CI / CD pipeline.

- continuous Integration (CI).
- continuous Delivery (CD).
- continuous Deployment (CD).

In CI/CD we need to build the source code and deploy it in to the target environment.

Creating a Jenkins Job

NewItem

↳ HelloWorld Job

↳ freestyle project.

Build

↳ Execute shell.

Command.

```
echo "Hello World".
uptime.
```

↳ execute uptime of system

Apply and Save.

To execute a job

BuildNow

Integrate Git with Jenkins:

To install plugins

↓

manage ~~plugins~~ Jenkins

↓

manage plugins

↓

available

Manage Jenkins → Global Tool configuration

Integrate maven with Jenkins:

manage jenkins

↓

manage plugins

↓

available

Integrate Tomcat with Jenkins:

deploy to container

↓

This plugin allows you to deploy a war to a container after a Successful build.

Post Build actions:

Deploy War/ear to a container

Deploy artifacts on a tomcat server:

Configure → source code management

Build Triggers

Poll SCM:

Schedule

↓

We need to schedule a Job

If there are any change Job will get executed

if there is no Job there is no build.

$ git add .

$ git commit . -m "password field in new link 'index.jsp'

$ git push origin master

Integrating Docker in CI/CD pipeline:-

Ec2 Instance

management console

AWS services:

EC2

↓

Launch instance

↓

Amazon Linux AMI (HVM) : SSD

↓

t2 micro

↓

add tag → key: Name, value = Docker-Host

↓

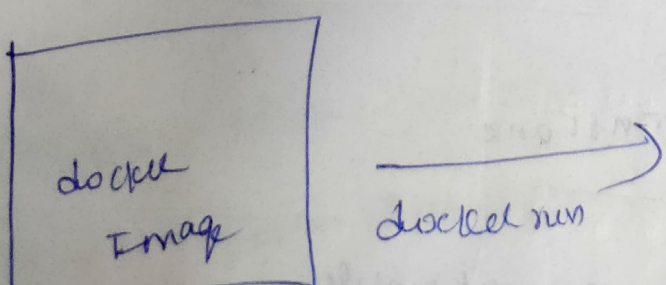
security group → select an existing security group

↓

Launch instance

How to create a docker container:

To create a docker container we need a docker image with the help of docker image we can run a command called docker run with additional specifications we can create docker container



we can get docker image in two ways:

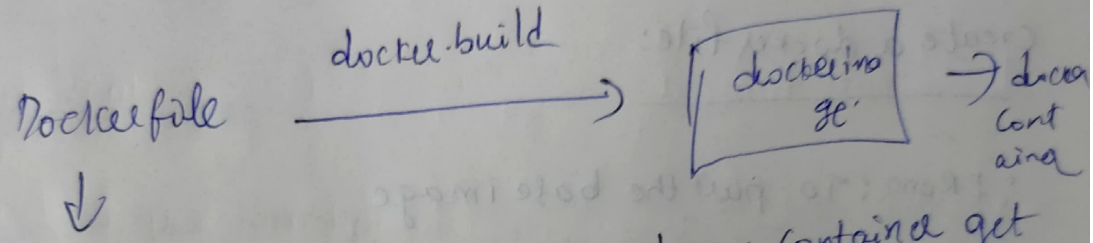
1. docker hub → public repository contains docker images

   we can use command called

   docker pull to take the image on to

   local system.

2. ~~De~~ creating our own docker file.



If contain instructions what docker container get

to create docker image using dockerfile we need to

use docker build command.

hub.docker.com

$ docker pull tomcat

$ docker images

$ docker run -d --name tomcat-container -p

Port no

↓

detached mode

8081 : 8080 tomcat

Creating a docker container

docker ps -a

~ docker ps -a

~ docker exec -it tomcat-container/bin/bash

↓

log in to docker container with help of bin bash

docker ps show running containers

Create a docker file:-

- FROM : To pull the base image
- RUN : To execute commands
- CMD : To provide defaults for an existing container
- ENTRY POINT : To configure a container that will run as an executable. same as CMD.

CMD commands can be overriden

ENTRY POINT commands can not be overridden

- WORKDIR : To sets the working directory.

↳ like as cd command

to switch directory

- COPY :- To copy a directory from your local machine

ADD: To copy files and folders from your local machine to docker containers.

Expose: Informs Docker that container listens on specified network ports at runtime

ENV: To set environment variables.

Creating a Dockerfile.

