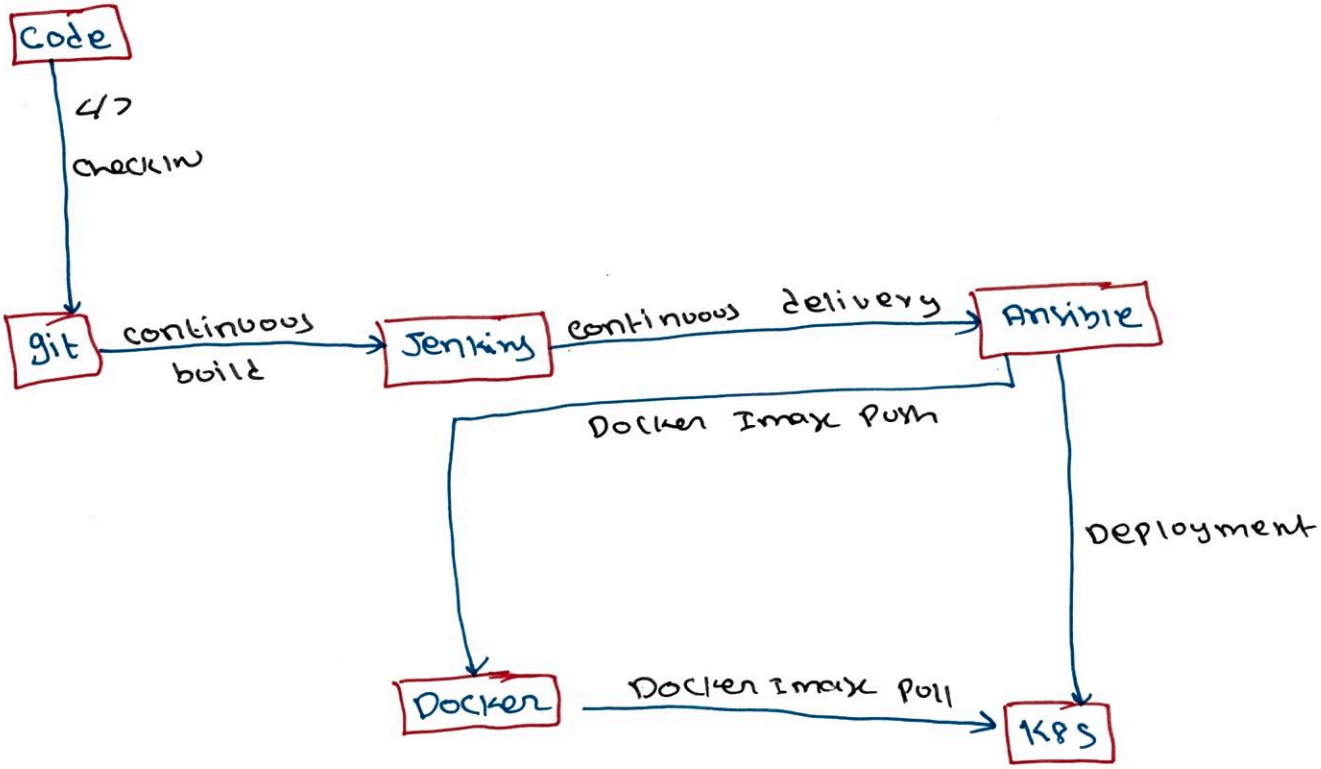
Devops- Project : 1

Tools used for this Project

1. Git :→ for source code mgmt
2. Jenkins :- To create CI/CD Pipelines
3. maven : To build tool
4. Ansible :- for configuration mgmt & deployment
5. Docker : is for target environment to host our app
6. K8S :- is to manage our containers
7. Aws :- To setup this environment

