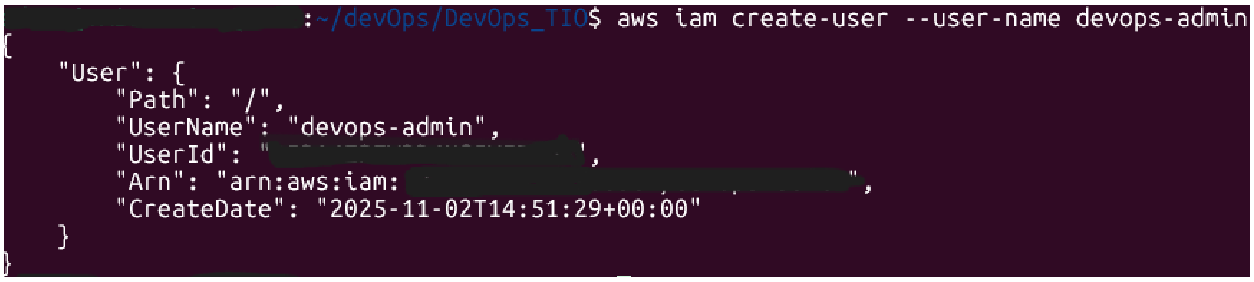
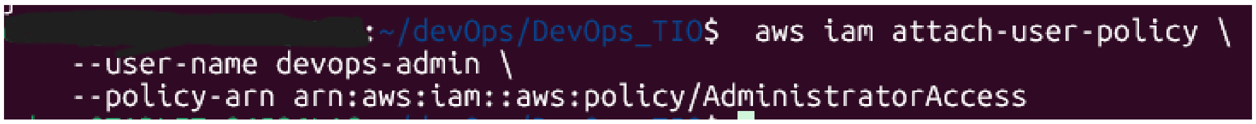
Phase 1: IAM and S3 Setup

I started by setting up Identity and Access Management to handle permissions securely, because you don't want to use root credentials for everything. Using the AWS CLI, I created a new IAM user called devops-admin with admin access. The command was **aws iam create-user –-user-name devops-admin**, and it gave me the user details.



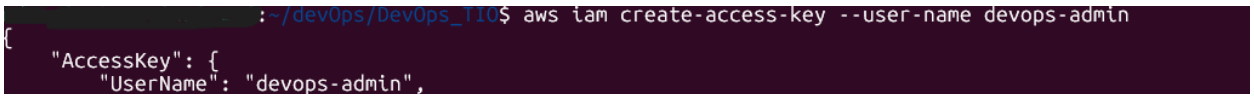
```
[redacted]:~/devOps/DevOps_TIO$ aws iam create-user --user-name devops-admin
{
    "User": {
        "Path": "/",
        "UserName": "devops-admin",
        "UserId": "[redacted]",
        "Arn": "arn:aws:iam:[redacted]",
        "CreateDate": "2025-11-02T14:51:29+00:00"
    }
}
```

Then, I attached the AdministratorAccess policy to it **with aws iam attach-user-policy –user-name devops-admin –policy-arn arn:aws:iam::aws:policy/AdministratorAccess**.



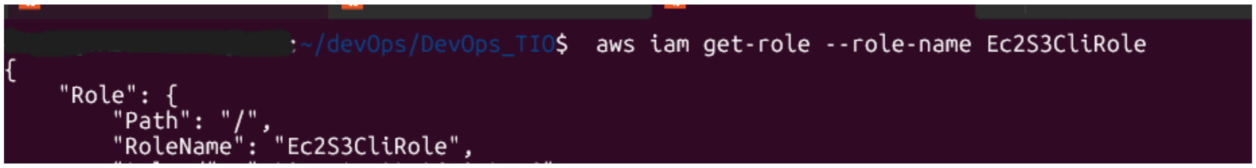
```
[redacted]:~/devOps/DevOps_TIO$  aws iam attach-user-policy \
    --user-name devops-admin \
    --policy-arn arn:aws:iam::aws:policy/AdministratorAccess
```

Next, I generated access keys for this user using **aws iam create-access-key –user-name devops-admin** so I could configure profiles for CLI access.



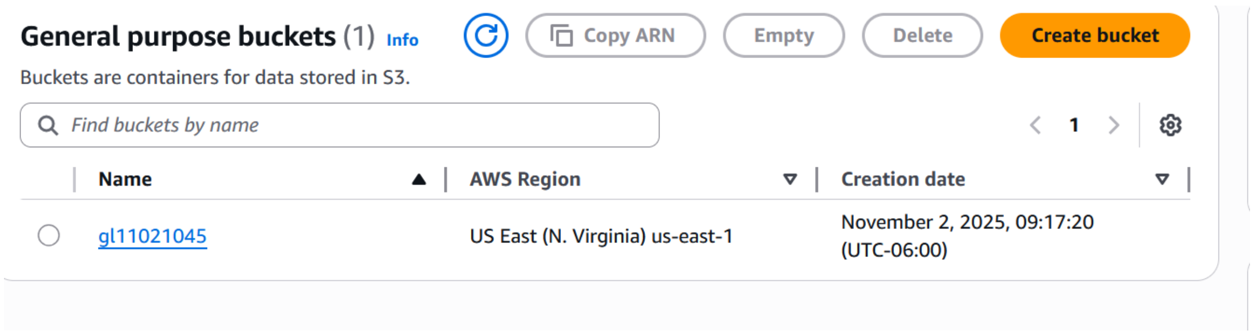
```
[redacted]:~/devOps/DevOps_TIO$ aws iam create-access-key --user-name devops-admin
{
    "AccessKey": {
        "UserName": "devops-admin",
```

After that, I created an IAM role for EC2 instances called Ec2S3CliRole, which allows EC2 to assume the role and get full S3 access. I **used aws iam create-role –role-name Ec2S3CliRole –assume-role-policy-document** with thetrust policy for ec2 and attached the AmazonS3FullAccess policy **via aws iam attach-role-policy –role-name Ec2S3CliRole –policy-arn.**



```
[redacted]:~/devOps/DevOps_TIO$  aws iam get-role --role-name Ec2S3CliRole
{
    "Role": {
        "Path": "/",
        "RoleName": "Ec2S3CliRole",
```

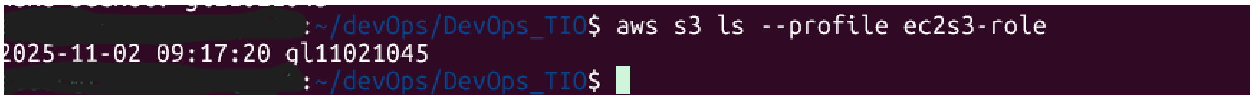
With that role in place, I created an S3 bucket for storing scripts and files. The bucket name was gl11021045. Command: **aws s3 mb s3://gl11021045 –region us-east-1 –profile ec2s3-role.**



Buckets are containers for data stored in S3.

Name	AWS Region	Creation date
gl11021045	US East (N. Virginia) us-east-1	November 2, 2025, 09:17:20 (UTC-06:00)

I checked it with **aws s3 ls –profile ec2s3-role**, and it listed the bucket and date it was created.

