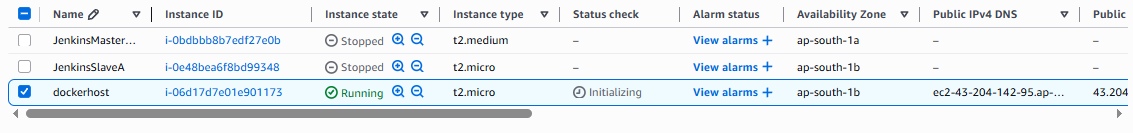
Container Operations

>> Download IMAGE

>> Create DOCKER

>> Docker commands

Dockerhost ……server



* Docker ps ……list the running containers
* Docker ps –a ……list all containers
* docker stats 81a57bce41e6 ……CPU, memory & storage, network performance of container
* Docker restart 81a57bce41e6 ……restarting the container
* Docker pause 81a57bce41e6 ……pause the container
* Docker unpause 81a57bce41e6 ……unpausing the container
* Docker rename 81a57bce41e6 web-server-01 ……rename the docker name
* Docker stop 81a57bce41e6 ……stop your container
* Docker start 81a57bce41e6 ……start your container
* Docker rm 81a57bce41e6 ……delete your container

>> Note: You cannot delete a container which is in start state

>> Also validate container doesn’t exist in /var/lib/docker/containers

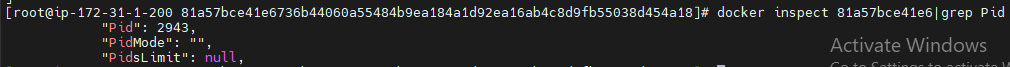
* Docker rm –f 81a57bce41e6 ……force delete a running container

>> Observe the directory before creation of container

* Docker run –itd ubuntu bash ……deploy a container
* Docker exec –it 81a57bce41e6 bash ……login to the container
* Ps –ef ……list process running in container
* Exit ……exit from the container

>> Observe the directory after creation of container

* Docker images ……listing the docker images
* Docker inspect 81a57bce41e6 ……fetch info about container
* Cat /var/lib/docker/containers/81a57bce41e6736b44060a55484b9ea184a1d92ea16ab4c8d9fb55038d454a18/config.v2.json ……inspect command output comes from this
* Ip a ……validate the ip address
* Docker inspect 81a57bce41e6; grep Pid;kill 2943



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Dockerfile formatting

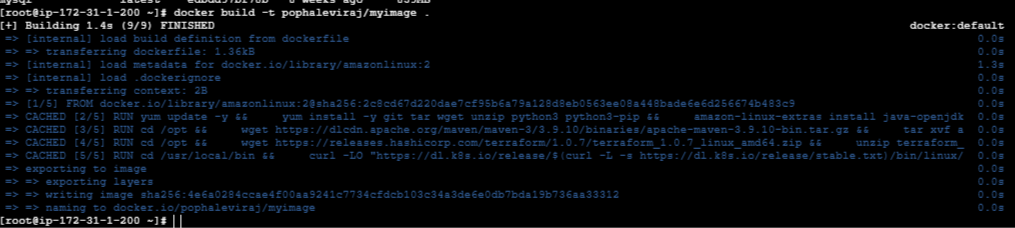
Goto Dockerhost >> /usr/bin/dockerfile >> Install java21, maven, terraform, kubectl >> point A

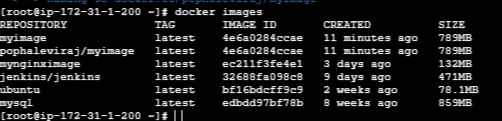
Create IMAGE from Dockerfile docker build -t <imagename> . ……displays error



Create image as dockerhub repo name alongwith <reponame>/<imagename>

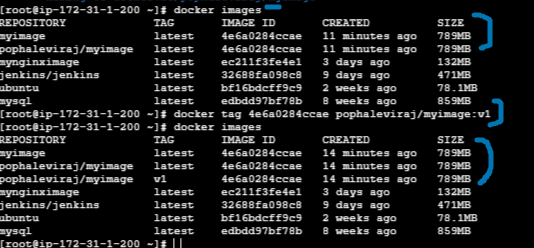
* Docker build -t <reponame>/<imagename> . ……eligible for hub upload



