**Integrate Kubernetes With Jenkins Pipeline**

**Jenkins Machine:**

Create EC2 Instance (Ubuntu Server) with Instance Type: t2.micro 🡪For Installing Jenkins

**Master Machine:**

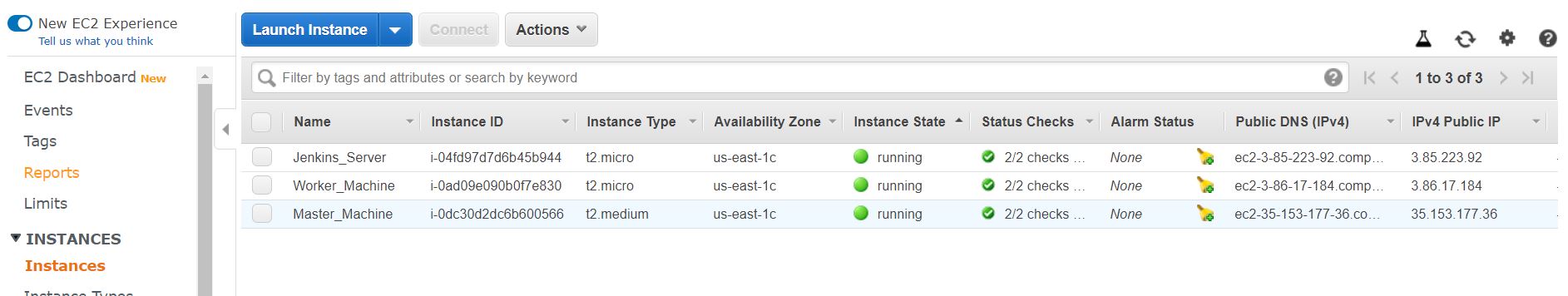
Create EC2 Instance (Ubuntu Server) with Instance Type: t2.medium 🡪For Kubeadm cluster

Kubeadm cluster :

Requirements: Minimum one Master Machine and one Worker Machine

Master Machine: Requires minimum 2CPU’s and 4GiB

Worker Machine: Requires minimum 1CPU and 1GiB



**Installations on Jenkins Machine:**

Install Jenkins and Docker on Jenkins Machine using below commands

sudo apt update

sudo apt install openjdk-8-jdk -y

jenkis installation:

wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -

sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'

sudo apt update

sudo apt install jenkins

sudo systemctl start jenkins

sudo systemctl enable jenkins

Docker installation:

sudo curl -fsSL get.docker.com | /bin/bash

sudo usermod -aG docker jenkins

sudo systemctl restart Jenkins

To get password of Jenkins for the first time:

sudo cat /var/lib/jenkins/secrets/initialAdminPassword

**Installations on Master Machine and Worker Machine:**

Below commands are common for both Machines

sudo apt-get update

sudo apt-get install -y apt-transport-https

switch to root user:

sudo su -

curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -

cat <<EOF >/etc/apt/sources.list.d/kubernetes.list

deb http://apt.kubernetes.io/ kubernetes-xenial main

EOF

swapoff -a

sed -i '/ swap / s/^\(.\*\)$/#\1/g' /etc/fstab

modprobe br\_netfilter

sysctl -p

sysctl net.bridge.bridge-nf-call-iptables=1

Docker Installation:

apt-get update -y

apt install docker.io -y

usermod -aG docker ubuntu

Install kubelet, kubeadm, kubectl, kubernetes-cni:

