

Production-Grade CI/CD Pipeline

GitHub → Jenkins → AWS SSM → EC2 Worker → Docker → ECR → ECS Fargate → ALB → Route53 → End User

Key Design Principles:

- No SSH access to servers
- Jenkins does NOT use worker nodes as build agents
- Jenkins triggers deployment using AWS SSM RunCommand
- No SSH
- No passwords in Jenkins
- No static AWS keys
- IAM Roles for:
 - Jenkins EC2
 - Worker EC2
- SSM for all remote execution
- GitHub PAT stored in Secrets Manager
- Worker fetches secrets securely at runtime

Architecture satisfies several PCI DSS (Payment Card Industry) and SOC2 frame.

- PCI Requirement 1: Install and maintain a firewall configuration (Achieved via Security Groups & No SSH).
- PCI Requirement 7: Restrict access to data by business need-to-know (Achieved via IAM Roles and Secrets Manager).
- SOC2 Trust Principle (Security): The system is protected against unauthorized access (Achieved via SSM and Zero-Trust logic).

Components Used

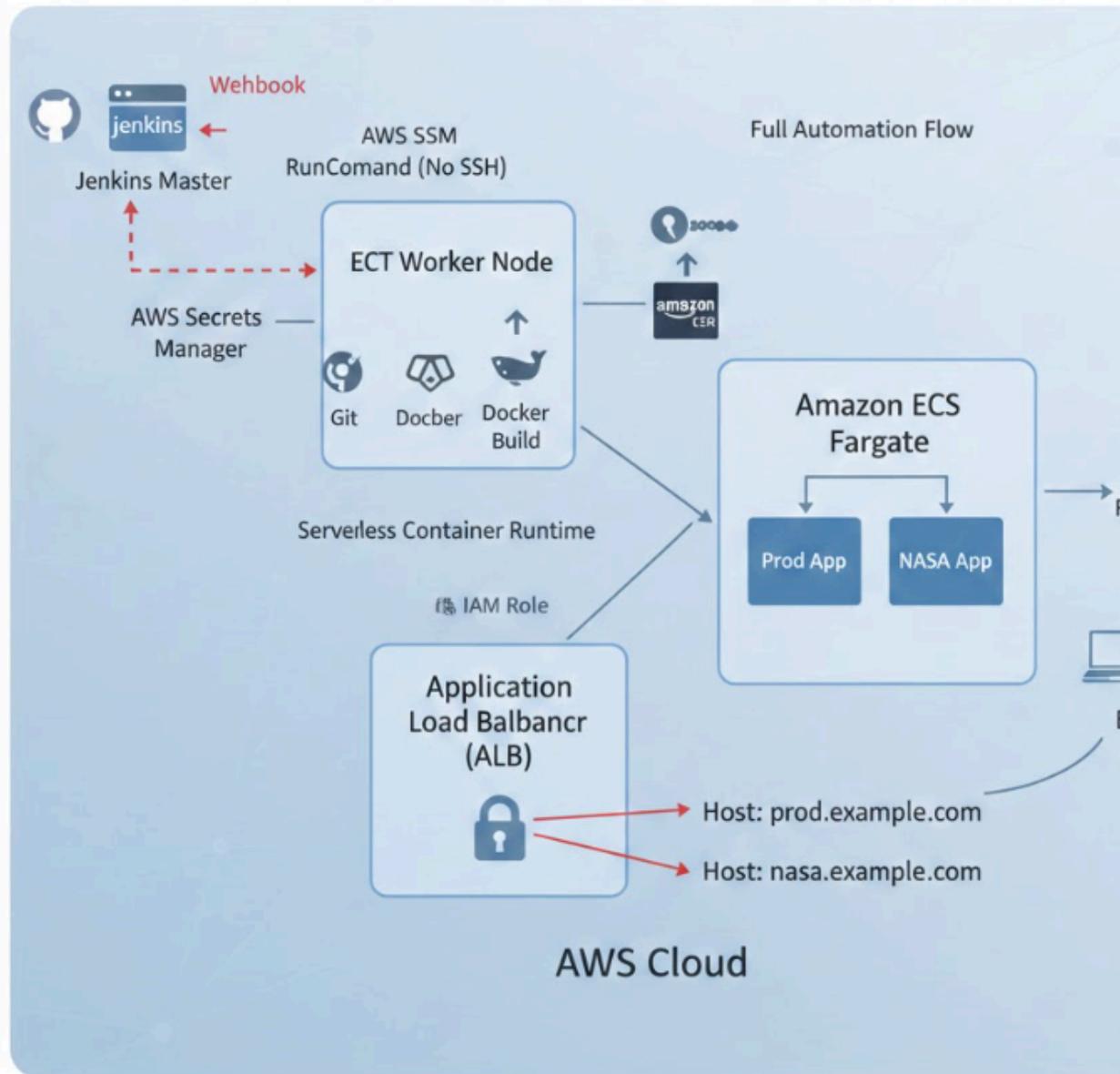
- GitHub – Source control
- Jenkins – CI/CD orchestrator
- AWS SSM – Secure remote command execution
- Docker – Containerization
- Amazon ECR – Container registry
- Amazon ECS (Fargate) – Container runtime
- AWS Secrets Manager – Secure GitHub PAT storage
- IAM Roles – No static credentials anywhere
- ALB
- Route53
- ACM
- EC2 Worker Node

Full Automation Flow

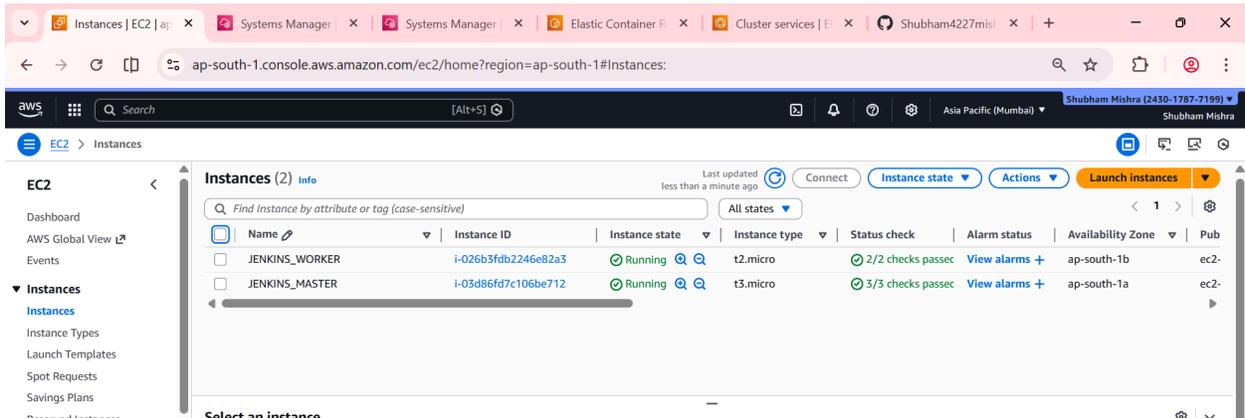
- 1) Developer pushes code to GitHub
- 2) GitHub Webhook triggers Jenkins
- 3) Jenkins pulls Jenkinsfile
- 4) Jenkins analyzes which service changed (monorepo logic)
- 5) Jenkins calls AWS SSM RunCommand
- 6) SSM executes `deploy.sh` on Worker EC2
- 7) Worker:
 - a) Pulls latest code

- b) Builds new Docker image (no cache)
- c) Tags with timestamp version
- d) Pushes image to ECR
- e) Fetches current ECS task definition
- f) Updates image using `jq`
- g) Registers new task revision
- h) Updates ECS service
- i) Waits for service to stabilize
- 8) ECS performs rolling update
- 9) ALB continues serving traffic with zero downtime
- 10) Users see new version live.

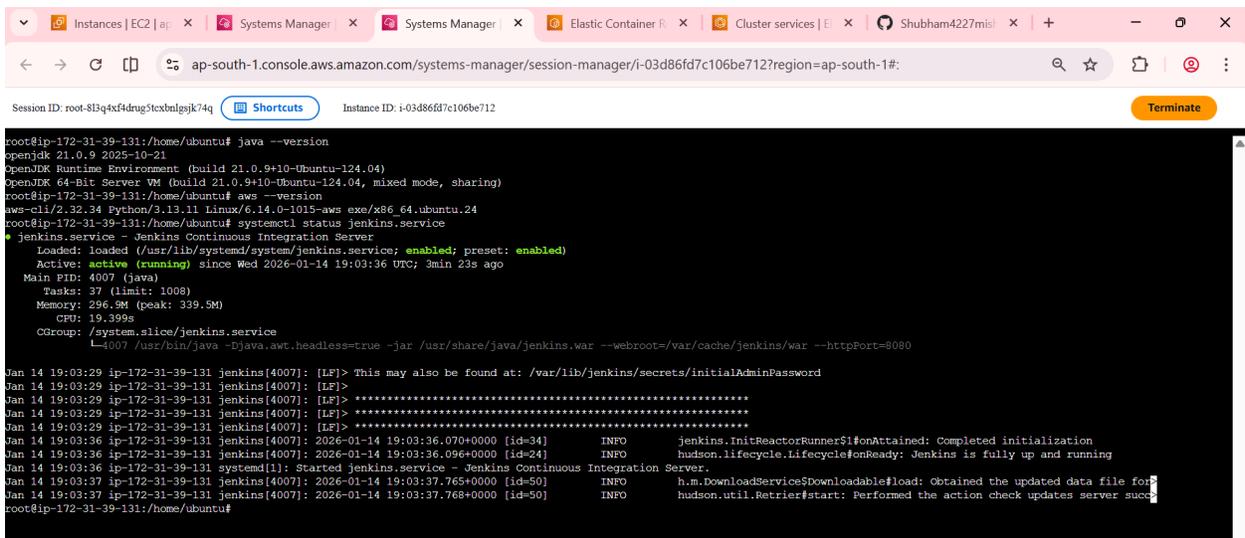
Zero-Trust, SSM-Driven CI/CD Pipeline



Step 1) We will launch two ec2 instances 1 will act as jenkins master and other will act as worker node



Step 2) On jenkins master we installed java, jenkins and awscli



Step 3) Now on worker ec2 we will install java, awscli and docker

```
Instances | EC2 | ap-south-1 | Systems Manager | Elastic Container Registry | Cluster services | Shubham4227mishra | ap-south-1-console.aws.amazon.com/systems-manager/session-manager/i-026b3fdb2246e82a3?region=ap-south-1#

Session ID: root-9ngsb64qpx5o86iyggyzti Instance ID: i-026b3fdb2246e82a3

root@ip-172-31-11-135:~# java --version
openjdk 21.0.9 2025-10-21
OpenJDK Runtime Environment (build 21.0.9+10-Ubuntu-124.04)
OpenJDK 64-Bit Server VM (build 21.0.9+10-Ubuntu-124.04, mixed mode, sharing)
root@ip-172-31-11-135:~# aws --version
[aws-cli/2.32.34 python/3.11 linux/6.14.0-1015-aws exe/x86_64.ubuntu.24]
root@ip-172-31-11-135:~# systemctl status docker.service
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: enabled)
   Active: active (running) since Wed 2026-01-14 19:07:08 UTC; 3min 5s ago
   TriggeredBy: ● docker.socket
     Docs: https://docs.docker.com
   Main PID: 3859 (dockerd)
     Tasks: 8
    Memory: 35.1M (peak: 36.1M)
       CPU: 323ms
    CGroup: /system.slice/docker.service
           └─3859 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Jan 14 19:07:08 ip-172-31-11-135 dockerd[3859]: time="2026-01-14T19:07:08.307948157Z" level=info msg="Loading containers: start."
Jan 14 19:07:08 ip-172-31-11-135 dockerd[3859]: time="2026-01-14T19:07:08.773847002Z" level=info msg="Loading containers: done."
Jan 14 19:07:08 ip-172-31-11-135 dockerd[3859]: time="2026-01-14T19:07:08.801315599Z" level=info msg="docker daemon commit="28.2.2-0ubuntu1-24.04.1" containerd-snapshotter=false"
Jan 14 19:07:08 ip-172-31-11-135 dockerd[3859]: time="2026-01-14T19:07:08.801403256Z" level=info msg="Initializing buildkit"
Jan 14 19:07:08 ip-172-31-11-135 dockerd[3859]: time="2026-01-14T19:07:08.805836679Z" level=warning msg="CDI setup error /etc/cdi: failed to monitor for changes: no such file or directory"
Jan 14 19:07:08 ip-172-31-11-135 dockerd[3859]: time="2026-01-14T19:07:08.805853198Z" level=warning msg="CDI setup error /var/run/cdi: failed to monitor for changes: no such file or directory"
Jan 14 19:07:08 ip-172-31-11-135 dockerd[3859]: time="2026-01-14T19:07:08.818300353Z" level=info msg="Completed buildkit initialization"
Jan 14 19:07:08 ip-172-31-11-135 dockerd[3859]: time="2026-01-14T19:07:08.83130474Z" level=info msg="Daemon has completed initialization"
Jan 14 19:07:08 ip-172-31-11-135 dockerd[3859]: time="2026-01-14T19:07:08.83138875Z" level=info msg="API listen on /run/docker.sock"
Jan 14 19:07:08 ip-172-31-11-135 systemd[1]: Started docker.service - Docker Application Container Engine.
```

Step 4) Now on both master and worker node ssm agent is running as we wont use ssh or pem key for robust security

```
Instances | EC2 | ap-south-1 | System | System | Elastic Container Registry | Cluster services | Elastic | Shubham4227mishra | ap-south-1-console.aws.amazon.com/systems-manager/session-manager/i-03d86fd7c106be712?region=ap-south-1#

Session ID: root-9ngsb64qpx5o86iyggyzti Instance ID: i-026b3fdb2246e82a3
Session ID: root-813q4xf4drug5tcbnlgsjk74q Instance ID: i-03d86fd7c106be712

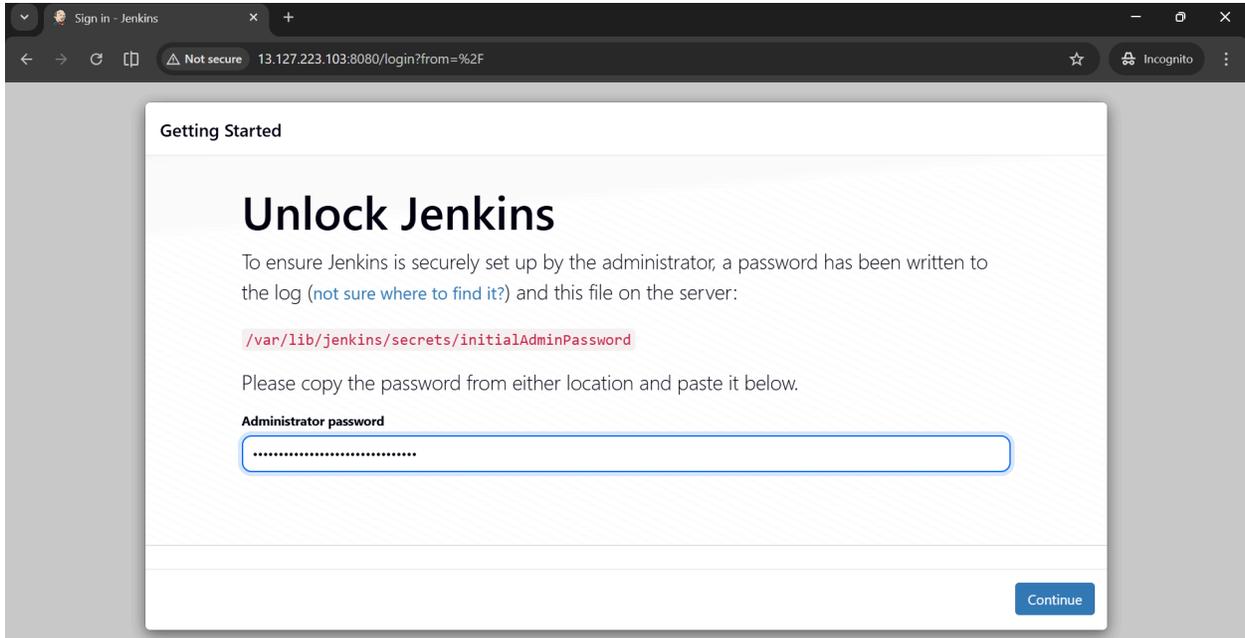
● snap.amazon-ssm-agent.amazon-ssm-agent.service - Service for snap application amazon-ssm-agent
   Loaded: loaded (/etc/systemd/system/snap.amazon-ssm-agent.amazon-ssm-agent.service; enabled; preset: enabled)
   Active: active (running) since Wed 2026-01-14 19:03:50 UTC; 7min ago
   Main PID: 514 (amazon-ssm-agent)
     Tasks: 27 (limit: 1121)
    Memory: 113.9M (peak: 121.4M)
       CPU: 3.174s
    CGroup: /system.slice/snap.amazon-ssm-agent.amazon-ssm-agent.service
           └─514 /snap/amazon-ssm-agent/11797/amazon-ssm-agent
             └─513 /snap/amazon-ssm-agent/11797/ssm-agent-worker root-9ngsb64qpx5o86iyggyzti
             └─835 /snap/amazon-ssm-agent/11797/ssm-session-worker root-9ngsb64qpx5o86iyggyzti
             └─863 sh

Jan 14 19:03:52 ip-172-31-11-135 amazon-ssm-agent.amazon-ssm-agent[514]: 2026-01-14 19:03:52.000000 [INFO] Starting Amazon SSM Agent
Jan 14 19:03:52 ip-172-31-11-135 amazon-ssm-agent.amazon-ssm-agent[514]: 2026-01-14 19:03:52.000000 [INFO] Amazon SSM Agent version: 3.1.0
Jan 14 19:03:52 ip-172-31-11-135 amazon-ssm-agent.amazon-ssm-agent[514]: 2026-01-14 19:03:52.000000 [INFO] Amazon SSM Agent commit: 2026-01-14 19:03:52.000000
Jan 14 19:03:52 ip-172-31-11-135 amazon-ssm-agent.amazon-ssm-agent[514]: 2026-01-14 19:03:52.000000 [INFO] Amazon SSM Agent build: 2026-01-14 19:03:52.000000
Jan 14 19:03:54 ip-172-31-11-135 amazon-ssm-agent.amazon-ssm-agent[514]: 2026-01-14 19:03:54.000000 [INFO] Amazon SSM Agent build: 2026-01-14 19:03:54.000000
Jan 14 19:05:32 ip-172-31-11-135 useradd[848]: new group: name=ssm-user, GID=1001
Jan 14 19:05:32 ip-172-31-11-135 useradd[848]: new user: name=ssm-user, UID=1001, GID=1001
Jan 14 19:05:38 ip-172-31-11-135 sudo[864]: ssm-user : TTY=pts/0 ; FWD=/root ; USER=root ; COMMAND=ssh -i /root/.ssh/agent_key.pem ec2-user@ip-172-31-11-135
Jan 14 19:05:38 ip-172-31-11-135 sudo[864]: pam_unix(sudo-1:session): session opened for user ssm-user by ssm-user

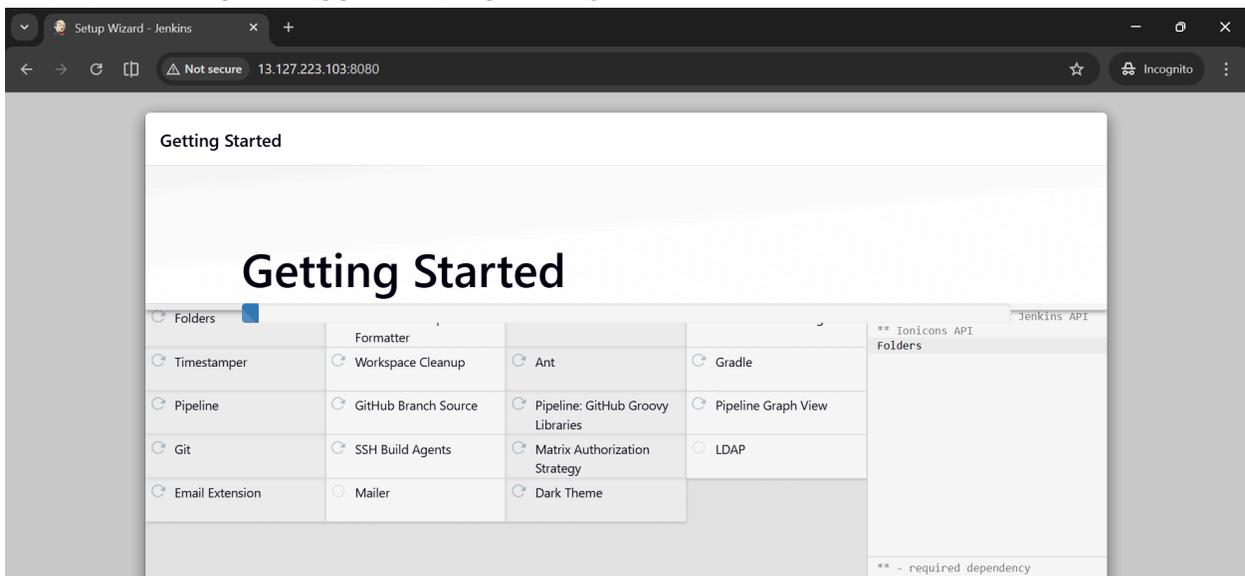
● snap.amazon-ssm-agent.amazon-ssm-agent.service - Service for snap application amazon-ssm-agent
   Loaded: loaded (/etc/systemd/system/snap.amazon-ssm-agent.amazon-ssm-agent.service; enabled; preset: enabled)
   Active: active (running) since Wed 2026-01-14 18:53:39 UTC; 17min ago
   Main PID: 971 (amazon-ssm-agent)
     Tasks: 31 (limit: 1008)
    Memory: 62.0M (peak: 101.3M)
       CPU: 7.876s
    CGroup: /system.slice/snap.amazon-ssm-agent.amazon-ssm-agent.service
           └─971 /snap/amazon-ssm-agent/11797/amazon-ssm-agent
             └─971 /snap/amazon-ssm-agent/11797/amazon-ssm-agent
             └─1004 /snap/amazon-ssm-agent/11797/ssm-agent-worker root-813q4xf4drug5tcbnlgsjk74q
             └─1057 /snap/amazon-ssm-agent/11797/ssm-session-worker root-813q4xf4drug5tcbnlgsjk74q
             └─1085 sh

Jan 14 18:53:40 ip-172-31-39-131 amazon-ssm-agent.amazon-ssm-agent[971]: 2026-01-14 18:53:40.000000 [INFO] Starting Amazon SSM Agent
Jan 14 18:53:40 ip-172-31-39-131 amazon-ssm-agent.amazon-ssm-agent[971]: 2026-01-14 18:53:40.000000 [INFO] Amazon SSM Agent version: 3.1.0
Jan 14 18:53:40 ip-172-31-39-131 amazon-ssm-agent.amazon-ssm-agent[971]: 2026-01-14 18:53:40.000000 [INFO] Amazon SSM Agent commit: 2026-01-14 18:53:40.000000
Jan 14 18:53:40 ip-172-31-39-131 amazon-ssm-agent.amazon-ssm-agent[971]: 2026-01-14 18:53:40.000000 [INFO] Amazon SSM Agent build: 2026-01-14 18:53:40.000000
Jan 14 18:53:41 ip-172-31-39-131 amazon-ssm-agent.amazon-ssm-agent[971]: 2026-01-14 18:53:41.000000 [INFO] Amazon SSM Agent build: 2026-01-14 18:53:41.000000
Jan 14 18:53:41 ip-172-31-39-131 amazon-ssm-agent.amazon-ssm-agent[971]: 2026-01-14 18:53:41.000000 [INFO] Amazon SSM Agent build: 2026-01-14 18:53:41.000000
Jan 14 18:56:50 ip-172-31-39-131 useradd[1071]: new group: name=ssm-user, GID=1001
Jan 14 18:56:50 ip-172-31-39-131 useradd[1071]: new user: name=ssm-user, UID=1001, GID=1001
Jan 14 18:58:05 ip-172-31-39-131 sudo[1090]: ssm-user : TTY=pts/0 ; FWD=/root ; USER=root ; COMMAND=ssh -i /root/.ssh/agent_key.pem ec2-user@ip-172-31-11-135
Jan 14 18:58:05 ip-172-31-39-131 sudo[1090]: pam_unix(sudo-1:session): session opened for user ssm-user by ssm-user
```