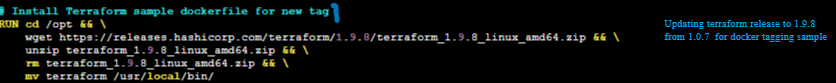


Create tagged version of image from earlier image which looks like v1 for versioning in dockerhub

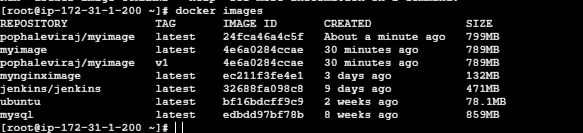
* Docker tag <imageid> <repositoryname>:v1
* Docker images ……displays two images having only tagname separate



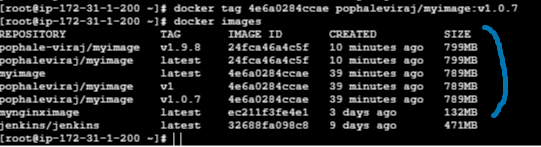
* Another update to dockerfile as per abv for terraform alongwith wget

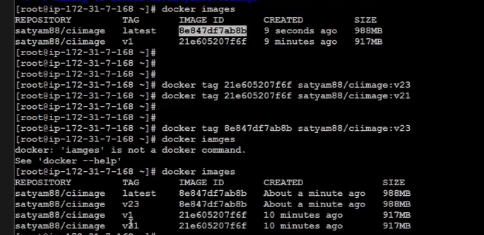






* Create tagged images as v1.9.8, v1.0.7 ……creates separate images total 4 in all

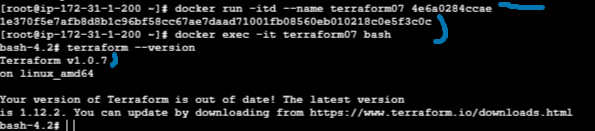




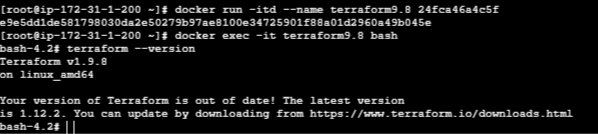
* Create CONTAINER using v1.0.7, v1.9.8 ……2 docker created

