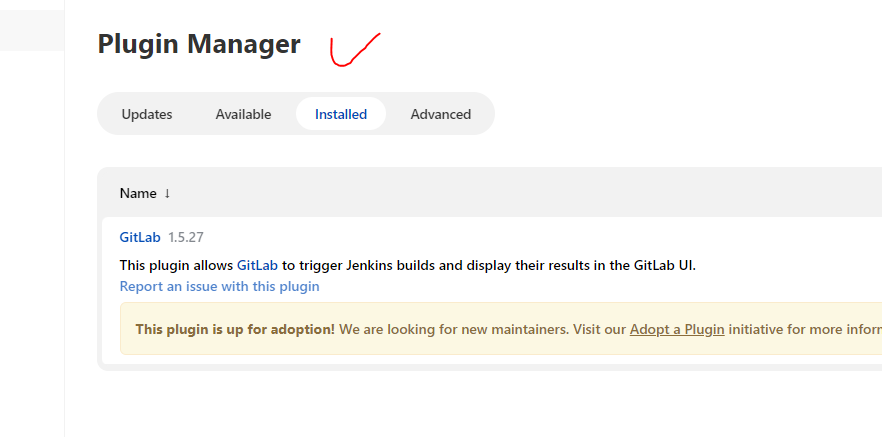
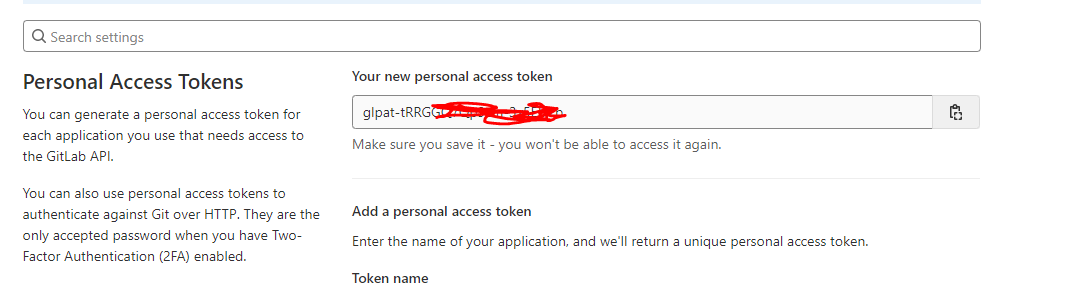
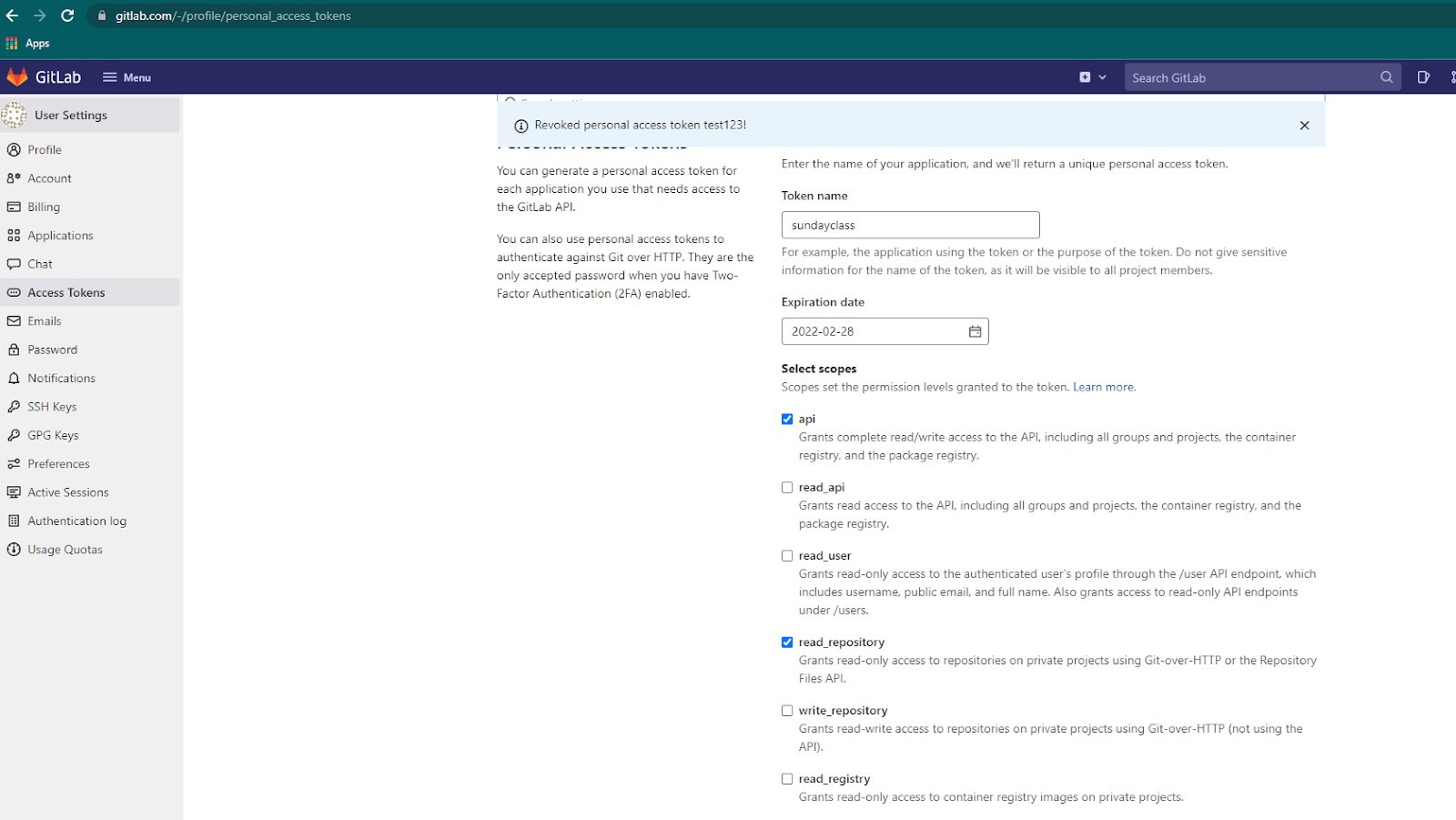
**webhook (kind of gitops - ArgoCD )**