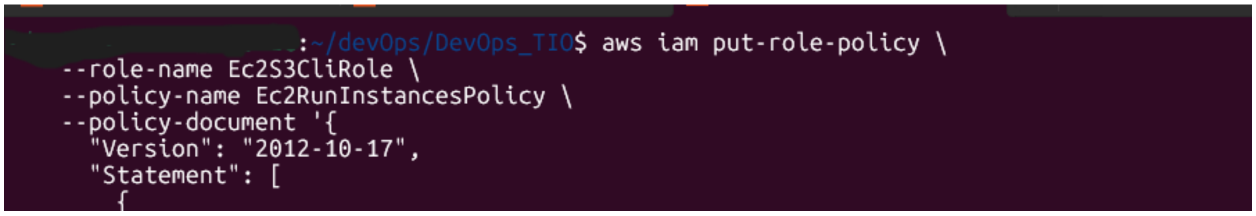


```
[redacted]:~/devOps/DevOps_TIO$ aws s3 ls --profile ec2s3-role
2025-11-02 09:17:20 gl11021045
[redacted]:~/devOps/DevOps_TIO$
```

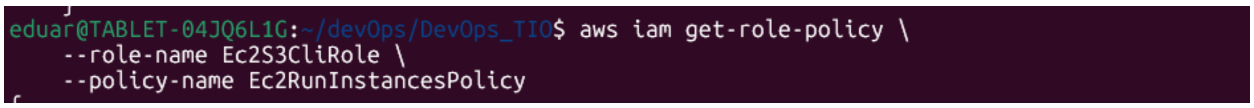
Tools used: AWS CLI for all IAM and S3 operations.

Phase 2: EC2 Permissions & Network Setup

Now I needed to add EC2 permissions to the role so the instance could launch itself and manage resources. I added a custom policy called Ec2RunInstancesPolicy to the Ec2S3Policy to the Ec2S3CliRole using **aws iam put-role-policy --role-name Ec2S3CliRole –policy=name Ec2RunInstancesPolicy –policy-document**. Then veried it with **aws iam get-role-policy –role-name Ec2S3CliRole –policy-name Ec2RunInstancePolicy.**

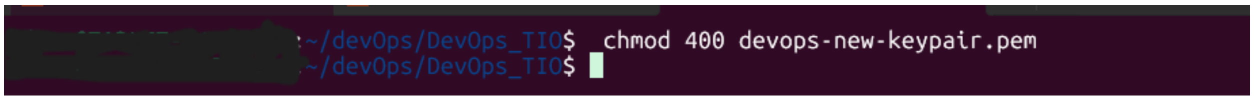


```
[redacted]:~/devOps/DevOps_TIO$ aws iam put-role-policy \
    --role-name Ec2S3CliRole \
    --policy-name Ec2RunInstancesPolicy \
    --policy-document '{
      "Version": "2012-10-17",
      "Statement": [
        {
```

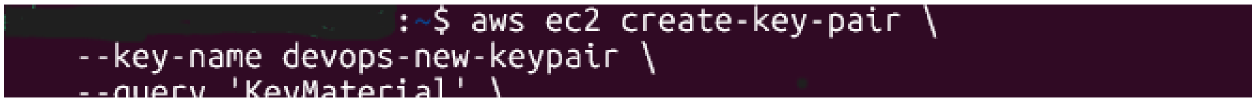


```
eduar@TABLET-04JQ6L1G:~/devOps/DevOps_TIO$ aws iam get-role-policy \
    --role-name Ec2S3CliRole \
    --policy-name Ec2RunInstancesPolicy
```

For SSH access, I set permissions on my key pair file with **chmod 400 devops-new-keypair.pem.** Then, I created the key pair in AWS: **aws ec2 create-key-pair –key-name –etc…**

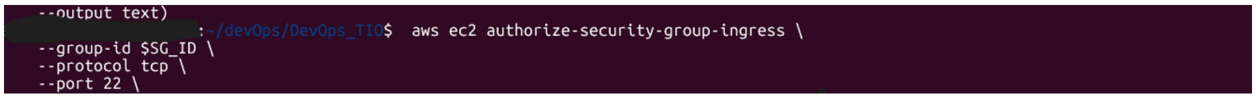


```
[redacted]:~/devOps/DevOps_TIO$  chmod 400 devops-new-keypair.pem
[redacted]~/devOps/DevOps_TIO$
```

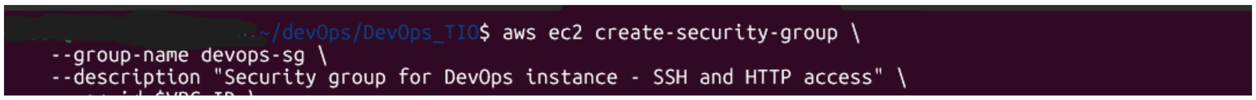


```
[redacted]:~$ aws ec2 create-key-pair \
    --key-name devops-new-keypair \
    --query 'KeyMaterial' \
```

I also created a security group for the EC2 instance called devops-sg, allowing SSH and HTTP. Command: **aws ec2 create-security-group --group-name devops-sg --description "Security group for DevOps instance - SSH and HTTP access" --vpc-id $VPC_ID --region us-east-1 --profile ec2s3-role.** Then, I authorized ingress for port 22 (SSH) **with aws ec2 authorize-security-group-ingress --group-id $SG_ID --protocol tcp --port 22 --cidr 0.0.0.0/0 --region us-east-1 --profile ec2s3-role,** and similarly for port 80 (HTTP).



```
  --output text)
[redacted]:~/devOps/DevOps_TIO$  aws ec2 authorize-security-group-ingress \
  --group-id $SG_ID \
  --protocol tcp \
  --port 22 \
```

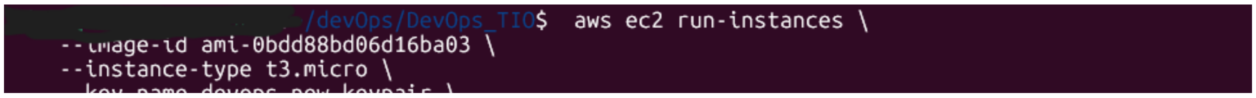


```
[redacted]~/devOps/DevOps_TIO$ aws ec2 create-security-group \
  --group-name devops-sg \
  --description "Security group for DevOps instance - SSH and HTTP access" \
```

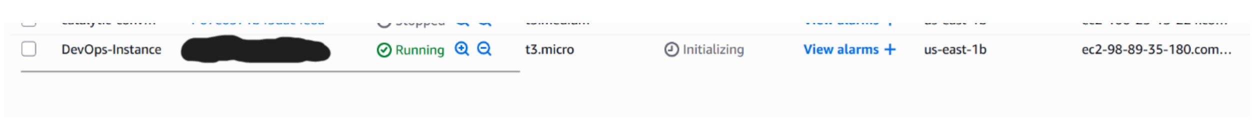
Tools used: AWS CLI for ec2 configs, chmod for file permissions.

