023.10.20260202:
```

Step 3: Install Docker



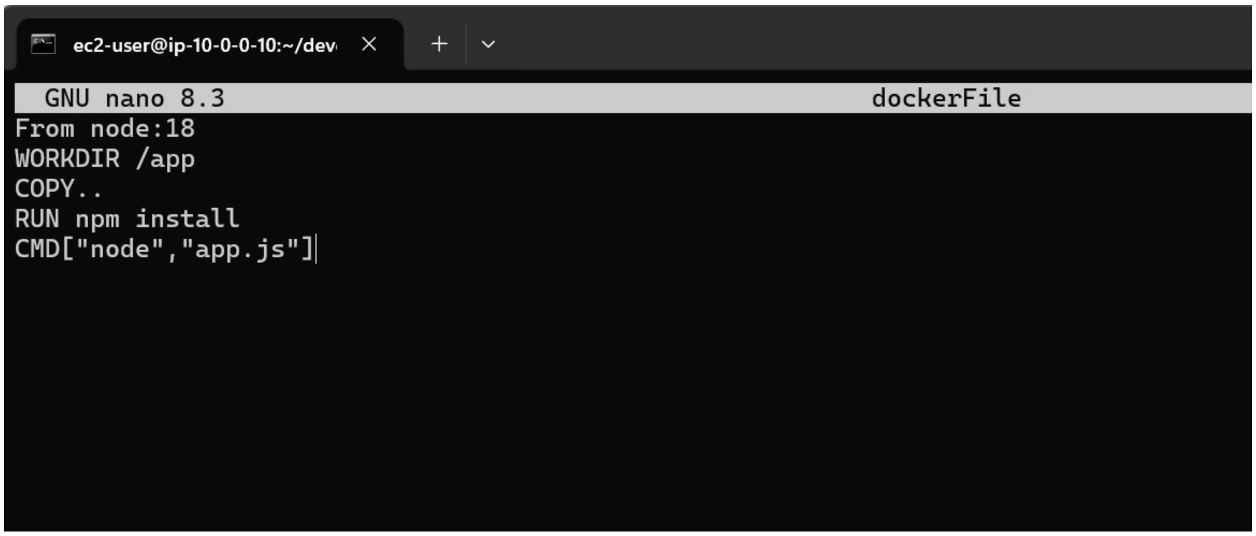
```
Complete!
[ec2-user@ip-10-0-0-10 ~]$ sudo yum install docker -y
Last metadata expiration check: 0:00:29 ago on Sat Feb  7 09:30:41 2026.
Dependencies resolved.
=====================================================================================================================================
 Package                              Architecture          Version                                    Repository               Size
=====================================================================================================================================
Installing:
 docker                               x86_64                25.0.14-1.amzn2023.0.1                     amazonlinux              46 M
Installing dependencies:
 container-selinux                    noarch                4:2.242.0-1.amzn2023                       amazonlinux              58 k
 containerd                           x86_64                2.1.5-1.amzn2023.0.4                       amazonlinux              23 M
 iptables-libs                        x86_64                1.8.8-3.amzn2023.0.2                       amazonlinux             401 k
 iptables-nft                         x86_64                1.8.8-3.amzn2023.0.2                       amazonlinux             183 k
 libcgroup                            x86_64                3.0-1.amzn2023.0.1                         amazonlinux              75 k
