Practical of Complete CICD Pipeline

24 October 2024 12:29

Frist we need to create an ec2 instance but free won't work as it contains docker sonar and all therefore ram and space will required more. So, we gonna take ubuntu + t2.large which contains 2 cpu and 8gb memory.

Now we will fetch the git repositroy of the java-maven-sonar-argocd-k8s to the terminal.

https://github.com/iam-veeramalla/Jenkins-Zero-To-Hero.git

Switch to the java-maven-sonar-argocd-k8s and then into the spring-boot-app.

Now install maven

- mvn clean package

Now as the docker file is already present in the application directory we will build the docker image.

Now running this application locally once.

- docker build -t ultimate-cicd-pipeline:v1 .

To set the 8081 port to the application and run it on locally.

- docker run -d -p 8020:8080 -t ultimate-cicd-pipeline:v1

Now the application is running locally as a docker image.
Using CICD now we have to deploy this application on the kubernetes.

Now we go to our ec2 instance.

So, Now we have to login to the ec2 instance using pem file.

- scp /mnt/c/Users/sansk/Downloads/cicd.pem sanskar@172.26.43.212:/home/sanskar cicd.pem
- chmod 400 /home/sanskar/cicd.pem
- ssh -i /home/sanskar/cicd.pem ubuntu@13.201.57.100

Now we will install open jdk into the server.
Prerequisites for jenkins -

- sudo apt update
- sudo apt install openjdk-17-jre

Now after verify we install jenkins in it.

- java -version

- curl -fsSL https://pkg.jenkins.io/debian/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 binary/ | sudo tee \
- /etc/apt/sources.list.d/jenkins.list > /dev/null
- sudo apt-get update
- sudo apt-get install jenkins

Now bydefault the jenkins server will be started at port 8080.

Now if you run this jenkins on local it won't work as inbound traffic rules are not configured on the local.

Basically they are ingress rules which prevents the incoming traffic.

Now we opened port for all the inbound incoming traffic by selecting all traffic and ipv4.

To check if jenkins is running.

- ps -ef | grep jenkins

- [ ] Debian -
- curl ifconfig.me

Public ip of ec2 on which jenkins is intalled and running and port 8080 which is given as by default.
http://13.201.57.100:8080/login?from=%2F

Now get the passoword

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

e299fba74abe4c05b23d961a85b8f281

Now just start jenkins as admin as its just for demo.
Now the first thing we will do we will create the pipeline.

If we choose pipeline option than the pipeline will be written in groovy code which can be saved in git repository so that we can collabrate to with others.

Jenkins also provides the option to put the jenkins file where your source code is which is pipeline script from SCM.

We can write jenkins file at the root of your application where we have the docker file.

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

```
git config --global user.name "sansugupta"
git config --global user.email "sanskargupta966@gmail.com"
```

To enter the repository into the creation of jenkins pipeline.
https://github.com/sansugupta/Jenkins-Zero-To-Hero.git

Now the Jenkins file for jenkins is created.

Once the pipeline is triggered it takes the reponsibility of creating docker conatiner executes all the stages inside it and at last destroy the conatainer automatically.

Now our repsonsibility is to write the docker file and jenkins is to trigger the docker container which has maven and docker in it.

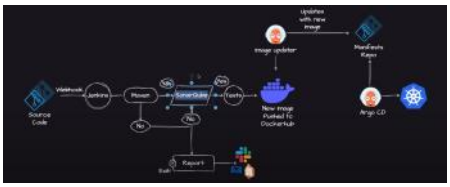
So first we will install the docker pipeline plugin in jenkins.

So now we have installed the docker pipeline into the jenkins and already it has maven.
It all happening in the jenkins of that AWS instance which we created.

Now we need to install sonar server. We need to do both configure the plugin of sonar in jenkins and install sonar on ec2 ser ver also.

SonarQube Scanner

Now we need to install Sonar Server on our instance.



Configure a Sonar -

```
apt install unzip
adduser sonarqube
wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.4.0.54424.zip
unzip *
chmod -R 755 /home/sonarqube/sonarqube-9.4.0.54424
chown -R sonarqube:sonarqube /home/sonarqube/sonarqube-9.4.0.54424
cd sonarqube-9.4.0.54424/bin/linux-x86-64/
./sonar.sh start
```

To install it first we create sonarqube user as an admin then switch into it.