1. install gitlab plugin

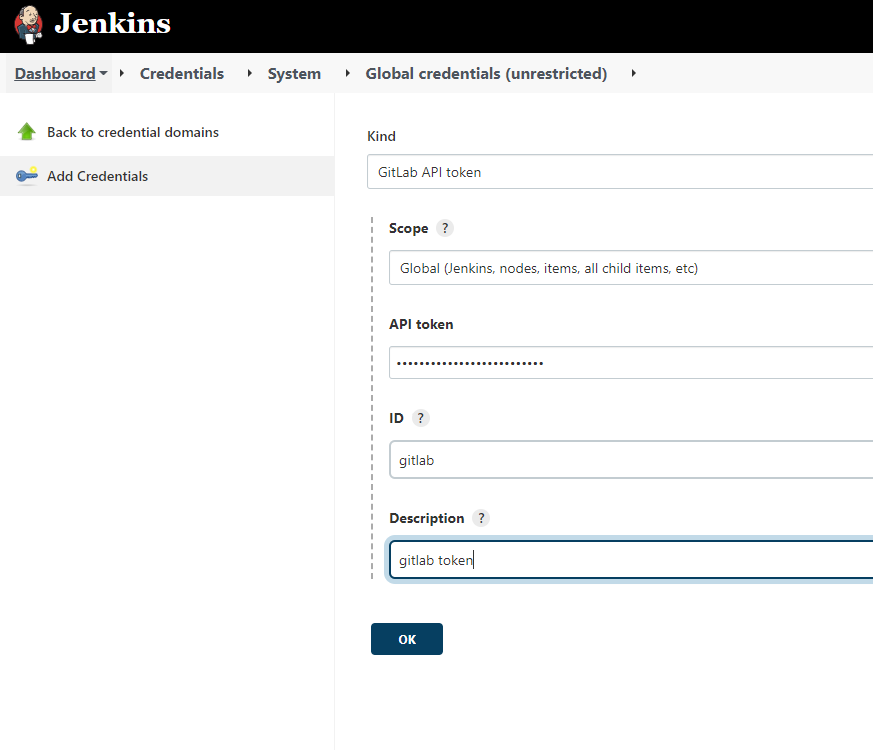




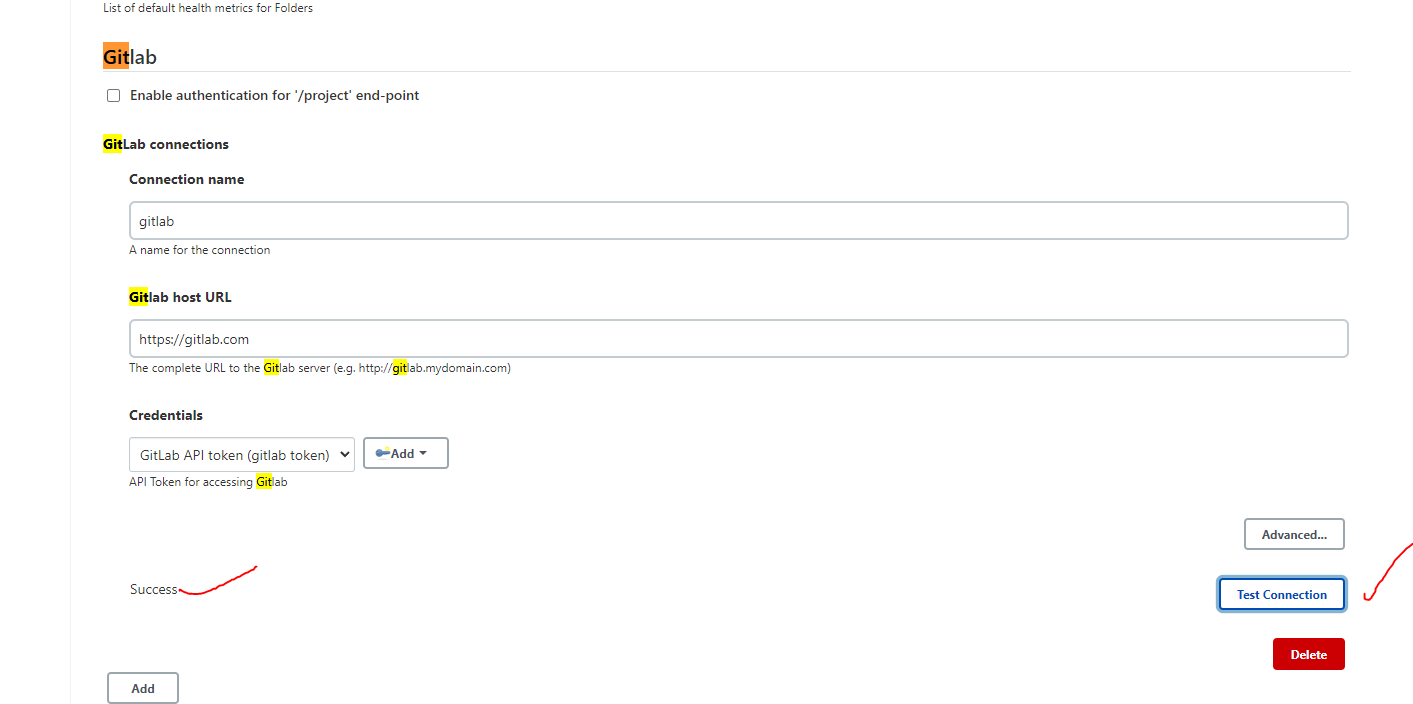


copy the token and create a credentials in Jenkins :

