**Agenda:**

Setup an application with Docker Stack & Jenkins

When developer upload the latest code on github: (Jenkins Pipeline)

* Pull the code
* Create image
* Push to docker hub
* Deploy the application on multiple containers via Docker stack

**Prerequisites:-**

* Create 3 servers, 1 master (4 GB RAM), 2 slaves (1 GB RAM)
* Configure host accordingly
* Install docker on all servers
* Setup docker swarm and copy token on slave servers
* Give full permission to docker.sock file
* Install Git, Java & Jenkins on Master Server

**Project Steps:-**

* Pull the code on master server
* Create Dockerfile for Apache service
* Create Image from Dockerfile
* Create docker-compose file for multiple containers setup
* Configure the Jenkins server
* Setup pipelines for application deploy

**==============================================================**

**Configure host:**

hostnamectl set-hostname master

hostnamectl set-hostname slave

hostnamectl set-hostname slave

**Install docker on servers:**

yum install docker -y

systemctl start docker

systemctl status docker

chkconfig docker on

**Setup docker swarm:**

docker swarm init (#copy the token and paste on slave servers) docker node ls (#check the join nodes)

docker swarm leave (#remove the join nodes)

docker swarm join-token manager (#check the token)

docker node rm node\_id (#remove working node completely)

**Give full permission:**

chmod 777 /var/run/docker.sock

systemctl daemon-reload

systemctl restart docker.service

**Install Git, Java & Jenkins on Master Server:**

yum install git java-1.8.0-openjdk maven -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>

amazon-linux-extras install java-openjdk11 -y

yum install jenkins -y

update-alternatives --config java

systemctl start jenkins.service

systemctl status jenkins.service

**Clone the code on master server:**

git clone <https://github.com/RAHAMSHAIK007/dockernewproject.git>

**Create Dockerfile for Apache:**

FROM ubuntu

RUN apt-get update -y

RUN apt-get install apache2 -y

COPY index.html /var/www/html/

CMD ["/usr/sbin/apachectl", "-D", "FOREGROUND"]

**Create Image from Dockerfile: (Manual Effort/Not Required)**

docker build -t paytrain:v1 .

**Create docker-compose file for multiple containers setup: (Manual Effort)**

version: "3.8"

services:

movies:

image: mykt/paymovies:latest

ports:

- 81:80

deploy:

replicas: 3

train:

image: mykt/paytrain:latest

ports:

- 82:80

deploy:

replicas: 3

dth:

image: mykt/paydth:latest

ports:

- 83:80

deploy:

replicas: 3

recharge:

image: mykt/payrecharge:latest

ports:

- 84:80

deploy:

replicas: 3

**Configure the Jenkins server:**

Create Pipeline: (DockerPipeline)

pipeline {

agent any

stages {

stage ('CheckOut') {

steps {

git 'https://github.com/myktiwari/dockernewproject.git'

}

}

stage ('Build') {

steps {

sh 'docker build -t $img .' // docker build -t paytrain:v1

}

}

stage ('Tag Image') {

steps {

sh 'docker tag $img $repo' // docker tag paytrain:v1 mykt/paytrain

}

}

stage ('Push To Dockerhub') {

steps {

sh 'docker login -u mykt -p $password' // docker login -u mykt -p

sh 'docker push $repo' // docker push mykt/paytrain

}

}

}

}

**Choose the Parameters:**

Check 🡪 This project is parameterized

Add Parameter 🡪 Choice Parameter

Name: img

Choice: movies:v1

train:v1

recharge:v1

dth:v1

Name: repo

Choice: mykt/paymovies

mykt/paytrain

mykt/payrecharge

mykt/paydth

**Add Global Environment Variables:**

Manage Jenkins 🡪 System 🡪 Global properties 🡪 Environment variables

List of variables 🡪 Name: password

Value: <dockerhub\_password>

**Create Pipeline: (DockerPipeline)**

pipeline {

agent any

stages {

stage('CheckOut'){

steps {

git 'https://github.com/myktiwari/dockernewproject.git'

}

}

stage('Deploy') {

steps {

sh 'docker stack deploy -c docker-compose.yml paytm'

}

}

}

}

Open the browser and check all the master & workers node with port number

**Check running services on Worker Node:**

docker service ls

docker node ls