 ……example of docker creation

* Docker exec -it <containername\_07> bash
* Java –version
* Exit

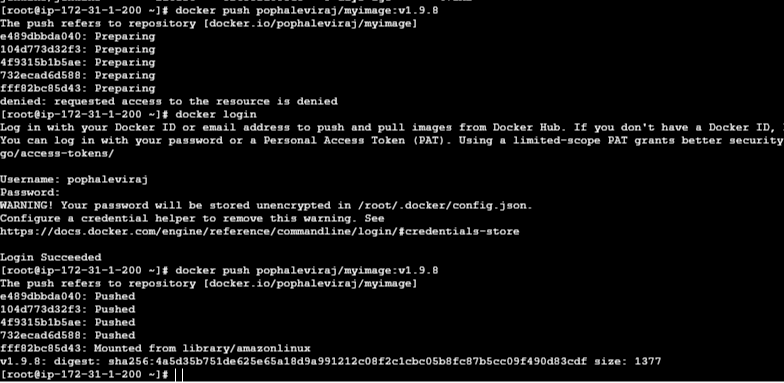


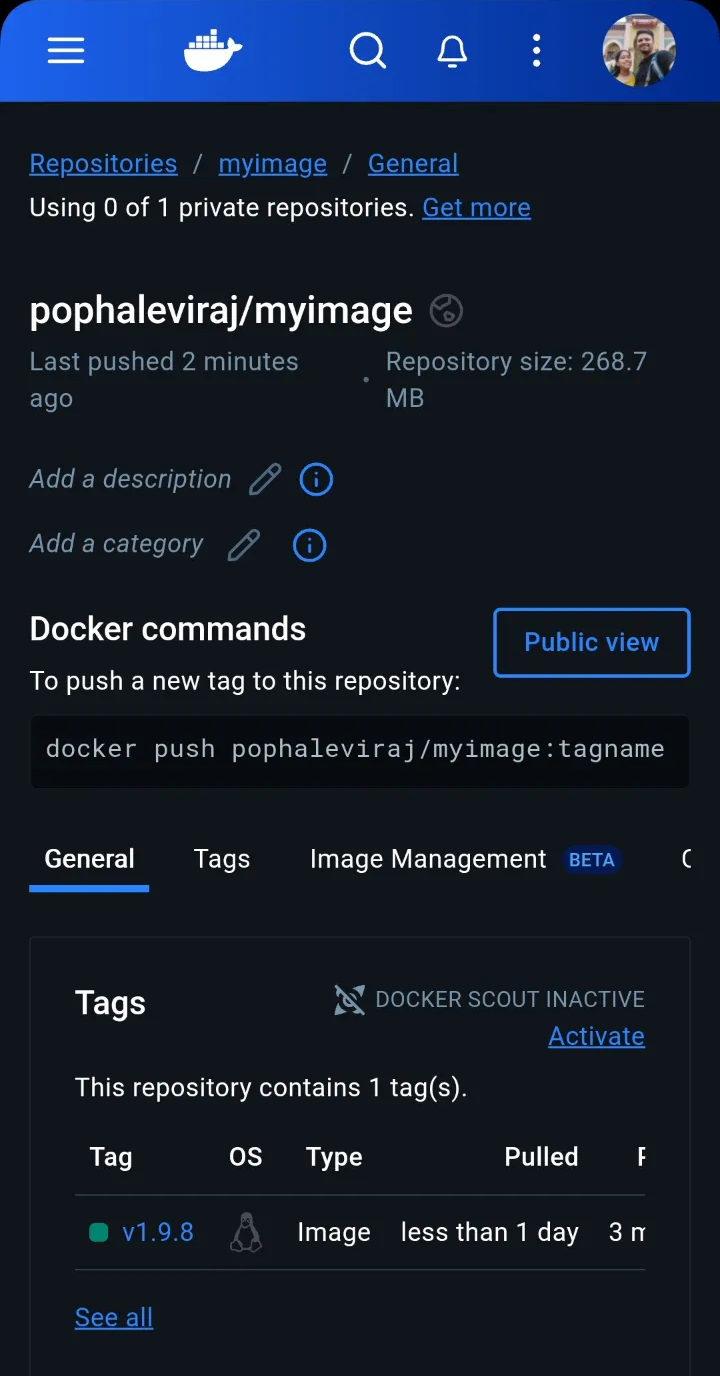
* Docker exec -it <containername\_9.8> bash
* Java –version
* Exit



Goto Dockerhub >> get docker push commands from repo

* Docker push <reponame>/<imagename>:<versiontag> ……displays error
* Docker login
* Docker push <reponame>/<imagename>:<versiontag>





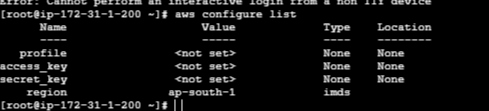
Goto AWS ECR >>

>> Create private repository=myimage >> Immutability=No >> Create >> View push commands



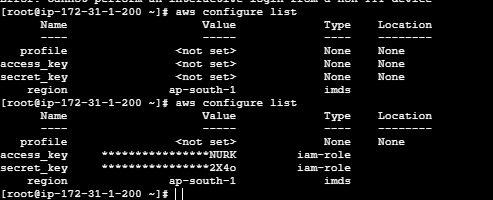
Try using 1st push command on dockerhost and login ……displays error that role missing



 ……no IAM role

Goto IAM >> Create Role >> AWS service >> use case=EC2 >> Next >> AdministratorAccess >> Next >> Rolename=dockerimage-upload >> Create Role

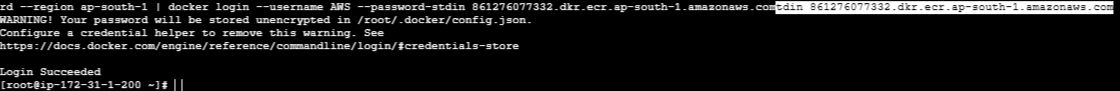
Goto EC2 >> select EC2 >> Modify IAM role >> Attach role