jenkins - manage credentials - global

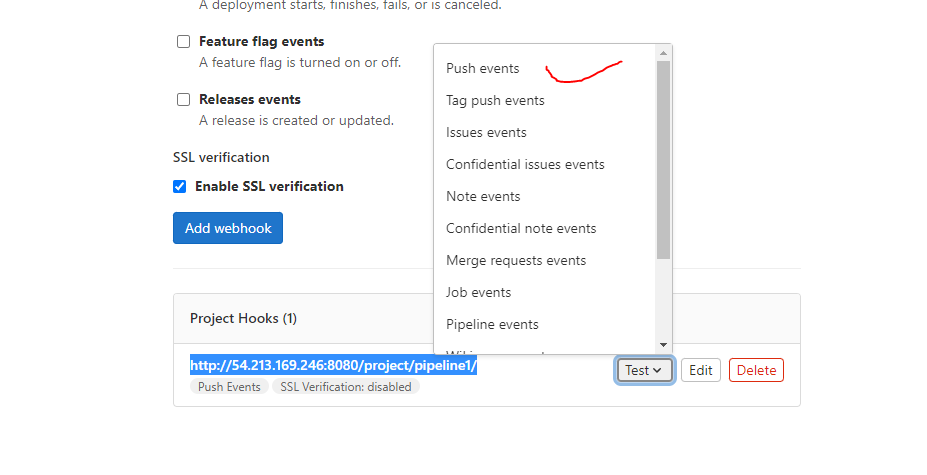


**add the gitlab url to jenkins through configure systems:**

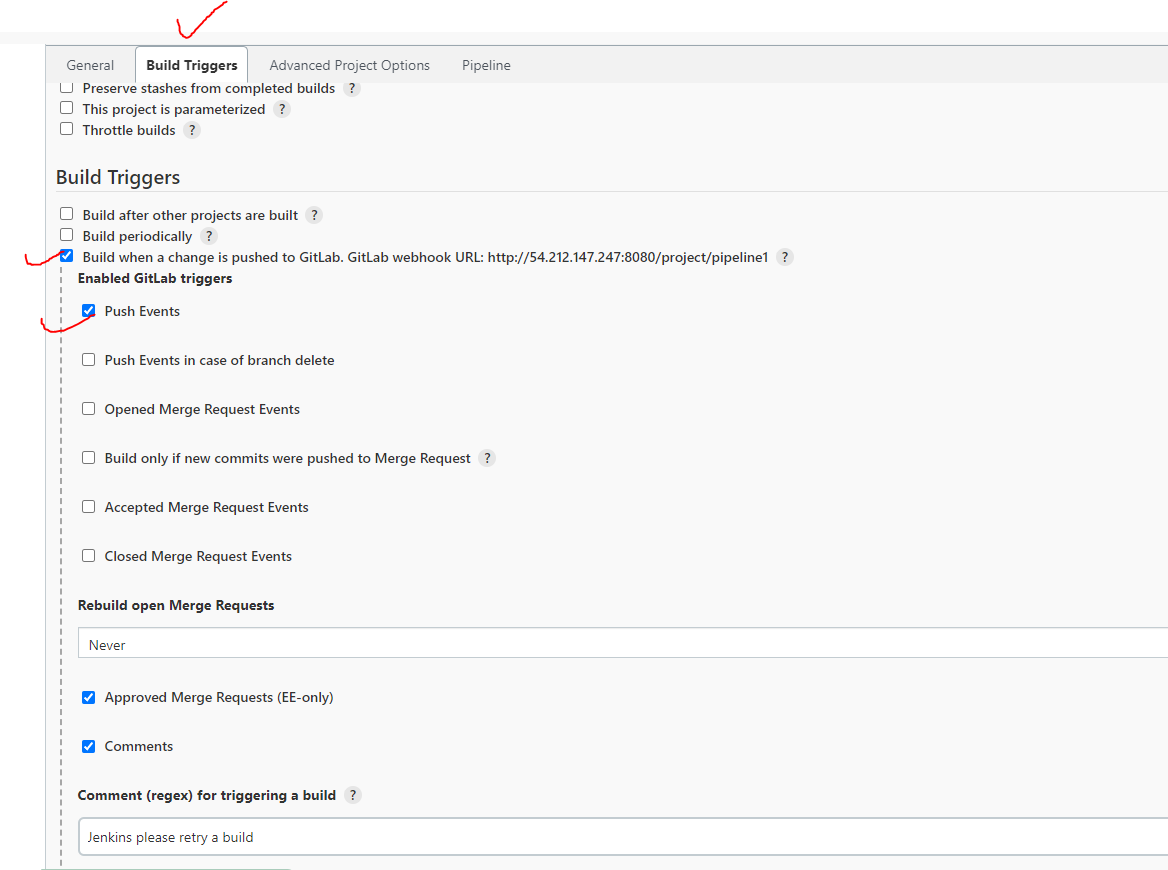


**Got to git lab - select your repository - settings - webhook - disable the ssl verification - then add the jenkins job url and replace the ‘job’ to ‘project’ like this** <http://54.213.169.246:8080/project/pipeline1/> -

Add webhook then test it with a push events option



Then go to jenkins and follow the below steps.



**After this test and do a commit in git lab and see the changes.**

**—----------------------------------------------------------------------**

**Pipe line :**

github

conditions

build (.jar)

