DevOps(DS457)

Assignment - 1

Jenkins CICD pipeline for Kubernetes

Submitted to
Dr.Uma.S

Submitted by
Team13

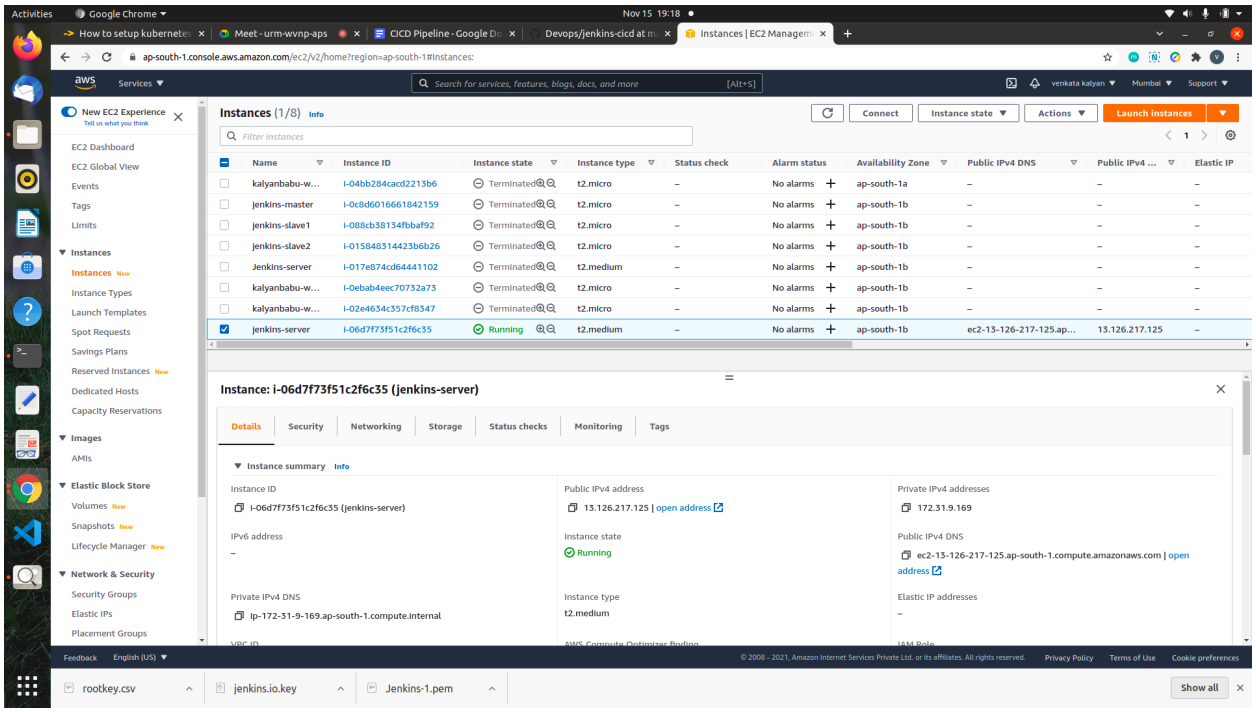
M.P. Bharath(18BCS057)
T.Umesh Anand babu(18BCS105)
M.Venkata Kalyan Babu(18BCS049)
Y.Mokshith ramendra(18BCS112)

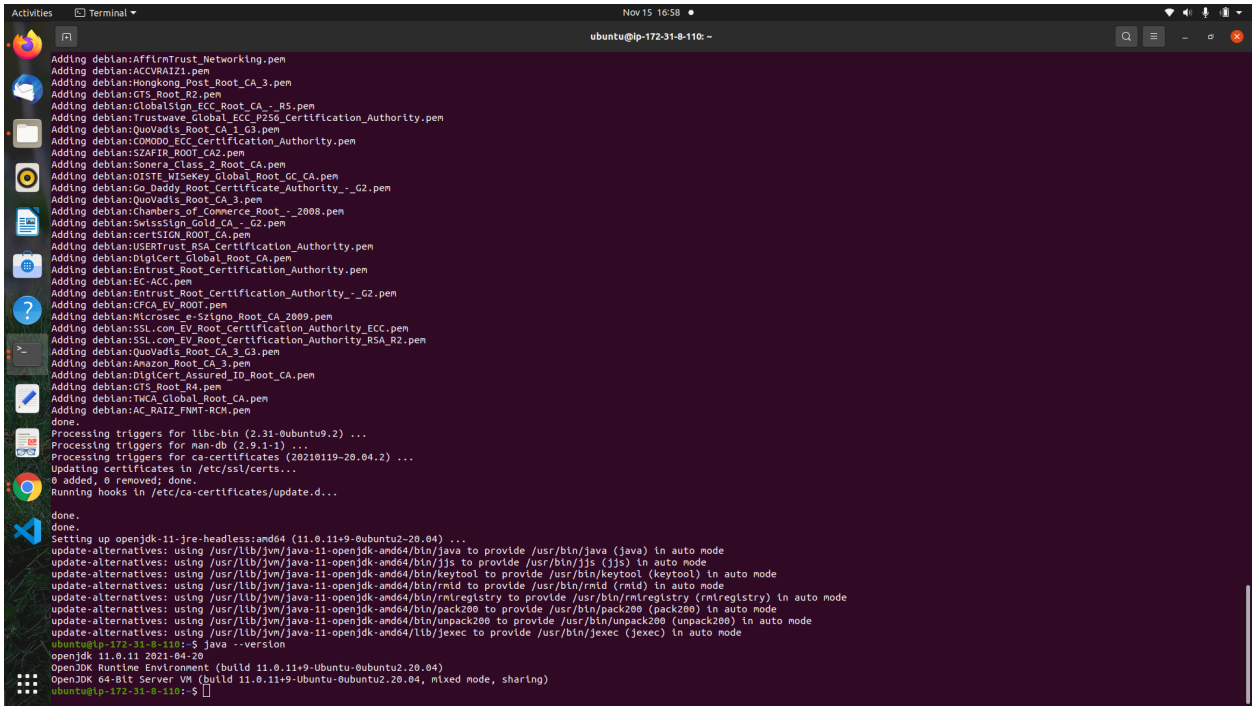
Setting up CI CD pipeline using jenkins to deploy on kubernetes

- The first step would be for us to set up an EC2 instance and on this instance, we will be installing -

    - JDK
    - Jenkins
    - eksctl
    - kubectl

Create and Launch EC2 instance of type t2.medium with ubuntu 20.04 OS and with all other options as default. Also connect to the ec-2 instance using Putty for Windows or using ssh for ubuntu respectively.





