JENKINS CI/CD PIPELINE WITH DOCKER AND EKS

(JENKINS + MAVEN + DOCKER + KUBERNETES (EKS))

Project Documentation

Prepared By: Jenifa R

Date: 29 January 2026

❖ PROJECT DETAILS

Item	Details
Project Title	End to End CI/CD Pipeline (Jenkins + Maven + Docker + Kubernetes EKS)
Domains	DevOps / CI-CD Automation
Application Type	Java Maven Application (.jar → Docker Image)
Source Code Repository	GitHub – Maven Web App (example: https://github.com/suffixscope/maven-web-app)
Deployment Target	Kubernetes Deployment on Amazon EKS
Cloud Platform	AWS (EC2 for Jenkins, EKS for deployment)
Pipeline Tool	Jenkins
Build Tool	Apache Maven
Containerization	Docker
Orchestration	Kubernetes (Amazon EKS)
Artifact Repository	Docker Hub

Tool	Version
Jenkins	2.541.1

Tool	Version
Java (OpenJDK)	17.0.17
Maven	3.8.4
Eksctl	0.221.0
Docker	25.0.14

❖ OBJECTIVE

Implement a complete CI/CD workflow for a Maven-based Java web application to:

- Automatically pull source code from GitHub whenever developers push changes.
- Build and package the project using Apache Maven to generate a .war file.
- Containerize the application by creating a Docker image from the .war file.
- Push the Docker image to Docker Hub for centralized storage and versioning.
- Deploy the containerized application seamlessly into a Kubernetes cluster (Amazon EKS).
- Verify successful deployment by checking running pods, services, and accessing the application via the AWS Load Balancer URL.

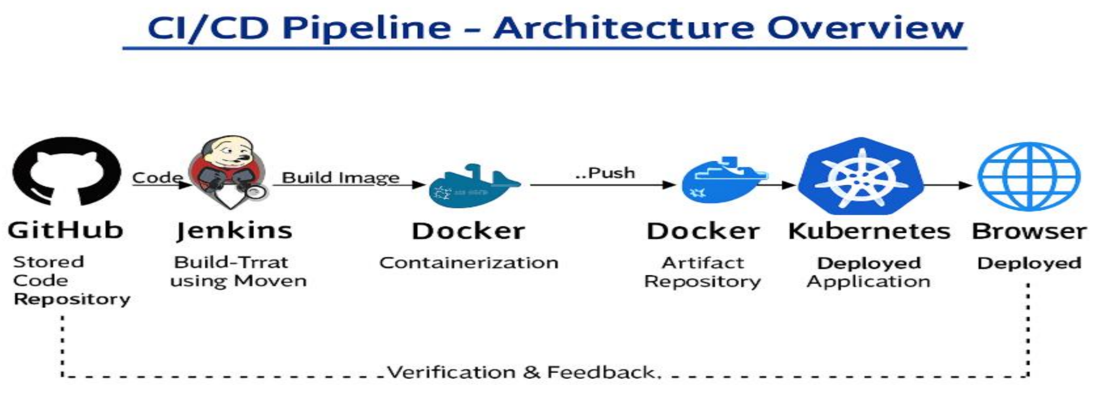
❖ HIGH-LEVEL ARCHITECTURE

Flow: GitHub → Jenkins Pipeline → (Build/Test with Maven) → Docker Image Build → Docker Hub Upload → Kubernetes (EKS) Deployment → Verification

Components

- GitHub: Stores and manages the source code repository.
- Jenkins: Orchestrates the CI/CD pipeline and automates build/test stages.
- Maven: Builds and packages the Java application into a .war file.
- Docker: Containerizes the application by creating Docker images.
- Docker Hub: Stores and versions the Docker images for deployment.
- Kubernetes (EKS): Orchestrates and manages containerized application deployment in AWS.
- AWS Load Balancer: Exposes the application to external users with a public endpoint.
- Browser: Used to validate successful deployment by accessing the application via the Load Balancer URL

Architecture Diagram



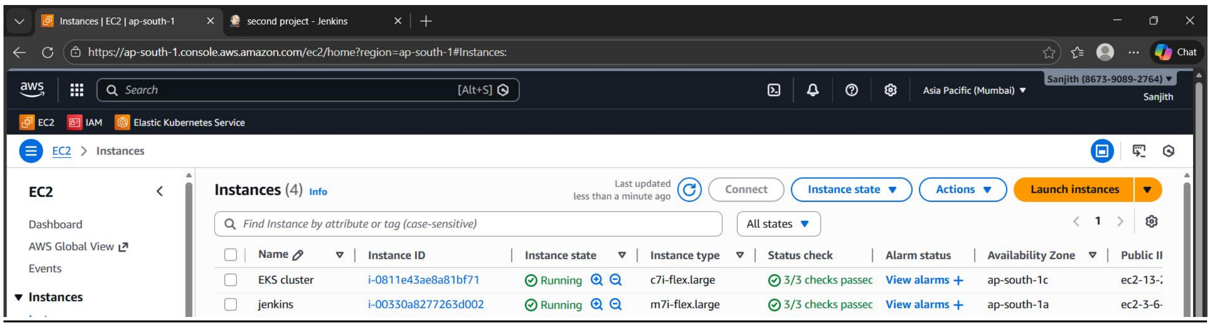
❖INFRASTRUCTURE SETUP – AWS EC2 ARCHITECTURE

Operating System Amazon Linux AMI

Service	EC2 Instance Name	Purpose	Default Port
Jenkins	jenkins	CI/CD pipeline execution	8080
EKS	EKS cluster	Cluster creation and management (kubectl, eksctl, AWS CLI)	22 (SSH)
Docker Hub	Cloud Service	Artifact repository for container images	–

Security Group – Inbound Ports:

- 22 (SSH): Restrict to your IP only
- 8080: Jenkins



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public II
EKS cluster	i-0811e43ae8a81bf71	Running	c7i-flex.large	3/3 checks passec	View alarms +	ap-south-1c	ec2-13-:
jenkins	i-00330a8277263d002	Running	m7i-flex.large	3/3 checks passec	View alarms +	ap-south-1a	ec2-3-6-

❖JENKINS SETUP

Jenkins Installation:

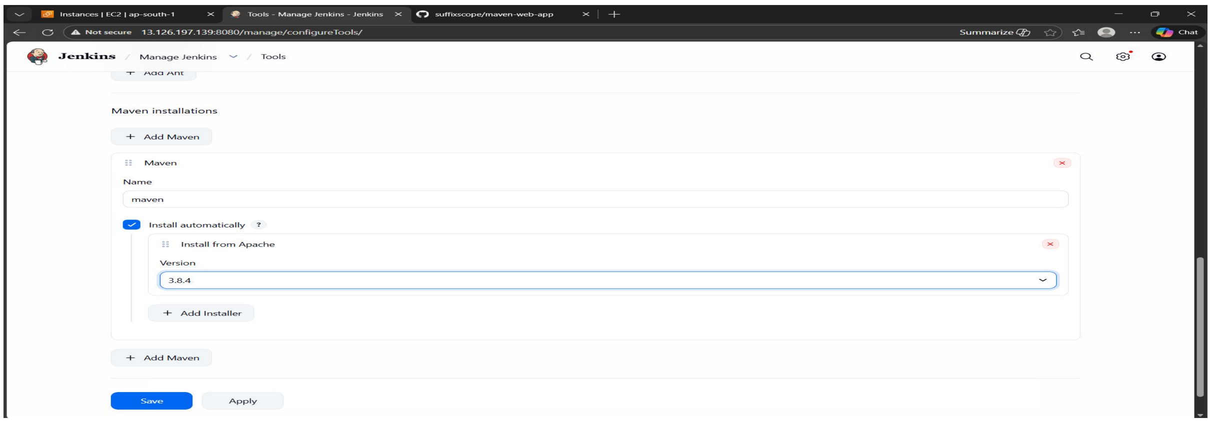
Jenkins was installed and started on the dedicated EC2 instance (jenkins).

Access URL:

https://13.126.197.139:8080/

⇒ Global Tool Configuration:

- Navigate to: Manage Jenkins → Global Tool Configuration
- Configured Maven installation with the name: maven


Instances | EC2 | ap-south-1
Tools - Manage Jenkins - Jenkins
suffixscope/maven-web-app
Not secure 13.126.197.139:8080/manage/configureTools/
Summarize
Chat
Jenkins / Manage Jenkins / Tools
+ Add Ant

Maven installations

Maven

Name: maven

[✓] Install automatically

Install from Apache

Version: 3.8.4

+ Add Installer
+ Add Maven
Save
Apply

⇒ Docker Installation

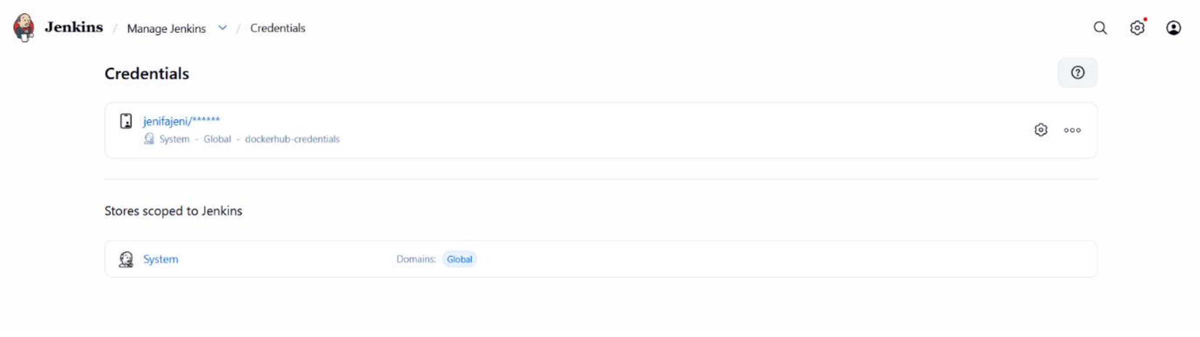
- Docker was installed on the Jenkins server to enable container image creation and pushing to Docker Hub as part of the CI/CD pipeline.
- Added Jenkins user to Docker group for pipeline execution.
- Verified installation with the command:
  docker --version

⇒ Credentials Stored in Jenkins:

Docker Hub credentials → credentials Id: dockerhub-credentials

Purpose:

Docker Hub credentials: Used by Jenkins to push Docker images into Docker Hub repositories.


Jenkins / Manage Jenkins / Credentials

Credentials

jenifajeni/******
System » Global » dockerhub-credentials

Stores scoped to Jenkins

System Domains: Global

❖ EKS HOST VM SETUP

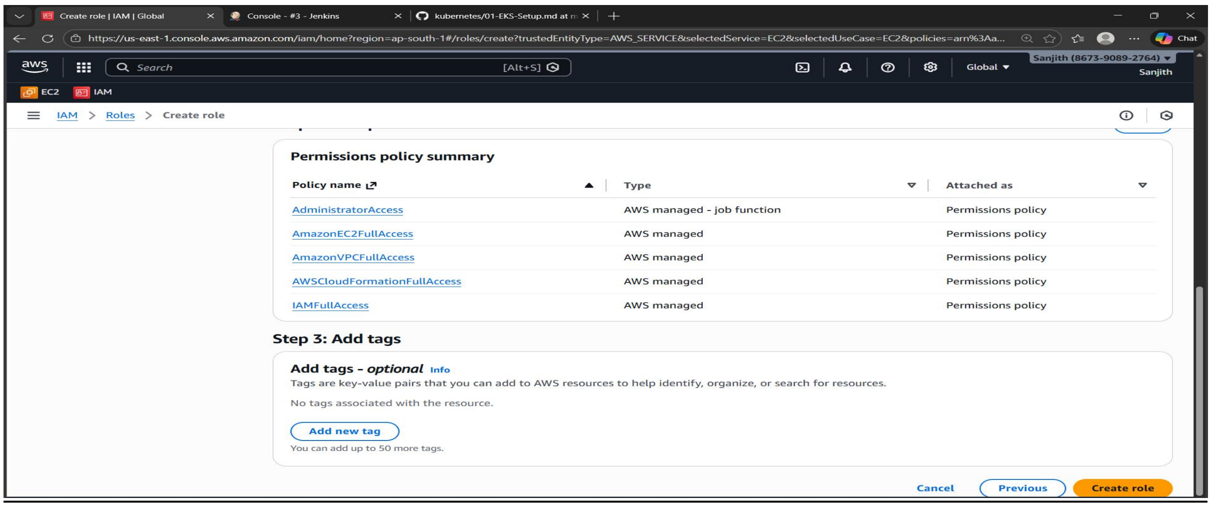
The EKS Host VM was launched and configured with AWS CLI, kubectl, and eksctl to manage the Kubernetes cluster.

⇒ IAM Role setup:

An IAM role was created and the following permissions were attached to the EKS Host VM to enable secure management of AWS resources and Kubernetes cluster operations.