 libnetfilter_conntrack               x86_64                1.0.8-2.amzn2023.0.2                       amazonlinux              58 k
 libnfnetlink                         x86_64                1.0.1-19.amzn2023.0.2                      amazonlinux              30 k
 libnftnl                             x86_64                1.2.2-2.amzn2023.0.2                       amazonlinux              84 k
 pigz                                 x86_64                2.5-1.amzn2023.0.3                         amazonlinux              83 k
 runc                                 x86_64                1.3.4-1.amzn2023.0.1                       amazonlinux             3.9 M

Transaction Summary
=====================================================================================================================================
Install  11 Packages

Total download size: 74 M
Installed size: 281 M
Downloading Packages:
(1/11): container-selinux-2.242.0-1.amzn2023.noarch.rpm                                        1.4 MB/s |  58 kB     00:00
(2/11): iptables-libs-1.8.8-3.amzn2023.0.2.x86_64.rpm                                           11 MB/s | 401 kB     00:00
(3/11): iptables-nft-1.8.8-3.amzn2023.0.2.x86_64.rpm                                           6.2 MB/s | 183 kB     00:00
(4/11): libcgroup-3.0-1.amzn2023.0.1.x86_64.rpm                                                3.6 MB/s |  75 kB     00:00
(5/11): libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64.rpm                                 2.5 MB/s |  58 kB     00:00