Install JDK on AWS EC2 Instance

- ➢ *sudo apt-get update*
- ➢ *sudo apt install openjdk-11-jre-headless*
- ➢ *java -version*

Install and Setup Jenkins

Setup jenkins

- add the Jenkins repository to the package manager

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

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

➢ *sudo apt-get update*

- After adding the repository link of Jenkins update the package manager

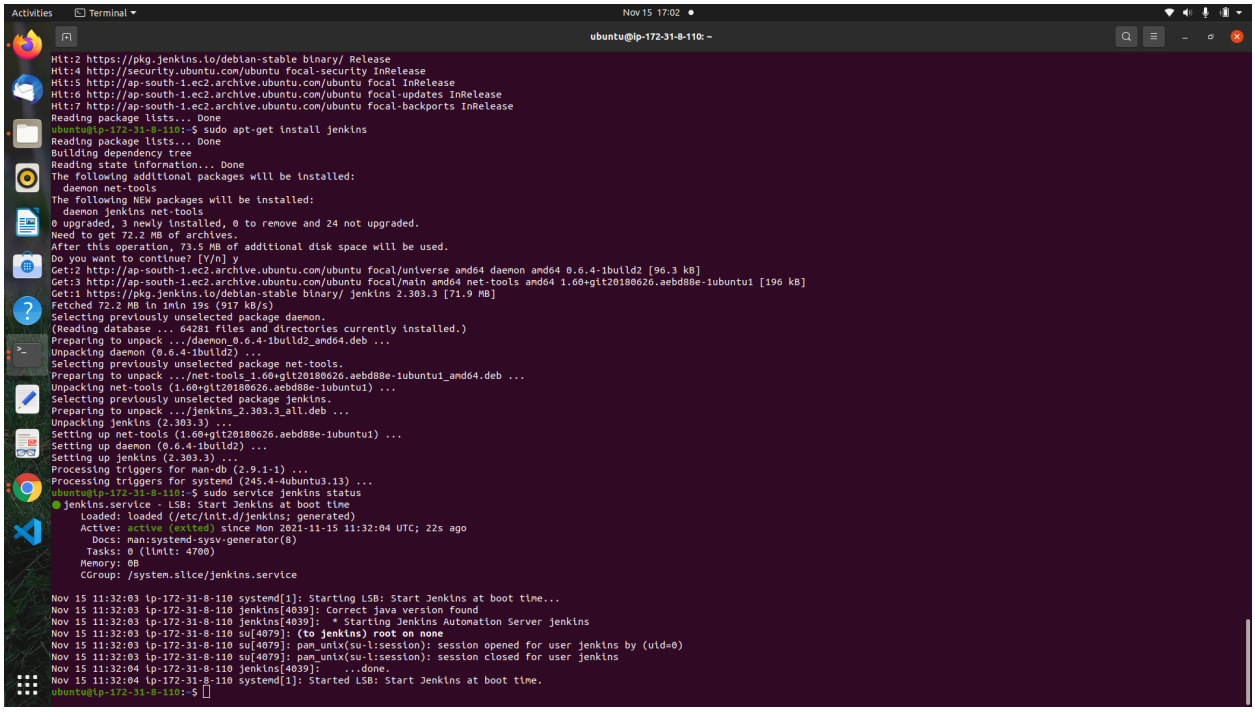
