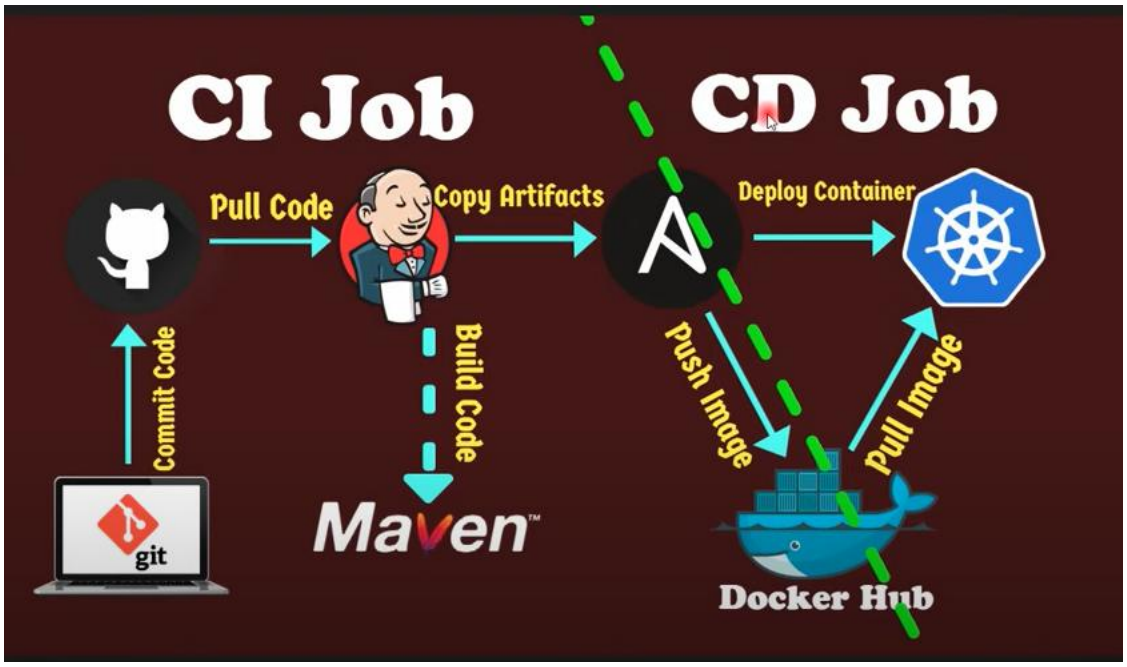
CICD Pipeline to Deploy to AWS EKS Cluster Using Jenkins

Architecture



Step 1: Install and Configure Jenkins

Launch Amazon linux2 instance in Aws console and allow port 8080.

Login the instance.

```
$ sudo yum update –y
```

```
sudo wget -O /etc/yum.repos.d/jenkins.repo \
    https://pkg.jenkins.io/redhat-stable/jenkins.repo
$ sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
$ sudo yum upgrade
$ amazon-linux-extras install epel
$ sudo amazon-linux-extras install java-openjdk11 -y
$ yum install java-11-amazon-corretto -y
$ sudo yum install jenkins -y
$ sudo systemctl enable jenkins          //Enable the Jenkins service to start at boot
$ sudo systemctl start jenkins           //Start Jenkins as a service
$ java -version
$ javac -version
$ systemctl status jenkins
```

Now you can able to access jenkins via browser <instance puplici ip>:8080

Set password and install suggested plugins in jenkins.

Start using jenkins.

Step 2 : Install and Configure Maven

Will configure maven on the above same server.

Login the instance.

