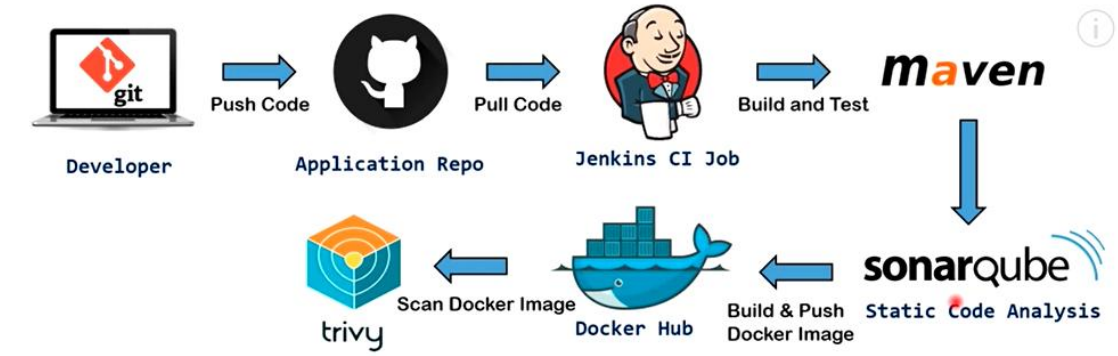
Devops Project: Deploy Application to kuberntes using Jenkins

Architecture



Configure Jenkins Master node

Create one ubuntu server in aws for jenkins master

enable security group port 8080 for jenkisn

login server enter below commands

#Install Java

$ sudo apt update

$ sudo apt upgrade

$ sudo nano /etc/hostname (to change the hostname as Jenkins-Master)

$ sudo init 6 (Reboot the server)

$ sudo apt install openjdk-17-jre

$ java -version

Install Jenkins

Refer--https://www.jenkins.io/doc/book/installing/linux/

```
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
```

$ sudo systemctl enable jenkins //Enable the Jenkins service to start at boot

$ sudo systemctl start jenkins //Start Jenkins as a service

$ systemctl status jenkins

Configure Jenkins Agent node

Create one ubuntu server in aws for jenkins agent

#Install Java

$ sudo apt update

$ sudo apt upgrade

$ sudo nano /etc/hostname (to change the hostname as Jenkins-Master)

$ sudo init 6 (Reboot the server)

$ sudo apt install openjdk-17-jre

$ java -version

#Install Docker

sudo apt-get install docker.io

sudo usermod -aG docker $USER (to give permission to current user)

sudo init 6

Uncommand publickeyauthorization and authorizedkeyfile in sshd configfile for both master and agent server

execute below commands in both servers

$ sudo nano /etc/ssh/sshd_config

$ sudo service sshd reload

Make passwordless authentication on both servers like you are doing in ansible

$ ssh-keygen OR $ ssh-keygen -t ed25519

$ cd .ssh

Jenkins Dashboard

open browser --> http:<private ip of master server>:8080

set password --> sudo cat /var/lib/jenkins/secrets/initialAdminPassword

install suggested plugins

login jenkins dashboard

Go to Manage jenkins->click nodes->click build in node->give no of executes 0->save

Add agent node in jenkins dashboard

Go to Manage jenkins->click nodes->click new node->node name,description(Jenkins-Agent), no of executers(2),remote root directory(/home/ubuntu),label(Jenkins-Agent),usage(use this node as much as possible),launch method(launch agent via ssh),host(private ip of jenkins agent server),credentials((click add --> kind(SSH Username with private key),id and description(Jenkins-Agent),Username(ubuntu),Privatekey(paste the content of masternode private key),credentials(select you already created),host key verification strategy(non verification strategy),save))

Now agent is added in jenkins dashboard.

To test above connectivity between masternode and agent node, run hello world job in dashboard

Go to jenkins dashboard->click new item->name(Test),select pipeline->select pipelinescript as helloword->save->build

Intergrate Maven to jenkins

go to manage jenkins-> click plugins->install thease(maven inetgration,pipeline maven integration,eclipse termurin insatller)

Tp cofigure above plugins,

go to manage jenkins->click tools->go to maven installastion and click add maven->name(Maven3),enabel install automaticall and click save,apply.

go to manage jenkins->click tools->go to jdk installastion and click add jdk->name(Java17),enabel install automatically and click add insataller as adoptium.net, version as 17.0.5+8 and click save,apply.