➢ *sudo apt-get update*

- Then finally install Jenkins using the following command

➢ *sudo apt-get install jenkins*

- On successful installation, you should see Active Status

➢ *sudo service jenkins status*

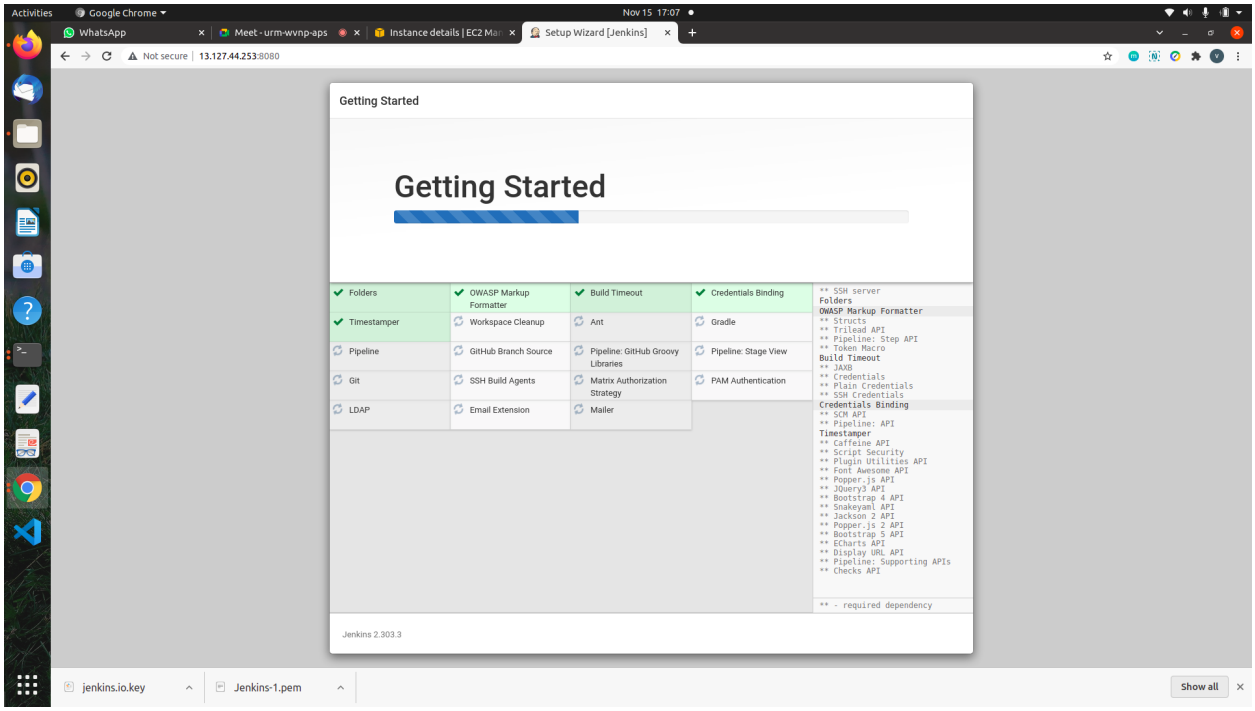


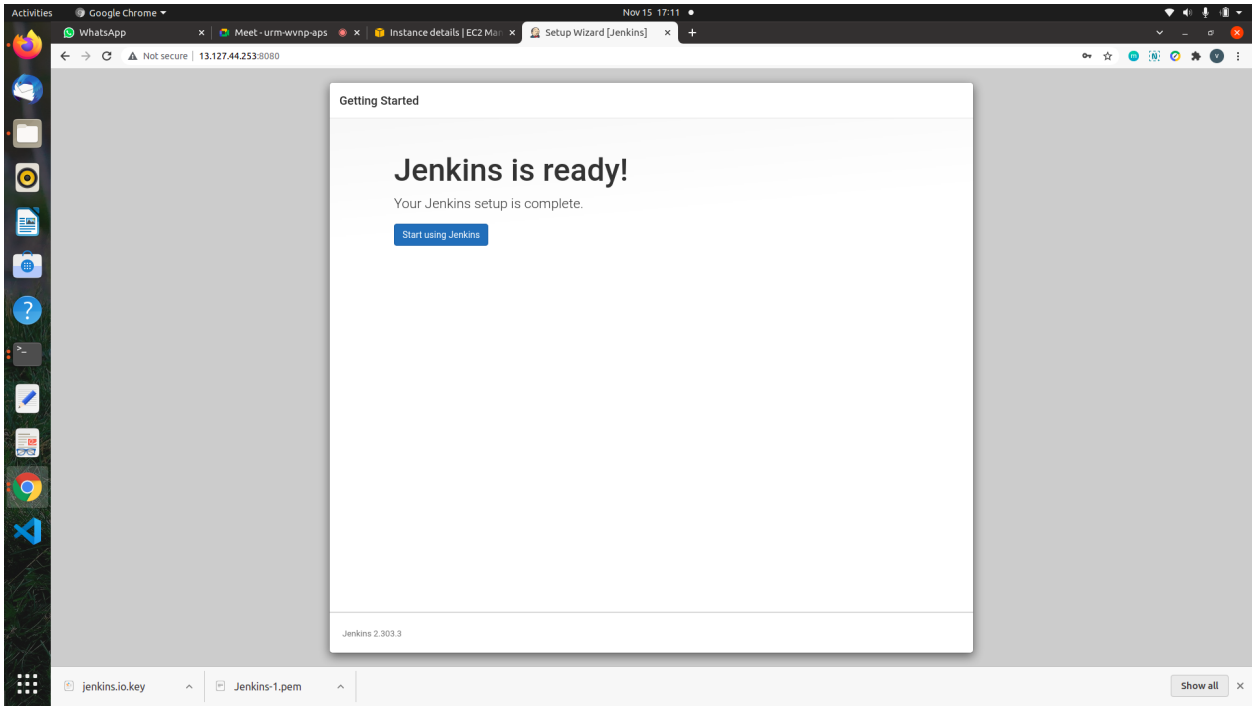
- Now we need to start using jenkins from the public ipv4 address of the instance created above.And jenkins by default runs on the port 8080.