(6/11): libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64.rpm                                          987 kB/s |  30 kB     00:00
(7/11): libnftnl-1.2.2-2.amzn2023.0.2.x86_64.rpm                                               2.9 MB/s |  84 kB     00:00
```

Step 4: Create Application Folder



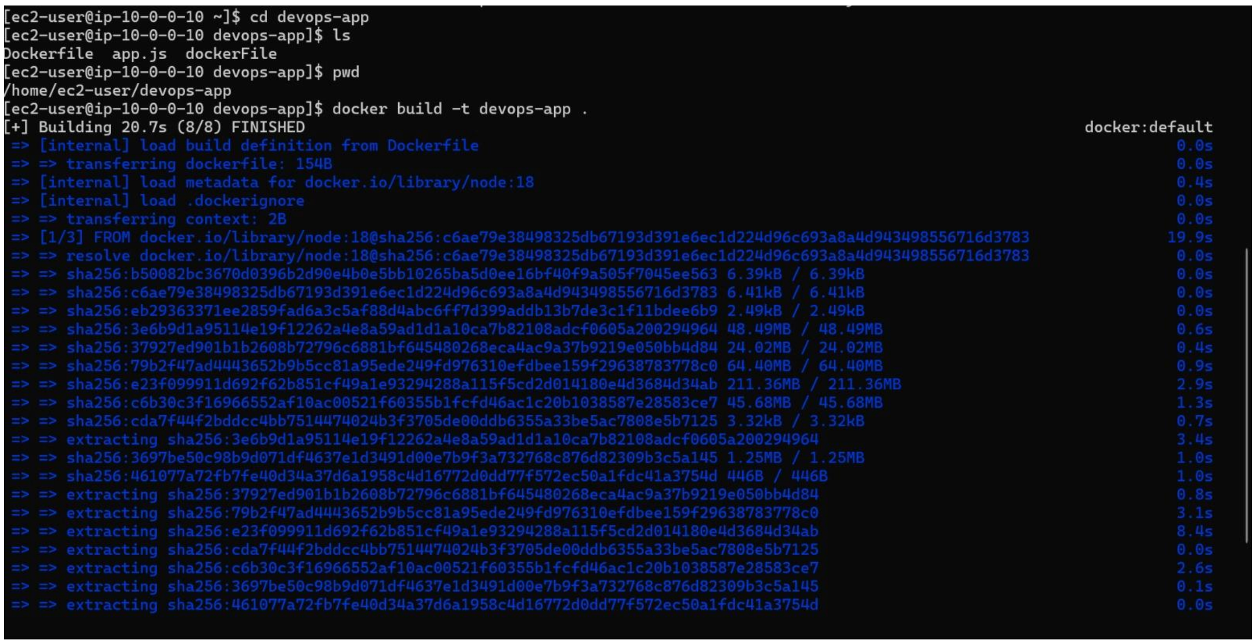
```
HP@DESKTOP-B1G4F7B MINGW64 ~/Downloads
$ ssh -i "demo.pem" ec2-user@54.164.111.99
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Sat Feb  7 09:26:07 2026 from 115.247.219.102
[ec2-user@ip-10-0-0-10 ~]$ docker --version
Docker version 25.0.14, build 0bab007
[ec2-user@ip-10-0-0-10 ~]$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[ec2-user@ip-10-0-0-10 ~]$ mkdir devops-app
[ec2-user@ip-10-0-0-10 ~]$ cd devops-app
[ec2-user@ip-10-0-0-10 devops-app]$ nano app.js
[ec2-user@ip-10-0-0-10 devops-app]$ 
```

Step 5: Create Dockerfile

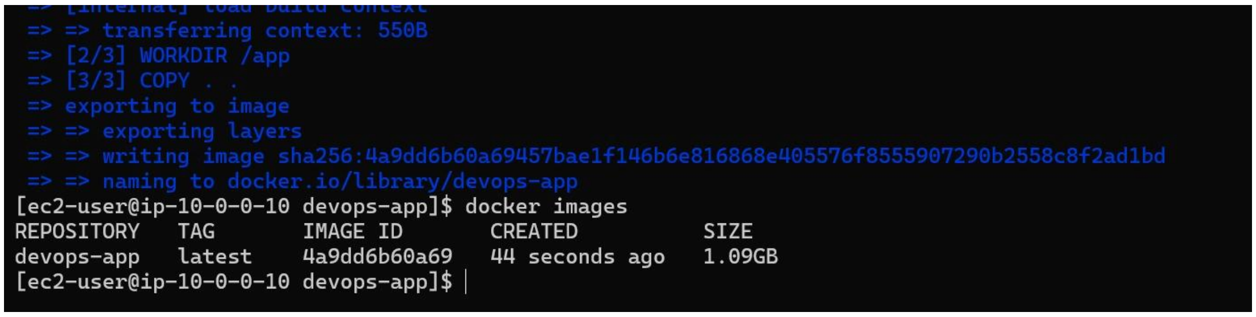


```
ec2-user@ip-10-0-0-10:~/dev
 GNU nano 8.3                                          dockerFile
From node:18
WORKDIR /app
COPY..
RUN npm install
CMD["node","app.js"]
```

Step 6: Build Docker Image

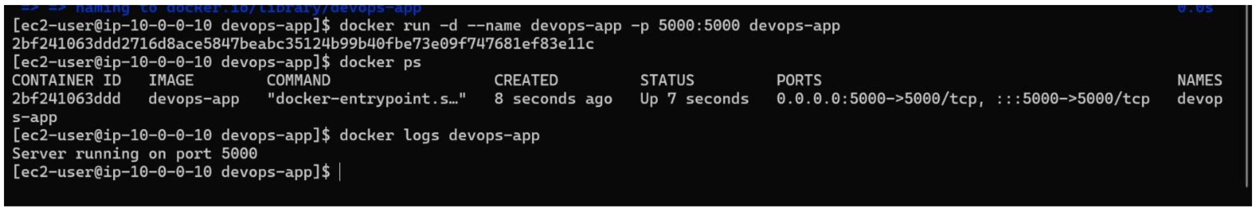


```
[ec2-user@ip-10-0-0-10 ~]$ cd devops-app
[ec2-user@ip-10-0-0-10 devops-app]$ ls
Dockerfile  app.js  dockerFile
[ec2-user@ip-10-0-0-10 devops-app]$ pwd
/home/ec2-user/devops-app
[ec2-user@ip-10-0-0-10 devops-app]$ docker build -t devops-app .