ADD Github credentials to jenkins

go to manage jenkins->click credentials->click new->give github username and token,id and description(github)->create

Pipeline Script in Github Repo

create jenkins file in your github repository

go to your repository which have your application->click add file-> create new file->name(Jenkinsfile)-> write a pipeline script->save the file

```
pipeline {
    agent { label 'Jenkins-Agent' }
    tools {
```

```
        jdk 'Java17'
        maven 'Maven3'
    }
 stages{
        stage("Cleanup Workspace"){
                steps {
                cleanWs()
                }
        }

        stage("Checkout from SCM"){
                steps {
                    git branch: 'main', credentialsId: 'github', url:
'https://github.com/Ashfaque-9x/register-app'
                }
        }

        stage("Build Application"){
            steps {
                sh "mvn clean package"
            }

       }

        stage("Test Application"){
            steps {
```

```
                sh "mvn test"
            }
        }
}
```

execute CI (code,build,test) in jenkins dashboward using manually

go to jenkins dashboard->click new item->name(register-app-ci),project(pipeline),enable discard old builds,max bulids to keep(2)-> go to pipeline option by scrolling down->SCM(git),repository url(github url),credentials(slect your github credentials),branch(*/main),scriptpath(Jenkinsfile)->apply and save-> clcik build now

SonarQube Server

Install and Configure the SonarQube

createone ubuntu server in aws and make port entry 9000 for soanrcube

login server and execute below comands

Update Package Repository and Upgrade Packages

```
$ sudo apt update
$ sudo apt upgrade
```

Add PostgresSQL repository

```
$ sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'
$ wget -qO- https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo tee /etc/apt/trusted.gpg.d/pgdg.asc &>/dev/null
```

Install PostgreSQL

```
$ sudo apt update
$ sudo apt-get -y install postgresql postgresql-contrib
```

$ sudo systemctl enable postgresql

Create Database for Sonarqube

$ sudo passwd postgres

$ su - postgres

$ createuser sonar

$ psql

$ ALTER USER sonar WITH ENCRYPTED password 'sonar';

$ CREATE DATABASE sonarqube OWNER sonar;

$ grant all privileges on DATABASE sonarqube to sonar;

$ \q

$ exit

Add Adoptium repository

$ sudo bash

$ wget -O - https://packages.adoptium.net/artifactory/api/gpg/key/public | tee /etc/apt/keyrings/adoptium.asc

$ echo "deb [signed-by=/etc/apt/keyrings/adoptium.asc] https://packages.adoptium.net/artifactory/deb $(awk -F= '/^VERSION_CODENAME/{print$2}' /etc/os-release) main" | tee /etc/apt/sources.list.d/adoptium.list

Install Java 17

$ apt update

$ apt install temurin-17-jdk

$ update-alternatives --config java

$ /usr/bin/java --version

$ exit

Linux Kernel Tuning

Increase Limits

$ sudo vim /etc/security/limits.conf

//Paste the below values at the bottom of the file

sonarqube - nofile 65536

sonarqube - nproc 4096

Increase Mapped Memory Regions

sudo vim /etc/sysctl.conf

//Paste the below values at the bottom of the file

vm.max_map_count = 262144

Sonarqube Installation

Download and Extract

$ sudo wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.9.0.65466.zip

$ sudo apt install unzip

$ sudo unzip sonarqube-9.9.0.65466.zip -d /opt

$ sudo mv /opt/sonarqube-9.9.0.65466 /opt/sonarqube

Create user and set permissions

$ sudo groupadd sonar

$ sudo useradd -c "user to run SonarQube" -d /opt/sonarqube -g sonar sonar

$ sudo chown sonar:sonar /opt/sonarqube -R

Update Sonarqube properties with DB credentials

$ sudo vim /opt/sonarqube/conf/sonar.properties

//Find and replace the below values, you might need to add the sonar.jdbc.url

sonar.jdbc.username=sonar

sonar.jdbc.password=sonar

sonar.jdbc.url=jdbc:postgresql://localhost:5432/sonarqube

Create service for Sonarqube

$ sudo vim /etc/systemd/system/sonar.service

//Paste the below into the file

```
[Unit]
Description=SonarQube service
After=syslog.target network.target

[Service]
Type=forking

ExecStart=/opt/sonarqube/bin/linux-x86-64/sonar.sh start
ExecStop=/opt/sonarqube/bin/linux-x86-64/sonar.sh stop

User=sonar
Group=sonar
Restart=always

LimitNOFILE=65536
LimitNPROC=4096

[Install]
WantedBy=multi-user.target
```