FROM centos:latest

RUN yum install java -y

RUN mkdir /opt/tomcat

WORKDIR /opt/tomcat

ADD URL . → current work Directory

RUN tar -xvzf apache-tomcat-9.0.54.tar.gz
→ To extract file

RUN mv apache-tomcat-9.0.54/* /opt/tomcat

Expose 8080

CMD ["/opt/tomcat/catalina.sh", "run"]
→ script
→ to start tomcat

~ docker build -t mytomcat .
↓ dockerfile location

~ docker · images

~ docker · run -d --name mytomcat - serve · -p 8083:8 (port)

(to create a container) (mytomcat → image)

~ docker · stop mytomcat-server ·

Create a customized Dockerfile for tomcat :-

docker pull tomcat ·

~ rm Dockerfile

~ vi · Dockerfile ·

From · tomcat : · latest · ·

× RUN · cp -R webapps · dist webapps ·

(located in Catalina home directory)

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

~ · docker build -t demotomcat · ·

(-t → tag ·) (demotomcat → Imagename) (. → represent in current directory we have docker file ·)

docker · ps -a ·

~

Integrate docker with jenkins:-

- create a dockeradmin user
- Install "Publish over SSH" plugin.
- Add Dockerhost to jenkins "Configure Systems".

cat /etc/passwd.

↓

list of users.

cat /etc/group

↓

to check groups

user add docker admin

↓

creating a user

Passwd dockeradmin

usermod -aG docker dockeradmin

↓

additional group

adding user to docker

to enable explicitly.

vi /etc/ssh/sshd_config

/ Password

Ec2 instance does not allow Password based authentication we should explicitly enable it.

service sshd reload.

To configure jenkins with docker

Mange Jenkins → configure system

↓

SSH servers

Name: dockerhost

Hostname

172.31.21.256 → privet Ip

Username

dockeradmin

Use password based authentication

Password

↓ Test configuration

$ ssh-keygen → to generate a key

Jenkins job to build and copy artifacts on to dockerhost

Source files

webapp/target/*.war

Remove prefix

webapp/target

Remote directory

/dockeradmin

Exec command.

cd /opt/ docker ;

docker build . -t regapp:v1 . ;

docker run -d --name registerapp -p 8057:8080 regapp:v1

~ docker Container prune

↓ to delete stopped container

~ docker image prune

↓ to delete images

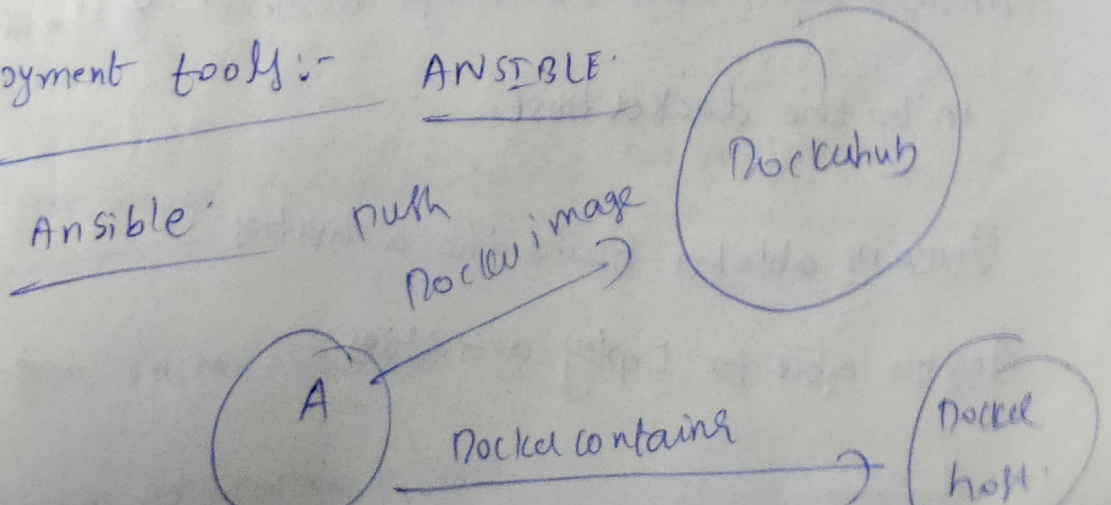
Before creating a container stop and delete the container

docker stop registerapp;

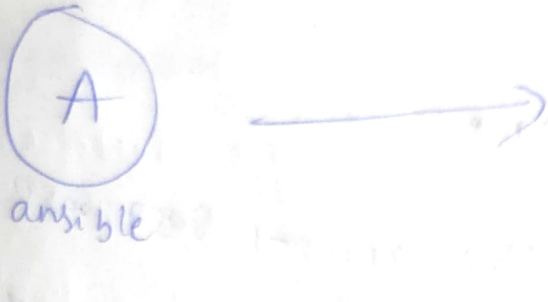
docker rm registerapp;

Deployment tools :- ANSIBLE

Ansible



Integrate Docker with Ansible:

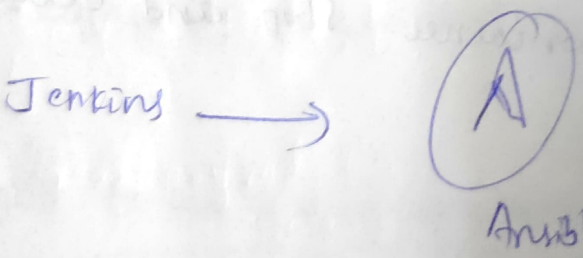


- on Docker Host
- Create ansadmin
- Add ansadmin to sudoers files.