[+] Building 20.7s (8/8) FINISHED                                                                    docker:default
 => [internal] load build definition from Dockerfile                                                          0.0s
 => => transferring dockerfile: 154B                                                                          0.0s
 => [internal] load metadata for docker.io/library/node:18                                                    0.4s
 => [internal] load .dockerignore                                                                             0.0s
 => => transferring context: 2B                                                                               0.0s
 => [1/3] FROM docker.io/library/node:18@sha256:c6ae79e38498325db67193d391e6ec1d224d96c693a8a4d943498556716d3783   19.9s
 => => resolve docker.io/library/node:18@sha256:c6ae79e38498325db67193d391e6ec1d224d96c693a8a4d943498556716d3783    0.0s
 => => sha256:b50082bc3670d0396b2d90e4b0e5bb10265ba5d0ee16bf40f9a505f7045ee563 6.39kB / 6.39kB                0.0s
 => => sha256:c6ae79e38498325db67193d391e6ec1d224d96c693a8a4d943498556716d3783 6.41kB / 6.41kB                0.0s
 => => sha256:eb29363371ee2859fad6a3c5af88d4abc6ff7d399addb13b7de3c1f11bdee6b9 2.49kB / 2.49kB                0.0s
 => => sha256:3e6b9d1a95114e19f12262a4e8a59ad1d1a10ca7b82108adcf0605a200294964 48.49MB / 48.49MB              0.6s
 => => sha256:37927ed901b1b2608b72796c6881bf645480268eca4ac9a37b9219e050bb4d84 24.02MB / 24.02MB              0.4s
 => => sha256:79b2f47ad4443652b9b5cc81a95ede249fd976310efdbee159f29638783778c0 64.40MB / 64.40MB              0.9s
 => => sha256:e23f099911d692f62b851cf49a1e93294288a115f5cd2d014180e4d3684d34ab 211.36MB / 211.36MB            2.9s
 => => sha256:c6b30c3f16966552af10ac00521f60355b1fcfd46ac1c20b1038587e28583ce7 45.68MB / 45.68MB              1.3s
 => => sha256:cda7f44f2bddcc4bb7514474024b3f3705de00ddb6355a33be5ac7808e5b7125 3.32kB / 3.32kB                0.7s
 => => extracting sha256:3e6b9d1a95114e19f12262a4e8a59ad1d1a10ca7b82108adcf0605a200294964                     3.4s
 => => sha256:3697be50c98b9d071df4637e1d3491d00e7b9f3a732768c876d82309b3c5a145 1.25MB / 1.25MB                1.0s
 => => sha256:461077a72fb7fe40d34a37d6a1958c4d16772d0dd77f572ec50a1fdc41a3754d 446B / 446B                    1.0s
 => => extracting sha256:37927ed901b1b2608b72796c6881bf645480268eca4ac9a37b9219e050bb4d84                     0.8s
 => => extracting sha256:79b2f47ad4443652b9b5cc81a95ede249fd976310efdbee159f29638783778c0                     3.1s
 => => extracting sha256:e23f099911d692f62b851cf49a1e93294288a115f5cd2d014180e4d3684d34ab                     8.4s
 => => extracting sha256:cda7f44f2bddcc4bb7514474024b3f3705de00ddb6355a33be5ac7808e5b7125                     0.0s
 => => extracting sha256:c6b30c3f16966552af10ac00521f60355b1fcfd46ac1c20b1038587e28583ce7                     2.6s
 => => extracting sha256:3697be50c98b9d071df4637e1d3491d00e7b9f3a732768c876d82309b3c5a145                     0.1s
 => => extracting sha256:461077a72fb7fe40d34a37d6a1958c4d16772d0dd77f572ec50a1fdc41a3754d                     0.0s
```



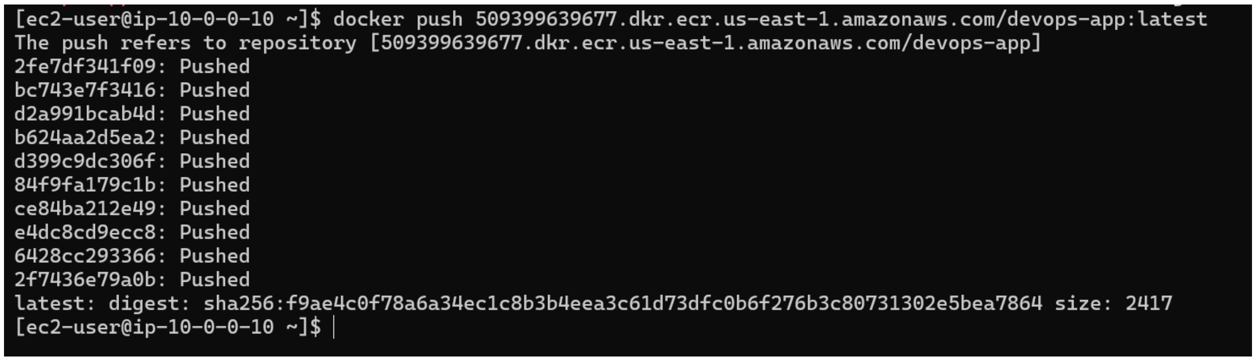
```
 => [internal] load build context
 => => transferring context: 550B
 => [2/3] WORKDIR /app
 => [3/3] COPY . .