Start Sonarqube and Enable service

$ sudo systemctl start sonar

$ sudo systemctl enable sonar

$ sudo systemctl status sonar

Watch log files and monitor for startup

$ sudo tail -f /opt/sonarqube/logs/sonar.log

SonarQube Dashboard

open browser-> <public ip of sonarcube server>:9000

login soanrcube-> username:admin, password:admin

set new password and again login sonarcube

Integrate sonarcube with jenkins

go to soanrcube homepage->go to my account->click security->generate token name(jenkins-sonarcube-token),type(Global Analysis Token),no expire->click genearte token->copy token

Go to jenkins dashboard->manage jenkins->credentials->add new credentials ->kind(secret text),secret(paste token),id and description(jenkins-sonarcube-token)

to install soanrcube plugins

Go to jenkins dashboard->manage jenkins->plugins->install(sonarqupe scanner,sonarqube quality gates,quality gates)->enable restart jenkins

to configure soanrcube plugins

Go to jenkins dashboard->manage jenkins->system->go to sonarcube installations ->click add sonarcube->name(sonarqube-server),url(http:private ip of sonarqube server:9000),serevr authentication(select your sonarcube token)->apply and save

Go to jenkins dashboard->manage jenkins->tools->go to sonarcube scanner installations ->name(sonarqube-scanner)->apply and save

Add soanrcube pipeline script in your jenkinsfile in your github repo and run jenkins job again

stage("SonarQube Analysis"){

```
        steps {
            script {
                withSonarQubeEnv(credentialsId: 'jenkins-sonarqube-token') {
                    sh "mvn sonar:sonar"
                }
            }
        }
    }
```

create webhook from soanrcube dashboard

go to soanrcube dashboward->click adminstration-->select webhook under the confugution->click create webhook->name(sonarqube-webhook)->url(http:private ip of jenkins master:8080/sonarqube-webhook/)->create

add pipelinescript for quality gate in jenkinsfile and run jenkins job again

```
    stage("Quality Gate"){
        steps {
            script {
                waitForQualityGate abortPipeline: false, credentialsId: 'jenkins-sonarqube-token'
            }
        }
    }
```

DOCKER IMAGE

To build docker image and upload to dockerhub

install docker plugins in jenkins(docker pipeline,docker api,docker-build-step,cloudbees docker build and publish,docker,docker commons)->enable restart jenkis when insatlling plugins

go to maanage jenkins->credentials->add new->username(dockerhub username),password(access token of dockhub account,id and description(dockerhub)

add pipeline script for docker image in jenkinsfile and run jenkins job manually

```
    environment {
                APP_NAME = "register-app-pipeline"
                 RELEASE = "1.0.0"
                 DOCKER_USER = "ashfaque9x"
                 DOCKER_PASS = 'dockerhub'
                 IMAGE_NAME = "${DOCKER_USER}" + "/" + "${APP_NAME}"
                 IMAGE_TAG = "${RELEASE}-${BUILD_NUMBER}"
                JENKINS_API_TOKEN = credentials("JENKINS_API_TOKEN")
      }


          stage("Build & Push Docker Image") {
                steps {
                     script {
                          docker.withRegistry('',DOCKER_PASS) {
                               docker_image = docker.build "${IMAGE_NAME}"
                          }


                          docker.withRegistry('',DOCKER_PASS) {
                               docker_image.push("${IMAGE_TAG}")
                               docker_image.push('latest')
                          }
                     }
                }
```

```
        }
    stage("Trivy Scan") {
            steps {
                script {
                    sh ('docker run -v /var/run/docker.sock:/var/run/docker.sock
aquasec/trivy image ashfaque9x/register-app-pipeline:latest --no-progress --scanners vuln
--exit-code 0 --severity HIGH,CRITICAL --format table')
                }
            }
        }

    stage ('Cleanup Artifacts') {
            steps {
                script {
                    sh "docker rmi ${IMAGE_NAME}:${IMAGE_TAG}"
                    sh "docker rmi ${IMAGE_NAME}:latest"
                }
            }
        }
```

now image is uploaded in your dockerhub account.

