Sign up

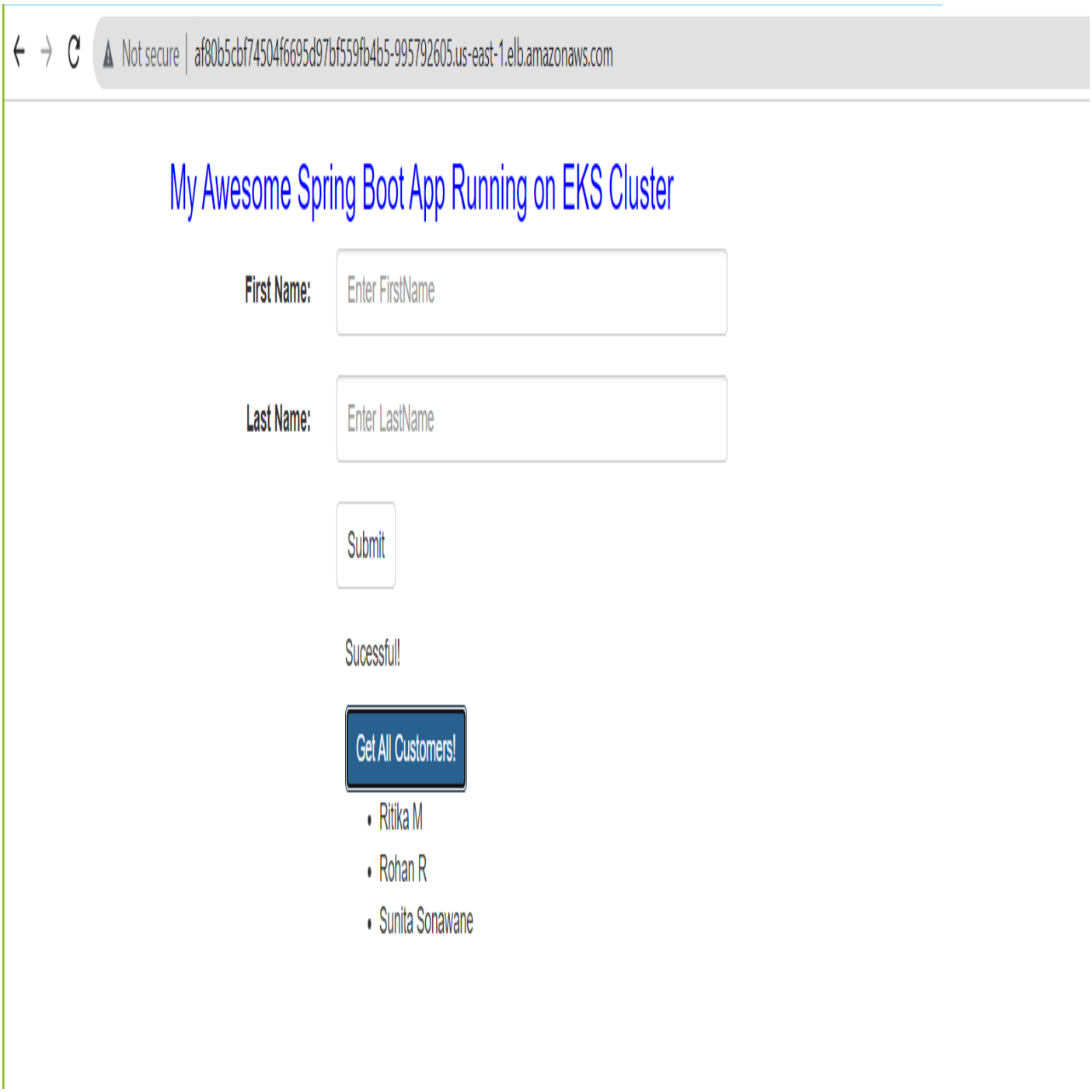
Sign In

Ritika_Mal

Apr 11

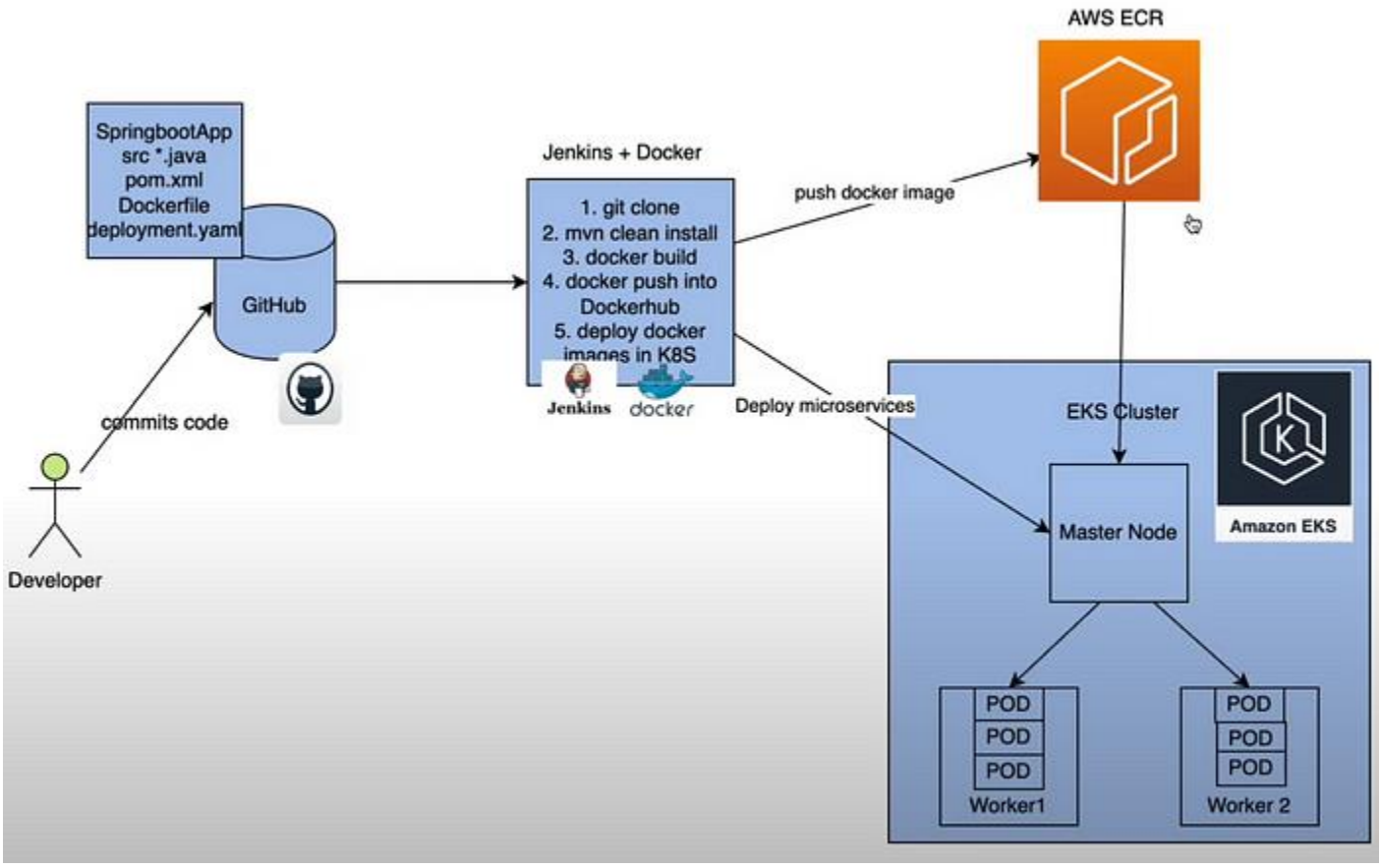
.
7 min read
.