Phase 3: EC2 Instance Launch

With permissions ready, I launched the EC2 instance using **aws ec2 run-instances….**



```
[redacted]/devOps/DevOps_TIO$  aws ec2 run-instances \
  --image-id ami-0bdd88bd06d16ba03 \
  --instance-type t3.micro \
```



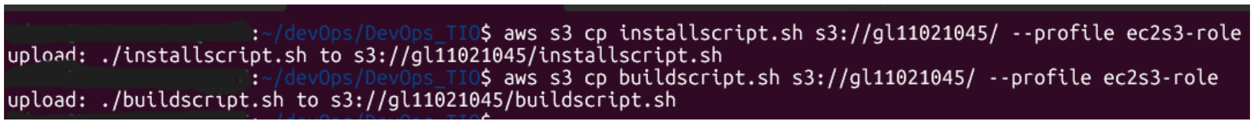
☐	DevOps-Instance	[redacted]	Running	t3.micro	Initializing	View alarms +	us-east-1b	ec2-98-89-35-180.com…

Phase 4: Connect to EC2 & Setup

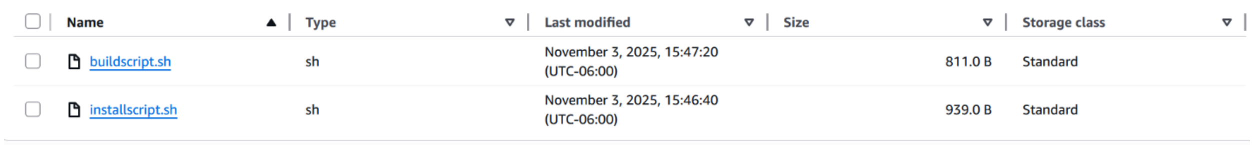
I connected to the EC2 instance via SSH: **ssh -I <keypair> ec2-user@<public ip>.** It was Amazon Linux 2023.

```
   ,     #_
   ~\_  ####_
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-[redacted]
```

Then, from my local machine, I uploaded scripts to S3: **aws s3 cp installscript.sh s3://gl11021045/ --profile ec2s3-role** and same for buildscript.sh. Verified with **aws s3 ls s3://gl11021045/ --profile ec2s3-role**, showing the files uploaded.



```
[redacted]:~/devOps/DevOps_TIO$ aws s3 cp installscript.sh s3://gl11021045/ --profile ec2s3-role
upload: ./installscript.sh to s3://gl11021045/installscript.sh
[redacted]:~/devOps/DevOps_TIO$ aws s3 cp buildscript.sh s3://gl11021045/ --profile ec2s3-role
upload: ./buildscript.sh to s3://gl11021045/buildscript.sh
```

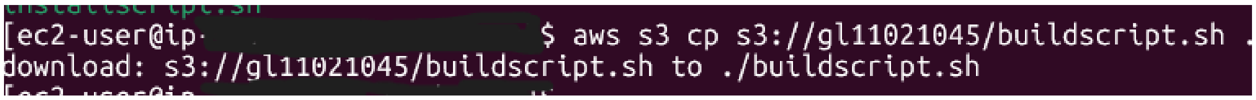


Name	Type	Last modified	Size	Storage class
buildscript.sh	sh	November 3, 2025, 15:47:20 (UTC-06:00)	811.0 B	Standard
installscript.sh	sh	November 3, 2025, 15:46:40 (UTC-06:00)	939.0 B	Standard

Tools used: SSH for connection, AWS CLI for S3 uploads.

Phase 5: Download Scripts & Execute.

On the EC2 instance, I downloaded the scripts from S3: aws s3 cp s3://gl11021045/installscript.sh . and same for buildscript.sh. Checked with ls.



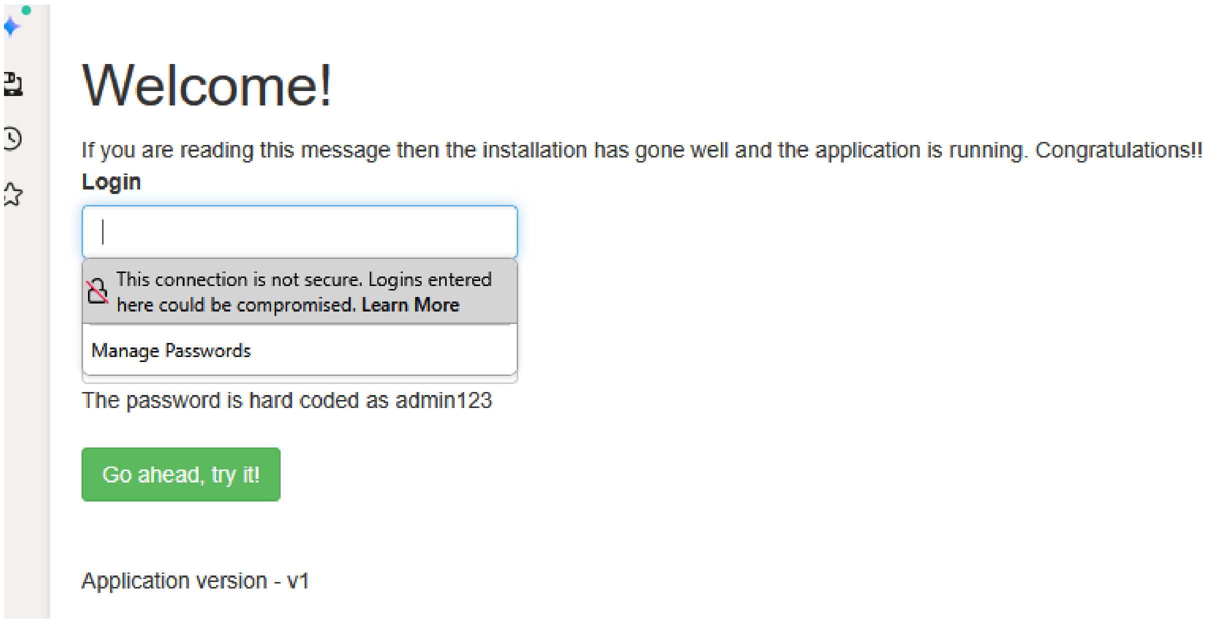
```
[ec2-user@ip-[redacted] $ aws s3 cp s3://gl11021045/buildscript.sh .
download: s3://gl11021045/buildscript.sh to ./buildscript.sh
```

Ran **bash installcript.sh**, which installed Java OpenJDK 11, Maven 3.9.4, downloaded the HelloWorld source code, and built the WAR file, Then **bash buildscritp.sh** to create Docker image helloworld:v1 and start a container mapping port 80 to 8080.

Tools used: AWS CLI for downloads, Bash for scripts, Maven for building, Docker CLI for image and container.

Phase 6: Verification and Success:

I accessed the app in my browser at http://<public-ip>, tested login, and checked the container with **sudo docker ps,** showing it running.



Welcome!

If you are reading this message then the installation has gone well and the application is running. Congratulations!!