Devops flow :-



Step: 1 Setup CI/CD with jenking, git, maven & Tomcat

a) * Setup Jenking

b) * Run a test job

c) * Setup & configure maven & git

d) * Setup Tomcat server

e) * Installing additional required Pluging

f) * Integrating Git, maven in jenking job

g) * Run CI/CD job

flow

Step: 2 Introducing Docker

a) Setting up Docker environment

b) managing Docker with Ansible

c) Docker Hub repository

d) writing a Docker file

e) Run a job

flow

Step 3:- Integration with Ansible

a) Setting up Ansible environment

b) Integrating Ansible with Jenking

c) writing a Ansible playbook to deploy a continer

d) Run a job

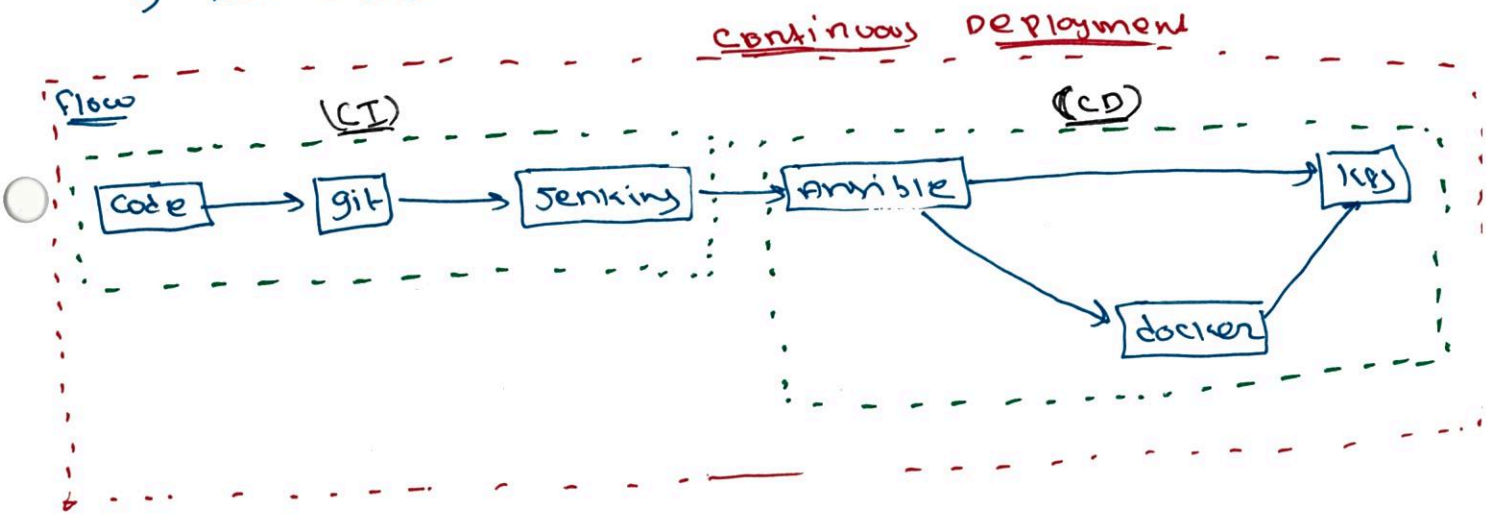
Flow

Step:4 :- Introducing k8s (kubernetes)

a) setting up k8s envivonment

b) writing deployment & service file

c) Run a job



⇒ Resources : required sample code (hello world) & / Aws / git / Ansible / docker / Jenkins / nexus / nginx / sonarqube / Tomcat.

STEP 1 : Setup CI/CD with Jenkins, git, maven & Tomcat

1. Launch EC2 instance & install Jenkins & configure Jenk

a) setup jenkins

⇒ remove java older version" yum remove java-1.7.0*

⇒ Install java 1.8 " yum install java-1.8* -y". (# java -version)

⇒ Set Home path for java

$ find /usr/lib/jvm/java-1.8* | head -n 3

↳ it will show where the jre file is available

↳ copy the path (jre path)

$ cd ~

↳ To go home dir

$ vi .bash_Profile ⇒ "to create Java Home path"

JAVA_HOME = Paste the Java path

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

$ echo $JAVA-HOME

↳ it shows /usr/lib/jvm/jre (it has to full path f

that exit the user & login) & check echo $JAVA-H

⇒ go to $ jenkins download site and follow the steps to install jenkins

1. get the repo & install

$ yum install jenkins

$ service jenkins status | start | restart | stop

=> open browser ip':8080, here will ask administrator password

$ cat path they given to get password

=> provide key in the password & select start using jenkins, it will open jenkin's console and change admin password

↳ admin → configure → enter new password

=> have to setup java home path

↳ manage jenkins → Global Tool configuration → Add JDK

name: JAVA_HOME

JAVA_HOME: ("To get $ echo $JAVA_HOME")

↳ Apply

---

5) Run a test job

→ open Jenkins → new item → name: my_first_job → select Free style project. → ok

↳ description: my first job

↳ source code mgmt: none (as of now, because there is no plugins)

↳ Build → Add build step → execute shell

command: echo "welcome to devops project"

↳ Apply → save

↳ "Build now" (or) if want to modify select "configure"

↳ check in Build History

↳ can see the console o/p by clicking #1

c) Setup & configure maven & git

↳ install & setup Git:-

If want change the EC2 instance (Jenkins server) hostname change as Jenkins

$hostname jenkins

$ sudo su -

→ Install git

$ yum install git -y

→ Install git pluging in Jenkins

↳ open Jenkins → manage Jenkins → manage Pluging → Available

→ search for github → select GitHub → Install without restart.

→ Setup git in Jenkins

↳ manage Jenkins → Global Tool configuration → Git name: github

↳ Apply → save

↳ Steup maven:-

↳ go to EC2 instance (Jenkins server)

$ cd ~

$ cd opt

$ wget http:// get link form download maven

$ tar -xvzf apache-maven.xx.xx.tar.gz

$ mv apach-maven.xx.xx maven (to chang name for easy)

$ vi ~/.bash_profile

↳ under Java_Home

M2_HOME = /opt/maven

M2 = /opt/maven/bin → save & Logoff & login

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

$ echo $M2

$ mvn --version (to check maven version.)