Attach Permissions

- IAM Full Access
- VPC Full Access
- EC2 Full Access
- CloudFormation Full Access
- Administrator Access



Permissions policy summary

Policy name	Type	Attached as
AdministratorAccess	AWS managed - job function	Permissions policy
AmazonEC2FullAccess	AWS managed	Permissions policy
AmazonVPCFullAccess	AWS managed	Permissions policy
AWSCloudFormationFullAccess	AWS managed	Permissions policy
IAMFullAccess	AWS managed	Permissions policy

Step 3: Add tags

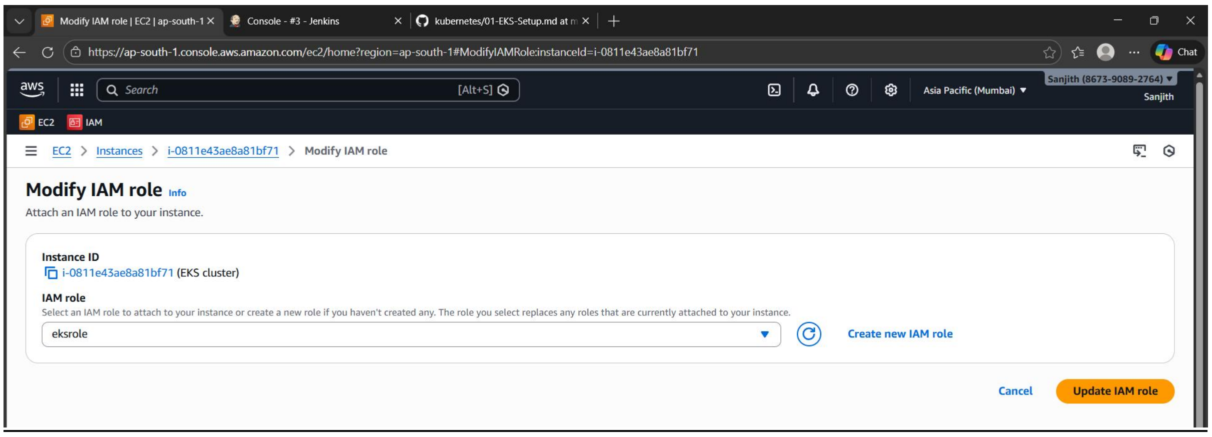
Add tags - *optional* Info

Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.

No tags associated with the resource.

You can add up to 50 more tags.

IAM > Roles > Create role
Add new tag
Cancel
Previous
Create role



Modify IAM role Info

Attach an IAM role to your instance.

Instance ID

i-0811e43ae8a81bf71 (EKS cluster)

IAM role

Select an IAM role to attach to your instance or create a new role if you haven't created any. The role you select replaces any roles that are currently attached to your instance.

eksrole

EC2 > Instances > i-0811e43ae8a81bf71 > Modify IAM role
Create new IAM role
Cancel
Update IAM role

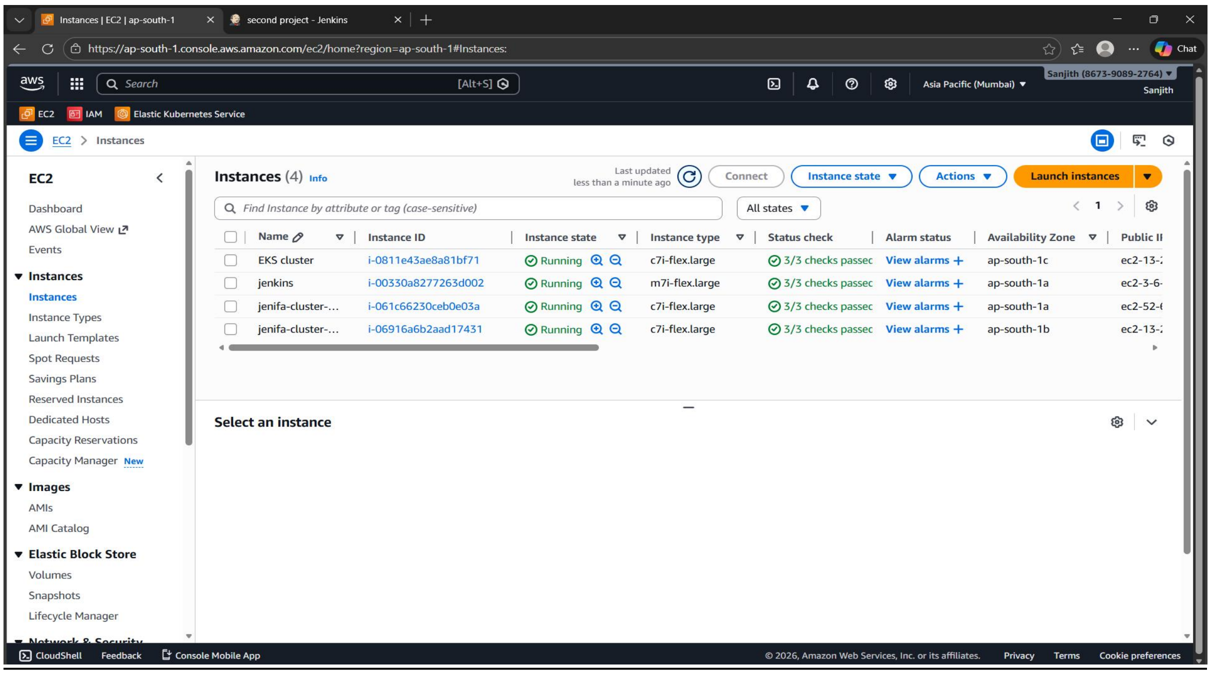
⇒ Cluster Creation

Create Kubernetes cluster using eksctl

```
eksctl create cluster --name jenifa-cluster --region ap-south-1 --node-type c7i-flex.large  --zones ap-south-1a,ap-south-1b
```

- **Purpose:**

This command provisions an Amazon EKS cluster named *jenifa-cluster* in the Mumbai region (ap-south-1). It uses worker nodes of type c7i-flex.large distributed across availability zones ap-south-1a and ap-south-1b to ensure high availability.

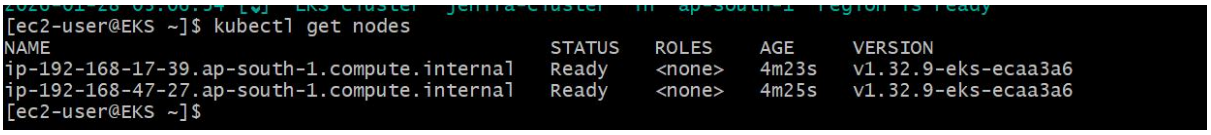


Instances (4) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public II
EKS cluster	i-0811e43ae8a81bf71	Running	c7i-flex.large	3/3 checks passec	View alarms +	ap-south-1c	ec2-13-:
jenkins	i-00330a8277263d002	Running	m7i-flex.large	3/3 checks passec	View alarms +	ap-south-1a	ec2-3-6-
jenifa-cluster-...	i-061c66230ceb0e03a	Running	c7i-flex.large	3/3 checks passec	View alarms +	ap-south-1a	ec2-52-(
jenifa-cluster-...	i-06916a6b2ad17431	Running	c7i-flex.large	3/3 checks passec	View alarms +	ap-south-1b	ec2-13-:

Select an instance

- **Verification:**



```
[ec2-user@EKS ~]$ kubectl get nodes
NAME                                          STATUS   ROLES    AGE     VERSION
ip-192-168-17-39.ap-south-1.compute.internal  Ready    <none>   4m23s   v1.32.9-eks-ecaa3a6
ip-192-168-47-27.ap-south-1.compute.internal  Ready    <none>   4m25s   v1.32.9-eks-ecaa3a6
[ec2-user@EKS ~]$
```

❖ JENKINS PIPELINE INTEGRATION WITH KUBERNETES (EKS)

⇒ IAM Role setup (Jenkins VM)

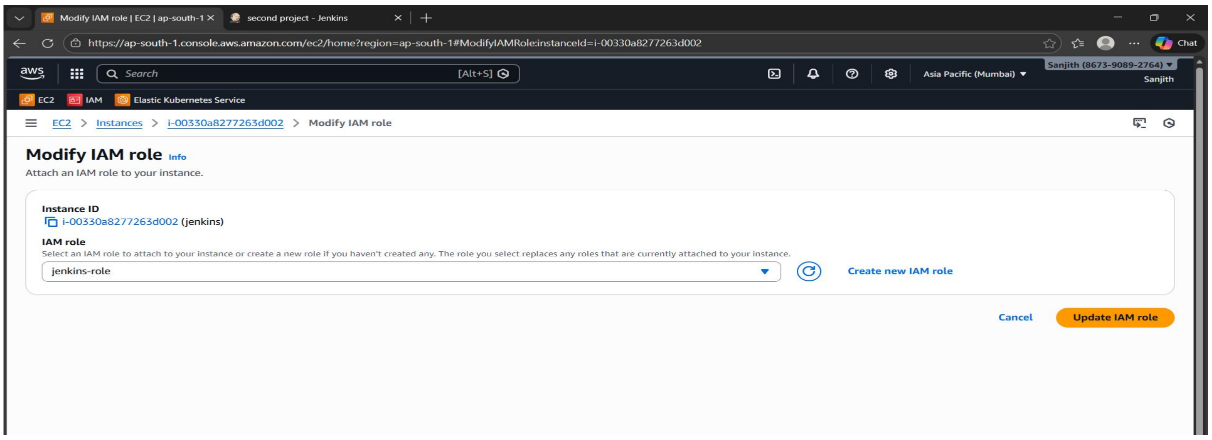
Create IAM Role and Attach Permissions

Attached Policies:

- AmazonEC2ContainerRegistryReadOnly
- AmazonEC2FullAccess
- AmazonEKSClusterPolicy
- AmazonEKSWorkerNodePolicy

Purpose:

The IAM role attached to the Jenkins VM included AWS managed policies that enable Jenkins to communicate with EC2 and EKS services for deployment operations.



Modify IAM role | EC2 | ap-south-1 — second project - Jenkins

https://ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#ModifyIAMRole:instanceId=i-00330a8277263d002

Sanjith (8673-9089-2764) — Asia Pacific (Mumbai)

EC2 > Instances > i-00330a8277263d002 > Modify IAM role

Modify IAM role Info

Attach an IAM role to your instance.

Instance ID
i-00330a8277263d002 (jenkins)

IAM role
Select an IAM role to attach to your instance or create a new role if you haven't created any. The role you select replaces any roles that are currently attached to your instance.

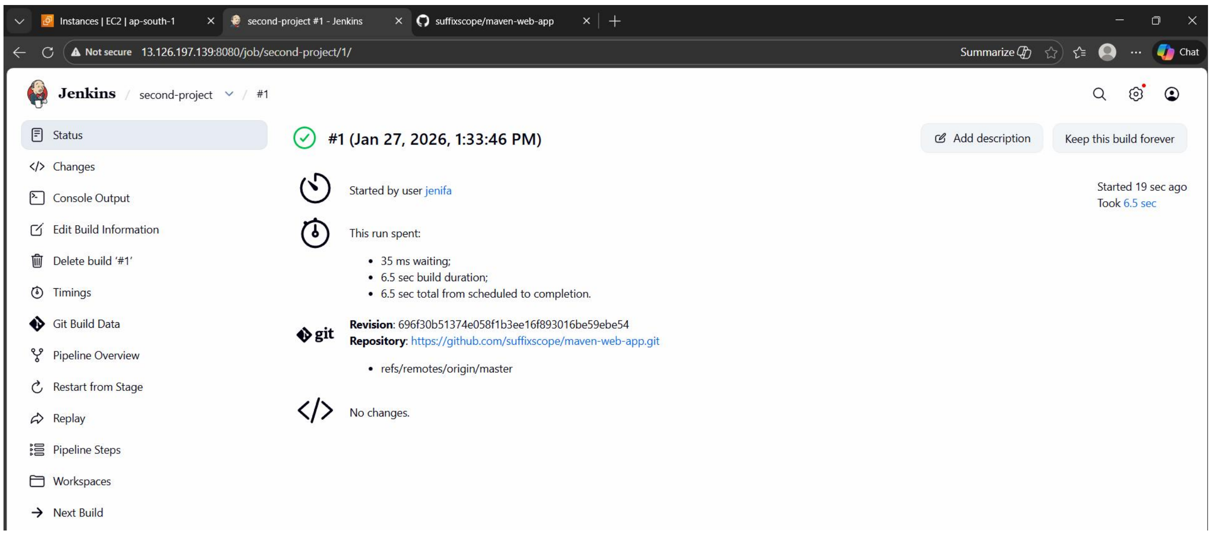
jenkins-role

Create new IAM role

Cancel | Update IAM role

⇒ Build stage-maven project compilation

```
pipeline {
    agent any

    tools{
        maven "maven"
    }

    stages {
        stage('Clone') {
            steps {
                git 'https://github.com/suffixscope/maven-web-app.git'
            }
        }
         stage('Build') {
            steps {
                sh 'mvn clean package'
            }
        }
    }
}
```



#1 (Jan 27, 2026, 1:33:46 PM)

Started by user jenifa

Started 19 sec ago
Took 6.5 sec

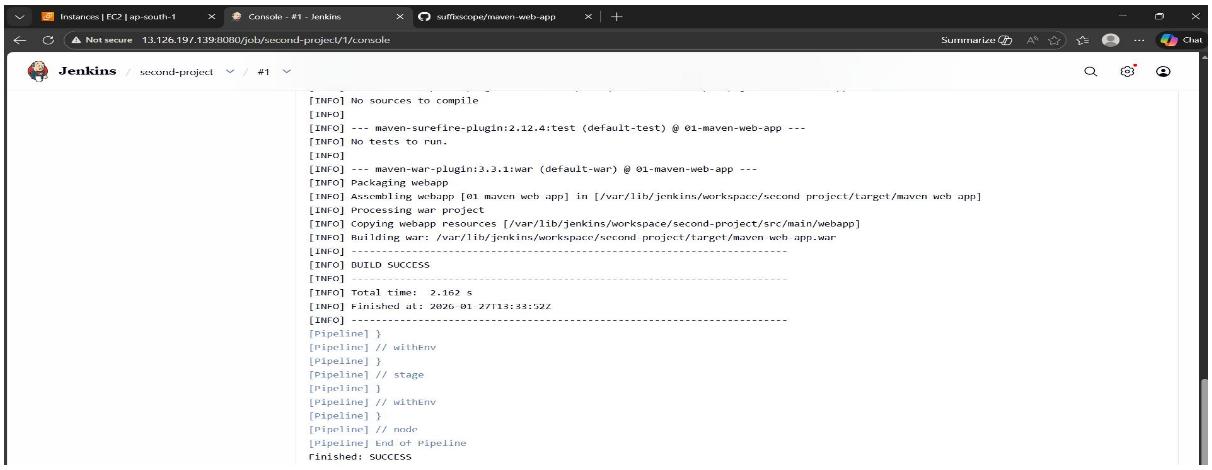
This run spent:

- 35 ms waiting;
- 6.5 sec build duration;
- 6.5 sec total from scheduled to completion.