**Bookmark EC2 & IAM**

>> Goto ECR and take “View push commands”

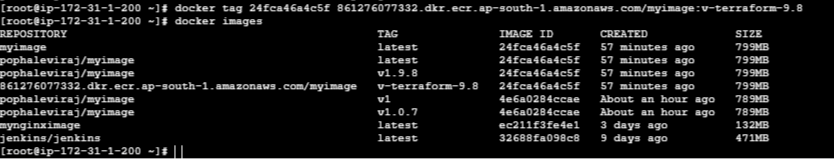
* aws ecr get-login-password --region ap-south-1 | docker login --username AWS --password-stdin [861276077332.dkr.ecr.ap-south-1.amazonaws.com](http://861276077332.dkr.ecr.ap-south-1.amazonaws.com)

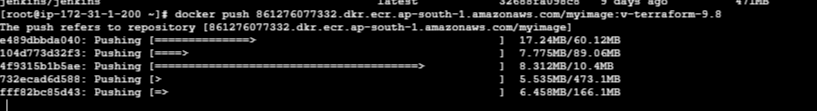


* docker build -t myimage .

>> docker tag myimage:latest [861276077332.dkr.ecr.ap-south-1.amazonaws.com/myimage:latest](http://861276077332.dkr.ecr.ap-south-1.amazonaws.com/myimage:latest) ……take repo name from here and use the repo for tagging to aws ecr

* docker tag 24fca46a4c5f [861276077332.dkr.ecr.ap-south-1.amazonaws.com/myimage:v-terraform-9.8](http://861276077332.dkr.ecr.ap-south-1.amazonaws.com/myimage:v-terraform-9.8)



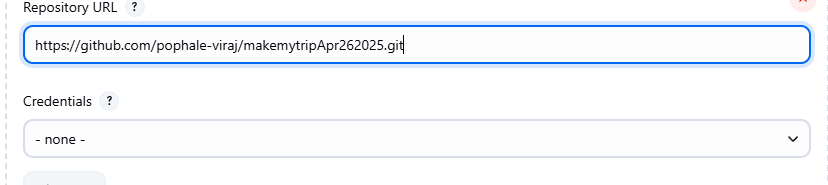




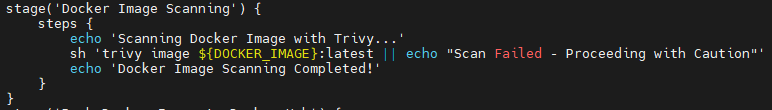
* Aws configure list ……displays type appropriately as per iam\_role
* Docker push commands from AWS
* Follow and verify the image in AWS
* Terminate latest EC2 work done & move for automation

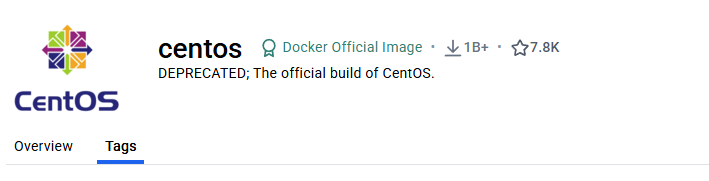
AUTOMATION of dockerfile steps(makemytrip-pipeline)

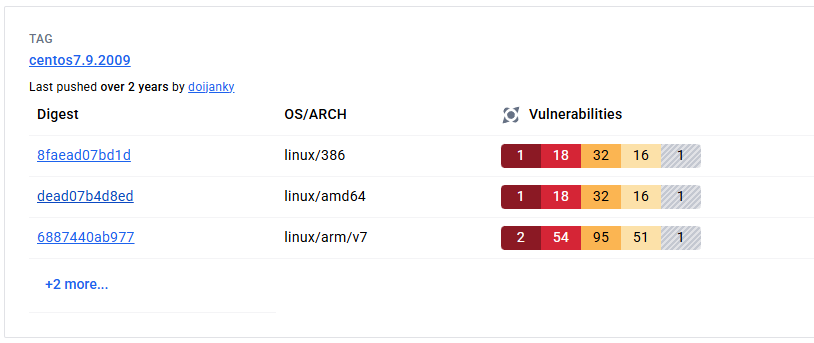
Create makemytrip-pipeline – public below show none credentials



Goto Jenkins >> update public-ip >> webhook update

 ……scans the code for vulnerabilities. AWS ECR has its own scanning capabilities at AWS level of the created image (Goto ECR >> images >> select image >> scan >> summary). SRE team takes responsibility of doing vulnerability scan. In dockerfile we will use trivy tool to scan the dockerfile.