→ go to jenkins console to install maven pluging

↳ manage Jenkins → manage Pluging → Available → search maven → select maven Integration → Install without restart.

→ setup maven in Jenkins

↳ manage Jenkins → Global Tool configuration → Add maven

Name: M2_HOME

MAVEN_HOME: /opt/maven

↳ Apply → save

⇒ create Job to build with maven

⇒ Jenkins → new Item → name: my_first_maven_Build → select maven project → Descri: First maven build → Source code management

↳ go to git → select hello-world project → copy the URL

↳ pom.xml required for maven pro

→ selet git → Paste the Repository URL.

→ Branch: */master

→ Build → Root Pom: Pom.xml

→ Go ↳ goals & options: clean install package.

↳ apply save.

⇒ If any changes → selet project name → configure & change.

& → select Build now

↳ can check the build History #1 → to open Build co

↳ once deploy → can be workspace folder (copy the code here)

⇒ go to jenkins server & check the projects under workspace dir

& cd /var/lib/jenkins/workspace/

↳ see the Project names

⇒ next

d) Setup Tomcat Server :-

→ use new EC2 instance (Tomcat Server) or use existing Jenkins server, but change the Tomcat port number 8080 to 8090, it will conflits because both tomcat & Jenkins use same port number 8080 :

→ Install Java & wget

→ change the hostname " # hostname Tomcat"

→ Install Tomcat

↳ $ cd ~

$ cd /opt

$ wget http:// get the url from Tomact downloads site . tar.gz

$ tar -xvzf apache-tomcat-8.xx.tar.gz (to unzip)

$ mv apache-tomcat-8xx. 5.43 tomcat (to chan name for easy)

$ cd tomcat

$ cd bin can see startup.sh & shutdown.sh

↳ start & stop the tomcat service.

$ ./startup.sh

↳ goto browser ip:8090 (can access tomcat console)

↳ it will not allow to login (edit context.xml)

find / -name context.*xml

↳ it will show the path & edit the context.xml file

comment the value class name "<!--" "-->"
start end

↳ do same thin in another file as well

↳ $ stop & start the service.

⇒ update the users information in the tomcat-users.xml file
↳ (/opt/tomcat/conf/)

<role rolename = "manager-gui" />

<user username = " admin" password = "admin" roles = "manager-gui" />

⇒ Deploy a war file on Tomcat server

e) Install additional requirement Plugins in Jenkins

→ manage Jenkins → manage plugins → Available → Select Deploy to container & install without restart

f/g) → new Item → name: Deploy_on-Tomcat-server → maven project

→ ok → Description: deploy on Tomcat server (vm) → ~~Re~~ git repository

url: paste the hello world url → branch: */master → Root pom: pom.xml → Goals & options: clean install package. → Select Add Post build action → Deploy war/ear to a container

WAR/EAR files: **/*.war

Container: Add container (Tomcat 8/9)

↳ credentials: add (admin/admin)

Tomcat url: IP:8090

→ Apply → save

→ Build now (open the console o/p to check)

→ The deploy/war file will save in webapps location in Tomcat server → Note:- cd /opt/tomcat/webap

→ go to browser http://IP:8080/webapp/ → (can see o/p)

⇒ How to build a Job when there is a new change in code (usin Poll scm

→ Jenkins → Select the project → configure → Build Triggers → Poll scm

Schedule: * * * * * (like Corn Job)

↳ it will check every min in the git hub repository, if any code change, it will build automatically.

Note:- hello world → webapp → src → main → webapp → index.jsp

Step 2: Docker :(deploy Docker using Jenkins)

a) Docker setup

→ Launch new EC2 Instance (Docker-Host) → change hostname as Docker-I

yum install docker -y (it will install community edition)

service docker start/stop/status/restart

docker PS (it will show running containers)

docker pull tomcat:latest (to pull tomcat image)

docker image ls (list out images)

docker run -d --name tomcat-container -P 8080:8080 tomcat:late

docker rm container ID

docker ~~rmi~~ PS

> note: add user
> # useradd dockeradmin
> # Passwd dockeradmin
> # usermod -aG docker dockeradmin
> ↳ to add docker group
> # id dockeradmin

b) Integrating Docker with Jenkins

→ open Jenkins Console → manage Jenkins → ~~search~~ manage Plugins →

search for Publish over SSH → Install

To add Docker credentials

→ manage Jenkins → config system → Publish over SSH : SSH Server : Add

SSH server name : docker-Host

Host name : ec2 instance private ip (10.10.1.xx)

username : dockeradmin → advanced

password : xxxxxx

☒ use password based authentication.