unittest

default code test

**pipeline {**

**agent none**

**options {**

**timeout(time: 1,unit: 'HOURS')**

**}**

**stages {**

**stage('Checkout')**

**{**

**agent { label 'slave' }**

**steps {**

**git branch: 'main', url: 'https://gitlab.com/kvpramasamy/worldjava.git'**

**}**

**}**

**stage('PreCheck')**

**{**

**agent { label 'slave' }**

**steps {**

**script {**

**env.BUILDME = "yes" //if yes then run below**

**}**

**}**

**}**

**stage('Build') {**

**when {environment name: 'BUILDME', value: 'yes'}**

**agent { label 'slave' }**

**steps{**

**echo "Build"**

**}**

**}**

**}**

**}**

**no changes happened output!**

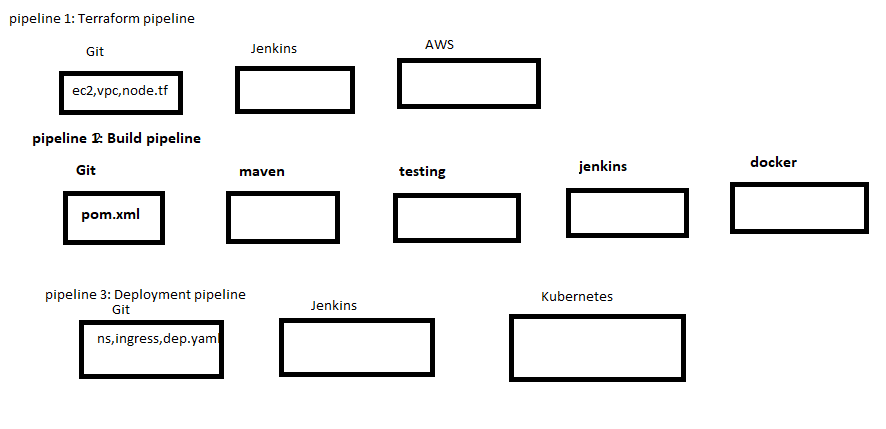
**now doing unit test:**

**Please make sure we have installed maven in the server and also added the path in node properties inside jenkins**

Unit test pipeline: parameter job (to enable or skip -Dmaven.test.skip=true)

| pipeline {  agent none  options {  timeout(time: 1,unit: 'HOURS')  }  parameters {  booleanParam(name: 'UNITTEST', defaultValue: true, description: 'Enable UnitTests ?') //if this enable we need to run this test  }    stages {  stage('Checkout')  {  agent { label 'slave' }  steps {  git branch: 'main', url: 'https://gitlab.com/sudarshanlnx1/worldjava.git'  }  }    stage('PreCheck')  {  agent { label 'slave' }    steps {  script {  env.BUILDME = "yes" //if yes then run below    }  }  }  stage('Build') {  when {environment name: 'BUILDME', value: 'yes'}  agent { label 'slave' }  steps{  script {  if (params.UNITTEST) {   unitstr = ""  } else {  unitstr = "-Dmaven.test.skip=true"  }    echo "Building jar Component ..."  dir ("./samplejar") {  sh "mvn clean package ${unitstr}"   }  // echo "Building War Component ..."  // dir ("./samplewar") {  // sh "mvn clean package"   // }    }  }  }  } } |
| --- |

**Pipeline**

****

**1.First Pipeline: Terraform in Jenkins : -------3**

**Install terraform in slave instance :**

sudo yum install -y yum-utils

sudo yum-config-manager --add-repo <https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo>

sudo yum -y install terraform

terraform --version

aws configure –for credentials

for ubuntu,

install jenkins,

sudo apt update

sudo apt install openjdk-17-jdk -y

<https://www.jenkins.io/doc/book/installing/linux/#debianubuntu>

sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \

<https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key>

echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]" \

https://pkg.jenkins.io/debian-stable binary/ | sudo tee \

/etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt-get update

sudo apt-get install jenkins

sudo systemctl status jenkins

sudo systemctl daemon-reexec

sudo systemctl restart jenkins

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

sudo apt-get install unzip

website, <https://www.terraform.io/downloads.html>

install->ubuntu -> then

wget -O - https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg

echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb\_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list

sudo apt update && sudo apt install terraform

terraform --version

========

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

unzip awscliv2.zip

sudo ./aws/install

aws --version

create secrete text credential AWS\_Access\_Key

create secrete text credential AWS\_Secret\_Key

copy the git url to jenkins in pipeline :

<https://gitlab.com/kvpramasamy/tf-cluster.git>

create credential with secret text for access key and secret key

Pipeline example:

pipeline {

agent any

environment {

AWS\_ACCESS\_KEY\_ID = credentials('AWS\_Access\_Key')

AWS\_SECRET\_ACCESS\_KEY = credentials('AWS\_Secret\_Key')

AWS\_DEFAULT\_REGION = 'ap-south-1'

}

stages{

stage('Clone the Code'){

steps{

git branch: 'main', url: 'https://github.com/Ramasamy3488/k8s-cluster.git'

}

}

stage('Terraform Initialization'){

steps{

script{

dir('dev'){

sh 'terraform init'

}

}

}

}

stage('Terraform Validation'){

steps{

script{

dir('dev'){

sh 'terraform validate'

}

}

}

}

stage('Infrastructure Checks'){

steps{

script{

dir('dev'){

sh 'terraform plan'

}

}

}

}

stage('Create/Destroy EKS cluster'){

steps{

script{

dir('dev'){

sh 'terraform apply --auto-approve'