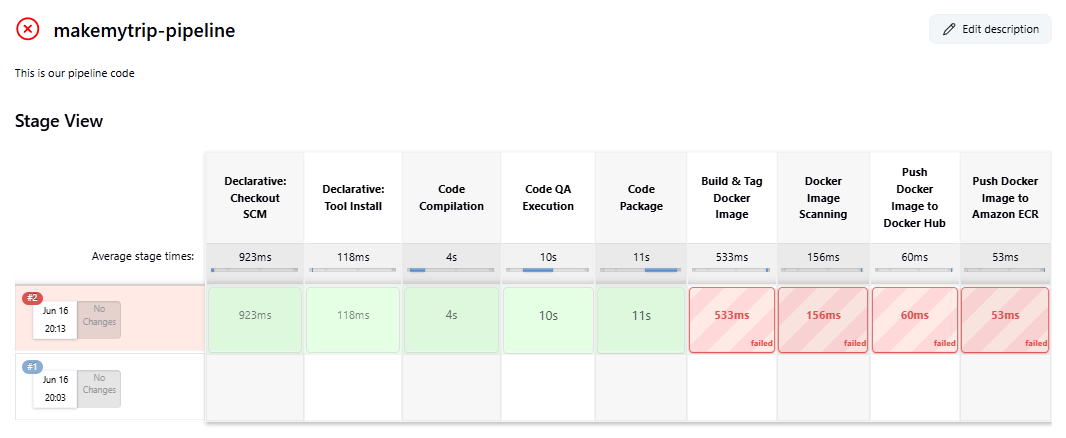


IntelliJ >>

* Git pull origin dev
* Git branch
* Edit Jenkinsfile >> point B
* Edit Dockerfile >> point C

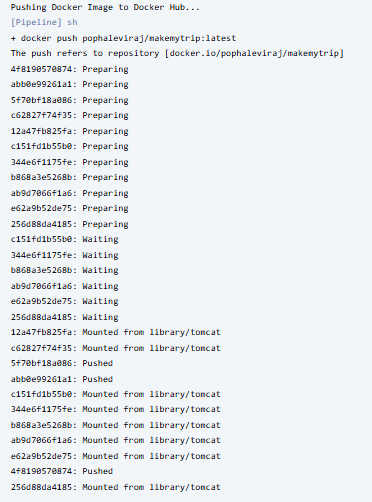
Jenkins >> update the nodes and delete the slave as agent used in Jenkinsfile is ANY

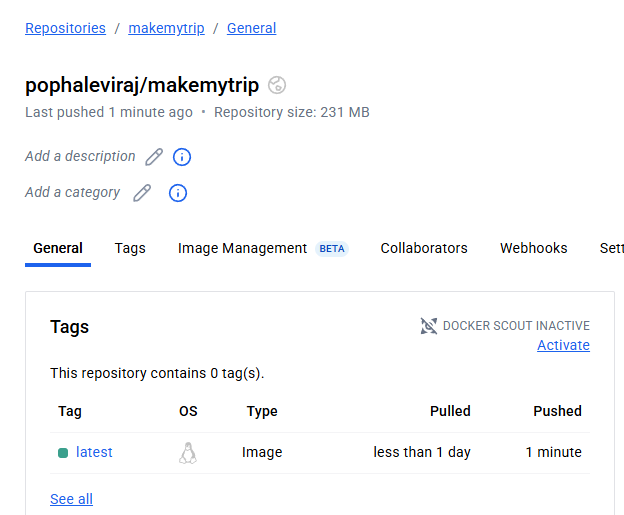
Trigger our code ……Trigger fails



Check error

* yum install -y docker; service doceker start; usermod -aG docker ec2-user; chkconfig docker on ……install docker on master
* Update the jenkins to use “secret text” credentials and add dockerhub credentials as ID=dockerhubCred and Password=dhV!raj0412
* Chmod 777 /var/run/docker.sock