Login

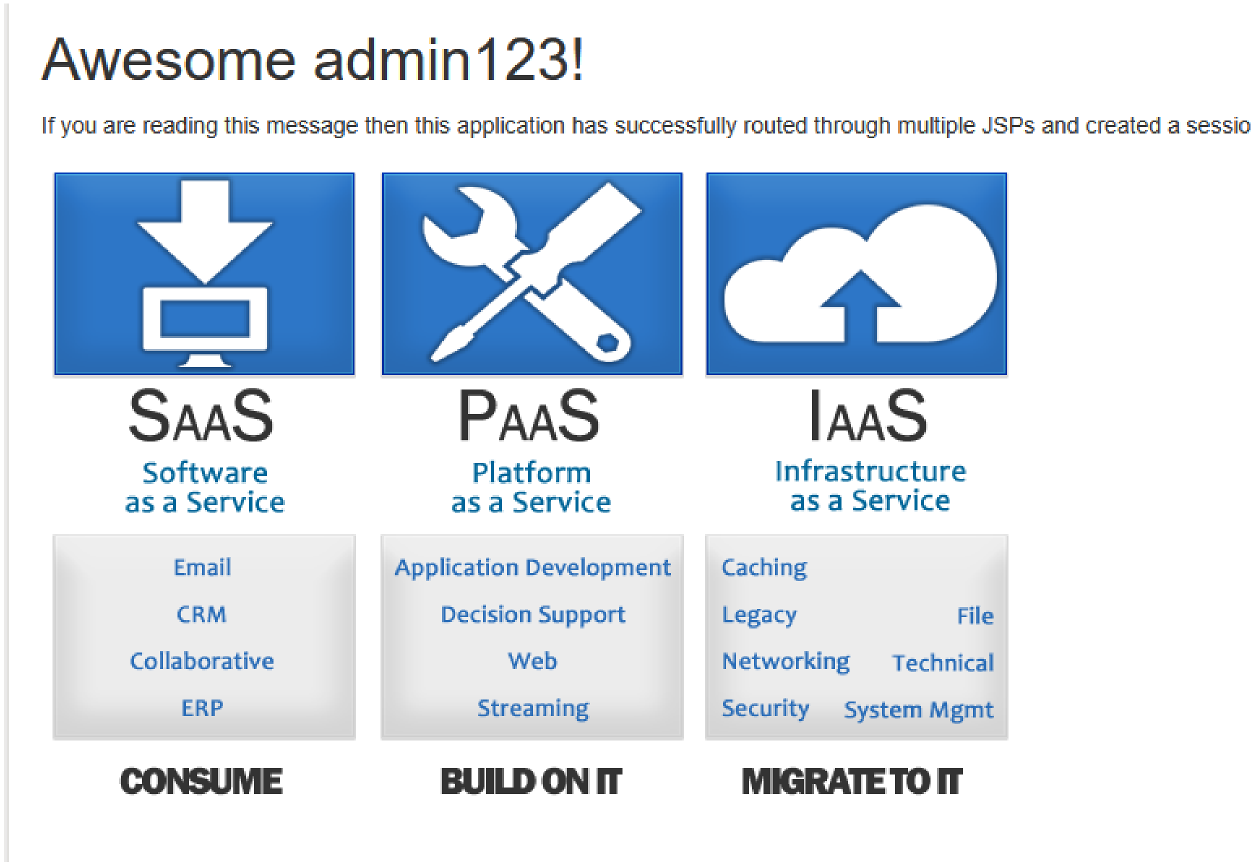
This connection is not secure. Logins entered here could be compromised. **Learn More**

Manage Passwords

The password is hard coded as admin123

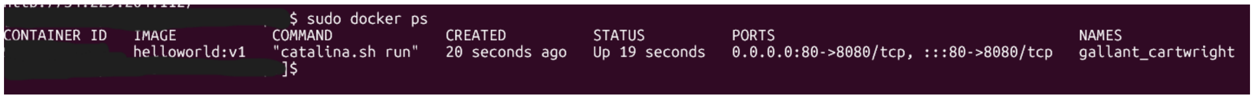
Go ahead, try it!

Application version - v1



Awesome admin123!

If you are reading this message then this application has successfully routed through multiple JSPs and created a sessio

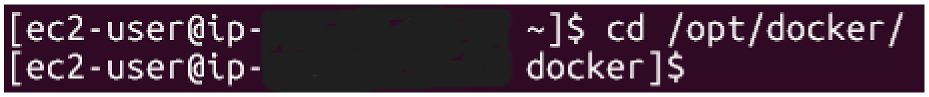


```
[redacted] $ sudo docker ps
CONTAINER ID   IMAGE           COMMAND              CREATED          STATUS          PORTS                                   NAMES
[redacted]     helloworld:v1   "catalina.sh run"    20 seconds ago   Up 19 seconds   0.0.0.0:80->8080/tcp, :::80->8080/tcp   gallant_cartwright
[redacted] ]$
```

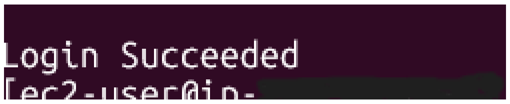
Tools used: Mozilla Firefox for testing, Docker CLI for verification.

Phase 7: Push Docker Image to ECR

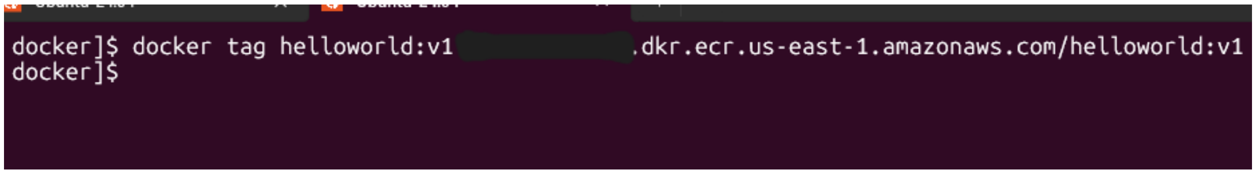
On EC2, I cd’d to /opt/docker, authenticated Docker to ECR **with aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin .dkr.ecr.us-east-1.amazonaws.com.** Tagged the image and pushed it. Verified ECR.



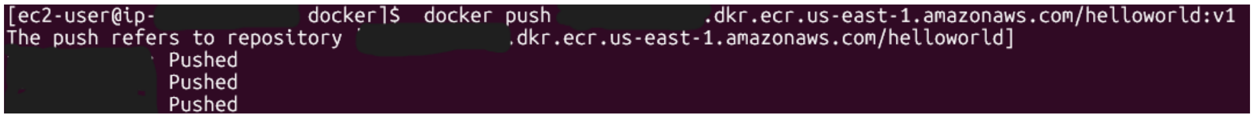
```
[ec2-user@ip-[redacted] ~]$ cd /opt/docker/
[ec2-user@ip-[redacted] docker]$
```



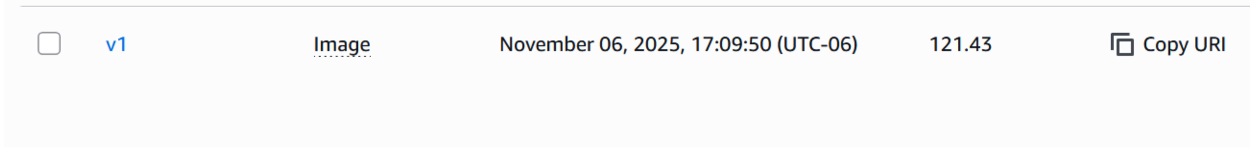
```
Login Succeeded
[ec2-user@ip-[redacted]
```



```
docker]$ docker tag helloworld:v1 [redacted].dkr.ecr.us-east-1.amazonaws.com/helloworld:v1
docker]$
```



```
[ec2-user@ip-[redacted] docker]$  docker push [redacted].dkr.ecr.us-east-1.amazonaws.com/helloworld:v1
The push refers to repository [[redacted].dkr.ecr.us-east-1.amazonaws.com/helloworld]
[redacted] Pushed
            Pushed
            Pushed
```



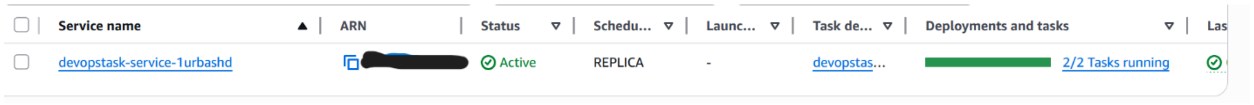
☐	v1	Image	November 06, 2025, 17:09:50 (UTC-06)	121.43	Copy URI

Phase 8: Configure IAM Permissions

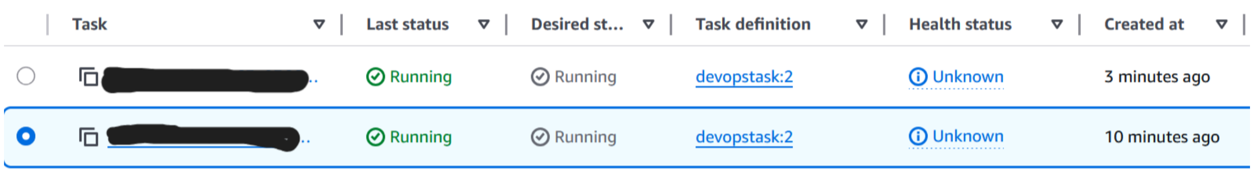
Added ECS permissions to the role with put-role-policy for iam actions and attached AmazonECS_FullAccess.