🌻Automate Spring Boot App with Jenkins Pipeline using Amazon ECR and EKS🌻



Spring Boot App on EKS Cluster

Agenda — Deploy microservices into EKS using Jenkins Pipeline



Steps :

1. Launch an AWS EC2 Instance
2. Install Java, Maven
3. Install Jenkins and setup Jenkins on this EC2 Instance
4. Install **AWS CLI**
5. Install and setup eksctl
6. Install and setup kubectl
7. Create IAM Role with Administrator Access
8. Create an Amazon EKS cluster using eksctl
9. Create a Repository in AWS ECR
10. Install Docker

11. On the Jenkins console, install these plugins namely Docker, Docker pipeline and Kubernetes CLI
12. Build the jar file and package the same in Dockerfile
13. Create Credentials for connecting to Kubernetes Cluster using kubeconfig
14. Create Jenkins pipeline to deploy microservices into EKS cluster
15. Verify deployment using kubectl
16. Access the microservices app.
17. Deprovision/Delete the cluster

Step 1 — Set up an AWS T2 Medium Ubuntu EC2 Instance.

You can select an existing key pair, and enable HTTP and HTTPS Traffic. Launch the instance and once it is launched you can connect to it using the key pair. Label it as Jenkins.

Since Jenkins works on Port 8080, configure the Security Group. Add Custom TCP Port 8080 to access Jenkins using the Public IP Address of the EC2 Instance. Inbound Security Group would be now modified for our Jenkins.

Step 2 — Install Maven. If we install Maven, Java automatically gets installed by default.

```
sudo hostname Jenkins
sudo apt update
sudo apt install maven -y
mvn --version
```

```
ubuntu@ip-172-31-0-11:~$ mvn --version
Apache Maven 3.6.3
Maven home: /usr/share/maven
Java version: 11.0.18, vendor: Ubuntu, runtime: /usr/lib/jvm/java-11-openjdk-amd64
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "5.15.0-1031-aws", arch: "amd64", family: "unix"
```

Step 3 — Install Jenkins and setup Jenkins on this EC2 Instance

```
curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo
tee \
  /usr/share/keyrings/jenkins-keyring.asc > /dev/null
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins
```

Start Jenkins

```
sudo systemctl enable jenkins
sudo systemctl start jenkins
sudo systemctl status jenkins
```

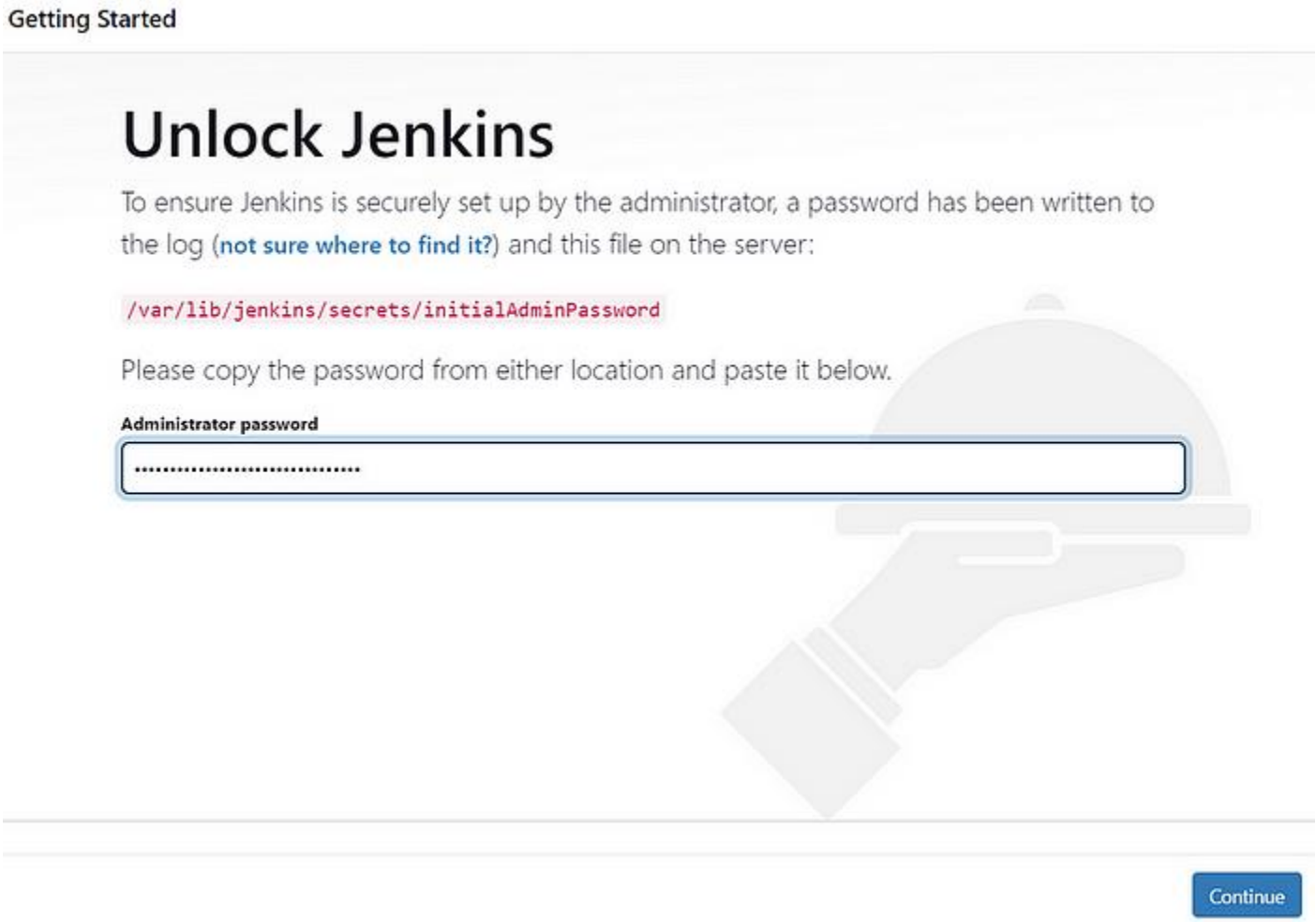
Now, Copy IP address of EC2 Instance and search in browser