}

}

}

}

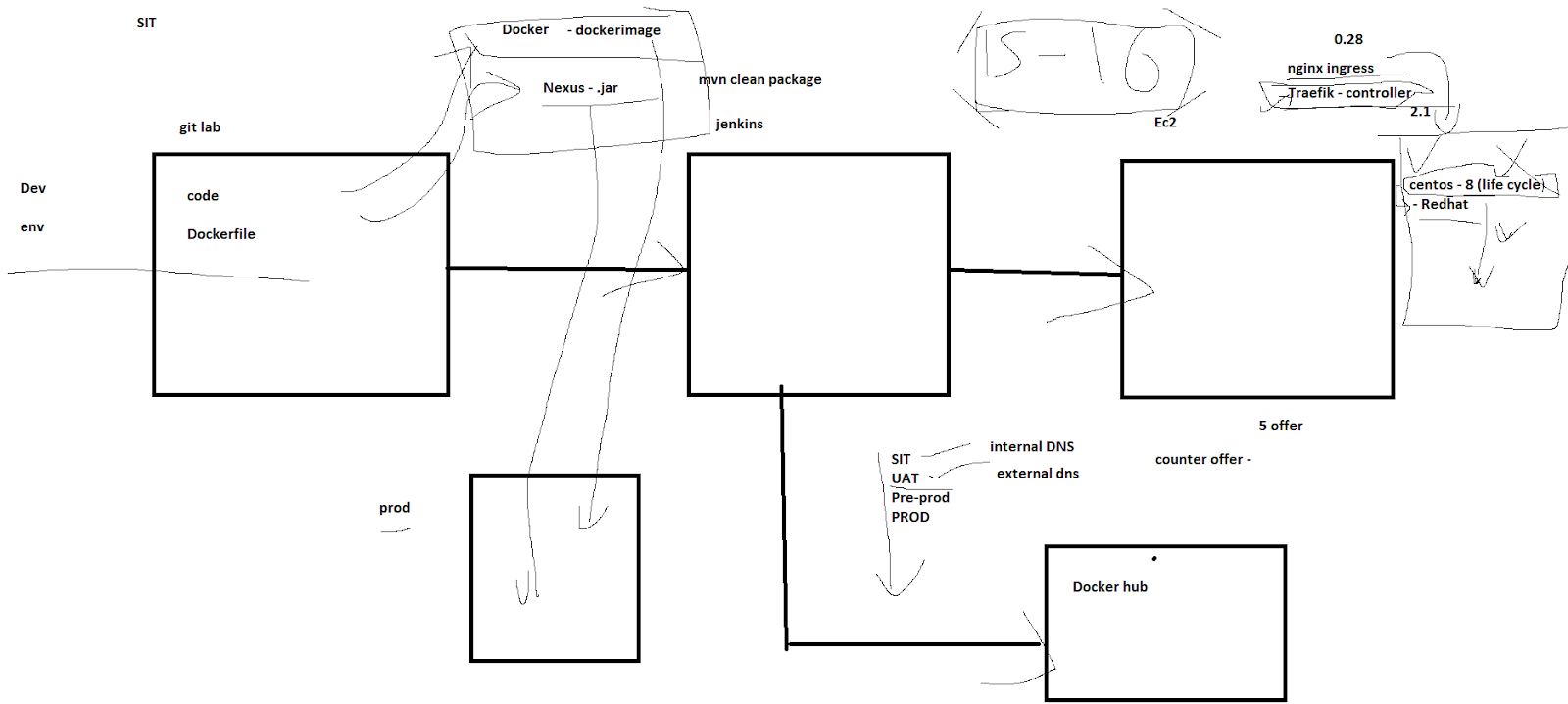
}

}

sh 'terraform init -reconfigure'

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

2.Docker pipeline:------1

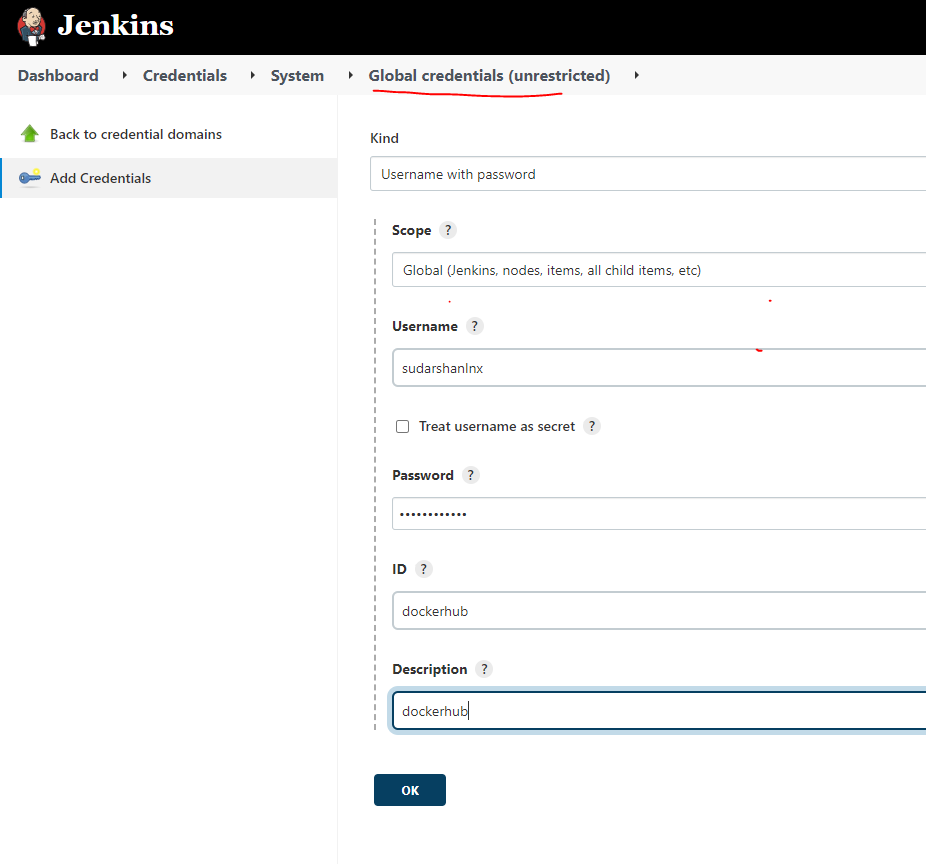


1.

install **docker pipeline** plugin inside jenkins

2.

then create a dockerhub credentials inside jenkins



3.

go to slave ec2 instance :