Password : xxxxxx → Test configuration → Apply
↳ save

note:- Enable the password authentication Docker-Host EC2 instance

vi /etc/ssh/sshd_config

↳ PasswordAuthentication yes

service sshd reload

c) Jenkins Job to deploy in a docker server

→ Jenkins Job to copy artifacts on to DockerHost

→ Jenkins → new item → name: Deploy_on_Docker → copy from: Deploy-on-Tomcat-server → ok → Description: Deploy on Docker →

↳ Source code mgmt: git → Repository URL: paste the url

↳ Branch: */master

↳ Root Pom: Pom.xml

↳ goals & options: clean install package

↳ Post-build Action → Add Post-build action → send build artifact over SSH

SSH server name: docker-Host (by default)

Soure files: /webapp/target/*.war (will get in ~~Jenkins~~ /var/lib/jenkins/workspace/Deploy-on Tomcat server/webapp/target/*.war)

(Optional) Remove prefix: /webapp/target (it don't want target)

Remote directory: . (current directory) in docker)

Exec cmd:

↳ Apply → save

→ before build Job go and check docker admin home dir

↳ Build now

d) create dockerfile

→ go to Docker-Host instance → login in dockeradmin → in home dir copy the webapp.war file.

→ vi Dockerfile

```
FROM tomcat : latest
MAINTAINER PAVAN
COPY ./webapp.war /usr/local/tomcat/webapps
     local dir      ↳ container dir
```

$ docker build -t devops-project . (it will pull the tomcat image, & it will build new image: devops-project)

$ docker images , $ docker run -d --name devops-container -p 8080:8080 devops-project

→ go to browser ip: 8080/webapp

e) Deploy a war file on container using Jenkins (CI/CD)

→ Jenkins console → new item: Deploy_on_container → copy from: Deploy_on_Docker → ok → Description: Deploy on container → git: url → Pom: pom.x

↳ Post-build Actions

↳ SSH Publishers

↳ SSH Server name: docker-Host

Source files: webapp/target/*.war

Remove Prefix: webapp/target

Remote directore: •

Exec command: cd /home/dockeradmin;

docker build -t devops-image . ;

docker run -d --name devops-container -P 8080:8080

devops-image;

→ Apply → save

Note: before build job delete the container, image & webapp.war file in home directory.

↳ Build now

↳ go to browser ip':8080/webapp

docker-Host

Note: The problem is already in use by container, can't run 2nd job with same name can't deploy a container, so to overcome these kind of problems we use Ansible (deployment tool)

Step 3: Integration with Ansible:

a) Ansible setup

→ open EC2 instance (Ansible-server) → change host name: "ansible-control-n

```
# yum install python -y
# python --version
# yum install python-pip
# pip install ansible
# ansible --version
# mkdir /etc/ansible  (create a director for ansible)
# useradd ansadmin
# passwd ansadmin  (add this user to sudo file)
# vi sudo
    ↳ add  ansadmin  ALL=(ALL)  NOPASSWD: ALL
# yum install docker -y
# service docker start
# usermod -aG docker ansadmin  (ansadmin add in docker gro
```

→ Enable password authentication

```
# vi /etc/ssh/sshd_config
    ↳ passwordAuthentication yes
# service sshd reload
```

⇒ Switch to ansadmin account

```
# su - ansadmin
# ssh-keygen  (to generate ssh key)
# cd .ssh  (to check keys "id_rsa, id_rsa.pub" → have to secure. → this key copy in the target environment
```

⇒ go to Docker_Host instance → ~~sudo~~ add user ansadmin

```
# useradd ansadmin    # passwd ansadmin
```

⇒ go to Ansible-Server

su - ansadmin

ssh-copy-id ansadmin@docker-HostIP (to copy the .pub file)

ssh-copy-id localhost

now can can login without password to Docker-Host instance

ssh ansadmin@docker-HostIP

exit

⇒ now test the connectivity by ovin ping test

cd /etc/ansible

sudo vi hosts

~ 10.10.1.100 (docker-Host IP)

~ localhost

~ 10.10.1.50 (Jenkins-Server IP)

ansible all -m ping (-m is module)

b) Integrate ansible with Jenkins:

→ Jenkins console → manage Jenkins → ~~manage plugins~~ → configure system

→ Perviously we add SSH Server docker-Host & add another SSH server

ansible-server

↳ Publish over SSH → Add

SSH Server name: ansible-server

Host name: ansible-server Private IP

username: ansadmin

Advanced

Use Pwd authentication

password: xxxxxx

Test configuration

↳ apply → save

→ we need to create jenkins job, that jenkin job's when ever I ru jenkins can able to copy artifacts on to target environment

↳ Jenkins → newitem: Deploy_on_container_using_Ansible

↳ copy from: Deploy_on_container → ok

↳ Description: Deploy on container using ansible

↳ scm → git: url (github repository url) → branches: */mas

↳ pom: pom.xml

↳ goals & options: clean install package

↳ post-build Actions → SSH server name: select ansible-server

Source files: webapp/target/*.war

Remove prefix:

Remote dir: //opt//docker (provide double slash)

exec cmd: ansible-playbook -i /opt/docker/hosts /opt/docker/simple-devops-i deploy.yml

save & build job

note:- go to ansible-server login in ansadmin

```
# cd /opt
# sudo mkdir docker
# sudo chown -R ansadmin:ansadmin /opt/docker
# ls -l /opt  (it shows aws, docker dir) //

# cd /opt/docker
# vi dockerfile
        FROM tomcat:latest
        MAINTAINER PAVAN
        COPY ./webapp.war /usr/local/tomcat/webapps

# vi simple-devops-image.yml (simple-devops-projects.yml)
---
- hosts: all
  become: true
  tasks:
  - name: build docker image using warfile
    command: docker build -t simple-devops-image .
```

vi hosts
↳ docker local host

```
        args:
          chdir: /opt/docker        → save
```

→ befor that check docker imass & container

→ To run # Play book " # ansible-playbook -i hosts simple_devops-project-ymd

```
    - name: create container using simple-devops-image
      command: docker run -d --name simple-devops-container -
               8080:8080 simple-devops-image
```

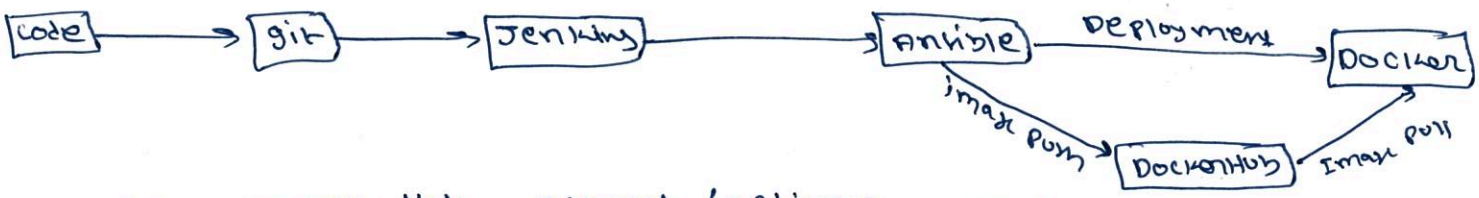
↳ save //

⇒ cd /opt/docker

vi simple-devops-project.ymd

```
---
- hosts: all
  become: true

  tasks:
  - name: stop current running container
    command: docker stop simple-devops-container
    ignore_errors: yes

  - name: remove stopped container
    command: docker rm simple-devops-container
    ignore_errors: yes

  - name: remove docker image
    command: docker rmi simple-devops-image
    ignore_errors: yes

  - name: build docker image using war file
    command: docker build -t simple-devops-image .
    args:
      chdir: /opt/docker

  - name: create container using simple-devops-image
    command: docker run -d --name simple-devops-container -p
             8080:8080 simple-devops-image.
```

c) Docker Hub Integration with Ansible



1. create Docker Hub account (optional create repository)
2. go to Ansible- server instance → $ docker images (if have delete)
3. create images using ansible playbook (cd docker)
   - $ ansible-playbook -i hosts simple-devops- project.yml
   - $ ↳ it will create image & container
   - $ docker images
   - $ docker ps -a
4. Push the image to docker hub
   - ↳ create a tag for image " docker tag simple-devop-image yan /simple-devops-image
   - ↳ To push his image to docker hub (1st login docker a/c)
   - $ docker login (provide credentials)
   - $ docker push yanikils/ simple- devops- image
   - ↳ go to docker a/c & check.
5. Remove the images in Ansible-server & pull the image form docker
   - $ docker rmi yanikils/ simple- devops- image
   - $ docker pull yanikils/simple- devops- image
   - $ docke images

⇒ go to docker-host instance

$ su - ansadmin (login in ansadmin)

$ id (to check group)

$ chmod usermod -aG docker ansadmin (do it in root account)

→ login in ansadmin & check group $ id

→ $ docker images (if have any images delete)

→ Pull the images from docker hub

$ docker pull yamkills/simple-devops-image

⇒ Create playbook To Push & Pull the docker images

→ go to ansible-server → cd /opt/docker (or) /opt/kks

↳ create hosts file $ vi hosts

↳ docker-host ip

→ create yaml file (to push image to docker hub & remove in ansible-s

$ vi create-simple-devops-image.yml (olp/kks)

- hosts: all (or) ansible-server

become: true

tasks:

- name: create docker image using war file

command: docker build -t simple-devops-image:latest .

args:

chdir: /opt/docker (or) (opt/kks)

- name: create tag to image

command: docker tag simple-devops-image yamkills/simple-devops-imag

- name: Push image on to dockerhub

command: docker push yamkills/simple-devops-image

- name: remove docker images from ansible server

command: docker rmi simple-devops-image:latest yamkills/simple-devops-image

ignore-errors: yes

Note: To check yaml run the playb and check in docker hub

$ ansible-playbook

⇒ create another yml file (to pull image from docker hub to docker host and deploy container)