Jenkins Terminal

Copy the password path and go to your terminal and run it using cat command

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

Copy the password and run it in browser



Install the suggested plugins,

Create a First Admin User and save it.

Getting Started

Create First Admin User

Username

admin

Password

••••••••

Confirm password

••••••••

Full name

Ritika Malhotra

E-mail address

writetoritika@gmail.com

Jenkins 2.387.1

Skip and continue as admin

Save and Continue

Copy this URL and paste it into a new tab and save it.

Instance Configuration

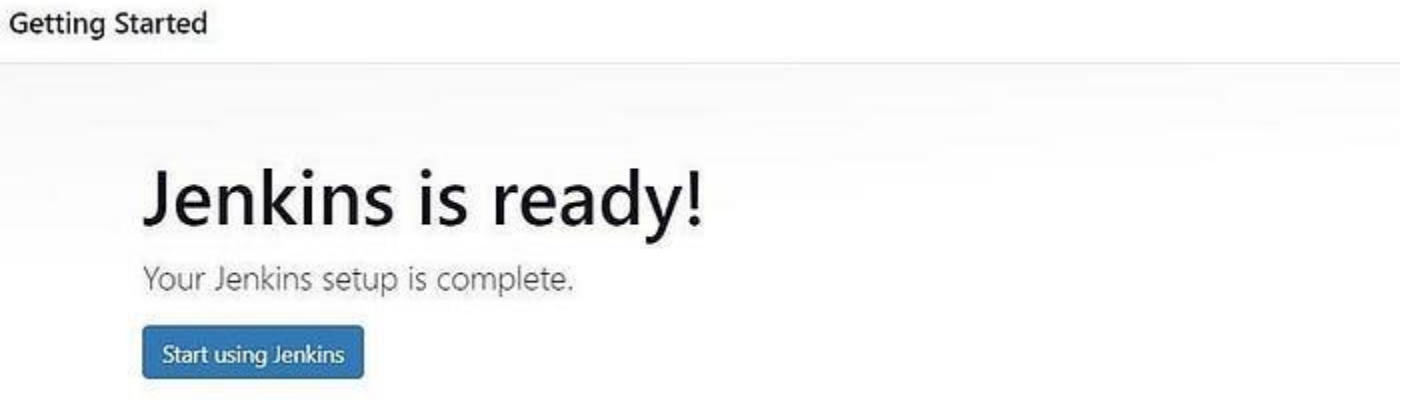
Jenkins URL:

http://54.164.7.106:8080/

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the `BUILD_URL` environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins is set up successfully now.



Jenkins is now ready!

Step 4 — Install AWS CLI


```
sudo apt install awscli
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
"awscliv2.zip"
sudo apt install unzip
unzip awscliv2.zip
sudo ./aws/install --update
aws --version
```

```
ubuntu@ip-172-31-0-122:~$ aws --version
aws-cli/2.11.11 Python/3.11.2 Linux/5.15.0-1031-aws exe/x86_64.ubuntu.22 prompt/off
ubuntu@ip-172-31-0-122:~$
```

Step 5 — Install eksctl

We have to install and set up eksctl using these commands.

```
curl --silent --location
"https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname
-s)_amd64.tar.gz" | tar xz -C /tmp
sudo mv /tmp/eksctl /usr/local/bin
eksctl version
```

```
ubuntu@ip-172-31-0-122:~$ curl --silent --location "https://github.com/weavewor
tl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp
ubuntu@ip-172-31-0-122:~$ sudo mv /tmp/eksctl /usr/local/bin
ubuntu@ip-172-31-0-122:~$ eksctl version
0.136.0
```

