Get the simple spring boot application

Step-1

Created EC2 instance called as a Jenkins server installed and configured Git, Java, Maven, Jenkins, Docker

GIT INSTALLATION:

```
sudo su -
sudo apt-get install software-properties-common
sudo add-apt-repository ppa:git-core/ppa -y
sudo apt-get update
sudo apt-get install git -y
git --version
```

JAVA INSTALLATION:

```
sudo add-apt-repository ppa:openjdk-r/ppa
sudo apt-get update
sudo apt-get install -y openjdk-8-jdk
```

MAVEN INSTALLATION:

```
sudo apt update
sudo apt install maven
mvn -version
```

JENKINS INSTALLATION:

```
wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-ke
echo deb https://pkg.jenkins.io/debian-stable binary/ | sudo tee /etc/apt/sour
sudo apt-get update
sudo apt-get install jenkins
```

DOCKER INSTALLATION:

```
sudo apt-get remove docker docker-engine docker.io containerd runc
sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl gnupg-agent s
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key ad
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ub
sudo apt-get update
sudo apt-get install -y docker-ce docker-ce-cli containerd.io
```

Step-2

Configure Jenkins with Docker:

by default Jenkins process runs with Jenkins User, which means any jenkins Jobs we run from jenkins console will be running jenkins user on Jenkins machine

we need to configure Jenkins user can run the docker commands by adding jenkins user to docker group

```
sudo usermod -aG docker jenkins
```

restart the Jenkins Service

```
sudo service jenkins restart
```

validate, run docker command with jenkins

```
su - jenkins  ## switch to jenkins user
docker ps     ## to list any containers running
docker pull nginx  ## pull a docker image
```

if the above commands execute without any error then we configured jenkins user properly

Step-3

Login to Jenkins UI:

```
 `http://IP:8080` in browser
enter `initialAdminPassword` the page to login ( cat /var/lib/jenkins/secrets/
click on `Install Suggested Plugins`
continue next and finish the setup.
```

Install reqired Plugins:

Docker Pipeline

cloudbees docker build and publish

Added Credentials to Jenkins

Github

Dockerhub

Configure JAVA - MAVEN - Git

Java configuration in Jenkins console

```
Manage Jenkins --> Global Tool Configuration --> JDK --> Add JDK
    Name: myjava
    JAVA_HOME: /usr/lib/jvm/java-8-openjdk-amd64
```

Maven Configuration in Jenkins console

```
Manage Jenkins --> Global Tool Configuration --> Maven --> Add Maven
    Name: maven3.6
    MAVEN_HOME: /opt/apache-maven-3.6.3
```

Git Configuration in Jenkins console

```
Manage Jenkins --> Global Tool Configuration --> Git --> Add Git
    Name: git
    MAVEN_HOME: /usr/bin/git
```

Creating CICD Pipeline:

---

1. Created new pipeline job and named as Buildimage

2. Written Jenkinsfile and Docker file and pushed to github repository where i have the src code and pom.xml

3. As soon as i pushed the code to github repository then jenkins job gets triggered which builds the docker container image and it saves the image in a Dockerhub registry. suppose if i changed a code then new docker container image will be created with new tag.

4. In github i have another repository for kubernetes manifest files. the name of the repo is kubernetesmanifest. we have deployment yaml files in this repo. so, this deployment yaml file should reference this newly created container inage.

5. when this jenkins job created docker container image after it will trigger another jenkins job update manifest,which will update the image in the deployment yaml file

6. Now i used Argo CD as Gitops tool

7. This gitops tool continously monitors this kubernets manifest repo and if the state in the kubernetes cluster deviates from the manifest files in the repo, gitops will grab those changes from the github repository and deploy to the kubernetes cluster

8. The jenkins job that is building the container image and updating the manifest file is the continous integration process.

9.

The gitops that is deploying the deployment file into kubernetes cluster is the continous deployment process. this is the gitops flow i used for this exercise.

Argo CD Installation:

---

created eks cluster

```
eksctl create cluster
```

Install Argocd:

```
kubectl create namespace argocd
kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/
```

Kubectl port-forwarding can also be used to connect to the API server without exposing the service. kubectl port-forward svc/argocd-server -n argocd 8080:443

after that logged in Argo CD UI & set up Argo cd app

By using loadbalcer DNS and port access from browser

Prometheus as a Monitoring tool:

Its Monitor the kubernetes cluster and create the alerts(cpu usage, memory usage)

prometheus runs as a daeomon on every node in the cluster

install prometheus:

I used Helm for prometheus installation. already installed helm

before install promentheus i have to install Metric sever

Deploy the Metrics Server with the following command:

```
kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/la
```

Verify that the metrics-server deployment is running the desired number of pods with the following command.

```
kubectl get deployment metrics-server -n kube-system
```

Deploying Prometheus: Create a Prometheus namespace.

```
kubectl create namespace prometheus
```

Add the prometheus-community chart repository.

```
helm repo add prometheus-community https://prometheus-community.github.io/helm
```

Deploy Prometheus.

```
helm upgrade -i prometheus prometheus-community/prometheus \
--namespace prometheus \
--set alertmanager.persistentVolume.storageClass="gp2",server.persistentVolume
```

Verify that all of the pods in the prometheus namespace are in the READY state

```
kubectl get pods -n prometheus
```

now we can see the control plane logs using Graph

Use kubectl to port forward the Prometheus console to your local machine.

```
kubectl --namespace=prometheus port-forward deploy/prometheus-server 9090
```

now open this port in localhost then we should see prometheus console.