➢ *<public-ipv4>:8080*

- Now use the command to unlock jenkins to access it

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

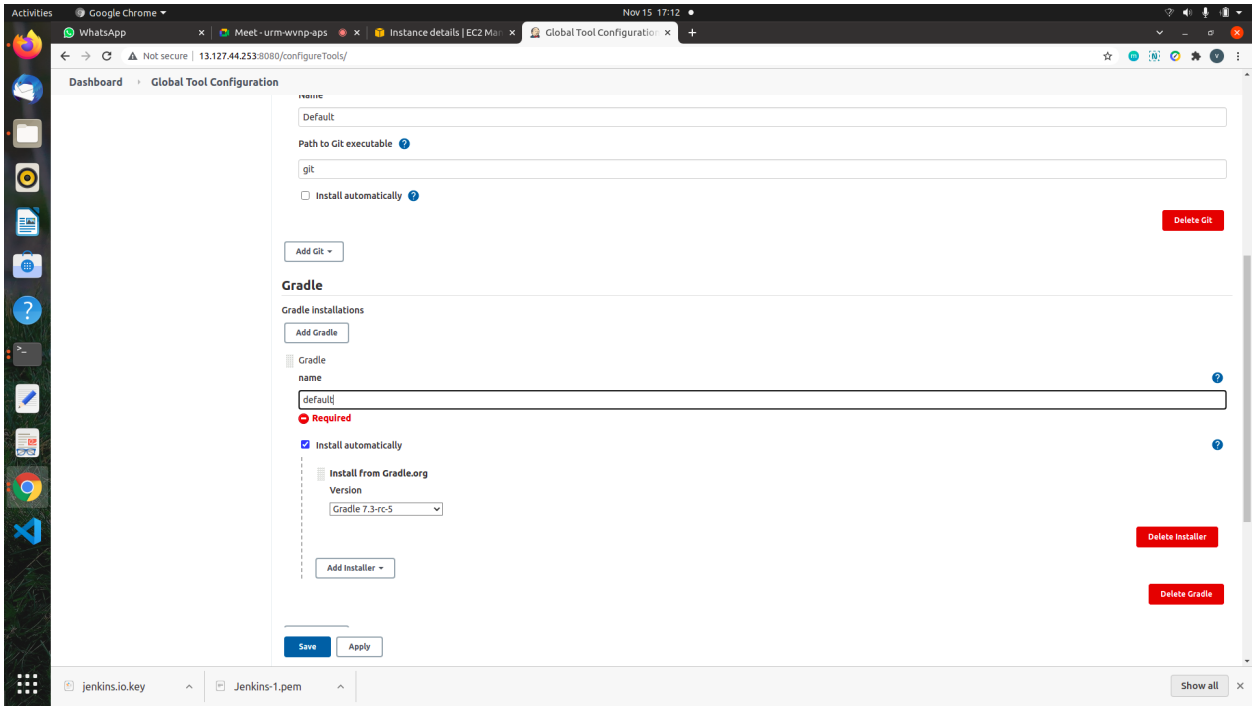




- Opt for install suggested plugin. After completing the installation of the suggested plugin you need to set the First Admin User for Jenkins. Also, check the instance configuration because it will be used for accessing the Jenkins. Now jenkins is ready to be used

Setup Gradle

- For setting up the gradle Goto -> Manage Jenkins -> Global Tool Configuration -> Gradle



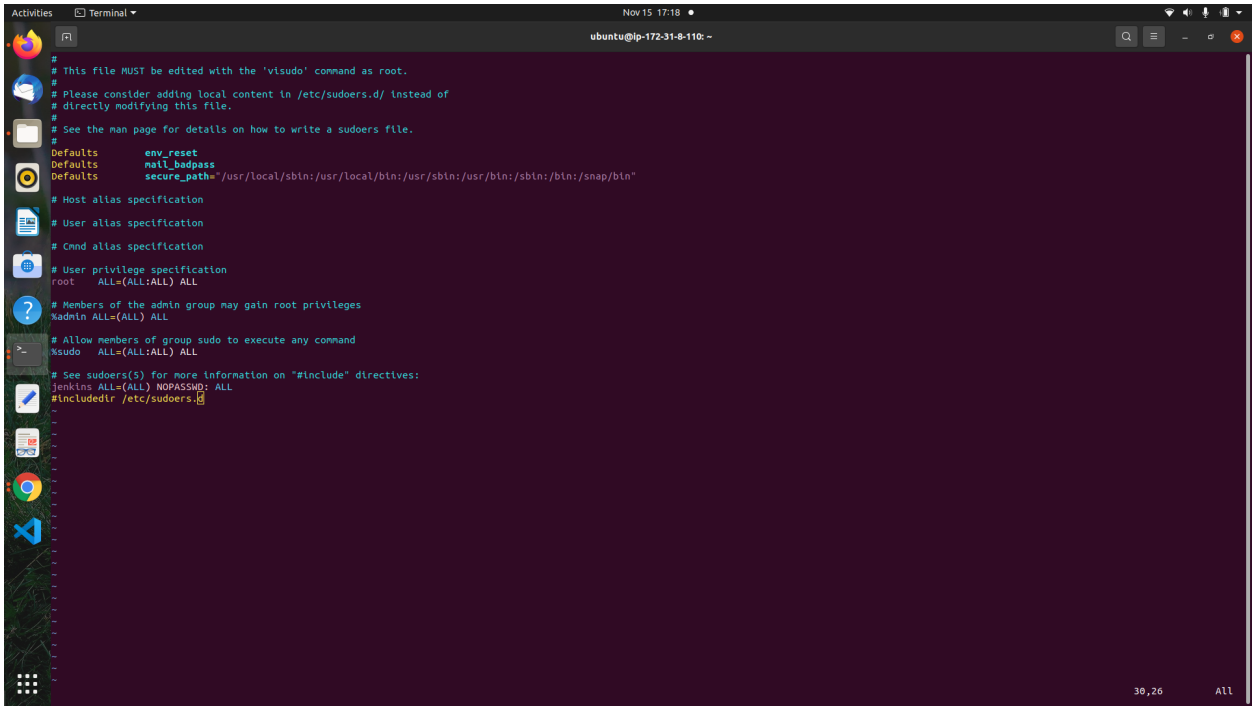
Update visudo and assign administration privileges to jenkins user