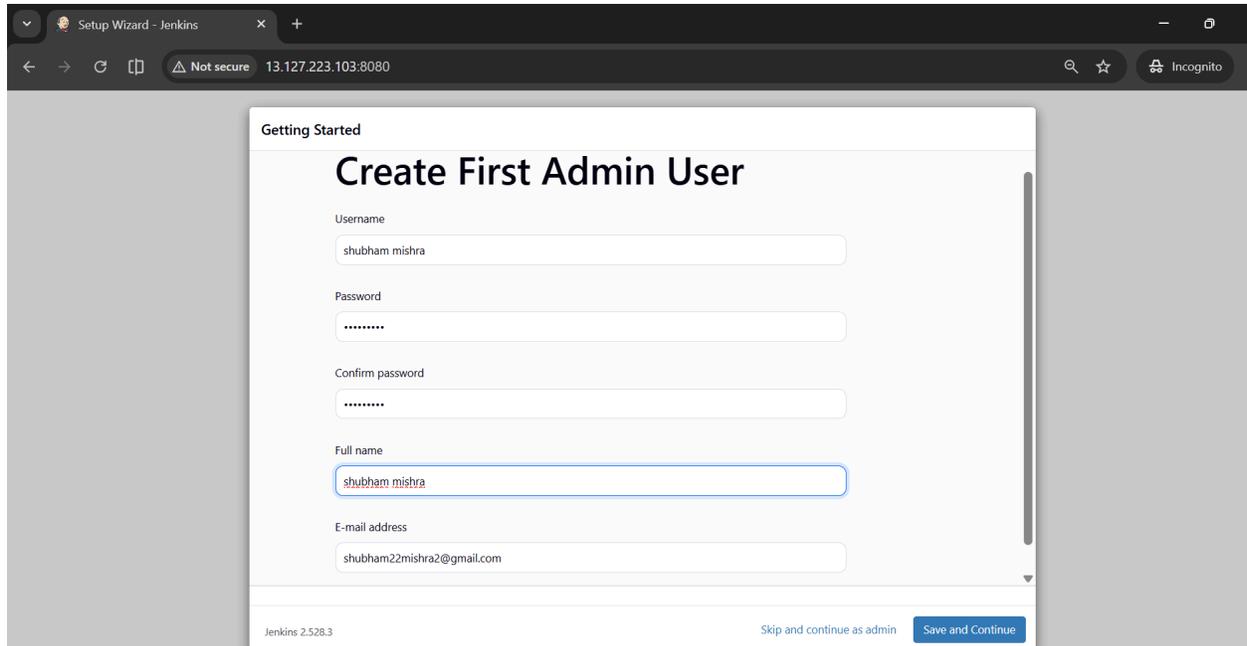
Step 5) Now we will setup jenkins on master node



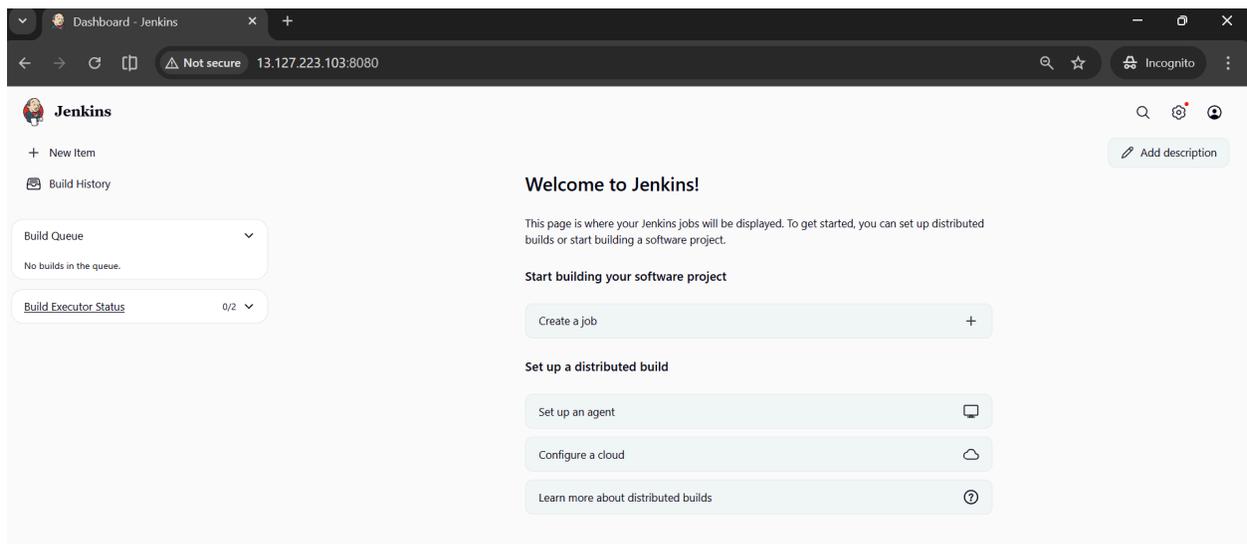
Step 6) Installing all suggested plugins on Jenkins



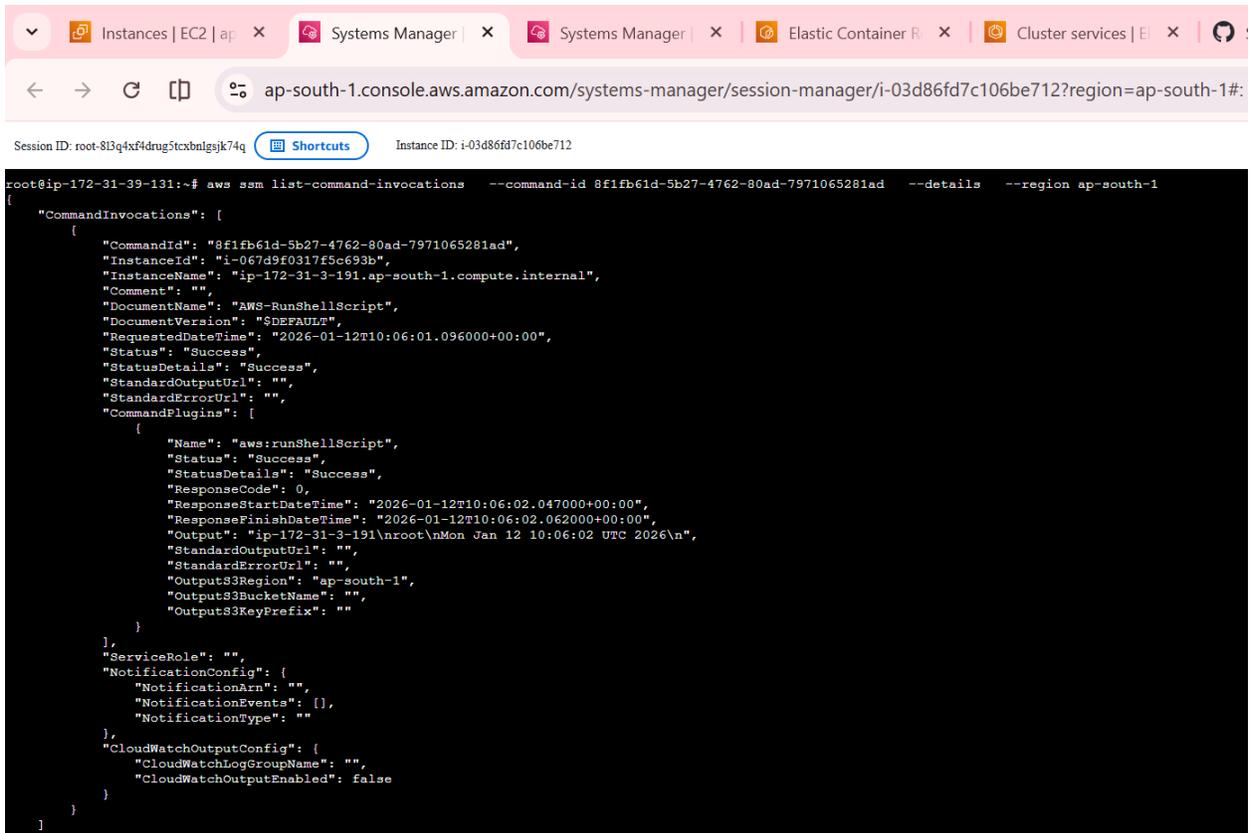
Step 7) We will setup username and password in Jenkins



Step 8) jenkins setup successfull



Step 9) Now we will check whether jenkins master can communicate with worker using ssm commands



The screenshot shows the AWS Systems Manager console interface. At the top, there are several browser tabs: 'Instances | EC2 | ap...', 'Systems Manager | x', 'Systems Manager | x', 'Elastic Container R...', and 'Cluster services | E...'. The address bar shows the URL: 'ap-south-1.console.aws.amazon.com/systems-manager/session-manager/i-03d86fd7c106be712?region=ap-south-1#...'. Below the address bar, the session ID is 'root-813q4xf4drug5tctxbnlgsjk74q' and the instance ID is 'i-03d86fd7c106be712'. A 'Shortcuts' button is visible. The main content is a terminal window with the following command and output:

```
root@ip-172-31-39-131:~# aws ssm list-command-invocations --command-id 8f1fb61d-5b27-4762-80ad-7971065281ad --details --region ap-south-1
{
  "CommandInvocations": [
    {
      "CommandId": "8f1fb61d-5b27-4762-80ad-7971065281ad",
      "InstanceId": "i-067d9f0317f5c693b",
      "InstanceName": "ip-172-31-3-191.ap-south-1.compute.internal",
      "Comment": "",
      "DocumentName": "AWS-RunShellScript",
      "DocumentVersion": "$DEFAULT",
      "RequestedDateTime": "2026-01-12T10:06:01.096000+00:00",
      "Status": "Success",
      "StatusDetails": "Success",
      "StandardOutputUrl": "",
      "StandardErrorUrl": "",
      "CommandPlugins": [
        {
          "Name": "aws:runShellScript",
          "Status": "Success",
          "StatusDetails": "Success",
          "ResponseCode": 0,
          "ResponseStartDateTime": "2026-01-12T10:06:02.047000+00:00",
          "ResponseFinishDateTime": "2026-01-12T10:06:02.062000+00:00",
          "Output": "ip-172-31-3-191\nroot\nMon Jan 12 10:06:02 UTC 2026\n",
          "StandardOutputUrl": "",
          "StandardErrorUrl": "",
          "OutputS3Region": "ap-south-1",
          "OutputS3BucketName": "",
          "OutputS3KeyPrefix": ""
        }
      ],
      "ServiceRole": "",
      "NotificationConfig": {
        "NotificationArn": "",
        "NotificationEvents": [],
        "NotificationType": ""
      },
      "CloudWatchOutputConfig": {
        "CloudWatchLogGroupName": "",
        "CloudWatchOutputEnabled": false
      }
    }
  ]
}
```

Step 10) Now we will create project folder and initialise git repo also create index.html file in folder /home/ubuntu/project/prod_application

Instances | EC2 | ap x | Systems Manager | x | Systems Manager | x | Elastic Container

← → ↻ 📄 🔍 ap-south-1.console.aws.amazon.com/systems-manager/session-manager/i-02

Session ID: root-9ngsbl64gqx5o8o6iygjztyzi

[Shortcuts](#)

Instance ID: i-026b3fdb2246e82a3

```
root@ip-172-31-11-135:/home/ubuntu/project/prod_application# ls
index.html
root@ip-172-31-11-135:/home/ubuntu/project/prod_application# cd ..
root@ip-172-31-11-135:/home/ubuntu/project# ls
prod_application
root@ip-172-31-11-135:/home/ubuntu/project# git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  prod_application/

nothing added to commit but untracked files present (use "git add" to track)
root@ip-172-31-11-135:/home/ubuntu/project# git add *
root@ip-172-31-11-135:/home/ubuntu/project# git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
  new file:   prod_application/index.html

root@ip-172-31-11-135:/home/ubuntu/project# git commit -m "added index.html file"
[master (root-commit) 8a0044e] added index.html file
Committer: root <root@ip-172-31-11-135.ap-south-1.compute.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

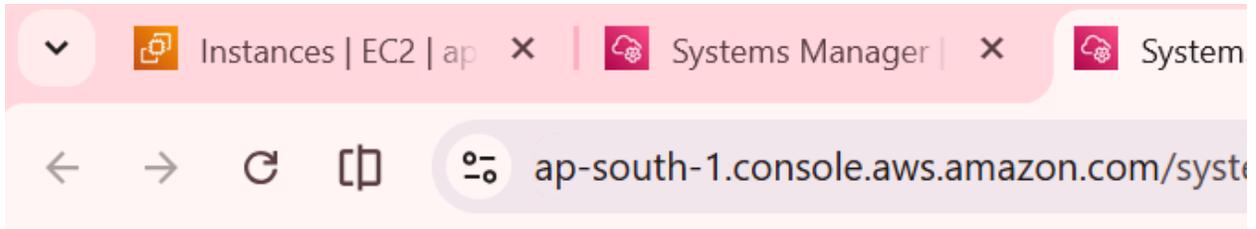
    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

1 file changed, 68 insertions(+)
 create mode 100644 prod_application/index.html
root@ip-172-31-11-135:/home/ubuntu/project# pwd
/home/ubuntu/project
root@ip-172-31-11-135:/home/ubuntu/project# cd prod_application/
```

Step 11) Now we will create Docker file for this index.html code on worker node



Session ID: root-9ngsbl64gqx5o8o6iygjgzyzi

 Shortcuts

Ins

```
FROM nginx:alpine

# Remove default nginx website
RUN rm -rf /usr/share/nginx/html/*

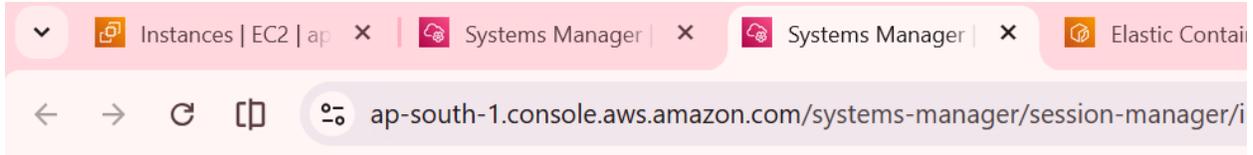
# Copy our production page
COPY index.html /usr/share/nginx/html/index.html

# Expose port 80
EXPOSE 80

# Start nginx
CMD ["nginx", "-g", "daemon off;"]

~
~
~
```