 => exporting to image
 => => exporting layers
 => => writing image sha256:4a9dd6b60a69457bae1f146b6e816868e405576f8555907290b2558c8f2ad1bd
 => => naming to docker.io/library/devops-app
[ec2-user@ip-10-0-0-10 devops-app]$ docker images
REPOSITORY   TAG      IMAGE ID       CREATED          SIZE
devops-app   latest   4a9dd6b60a69   44 seconds ago   1.09GB
[ec2-user@ip-10-0-0-10 devops-app]$
```

Step 7: Run Container with Port Mapping



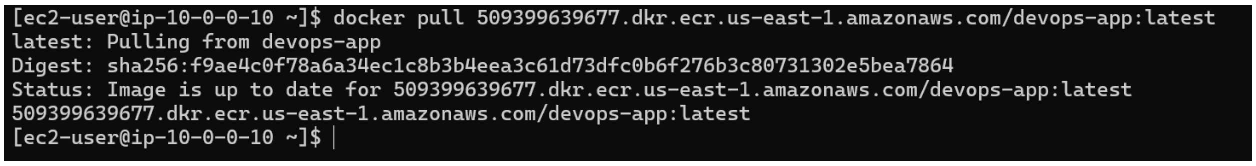
```
 => => naming to docker.io/library/devops-app                                                              0.0s
[ec2-user@ip-10-0-0-10 devops-app]$ docker run -d --name devops-app -p 5000:5000 devops-app
2bf241063ddd2716d8ace5847beabc35124b99b40fbe73e09f747681ef83e11c
[ec2-user@ip-10-0-0-10 devops-app]$ docker ps
CONTAINER ID   IMAGE        COMMAND                  CREATED         STATUS         PORTS                                       NAMES
2bf241063ddd   devops-app   "docker-entrypoint.s…"   8 seconds ago   Up 7 seconds   0.0.0.0:5000->5000/tcp, :::5000->5000/tcp   devop
s-app
[ec2-user@ip-10-0-0-10 devops-app]$ docker logs devops-app
Server running on port 5000
[ec2-user@ip-10-0-0-10 devops-app]$ 
```

Step 8: Push Image to AWS Container Registry (ECR)



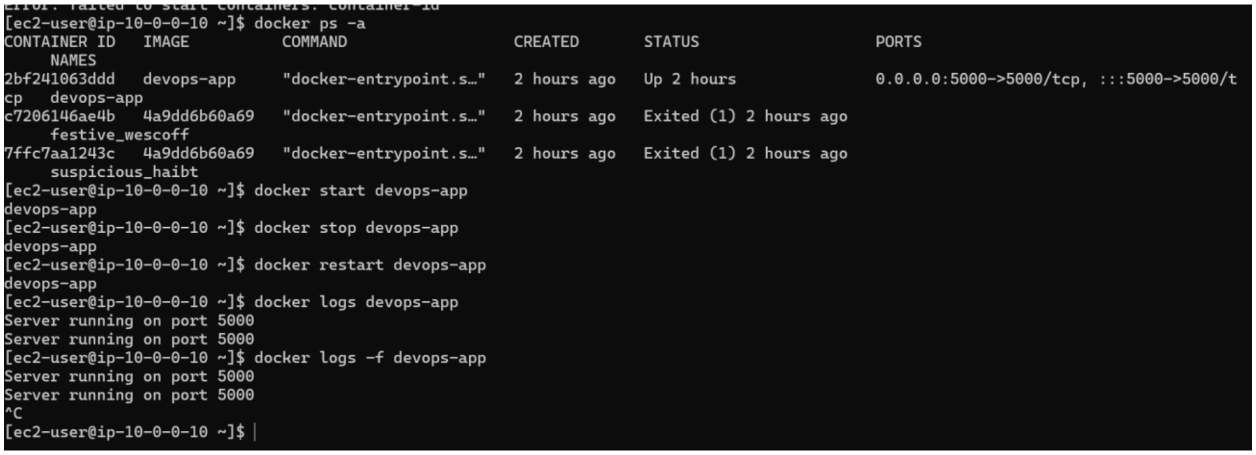
```
[ec2-user@ip-10-0-0-10 ~]$ docker push 509399639677.dkr.ecr.us-east-1.amazonaws.com/devops-app:latest
The push refers to repository [509399639677.dkr.ecr.us-east-1.amazonaws.com/devops-app]
2fe7df341f09: Pushed
bc743e7f3416: Pushed
d2a991bcab4d: Pushed
b624aa2d5ea2: Pushed
d399c9dc306f: Pushed
84f9fa179c1b: Pushed
ce84ba212e49: Pushed
e4dc8cd9ecc8: Pushed
6428cc293366: Pushed
2f7436e79a0b: Pushed
latest: digest: sha256:f9ae4c0f78a6a34ec1c8b3b4eea3c61d73dfc0b6f276b3c80731302e5bea7864 size: 2417
[ec2-user@ip-10-0-0-10 ~]$ 
```

Step 9: Pull Image from Registry



```
[ec2-user@ip-10-0-0-10 ~]$ docker pull 509399639677.dkr.ecr.us-east-1.amazonaws.com/devops-app:latest
latest: Pulling from devops-app
Digest: sha256:f9ae4c0f78a6a34ec1c8b3b4eea3c61d73dfc0b6f276b3c80731302e5bea7864
Status: Image is up to date for 509399639677.dkr.ecr.us-east-1.amazonaws.com/devops-app:latest
509399639677.dkr.ecr.us-east-1.amazonaws.com/devops-app:latest