$ vi create-simple-devops-project.yml

```
---
- hosts: all
  become: true

  tasks:
  - name: stop current running container
    command: docker stop simple-devops-container
    ignore_errors: yes

  - name: remove stopped container
    command: docker rm simple-devops-container
    ignore-errors: yes

  - name: remove docker image
    command: docker rmi yankils/simple-devops-image:latest
    ignore-errors: yes

  - name: pull docker image from dockerhub
    command: docker pull yankils/simple-devops-image:latest

  - name: create container using simple-devops-container image
    command: docker run -d --name simple-devops-container -p
             8080:8080 yankils/simple-devops-image
```

→ Save (to check run playbook.yml

$ ansible-playbook -i hosts create-simple-devops-proje

$ docker images $ docker ps -a

d) Jenkins Job to deploy on Docker container through Dockerhub

⇒ go to ansible server - $cd /opt/docker

$vi hosts

↳ local host

↳ docker-host ip

{ note: To limit

$ ansible-playbook -i hosts create-simple-devops-image.yml --limit localh

$ ansible-playbook -i hosts create-simple-devops-project.yml --limit dock-host ip

⇒ go to jenkins console → new item : Deploy_on_Docker_container_using_Ansible.

↳ copy from : Deploy-on-container_using_Ansible → ok

↳ Description: Deploy on docker container using ansible

↳ source code mgmt → git : paste helloworld url → branch : */master

↳ pom : pom.xml → Goals & options : clean install package

↳ Build Trigger → Poll scm : * * * * *

↳ Post-build Actions → Send build artifacts over SSH

SSH server name : ansible-server

Source files : webapp/target/*.war

Remove Prefix : webapp/target

Remote dive : //opt//docker

Exec cmd : ansible-playbook -i /opt/docker/hosts /opt/doc

ker/create-simple-devops-image.yml --limit local host;

ansible-playbook -i /opt/docker/hosts /opt/docker/create-simple-devops-

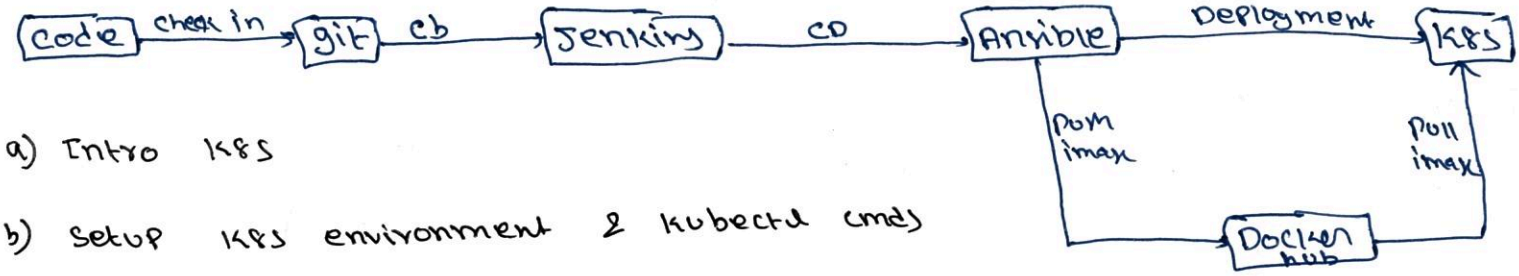
Project.yml --limit 'docker-host ip';

↳ Apply & Save → build now → open browser docker-host IP: 8080/webap

note: modify the code in github, automatically deploy code in container

→ If container get problems, manually have to do for this we use k8s - to maintai

Step 4: Deploy on K8S by using Ansible



a) Intro K8S

b) Setup K8S environment & kubectl cmds

c) writing deployment & services files

d) creating Ansible playbooky to deploy

e) Run a jenkins job

a) Intro: K8S is portable, extensible, open-source platform for managing containerized workloads & services, that facilite both declarative configuration & automation.

b) Setup

→ create & launch EC2 instance (ubuntu) → name: K8S-mgmt-server

1. Install AWSCLI

```
# curl https://s3.amazonaws.com/aws-cli/awscli-bundle.zip -o awsc
  bundle.zip
```

2. # apt install unzip Python (install Python & unzip)

```
# unzip awscli-bundle.zip
# ./awscli-bundle/install -i /usr/local/aws -b /usr/local/bin/aws
# aws --version
```

3. Install kubectl

```
# curl -LO https://storage.googleapis.com/kubernetes-release/release/$(cur
  -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bi
  n/linux/amd64/kubectl
```

Is it will download kubectl file

chmod +x ./kubectl

sudo mv ./kubectl /usr/local/bin/kubectl

4) Install kOPS

curl -LO https://github.com/kubernetes/kops/releases/download/$(curl -s https://api.github.com/repos/kubernetes/kops/releases/latest | grep tag_name | cut -d '"' -f 4)/kops-linux-amd64

chmod +x kops-linux-amd64

sudo mv kops-linux-amd64 /usr/local/bin/kops

5) Create IAM Role/usr with Route53, EC2, IAM & S3 full access

→ aws console → IAM → Roles → create role → select type of trusted entity : EC2 → next → search policies : EC2full access
: S3 full access
: Route53 full access
: IAM full access