Step 12) Now we will build docker file locally on worker node



Session ID: root-9ngsbl64gqx5o8o6iygijgzyzi

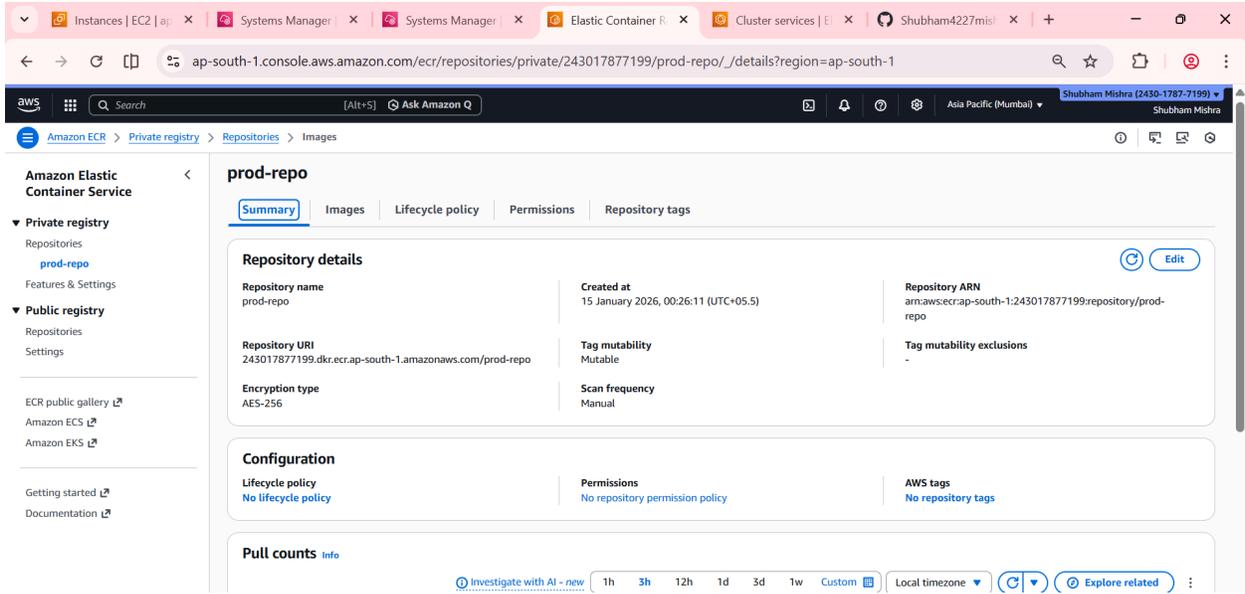
Shortcuts

Instance ID: i-026b3fdb2246e82a3

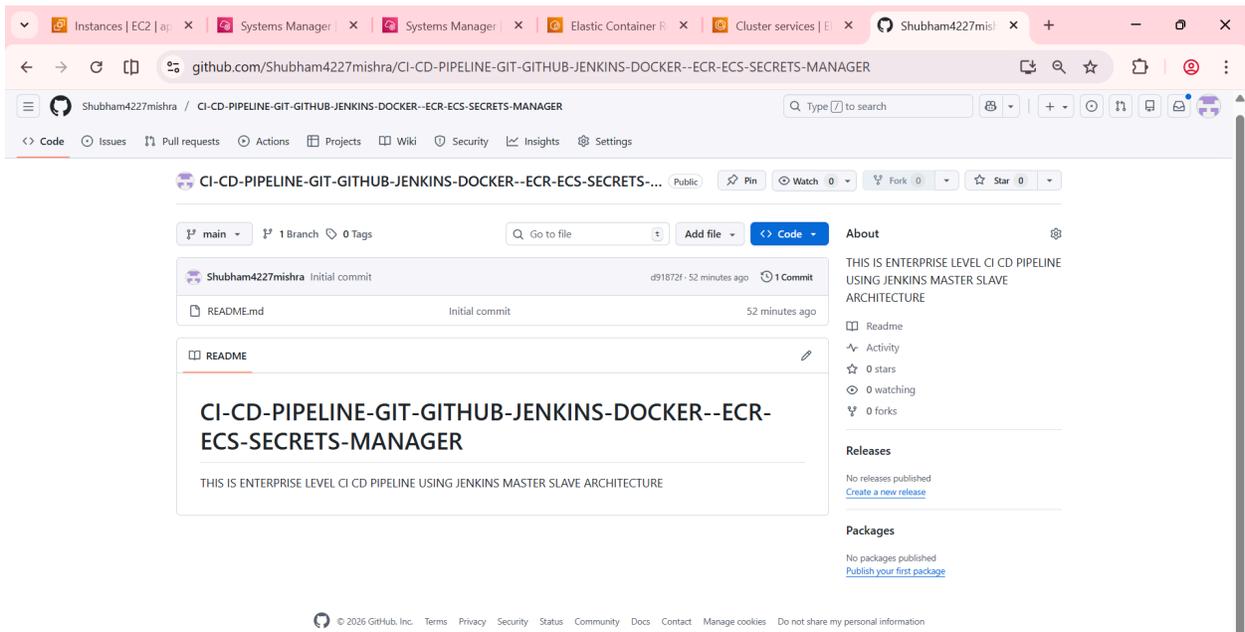
```
root@ip-172-31-11-135:/home/ubuntu/project/prod_application# docker build -t prod_application .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
Install the buildx component to build images with BuildKit:
https://docs.docker.com/go/buildx/

Sending build context to Docker daemon 4.608kB
Step 1/5 : FROM nginx:alpine
alpine: Pulling from library/nginx
1074353eec0d: Pulling fs layer
25f453064fd3: Pulling fs layer
567f84da6fbd: Pulling fs layer
da7c973d8b92: Pulling fs layer
33f95a0f3229: Pulling fs layer
085c5e5aaa8e: Pulling fs layer
0abf9e567266: Pulling fs layer
e096540205d5: Pulling fs layer
da7c973d8b92: Waiting
33f95a0f3229: Waiting
085c5e5aaa8e: Waiting
0abf9e567266: Waiting
e096540205d5: Waiting
25f453064fd3: Verifying Checksum
25f453064fd3: Download complete
567f84da6fbd: Verifying Checksum
567f84da6fbd: Download complete
1074353eec0d: Verifying Checksum
1074353eec0d: Download complete
1074353eec0d: Pull complete
25f453064fd3: Pull complete
567f84da6fbd: Pull complete
085c5e5aaa8e: Verifying Checksum
085c5e5aaa8e: Download complete
da7c973d8b92: Verifying Checksum
da7c973d8b92: Download complete
33f95a0f3229: Verifying Checksum
33f95a0f3229: Download complete
da7c973d8b92: Pull complete
33f95a0f3229: Pull complete
085c5e5aaa8e: Pull complete
0abf9e567266: Verifying Checksum
0abf9e567266: Download complete
0abf9e567266: Pull complete
e096540205d5: Verifying Checksum
e096540205d5: Download complete
e096540205d5: Pull complete
```

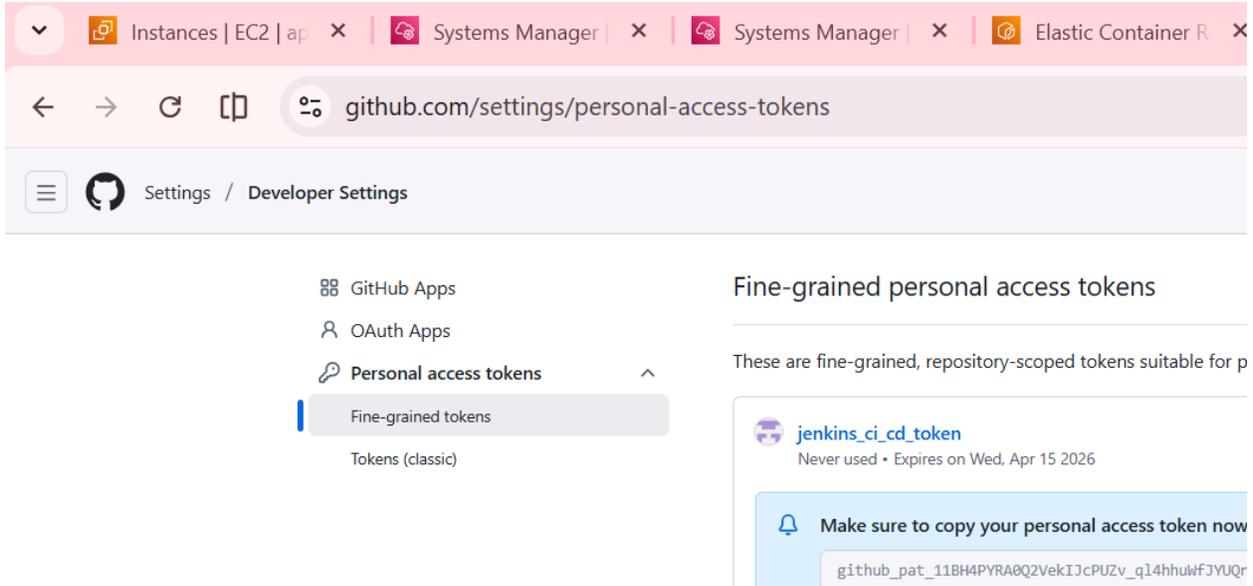
Step 13) Now we will create ECR private repo



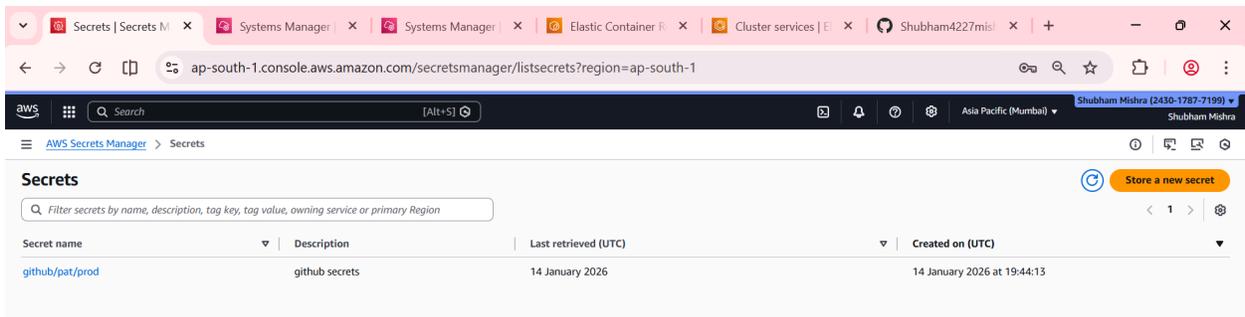
Step 14) Now we will also create github repository



Step 15) Now we will create PAT token to allow git on worker ec2 to connect to GITHUB and push code



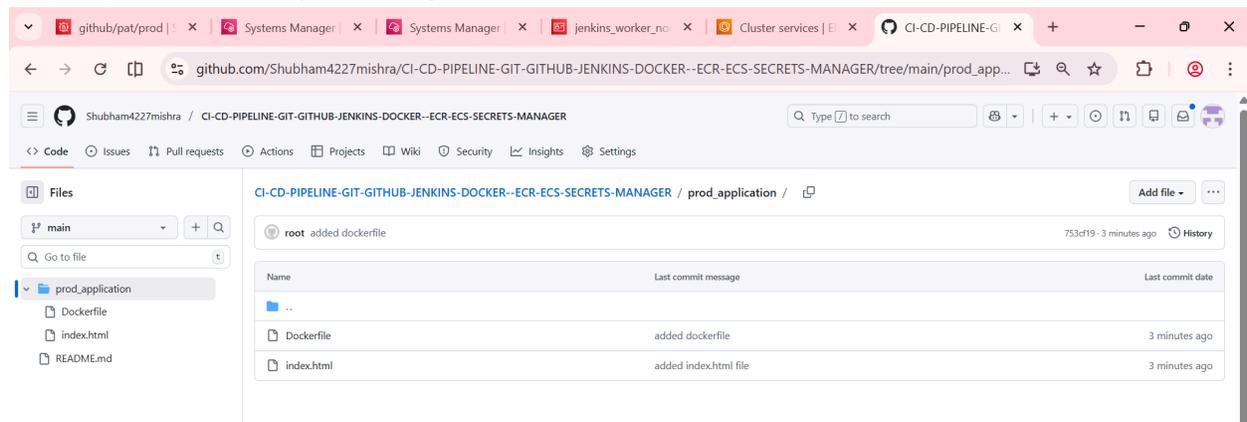
Step 16) Now we will store this pat token and username in aws secrets so that git on ec2 can use that and push code into github repo



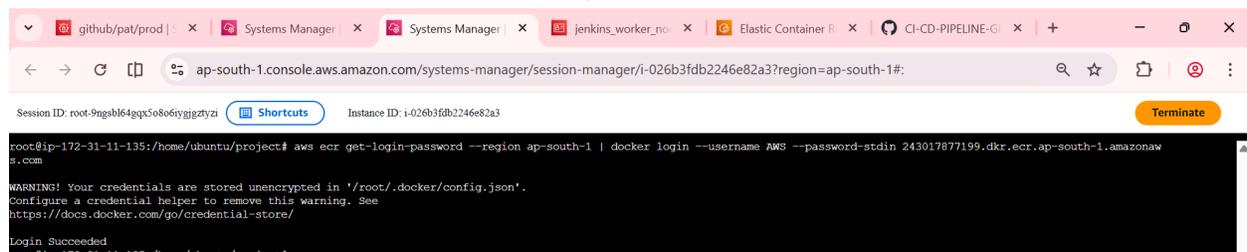
Step 17) Now we pushed our index.html and Dockerfile on github successfully

```
From https://github.com/Shubham4227mishra/CI-CD-PIPELINE-GIT-G
* branch          main          -> FETCH_HEAD
Successfully rebased and updated refs/heads/main.
root@ip-172-31-11-135:/home/ubuntu/project# git push -u origin
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Compressing objects: 100% (7/7), done.
Writing objects: 100% (8/8), 1.73 KiB | 1.73 MiB/s, done.
Total 8 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Shubham4227mishra/CI-CD-PIPELINE-GIT-GIT
d91872f..753cf19  main -> main
branch 'main' set up to track 'origin/main'.
root@ip-172-31-11-135:/home/ubuntu/project#
```

Step 18) We can verify files in github



Step 19) Now from worker node we will login into ECR private repo



Step 20) We will build docker image and push it into ecr

```
github/pat/prod | x Systems Manager | x Systems Manager | x jenkins_worker_no x Elastic Container R x CI-CD-PIPELINE-GI x
ap-south-1.console.aws.amazon.com/systems-manager/session-manager/i-026b3fdb2246e82a3?region=ap-south-1#

Session ID: root-9ngsbl64gqx5o8o6iygiztyzi Shortcuts Instance ID: i-026b3fdb2246e82a3

Install the buildx component to build images with BuildKit:
https://docs.docker.com/go/buildx/

Sending build context to Docker daemon 4.608kB
Step 1/5 : FROM nginx:alpine
----> 004f022f4ef5
Step 2/5 : RUN rm -rf /usr/share/nginx/html/*
----> Using cache
----> 9b06107cf546
Step 3/5 : COPY index.html /usr/share/nginx/html/index.html
----> Using cache
----> 5e5432ae50bf
Step 4/5 : EXPOSE 80
----> Using cache
----> e51991d3618e
Step 5/5 : CMD ["nginx", "-g", "daemon off;"]
----> Using cache
----> eeee592306b8
Successfully built eeee592306b8
Successfully tagged prod-app-test:latest
root@ip-172-31-11-135:/home/ubuntu/project/prod_application# ls
Dockerfile index.html
root@ip-172-31-11-135:/home/ubuntu/project/prod_application# docker tag prod-app-test:latest 243017877199.dkr.ecr.ap-south-1.amazonaws.com/prod-repo:v1
root@ip-172-31-11-135:/home/ubuntu/project/prod_application# docker push 243017877199.dkr.ecr.ap-south-1.amazonaws.com/prod-repo:v1
The push refers to repository [243017877199.dkr.ecr.ap-south-1.amazonaws.com/prod-repo]
cd9dd660a4b3: Pushed
e5a22a527d41: Pushed
4b6d03d0cebb: Pushed
67ea0b046e7d: Pushed
ed5fa8595c7a: Pushed
8ae63eb1f31f: Pushed
b3e3d1bbb64d: Pushed
48078b7e3000: Pushed
fd1e40d7f74b: Pushed
7bb20cf5ef67: Pushed
v1: digest: sha256:e2e9fe32bbb8484cab1e7d3228279b1b0fd9c0948f675a91ad0ebdbbb934a388 size: 2404
root@ip-172-31-11-135:/home/ubuntu/project/prod_application#
```

Step 21) We can verify image is pushed into ECR

The screenshot shows the AWS ECR console interface. The breadcrumb navigation is: Amazon ECR > Private registry > Repositories > Images > sha256:e2e9fe32bbb8484cab1e7d3228279b1b0fd9c0948f675a91ad0ebdbbb934a388. The main content area displays the 'Image' details for the tag 'v1'. The 'Details' section includes the URI '243017877199.dkr.ecr.ap-south-1.amazonaws.com/prod-repo/v1' and the Digest 'sha256:e2e9fe32bbb8484cab1e7d3228279b1b0fd9c0948f675a91ad0ebdbbb934a388'. The 'General information' section shows the artifact type as 'Image', the repository as 'prod-repo', and the pushed date as '15 January 2026, 01:47:02 (UTC+05.5)'. The 'Image status' is 'Active'. There is also a 'Scanning and vulnerabilities' section with a 'Scan' button.

Step 22) Now we will create ECS cluster

The screenshot shows the AWS Elastic Container Service (ECS) console. At the top, a green notification bar states "Cluster PROD_CLUSTER has been created successfully." Below this, the "PROD_CLUSTER" overview is displayed. Key details include:

- ARN:** arn:aws:ecs:ap-south-1:243017877199:cluster/PROD_CLUSTER
- Status:** Active
- CloudWatch monitoring:** Container Insights with enhanced observability
- Registered container instances:** -
- Services:** 0
- Draining:** -
- Active:** -
- Pending:** -
- Running:** -

 The left sidebar shows navigation options for Amazon Elastic Container Service, including Express Mode, Clusters, Namespaces, Task definitions, and Account settings. The main content area includes tabs for Services, Tasks, Infrastructure, Metrics, Scheduled tasks, Configuration, Event history, and Tags. A "Services (0)" section is visible with filters for launch type, scheduling strategy, and resource management type.

Step 23) Now we will create task definition for the cluster

The screenshot shows the "Create new task definition" page in the AWS Elastic Container Service console. The page is titled "Create new task definition" and contains the following configuration sections:

- Task definition configuration:**
 - Task definition family:** prod_task-definition
 - Up to 255 letters (uppercase and lowercase), numbers, hyphens, and underscores are allowed.
- Infrastructure requirements:**
 - Launch type:** AWS Fargate (selected)
 - Managed instances - new (unchecked)
 - Amazon EC2 instances (unchecked)
- OS, Architecture, Network mode:**
 - Operating system/Architecture:** Linux/X86_64
 - Network mode:** awsvpc
- Task size:**
 - CPU:** .5 vCPU
 - Memory:** 1 GB

 The left sidebar shows navigation options for Amazon Elastic Container Service, including Express Mode, Clusters, Namespaces, Task definitions, and Account settings. The main content area is filled with the configuration options for the new task definition.

Amazon Elastic Container Service > Create new task definition

Name
 app-demo
 Up to 255 letters (uppercase and lowercase), numbers, hyphens, and underscores are allowed.

Essential container
 Yes

Image URI
 243017877199.dkr.ecr.ap-south-1.amazonaws.com/prod-repo@sha256:e29fe32bb8484cab1e7d3228279b1b0f9c0948f675a91ad0ebdbb934a388
 Up to 255 letters (uppercase and lowercase), numbers, hyphens, underscores, colons, periods, forward slashes, and number signs are allowed.

Private registry
 Store credentials in Secrets Manager, and then use the credentials to reference images in private registries.
 Private registry authentication

Port mappings
 Add port mappings to allow the container to access ports on the host to send or receive traffic. For port name, a default will be assigned if left blank.

Container port	Protocol	Port name	App protocol
80	TCP	demo	HTTP

Read-only root file system
 When this parameter is turned on, the container is given read-only access to its root file system.
 Read only

Resource allocation limits - conditional
 Container-level CPU, GPU and memory limits are different from task-level values. They define how many resources are allocated for the container. If the container attempts to exceed the memory specified by the hard limit, the container is terminated.

CPU	GPU	Memory hard limit	Memory soft limit
0.5 in vCPU	1	1 in GB	0.5 in GB

Environment variables - optional

Step 24) Now we will create service for the task definition and cluster

Amazon Elastic Container Service > Clusters > PROD_CLUSTER > Create service

Create service

Service details

Task definition family
 Select an existing task definition. To create a new task definition, go to [Task definitions](#).

prod_task-definition

Task definition revision
 Select the task definition revision from the 100 most recent entries, or enter a revision. Leave the field blank to use the latest revision.