[ec2-user@ip-10-0-0-10 ~]$ 
```

Step 10: Manage Container Lifecycle



```
Error: failed to start containers: container-id
[ec2-user@ip-10-0-0-10 ~]$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED       STATUS                     PORTS
     NAMES
2bf241063ddd   devops-app     "docker-entrypoint.s…"   2 hours ago   Up 2 hours                 0.0.0.0:5000->5000/tcp, :::5000->5000/t
cp   devops-app
c7206146ae4b   4a9dd6b60a69   "docker-entrypoint.s…"   2 hours ago   Exited (1) 2 hours ago
     festive_wescoff
7ffc7aa1243c   4a9dd6b60a69   "docker-entrypoint.s…"   2 hours ago   Exited (1) 2 hours ago
     suspicious_haibt
[ec2-user@ip-10-0-0-10 ~]$ docker start devops-app
devops-app
[ec2-user@ip-10-0-0-10 ~]$ docker stop devops-app
devops-app
[ec2-user@ip-10-0-0-10 ~]$ docker restart devops-app
devops-app
[ec2-user@ip-10-0-0-10 ~]$ docker logs devops-app
Server running on port 5000
Server running on port 5000
[ec2-user@ip-10-0-0-10 ~]$ docker logs -f devops-app
Server running on port 5000
Server running on port 5000
^C
[ec2-user@ip-10-0-0-10 ~]$ 
```

6. Key DevOps Responsibilities Demonstrated

- Environment standardization
- Automated deployment
- Version-controlled images
- Reproducible builds
- Infrastructure consistency

7. Advantages of Containerized Deployment

- Eliminates "works on my machine" issue
- Faster deployment
- Lightweight compared to VMs
- Easy rollback
- Better scalability
- Simplified CI/CD integration

8. Real-Time Industry Use Cases

- Microservices deployment
- Cloud-native applications
- Dev/Test environment setup
- Continuous Integration pipelines

9. Challenges Faced

- Initial Docker configuration
- Port conflicts
- Image size optimization
- Learning container networking

10. Conclusion

By introducing Docker-based containerization through AWS cloud infrastructure, the deployment process becomes **consistent, reliable, and reproducible** across all environments.

This project reflects real-world DevOps practices such as:

- Standardized deployment

- Container orchestration readiness
- Cloud-based image management
- Production-like service handling

Implementing containerized deployment successfully resolves environment mismatch issues and improves the overall software delivery lifecycle.