Now downloading the sonarbinary as zip file.

```
wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.4.0.54424.zip
```

Install unzip after logout from sonarqube and then come back into it and then unzip the zipped file of sonarqube binary.

```
apt install unzip
unzip *
chmod -R 755 /home/sonarqube/sonarqube-9.4.0.54424
chown -R sonarqube:sonarqube /home/sonarqube/sonarqube-9.4.0.54424
```

Now we will switch to the directory on which platform we are on and then start sonar server which is by default on 9000 port.

```
cd sonarqube-9.4.0.54424/bin/linux-x86-64/

./sonar.sh start
```

http://13.201.57.100:9000/maintenance?return_to=%2F
By default.

Username - admin
Password - admin

As jenkins maven installed as docker and sonar spereately installed so how jenkins will authenticate with sonar.
For this we can generate a token for Jenkins in Sonar.

5eea31dd4791feb4daacac1fde969e96b08c3cfa

Now we can configure this token into the Jenkins like this -

Now as next step we need to install docker on our laptop directly as other things can be directly executed on the jenkins doc ker container.

Now we gonna install docker on our ec2 instance.

Switch to root user.

```
logout
```

Now providing the permissions and restarting the docker.

```
sudo apt install docker.io
usermod -aG docker jenkins
usermod -aG docker ubuntu
systemctl restart docker
```

Now restart Jenkins.

Now we will run minikube cluster on our local terminal.

minikube start --memory=4096 --driver=kvm2

Whenever we install kubernetes controller any these installations should take place with the kubernetes operators.

So, now we will install the Argo CD using the operator.
Go to -
Operatorhub.io

curl -sL https://github.com/operator-framework/operator-lifecycle-manager/releases/download/v0.28.0/install.sh | bash -s v0.28.0

Operators and Argo CD both will be installed by this.

kubectl create -f https://operatorhub.io/install/argocd-operator.yaml
kubectl get csv -n operators

Pom.xml file maintain the dependencies required for java using maven.

```
}
stages {
  stage('Checkout') {
    steps {
      sh 'echo passed'
      //git branch: 'main', url: 'https://github.com/iam-veeramalla/Jenkins-Zero-To-Hero.git'
    }
  }
  stage('Build and Test') {
    steps {
      sh 'ls -ltr'
      // build the project and create a JAR file
      sh 'cd java-maven-sonar-argocd-helm-k8s/spring-boot-app && mvn clean package'
    }
  }
  stage('Static Code Analysis') {
    environment {
      SONAR_URL = "http://34.233.125.26:9000"
    }
    steps {
      withCredentials([string(credentialsId: 'sonarqube', variable: 'SONAR_AUTH_TOKEN')]) {
        sh 'cd java-maven-sonar-argocd-helm-k8s/spring-boot-app && mvn sonar:sonar -Dsonar.login=$SONAR_AUTH_TOKEN
Dsonar.host.url=${SONAR_URL}'
      }
    }
  }
  stage('Build and Push Docker Image') {
    environment {
      DOCKER_IMAGE = "abhishekf5/ultimate-cicd:${BUILD_NUMBER}"
      // DOCKERFILE_LOCATION = "java-maven-sonar-argocd-helm-k8s/spring-boot-app/Dockerfile"
      REGISTRY_CREDENTIALS = credentials('docker-cred')
    }
    steps {
      script {
          sh 'cd java-maven-sonar-argocd-helm-k8s/spring-boot-app && docker build -t ${DOCKER_IMAGE} .'
          def dockerImage = docker.image("${DOCKER_IMAGE}")
```

```
-Dsonar.host.url=${SONAR_URL}'
      }
    }
  }
  stage('Build and Push Docker Image') {
    environment {
      DOCKER_IMAGE = "abhishekf5/ultimate-cicd:${BUILD_NUMBER}"
      // DOCKERFILE_LOCATION = "java-maven-sonar-argocd-helm-k8s/spring-boot-app/Dockerfile"
      REGISTRY_CREDENTIALS = credentials('docker-cred')
    }
    steps {
      script {
          sh 'cd java-maven-sonar-argocd-helm-k8s/spring-boot-app && docker build -t ${DOCKER_IMAGE} .'
          def dockerImage = docker.image("${DOCKER_IMAGE}")
          docker.withRegistry('https://index.docker.io/v1/', "docker-cred") {
              dockerImage.push()
          }
      }
    }
  }
  stage('Update Deployment File') {
      environment {
          GIT_REPO_NAME = "Jenkins-Zero-To-Hero"
          GIT_USER_NAME = "iam-veeramalla"
      }
      steps {
          withCredentials([string(credentialsId: 'github', variable: 'GITHUB_TOKEN')]) {
              sh '''
                  git config user.email "abhishek.xyz@gmail.com"
                  git config user.name "Abhishek Veeramalla"
                  BUILD_NUMBER=${BUILD_NUMBER}
                  sed -i "s/replaceImageTag/${BUILD_NUMBER}/g" java-maven-sonar-argocd-helm-k8s/spring-boot-app-manifests/deployment.yml
                  git add java-maven-sonar-argocd-helm-k8s/spring-boot-app-manifests/deployment.yml
                  git commit -m "Update deployment image to version ${BUILD_NUMBER}"
                  git push https://${GITHUB_TOKEN}@github.com/${GIT_USER_NAME}/${GIT_REPO_NAME} HEAD:main
```