Phase 9: Create ECS Infrastructure

Created ECS cluster, task definition, service with 2 tasks, load balancer.



☐	Service name	ARN	Status	Schedu...	Launc...	Task de...	Deployments and tasks	Las
☐	devopstask-service-1urbashd	[redacted]	✓ Active	REPLICA	-	devopstas...	🟩 2/2 Tasks running	✓

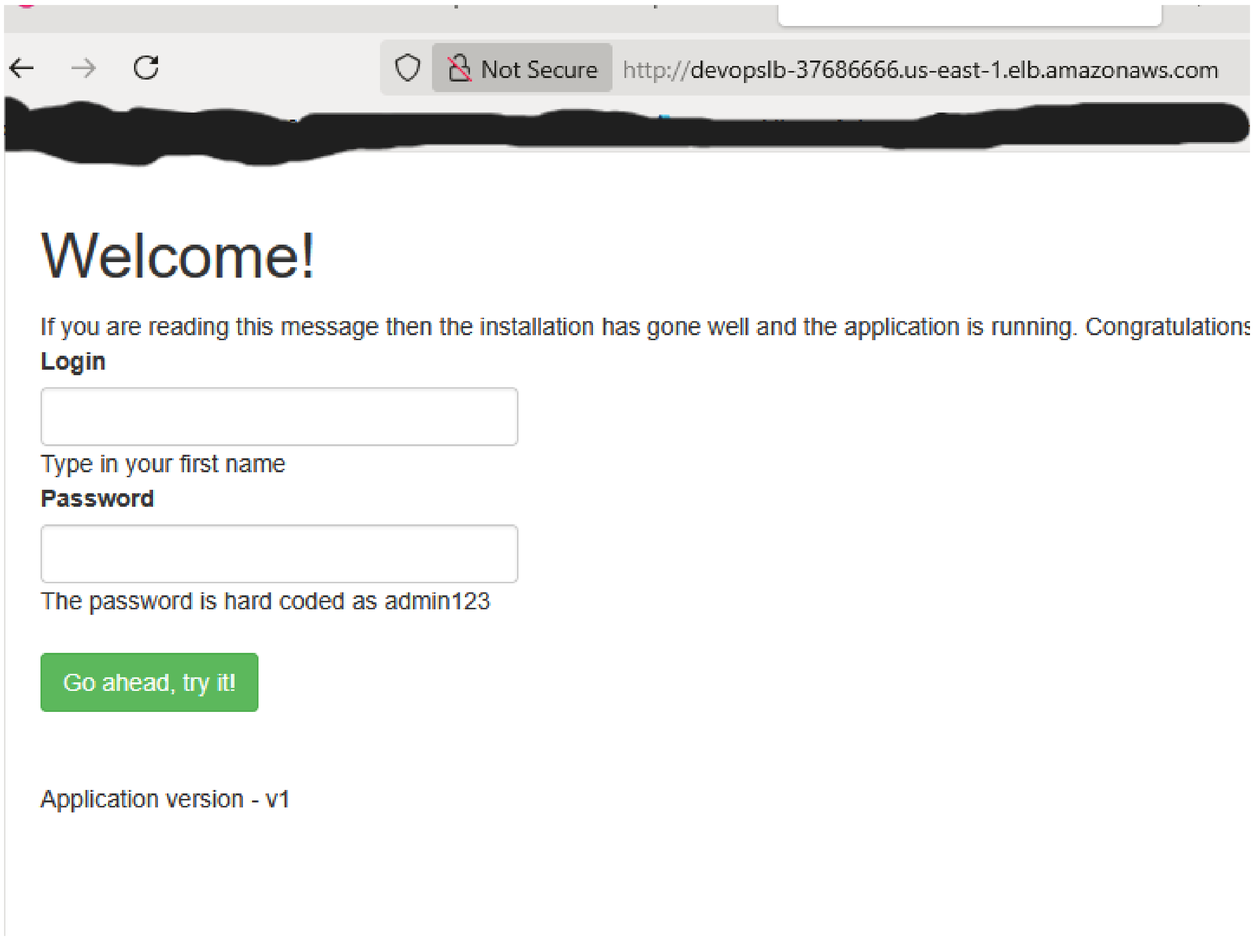


	Task	Last status	Desired st...	Task definition	Health status	Created at
○	[redacted]..	✓ Running	✓ Running	devopstask:2	ⓘ Unknown	3 minutes ago
●	[redacted]..	✓ Running	✓ Running	devopstask:2	ⓘ Unknown	10 minutes ago

Tools used: AWS CLI and Console for ECS Setup.

Phase 9: Access Application

Accessed via load balancer DNS, tested login, checked health.



Not Secure http://devopslb-37686666.us-east-1.elb.amazonaws.com

[redacted]

Welcome!

If you are reading this message then the installation has gone well and the application is running. Congratulations