apt-get install -y kubelet kubeadm kubectl kubernetes-cni

systemctl daemon-reload

systemctl start kubelet

systemctl enable kubelet.service

**Creating Kubeadm Cluster on Master Machine:**

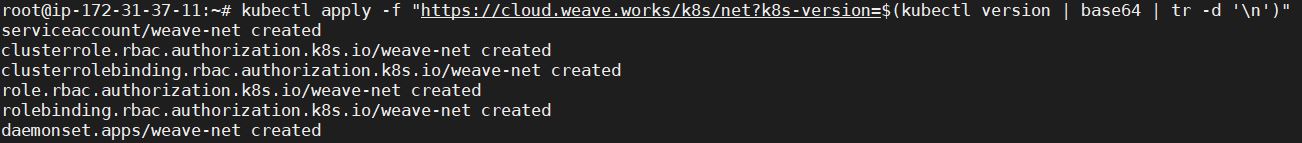
kubeadm init

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

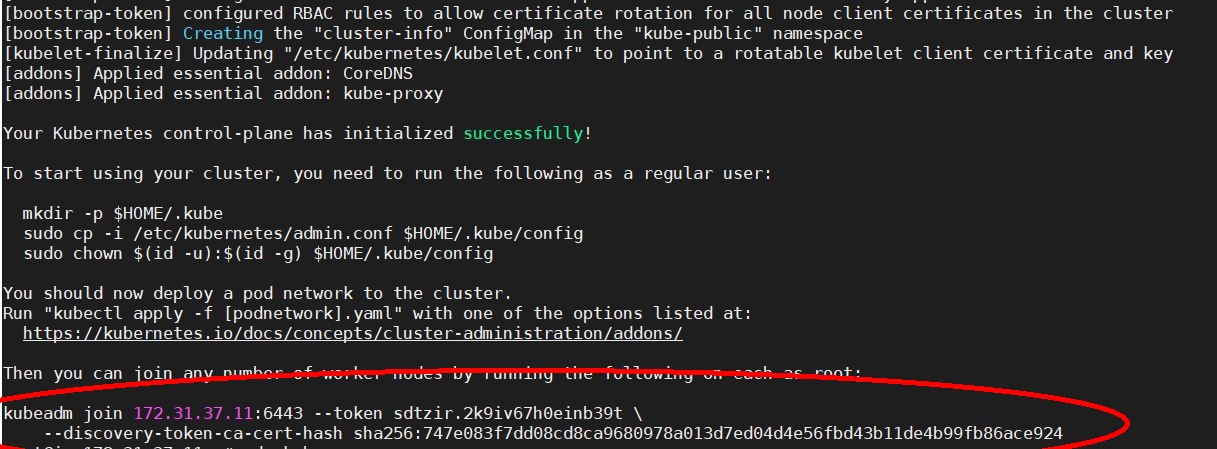
sudo chown $(id -u):$(id -g) $HOME/.kube/config

kubectl apply -f [https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')](https://cloud.weave.works/k8s/net?k8s-version=$(kubectl%20version%20|%20base64%20|%20tr%20-d%20'\n'))



To join nodes to Master Machine :

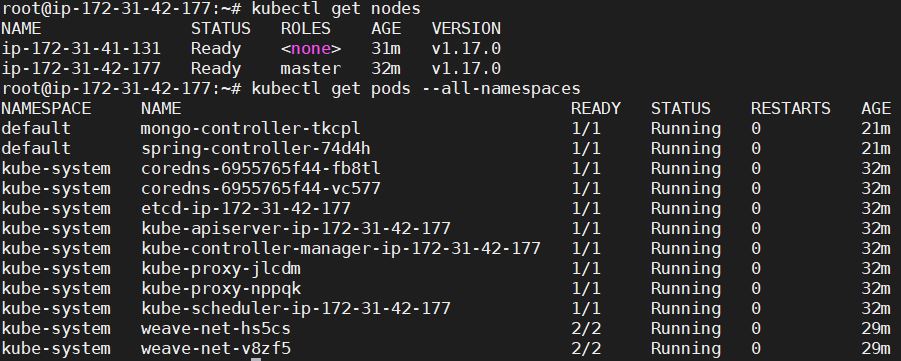
While we initiate kubeadm we get like as below. This kubeadm join to be run on worker machine to join nodes to Master Machine



checking nodes and pods:

kubectl get nodes

kubectl get pods --all-namespaces



**Open Jenkins on GUI:**

# Go Manage Jenkins 🡪 Manage Plugins 🡪 Available

# Add below plugins:

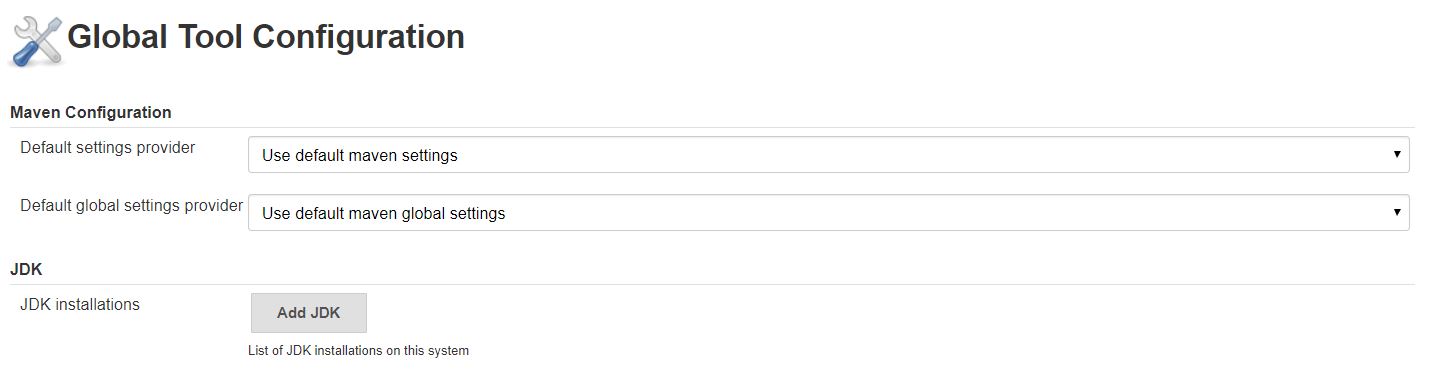
Maven Integration

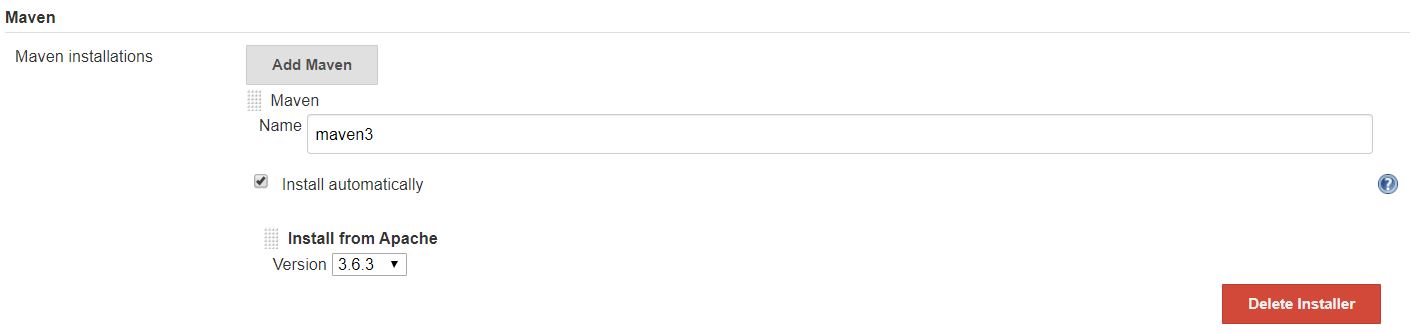
CloudBees Docker Build and Publish plugin