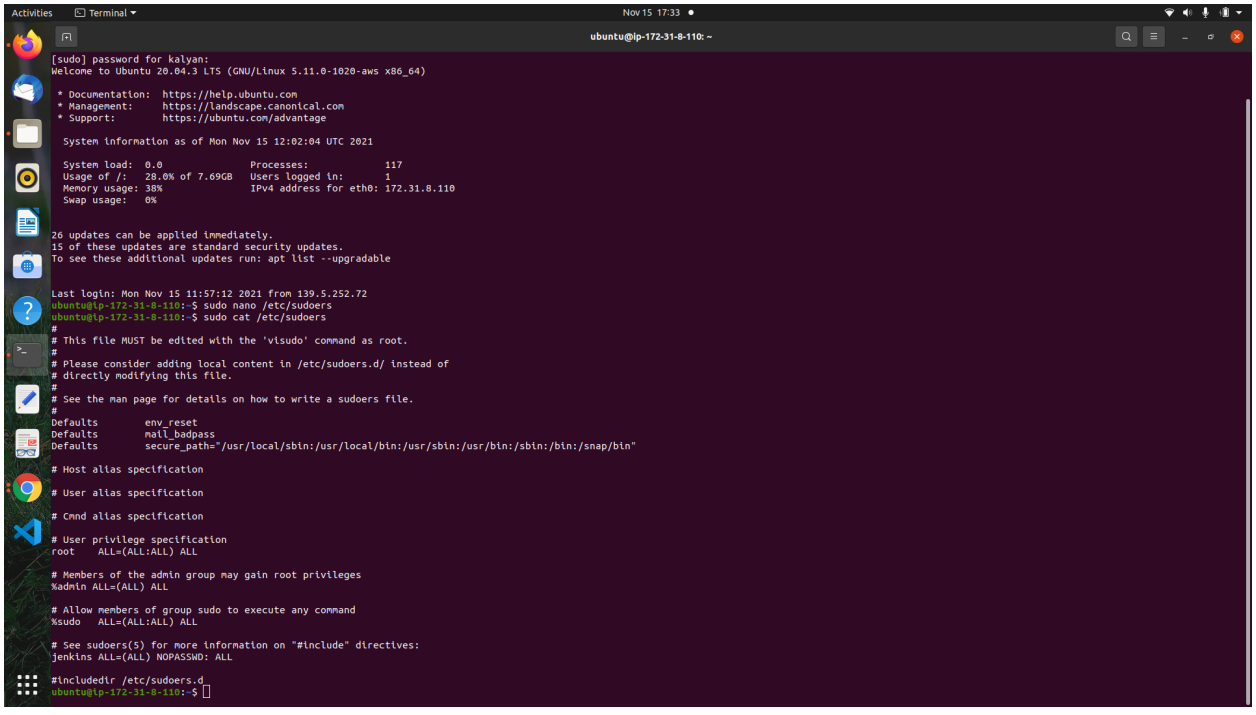
- To interact with the Kubernetes cluster Jenkins will be executing the shell script with the Jenkins user, so the Jenkins user should have an administration(superuser) role assigned forehand. Let's add jenkins user as an administrator and also ass NOPASSWD so that during the pipeline run it will not ask for root password. Open the file /etc/sudoers in vi mode.

➢ *sudo vi /etc/sudoers*

- Add the following line at the end of the file

➢ *jenkins ALL=(ALL) NOPASSWD: ALL*


Activities
Terminal
Nov 15 17:18
ubuntu@ip-172-31-8-110: ~
#
This file MUST be edited with the 'visudo' command as root.
#
Please consider adding local content in /etc/sudoers.d/ instead of
directly modifying this file.
#
See the man page for details on how to write a sudoers file.
#
Defaults env_reset
Defaults mail_badpass
Defaults secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin"
Host alias specification
User alias specification
Cmnd alias specification
User privilege specification
root ALL=(ALL:ALL) ALL
Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL
Allow members of group sudo to execute any command
%sudo ALL=(ALL:ALL) ALL
See sudoers(5) for more information on "#include" directives:
jenkins ALL=(ALL) NOPASSWD: ALL
#includedir /etc/sudoers.d
30,26 All


Activities
Terminal
Nov 15 17:33
ubuntu@ip-172-31-8-110: ~
[sudo] password for kalyan:
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.11.0-1020-aws x86_64)
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage
System information as of Mon Nov 15 12:02:04 UTC 2021
System load: 0.0 Processes: 117
Usage of /: 28.0% of 7.69GB Users logged in: 1
Memory usage: 38% IPv4 address for eth0: 172.31.8.110
Swap usage: 0%
26 updates can be applied immediately.
15 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable
Last login: Mon Nov 15 11:57:12 2021 from 139.5.252.72
ubuntu@ip-172-31-8-110:~$ sudo nano /etc/sudoers
ubuntu@ip-172-31-8-110:~$ sudo cat /etc/sudoers
#
This file MUST be edited with the 'visudo' command as root.
#
Please consider adding local content in /etc/sudoers.d/ instead of
directly modifying this file.
#
See the man page for details on how to write a sudoers file.
#
Defaults env_reset
Defaults mail_badpass
Defaults secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin"
Host alias specification
User alias specification
Cmnd alias specification
User privilege specification
root ALL=(ALL:ALL) ALL
Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL
Allow members of group sudo to execute any command
%sudo ALL=(ALL:ALL) ALL
See sudoers(5) for more information on "#include" directives:
jenkins ALL=(ALL) NOPASSWD: ALL
#includedir /etc/sudoers.d
ubuntu@ip-172-31-8-110:~$

- After adding the line save and quit the file. Now we can use Jenkins as root user and for that run the following command