1

Service name
 Assign a service name that is unique for this cluster.
 Up to 255 letters (uppercase and lowercase), numbers, underscores and hyphens are allowed. Service names must be unique within a cluster.

prod_task-definition-service-3054pvt

Environment
 Existing cluster: PROD_CLUSTER

Compute configuration - advanced

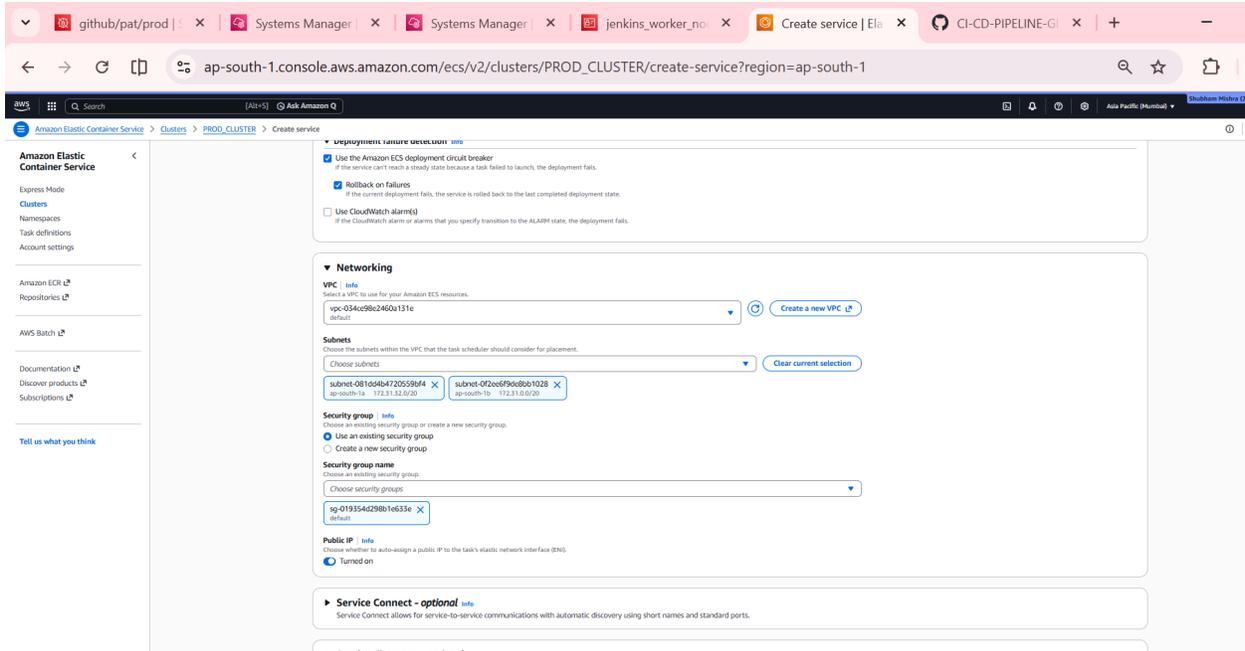
Compute options

Capacity provider strategy
 Specify a launch strategy to distribute your tasks across one or more capacity providers.

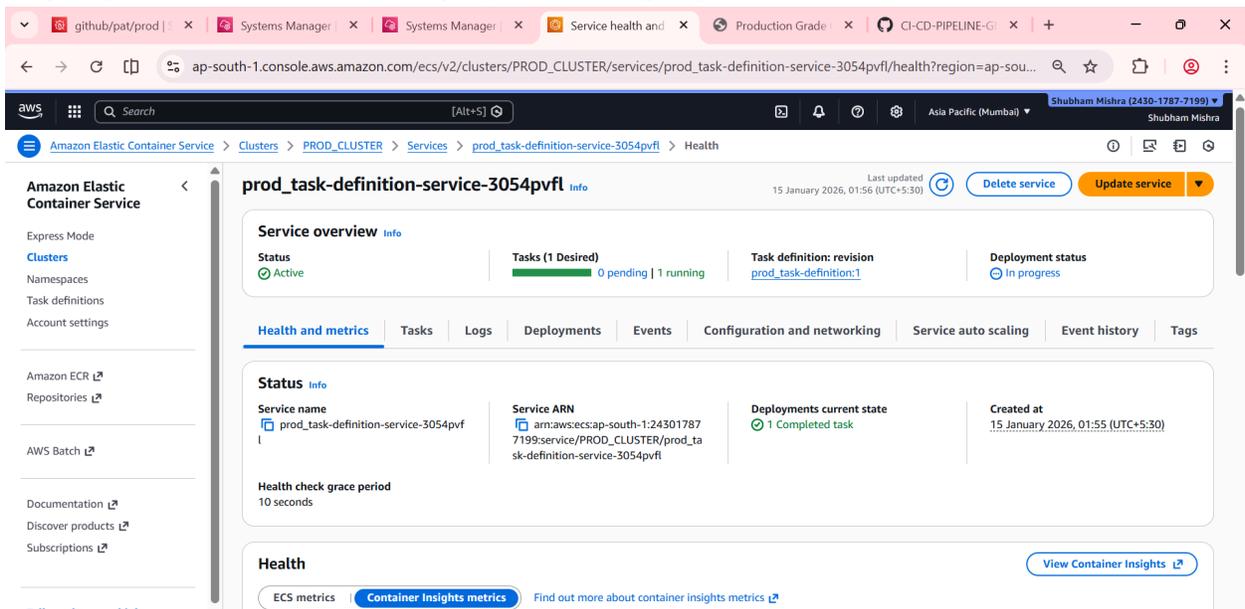
Launch type
 Launch tasks directly without the use of a capacity provider strategy.

Capacity provider strategy
 Select either your cluster default capacity provider strategy or select the customised option to configure a different strategy.
 Use cluster default
 Use custom (Advanced)

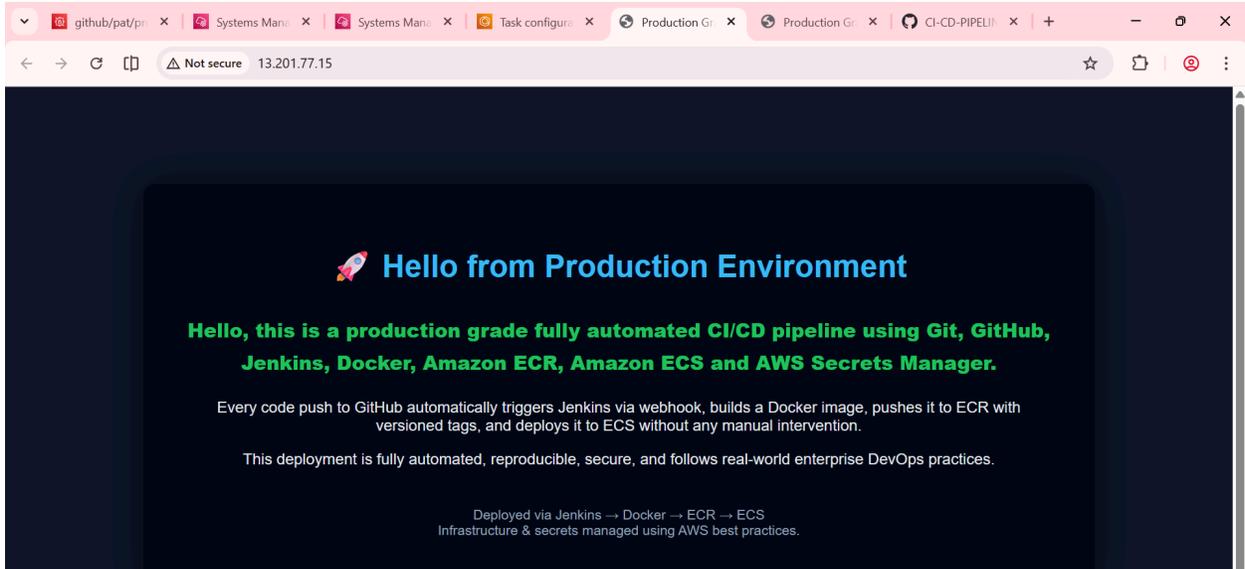
Capacity provider	Base	Weight
FARGATE	0	1



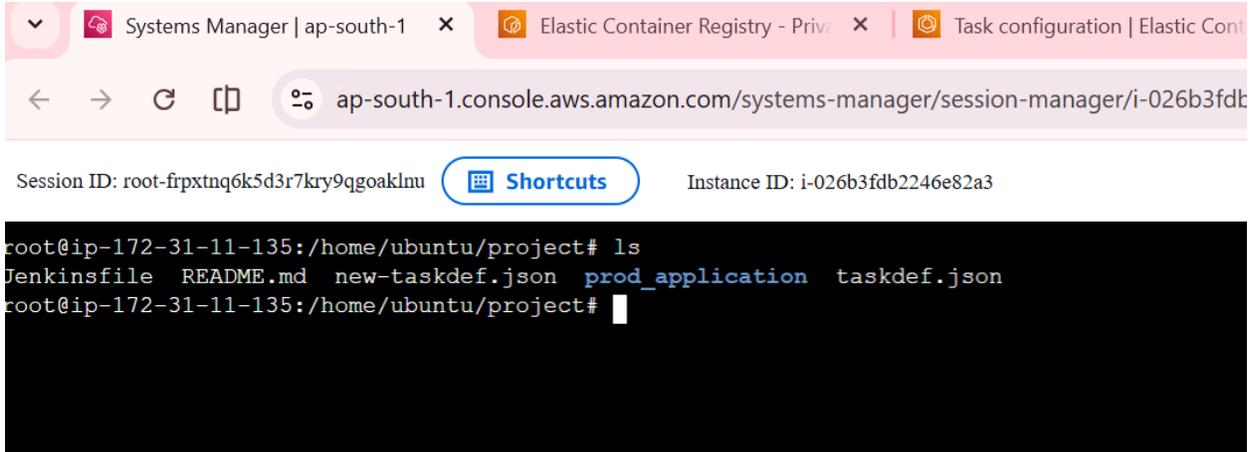
Step 25) Now we can see the deployment is in progress



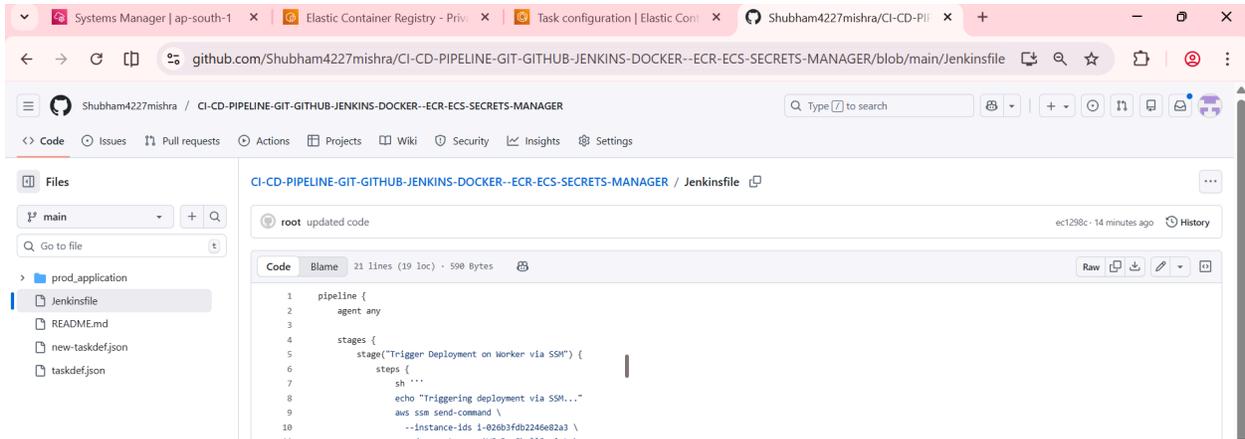
Step 26) Now after successful deployment we can see 1 container having private ip is launched and we can verify it



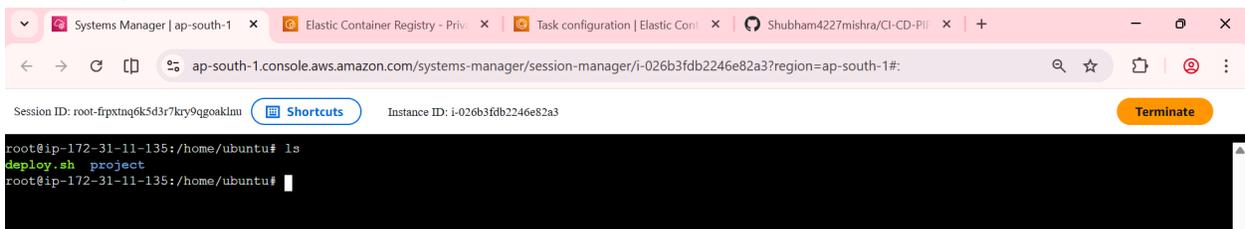
Step 27) Now we will automate this deployment on every code changes so we will create jenkins file on worker node and push it on github



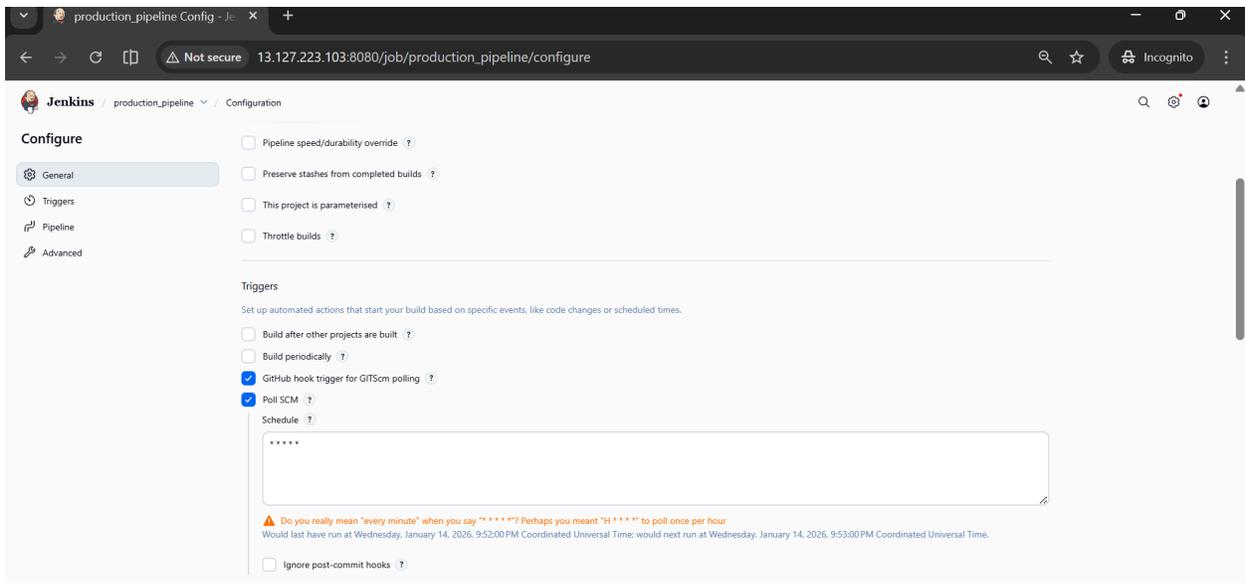
Step 28) Now Jenkinsfile is successfully pushed on github as jenkins will use this file to build the ci cd pipeline automatically

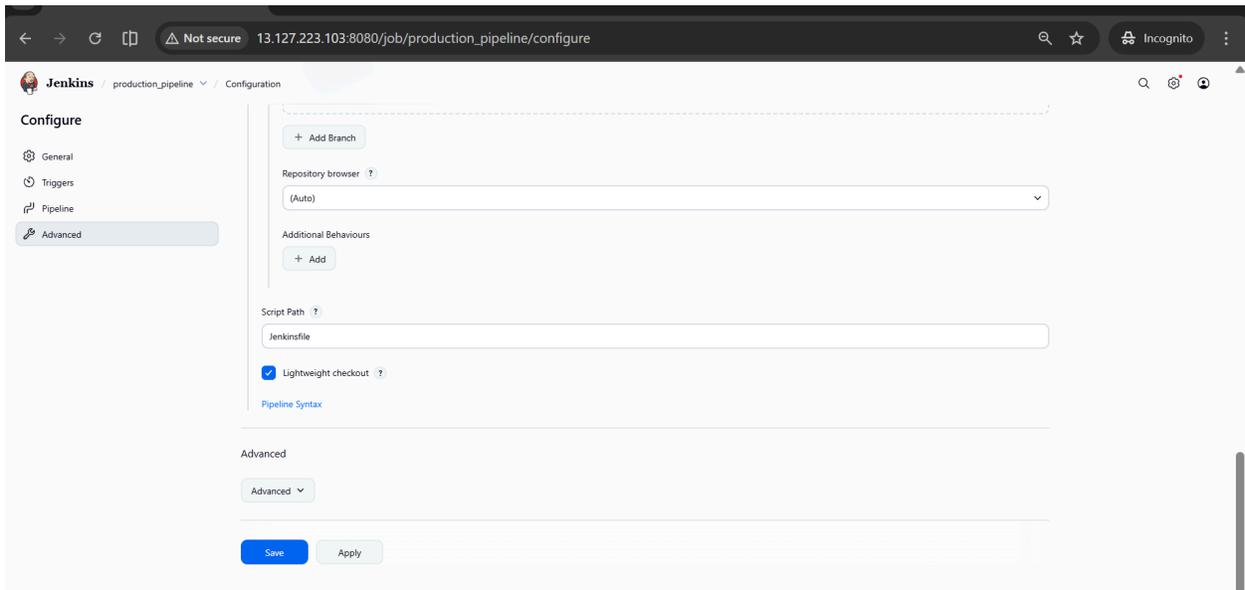
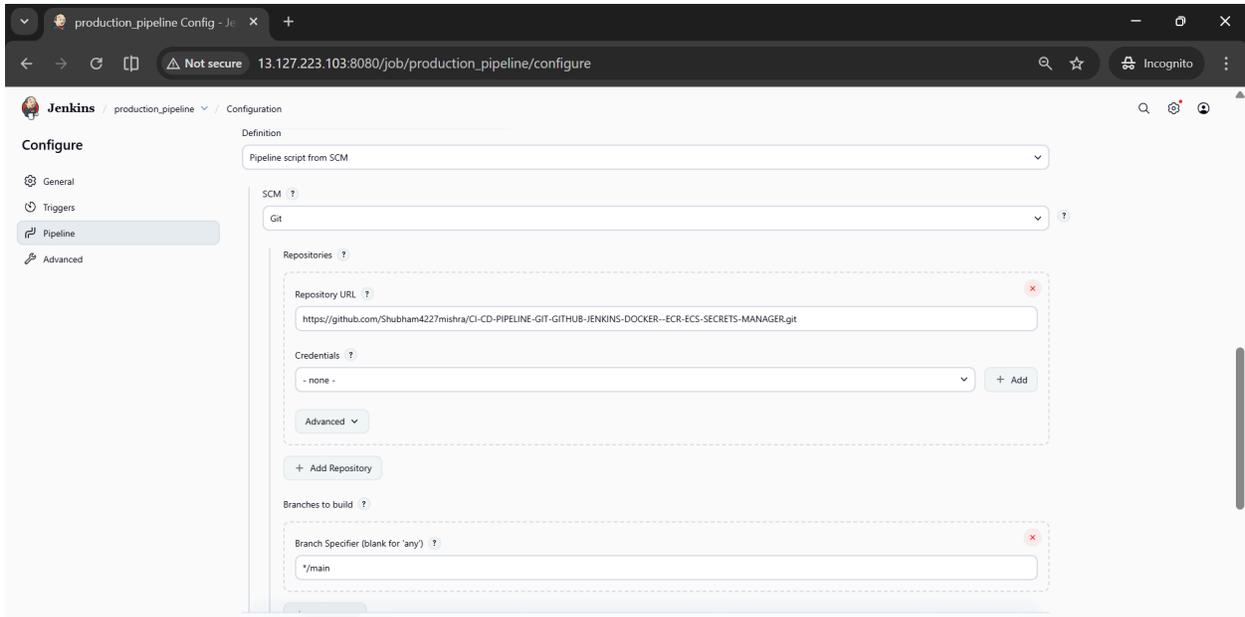


Step 29) Now we will create [deploy.sh](#) file on worker ec2 as we want master to use this file to deploy jobs on worker node only not on itself