BOOSTRAP SERVER FOR EKSCTL

Setup Bootstrap Server for eksctl and Setup Kubernetes using eksctl

create one ubuntu server and login

sudo apt update

sudo apt upgrade

sudo nano etc/hostname (Change name as EKS-Bootstrap-Server)

sudo reboot or sudo init 6

Install AWS Cli on the above EC2

Refer--https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html

$ sudo su

$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"

$ apt install unzip, $ unzip awscliv2.zip

$ sudo ./aws/install

OR

$ sudo yum remove -y aws-cli

$ pip3 install --user awscli

$ sudo ln -s $HOME/.local/bin/aws /usr/bin/aws

$ aws --version

Installing kubectl

Refer--https://docs.aws.amazon.com/eks/latest/userguide/install-kubectl.html

$ sudo su

$ curl -O
https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.1/2023-04-19/bin/linux/amd64/kubectl

$ ll , $ chmod +x ./kubectl //Gave executable permisions

$ mv kubectl /bin //Because all our executable files are in /bin

$ kubectl version --output=yaml

Installing eksctl

Refer---https://github.com/eksctl-io/eksctl/blob/main/README.md#installation

$ curl --silent --location "<https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz>" | tar xz -C /tmp

$ cd /tmp

$ ll

$ sudo mv /tmp/eksctl /bin

$ eksctl version

create IAM role with adminstrator access and attach to above instance for eksctl access.

Setup Kubernetes using eksctl

Refer--https://github.com/aws-samples/eks-workshop/issues/734

$ eksctl create cluster --name virtualtechbox-cluster \

--region ap-south-1 \

--node-type t2.small \

--nodes 3 \

$ kubectl get nodes

ArgoCD

ArgoCD Installation on EKS Cluster and Add EKS Cluster to ArgoCD
==========

1) First, create a namespace

$ kubectl create namespace argocd

2) Next, let's apply the yaml configuration files for ArgoCd

$ kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml

3) Now we can view the pods created in the ArgoCD namespace.

$ kubectl get pods -n argocd

4) To interact with the API Server we need to deploy the CLI:

$ curl --silent --location -o /usr/local/bin/argocd https://github.com/argoproj/argo-cd/releases/download/v2.4.7/argocd-linux-amd64

$ chmod +x /usr/local/bin/argocd

5) Expose argocd-server

$ kubectl patch svc argocd-server -n argocd -p '{"spec": {"type": "LoadBalancer"}}'

6) Wait about 2 minutes for the LoadBalancer creation

$ kubectl get svc -n argocd

(In this result you will get the loadbalancer link to access argocd in browser. copy the link and paste in browser and seup argocd)

7) Get pasword and decode it.

$ kubectl get secret argocd-initial-admin-secret -n argocd -o yaml

$ echo WXVpLUg2LWxoWjRkSHFmSA== | base64 --decode

(use this password to setup newpassword for argocd in browser)

Add EKS Cluster to ArgoCD

9) login to ArgoCD from CLI

$ argocd login a2255bb2bb33f438d9addf8840d294c5-785887595.ap-south-1.elb.amazonaws.com --username admin

10)

$ argocd cluster list

11) Below command will show the EKS cluster

$ kubectl config get-contexts

12) Add above EKS cluster to ArgoCD with below command

$ argocd cluster add i-08b9d0ff0409f48e7@virtualtechbox-cluster.ap-south-1.eksctl.io --name virtualtechbox-eks-cluster

Configure argocd to deploy pods on EKS and Automate ArgoCD Deployment job using GitOps Github Repository

take another github repository --> gitops-register-app(this repository contains manifest files for kubernetes)

go to argocd dashboard->click settings->click repositories-click connect-> via (HTTPS).type(git),project(default),repo url(give above repo url),username and token(your github account username and token)->click connect

go to argocd dashboard->click new app->name(register-app),project(default),sync policy(automatic),enable(prune resourcs,self heal),repository url(give above repo url),path(./),destination(select your eks cluster url),namespace(default)->click create

Now deployment has done.

now ypu can view your svc --> kubectl get svc

(in the result it shows loadbalancer service dns name and copy that)

open browser-> <loadbalancer dnc>:8080 --> it will show tomcat

open browser-> <loadbalancer dnc>/webapp --> it will show your application register app

To automate above deployment process

Go to jenkins dashboard->click new item-> name(gitops-register-app-cd),project(pipeline),discard old build,max builds(2),enable(this project is parameterized),add string parameter(name:IMAGE_TAG),enable(trigger build remotely),authentication token(gitops-token),pipeline definition(pipeline script from SCM),SCM(git),url(gitops repo url),credentials(github credemtilas),branch(*/main),scriptpath(Jenkinsfile)->apply and save.