Now we need to put the docker and git credentials inside the jenkins.

Go to Jenkins manage and add credentials of your docker-hub.

ID - docker-cred
username - sansugupta
Password - jhoncena@966

```
    steps {
      withCredentials([string(credentialsId: 'sonarqube', variable: 'SONAR_AUTH_TOKEN')]) {
        sh 'cd java-maven-sonar-argocd-helm-k8s/spring-boot-app && mvn sonar:sonar -Dsonar.login=$SONAR_AUTH_TOKEN
-Dsonar.host.url=${SONAR_URL}'
      }
    }
  }
  stage('Build and Push Docker Image') {
    environment {
      DOCKER_IMAGE = "abhishekf5/ultimate-cicd:${BUILD_NUMBER}"
      // DOCKERFILE_LOCATION = "java-maven-sonar-argocd-helm-k8s/spring-boot-app/Dockerfile"
      REGISTRY_CREDENTIALS = credentials('docker-cred')
    }
    steps {
      script {
          sh 'cd java-maven-sonar-argocd-helm-k8s/spring-boot-app && docker build -t ${DOCKER_IMAGE} .'
          def dockerImage = docker.image("${DOCKER_IMAGE}")
          docker.withRegistry('https://index.docker.io/v1/', "docker-cred") {
              dockerImage.push()
          }
      }
    }
  }
  stage('Update Deployment File') {
      environment {
          GIT_REPO_NAME = "Jenkins-Zero-To-Hero"
          GIT_USER_NAME = "iam-veeramalla"
      }
      steps {
          withCredentials([string(credentialsId: 'github', variable: 'GITHUB_TOKEN')]) {
              sh '''
                  git config user.email "abhishek.xyz@gmail.com"
                  git config user.name "Abhishek Veeramalla"
                  BUILD_NUMBER=${BUILD_NUMBER}
```

Now, the git-hub credentials.

Choose Secret-text for this as there is access key not password.

ID - github

Now we need to replace the Sonarqube url in the jenkins file of github repository for the sonarqube isntallation.

Now we will proceed with the continous delivery configurations.

Now, Do the build now.

Now after running the jenkins pipeline successfully and updated the image to the repository we will use argo cd to deploy thi s on the kubernetes.

Now we will create the Argo CD controller.

→ vim argocd-basic.yml

```
apiVersion: argoproj.io/v1beta1   # Changed from v1alpha1 to v1beta1
kind: ArgoCD
metadata:
  name: example-argocd
  labels:
    example: basic
spec: {}
```

→ kubectl apply -f argocd-basic.yml

→ kubectl get pods

Now all the confgurations related to controller and operators are done.

Now just go to argo cd UI to pull the lastest image from the git repository and deploy it into the kubernetes cluster using t he CD process.

As we need to run the Argo cd on our browser so we will change the cluster ip to the Node Port.

→ kubectl edit svc

→ minikube service argocd-server

→ minikube service example-argocd-server

Now to login
username- admin

for password argocd stores the password in secret.

→ kubectl get secret

→ kubectl edit secret example-argocd-cluster

Copy admin password

→ echo password |base64 -d

CJHB6X71fW3QTDAxIZVYLcs4EjanyKkO

Now,
Provide the name of application anything.
default
Sync - automatic
url of repository.

https://github.com/sansugupta/Jenkins-Zero-To-Hero

path of deployment file.

https://github.com/sansugupta/Jenkins-Zero-To-Hero/tree/main/java-maven-sonar-argocd-helm-k8s/spring-boot-app-manifests

cluster name - https://kubernetes.default.svc
Namespace - default

Now, Deploy.

→ kubectl get deploy