Copy the download link from https://maven.apache.org/download.cgi

```
$ sudo su    & cd ~
$ cd /opt
$ wget https://dlcdn.apache.org/maven/maven-3/3.9.3/binaries/apache-maven-3.9.3-bin.tar.gz
$ tar -xzvf apache-maven-3.9.3-bin.tar.gz
$ ls

$ mv apache-maven-3.9.3 maven

$ ll

$ cd maven

$ cd bin/

$ ./mvn -v

$ cd ~

$ pwd

$ ll -a           //It will show the hidden files also

$ vim .bash_profile

$ find / -name java-11*

//enter below lines below the 2nd fi

M2_HOME=/opt/maven

M2=/opt/maven/bin

JAVA_HOME=/usr/lib/jvm/java-11-openjdk-11.0.19.0.7-1.amzn2.0.1.x86_64

PATH=$PATH:$HOME/bin:$JAVA_HOME:$M2_HOME:$M2

$ echo $PATH

$ source .bash_profile

$ echo $PATH

$ mvn -v
```

Step3 : Configure maven with jenkins and create maven test project to build artifacts

Now Go to jenkins dashboard.

Install maven plugins(maven integration)

Add maven under maven installations(maven path: /opt/maven)

Add jdk under jdk installations(java path:
/usr/lib/jvm/java-11-openjdk-11.0.19.0.7-1.amzn2.0.1.x86_64)

enable github plugin under installed plugins option.

restart jenkins.

Create new job with maven project->git repo url->Build(Goals and option-> clean install)->apply and save

Build now. You can see artifacts produced in this->go to job->workspace->webapp->target-> .war file

Step 4:Ansible server setup and Ansible installation

create 1 amazon linux2 server in aws console. enable   port range(8080-8090)

login instance.

$ sudo nano /etc/hostname

$ useradd ansadmin

$ passwd ansadmin

$ visudo

ansadmin ALL=(ALL)   NOPASSWD: ALL   //add this in sudo file.

$ cd /etc/ssh

$ nano sshd_config(enable password authentication yes)

$ service sshd reload

sudo su - ansadmin

$ ssh-keygen(generate key because we have to enable passworlaess athentictication between ansible and bootstrap server)

public key is at /home/ansadmin/.ssh/id_rsa.pub

$ sudo su

  amazon-linux-extras install ansible2

$ ansible --version

Step 5: Integrate ansible with jenkins

mangage jenkins->system->puplich over ssh->add ssh server->hostname(public ip),username(ansadmin),password9use password authentication)->save

login ansible server

```
$ cd /opt
```

```
$ sudo mkdir Docker
```

```
$ sudo chown ansadmin:ansadmin Docker
```

mangage jenkins->system->post build actions->select send build artifacts over ssh ->Name(Ansible-Server),Source files(webapp/target/*.war),remove prefix(webapp/target)->remote directory(//opt//Docker)->apply and save.

Build now.

Now you can see aritifacts are tehre in ansible server(/opt/Docker)

Step 6: Install and configure docker on ansible server

```
$    sudo yum install docker
```

```
$ sudo usermod -aG docker ansadmin
```

```
$ id ansadmin
```

```
$ sudo service docker start
```

```
$ sudo systemctl start docker
```

```
$ nano Dockerfile
```

```
FROM tomcat:latest

RUN cp -R /usr/local/tomcat/webapps.dist/* /usr/local/tomcat/webapps

COPY ./*.war /usr/local/tomcat/webapps
```

Step 7 : Create ansible playbook to create docker image and push image to dockerhub

```
$ ifconfig (to fetch ip of ansible server)
$ sudo nano /etc/ansible/hosts

[ansible]

local-host-ip
```

$ ssh-copy-id local-host-ip(adding self ssh key so that ansible playbook can be connect to self host)

$ nano regapp.yml

```
---

- hosts: ansible


  tasks:

  - name: create docker image

    command: docker build -t regapp:latest .

    args:

     chdir: /opt/Docker


  - name: create tag to push image onto dockerhub

    command: docker tag regapp:latest ashfaque9x/regapp:latest


  - name: push docker image

    command: docker push ashfaque9x/regapp:latest
```

docker login(login your dockerhub account)

```
$ ansible-playbook regapp.yml --check

$ ansible-playbook regapp.yml
```

docker images(now you can view images)