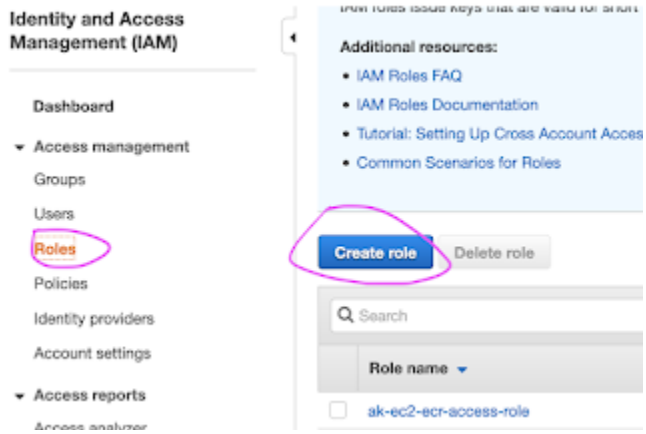
Step 6 — Install kubectl

```
sudo curl --silent --location -o /usr/local/bin/kubectl   https://s3.us-west-
2.amazonaws.com/amazon-eks/1.22.6/2022-03-09/bin/linux/amd64/kubectl
sudo chmod +x /usr/local/bin/kubectl
kubectl version --short --client
```

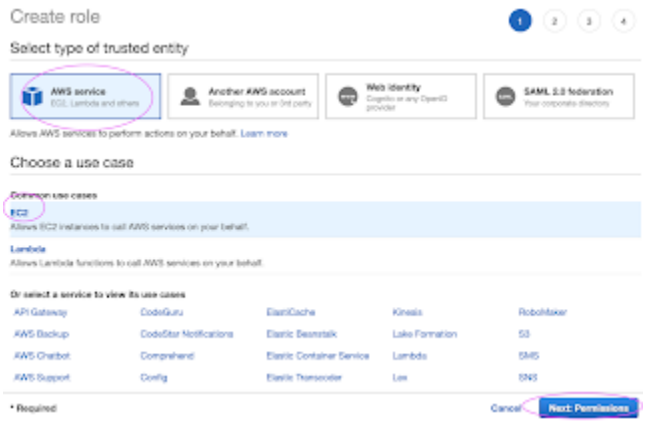
```
ubuntu@ip-172-31-0-11:~$ sudo curl --silent --location -o /usr/local/bin/kubectl   htt
amazon-eks/1.22.6/2022-03-09/bin/linux/amd64/kubectl
ubuntu@ip-172-31-0-11:~$ sudo chmod +x /usr/local/bin/kubectl
ubuntu@ip-172-31-0-11:~$ kubectl version --short --client
Client Version: v1.22.6-eks-7d68063
```

Step 7 — Create an IAM Role with Administrator Access

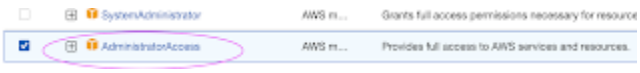
We need to create an IAM role with AdministratorAccess policy.
Go to AWS console, IAM, click on Roles. create a role



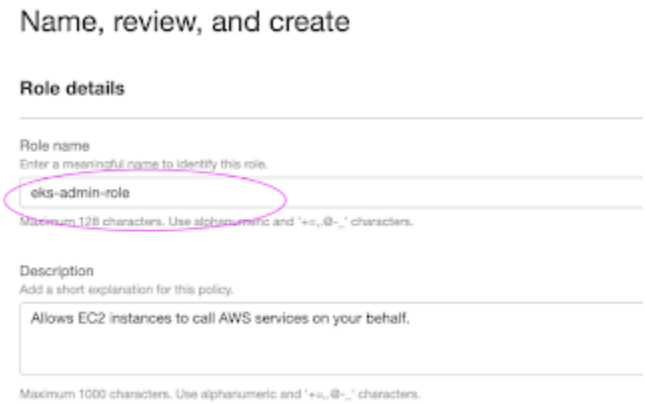
Select AWS services, Click EC2, Click on Next permissions.



Now search for AdministratorAccess policy and click

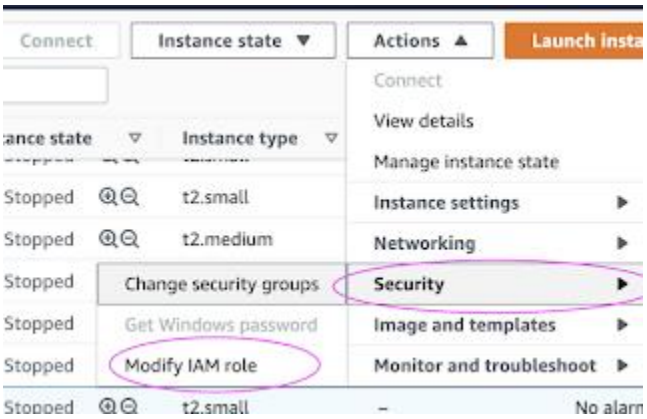


Skip on create tag. Now, give a role name and create it.

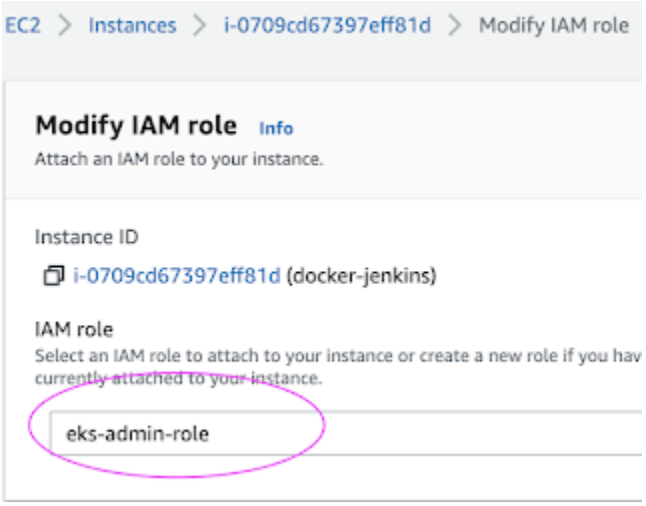


Assign the role to EC2 instance

Go to AWS console, click on EC2, select EC2 instance, Choose Security.
Click on Modify IAM Role



Choose the role you have created from the dropdown.
Select the role and click on Apply.



Now, the IAM role is attached to the Jenkins EC2 Instance.