➢ *sudo su - jenkins*

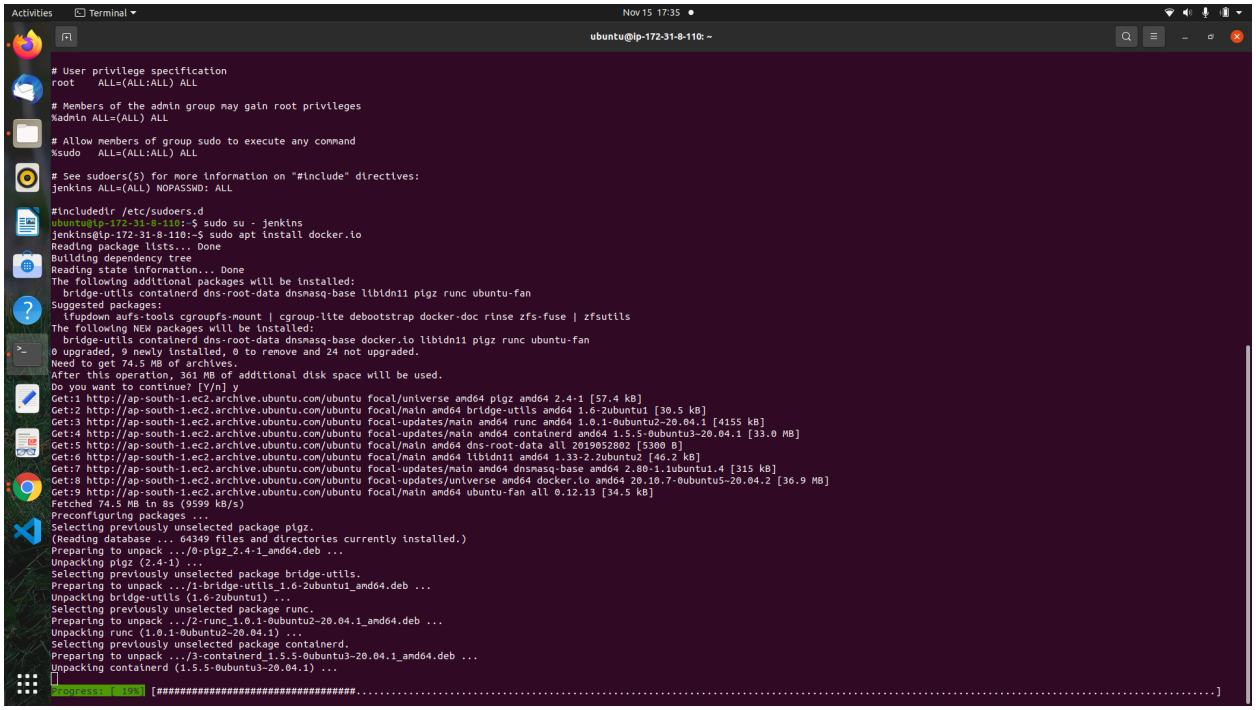
Install Docker

- Now we need to install the docker after installing the Jenkins. The docker installation will be done by the Jenkins user because now it has root user privileges.
- Use the following command for installing the docker

➢ *sudo apt install docker.io*

- After installing the docker you can verify it by simply typing the docker --version onto the terminal. It should return you with the latest version of the docker. Jenkins will be accessing the Docker for building the application Docker images, so we need to add the Jenkins user to the docker group.

➢ *sudo usermod -aG docker jenkins*



Install and Setup AWS CLI

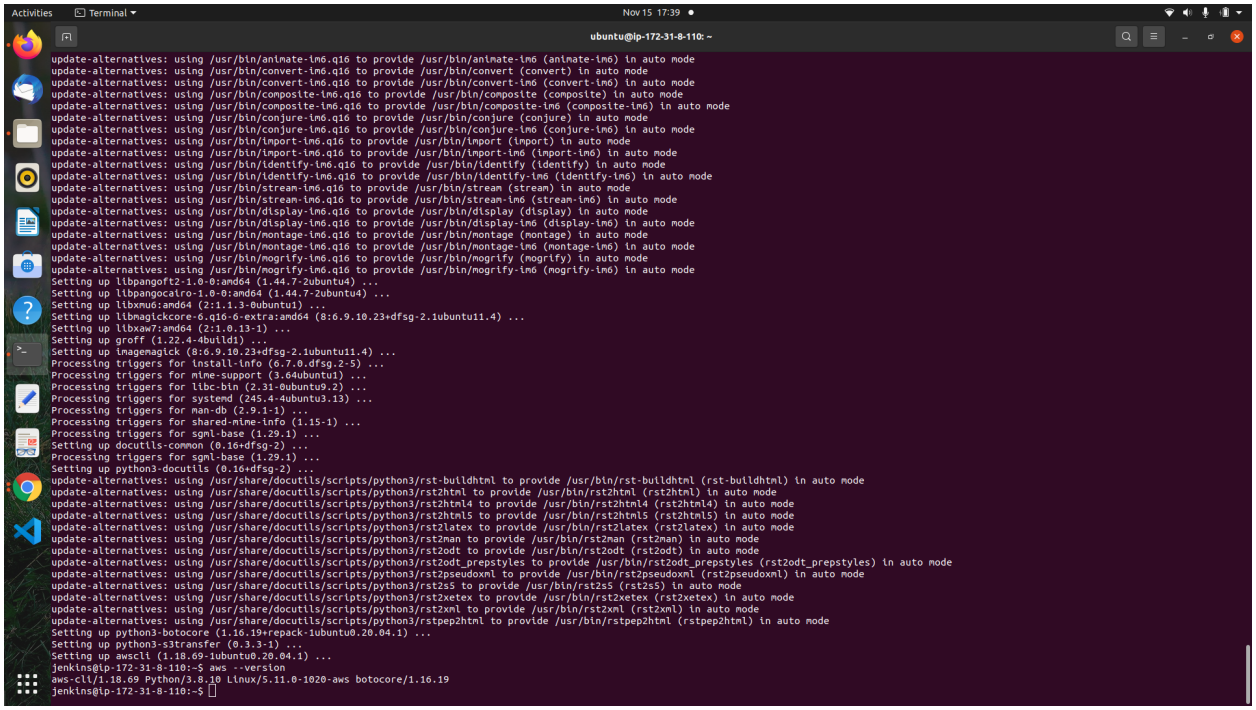
- Now we have our EC2 machine and Jenkins installed. Now we need to set up the AWS CLI on the EC2 machine so that we can use eksctl in the later stages. Let us get the installation done for AWS CLI