→ next : Tags → name : k8s-role → Review → Role name : k8s-role

→ create role

→ attach this role to EC2 instance → go to instance & select k8s-mgmt-server → Actions → Instance settings → Attach IAM Role → select k8s-role → Apply

6) go to k8s mgmt-server

$ aws configure → Access key & Secret key : ignore
↳ region name : ap-southeast-1 (Singapore)

7) create Route53 hosted zone private/ Public

→ Route53 → DNS mgmt → create hosted zone → Domain name : valaxy.ne
Type : private/public → select VPC ID : Singapore → create

8) create env - S3

8) create S3 bucket

→ S3 → c.

→ go to k8-mgmt-server → $ aws s3 mb s3://demo.kps.valany.net
↓
if name not available chang name

→ go to S3 & select bucket (demo.kps.valany.net)

9) expor Environment variable

→ In k8-mgmt-server → $ enport KOPS_STATE_STORE = s3://demo.kps.valan

10) create SSH keys before creating cluster

→ go to k8-mgmt-server → # ssh-keygen (it create under .ssl
↳ cd .ssh → ls -l

11) create kps cluster definitions on S3 bucket

kops create cluster --cloud=aws --zones=ap-southeast-1a
--name=demo.kps-valany.net --dns-zone=valany.net --dns priva

↳ it will create seperate VPC, subnet, SG, RT & KeyPair, IAM r
EBS volume, Auto scaling group.

⇒ to see out cluster

$ kops get cluster

⇒ create cluster

$ kops update cluster --name demo.kps-valany.net --yes

↳ go to S3 bucket & check will be some files &
IAM Roles also will create → go to Route 53
addition Record sets will create & additions instanc
also launched. & also will create Launch
configuration. (Auto scaling)

$ kops validate cluster (to check)

$ ssh -i ~/.ssh/id_rsa admin@api.demo.kps.valany.net

↳ to connect kps cluster

$ kubectl get nodes (is the cmd to see out nodes)

c) create deployment & service using kubectl cmds:

→ open → kps-mgmt-server → login in kps master node admin a/c

$ ssh -i ~/.ssh/id_rsa admin@api.demo.kps.valany.net

$ sudo su - (in master node)

$ kubectl (to check it's working or not)

$ kubectl get nodes (list out the nodes)

$ kubectl get pods (list out the pods/container)

$ kubectl get deploy (list out deployments)

$ kubectl get service (list out services)

i) Deploy nginx pods on kps

⇒ deploy nginx container

$ kubectl run sample-nginx --image=nginx --replicas=2 --port=80

$ kubectl get deploy (to see deployments) ⇒ deploy/deployemen

$ kubectl get pods -o wide (to get pod details)

⇒ To access the pods from another systems (publicly access), expose the deployment as service

$ kubectl expose-deployment sample-nginx --port=80 --type=loa balancer.

$ kubectl get services -o wide → open browser masternodeIP:Port

$ kubectl ~~get~~ delete pod podname1

↳ To delete pods

note:- if delete pods, it will automatically recreate pods.

To delete deployment # kubectl delete deployment sample-nginx

c) create deploy & service using YAML files

→ In k8s-mgmt-server - Create 2 yaml files

1. valaxy-deploy.yml 2. valaxy-service.yml

1. valaxy-deploy.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: valaxy-deployment
  space:
  selector:
    matchLabels:
      app: valaxy-devops-project
  replicas: 2
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
  template:
    metadata:
      labels:
        app: valaxy-devops-project
    spec:
      containers:
      - name: valaxy-devops-project
        image: yankils/simple-devops-demo
        imagePullPolicy: Always  →[imagePullPolicy: Always]
        ports:
        - containerPort: 8080
```

2. valary-service.yml

```
apiVersion: v1
kind: Service
metadata:
  name: valary-service
  labels:
    app: valary-devops-Project
spec:
  selector:
    app: valary-devops-Project
  type: LoadBalancer
  ports:
    - Port: 8080
      targetport: 8080
      nodeport: 31200
```

eg: To execute these yaml in master node

kubectl apply -f valary-deploy.yml (it will deploy)

kubectl get deployments

kubectl apply -f valary-service.yml (it will run the service)

kubectl get services

kubectl get pods -o wide

=> open browser and type master node ip: Port#/webapp

e) Integrate K8S with Ansible

1. Login to ansible server and copy publickey onto k8s clus master account.

→ Login in ansible server → sudo su - ansadmin → cd .ssh → ls
→ copy the id_rsa.pub file.
→ cat id_rsa.pub → copy the data

→ go to k8s master node → cd ~/.ssh → ls → authorized_keys
vi authorized_keys
↳ paste the id_rsa.pub data
(or)
cat >> authorized_keys
↳ paste the id_rsa.pub data (added under root account)

To check :- go ansible server → $ ssh -i ~/.ssh/id_rsa root@ k8s mas node public IP, now it will connect to master node fro ansible server. → exit

2. Update hosts file with new group called k8s and add k8s master in that.

→ open ansible-server → # cd /opt → ls → #sudo mkdir k8s → #cd k8

sudo vi hosts
localhost {ansible server}
13.233.7464 {master pvt IP}

{ansible-server}
localhost

[K8S]
13.233.24.64 (master public i

3. create ansible playbooks to create deployment & service

i) k8s_valaxy_deployyml (in ansibleserver)