>> Expected dockhub push output 



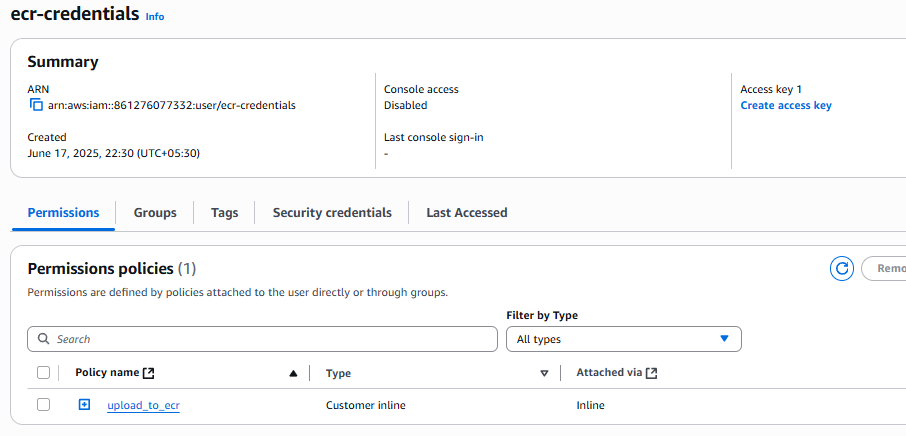
>> Add cleanup code to delete jenkins local images from jenkins server as images are pushed to remote

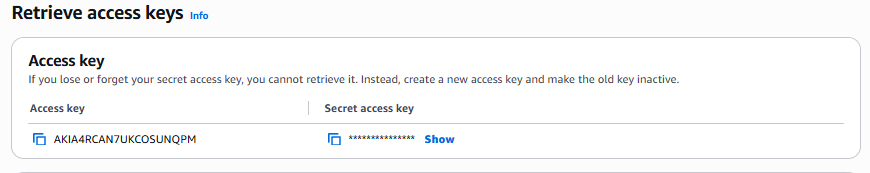


>> To resolve ECR login error:

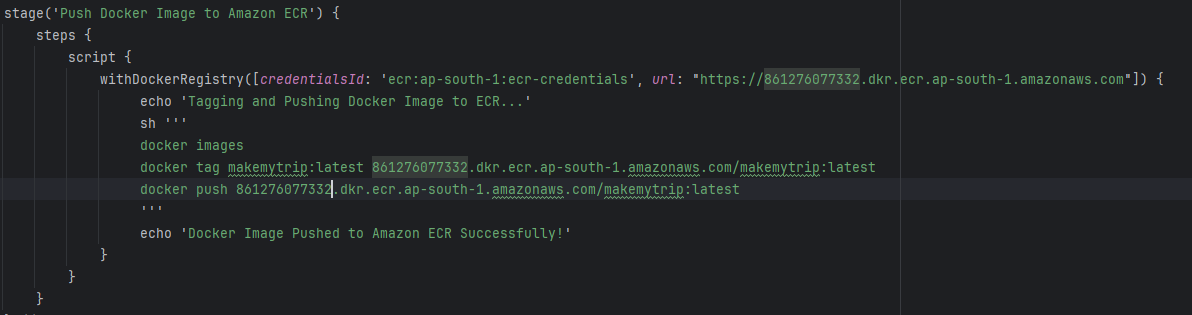
Goto AWS >> create IAM user=ecr-credentials >> Next >> Finish >> Select user >> Edit permissions >> Specify permissions >> JSON policy editor >> point D >> Next >> Policy name=upload\_to\_ecr

Select ecr-credentials >> Security credentials >> Create access keys >> Application running on an AWS compute service >> Next