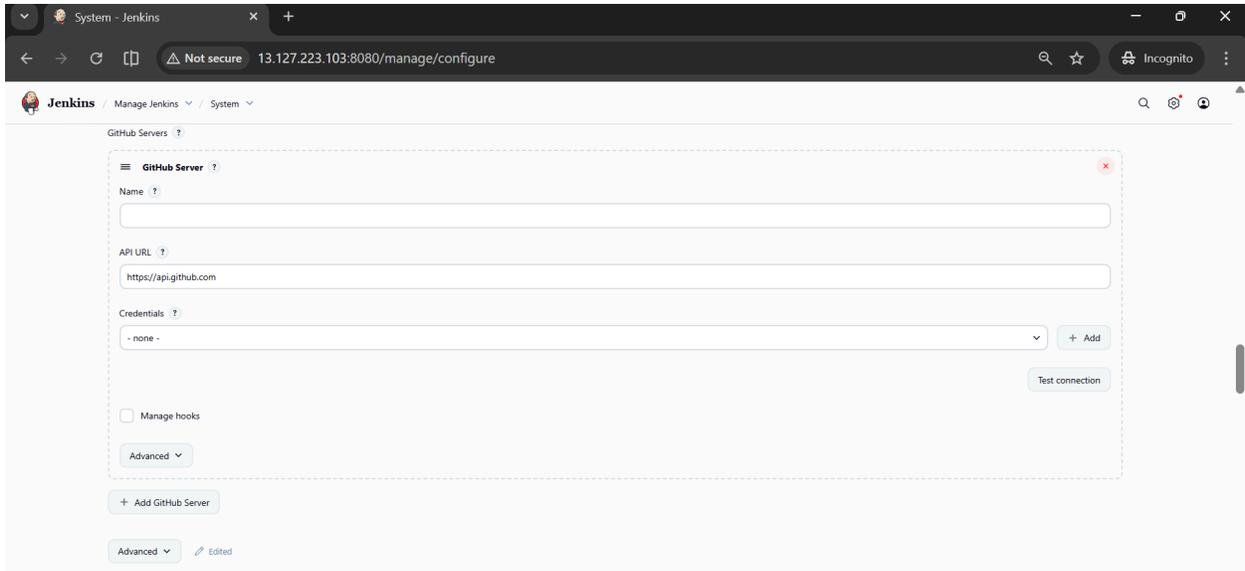


Step 30) Now we will create job in Jenkins

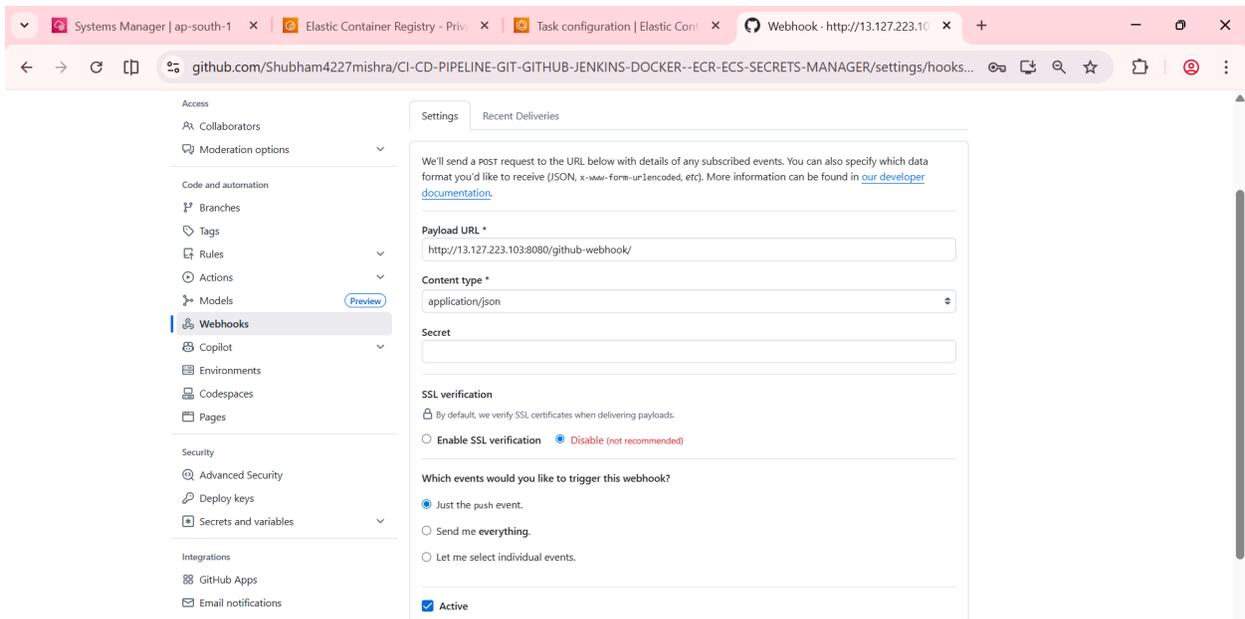




Step 31) Also make sure this is configured in jenkins system



Step 32) Now we will create webhook in github to connect jenkins to github



Step 33) Now we will update the code and push it from git to github and then we should see jenkins job automatically triggered

```
modified: ../taskdef.json

no changes added to commit (use "git add" and/or "git commit -a")
root@ip-172-31-11-135:/home/ubuntu/project/prod_application# cd ..
root@ip-172-31-11-135:/home/ubuntu/project# git add *
root@ip-172-31-11-135:/home/ubuntu/project# git commit -m "updated code"
[main 9dd208f] updated code
Committer: root <root@ip-172-31-11-135.ap-south-1.compute.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

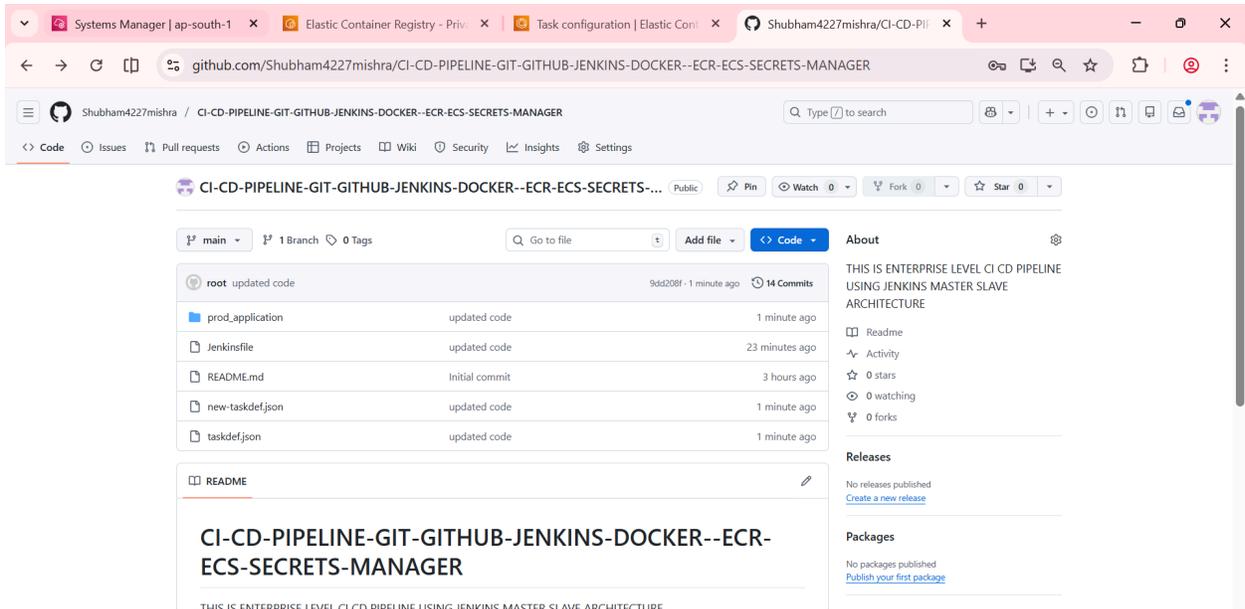
    git config --global --edit

After doing this, you may fix the identity used for this commit with:

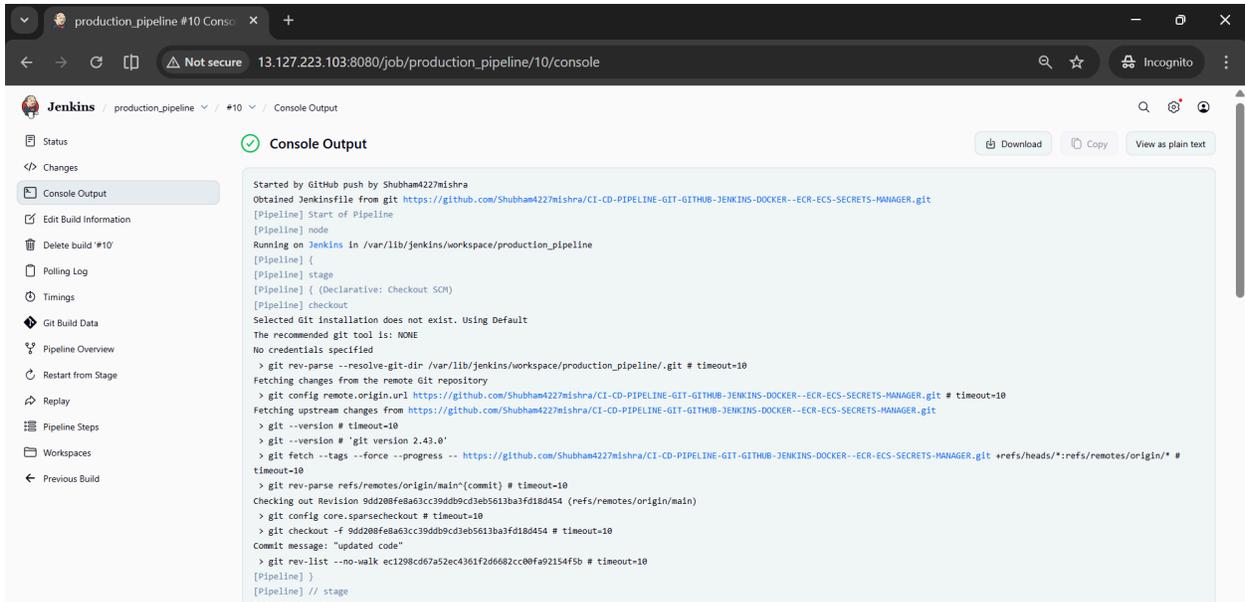
    git commit --amend --reset-author

3 files changed, 140 insertions(+), 37 deletions(-)
root@ip-172-31-11-135:/home/ubuntu/project# git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 2.19 KiB | 2.19 MiB/s, done.
Total 6 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Shubham4227mishra/CI-CD-PIPELINE-GIT-GITHUB-JENKINS-DOCKER--ECR-ECS-SECRETS-MANAGER.git
   ec1298c..9dd208f  main -> main
root@ip-172-31-11-135:/home/ubuntu/project#
```

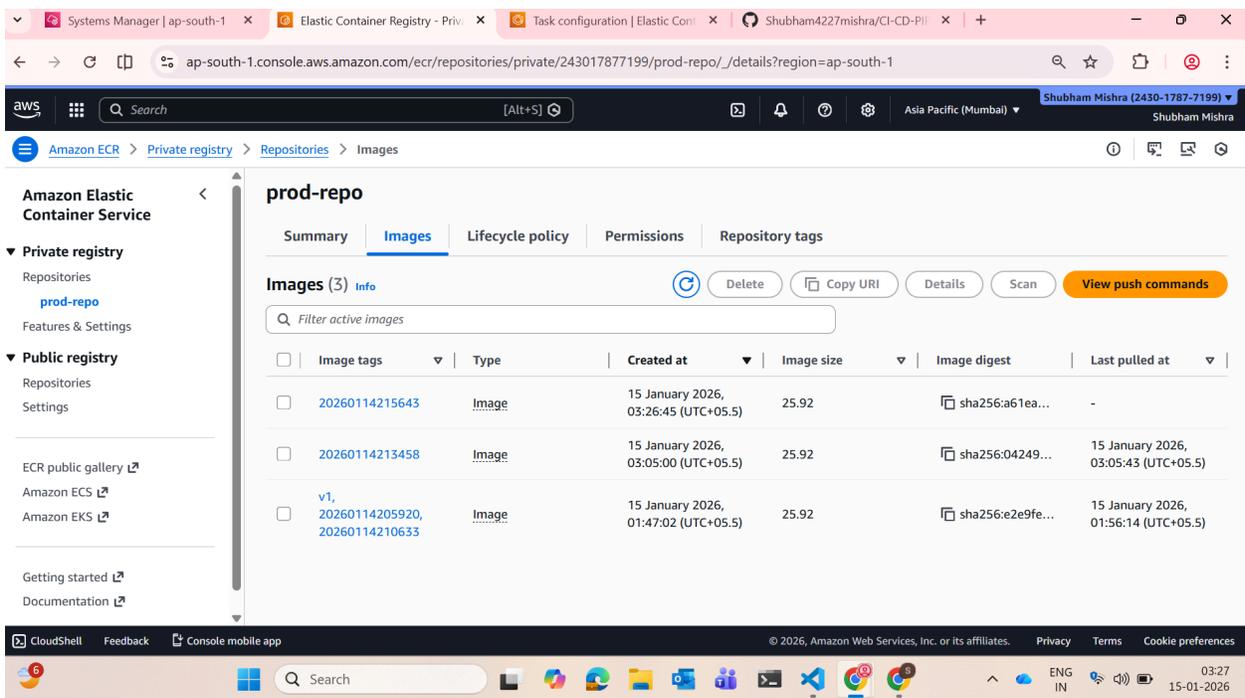
Step 34) As soon as new code pushed we can see it in github



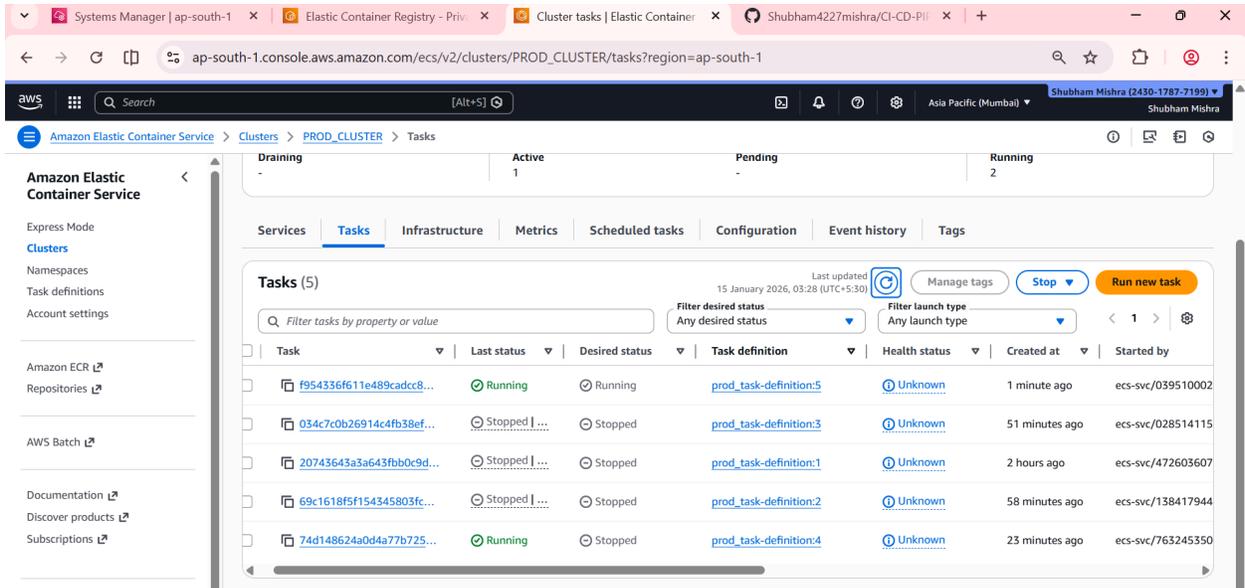
Step 35) Now jenkins detected updated code in github so it started job automatically



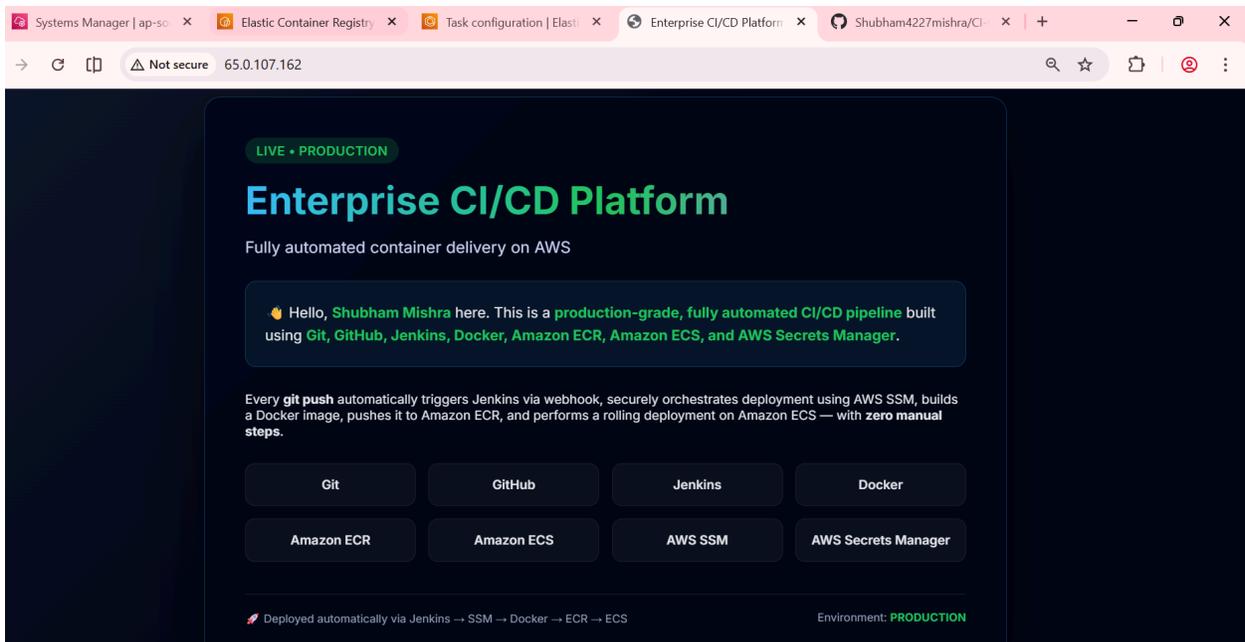
Step 36) New image created automatically



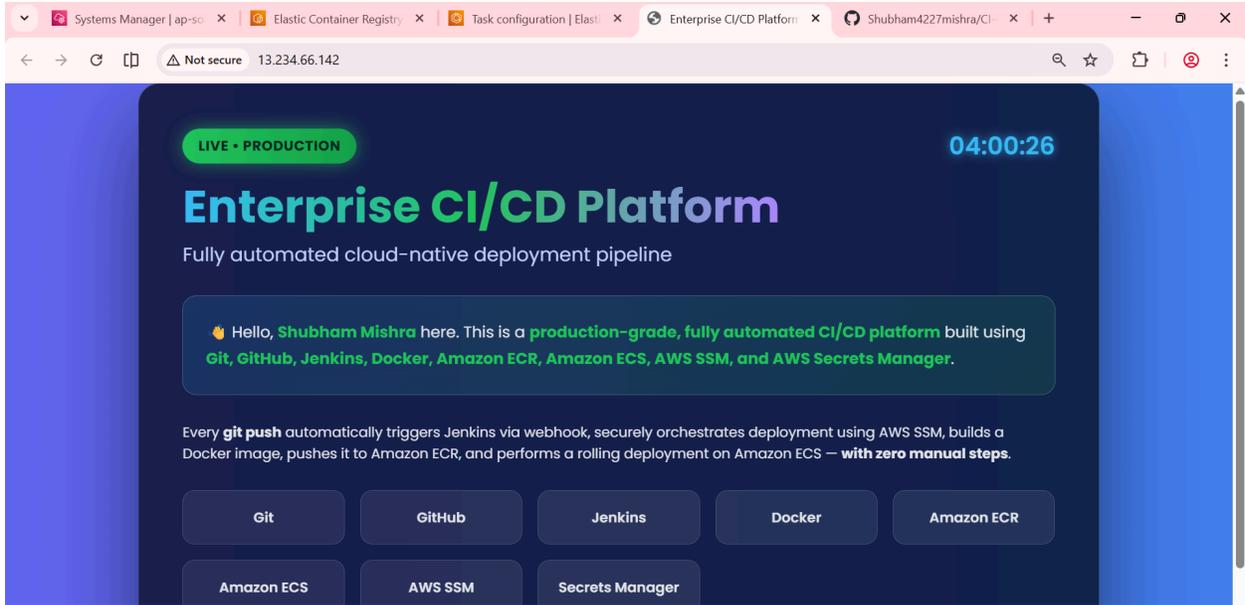
Step 37) New task updated to new image and we can see new container started