Step 8: Update jenkins job to use ansible playbook

mangage jenkins->system->post build actions->Exec Command:ansible-playbook /opt/Docker/regapp.yml

Step 9: Setup Bootstrap Server for eksctl

Create 1 amazon linux2 server in aws console.

login instance.

```
sudo su
```

# Install AWS Cli

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

$ unzip awscliv2.zip

$ sudo ./aws/install
```

# Installing kubectl

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

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

$ chmod +x ./kubectl

$ mv kubectl /bin    OR $ mv kubectl /usr/local/bin

$ kubectl version --output=yaml
```

#Installing or 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

$ sudo mv /tmp/eksctl /bin      OR    $ sudo mv /tmp/eksctl /usr/local/bin

$ eksctl version
```

create iam role in aws console and give admistrator access policy to access EKS cluter and attach it o bootstrap server.

# Setup Kubernetes using eksctl

```
eksctl create cluster --name virtualtechbox-cluster \

--region ap-south-1 \

--node-type t2.small \

$ kubectl get nodes
```

# Create deployment Manifest File

Refer===https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

$ nano regapp-deployment.yml

```
apiVersion: apps/v1

kind: Deployment

metadata:

  name: virtualtechbox-regapp

  labels:

     app: regapp


spec:

  replicas: 2

  selector:

    matchLabels:

      app: regapp


  template:

    metadata:

      labels:
          app: regapp
      spec:
        containers:
        - name: regapp
          image: ashfaque9x/regapp
          imagePullPolicy: Always
          ports:
          - containerPort: 8080
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
```

# Create Service Manifest File

Refer===https://kubernetes.io/docs/tutorials/services/connect-applications-service/

$ nano regapp-service.yml

```
apiVersion: v1
kind: Service
metadata:
  name: virtualtechbox-service
  labels:
    app: regapp
spec:
  selector:
      app: regapp

  ports:
    - port: 8080
      targetPort: 8080

  type: LoadBalancer
```

enable passwordauthenticstion yes (in bootstrap server /etc/ssh/sshd_config)

passwd root(set password for root)

service sshd reload

Step 10: Integrate Bootstrap Server with Ansible

Login ansible server

```
$ nano /etc/ansible/hosts
[ansible]
localhost

[kubernetes]
BootStrap-Server-IP
```

$ ssh-copy-id root@BootStrap-Server-IP(so that ansible playbook can integrate with eks cluster which is in root user of bootstrap server)

Step 11: Create Ansible Playbook to Run Deployment and Service Manifest files

$ nano kube_deploy.yml

```
---

- hosts: kubernetes

  user: root


  tasks:

    - name: deploy regapp on kubernetes

      command: kubectl apply -f regapp-deployment.yml


    - name: create service for regapp

      command: kubectl apply -f regapp-service.yml


    - name: update deployment with new pods if image updated in docker hub

      command: kubectl rollout restart deployment.apps/virtualtechbox-regapp
```

$ ansible-playbook kube_deploy.yml(run the playbook)

Now go to bootstarp server and check deployment has done.

sudo su

kubectl get all

copy loadbalancer dns and paster browser:8080/webapp (you can see your application is running)

Step 11 : Create CD job freestyle project for deployment in jenkins

go to create job ->freestyle project->go to post build actions->select ansible server->Exec command:ansible-playbook /opt/Docker/kube_deploy.yml->apply and save.

Now go to CI job in jenkins->configure->select poll scm(*****)->post build action(build another projects)->give your CD job->apply and save.

Step 12: verify CICD job by doing test commit on github repo.

Go to your github repo and change one line readme file and save.

Now your jenkins CI job will automatically trigger and after finishing CI job CD job also automatically trigger.

Cleanup commands

```
$ kubectl delete deployment.apps/virtualtechbox-regapp
```

```
$ kubectl delete service/virtualtechbox-service
```

```
$ eksctl delete cluster virtualtechbox --region ap-south-1
```