Be as a ram user

sudo -i

su - ram

then

install docker

sudo yum install docker -y

sudo systemctl restart docker

sudo systemctl enable docker

sudo usermod -a -G docker ramadm

sudo usermod -a -G docker ec2-user

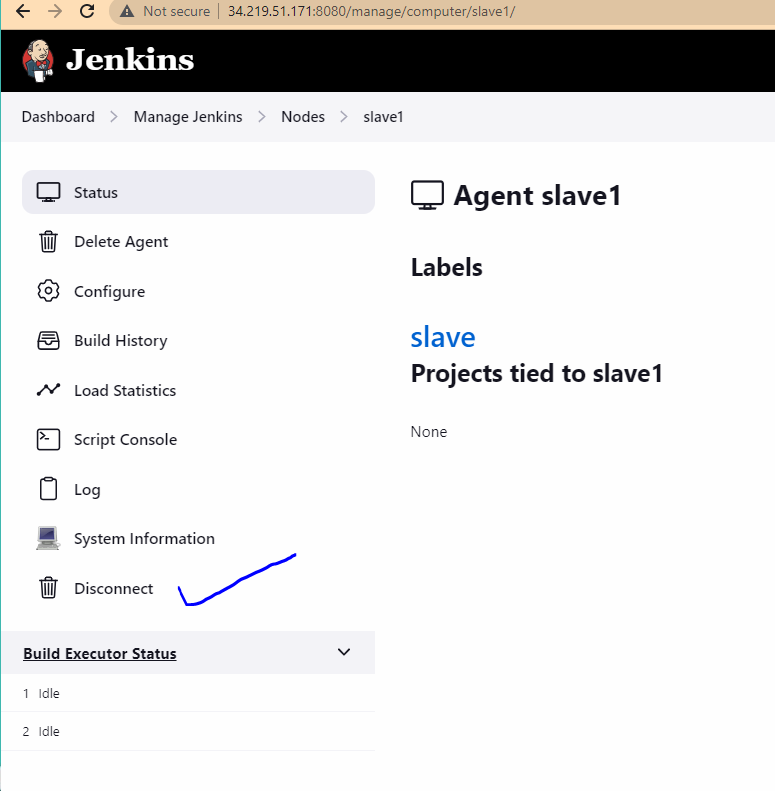
then check with

sudo docker info

o/p should look like this below:

After this relaunch the node!

**disconnect** and relaunch the slave!



sudo -i

useradd ramadm

passwd ramadm

vi /etc/ssh/sshd\_config

PermitRootLogin yes

PasswordAuthentication yes

systemctl restart sshd

vi /etc/sudoers

root ALL=(ALL) ALL

ramadm ALL=(ALL) NOPASSWD: ALL

jenkins ALL=(ALL) NOPASSWD: ALL

sudo usermod -aG docker jenkins

sudo usermod -aG docker ramadm

docker login -u ramasamy123

docker ps

sudo docker ps

docker login

docker ps

cd /etc/sudoers

sudo visudo

docker ps

systemctl restart sshd

docker ps

sudo docker ps

sudo usermod -aG docker ramadm

docker ps

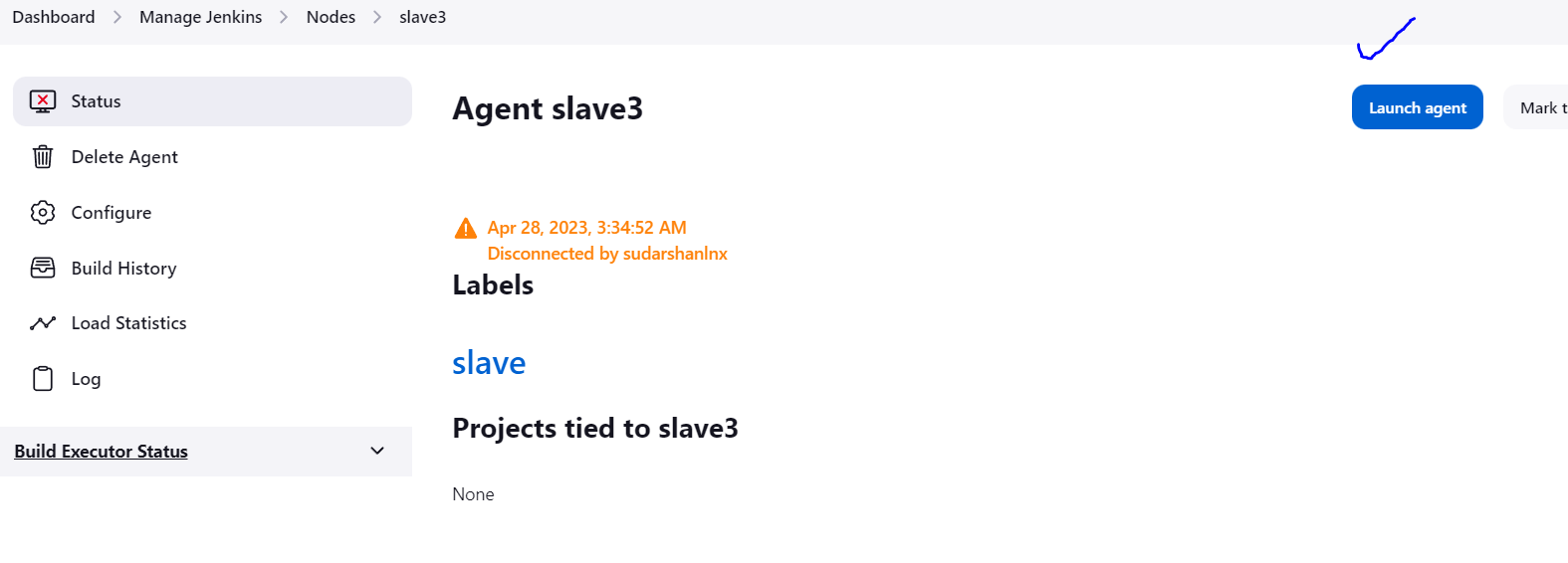
sudo systemctl start docker

docker ps

ls -l /var/run/docker.sock

sudo chmod 666 /var/run/docker.sock

docker ps



Then in slave machine:

then check with

sudo docker info

o/p should look like this below:

| init version: de40ad0  Security Options:  seccomp  Profile: default  Kernel Version: 5.10.82-83.359.amzn2.x86\_64  Operating System: Amazon Linux 2  OSType: linux  Architecture: x86\_64  CPUs: 1  Total Memory: 965.5MiB  Name: ip-172-31-19-145.us-west-2.compute.internal  ID: FO6R:6HZC:TALV:XE2G:GITX:BIHQ:UQHG:45Q7:NPCE:WBTR:JXKY:YT3M  Docker Root Dir: /var/lib/docker  Debug Mode: false  Registry: https://index.docker.io/v1/  Labels:  Experimental: false  Insecure Registries:  127.0.0.0/8  Live Restore Enabled: false |
| --- |