Step 8 — Create an Amazon EKS cluster using eksctl

Switch from Ubuntu user to Jenkins user

```
sudo su - jenkins
```

You can refer to this URL to create an Amazon EKS Cluster.
https://docs.aws.amazon.com/eks/latest/userguide/create-cluster.html .

Here I have created cluster using eksctl

You need the following in order to run the eksctl command

1. Name of the cluster : — demo-eks
2. Version of Kubernetes : — version 1.24
3. Region : — region us-east-1
4. Nodegroup name/worker nodes : — nodegroup-name worker-nodes
5. Node Type : — nodegroup-type t2.small
6. Number of nodes: — nodes 2

This command will set up the EKS Cluster in our EC2 Instance. The command for the same is as below,

```
eksctl create cluster --name demo-eks --region us-east-1 --nodegroup-name my-
nodes --node-type t3.small --managed --nodes 2
```

This command would create an EKS cluster in Amazon. It would take atleast 15–20 minutes for this.

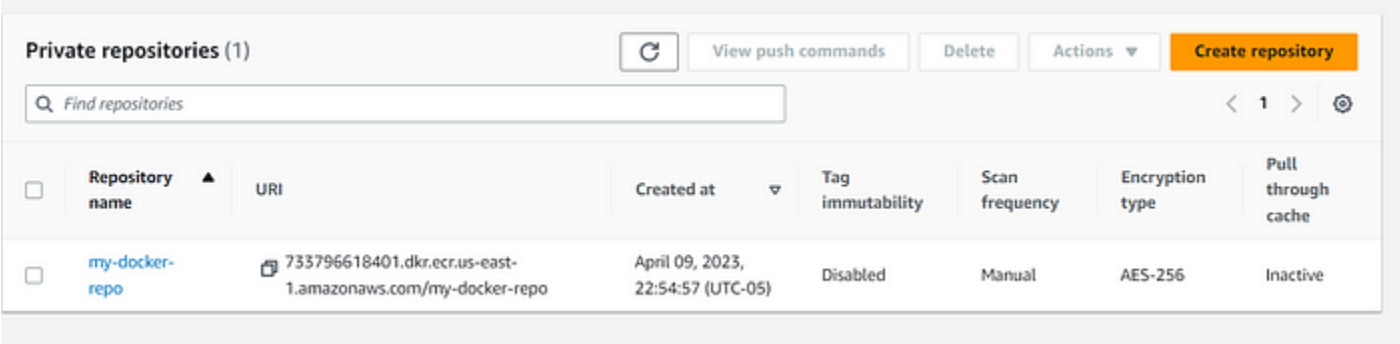
Step 9— Create an Amazon ECR

First, connect to your EC2 instance and install docker using these commands

```
sudo apt update
sudo apt install docker.io
docker --version
```

1. Create a Repository in AWS ECR
2. Create an IAM role with ContainerRegistryFullAccess
3. Attach an IAM role to the EC2 instance

Click on Create Repository, and Enter name for your repository



Create an IAM Role of ContainerRegistryFullAccess and attach the same to your EC2 Instance

Step 10 — Install Docker. Now Login to Jenkins EC2 instance, execute below commands:

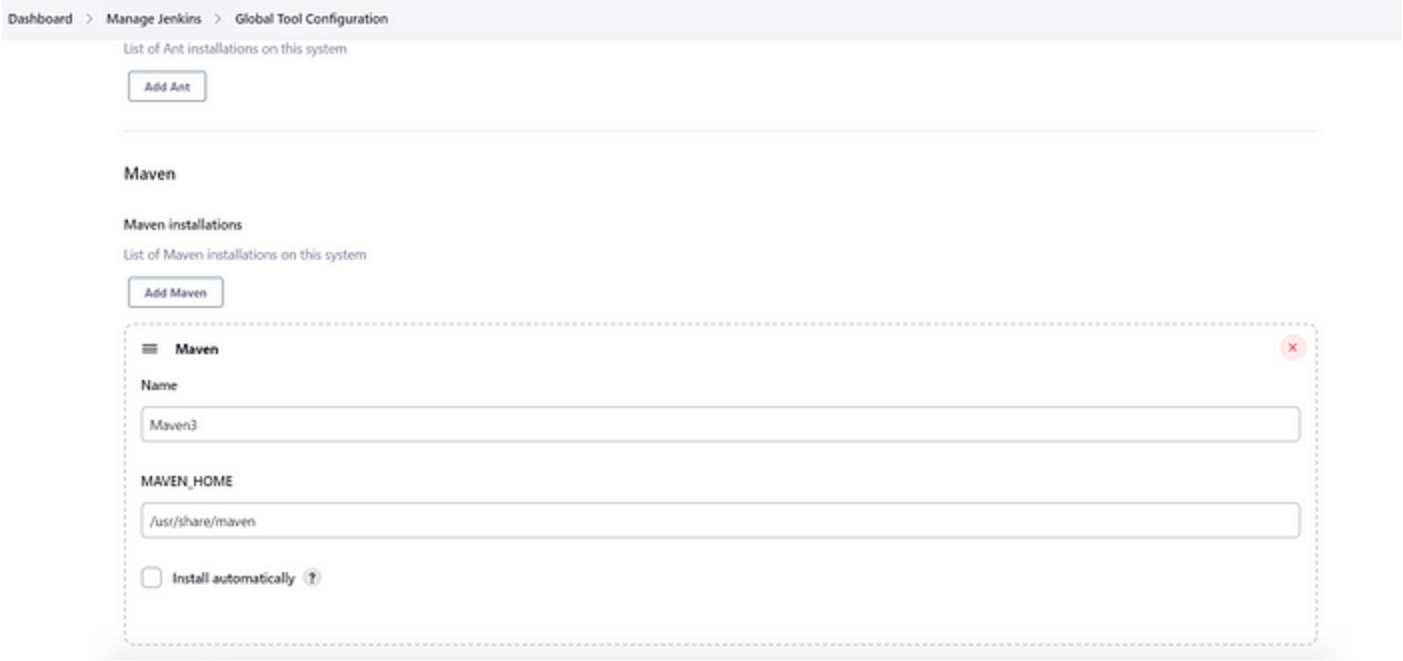
```
sudo apt-get update
sudo apt install docker.io
sudo usermod -a -G docker jenkins
sudo service jenkins restart
sudo systemctl daemon-reload
sudo service docker stop
sudo systemctl start docker
sudo systemctl enable docker
sudo systemctl status docker
```

Step 11 — Install the following plugins on Jenkins

- Docker
- Docker Pipeline
- Kubernetes CLI

Step 12 — Build the jar file and package the same in Dockerfile

On the Jenkins Dashboard, go to Global Tool Configuration and register Maven



Click on apply and save.