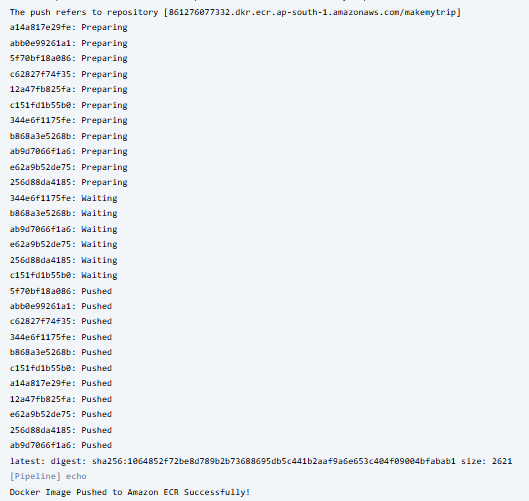
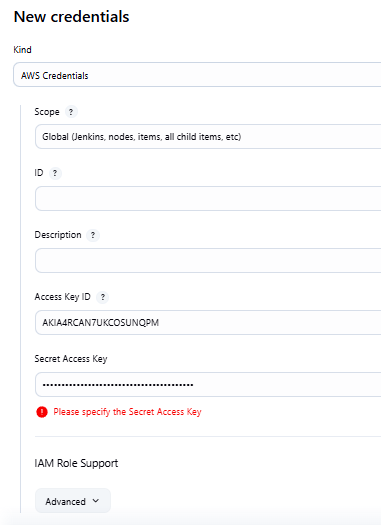


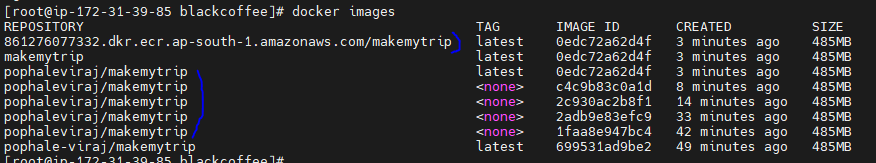


Take the account number and uri of region and update jenkinsfile



Goto Jenkins >> Manage jenkins >> Credentials >> Kind=aws credentials >> ID=ecr-credentials >> Access key ID=AKIA4RCAN7UKCOSUNQPM >> update secret access key





>> Cleanup will look for missing plugins, install them and always have a habit to package code, git push to dev, and build jenkins job now

A:dockerfileused in project:

===============================================================

# Use the Amazon Linux base image

FROM amazonlinux:2

# Update the system and install necessary packages

RUN yum update -y && \

yum install -y git tar wget unzip python3 python3-pip && \

amazon-linux-extras install java-openjdk11 -y && \

yum clean all

# Set up Maven

RUN cd /opt && \

wget https://dlcdn.apache.org/maven/maven-3/3.9.10/binaries/apache-maven-3.9.10-bin.tar.gz && \

tar xvf apache-maven-3.9.10-bin.tar.gz && \

rm apache-maven-3.9.10-bin.tar.gz && \

echo "export M2\_HOME=/opt/apache-maven-3.9.10" >> /root/.bashrc && \

echo "export M2=\$M2\_HOME/bin" >> /root/.bashrc && \

echo "export PATH=\$M2:\$PATH" >> /root/.bashrc

# Install Terraform

RUN cd /opt && \

wget https://releases.hashicorp.com/terraform/1.0.7/terraform\_1.0.7\_linux\_amd64.zip && \

unzip terraform\_1.0.7\_linux\_amd64.zip && \

rm terraform\_1.0.7\_linux\_amd64.zip && \

mv terraform /usr/local/bin/

# Install kubectl

RUN cd /usr/local/bin && \

curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl" && \

chmod +x kubectl

# Set environment variables for the current shell session

ENV M2\_HOME=/opt/apache-maven-3.9.4

ENV M2=$M2\_HOME/bin

ENV PATH=$M2:/usr/local/bin:$PATH

==========================================================

B: Jenkinsfileused in the IntelliJ for automation

pipeline {

agent any

options {

buildDiscarder(logRotator(numToKeepStr: '3', artifactNumToKeepStr: '3'))

}

tools {

maven 'mvn\_3.9.10' // Make sure this Maven tool is defined in Jenkins global tools

}

stages {

stage('Code Compilation') {

steps {

echo 'Starting Code Compilation...'

sh 'mvn clean compile'

echo 'Code Compilation Completed Successfully!'