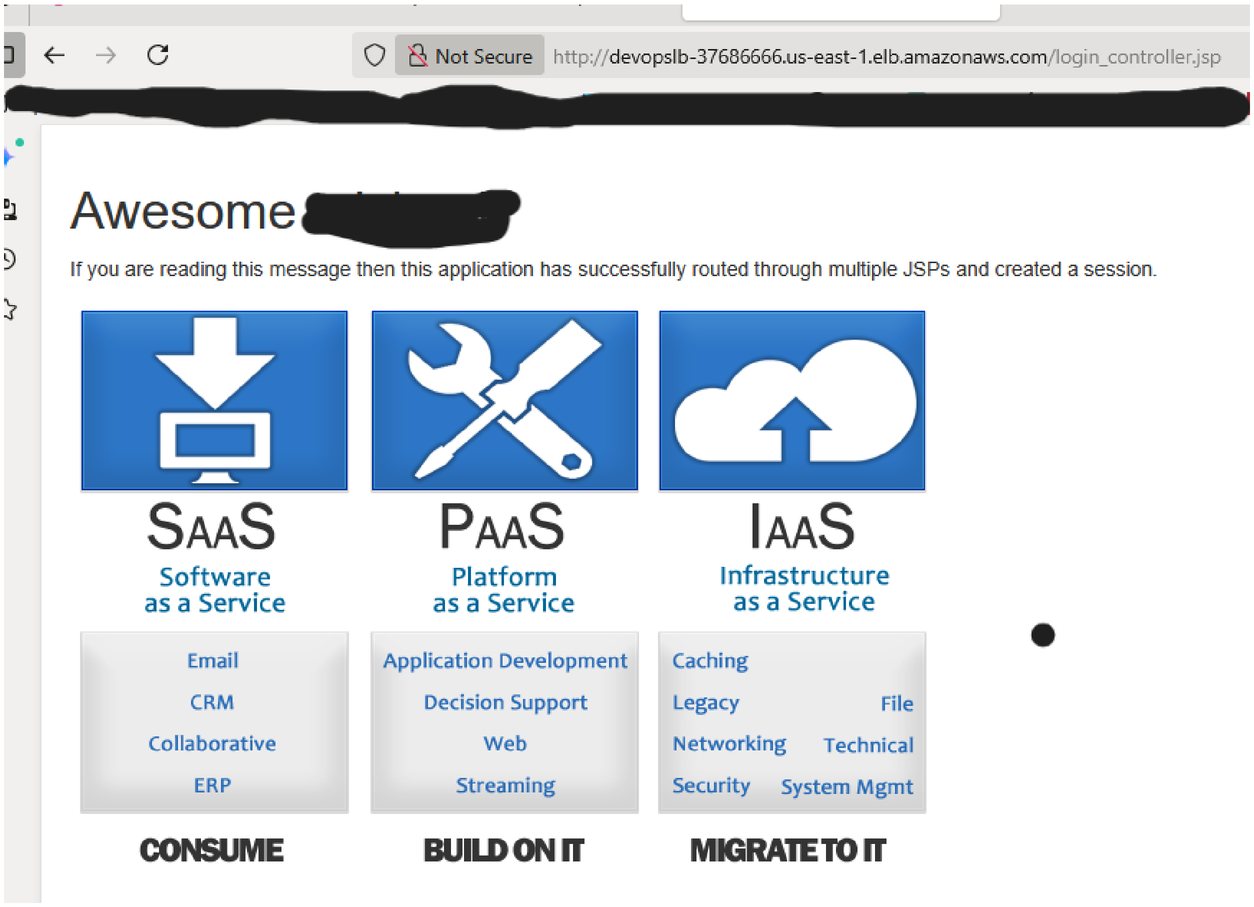
Login

Type in your first name

Password

The password is hard coded as admin123

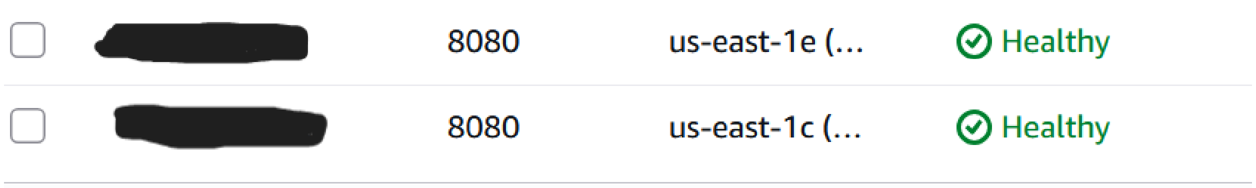
Application version - v1


Not Secure http://devopslb-37686666.us-east-1.elb.amazonaws.com/login_controller.jsp

[redacted]

Awesome [redacted]

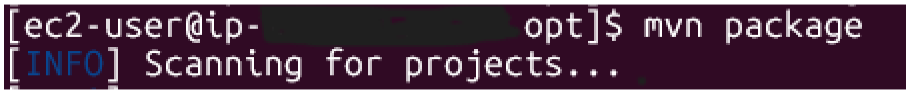
If you are reading this message then this application has successfully routed through multiple JSPs and created a session.



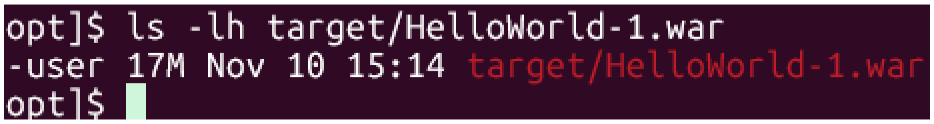
☐	[redacted]	8080	us-east-1e (…	✓ Healthy
☐	[redacted]	8080	us-east-1c (…	✓ Healthy

Phase 10: Update Application Code and Rebuild on EC2

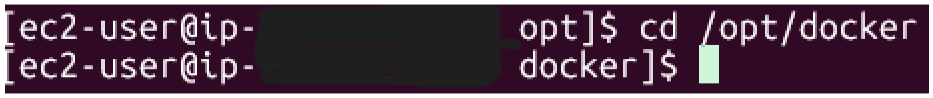
Edited index.jsp to v2, rebuilt with Maven, copied WAR to docker dir.



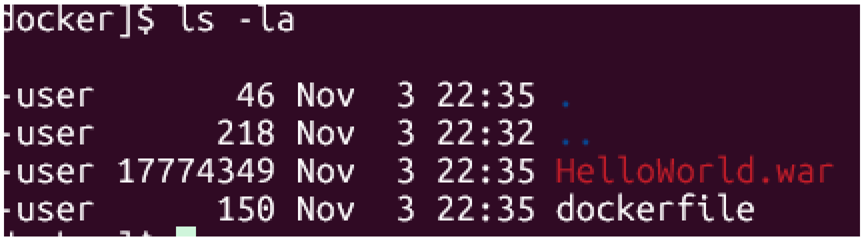
```
[ec2-user@ip-[redacted] opt]$ mvn package
[INFO] Scanning for projects...
```



```
opt]$ ls -lh target/HelloWorld-1.war
-user 17M Nov 10 15:14 target/HelloWorld-1.war
opt]$
```



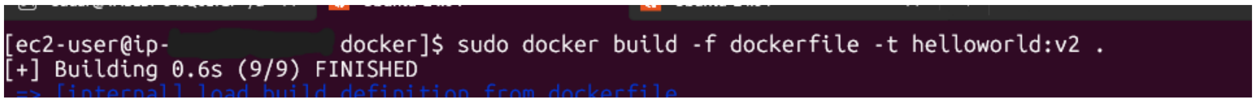
```
[ec2-user@ip-[redacted] opt]$ cd /opt/docker
[ec2-user@ip-[redacted] docker]$
```



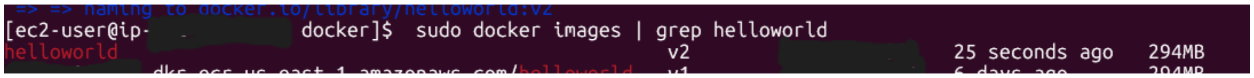
```
docker]$ ls -la

user        46 Nov  3 22:35 .
user       218 Nov  3 22:32 ..
user 17774349 Nov  3 22:35 HelloWorld.war
user       150 Nov  3 22:35 dockerfile
```