```
---
- name: create pods using deployment
  hosts : k8s
  !
```

```
become: true
user: root

tasks:
- name: create a deployment
  command: kubectl apply -f valany-deploy.yml

# - name: update deployment with new pods if image updated in
      docker hub
  command: kubectl rollout restart deployment.v1.apps/vava
      - deployment #
```

2) k8s-valany-service.yml

```
---
- name: create service for deployment
  hosts: k8s
  # become: true
  user: ubuntu

  tasks:
  - name: create a service
    command: kubectl apply -f valany-service.yml
```

en: To execute to check the .yaml files

→ before that delete all the deployments & services in k8s-mgmt-ser

1# kubectl delete deployment valany-deployment

kubectl delete service valany-service

→ go to ansible-server → sudo -su ansadmin → cd /opt/k8s/

ansible-playbook -i hosts k8s-valany-deployment.yml

↳ go to k8s-mgmt-server and check the deployments

ansible-playbook -i hosts k8s-valany-service.yml

↳ goto k8s-mgmt-server and check the services

f) Jenkins CD Job to Deploy K8S

→ Login in Jenkins console → new item: deploy_on_K8S_CD → Free style project → ok → Description: deploy on K8S CD

↳ Soc: none

↳ Post-build Actions → Send build artifacts over SSH

SSH Server Name: select ansible-server

Exec cmd: ansible-playbook -i /opt/K8S/hosts /opt/K8S/K8S-valany-deployment.yml;

ansible-playbook -i /opt/K8S/hosts /opt/K8S/K8S-my-service.yml;

↳ Apply - ok → Build now (before that delete all the deployments & services in K8S-mgmt-server)

g) Jenkins CI Job to create an Docker image

→ open Jenkins → new item → deploy-on-K8S-CI → copy from: Deploy_on_Docker_container_using_Ansible_playbooks → ok → Desc: deploy on K8S CI (create docker image using war file)

↳ SCM: Git → repository url: github url → branche: */master

↳ Build Trigger: Poll Scm: * * * * *

↳ Build: Pom: pom.xml

Goals & options: clean install package

↳ Post-build-Actions → send build artifacts over SSH

Name: ansible-server

Src files: webapp/target/*.war Remove prefix: webapp/target

Remote directories: //opt//docker (or) //opt//K8S

Exec cmd: ansible-playbook -i /opt/docker/hosts /opt/docker/create-image-devops-image.yml --limit localhost;

↳ Apply - Save

Note: mv the /opt/docker/create-simple-devops-img.yml to /opt/k8s/

& hosts also, mover change for [ansible-server]
localhost

[k8s]
master node ip

& # chmod -R
# echo chown -R ansadmin:ansadmin /opt/k8s

h) Integrating CICD Jobs to deploy on k8s

→ open Jenkins console → select deploy_on_k8s_CI project → configure
→ post-build Action → Add post-build action: Build other projects

Projects to build: select deploy_on_k8s_CD

⊙ Trigger only if build is stable

↳ Apply → save

now change the code and check, in k8s-pods it will maintain one image only, if there is a version change then only it will override.

i) Automate deployment on k8s with CICD Job

if there is change it will create new image → Push that image in to docker hub and at the same time it will initiate deployment, deployment can able to pull it the image in docker hub and terminate to old pods and create new pods.

1. delete all the deployments & services in k8s-cluster
2. check the kubectl version, if want to use Rollout, kubectl version should be more than v.1.15 (min)

kubectl version

⇒ To update kubectl version

→ In k8-mgmt-server

curl -LO https://github.com/kubernetes/kops/release/download/$(curl -s https://api.github.com/repos/kubernetes/kops/release/latest | grep tag_name | cut -d'"' -f 4)/kops-linux-amd64

chmod +x kops-linux-amd64

sudo mv kops-linux-amd64 /usr/local/bin/kops

kubectl version

⇒ go valany-deploy.yml and add the below portion.

under replicas: 2

strategy:

type: RollingUpdate

rollingUpdate:

maxSurge: 1

maxUnavailable: 1 #

2 edit k8s-valany-deploymen.yml file in Ansible-Server

add under tasks:

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

command: kubectl rollout restart deployment.v1.apps/valan-deployment

now change the code & check

"Deploying on a k8s using ansible"