Revision: 696f30b51374e058f1b3ee16f893016be59ebe54
Repository: https://github.com/suffixscope/maven-web-app.git

- refs/remotes/origin/master

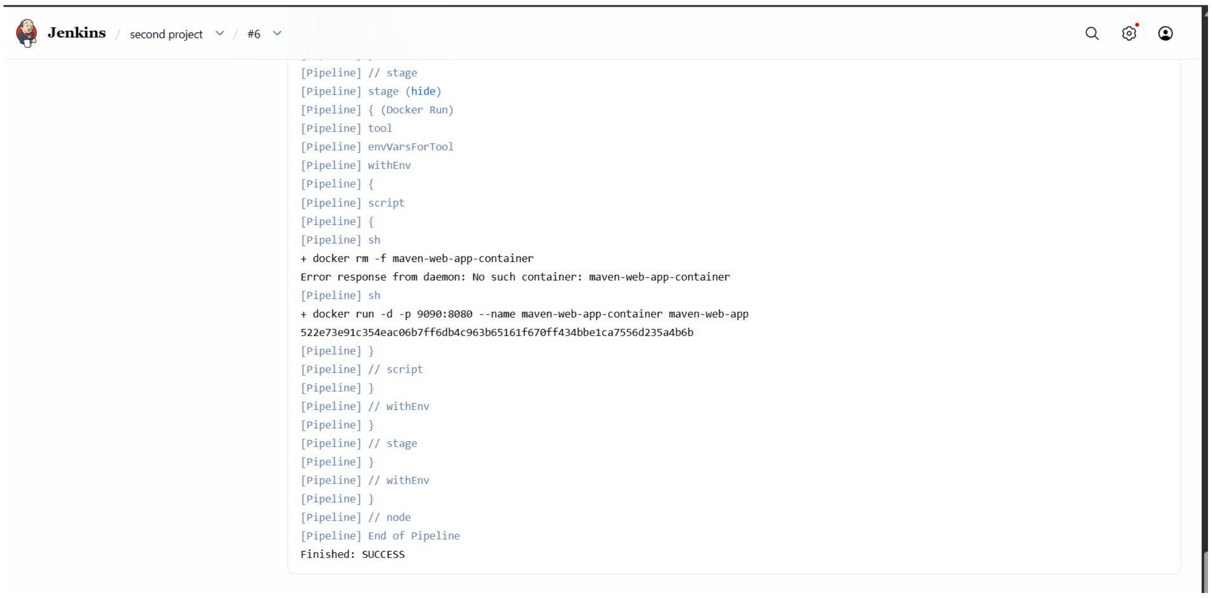
No changes.


Instances | EC2 | ap-south-1
Console - #1 - Jenkins
suffixscope/maven-web-app
Not secure 13.126.197.139:8080/job/second-project/1/console
Summarize
Chat
Jenkins / second-project / #1

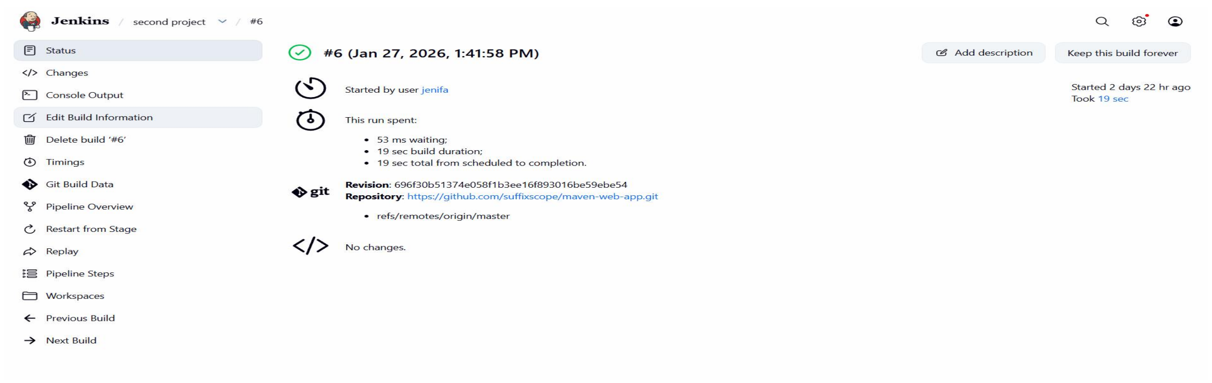
```
[INFO] No sources to compile
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ 01-maven-web-app ---
[INFO] No tests to run.
[INFO]
[INFO] --- maven-war-plugin:3.3.1:war (default-war) @ 01-maven-web-app ---
[INFO] Packaging webapp
[INFO] Assembling webapp [01-maven-web-app] in [/var/lib/jenkins/workspace/second-project/target/maven-web-app]
[INFO] Processing war project
[INFO] Copying webapp resources [/var/lib/jenkins/workspace/second-project/src/main/webapp]
[INFO] Building war: /var/lib/jenkins/workspace/second-project/target/maven-web-app.war
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.162 s
[INFO] Finished at: 2026-01-27T13:33:52Z
[INFO] ------------------------------------------------------------------------
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

⇒ Docker Execution (Image build & Run)

```
pipeline {
    agent any

    tools {
        maven 'maven'
    }

    stages {
        stage('Clone') {
            steps {
                git url: 'https://github.com/suffixscope/maven-web-app.git'
            }
        }

        stage('Build') {
            steps {
                sh 'mvn clean package'
            }
        }

        stage('Docker Build') {
            steps {
                script {
                    // Build Docker image using your Dockerfile
                    sh 'docker build -t maven-web-app .'
                }
            }
        }

        stage('Docker Run') {
            steps {
                script {
                    // Remove old container if it exists
                    sh 'docker rm -f maven-web-app-container || true'