Kubernetes Continuous Deploy Plugin

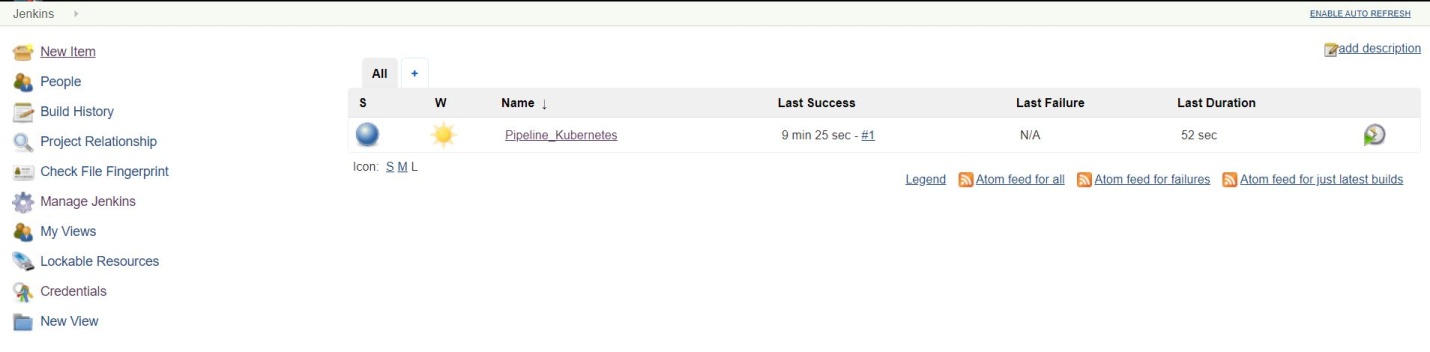
Manage Jenkins 🡪 Global Tool Configuration 🡪 Maven

Give Maven name as maven3



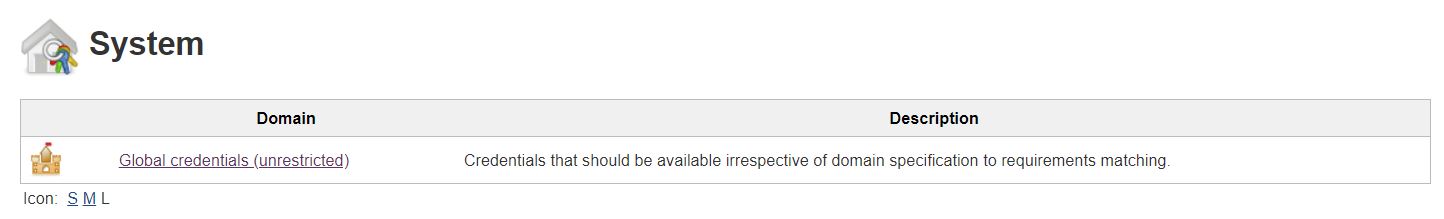


**Credentials:**

* Open Jenkins GUI 
* Click on Credentials



* Click on Jenkins



* Click on Global Credentials(unrestricted)

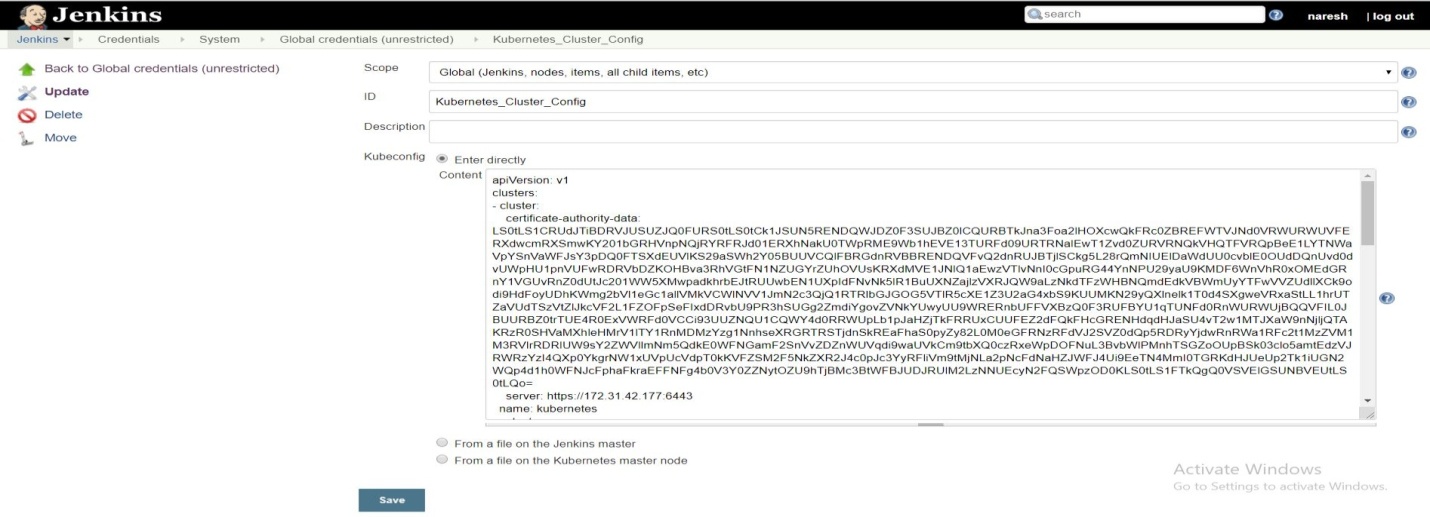
Git credentials:



Docker credentials:

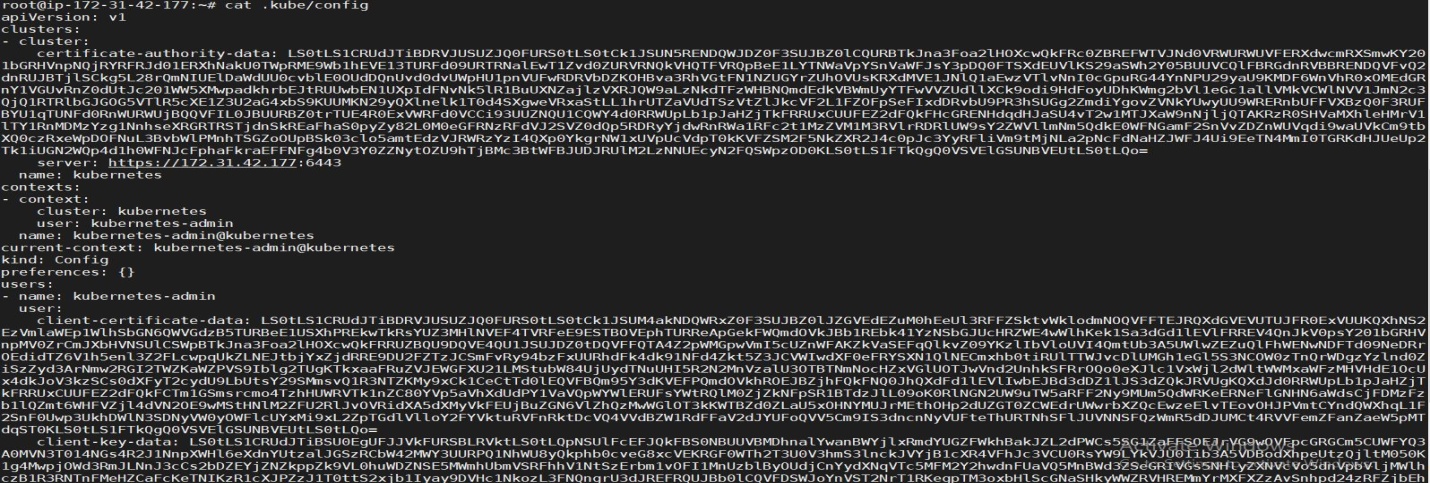


Kubernetes configuration (kubeconfig):