Step 13 — Create Credentials for connecting to Kubernetes Cluster using kubeconfig.


```
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data:
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUMvakNDQWVhZ0F3SUJBZ0lCQURBTkJna3Foa
2lHOXcwQkFRc0ZBREFWTVJNd0VRWURWUVFERXdwcmRXSmwKY201bGRHVnpNQjRYRFRJek1EUXhNRE
F6TWpVeU5Gb1hEVE16TURRd056QXpNalV5TkZvd0ZURVRNQkVHQTFVRQpBeE1LYTNWaVpYSnVaWFJ
sY3pDQ0FTSXdEUVlKS29aSWh2Y05BUUVCQlFBRGdnRVBBRENDQVFvQ2dnRUJBS1V4ClJHbHVMU1cv
anE3emhEd0swV0wwd0g2RHFSY1ZpeEpxd2l6cFZLQTNBWUx5cGpoZWZDTUIwVlRMUGZBVHd5YjAKU
1g0algyT2wwQmRVY25HTG5PNitwSHJoWVdCZS9abWFFSXBkaGNnRC9GQWpLR3ltN3UyaU02T0dtNl
ZSc0NNQwpxcmtZL29zY2dhMjNRb1dhb3VrS0RMYnQzUUc0TmFvTnhLYkU5R2hJZDJTTUVCd2lzc2d
ZY1d2Tkt5TGZQU0hHCmpxcVhlMlU5am1wRGVFZVdCSllQZWN5WVpFODNHTkVOWGtZc0NZY0ZvbS9n
L2RsM3dCSkV6YmtSVlE0Q0NEclcKTUVYem84cHNSR29UL2JEQ0xLaUdoNGhUc1Rld3RubWY1K0RiU
ExqVldZY095Z29iQm9FWDVIYjZIOEZoYkJyaApWY2lXNGc3cUVFdWtXcmNNL2pjQ0F3RUFBYU5aTU
Zjd0RnWURWUjBQQVFIL0JBUURBZ0trTUE4R0ExVWRFd0VCCi93UUZNQU1CQWY4d0hRWURWUjBPQkJ
ZRUZJU1ZWMW5RS2NJYmZhTkRxTlNTUTFOUlEyL0JNQlVHQTFVZEVRUU8KTUF5Q0NtdDFZbVZ5Ym1W
MFpYTXdEUVlKS29aSWh2Y05BUUVMQlFBRGdnRUJBQU02UFF2TDBEK1lkL0NJQzM4ZQpKS3ZndEZXe
U1CYjlTQkZVMWx6YndXdE44WnZWemdyai9EWHFCUVM2TjRSaE5GWnpvdzJlRVdWK013ZXladGhZCj
B0ZlVIVHVoZ2JuVXIyQ1NqM2ZLaFprNWVMVENxUVZYNmRRakJobE5Sbk1CbzZUTEJNY3pFVjJtMlR
sankxT0MKaklHTkk2Y1RTQlhmTTlLd0R6TmFFbmFKSHYrTm4vRWdaS1UxZjJsU2lBblB4bFZVczVE
SGU5QjUvdDZ6bWVWSgozYmJ2RCtOSEsraUx3Kzk4V3N6Ti9SU2EzdW9ISitZTmM1NGgvbmVSOUZ6e
k5nei8rczg3Y1paYm9MSUtqdlVxCmVQKzE4a2pDOXNUcEZycEZIdjVUMjY1S21tR1I0MmpCall0QW
UzdDA5QlVkQm85bnFDUEZlb25qQnF0Z3A0c3kKcExFPQotLS0tLUVORCBDRVJUSUZJQ0FURS0tLS0
tCg==
    server: https://47F4EFB258581727F4977E2E1324A187.gr7.us-east-
1.eks.amazonaws.com
  name: demo-eks.us-east-1.eksctl.io
```

```
- cluster:
    certificate-authority-data:
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUMvakNDQWVhZ0F3SUJBZ0lCQURBTkJna3Foa
2lHOXcwQkFRc0ZBREFWTVJNd0VRWURWUVFERXdwcmRXSmwKY201bGRHVnpNQjRYRFRJek1EUXhNRE
F6TWpVeU5Gb1hEVE16TURRd056QXpNalV5TkZvd0ZURVRNQkVHQTFVRQpBeE1LYTNWaVpYSnVaWFJ
sY3pDQ0FTSXdEUVlKS29aSWh2Y05BUUVCQlFBRGdnRVBBRENDQVFvQ2dnRUJBS1V4ClJHbHVMU1cv
anE3emhEd0swV0wwd0g2RHFSY1ZpeEpxd2l6cFZLQTNBWUx5cGpoZWZDTUIwVlRMUGZBVHd5YjAKU
1g0algyT2wwQmRVY25HTG5PNitwSHJoWVdCZS9abWFFSXBkaGNnRC9GQWpLR3ltN3UyaU02T0dtNl
ZSc0NNQwpxcmtZL29zY2dhMjNRb1dhb3VrS0RMYnQzUUc0TmFvTnhLYkU5R2hJZDJTTUVCd2lzc2d
ZY1d2Tkt5TGZQU0hHCmpxcVhlMlU5am1wRGVFZVdCSllQZWN5WVpFODNHTkVOWGtZc0NZY0ZvbS9n
L2RsM3dCSkV6YmtSVlE0Q0NEclcKTUVYem84cHNSR29UL2JEQ0xLaUdoNGhUc1Rld3RubWY1K0RiU
ExqVldZY095Z29iQm9FWDVIYjZIOEZoYkJyaApWY2lXNGc3cUVFdWtXcmNNL2pjQ0F3RUFBYU5aTU
Zjd0RnWURWUjBQQVFIL0JBUURBZ0trTUE4R0ExVWRFd0VCCi93UUZNQU1CQWY4d0hRWURWUjBPQkJ
ZRUZJU1ZWMW5RS2NJYmZhTkRxTlNTUTFOUlEyL0JNQlVHQTFVZEVRUU8KTUF5Q0NtdDFZbVZ5Ym1W
MFpYTXdEUVlKS29aSWh2Y05BUUVMQlFBRGdnRUJBQU02UFF2TDBEK1lkL0NJQzM4ZQpKS3ZndEZXe
U1CYjlTQkZVMWx6YndXdE44WnZWemdyai9EWHFCUVM2TjRSaE5GWnpvdzJlRVdWK013ZXladGhZCj
B0ZlVIVHVoZ2JuVXIyQ1NqM2ZLaFprNWVMVENxUVZYNmRRakJobE5Sbk1CbzZUTEJNY3pFVjJtMlR
sankxT0MKaklHTkk2Y1RTQlhmTTlLd0R6TmFFbmFKSHYrTm4vRWdaS1UxZjJsU2lBblB4bFZVczVE
SGU5QjUvdDZ6bWVWSgozYmJ2RCtOSEsraUx3Kzk4V3N6Ti9SU2EzdW9ISitZTmM1NGgvbmVSOUZ6e
k5nei8rczg3Y1paYm9MSUtqdlVxCmVQKzE4a2pDOXNUcEZycEZIdjVUMjY1S21tR1I0MmpCall0QW
UzdDA5QlVkQm85bnFDUEZlb25qQnF0Z3A0c3kKcExFPQotLS0tLUVORCBDRVJUSUZJQ0FURS0tLS0
tCg==
    server: https://47F4EFB258581727F4977E2E1324A187.gr7.us-east-
1.eks.amazonaws.com
  name: arn:aws:eks:us-east-1:733796618401:cluster/demo-eks
contexts:
- context:
    cluster: demo-eks.us-east-1.eksctl.io
    user: i-0d43aa396bb1b54f3@demo-eks.us-east-1.eksctl.io
  name: i-0d43aa396bb1b54f3@demo-eks.us-east-1.eksctl.io
- context:
    cluster: arn:aws:eks:us-east-1:733796618401:cluster/demo-eks
    user: arn:aws:eks:us-east-1:733796618401:cluster/demo-eks
  name: arn:aws:eks:us-east-1:733796618401:cluster/demo-eks
current-context: arn:aws:eks:us-east-1:733796618401:cluster/demo-eks
kind: Config
preferences: {}
users:
- name: i-0d43aa396bb1b54f3@demo-eks.us-east-1.eksctl.io
  user:
    exec:
      apiVersion: client.authentication.k8s.io/v1beta1
      args:
      - eks
      - get-token
      - --cluster-name
      - demo-eks
      - --region
      - us-east-1
      command: aws
      env:
      - name: AWS_STS_REGIONAL_ENDPOINTS
        value: regional
      provideClusterInfo: false
- name: arn:aws:eks:us-east-1:733796618401:cluster/demo-eks
  user:
    exec:
```

```
      apiVersion: client.authentication.k8s.io/v1beta1
      args:
      - --region
      - us-east-1
      - eks
      - get-token
      - --cluster-name
      - demo-eks
      - --output
      - json
      command: aws
```