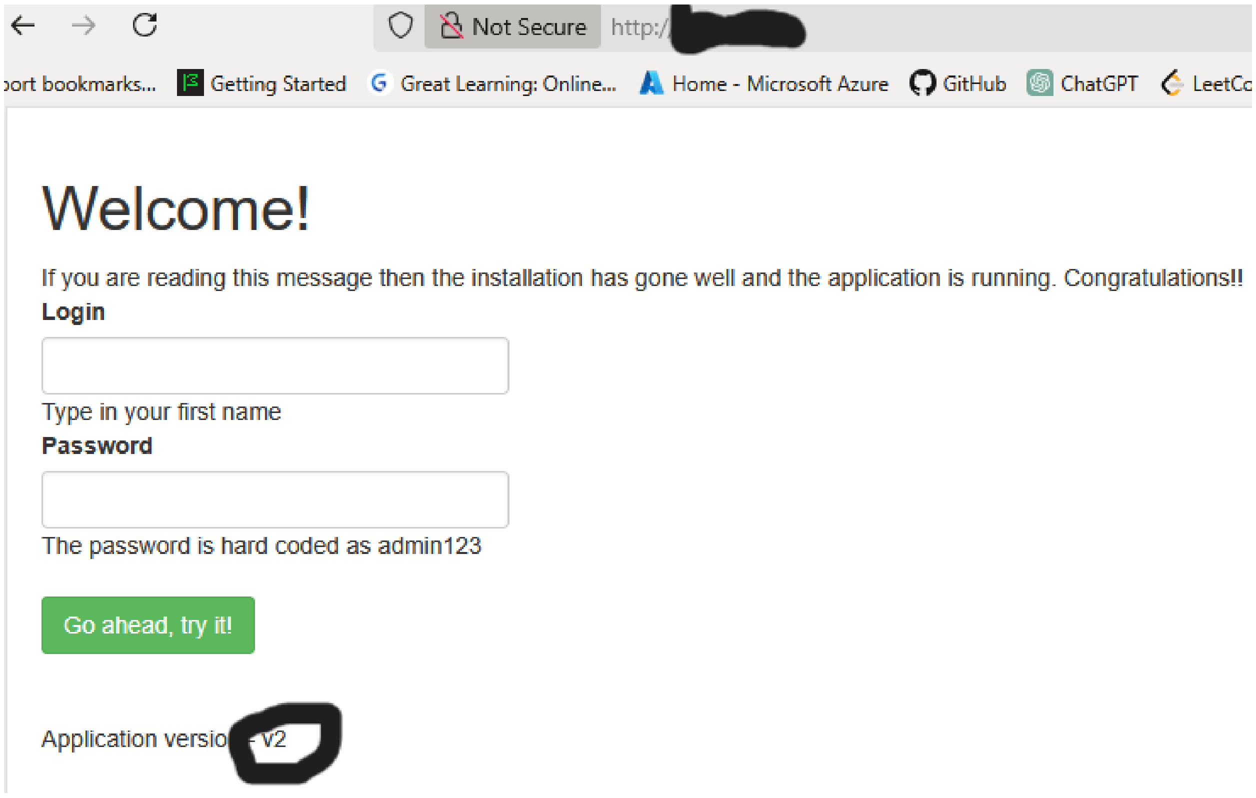
Phase 11: Build and Test Docker Image



```
[ec2-user@ip-[redacted] docker]$ sudo docker build -f dockerfile -t helloworld:v2 .
[+] Building 0.6s (9/9) FINISHED
 => [internal] load build definition from dockerfile
```



```
 => => naming to docker.io/library/helloworld:v2
[ec2-user@ip-[redacted] docker]$  sudo docker images | grep helloworld
helloworld                                          v2      [redacted]      25 seconds ago   294MB
[redacted]  dkr.ecr.us-east-1.amazonaws.com/helloworld   v1                  6 days ago       294MB
```



Not Secure http:/[redacted]

Welcome!

If you are reading this message then the installation has gone well and the application is running. Congratulations!!

Login

Type in your first name

Password

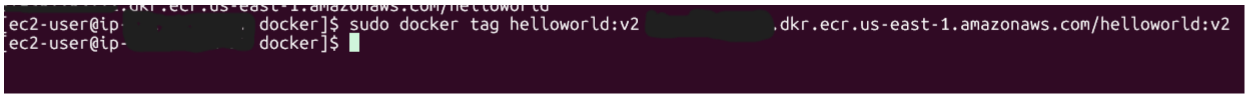
The password is hard coded as admin123

Go ahead, try it!

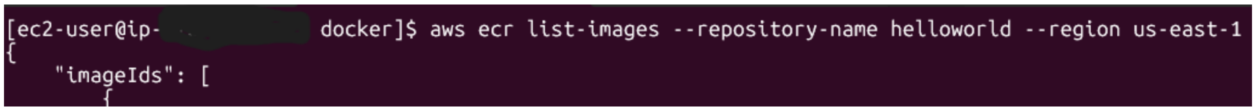
Application version v2

Phase 12: Push to ECR

Authenticated, tagged, pushed v2, verified.



```
[redacted].dkr.ecr.us-east-1.amazonaws.com/helloworld
[ec2-user@ip-[redacted] docker]$ sudo docker tag helloworld:v2 [redacted].dkr.ecr.us-east-1.amazonaws.com/helloworld:v2
[ec2-user@ip-[redacted] docker]$
```



```
[ec2-user@ip-[redacted] docker]$ aws ecr list-images --repository-name helloworld --region us-east-1
{
    "imageIds": [
        {
```

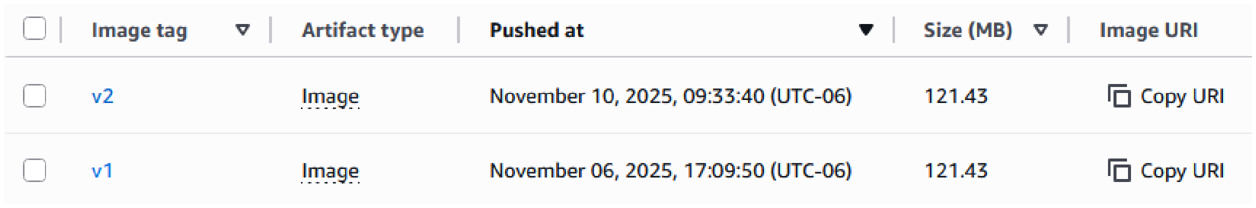
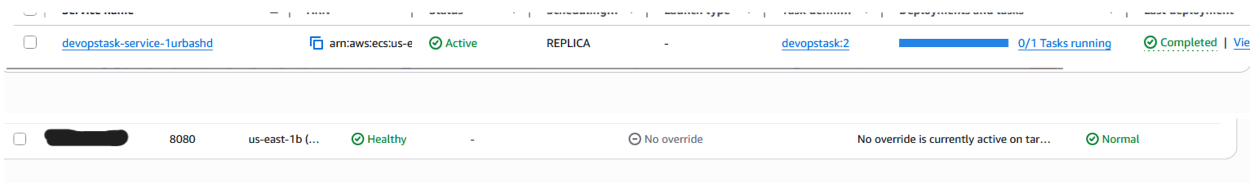


	Image tag	Artifact type	Pushed at	Size (MB)	Image URI
☐	v2	Image	November 10, 2025, 09:33:40 (UTC-06)	121.43	Copy URI
☐	v1	Image	November 06, 2025, 17:09:50 (UTC-06)	121.43	Copy URI

