Deploying a Spring-boot Application on AWS EKS using Jenkins CICD

- Example Project:

https://github.com/hosniah/springboot-app-for-aks

- Creating and Managing EKS Clusters

- Install Jenkins on an AWS Linux EC2:

```
sudo yum update –y
sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo
sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key
sudo yum upgrade
sudo amazon-linux-extras install java-openjdk11 -y
sudo yum install jenkins -y --nogpgcheck
sudo systemctl enable jenkins
sudo systemctl start jenkins
sudo systemctl status jenkins
```

- Connect to http://<instance_public_ip>:8080 from your browser. You will be able to access Jenkins through Management Interface.
- Now on the left-hand side, go to Manage Jenkins and then select Manage Plugins
- Go to the Available tab and then type Amazon EC2 plugin at the top right.

* Select Manage Jenkins and then select Manage Nodes and Clouds
* Select Configure Cloud and then Add a new cloud and select Amazon EC2
* In the fields that appear on the window, Give some name to Amazon EC2, Click Add under Amazon EC2 Credentials
* Select AWS Credentials as the Kind from the Jenkins Credentials Provider.
* Enter the IAM User programmatic access keys with EC2 instance launch permissions and click Add.
* Select region from the drop-down and Add EC2 Key Pair’s Private Key,
* Select an SSH Username with Private Key as the Kind and ec2-user as the Username from the Jenkins Credentials Provider.
* Enter your Private Key directly and click on Add
* Click on Test connection and make sure that it states “Success."

- **Install Docker on Amazon Linux Machine**

  ```
  sudo yum update -y
  ```

sudo yum install docker -y
sudo systemctl start docker
sudo docker run hello-world
sudo systemctl enable docker
docker --version
sudo usermod -a -G docker $(whoami)
newgrp docker

- **Install required Plugins in Jenkins**

  Amazon EC2 plugin
  Amazon ECR plugin
  Docker plugin
  Docker Pipeline
  CloudBees Docker Build and Publish plugin
  Kubernetes CLI Plugin
  Pipeline: AWS Steps

- **Integrate Docker with Jenkins**
- Add Jenkins user to the Docker group:
  sudo usermod -a -G docker jenkins
  sudo systemctl restart jenkins
  sudo systemctl daemon-reload
  sudo service docker stop
  sudo service docker start

- **Create a repository in ECR**

- Log in to the AWS Management Console and navigate to the Amazon ECR service.
- Click on the “Create repository” button.
- Enter a name for your repository. This name must be unique within your AWS account.
- (Optional) Add a description for your repository.
- Click on the “Create repository” button.
- You will now see your newly created repository in the repository list.
- Select the newly created repo and then choose to view push commands.
- Use those commands to authenticate and push an image to your repository while writing Jenkinsfile.

- **Add Maven to Jenkins**

  tools {

```
            maven 'Maven3'
        }
```

- **Write Jenkinsfile**

- Write a Jenkinsfile to define the steps in a pipeline for deploying a spring-boot application to an EKS cluster using Jenkins.

```
pipeline {
    tools {
        maven 'Maven3'
    }
    agent any
    stages {
        stage('Checkout') {
            steps {
                checkout([$class: 'GitSCM', branches: [[name: '*/main']], extensions: [],
userRemoteConfigs: [[url: '<GIT_REPO_URL>']]])
            }
        }
        stage('Build Jar') {
            steps {
                sh 'mvn clean package'
            }
        }
        stage('Docker Image Build') {
            steps {
                sh 'docker build -t <IMAGE_NAME> .'
            }
        }
        stage('Push Docker Image to ECR') {
            steps {
                withAWS(credentials: '<AWS_CREDENTIALS_ID>', region: '<AWS_REGION>') {
                    sh 'aws ecr get-login-password --region <AWS_REGION> | docker login
--username AWS --password-stdin <ECR_REGISTRY_ID>'
                    sh 'docker tag <IMAGE_NAME>:latest
<ECR_REGISTRY_ID>/<IMAGE_NAME>:latest'
                    sh 'docker push <ECR_REGISTRY_ID>/<IMAGE_NAME>:latest'
                }
            }
        }
        stage('Integrate Jenkins with EKS Cluster and Deploy App') {
            steps {
                withAWS(credentials: '<AWS_CREDENTIALS_ID>', region: '<AWS_REGION>') {
```

```
                script {
                  sh ('aws eks update-kubeconfig --name <EKS_CLUSTER_NAME> --region
<AWS_REGION>')
                  sh "kubectl apply -f <K8S_DEPLOY_FILE>.yaml"
                }
                }
      }
   }
   }
}
```

- Interact with a cluster from terminal

- Retrieve the status of an Amazon Elastic Container Service for Kubernetes (EKS) cluster
  aws eks describe-cluster --region <region-name> --name <cluster-name> --query cluster.status

- Update the kubeconfig file
  aws eks --region <region-name> update-kubeconfig --name <cluster-name>

- Retrieve data from the cluster
  kubectl get nodes
  kubectl get pods
  kubectl get services
  kubectl get deployments

- Expose the service
- Get the external IP
- Allow the required ports