**Use case:**

Create a CICD pipeline in Jenkins to fetch Java code from GitHub repository, containerize using Docker, push to ECR and deploy to EKS so that users can access the application viz URL.

---------------------------------------------------------------------------------------------------------------------------------------

**Technology Used:**

Docker, ECR, EKS, GitHub Actions

---------------------------------------------------------------------------------------------------------------------------------------

**Folder structure:**

**|\_** src

**|\_** Dockerfile

|\_ Jenkinsfile

|\_ k8s/

**|\_** deployment.yaml

**|\_** service.yaml

---------------------------------------------------------------------------------------------------------------------------------------

**Prerequisites**

* Launch instance for Jenkins.
* Install AWS CLI on Local/Jenkins server.

---------------------------------------------------------------------------------------------------------------------------------------

**Local Setup**

**Configure Git**

* Install Git

Browse – <https://git-scm.com/downloads>

* Open Git Bash
* Configure Git user info

$ git config --global user.name "shivam"

$ git config --global user.email [shivamthakur0567@gmail.com](mailto:shivamthakur0567@gmail.com)

**Push code from Local to Remote Repo**

* Create new repository in GitHub

GiHub > Plus icon > New repository

Repository name – cicd-using-jenkins-docker-ecr-eks-helm > Add README **>** Create repository

* Go into project folder

$ cd cicd-using-jenkins-docker-ecr-eks-helm

* Initialize Git Repository

$ git init

* Add files to Git

$ git add .

* Commit changes

$ git commit -m "Initial commit"

* Check current local branch

$ git branch

Master

* Rename local branch to main

$ git branch -m main

* Add our remote repository

$ git remote add origin <https://github.com/shivam-th/cicd-using-jenkins-docker-ecr-eks.git>

* Push the code from local to remote repository

$ git push -u origin main --force

**Configure AWS**

* Install AWS CLI

Browse – <https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html>

* Create user

IAM > Users > Create user

User name – admin > Check – Provide access to the AWS Management Console > Next  
Permissions options – Attach policies directly > Permissions policies – [AdministratorAccess](https://983877353540-643piekf.us-east-1.console.aws.amazon.com/iam/home?region=us-east-1#/policies/details/arn%3Aaws%3Aiam%3A%3Aaws%3Apolicy%2FAdministratorAccess)  
Next > Create user

* Create access key

IAM > Users > admin > Create access key

Use case – Command line Interface (CLI)/Other > Next > Create access key

* Configure AWS in local

$ aws configure

AWS Access Key ID [None]: <put-access-key>

AWS Secret Access Key [None]: <put-secret-key>

Default region name [None]: <enter>

Default output format [None]: <enter>

**Helm Chart**

* Install Helm

Browse – <https://helm.sh/docs/intro/install/>

* Add in System variable

Edit environment variable > Path > Add path – C:\Program Files\Helm-3.17.3 > OK

---------------------------------------------------------------------------------------------------------------------------------------

**Jenkins Setup**

**Launch Jenkins Instance**

* Visit

EC2 > Instances > Launch instances

* Give name for instance

Names and tags > Name – jenkins-server

* Select AMI

Application and OS Images > Quick Start > Select – Ubuntu (Ubuntu Server 22.04)

* Select Instance type

Instance type > Instance type – t2.small

* Create or choose existing key

Key pair > Create new key pair > Key pair name – jenkins-key > Type – RSA > File Format – .pem > Create Key pair

* Create security group

Network settings > Choose – Create security group > Edit

Security group name – jenkins-sg > Description – jenkins-sg

Type – ssh > Source type – Anywhere > Add security group

Type – Custom TCP > Port range – 8080 > Source type – Anywhere > Description – allow port 8080

* Add provisioner script

Advanced details > User data

#!/bin/bash

# Update system

sudo apt-get update -y

# Install Java (required for Jenkins)

sudo apt install openjdk-17-jdk -y

# Add Jenkins repository and key

sudo wget -O /etc/apt/keyrings/jenkins-keyring.asc \

https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

echo "deb [signed-by=/etc/apt/keyrings/jenkins-keyring.asc]" \

https://pkg.jenkins.io/debian-stable binary/ | sudo tee \

/etc/apt/sources.list.d/jenkins.list **>** /dev/null

# Install Jenkins

sudo apt-get update -y

sudo apt-get install jenkins unzip -y

# Install AWS CLI

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

unzip awscliv2.zip

sudo ./aws/install

# Install Helm

curl -fsSL -o get\_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3

chmod 700 get\_helm.sh

./get\_helm.sh

# Install kubectl

curl -LO https://dl.k8s.io/release/$(curl -L -s \https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl

sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

# Install eksctl

curl –silent \

  --location "https://github.com/eksctl-io/eksctl/releases/latest/download/eksctl\_$(uname -s)\_amd64.tar.gz" \

| tar xz -C /tmp

sudo mv /tmp/eksctl /usr/local/bin

# Install Docker

# Add Docker's official GPG key:

sudo apt-get update -y

sudo apt-get install ca-certificates curl

sudo install -m 0755 -d /etc/apt/keyrings

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc

sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:

echo \

"deb [arch=$(dpkg --print-architecture) \

signed-by=/etc/apt/keyrings/docker.asc] \

https://download.docker.com/linux/ubuntu \

$(. /etc/os-release && echo "${UBUNTU\_CODENAME:-$VERSION\_CODENAME}") stable" | \

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update -y

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin -y

# Install maven

sudo apt install maven -y

# Clean up

rm -rf awscliv2.zip aws get\_helm.sh

# Reboot

sudo reboot

Note: Modify commands according to below installation documentation if required.

Jenkins – <https://www.jenkins.io/doc/book/installing/linux/>

AWS CLI – <https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html>

HELM – <https://helm.sh/docs/intro/install/>

kubectl – <https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/>

eksctl – <https://eksctl.io/installation/>

Docker – <https://docs.docker.com/engine/install/ubuntu/>

* Launch instance
* Login to Jenkins server

$ ssh -i <key-pair-file-name> ubuntu@<public-p>

* Verify

$ systemctl status jenkins

$ aws --version

$ helm version

$ kubectl version --client

$ eksctl version

$ docker version

$ mvn -v

**Configure Jenkins**

* Add jenkins user to Docker group ( so Jenkins can run Docker commands)

$ sudo usermod -aG docker jenkins

$ sudo systemctl restart jenkins

$ sudo systemctl status jenkins

* Reload Docker daemon files & restart docker service

$ sudo systemctl daemon-reload

$ sudo systemctl restart docker

$ systemctl status docker

* Browse – <instance public IP>:8080
* Copy password

$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword

<copy-password>

* Unlock Jenkins

Administrator password – <paste-password> > Continue

* Customize Jenkins

Select plugins to install > Uncheck plugins if not required (ant) >

Search – docker > Select & Install – Docker, Docker Pipeline, AWS Plugin (AWS Credentials)

* Create First Admin User

Username – admin

Password – admin@123

Fullname – admin

E-mail address – [admin@gmail.com](mailto:admin@gmail.com)

* Save & Continue
* If your instance IP is dynamic, then give any domain

Jenkins URL – <http://jenkins.xyz> > Save and Finish > Start using Jenkins

**Configure AWS CLI on Jenkins server**

* Configure AWS

$ aws configure

AWS Access Key ID [None]: <put-access-key>

AWS Secret Access Key [None]: <put-secret-key>

Default region name [None]: <enter>

Default output format [None]: <enter>

* Verify

$ aws sts get-caller-identity

**Configure Maven**

* Jenkins > Settings > Manage Jenkins > Tools > Add Maven

Name – Maven3.8

Uncheck – Install automatically

MAVEN\_HOME – /usr/share/maven

Apply > Save

**Configure AWS Credentials in**

* Jenkins > Settings > Manage Jenkins > Credentials

Domain – global > Add Credentials

Kind – AWS Credentials

Username – aws-eks-creds

Access Key ID – <pub-access-key>

Secret Access Key – <put-secret-key>

---------------------------------------------------------------------------------------------------------------------------------------

**ECR Setup**

* Create ECR repo

ECR > Create repository

Visibility – private > Repository name – my-docker-rep > Create repository

OR

$ aws ecr create-repository \

    --repository-name cicd-using-jenkins-docker-ecr-eks-helm \

    --region us-east-1

---------------------------------------------------------------------------------------------------------------------------------------

**EKS Setup**

* Create EKS cluster

$ eksctl create cluster \

--name my-cluster \

--region us-east-1 \

--nodegroup-name my-nodes \

--node-type t3.small \

--nodes 2 \

--nodes-min 1 \

--nodes-max 3 \

--managed

Verify

$ eksctl get cluster --name my-cluster --region us-east-1

* Update Kube config

$ aws eks update-kubeconfig --name my-cluster --region us-east-1

* Create role

IAM > Roles > Create role

Trusted entity type – AWS Service

Service or use case – EC2 > Use case – EC2 > Next

Permissions policies – AdministratorAccess > Next

Role name – jenkins-eks-access-role > Create role

* Attach role to Jenkins instance

EC2 > Instances > Select – jenkins-server > Actions > Security > Modify IAM role

IAM role – jenkins-eks-access-role > Update IAM role

Verify

$ kubectl get nodes

$ kubectl create deployment nginx --image=nginx

$ kubectl create deployment nginx --image=nginx

$ kubectl get deployments

* Create namespace

$ kubectl create ns helm-deployment

Verify

$ kubectl get ns

* Grant Jenkins Access to Amazon EKS Cluster

$ kubectl edit configmap aws-auth -n kube-system

- rolearn: arn:aws:iam::983877353540:role/jenkins-[eks-access-role](https://983877353540-643piekf.us-east-1.console.aws.amazon.com/iam/home?region=us-east-1" \l "/roles/details/eks-admin-role" \t "_self)

username: jenkins

groups:

- system:masters

Verify

$ kubectl get configmap aws-auth -n kube-system -o yaml

---------------------------------------------------------------------------------------------------------------------------------------

**Create Helm chart (from local)**

* Create "helm" folder
* Create helm chart

$ helm create helm/mychart

$ tree helm

* Modify in helm files

helm/value.yaml

service:

type: LoadBalancer

port: 80

livenessProbe:

  httpGet:

    path: /

    port: 8080

readinessProbe:

  httpGet:

    path: /

    port: 8080

helm/service.yaml

ports:

    - port: {{ .Values.service.port }}

      targetPort: 8080

* Push changes to repo

$ git push

---------------------------------------------------------------------------------------------------------------------------------------

**Create a pipeline in Jenkins**

* Jenkins > New item > MyAppDeployJob > Pipeline > OK
* Testing

Pipeline Script > Pipeline Syntax

Sample Step – checkout: Check out from version control

Repository URL – <your-repo-url>

Branch – main

Generate Pipeline Script >

<copy-script>

* Test Pipeline

Pipeline Script > try sample Pipeline > Hello World > Pipeline Syntax

pipeline {

agent any

stages {

stage('Checkout') {

steps {

checkout scmGit(branches: [[name: '\*/main']], extensions: [], userRemoteConfigs: [[url: 'https://github.com/shivam-th/cicd-using-jenkins-docker-ecr-eks.git']])

}

}

}

}

Verify

Apply > Save > Build Now

* Configure Jenkins Pipeline Using SCM

Jenkins Pipeline > Configure > Pipeline > Definition – Script from SCM

SCM – Git > Repository URL – <repo-url> > Branch to build – \*/main > Script Path – Jenkinsfile

Apply > Save > Build Now

Verify

Apply > Save > Build Now

---------------------------------------------------------------------------------------------------------------------------------------

**Configure Webhooks**

* Install GitHub plugin

Jenkins > Manage plugins > Plugins > Available plugins > Search – GitHub Integration Plugin

* Configure Jenkins Job

Jenkins Pipeline > Configure > Triggers > GitHub hook trigger for GITScm polling

* Add Webhook in GitHub

GitHub repo > Settings > Webhooks > Add webhooks

Payload URL – http://<jenkins-url>/github-webhook/

Which events would you like to trigger this webhook? – Just the push event > Add webhook

* Verify

Push code to repo > Pipe line should triggered

---------------------------------------------------------------------------------------------------------------------------------------

**Application Check**

* Check pods

$ kubectl get pods -n helm-deployment

* Check service

$ kubectl get svc -n helm-deployment

* Browse – Load balancer URL

---------------------------------------------------------------------------------------------------------------------------------------

**Commands**

* Clean up commands

$ sudo rm -rf /var/lib/jenkins/workspace/\*

$ sudo docker system prune -a -f

$ sudo rm -rf /var/lib/jenkins/.cache/\*

$ sudo journalctl --vacuum-time=3d

$ sudo apt-get clean

* Kubectl commands

$ kubectl get svc -n helm-deployment

$ kubectl get pods -n helm-deployment

$ kubectl delete pods first-mychart-6c7f69688b-gdn6j -n helm-deployement

$ kubectl delete deployment helm-deployment

* Helm commands

$ helm ls -n helm-deployment

$ helm uninstall first -n helm-deployment

---------------------------------------------------------------------------------------------------------------------------------------

**Troubleshooting**

**IF Error 502**

* Check Service Configuration

$ kubectl get pods -n <name-space>

* Describe the Pod

$ kubectl describe svc -n <name-space>

* Check Deployment Logs

$ kubectl logs <pod-name> -n <name-space>

* Check Service Configuration

$ kubectl get svc -n <name-space>

* Check liveness and readiness in kubernetes manifest files
* Check port in service.yaml

spec:

type: LoadBalancer

ports:

- port: 80 # external port

targetPort: 8080 # internal port

Here 8080 is a container port , 80 is loadbalancer target port

Browse URL as per external port