                    // Run container on a different host port to avoid conflicts
                    sh 'docker run -d -p 9090:8080 --name maven-web-app-container maven-web-app'
                }
            }
        }
    }
}
```



```
[Pipeline] // stage
[Pipeline] stage (hide)
[Pipeline] { (Docker Run)
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] withEnv
[Pipeline] {
[Pipeline] script
[Pipeline] {
[Pipeline] sh
+ docker rm -f maven-web-app-container
Error response from daemon: No such container: maven-web-app-container
[Pipeline] sh
+ docker run -d -p 9090:8080 --name maven-web-app-container maven-web-app
522e73e91c354eac06b7ff6db4c963b65161f670ff434bbe1ca7556d235a4b6b
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```



#6 (Jan 27, 2026, 1:41:58 PM)

Started by user jenifa

Started 2 days 22 hr ago
Took 19 sec

This run spent:

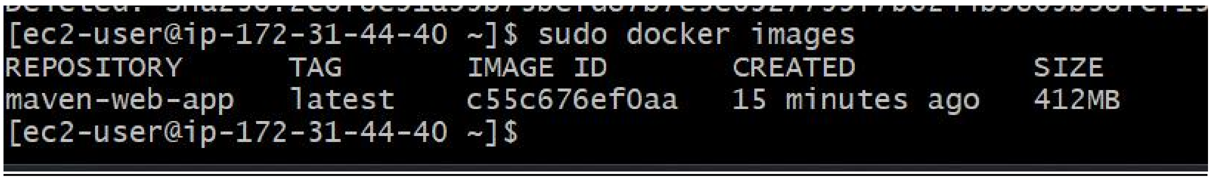
- 53 ms waiting;
- 19 sec build duration;
- 19 sec total from scheduled to completion.

git
Revision: 696f30b51374e058f1b3ee16f893016be59ebe54
Repository: https://github.com/suffixscope/maven-web-app.git

- refs/remotes/origin/master

No changes.

Verification:



```
[ec2-user@ip-172-31-44-40 ~]$ sudo docker images
REPOSITORY      TAG       IMAGE ID       CREATED          SIZE
maven-web-app   latest    c55c676ef0aa   15 minutes ago   412MB
[ec2-user@ip-172-31-44-40 ~]$
```

❖KUBERNETES DEPLOYMENT SETUP (JENKINS CLUSTER CONNECTION)

1. Install Required Tools on Jenkins Server

- AWS CLI → to connect Jenkins with AWS EKS.
- kubectl → to interact with the Kubernetes cluster.
- Ensure both are installed and available in Jenkins environment.

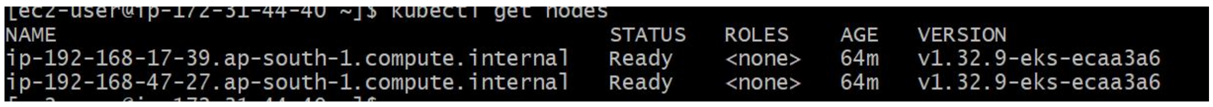
2. Copy Kubeconfig File to Jenkins

- When you create the cluster in EKS VM, a kubeconfig file (.kube/config) is generated.
- Copy this file into the Jenkins server.

3. Create Directory for Config in Jenkins(optional)

- Inside Jenkins workspace, create a directory (qube).
- Place the kubeconfig file inside it.

4. Verify Cluster Nodes

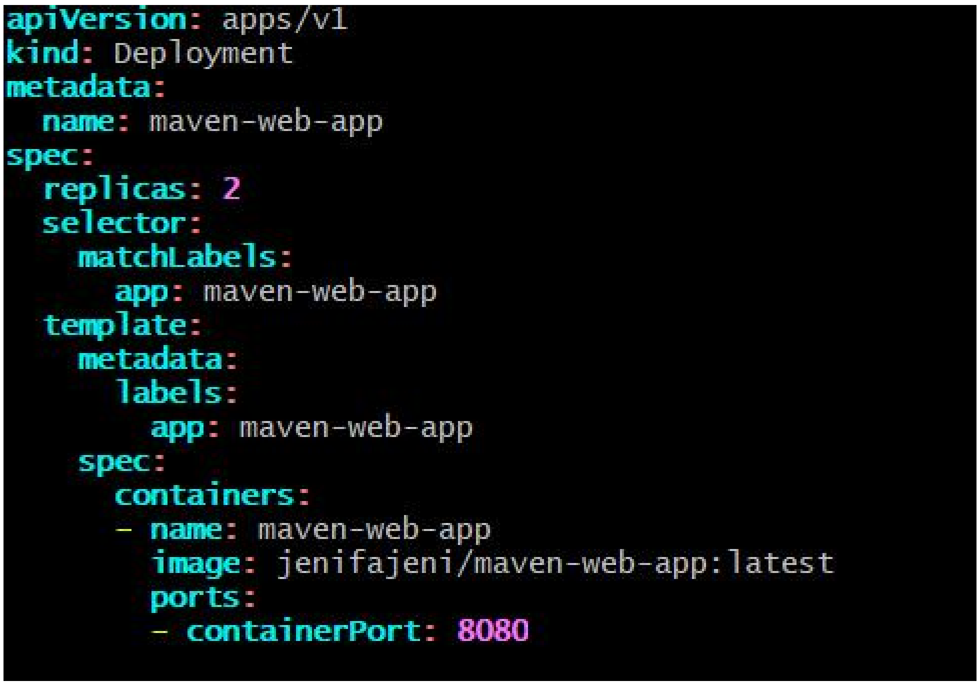


```
[ec2-user@ip-172-31-44-40 ~]$ kubectl get nodes
NAME                                          STATUS   ROLES    AGE   VERSION
ip-192-168-17-39.ap-south-1.compute.internal  Ready    <none>   64m   v1.32.9-eks-ecaa3a6
ip-192-168-47-27.ap-south-1.compute.internal  Ready    <none>   64m   v1.32.9-eks-ecaa3a6
```

5. Create deployment.yaml

This file defines how your app runs in Kubernetes.