**Docker working code:**

**make sure that the below git repo is having Dockerfile**

pipeline {  
 agent any

options {  
 timeout(time: 1,unit: 'HOURS')  
 }  
 parameters {  
 booleanParam(name: 'UNITTEST', defaultValue: true, description: 'Enable UnitTests ?') //if this enable we need to run this test  
 }  
   
 stages {  
 stage('Checkout')  
 {  
   
 steps {  
 git branch: 'main', url: 'https://gitlab.com/kvpramasamy/worldjava.git'  
 }  
 }  
   
 stage('PreCheck')  
 {  
 agent { label 'slavegroup' }  
   
 steps {  
 script {  
 env.BUILDME = "yes" //if yes then run below  
   
 }  
 }  
 }  
 stage('Build') {  
 when {environment name: 'BUILDME', value: 'yes'}  
 agent { label 'slavegroup' }  
 steps{  
 script {  
 if (params.UNITTEST) {   
 unitstr = ""  
 } else {  
 unitstr = "-Dmaven.test.skip=true"  
 }  
   
 echo "Building jar Component ..."  
 dir ("./samplejar") {  
 sh "mvn clean package ${unitstr}"   
 }  
 // echo "Building War Component ..."  
 // dir ("./samplewar") {  
 // sh "mvn clean package"   
 // }   
  
 }  
 }  
 }  
 stage('Build Image')   
 {  
 agent { label 'slavegroup' }  
 when {environment name: 'BUILDME', value: 'yes'}  
 steps{  
 script {  
 docker.withRegistry( 'https://registry.hub.docker.com', 'dockerhub' ) {  
 /\* Build Docker Image locally \*/  
 myImage = docker.build("ramasamy123/ramjava:v1")  
  
 /\* Push the container to the Registry \*/  
 myImage.push()  
 }  
 }  
 }  
 }  
  
 }  
}

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

**3.Kubernetes deployment:**

**Jenkins - kubernetes deployment**

**2- tier**

**Example for kubernetes :**

[**https://shadow-soft.com/jenkins-pipeline-containerized-applications-kubernetes/**](https://shadow-soft.com/jenkins-pipeline-containerized-applications-kubernetes/)

**example github:**

[**https://github.com/kitconcept/jenkins-pipeline-examples**](https://github.com/kitconcept/jenkins-pipeline-examples)