Edit jenkinsfile in your application repository(register-app)

add below script also.

```
        stage("Trigger CD Pipeline") {
                steps {
                    script {
                        sh "curl -v -k --user clouduser:${JENKINS_API_TOKEN} -X POST -H 'cache-control: no-cache' -H 'content-type: application/x-www-form-urlencoded' --data 'IMAGE_TAG=${IMAGE_TAG}' 'ec2-13-232-128-192.ap-south-1.compute.amazonaws.com:8080/job/gitops-register-app-cd/buildWithParameters?token=gitops-token'"
                    }
                }
            }
        }
```

Generate jenkins api token

Go to jenkisn dashboard ->click configure under your user profile->->click add new token under API token->name(JENKINS_API_TOKEN)->click generate->copu the token amd save in your local(because it will not appear later)

Go to jenkisn dashboard ->managae jenkins->credentilas->add credentials->kind(Secret text),Secret(paste jenkins api token)->id and description(JENKINS_API_TOKEN)-.>click create

app JENKINS_API_TOKEN varaiable under your environment in jenkinsfile.

Create jenkinsfile in gitops-register-app repository

```
pipeline {
    agent { label "Jenkins-Agent" }
    environment {
            APP_NAME = "register-app-pipeline"
    }

    stages {
        stage("Cleanup Workspace") {
            steps {
                cleanWs()
            }
        }

        stage("Checkout from SCM") {
                steps {
                    git branch: 'main', credentialsId: 'github', url:
'https://github.com/Ashfaque-9x/gitops-register-app'
                }
        }

        stage("Update the Deployment Tags") {
            steps {
                sh """
                    cat deployment.yaml
```

```
                    sed -i 's/${APP_NAME}.*/${APP_NAME}:${IMAGE_TAG}/g'
deployment.yaml
                    cat deployment.yaml
                """
            }
        }

        stage("Push the changed deployment file to Git") {
            steps {
                sh """
                    git config --global user.name "Ashfaque-9x"
                    git config --global user.email "ashfaque.s510@gmail.com"
                    git add deployment.yaml
                    git commit -m "Updated Deployment Manifest"
                """
                withCredentials([gitUsernamePassword(credentialsId: 'github', gitToolName:
'Default')]) {
                   sh "git push https://github.com/Ashfaque-9x/gitops-register-app main"
                }
            }
        }

    }
}
```

Schedule your job in jenkins

Go to your job in jenkins dashboard->enavle poll scm under build triggers->cronjob(* * * * *)it will monitor github repo everymin and trigger job

Verify all working fine

Go to your github repo(register-app)->connect codesapce->open index.jsp file(nano index.jsp)->add new line amd save->git add . ->git commit -m "1st commit" -> git push origin main

Now your jenkins job automaticall triggered.

go and check dockerhub(if any new image updated)

go to gitops-register-app repo(see change happen in deplyment.yam file)

go to argocd and go to app (if app using latest docker image)

open browser(http:<loadbalancer dns>:8080/webapp/)

13) $ kubectl get svc

========== Cleanup ==========

$ kubectl get all

$ kubectl delete deployment.apps/virtualtechbox-regapp //it will delete the deployment

$ kubectl delete service/virtualtechbox-service //it will delete the service

$ eksctl delete cluster virtualtechbox --region ap-south-1 OR eksctl delete cluster --region=ap-south-1 --name=virtualtechbox-cluster //it will delete the EKS cluster

SUMMARY

create aws ubuntu server for jenkins master(install jdk and jenkins)

create aws ubuntu server for jenkins agent(install jdk and docker)

setup jenkins dashboard and configure jenkins agent server on it

Add maven plugins and credentials,installations in jenkins dashboard(MAVEN 7)

Add jdk installations in jenkins dashboard(JAVA 17)

Add Github Credentials in jenkins dashbaord

Writh pipelinescript on Github repo

Create a new job and test CI Process in jenkins(code,build,test)

Install docker plugins on jenkins and add credentials of dockerhub in jenkins

Build docker image and upload into dockerhub and scan using trivy using piplinescript

Create aws ubuntu server fo sonarcube(install postgresqal,jdk,sonarcube)

Install sonarcube and quality gates plugins on jenkins dashboard

Add pipeline script for soanrcube and quality gates

Create aws ubuntu server for eksctl,kubectl and argocd(install eksctl,kubectl,argo cd)

Add eksctl cluster with argocd by setup argocd using loadbalancer service dns

Prepare another repo in github for k8s manifestfiles and connect repo with argo cd for deployment

Add pipeline script for argocd deployment

Create another jenkinsfile pipeline script in new repo

Generate jenkins api token and add in jenkins dashboard

Authomate job in jenkins dashboard using pollscm cronob

Verify all process by connect your remote repo and change index.jsp file