- on Ansible Node.
- Add to hosts file
- copy ssh keys

managed node which are managed by ansible

Integrate ansible with Jenkins:-



Jenkins able to copy artifacts into Ansible.

Ansible will create an image and deploy a container in to the docker Host.

Jenkins able to build the activities.

Ansible able to Deploy activities.

manage Jenkins

↓

Configure system.

↓

Publish over SSH

Add

SSH Servers

Name

ansible-server

HostName.

172.31.30.109.

UserName

ansadmin

Password

Use password authentication

Apply and Save.

New item → Enter an item name.

Build an image create Container on Ansible:-

Sudo yum install docker -y

Executing as root user.

$ docker run -t --name regapp

↓

terminal

regapp:v1

Ansible playbook to create image and container :-

cd /opt

$ ansible-playbook -i

$ ansible all -a uptime

↓

Communicate with docker system not ansi system.

$ ssh-copy-id 172.31.30.109

$ ssh-copy-id localhost

$ vi reaapp.yml

writing ansible playbook

Start with ---

- hosts: ansible

hosts represents on which system we are executing

become: true → to become root

tasks:

- name: create docker image

~~docker image~~

command : docker build -t regapp:latest . -f

↓

file

args :

↓

Some other location we are executing

chdir : /opt/docker

- name : create tag to push image onto dockerhub

$ ansible-playbook regapp.yml --check

↓

going to run successfully or not

$ docker images

Copy image to dockerhub :-

$ docker login

$ docker push regapp - not possible

$ docker images

$ docker tag image-Id username/regapp:latest

$ docker push username/regapp

↓

Pushing image in to dockerhub or committing

in ansible playbook add two commands

- name: Create tag to push image on to dockerhub

  Command: docker tag regapp:latest valaxy/regapp:lat

- name: push docker image

  Command: docker push username/regapp:latest

$ ansible-playbook regapp.yml --check

$ cat /etc/ansible/hosts

to copy artifacts in ansible-playbook

↓

Configure

↓

Execute command.

ansible-playbook regapp.yml

Configure

↓

poll cm

* * * * *

means execute for every minute

Create container on dockerhost:

- hosts: dockerhost

tasks:

- name: create container

command: docker run -d --name regapp -server

-p 8082:8080 username/regapp

:latest

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

$ docker ps -a

- Remove existing container
- Remove existing image
- Create a new container

If image exists in local system we can't push in to docker

- - hosts: dockerhost

tasks:

- name: stop existing container

command: docker stop regapp-server

- name: remove the container

command: docker rm regapp-server

- name: remove image

command: username/regapp:latest

- name : create container

Command : docker run -d --name regapp-server -p 8082:8080 username/regapp:latest

$ ansible-playbook dpby-regapp.yml

Kubernetes on AWS:-

Kubernetes basic commands

Deploying nginx container

Whenever we create a deployment in backend it is going to initiate a replicaset

→ This replicaset going to create a pod

→ To access this pody we need to create a service by using expose command.

It will create service in the background. This service type is a load balancer

↓ Kubectl get all

↓

getting all resources in kubernetes cluster

~ kubectl create deployment demo-nginx --image=nginx
--port=80 --replicas=2

↓

Using this command we are pulling the image from docker hub and creating a deployment.
and in the backend it will creat replicaset.

~ kubectl get deployment or ~ kubectl get deploy

↓

to check out deployments

~ kubectl get replicaset

↓

to get replicaset.

~ kubectl get pod.

↓

display nods available on cluster

~ kubectl expose deployment demo-ngix --port=80 --type=

↓ ↓

deployment name. LoadBalancer

exposing service LoadBalancer on port 80.

Creating Manifest file:

kubectl delete deployment demo-nginx

~kubectl delete service/demo-nginx

(deleting load Balancer

~vi pod.yml

```
apiVersion: v1
kind: Pod
metadata:
    name: demo-pod
    labels:
      app: demo-app
```

} used to give name of pod

```
Spec:
  containers:
  - name: demo-nginx
    image: nginx
    ports:
    -name: demo-nginx
     containerPort: 8080
```

To expose pod we need to

vi service.yml

```
apiversion : v1
kind : Service
metadata:
    name : demo-service
spec:
    ports:
    -name : nginx-port
      port : 80
      targetPort : 80
    selector :
        app : demo-app
    type : LoadBalancer
```

Service manifest file.

To create pod with manifest file we need to use

~ kubectl apply -f pod.yml → creates pod.

~ kubectl get all

To create a service.

↓

~ kubectl apply -f service.yml

using labels and selector

~ kubectl apply -f pod.yml

↓ It will update the pod.

~kubectl get pod -o wide

↓

It will give pod-name, Node-IP and on wh

Server it ~~will~~ is running