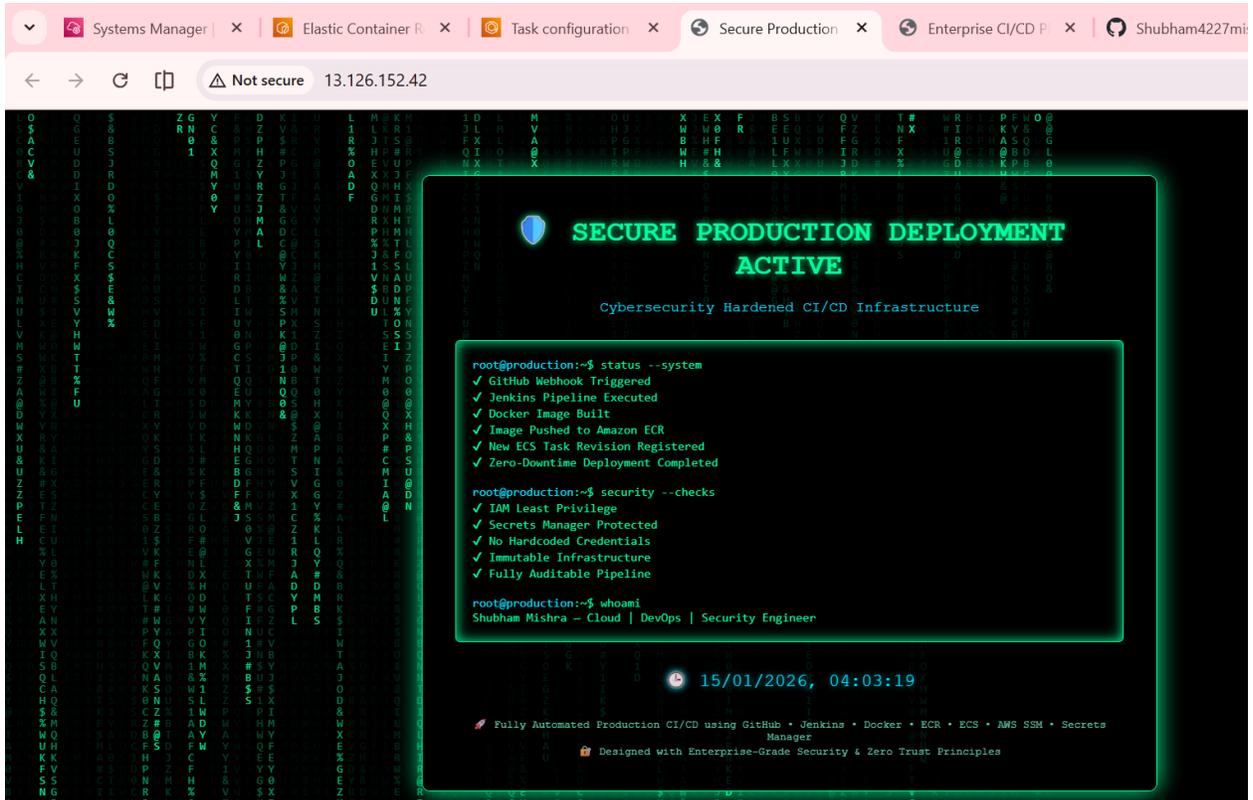
Step 38) Now we can the updated website on public ip of task

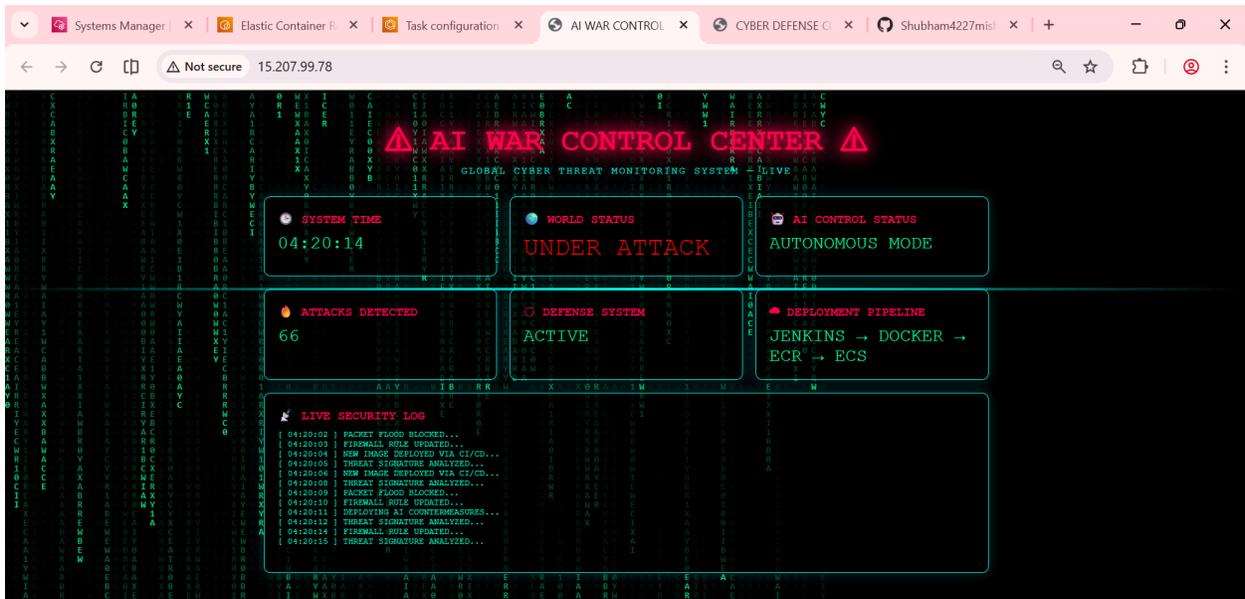
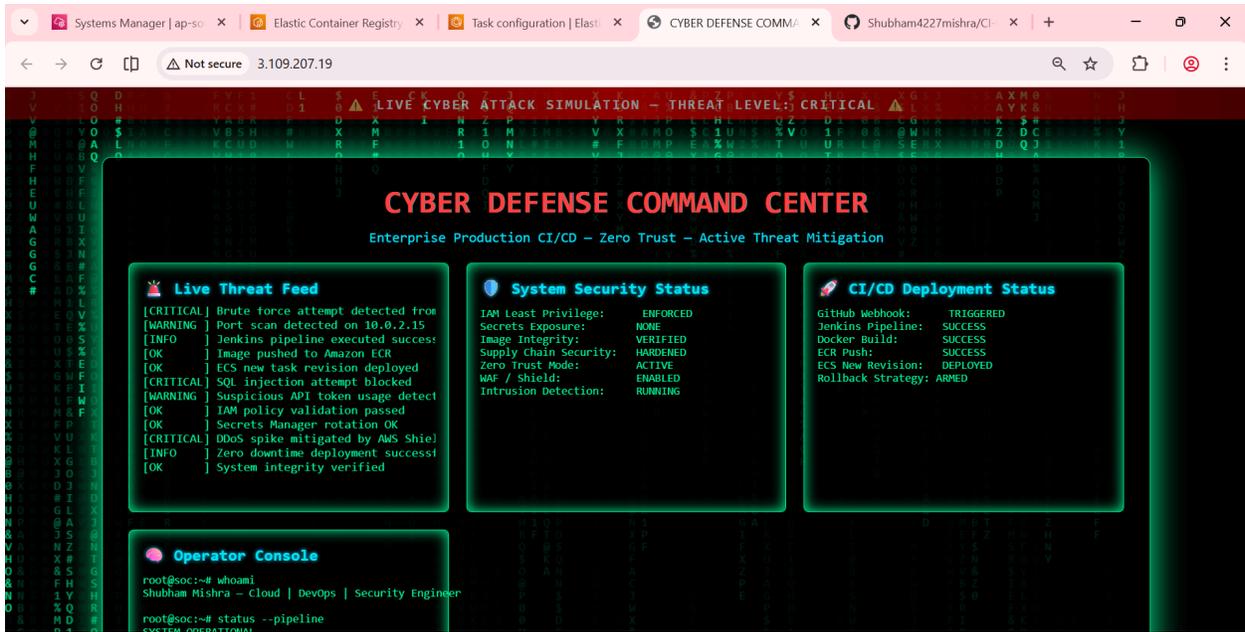


Step 39) Similarly again pushing new code we can see new website updated one

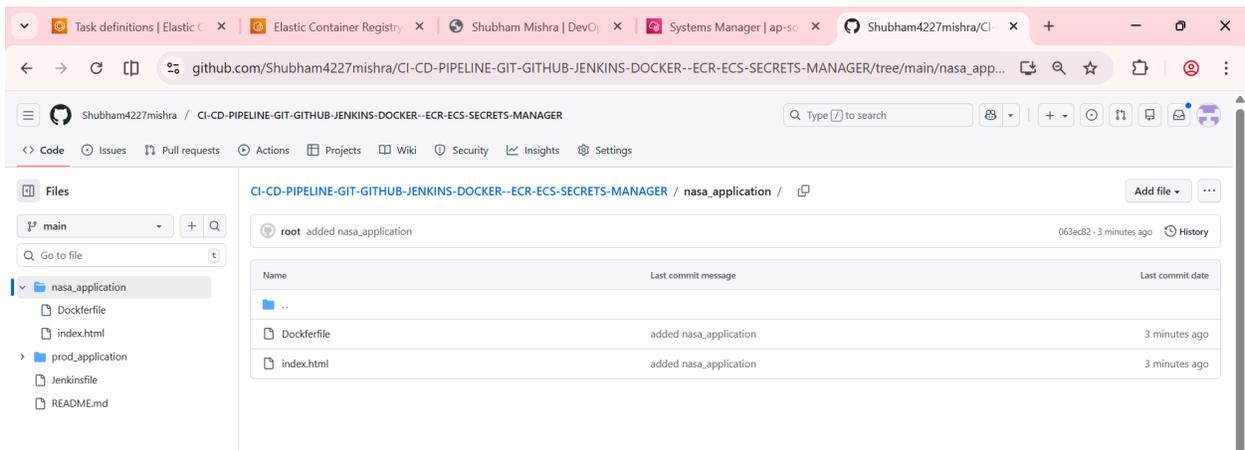
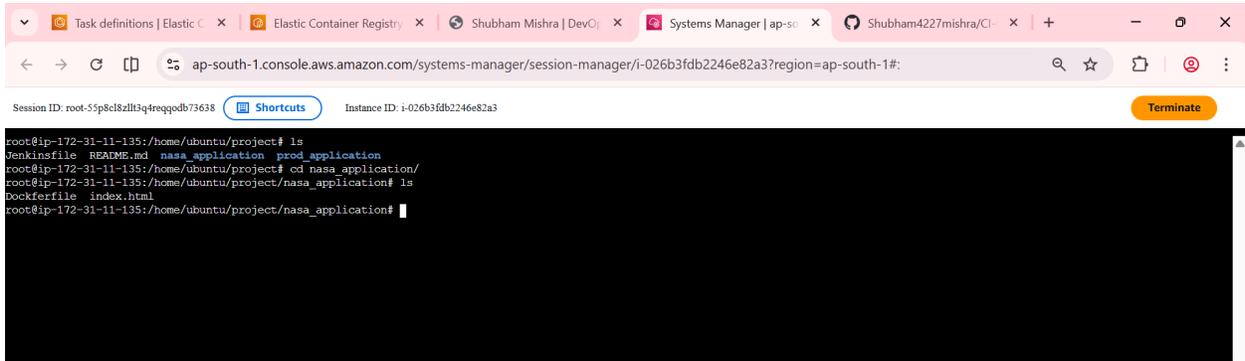


Step 40) Again pushed new code new website

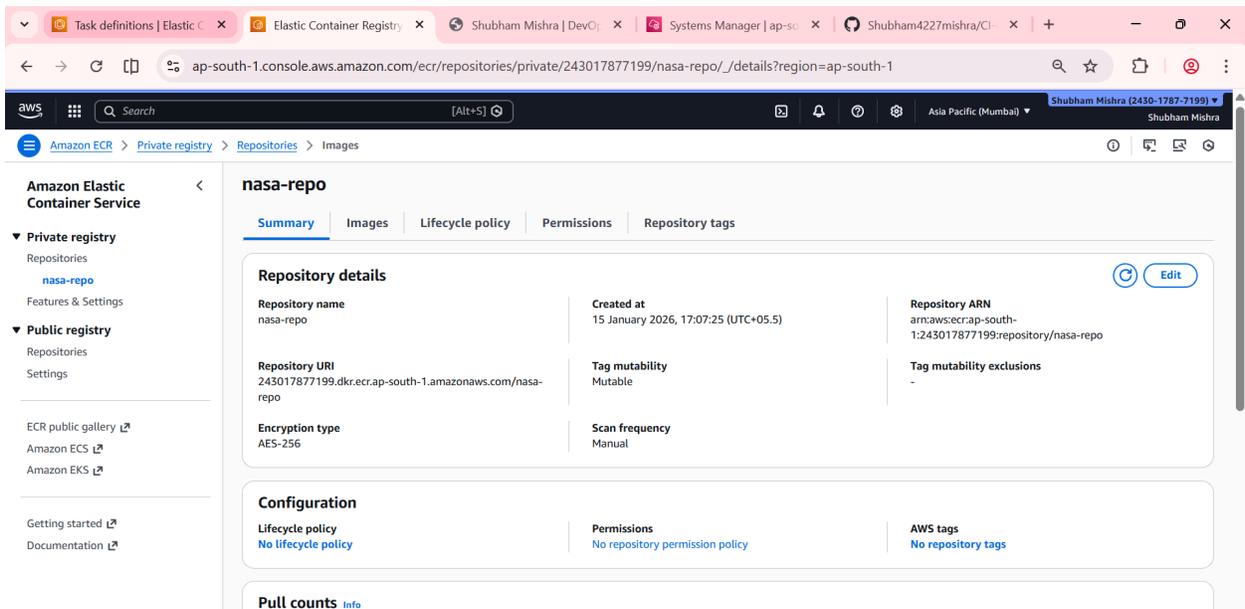




Step 41) Now we will add one more project in git and then push it into github



Step 42) Now we will create one more ECR for nasa_application

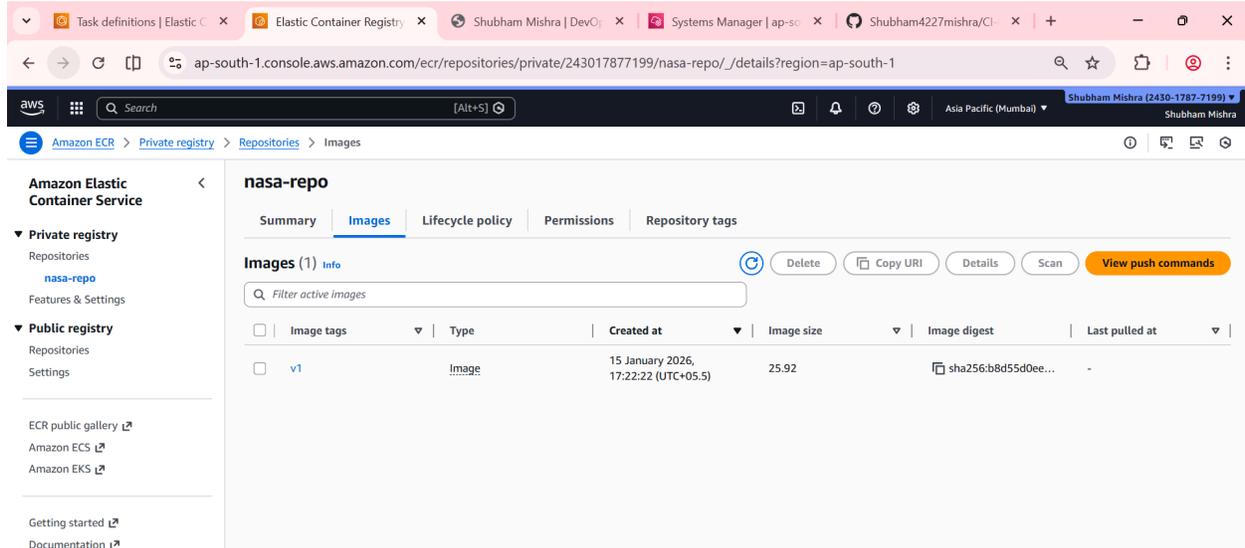


Step 43) We will login into ecr nasa-repo

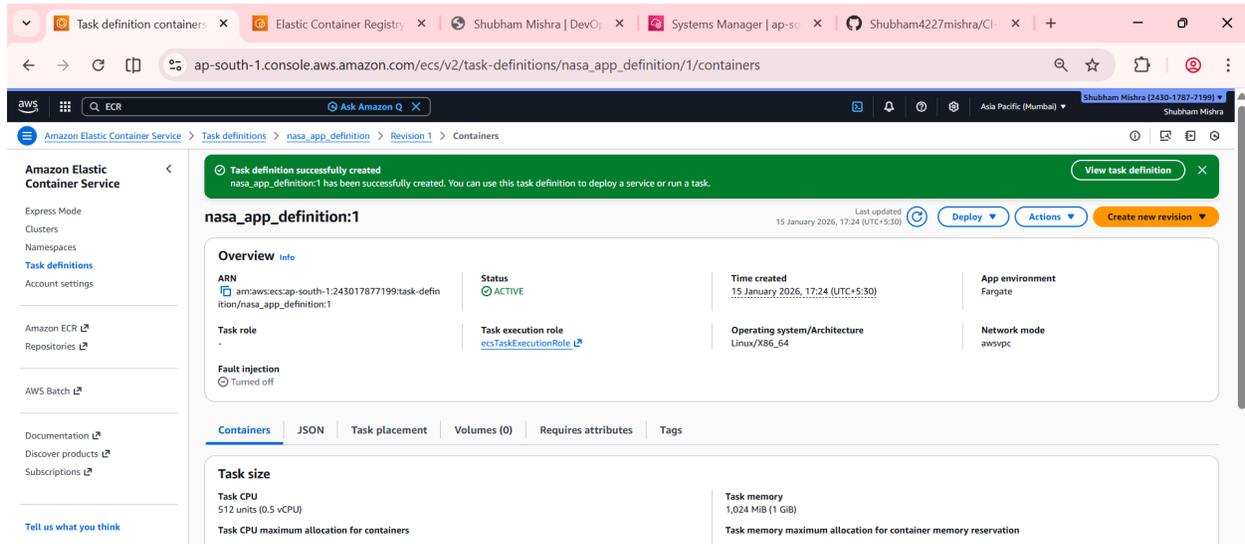
```
Run 'docker build --help' for more information
root@ip-172-31-11-135:/home/ubuntu/project/nasa_application# aws ecr get-login-password --region ap-south-1 | docker login --username AWS --password-stdin 243017877199.dkr.ecr.ap-south-1.amazonaws.com
WARNING! Your credentials are stored unencrypted in '/root/.docker/config.json'.
Configure a credential helper to remove this warning. See
https://docs.docker.com/go/credential-store/

Login Succeeded
root@ip-172-31-11-135:/home/ubuntu/project/nasa_application#
```

Step 44) Docker image pushed in ECR for nasa_application



Step 45) Task definition for nasa_application



Step 46) Created new service for nasa_application to use nasa_application task definition

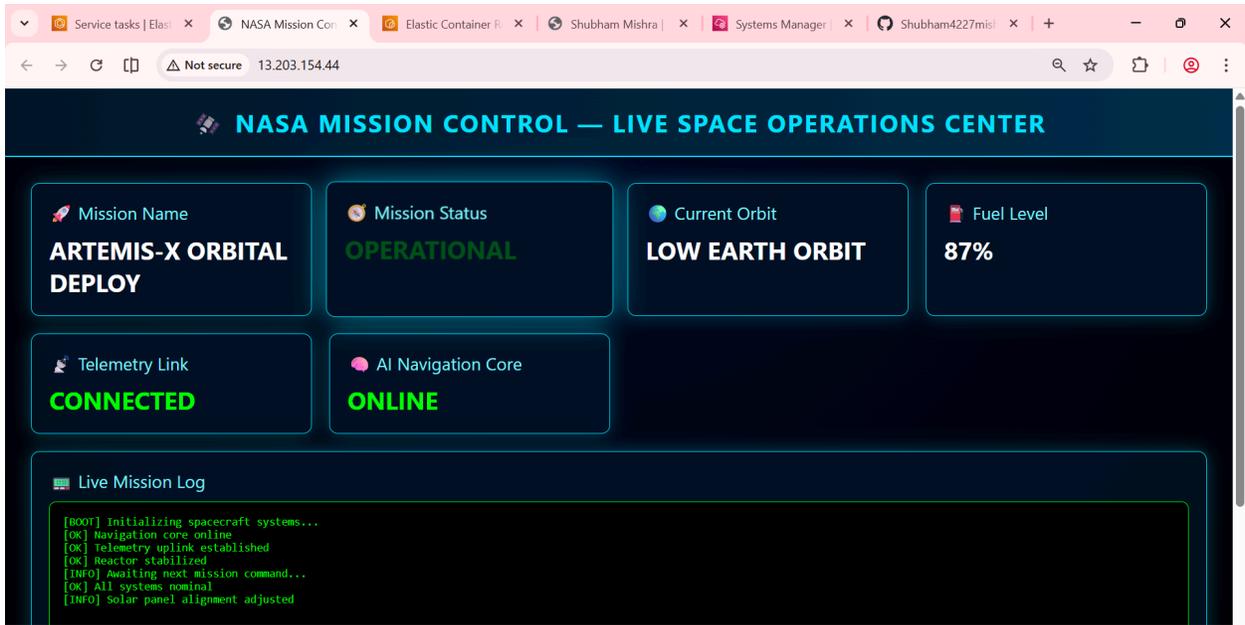
The screenshot shows the AWS Management Console for the service 'nasa_app_definition-service-292a50m3'. The status is 'Active' and the deployment is 'In progress'. The 'Tasks' section shows 1 desired, 0 pending, and 0 running tasks. The 'Health and metrics' section shows 0 completed tasks.