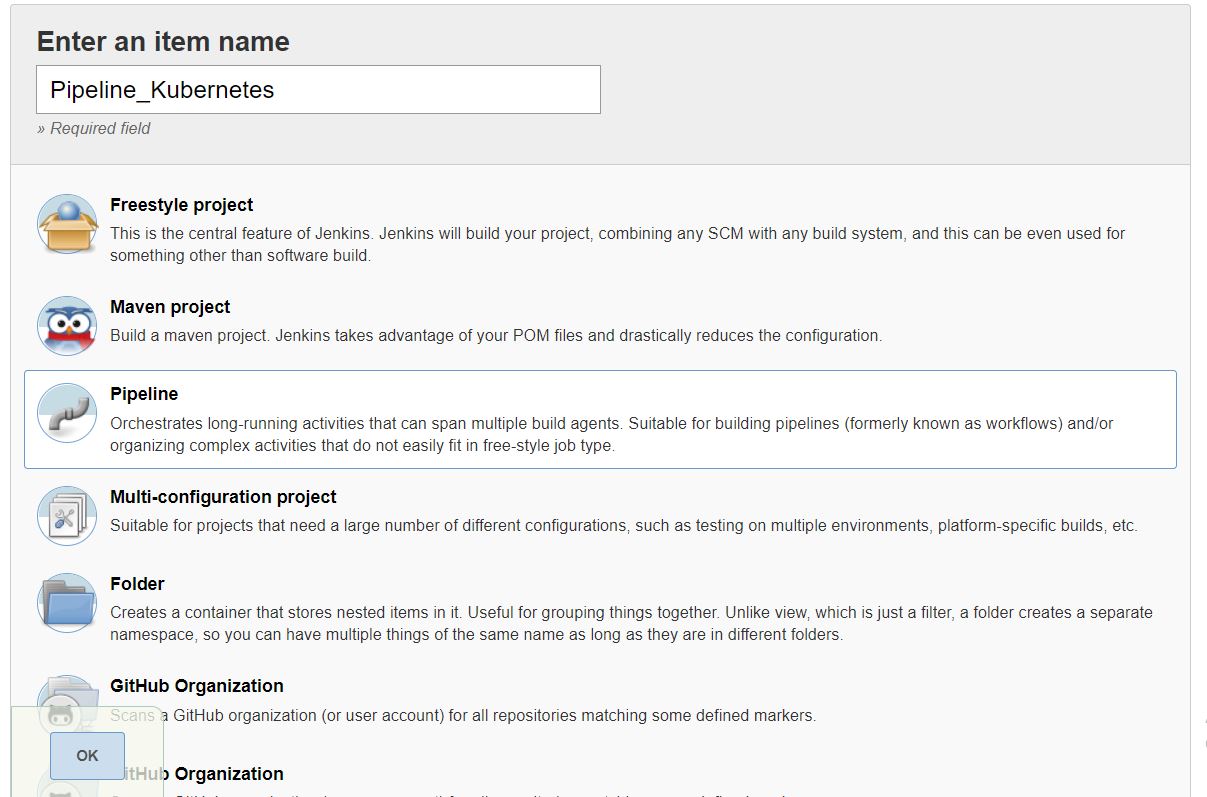


Open Master machine and run below command for getting kubeconfig content

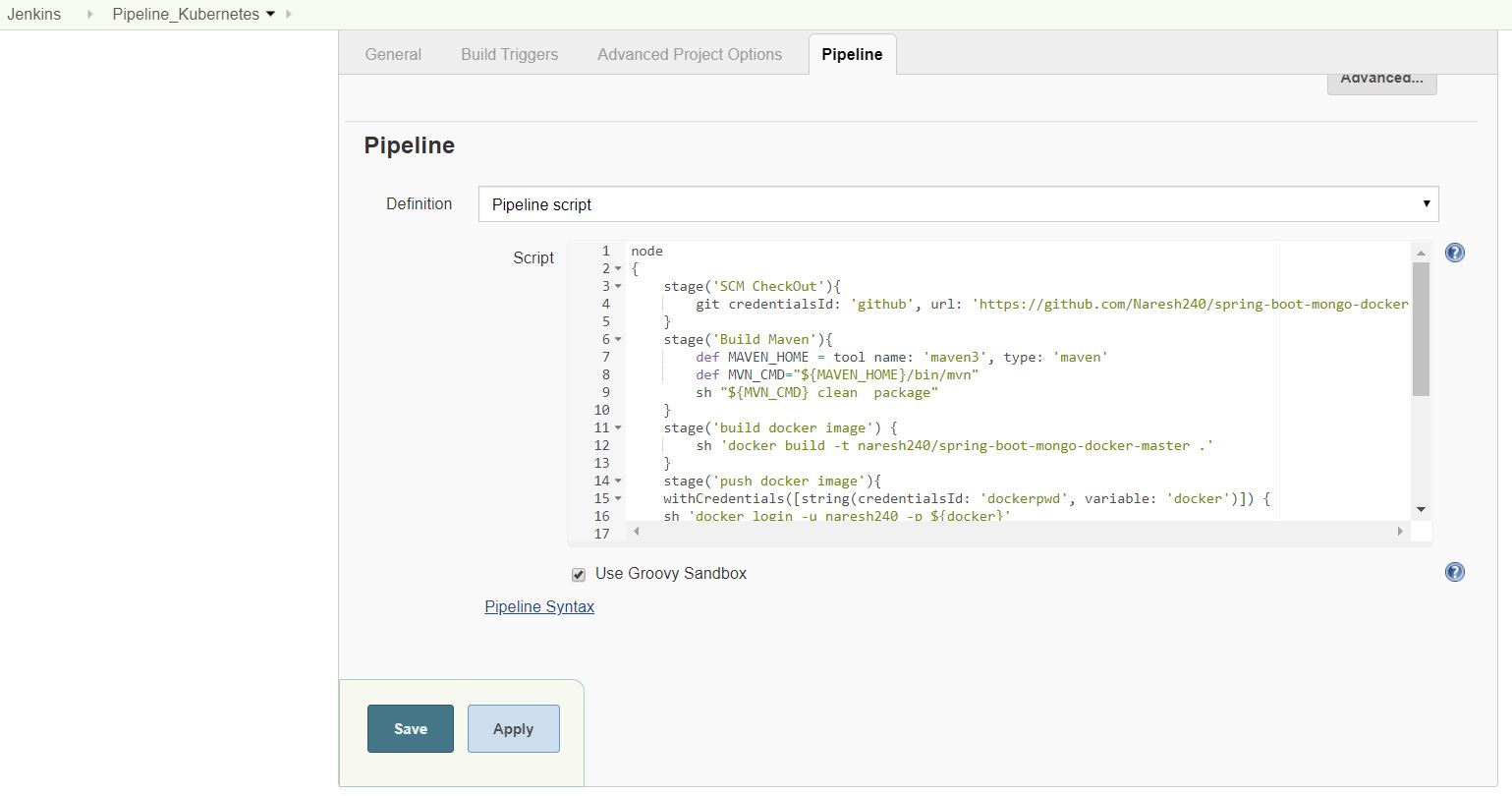
cat .kube/config



Create New job using Jenkins pipeline:



Pipeline Code:



* Open pipeline tab and paste below code

node

{

stage('SCM CheckOut'){

git credentialsId: 'github', url: 'https://github.com/Naresh240/spring-boot-mongo-docker-master.git'

}

stage('Build Maven'){

def MAVEN\_HOME = tool name: 'maven3', type: 'maven'

def MVN\_CMD="${MAVEN\_HOME}/bin/mvn"

sh "${MVN\_CMD} clean package"

}

stage('build docker image') {

sh 'docker build -t naresh240/spring-boot-mongo-docker-master .'

}

stage('push docker image'){

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

sh 'docker login -u naresh240 -p ${docker}'

}

sh 'docker push naresh240/spring-boot-mongo-docker-master'

}

stage('Deploy Application on K82 Cluster'){

kubernetesDeploy(

configs: 'springBootMongo.yml',

kubeconfigId: 'Kubernetes\_Cluster\_Config',

enableConfigSubstitution: true

)

}

}

Build Project:

* Click on Build Now