Step 14 — Create Jenkins pipeline to deploy microservices into EKS cluster

Copy the pipeline code from below

```
pipeline {
   tools {
        maven 'Maven3'
    }
    agent any
    environment {
        registry = "account_id.dkr.ecr.us-east-1.amazonaws.com/my-docker-
repo"
    }

    stages {
        stage('Cloning Git') {
            steps {
                checkout([$class: 'GitSCM', branches: [[name: '*/main']],
doGenerateSubmoduleConfigurations: false, extensions: [], submoduleCfg: [],
userRemoteConfigs: [[credentialsId: '', url:
'https://github.com/writetoritika/springboot-app']]])
            }
        }
      stage ('Build') {
          steps {
            sh 'mvn clean install'
            }
      }
    // Building Docker images
    stage('Building image') {
      steps{
        script {
          dockerImage = docker.build registry
        }
      }
    }

    // Uploading Docker images into AWS ECR
    stage('Pushing to ECR') {
     steps{
         script {
                sh 'aws ecr get-login-password --region us-east-1 | docker
```

```
login --username AWS --password-stdin account_id.dkr.ecr.us-east-
1.amazonaws.com'
                sh 'docker push account_id.dkr.ecr.us-east-
1.amazonaws.com/my-docker-repo:latest'
          }
        }
      }

       stage('K8S Deploy') {
        steps{
            script {
                withKubeConfig([credentialsId: 'K8S', serverUrl: '']) {
                sh ('kubectl apply -f eks-deploy-k8s.yaml')
                }
            }
        }
       }
    }
}
```