Step 47) Now new task container created successfully for nasa_application

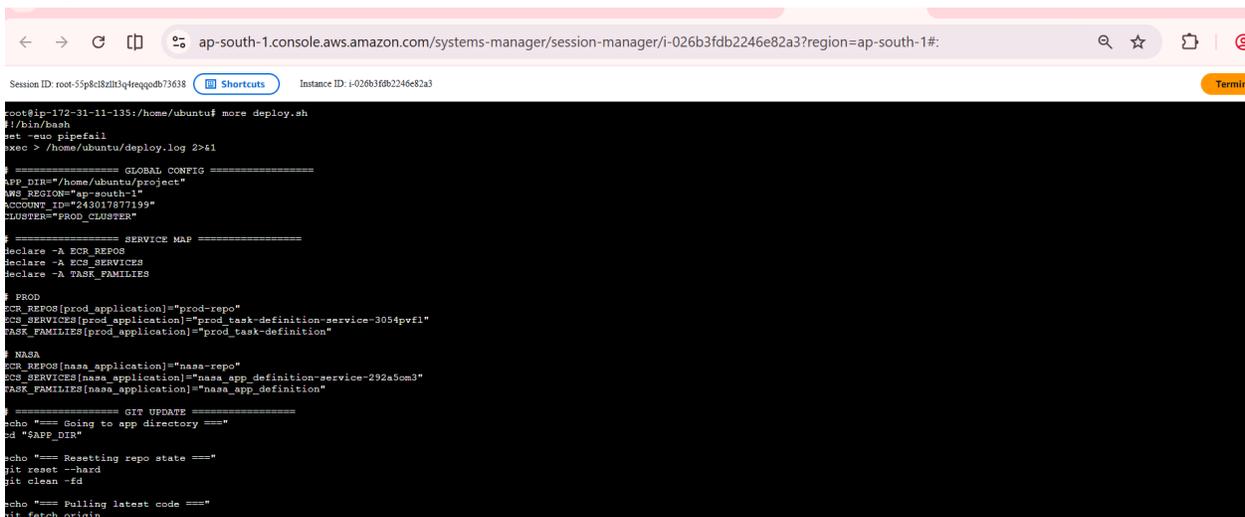
The screenshot shows the AWS Management Console for the service 'nasa_app_definition-service-292a50m3' with the 'Tasks' tab selected. The status is 'Active' and the deployment is 'In progress'. The 'Tasks' section shows 1/1 task running successfully.

Task	Last status	Desired status	Task definition	Health status	Created at	Started by	Started at	Container ins
a619a6ff8f964d6399ee5...	Running	Running	nasa_app_definition:1	Unknown	1 minute ago	ecs-svc/12924303006...	46 seconds ago	-

Step 48) We can verify the nasa_application website



Step 49) Now we will update [deploy.sh](#) and Jenkinsfile to fully automate this on new code changes it should build new task



```
root@ip-172-31-11-195:/home/ubuntu# ls
deploy.log  deploy.sh  project
root@ip-172-31-11-195:/home/ubuntu# cd project/
root@ip-172-31-11-195:/home/ubuntu/project# ls
Jenkinsfile  README.md  nasa_application  prod_application
root@ip-172-31-11-195:/home/ubuntu/project# more Jenkinsfile
pipeline {
  agent any

  environment {
    AWS_REGION = "ap-south-1"
    INSTANCE_ID = "i-026b3fdb2246e82a3"
  }

  stages {
    stage("Checkout") {
      steps {
        checkout scm
      }
    }

    stage("Deploy via SAM") {
      steps {
        sh '''
        set -e

        echo "Triggering deployment via SAM..."

        COMMAND_ID=$(aws sam send-command \
          --instance-ids $INSTANCE_ID \
          --document-name AWS-RundbShellScript \
          --parameters commands="/home/ubuntu/deploy.sh" \
          --region $AWS_REGION \
          --query "Command.CommandId" \
          --output text)

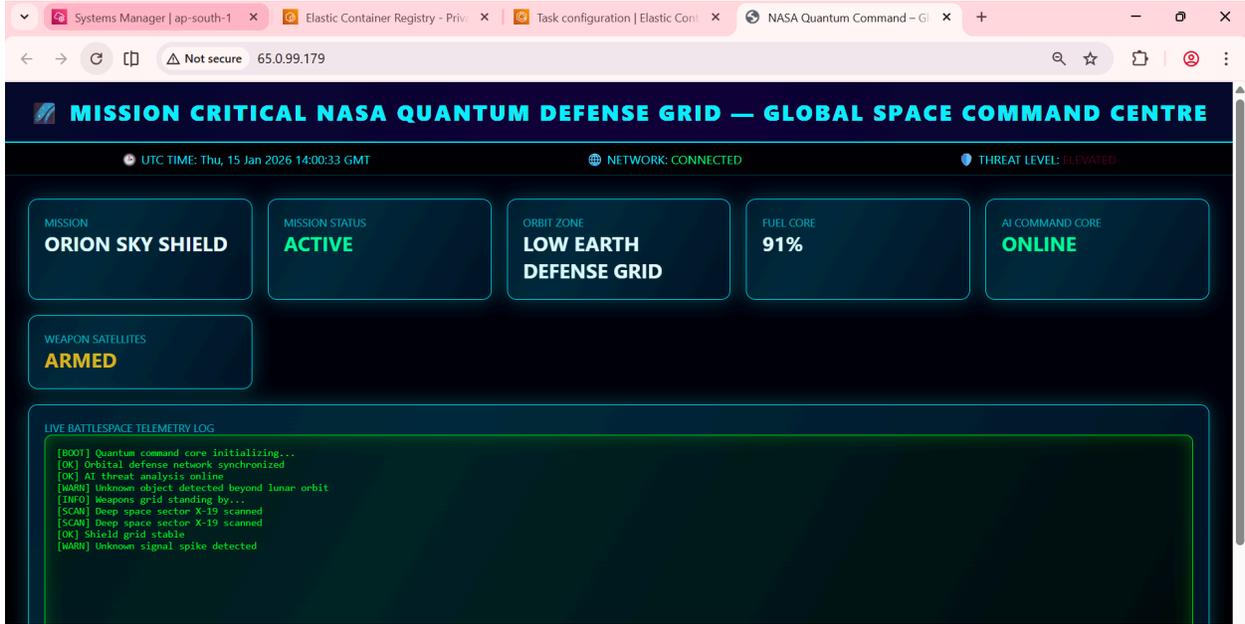
        echo "SAM Command ID: $COMMAND_ID"

        echo "Waiting for SAM command to finish..."

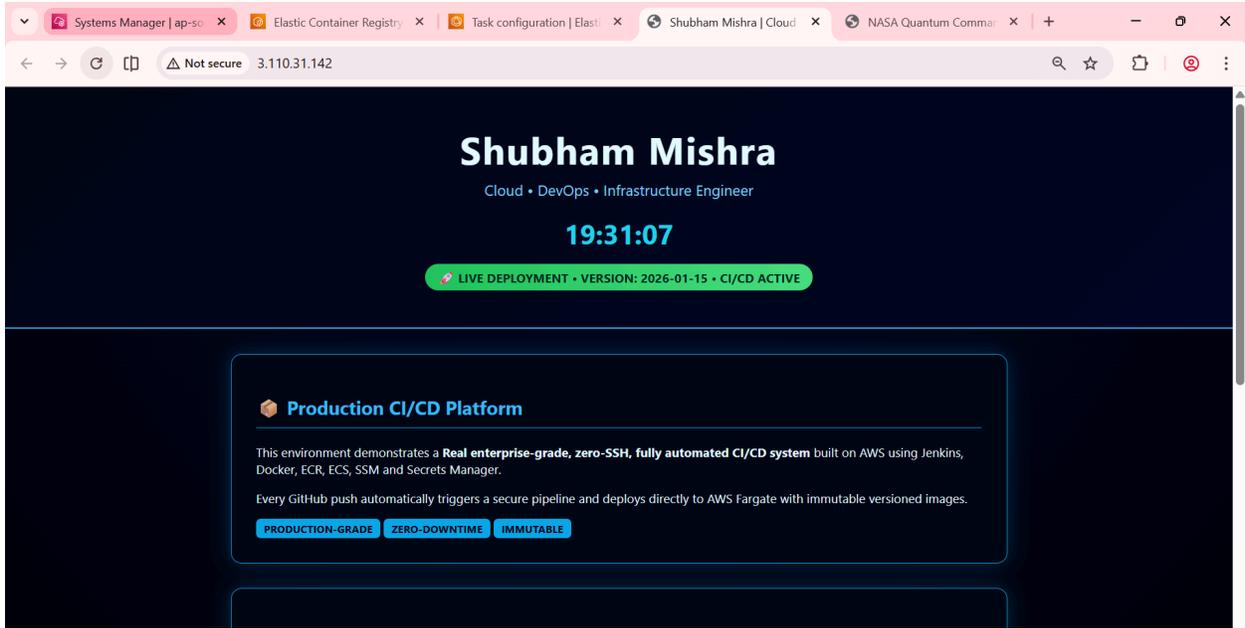
        STATUS="InProgress"

```

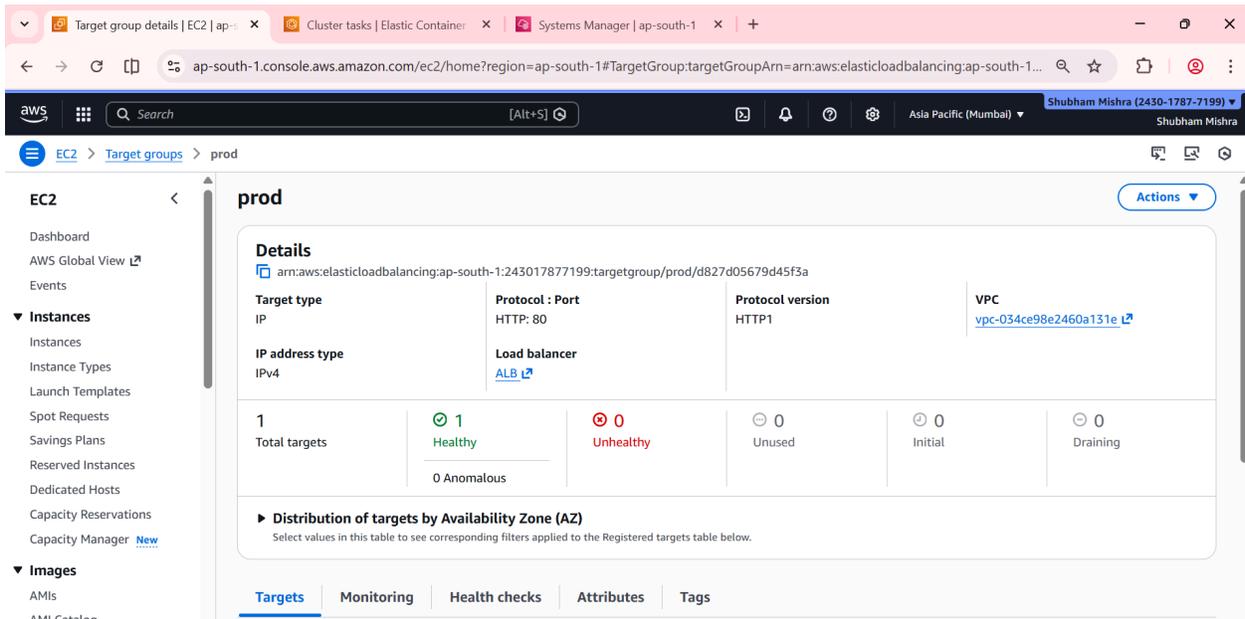
Step 50) Now we will make changes in nasa_application code and check the fully automated pipeline



Step 51) Also making change in prod application also we can see the automated deployment



Step 52) Now we will create target group for nasa and prod application



The screenshot shows the AWS Management Console interface for a Target Group named 'nasa'. The breadcrumb navigation is 'EC2 > Target groups > nasa'. The main content area displays the following details:

- Target type:** IP
- Protocol : Port:** HTTP: 80
- Protocol version:** HTTP1
- VPC:** vpc-034ce98e2460a131e
- IP address type:** IPv4
- Load balancer:** ALB

Below the details, there is a summary row showing:

- 1 Total targets
- 1 Healthy (0 Anomalous)
- 0 Unhealthy
- 0 Unused
- 0 Initial
- 0 Draining

A section titled 'Distribution of targets by Availability Zone (AZ)' is present, with a note: 'Select values in this table to see corresponding filters applied to the Registered targets table below.' Below this, there are tabs for 'Targets', 'Monitoring', 'Health checks', 'Attributes', and 'Tags'. At the bottom, it shows 'Registered targets (1)' with an 'Anomaly mitigation: Not applicable' status and buttons for 'Deregister' and 'Register targets'.

Step 53) Now we will create Application Load Balancer and add this target group

The screenshot shows the AWS Management Console interface for an Application Load Balancer (ALB). The breadcrumb navigation is 'EC2 > Load balancers > ALB'. A notification banner at the top reads 'Introducing ALB target optimizer'. The main content area displays the following details:

- Load balancer type:** Application
- Status:** Active
- VPC:** vpc-034ce98e2460a131e
- Load balancer IP address type:** IPv4
- Scheme:** Internet-facing
- Hosted zone:** ZP97RAFXTNZK
- Availability Zones:** subnet-081dd4b4720559bf4 (ap-south-1a (aps1-az1)), subnet-0f2ee6f9de8bb1028 (ap-south-1b (aps1-az3))
- Date created:** January 16, 2026, 11:18 (UTC+05:30)
- Load balancer ARN:** arn:aws:elasticloadbalancing:ap-south-1:243017877199:loadbalancer/app/ALB/94d9b19bc2ba8aaf
- DNS name:** ALB-798034012.ap-south-1.elb.amazonaws.com (A Record)

Below the details, there are tabs for 'Listeners and rules', 'Network mapping', 'Resource map', 'Security', 'Monitoring', 'Integrations', 'Attributes', 'Capacity', and 'Tags'. The 'Listeners and rules (1)' section is active, showing a table with the following information:

Protocol:Port	Default action	Rules	ARN	Security policy	Default SSL/TLS certificate	mTLS	Trust store
HTTP:80	Return fixed response Response code: 404	3 rules	ARN	Not applicable	Not applicable	Not applicable	Not applicable

Step 54) Now we will create rules on port 80 for Application load balancer bases on host header based routing

The screenshot shows the AWS Management Console interface for an HTTP-80 listener. The left sidebar contains navigation options like Dashboard, Instances, Images, Elastic Block Store, Network & Security, and Load Balancing. The main content area displays the 'HTTP-80' listener details, including its ARN and default actions. Below this, the 'Listener rules' section shows a table of rules:

Priority	Name tag	Conditions (If)	Transforms	Actions (Then)	ARN	Actions
100		Host header (value) = prod.shubham-mishra.online		Forward to target group (target group: prod-shubham-mishra-on-line)	ARN	Forward to target group
200		Host header (value) = prod.shubham-mishra.online		Forward to target group (target group: prod-shubham-mishra-on-line)	ARN	Forward to target group
Last (default)	Default	If no other rule applies		Return fixed response (Response code: 404, Response body: PAGE NOT FOUND, Response content type: text/plain)	ARN	Return fixed response

Step 55) Now we will add this ALB as CNAME record in route53 public hosted zone in other aws account so to route traffic based on the host header

The screenshot shows the AWS Route 53 console. The main area displays a list of records with the following columns: Record name, Type, Routing policy, Difference, Alias, Value/Route traffic to, and TTL (s...). The record 'prod.shubha...' is selected. The 'Record details' panel on the right shows the following information:

- Record name: prod.shubham-mishra.online
- Record type: CNAME
- Value: ALB-798034012.ap-south-1.elb.amazonaws.com
- Alias: No
- TTL (seconds): 300
- Routing policy: Simple

The screenshot shows the AWS Route 53 console for the hosted zone 'shubham-mishra.online'. The 'Records' section displays a table with 12 records. The record 'nasa.shubham...' is selected, and its details are shown on the right.

Record name	Type	Routin...	Differ...	Alias	Value/Route traffic to	TTL (s...)
<input type="checkbox"/> ec2-alb-docke...	CNAME	Simple	-	No	CLOUDZENIA-ALB-34356288...	300
<input type="checkbox"/> ec2-alb-instan...	CNAME	Simple	-	No	CLOUDZENIA-ALB-34356288...	300
<input type="checkbox"/> ec2-docker1.s...	CNAME	Simple	-	No	CLOUDZENIA-ALB-34356288...	300
<input type="checkbox"/> ec2-docker2.s...	CNAME	Simple	-	No	CLOUDZENIA-ALB-34356288...	300
<input type="checkbox"/> ec2-instance1...	CNAME	Simple	-	No	CLOUDZENIA-ALB-34356288...	300
<input type="checkbox"/> ec2-instance2...	CNAME	Simple	-	No	CLOUDZENIA-ALB-34356288...	300
<input type="checkbox"/> microservice.s...	CNAME	Simple	-	No	CLOUDZENIA-ALB-34356288...	300
<input checked="" type="checkbox"/> nasa.shubham...	CNAME	Simple	-	No	ALB-798034012.ap-south-1....	300
<input type="checkbox"/> prod.shubha...	CNAME	Simple	-	No	ALB-798034012.ap-south-1....	300
<input type="checkbox"/> wordpress.shu...	CNAME	Simple	-	No	CLOUDZENIA-ALB-34356288...	300
<input type="checkbox"/> www.shubha...	A	Simple	-	Yes	dqzfkzopmjw8.cloudfront.net.	-

Record details for 'nasa.shubham...':

- Record name: nasa.shubham-mishra.online
- Record type: CNAME
- Value: ALB-798034012.ap-south-1.elb.amazonaws.com
- Alias: No
- TTL (seconds): 300
- Routing policy: Simple

Step 56) Now we will verify the setup based on host header it should go to particular ecs task

The screenshot shows a web browser displaying the production CI/CD platform for Shubham Mishra. The page features a dark blue background with white and light blue text.