➢ *sudo apt install awscli*

- Verify your AWS CLI installation by running the following command

➢ *aws --version*

- It should return you with the version of CLI

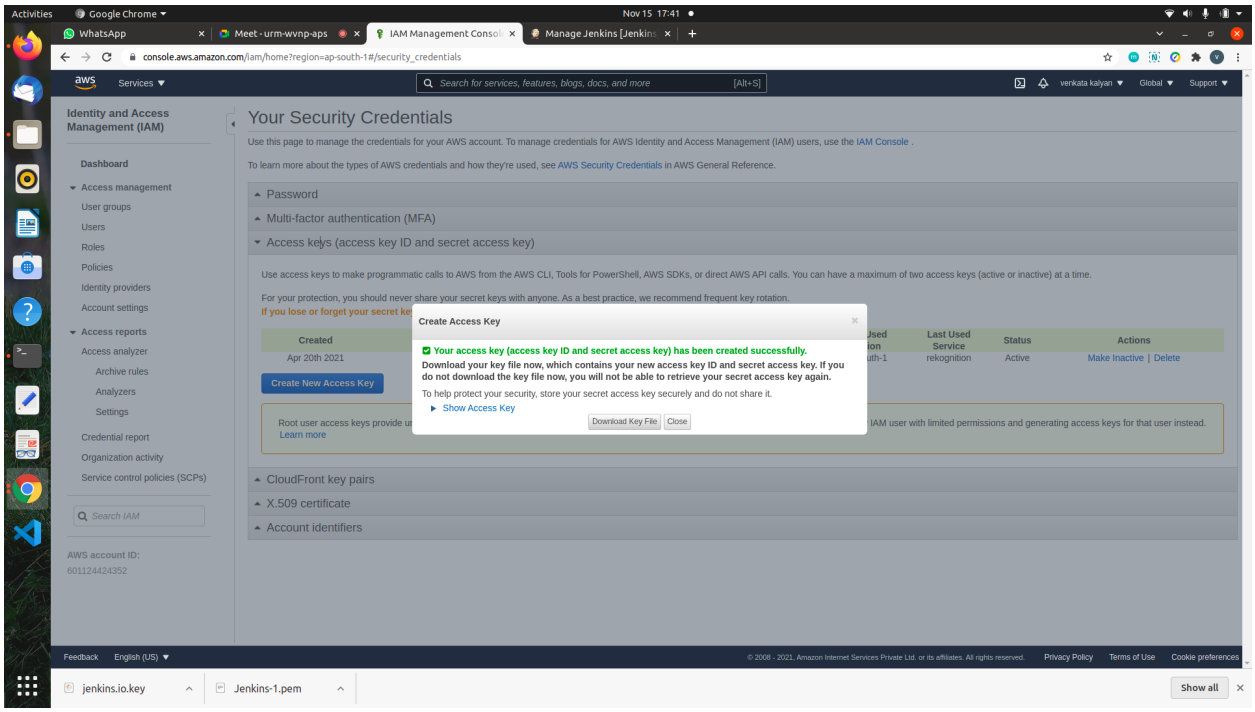


Configure AWS CLI

- Now after installing the AWS CLI, let's configure the AWS CLI so that it can authenticate and communicate with the AWS environment. To configure the AWS the first command we are going to run is:

➢ *aws configure*

- Once you execute the above command it will ask for the following information:
    - AWS Access Key ID [None]:
    - AWS Secret Access Key [None]:
    - Default region name [None]:
    - Default output format [None]:
- You can find this information by going into AWS -> My Security Credentials. Then navigate to Access Keys (access key ID and secret access key). You can click on the Create New Access Key and it will let you generate - AWS Access Key ID, AWS Secret Access Key. Default region name - You can find it from the menu.



- Alright now we have installed and set up AWS CLI.

Install and Setup Kubectl

- Moving forward now we need to set up the kubectl also onto the EC2 instance where we set up the Jenkins in the previous steps. Here is the command for installing kubectl:

➢ *curl -LO "https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux/amd64/kubectl"*

➢ *chmod +x ./kubectl*

➢ *sudo mv ./kubectl /usr/local/bin*

- Verify the kubectl installation by running the command kubectl version and you should see the following output

➢ *Client Version: version.Info{Major:"1", Minor:"21", GitVersion:"v1.21.2", GitCommit:"092fbfbf53427de67cac1e9fa54aaa09a28371d7", GitTreeState:"clean", BuildDate:"2021-06-16T12:59:11Z", GoVersion:"go1.16.5", Compiler:"gc", Platform:"linux/amd64"}*

➢ *Error from server (Forbidden): <html><head><meta http-equiv='refresh' content='1;url=/login?from=%2Fversion%3Ftimeout%3D32s'/><script>window.location.replace('/login?from=%2Fversion%3Ftimeout%3D32s');</script></head><body style='background-color:white; color:white;'>*

Install and Setup eksctl

- The next thing which we are gonna do is to install the eksctl, which we will be using to create AWS EKS Clusters. Okay, the first command which we are gonna run to install the eksctl

➢ *curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp*

➢ *BASH*

➢ *sudo mv /tmp/eksctl /usr/local/bin*

- Verify the installation by running the command

➢ *eksctl version*

- In all the previous steps we were preparing our AWS environment. Now in this step, we are going to create EKS cluster using eksctl.