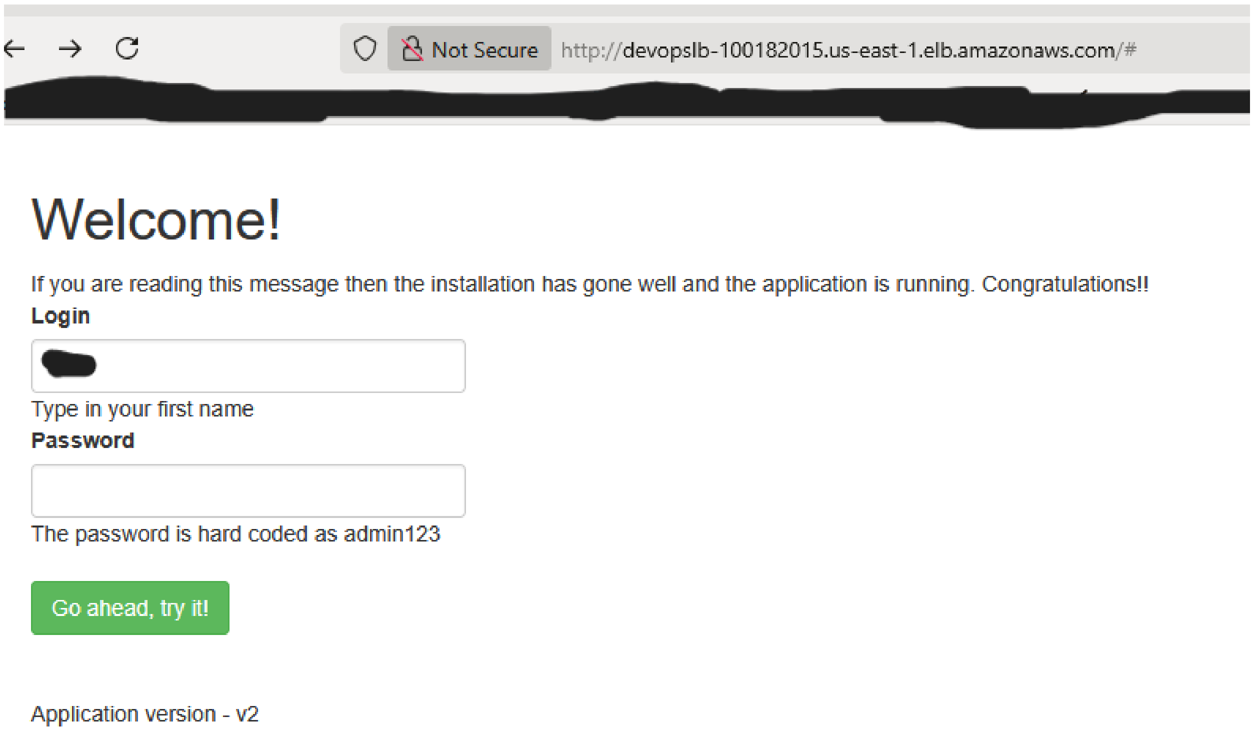
Phase 13: Update ECS Service

Created new task definition revision for v2, recreated load balancer infrastructure, updated service, verified v2 in browser.



☐	devopstask-service-1urbashd	arn:aws:ecs:us-e	Active	REPLICA	-	devopstask:2	0/1 Tasks running	Completed | Vie

☐	[redacted]	8080	us-east-1b (...	Healthy	-	No override	No override is currently active on tar...	Normal



Not Secure http://devopslb-100182015.us-east-1.elb.amazonaws.com/#

[redacted]

Welcome!

If you are reading this message then the installation has gone well and the application is running. Congratulations!!

Login

[redacted]

Type in your first name

Password

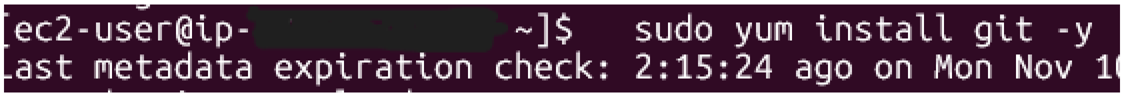
The password is hard coded as admin123

Go ahead, try it!

Application version - v2

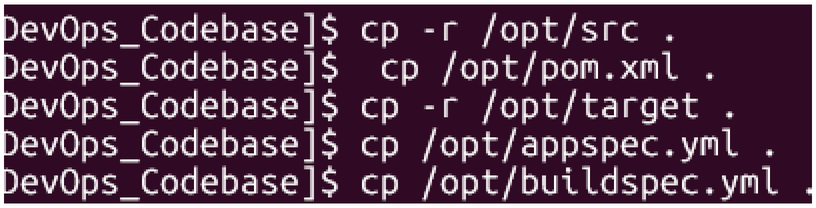
Phase 14: Install and Configure Git on EC2

Installed Git with yum, configured user.name, email, default branch.

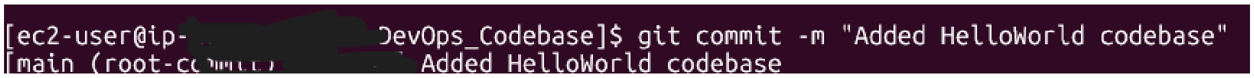


```
[ec2-user@ip-[redacted] ~]$   sudo yum install git -y
ast metadata expiration check: 2:15:24 ago on Mon Nov 1
```

Phase 15: Clone GitHub Repository & Copy Application Files

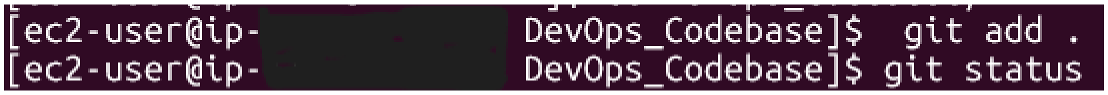


```
DevOps_Codebase]$ cp -r /opt/src .
DevOps_Codebase]$  cp /opt/pom.xml .
DevOps_Codebase]$ cp -r /opt/target .
DevOps_Codebase]$ cp /opt/appspec.yml .
DevOps_Codebase]$ cp /opt/buildspec.yml .
```

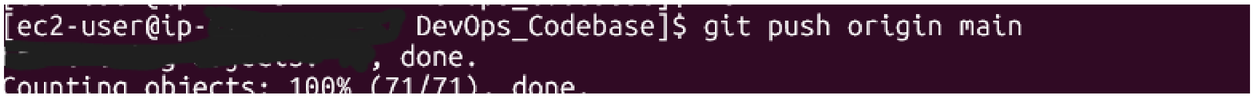


```
[ec2-user@ip-[redacted]DevOps_Codebase]$ git commit -m "Added HelloWorld codebase"
[main (root-co[redacted] Added HelloWorld codebase
```

Phase 16: Stage Files with Git



```
[ec2-user@ip-[redacted] DevOps_Codebase]$  git add .
[ec2-user@ip-[redacted] DevOps_Codebase]$ git status
```



```
[ec2-user@ip-[redacted] DevOps_Codebase]$ git push origin main
[redacted], done.
Counting objects: 100% (71/71), done.
```

Phase 17: Commit and Push to Github.

Overall, this project taught me a ton about AWS services integration, containerization, and CI/CD foundations. I used AWS CLI extensively for automation, Docker for app packaging, Maven for builds, Git for code management, and the AWS console for some visual verifications. If I had to do it again, I'd script more of it in a pipeline.

References:

- PGP in Cloud Computing - Great Learning
  - DevOps Training in Operations (TIO) course materials

Source Code Repository

- HelloWorld Java Application
  - GitHub: https://github.com/pbharadwaj1608/helloworld/raw/main/HelloWorld.zip

- Author: pbharadwaj1608

Software & Tools

Apache Maven

- Version: 3.9.4
- Download URL: https://dlcdn.apache.org/maven/maven-3/3.9.4/binaries/apache-maven-3.9.4-bin.tar.gz
- Official site: https://maven.apache.org/

Java OpenJDK

- Version: OpenJDK 11
- Installed via: Amazon Linux Extras

Apache Tomcat

- Docker Image: tomcat:jre11 (from Docker Hub)
- Official site: https://tomcat.apache.org/

Docker

- Official site: https://www.docker.com/
- Documentation: https://docs.docker.com/

Git & git-remote-codecommit

- PyPI package: git-remote-codecommit
- Official Git: https://git-scm.com/

AWS Services Documentation

Amazon Web Services (AWS)

- Region: us-east-1 (N. Virginia)
- Services used:
  - EC2 (Elastic Compute Cloud)
  - S3 (Simple Storage Service)
  - ECR (Elastic Container Registry)
  - ECS (Elastic Container Service) - Fargate launch type
  - ALB (Application Load Balancer)
  - IAM (Identity and Access Management)
  - CodeCommit (Git-based source control)

AWS Documentation URLs:

- AWS CLI: https://aws.amazon.com/cli/
- Amazon ECR: https://docs.aws.amazon.com/ecr/
- Amazon ECS: https://docs.aws.amazon.com/ecs/
- AWS CodeCommit: https://docs.aws.amazon.com/codecommit/

External Services

OpenDNS (for IP detection)

- Service: myip.opendns.com via resolver1.opendns.com
- Used in buildscript.sh:14 to detect EC2 public IP address