}

}

stage('Code QA Execution') {

steps {

echo 'Running JUnit Test Cases...'

sh 'mvn clean test'

echo 'JUnit Test Cases Completed Successfully!'

}

}

stage('Code Package') {

steps {

echo 'Creating WAR Artifact...'

sh 'mvn clean package'

echo 'WAR Artifact Created Successfully!'

}

}

stage('Build & Tag Docker Image') {

steps {

echo 'Building Docker Image with Tags...'

sh "docker build -t satyam88/booking-ms:latest -t booking-ms:latest ."

echo 'Docker Image Build Completed!'

}

}

stage('Docker Image Scanning') {

steps {

echo 'Scanning Docker Image with Trivy...'

sh 'trivy image ${DOCKER\_IMAGE}:latest || echo "Scan Failed - Proceeding with Caution"'

echo 'Docker Image Scanning Completed!'

}

}

stage('Push Docker Image to Docker Hub') {

steps {

script {

withCredentials([string(credentialsId: 'dockerhubCred', variable: 'dockerhubCred')]) {

sh 'docker login docker.io -u satyam88 -p ${dockerhubCred}'

echo 'Pushing Docker Image to Docker Hub...'

sh 'docker push satyam88/booking-ms:latest'

echo 'Docker Image Pushed to Docker Hub Successfully!'

}

}

}

}

stage('Push Docker Image to Amazon ECR') {

steps {

script {

withDockerRegistry([credentialsId: 'ecr:ap-south-1:ecr-credentials', url: "https://533267238276.dkr.ecr.ap-south-1.amazonaws.com"]) {

echo 'Tagging and Pushing Docker Image to ECR...'

sh '''

docker images

docker tag booking-ms:latest 533267238276.dkr.ecr.ap-south-1.amazonaws.com/booking-ms:latest

docker push 533267238276.dkr.ecr.ap-south-1.amazonaws.com/booking-ms:latest

'''

echo 'Docker Image Pushed to Amazon ECR Successfully!'

}

}

}

}

stage('Upload Docker Image to Nexus') {

steps {

script {

withCredentials([usernamePassword(credentialsId: 'nexus-credentials', usernameVariable: 'USERNAME', passwordVariable: 'PASSWORD')]) {

sh 'docker login http://43.205.196.227:8085/repository/booking-ms/ -u admin -p ${PASSWORD}'

echo "Push Docker Image to Nexus : In Progress"

sh 'docker tag booking-ms 43.205.196.227:8085/booking-ms:latest'

sh 'docker push 43.205.196.227:8085/booking-ms'

echo "Push Docker Image to Nexus : Completed"

}

}

}

}

post {

success {

echo '✅ Build completed successfully.'

}

failure {

echo '❌ Build failed.'

}

}

}

============================================================

C: Dockerfile used in IntelliJ for automation

# Use official Tomcat 9 with Java 21 pre-installed

FROM tomcat:9.0.82-jdk21-temurin

# Set maintainer label (optional but good practice)

LABEL maintainer="viraj.pophale@example.com"

# Remove default ROOT app (optional, keeps container clean)

RUN rm -rf /usr/local/tomcat/webapps/ROOT

# Create a user for running the application

RUN useradd -m makemytrip

# Copy your WAR file into the webapps directory

COPY ./target/makemytrip-ms\*.jar /usr/local/tomcat/webapps/

# Expose the default Tomcat port

EXPOSE 8080

# Set the user to 'mbooking-ms' for security

USER makemytrip

# Default command to run Tomcat

CMD ["catalina.sh", "run"]

D.

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"ecr:CompleteLayerUpload",

"ecr:UploadLayerPart",

"ecr:InitiateLayerUpload",

"ecr:BatchCheckLayerAvailability",

"ecr:PutImage",

"ecr:BatchGetImage"

],

"Resource": "arn:aws:ecr:ap-south-1:861276077332:repository/makemytrip"

},

{

"Effect": "Allow",

"Action": "ecr:GetAuthorizationToken",

"Resource": "\*"

}

]

}