- On your Jenkins server (or local machine), go to your project workspace
- Create the file: deployment.yaml
- Paste this content:



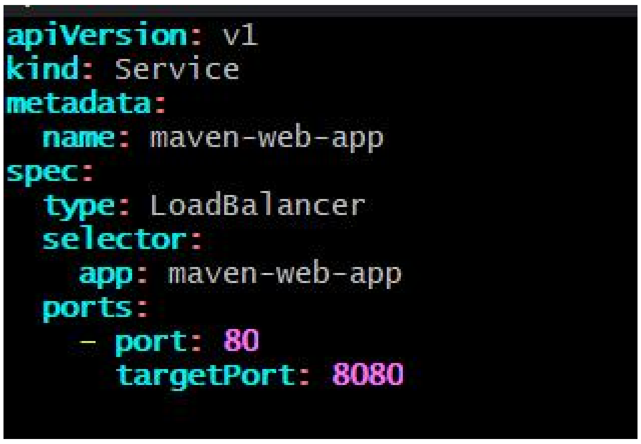
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: maven-web-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: maven-web-app
  template:
    metadata:
      labels:
        app: maven-web-app
    spec:
      containers:
      - name: maven-web-app
        image: jenifajeni/maven-web-app:latest
        ports:
        - containerPort: 8080
```

- Save and exit

6. Create service.yaml

This file exposes your app via AWS LoadBalancer

- On your Jenkins server (or local machine), go to your project workspace.
- Create the file: service.yaml
- Paste this content:



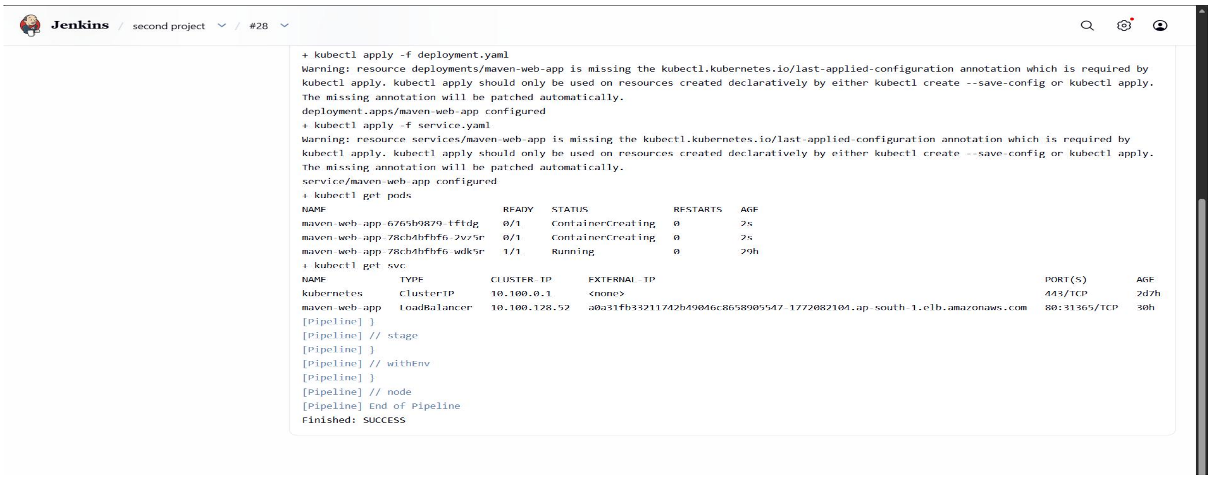
```
apiVersion: v1
kind: Service
metadata:
  name: maven-web-app
spec:
  type: LoadBalancer
  selector:
    app: maven-web-app
  ports:
    - port: 80
      targetPort: 8080
```

- save and exit

7. Create Jenkins Job for Deployment

- **In Jenkins, create a Pipeline job.**
- **Add a stage for Kubernetes deployment.**

```
pipeline {
    agent any

    environment {
        KUBECONFIG = "/var/lib/jenkins/qube/config"
    }

    stages {
        stage('Test EKS Connection') {
            steps {
                sh 'kubectl get nodes'
            }
        }

        stage('Deploy by Script') {
            steps {
                sh '''
                   kubectl apply -f deployment.yaml
                   kubectl apply -f service.yaml
                   kubectl get pods
                   kubectl get svc
                '''
            }
        }
    }
}
```



```
+ kubectl apply -f deployment.yaml
Warning: resource deployments/maven-web-app is missing the kubectl.kubernetes.io/last-applied-configuration annotation which is required by
kubectl apply. kubectl apply should only be used on resources created declaratively by either kubectl create --save-config or kubectl apply.
The missing annotation will be patched automatically.
deployment.apps/maven-web-app configured
+ kubectl apply -f service.yaml
Warning: resource services/maven-web-app is missing the kubectl.kubernetes.io/last-applied-configuration annotation which is required by
kubectl apply. kubectl apply should only be used on resources created declaratively by either kubectl create --save-config or kubectl apply.
The missing annotation will be patched automatically.
service/maven-web-app configured
+ kubectl get pods
NAME                              READY   STATUS              RESTARTS   AGE
maven-web-app-6765b9879-tftdg     0/1     ContainerCreating   0          2s
maven-web-app-78cb4bfbf6-2vz5r    0/1     ContainerCreating   0          2s
maven-web-app-78cb4bfbf6-wdk5r    1/1     Running             0          29h
+ kubectl get svc
NAME            TYPE           CLUSTER-IP      EXTERNAL-IP                                                               PORT(S)        AGE
kubernetes      ClusterIP      10.100.0.1      <none>                                                                    443/TCP        2d7h
maven-web-app   LoadBalancer   10.100.128.52   a0a31fb33211742b49046c8658905547-1772082104.ap-south-1.elb.amazonaws.com   80:31365/TCP   30h
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

❖ CI/CD Pipeline Implementation (Jenkins + Docker + EKS)

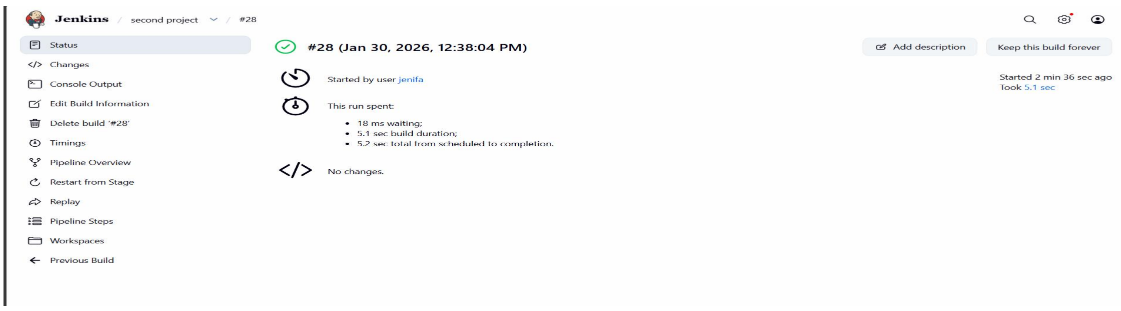
Pipeline Stages

Stage	What it does	Output / Result
Clone Repo	Clones the master branch from GitHub into Jenkins workspace	Source code available in Jenkins workspace

Stage	What it does	Output / Result
Maven Build	Runs mvn -B clean package using Maven	WAR file generated: target/maven-web-app.war
Docker Build	Builds Docker image from the WAR file using Dockerfile	Docker image created: jenifajeni/maven-web-app:latest
Docker Push	Pushes the Docker image to Docker Hub using stored credentials	Image stored and versioned in Docker Hub
Kubernetes Deployment	Applies deployment.yaml manifest to EKS cluster via Jenkins pipeline	Pods created in EKS cluster
Kubernetes Service	Applies service.yaml manifest to expose app via AWS LoadBalancer	Service created with LoadBalancer DNS
Verify Pods	Runs kubectl get pods to check pod status	Pods show STATUS = Running
Verify Service	Runs kubectl get svc to check service status	LoadBalancer provisioned, EXTERNAL-IP shows AWS DNS
Access Application	Copy LoadBalancer DNS and paste in browser with /maven-web-app	Application accessible via public URL

