CI/CD

Using webhook in github , jenkins pipeline is triggered where we use declarative pipeline in which multiple stages are defined like build and test the code and also code scanning stage Is also added to check the code quality , without any vulnerabilities it is accomplished by sonar qube ,if any stage fails then eamail send notification is triggerd.

Furthur docker image is build and stored to dockerhub

This is the CI part

For CD part we use GitOps model like ARgo CD and Argo image updater which are part of kubernetes itself . In this ,image updator continusly monitors the container registry with the new image and it picks the new image and update the new image in new repository which is merely for image manifests and as soon as new image is updated, argocd deploys the new image on kubernetes cluster

##

Created EC2 instance .openjdk -11 is installed

Intall jenkins, default port of jenkins is 8080 so to access jenkins, port 8080 is added in SG of EC2.

Now access url of jenkins

Create pipeline project (by clicking new item)

And in pipeline script from scm, path for repository, branch and file is provided to fetch the declarative pipeline script is stored in a file.

In jenkins file, we use agent as a docker which creates image to execute the all stages , once all the stages are done then container is deleted and resources are freed up and resources become available to run the other jobs, if we use instances then configuration and its management is needed which is considered as not a good practice to be implemented.

For docker agent, we need to have docker pipeline plugin,

Docker image is build and run the container which also has maven.

Now sonar qube scanner plugin is added in jenkins, add user sonar qube in the server you are working sudo su - sonarqube. and download binaries , unzip sonarqube folder,change permission and ownership of sonarqube file, in bin folder choose os which is linux ,now start sonar server , port 9000 for sonar server. Login to sonar server after accessing sonar url with port 9000

To authenticate sonar server with jenkins, in sonar- my account-security-generate token- copy token and add it to credential section of jenkins - system and global credentials- select secret text, paste token, paste sonarqube ID and create .

Now with root user of EC2 , install docker.io, grant permission to jenkins and ubuntu user to create,run, pull the container now restart docker and jenkins also.

Install EKs in ec2. Install Kubernetes controller using kubernetes operator which manages lifecycle of controller like upgrades to the controller ,versions or seamless updates. It also helps to install easily because it comes with default configuration as well.

Open operatorhub.io.,in kubernetes created, install OLM(operator lifecycle manager which manages opertaors running on the cluster and install operator.)

If pipeline sscript is written in jenkins itself then checkout stage is needed

In build stage, mvn clean target or package is executed and this mvn clean target or package finds pom.xml . pom.xml is responsible for getting dependecy’s runtime and help in building the application, (artifact) jar file is generated in target group.

Copy this jar file in docker file to execute this jar file (runs on 8080 port)

Static code analysys using sonarqube , put sonar url, mvn target to execute the sonar. It needs authentication token and sonar url

Steps to build the docker image with build tag accordingly and push to conatiner registry

For deployment, this stage updates the image tag and push deployment file to github repository if any change in deployment file is occurred, so for this stage put dockerhub and github credential in jenkins.

Go in manage jenkins- credentials -system-global cred- username and password for dockerhub and github secret text and add token of github token -add in jenkins

Restart jenkins

Add url of sonar in sonar stage.

Check argocd operator is running

Build from jenkins

CI is done

Shell script updated the repo

##

Create argocd controller using yaml script in argocd.yml file and Kubectl apply -f argocd.yml

Kubectl get pods to see the argocd pods are running

To run argocd on browser- kubectl edit svc argocd\_server(cluster name) - change to NodePort

Get url of argo cd using minicube service argocd-server,

Minicube service list

After checking pods are running-now- access argocd url

Admin, password from - kubectl get secret =argocd cluster - kubectl edit secret argocd cluster

Echo admin\_password | base64 -d == copy password without %

Login to argocd - source github url and path for deplyment.yml and Destination is Kubernetes cluster,namespace -create

If -- kubectl get deploy ,,copy springboot app and -- kubectl edit deploy spring-boot-app---change build number of image , refresh argocd -- will through error, application out of sync

Click diff- to see the change in the configuration-- argocd heal it by itself