Jenkins job

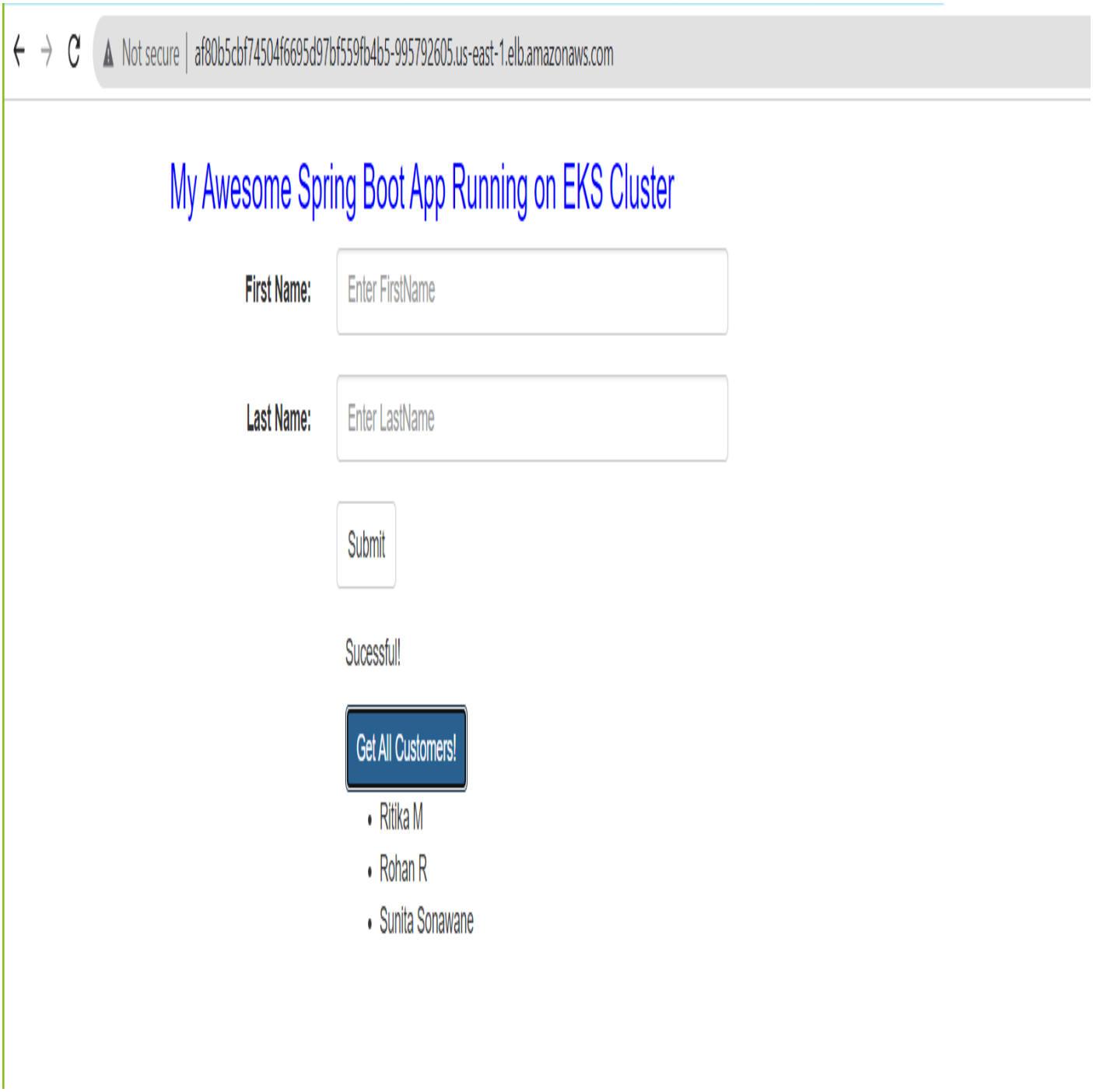
Step 16 — Verify the deployment using

```
kubectl get deployments
kubectl get pods
kubectl get svc
```

```
springboot-app   LoadBalancer   10.100.124.79   af80b5cbf74504f6695d97bf559fb4b5-995792605.us-east-1.elb.amazon
aws.com   80:31739/TCP   89m
jenkins@ip-172-31-0-11:~$ kubectl get svc
NAME             TYPE           CLUSTER-IP      EXTERNAL-IP
          PORT(S)        AGE
kubernetes       ClusterIP      10.100.0.1      <none>
          443/TCP        11h
springboot-app   LoadBalancer   10.100.124.79   af80b5cbf74504f6695d97bf559fb4b5-995792605.us-east-1.elb.amazon
aws.com   80:31739/TCP   91m
jenkins@ip-172-31-0-11:~$
```

Copy the URL for Load Balancer and when you open in another browser,

You will get this output



Step 17 — Deprovision the cluster either from the console or command line utility

Hope you liked this blog. You can follow the below github repo.

GitHub - writetoritika/springboot-app

You can't perform that action at this time. You signed in with another tab or window. You signed out in another tab or…

github.com

Follow me for more interesting projects. Thank you! 🙏

AWS

DevOps

Kubernetes

Devops Pipeline

Devops Project

53

53

More from Ritika_Mal

Follow

Azure Certified Solution Architect. Devops Engineer (Sister of Ansible, Knight of Vim, Chaplain of Terraform)

Published in **AWS Tip**

·Mar 16

⚙️Jenkins Multi Stage CICD Pipeline with Amazon EKS⚙️

Jenkins

8 min read

Jenkins

8 min read

Mar 23

⚙️Jenkins on AWS EC2 Instance Using Docker as Agents(Three Projects)⚙️

Jenkins

6 min read

Jenkins

6 min read

Published in **AWS Tip**

·Feb 22

🌼Learn Ansible With Real-Time Project 🌼

Ansible

4 min read

Ansible

4 min read

Mar 30

🌼Most commonly used Kubernetes Commands🌼

K8s

5 min read

K8s

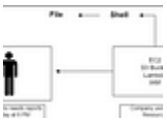
5 min read

Feb 20

🌼AWS Project Using Shell Scripting for DevOps Engineers 🌼

Shell Script

3 min read



Shell Script

3 min read

Ritika_Mal

101 Followers

Azure Certified Solution Architect. Devops Engineer (Sister of Ansible, Knight of Vim, Chaplain of Terraform)

Follow

More from Medium

Dmit

in

DevOps.dev

Blue-Green Deployment (CI/CD) Pipelines with Docker, GitHub, Jenkins and SonarQube

George Baidoo Jr.

in

AWS Tip

Deploying an AWS EC2 Instance Using a Terraform Module

Zaid Alissa Almaliki

How to Build a Kubernetes Cluster with Jenkins Using Terraform and Helm: Part One

headintheclouds

in

Dev Genius

Application Load Balancer vs Network Load Balancer

Help

Status

Writers

Blog

Careers

Privacy

Terms

About

Text to speech