❖TESTING AND VERIFICATION

Verify Pods and Service



Jenkins / second project / #28

- Status
- Changes
- Console Output
- Edit Build Information
- Delete build '#28'
- Timings
- Pipeline Overview
- Restart from Stage
- Replay
- Pipeline Steps
- Workspaces
- Previous Build

#28 (Jan 30, 2026, 12:38:04 PM)

Add description | Keep this build forever

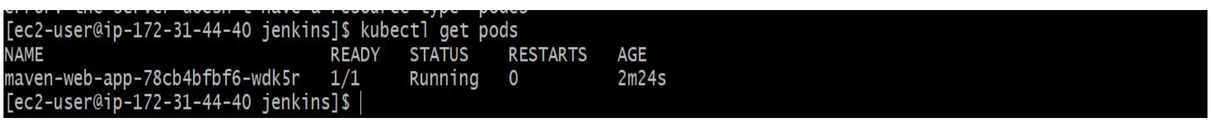
Started by user jenifa

Started 2 min 36 sec ago
Took 5.1 sec

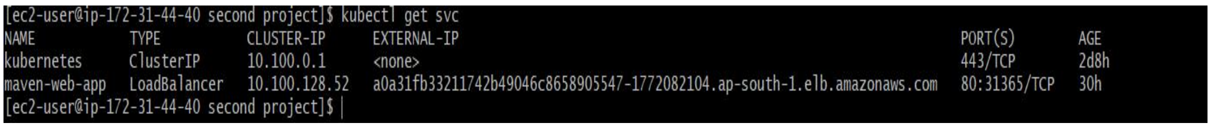
This run spent:

- 18 ms waiting;
- 5.1 sec build duration;
- 5.2 sec total from scheduled to completion.

No changes.



```
error: the server doesn't have a resource type "pods"
[ec2-user@ip-172-31-44-40 jenkins]$ kubectl get pods
NAME                            READY   STATUS    RESTARTS   AGE
maven-web-app-78cb4bfbf6-wdk5r   1/1     Running   0          2m24s
[ec2-user@ip-172-31-44-40 jenkins]$ |
```

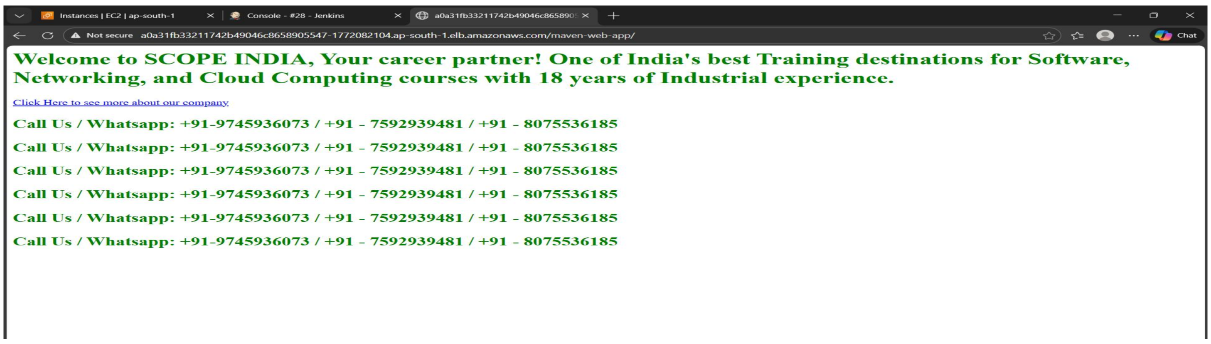


```
[ec2-user@ip-172-31-44-40 second project]$ kubectl get svc
NAME            TYPE           CLUSTER-IP      EXTERNAL-IP                                                                PORT(S)        AGE
kubernetes      ClusterIP      10.100.0.1      <none>                                                                     443/TCP        2d8h
maven-web-app   LoadBalancer   10.100.128.52   a0a31fb33211742b49046c8658905547-1772082104.ap-south-1.elb.amazonaws.com   80:31365/TCP   30h
[ec2-user@ip-172-31-44-40 second project]$ |
```

Access Application

Copy the LoadBalancer DNS and paste in browser as :

0a31fb33211742b49046c8658905547-1772082104.ap-south-1.elb.amazonaws.com/maven-web-app/



Instances | EC2 | ap-south-1 | Console - #28 - Jenkins | a0a31fb33211742b49046c86589

Not secure a0a31fb33211742b49046c8658905547-1772082104.ap-south-1.elb.amazonaws.com/maven-web-app/

Welcome to SCOPE INDIA, Your career partner! One of India's best Training destinations for Software, Networking, and Cloud Computing courses with 18 years of Industrial experience.

Click Here to see more about our company

Call Us / Whatsapp: +91-9745936073 / +91 - 7592939481 / +91 - 8075536185
Call Us / Whatsapp: +91-9745936073 / +91 - 7592939481 / +91 - 8075536185
Call Us / Whatsapp: +91-9745936073 / +91 - 7592939481 / +91 - 8075536185
Call Us / Whatsapp: +91-9745936073 / +91 - 7592939481 / +91 - 8075536185
Call Us / Whatsapp: +91-9745936073 / +91 - 7592939481 / +91 - 8075536185
Call Us / Whatsapp: +91-9745936073 / +91 - 7592939481 / +91 - 8075536185

❖ CONCLUSION

This project successfully implemented an end-to-end CI/CD pipeline on AWS using:

- Jenkins as the automation server
- Maven for build and packaging of the Java application
- Docker for containerization of the application

• Docker Hub for artifact storage and versioning

• Kubernetes (Amazon EKS) for orchestration and deployment

• AWS Load Balancer for external application access

The pipeline integrates source code management, automated builds, containerization, image repository management and Kubernetes deployment into a seamless workflow. Each stage was tested and verified from cloning the repository, building with Maven, pushing Docker images, deploying to EKS, verifying pods and services and finally accessing the application via the LoadBalancer DNS. This ensures reliability, scalability and efficiency in delivering the Maven web application to real-world users.

❖Final Outcomes

• Automated build and packaging with Maven

• Docker image creation from the packaged application

• Image pushed and versioned securely in Docker Hub

• Kubernetes deployment applied on Amazon EKS cluster via Jenkins pipeline

• Service exposed through AWS LoadBalancer with external DNS

• Application successfully accessible via browser using the LoadBalancer URL

Authored by: Jenifa R | 29 January 2026