Shubham Mishra
 Cloud • DevOps • Infrastructure Engineer

11:37:54

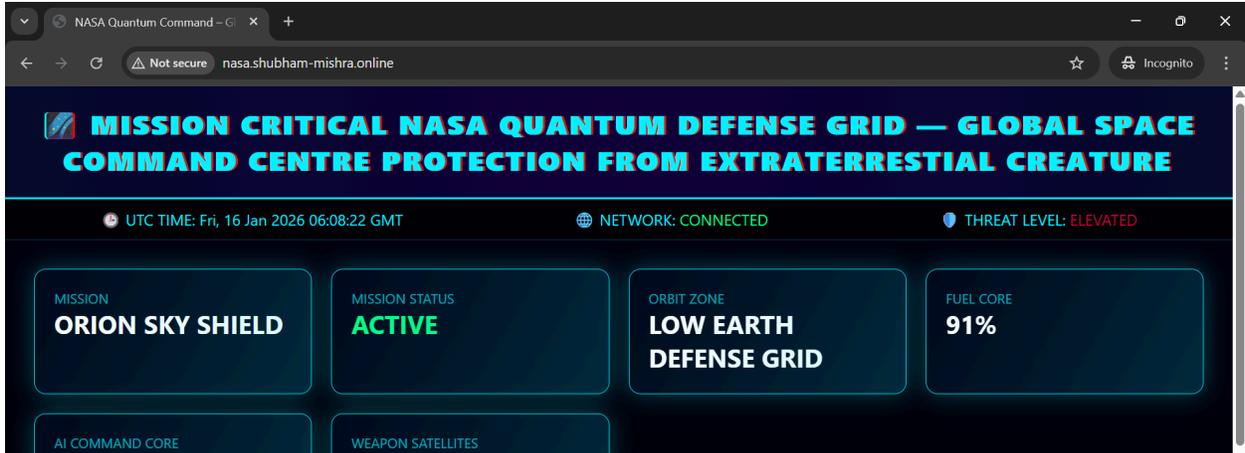
LIVE DEPLOYMENT • VERSION: 2026-01-15 • CI/CD ACTIVE

Production CI/CD Platform

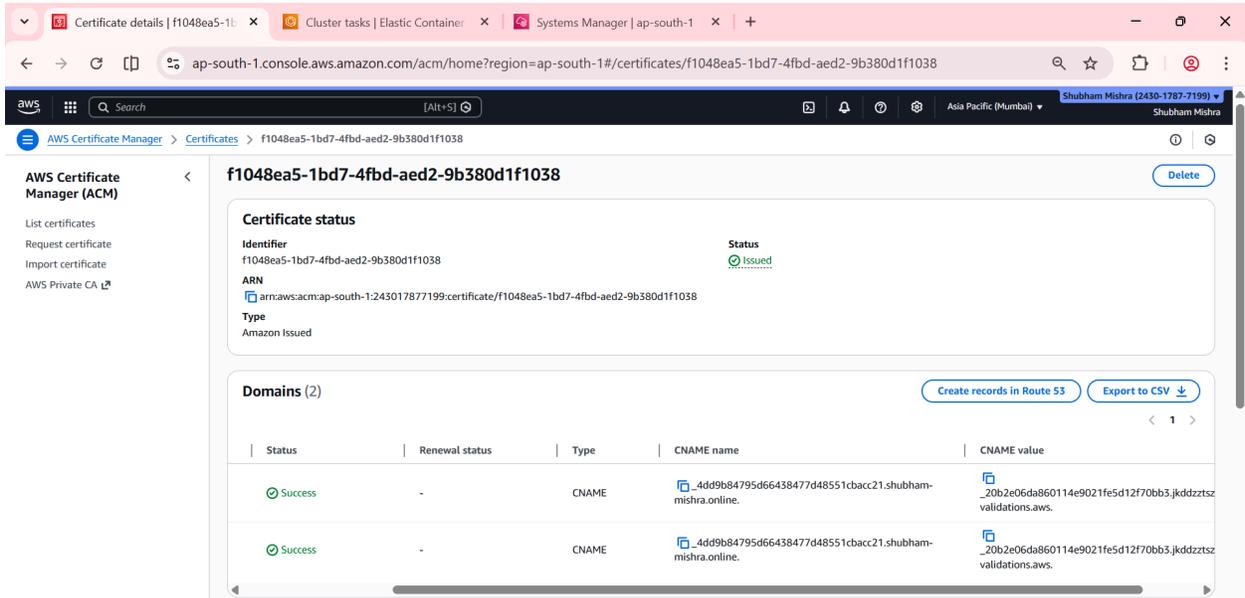
This environment demonstrates a **Real fintech enterprise-grade, zero-SSH, fully automated CI/CD system** built on AWS using Jenkins, Docker, ECR, ECS, SSM and Secrets Manager.

Every GitHub push automatically triggers a secure pipeline and deploys directly to AWS Fargate with immutable versioned images.

PRODUCTION-GRADE **ZERO-DOWNTIME** **IMMUTABLE**



Step 57) Now we will create SSL certificate and add the records in route53 to use this certificate



The screenshot shows the AWS Route 53 Hosted Zones console. The left sidebar contains navigation options for Route 53, Global Resolver, VPC Resolver, and Domains. The main area displays 'Records (1/26)' with a table of records. One record is selected, and its details are shown on the right.

Record	Type	Routin...	Differ...	Alias	Value/Route traffic to
<input type="checkbox"/> shubham-...	NS	Simple	-	No	ns-444.awsdns-55.com ns-1852.awsdns-39.co.uk ns-1491.awsdns-58.org ns-978.awsdns-58.net
<input type="checkbox"/> shubham-...	SOA	Simple	-	No	ns-444.awsdns-55.com. av
<input type="checkbox"/> _350dc74...	CNAME	Simple	-	No	_b81b891071f3ab60a4e4
<input checked="" type="checkbox"/> _4dd9b84...	CNAME	Simple	-	No	_20b2e06da860114e9021
<input type="checkbox"/> _7017931...	CNAME	Simple	-	No	_2775608888ccf4dba500
<input type="checkbox"/> _9ca0638...	CNAME	Simple	-	No	_63ee656e0d59e234b5c

Record details:
 Record name: _4dd9b84795d66438477d48551cb
 Record type: CNAME
 Value: _20b2e06da860114e9021fe5d12f70bb3.jkddztszm.acm-validations.aws.
 Alias: No
 TTL (seconds):

Step 58) Now we will add https 443 listener and add this ssl certificate over there

The screenshot shows the AWS Elastic Load Balancing console for an HTTPS:443 listener. The left sidebar shows navigation options for Elastic Block Store, Network & Security, and Load Balancing. The main area displays the listener details.

HTTPS:443 Details

A listener checks for connection requests using the protocol and port that you configure. The default action and any additional rules that you create determine how the Application Load Balancer routes requests to its registered targets.

Protocol:Port HTTPS:443	Load balancer ALB	Default SSL/TLS certificate shubham-mishra.online (Certificate ID: f1048ea5-1bd7-4fbd-aed2-9b380d1f1038)
-----------------------------------	---	---

Default actions

- Forward to target group
 - [nasa](#): 1 (50%)
 - [prod](#): 1 (50%)

Target group stickiness: Off

Listener ARN
[arn:aws:elasticloadbalancing:ap-south-1:243017877199:listener/app/ALB/94d9b19bc2ba8aaf/00bf6624990e4d0d](#)

Listener rules (3) info

Traffic received by the listener is routed according to the default action and any additional rules. Rules are evaluated in priority order from the lowest value to the highest value.

Filter rules

Priority	Name tag	Conditions (If)	Transforms	Actions
100	-	Host header (value) = prod.shubham-mishra.online	-	[Edit] [Delete]
200	-	Host header (value) = nasa.shubham-mishra.online	-	[Edit] [Delete]
Last (default)	Default	If no other rule applies	-	[Edit] [Delete]

Step 59) Now we will also redirect http request to https

Successfully modified listener.

HTTP:80 info

Details

A listener checks for connection requests using the protocol and port that you configure. The default action and any additional rules that you create determine how the Application Load Balancer routes requests to its registered targets.

Protocol/Port: HTTP:80

Load balancer: ALB

Default actions:

- Redirect to HTTPS
- Redirect to HTTP_301

Listener ARN: arn:aws:elasticloadbalancing:ap-south-1:24301787199:listener/app/ALB/94d9b19c2b8baaf3c5610a685ab0a1

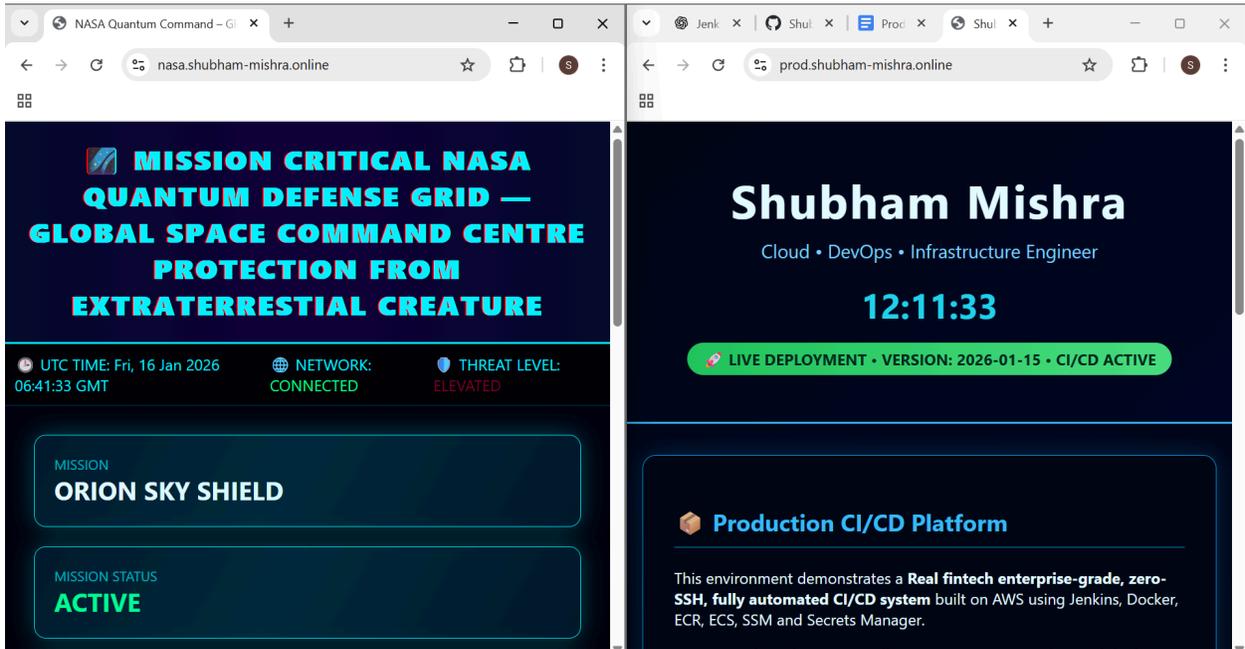
Rules (3) info

Traffic received by the listener is routed according to the default action and any additional rules. Rules are evaluated in priority order from the lowest value to the highest value.

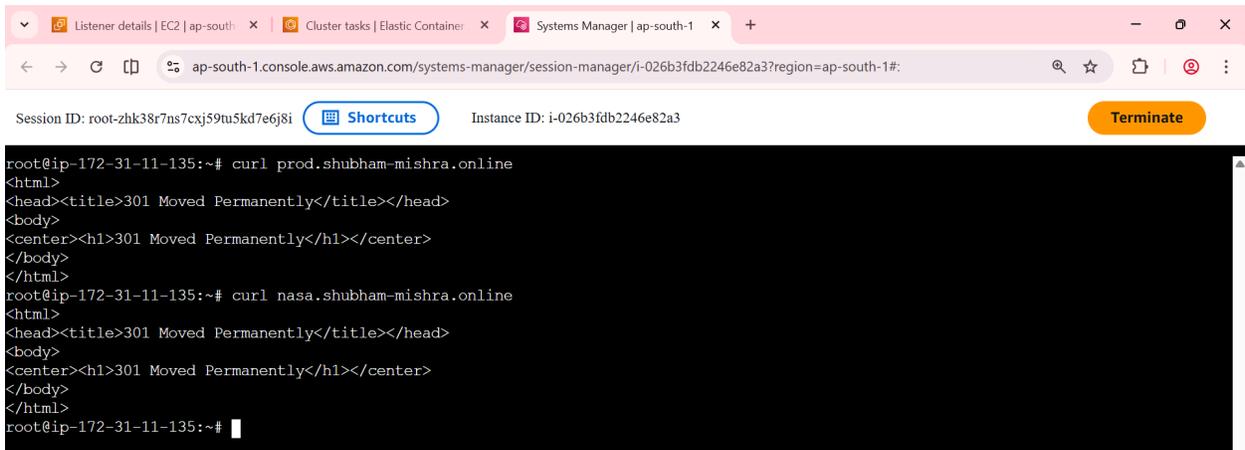
Filter rules

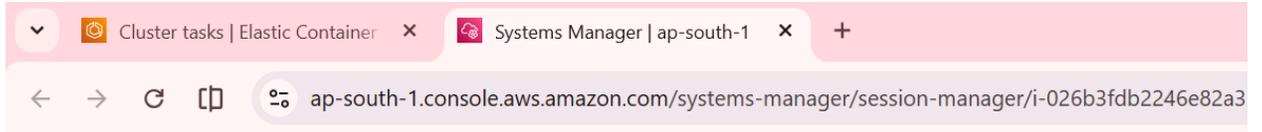
Priority	Name tag	Conditions (If)	Transforms	Actions (Then)	ARN	Ttl	Actions
100	-	Host header (value) = nasa.shubham-mishra.online	-	Redirect to HTTPS	ARN	0:5	[Edit] [Delete]
200	-	Host header (value) = prod.shubham-mishra.online	-	Redirect to HTTP	ARN	0:5	[Edit] [Delete]
Last (default)	Default	If no other rule applies	-	Redirect to HTTPS	ARN	0:5	[Edit] [Delete]

Step 60) Now we will test whether our ssl setup is working corectly and also http request is getting redirected to https



Also see the http request is not redirected to https





Session ID: root-zhk38r7ns7cxj59tu5kd7e6j8i

[Shortcuts](#)

Instance ID: i-026b3fdb2246e82a3

```
root@ip-172-31-11-135:/home/ubuntu/project# git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   nasa_application/index.html
        modified:   prod_application/index.html

no changes added to commit (use "git add" and/or "git commit -a")
root@ip-172-31-11-135:/home/ubuntu/project#
```

b) Now we will commit and push this changes in github

```
        modified:   prod_application/index.html

no changes added to commit (use "git add" and/or "git commit -a")
root@ip-172-31-11-135:/home/ubuntu/project# git add *
root@ip-172-31-11-135:/home/ubuntu/project# git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   nasa_application/index.html
        modified:   prod_application/index.html

root@ip-172-31-11-135:/home/ubuntu/project#
```

```
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   nasa_application/index.html
        modified:   prod_application/index.html

root@ip-172-31-11-135:/home/ubuntu/project# git commit -m "updated code for both nasa and prod application"
[main 6d8915f] updated code for both nasa and prod application
  Committer: root <root@ip-172-31-11-135.ap-south-1.compute.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

2 files changed, 154 insertions(+), 158 deletions(-)
```

```
root@ip-172-31-11-135:/home/ubuntu/project# git commit -m "updated code for both nasa and prod application"
[main 6d8915f] updated code for both nasa and prod application
  Committer: root <root@ip-172-31-11-135.ap-south-1.compute.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

2 files changed, 154 insertions(+), 158 deletions(-)
root@ip-172-31-11-135:/home/ubuntu/project# git push origin main
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 3.48 KiB | 3.48 MiB/s, done.
Total 6 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Shubham4227mishra/CI-CD-PIPELINE-GIT-GITHUB-JENKINS-DOCKER--ECR-ECS-SECRETS-MANAGER.git
   2b2bdef..6d8915f  main -> main
root@ip-172-31-11-135:/home/ubuntu/project#
```

c) Now jenkins job started running

Jenkins / production_pipeline / #45

Status: ✔ #45 (16 Jan 2026, 06:51:29) Progress: Add description Keep this build forever

Changes

Console Output

Edit Build Information

Polling Log

Timings

Git Build Data

Pipeline Overview

Thread Dump

Pause/resume

Started by GitHub push by Shubham4227mishra Started 30 sec ago
Build has been executing for 30 sec

This run spent 9.9 sec waiting in the queue.

git Revision: 6d8915fca519bf70ee987f42c0f95208a523bbc8
Repository: <https://github.com/Shubham4227mishra/CI-CD-PIPELINE-GIT-GITHUB-JENKINS-DOCKER--ECR-ECS-SECRETS-MANAGER.git>

- refs/remotes/origin/main

Changes

- updated code for both nasa and prod application ([details](#) / [githubweb](#))

d) Now we can see new images in ECR

Cluster services | Elastic Contain... | Elastic Container Registry - Priv... | Systems Manager | ap-south-1

ap-south-1.console.aws.amazon.com/ecr/repositories/private/243017877199/nasa-repo/_/details?region=ap-south-1

Amazon ECR > Private registry > Repositories > Images

Amazon Elastic Container Service

Private registry

- Repositories
 - nasa-repo
- Features & Settings

Public registry

- Repositories
- Settings

ECR public gallery

Amazon ECS

Amazon EKS

nasa-repo

Summary Images Lifecycle policy Permissions Repository tags

Images (2) Info Delete Copy URI Details Scan View push commands

Filter active images

<input type="checkbox"/>	Image tags	Type	Created at	Image size	Image digest	Last pulled at
<input type="checkbox"/>	20260116072053	Image	16 January 2026, 12:50:56 (UTC+05.5)	25.92	sha256:598e3...	-
<input type="checkbox"/>	20260116045444	Image	16 January 2026, 10:24:48 (UTC+05.5)	25.92	sha256:6bceb...	16 January 2026, 10:25:28 (UTC+05.5)

Amazon Elastic Container Service

Private registry

Repositories

prod-repo

Features & Settings

Public registry

Repositories

Settings

ECR public gallery

Amazon ECS

Amazon EKS

Getting started

Documentation

prod-repo

Summary | **Images** | Lifecycle policy | Permissions | Repository tags

Images (2) info

Filter active images

<input type="checkbox"/>	Image tags	Type	Created at	Image size	Image digest	Last pulled at
<input type="checkbox"/>	20260116072354	Image	16 January 2026, 12:53:57 (UTC+05.5)	25.92	sha256:ca8f41a83b3...	-
<input type="checkbox"/>	20260116070502	Image	16 January 2026, 12:35:05 (UTC+05.5)	25.92	sha256:3b4a5e16b4a...	16 January 2026, 12:35:46 (UTC+05.5)

e) Now new task is getting created using updated image

Amazon Elastic Container Service

Express Mode

Clusters

Namespaces

Task definitions

Account settings

Amazon ECR

Repositories

AWS Batch

Documentation

Discover products

Subscriptions

nasa_app_definition-service-292a5om3

Last updated 16 January 2026, 12:51 (UTC+5:30)

Delete service | Update service

Service overview

Status: Active

Tasks (1 Desired): 1 pending | 1 running

Task definition: revision nasa_app_definition:5

Deployment status: In progress

Health and metrics | **Tasks** | Logs | Deployments | Events | Configuration and networking | Service auto scaling | Event history | Tags

Tasks (1/2)

Filter tasks by property or value

Filter desired status: Any desired status

Filter launch type: Any launch type

<input type="checkbox"/>	Task	Last status	Desired status	Task definition	Health status	Created at	Started by
<input checked="" type="checkbox"/>	43faa063ca7743c9a28aef...	Running	Running	nasa_app_definition:4	Unknown	1 hour ago	ecs-svc/08152289766...
<input type="checkbox"/>	13e800ded5ed4e55ac1ab...	Activating	Running	nasa_app_definition:5	Unknown	20 seconds ago	ecs-svc/43207887050...

The screenshot shows the AWS Management Console for the service 'prod_task-definition-service-3054pvfl'. The service is in an 'Active' state with 1 pending and 1 running task. The task list shows two tasks: one with ID '824d93c3ea4b427b8de5...' in 'Activating' status and another with ID '9f595ad01b2349d8b744...' in 'Running' status. The console interface includes a sidebar with navigation options like 'Amazon Elastic Container Service', 'Amazon ECR', and 'AWS Batch'.

g) We can see the new changes visible to end user using domain and traffic is also getting routed

The screenshot displays a futuristic web application interface titled 'GLOBAL WAR COMMAND CENTER — RED ALERT MODE FOR EXTRA TERRESTRIAL CREATURES'. The interface features a dark theme with red and green accents. At the top, it shows 'UTC TIME: Fri, 16 Jan 2026 07:22:42 GMT', 'NETWORK: ONLINE', and 'THREAT LEVEL: CRITICAL'. Below this, there are several status panels: 'MISSION: IRON SKYFALL', 'MISSION STATUS: COMBAT MODE', 'DEFENSE ZONE: PLANETARY SHIELD GRID', 'REACTOR CORE: 97%', 'AI WAR CORE: ACTIVE', and 'WEAPONS PLATFORM: HOT'. The interface is designed to look like a high-tech command center.

GLOBAL WAR COMMAND - RE | Shubham Mishra | Cloud & Dev | prod.shubham-mishra.online

Shubham Mishra

Cloud • DevOps • Platform Architect

12:55:43

LIVE DEPLOYMENT • VERSION: PURPLE-NEON-2026 • CI/CD ACTIVE

Enterprise CI/CD Platform

This environment demonstrates a **bank-grade, zero-SSH, fully automated CI/CD platform** built on AWS using Jenkins, Docker, ECR, ECS, SSM and **Secrets Management**.