**task 1**

1. **bluegoon website**
2. **docker image deploy - Ec2**

**task 2**

1. **bluegoon website**
2. **docker image deploy - EKS**

**1. Slave preparation :**

**1.**

**install latest aws cli and set the path in .bash\_profile \*\*important**

**- aws configure**

**- awscli download**

**download the latest aws-cli**

**(always refer the website for latest versions)**

**https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html**

**history:**

**sudo curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"**

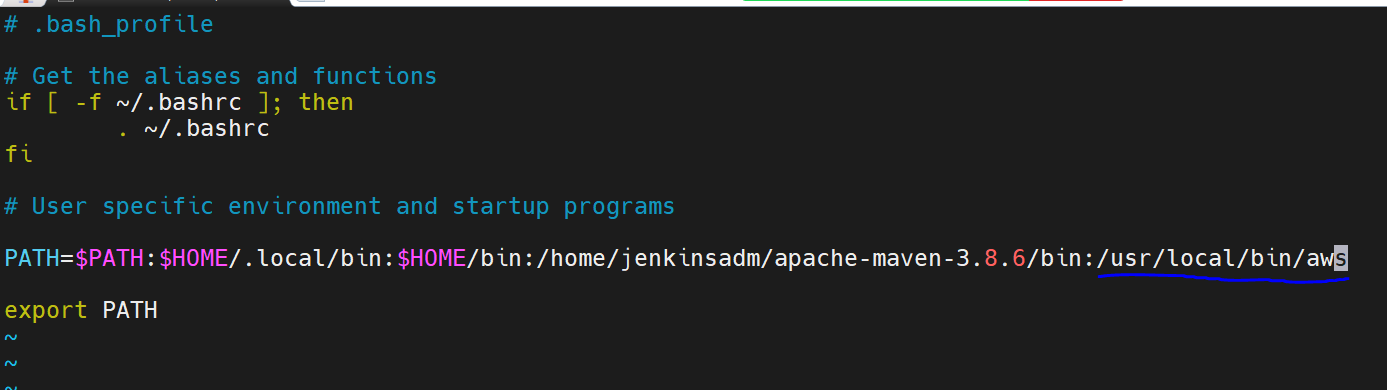
**sudo unzip awscliv2.zip**

**sudo ./aws/install**

**sudo vi .bash\_profile \*\*important**

**source .bash\_profile**

**add the latest aws cli in .bash\_profile PATH**

****

**source .bash\_profile**

**aws configure**

**aws s3 ls**

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

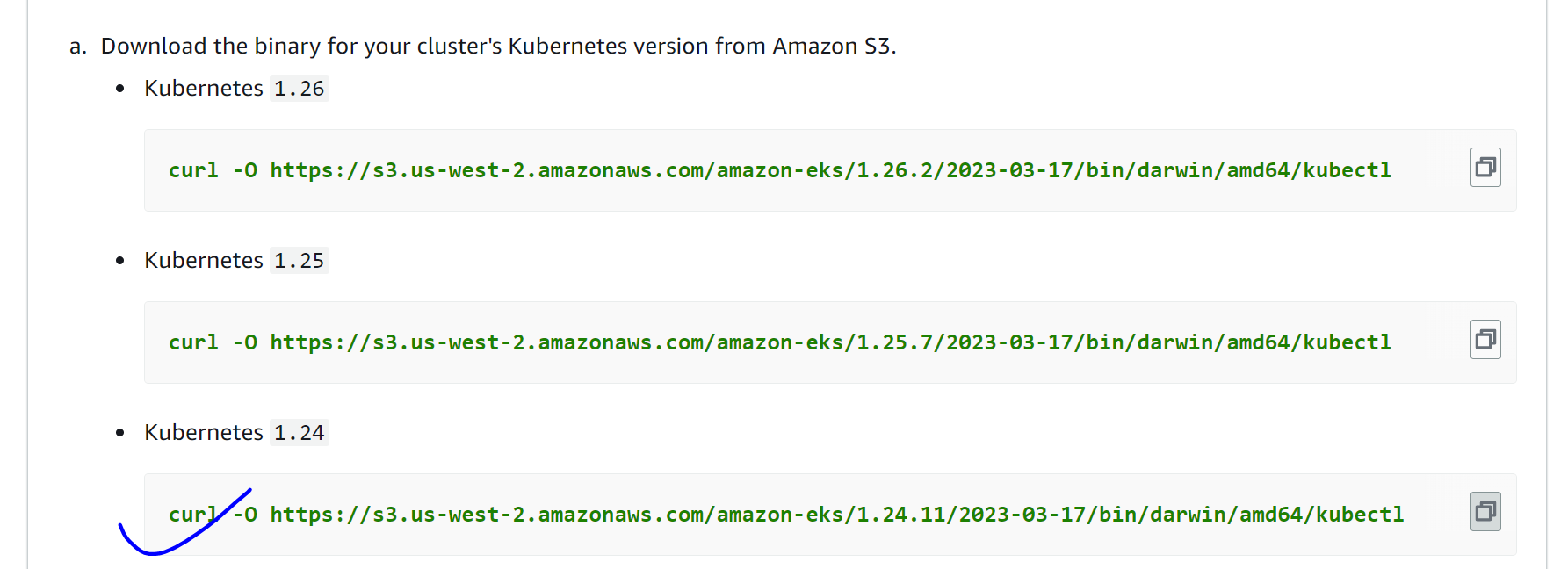
**check:**

**- kubectl download**

**#Download kubectl according to the kubernetes cluster version:**

**2.download kubectl for 1.31(as your cluster version)**

[**https://docs.aws.amazon.com/eks/latest/userguide/install-kubectl.html**](https://docs.aws.amazon.com/eks/latest/userguide/install-kubectl.html)

****

**1.**

**curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.0/2024-05-12/bin/linux/amd64/kubectl**

curl -LO [https://dl.k8s.io/release/**$(**curl -L -s https://dl.k8s.io/release/stable.txt**)**/bin/linux/amd64/kubectl](https://dl.k8s.io/release/$(curl%20-L%20-s%20https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl)

sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

**sudo chmod +x ./kubectl**

**mkdir bin**

**sudo chown ramadm:ramadm bin**

**sudo mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$PATH:$HOME/bin**

**sudo mv kubectl /usr/local/bin/**

**for ubuntu,**

[**https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/**](https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/)

**curl -LO** [**https://dl.k8s.io/release/v1.32.0/bin/linux/amd64/kubectl**](https://dl.k8s.io/release/v1.32.0/bin/linux/amd64/kubectl)

**sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl**

**chmod +x kubectl**

**mkdir -p ~/.local/bin**

**mv ./kubectl ~/.local/bin/kubectl**

**kubectl version --client**

**aws-cli,**

**sudo apt install unzip -y**

**curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"**

**unzip awscliv2.zip**

**sudo ./aws/install --update**

**cat /var/lib/jenkins/.kube/config**

**exec:**

**apiVersion: client.authentication.k8s.io/v1**

**v1 change to v1alpha1, you can manually edit it or delete the config and regenerate.**

**aws eks --region ap-south-1 update-kubeconfig --name devcluster**

**aws configure**

**kubectl get ns**

**2. kubernetes cluster preparation:**

**- create ns (dev project)**

**- create Ingress controller**

**- Add the LB to Route53**

**- EBS CSI driver preparation for the DB container**

**3. Add the yaml files to GITLAB**

**4. Create the pipeline**

**- connect GITLab**

**- add the kube files to be applied**

| **pipeline {  agent none  options {  timeout(time: 1,unit: 'HOURS')  }  stages {  stage('Checkout')  {  agent { label 'slavegroup' }  steps {  git branch: 'main',   credentialsId: 'gitlab\_user',  url: 'https://gitlab.com/kvpramasamy/sat-kube-deploy.git'   }  }    stage('PreCheck')  {  agent { label 'slave' }    steps {  script {  env.BUILDME = "yes" //if yes then run below    }  }  }    stage('Deploy Kubernates')   {  agent { label 'slave' }  when {environment name: 'BUILDME', value: 'yes'}  steps{  script {**  **sh 'aws eks --region ap-south-1 update-kubeconfig --name devcluster'**  **sh 'kubectl apply -f ns.yaml'  sh 'kubectl apply -f pvc.yaml'  sh 'kubectl apply -f mongodb.yaml'  sh 'kubectl apply -f webcon.yaml'  }   }  }    } }** |
| --- |
|  |

**pipeline {**

**agent any**

**stages {**

**stage('Install Kubectl') {**

**steps {**

**sh '''**

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

**chmod +x kubectl**

**sudo mv kubectl /usr/local/bin/**

**'''**

**}**

**}**

**stage('Apply Kubernetes Resources') {**

**steps {**

**sh 'kubectl apply -f ns.yaml'**

**}**

**}**

**}**

**}**

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

pipeline {

agent any

environment {

AWS\_ACCESS\_KEY\_ID = credentials('AWS\_Access\_Key')

AWS\_SECRET\_ACCESS\_KEY = credentials('AWS\_Secret\_Key')

AWS\_DEFAULT\_REGION = 'us-east-2'

}

options {

timeout(time: 1,unit: 'HOURS')

}

stages {

stage('Checkout')

{

steps {

git branch: 'main',

credentialsId: 'gitlab\_user',

url: 'https://gitlab.com/kvpramasamy/sat-kube-deploy.git'

}

}

stage('PreCheck')

{

steps {

script {

env.BUILDME = "yes" //if yes then run below

}

}

}

stage('Deploy Kubernates')

{

when {environment name: 'BUILDME', value: 'yes'}

steps{

script {

sh 'aws eks --region us-east-2 update-kubeconfig --name devcluster'

sh 'kubectl apply -f ns.yaml'

// sh 'kubectl apply -f pvc.yaml'

// sh 'kubectl apply -f mongodb.yaml'

sh 'kubectl delete -f newIngressctrl.yaml'

// sh 'kubectl delete -f webcon.yaml'

}

}

}

}

}