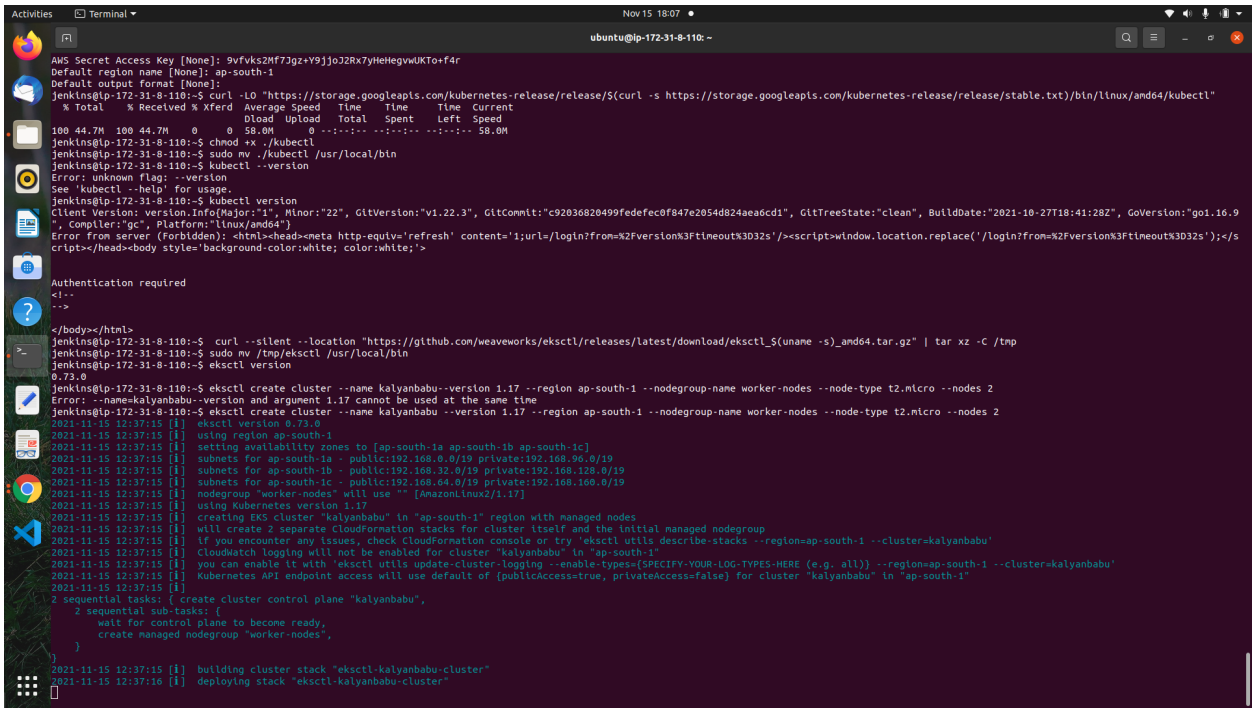
- You need the following in order to run the eksctl command

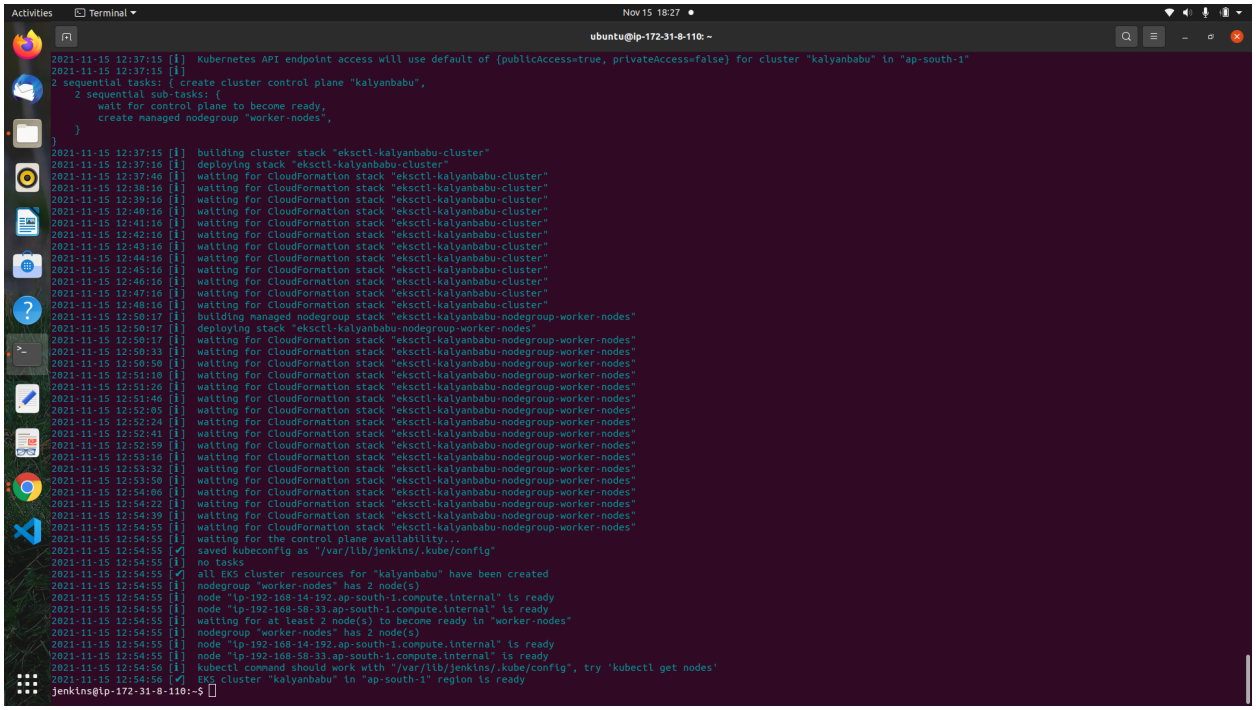
    - Name of the cluster : –name jhooq-test-cluster

- Version of Kubernetes : –version 1.17
- Region : –name eu-central-1
- Nodegroup name/worker nodes : worker-nodes
- Node Type : t2.micro
- Number of nodes: -nodes 2

- Here is the eksctl command

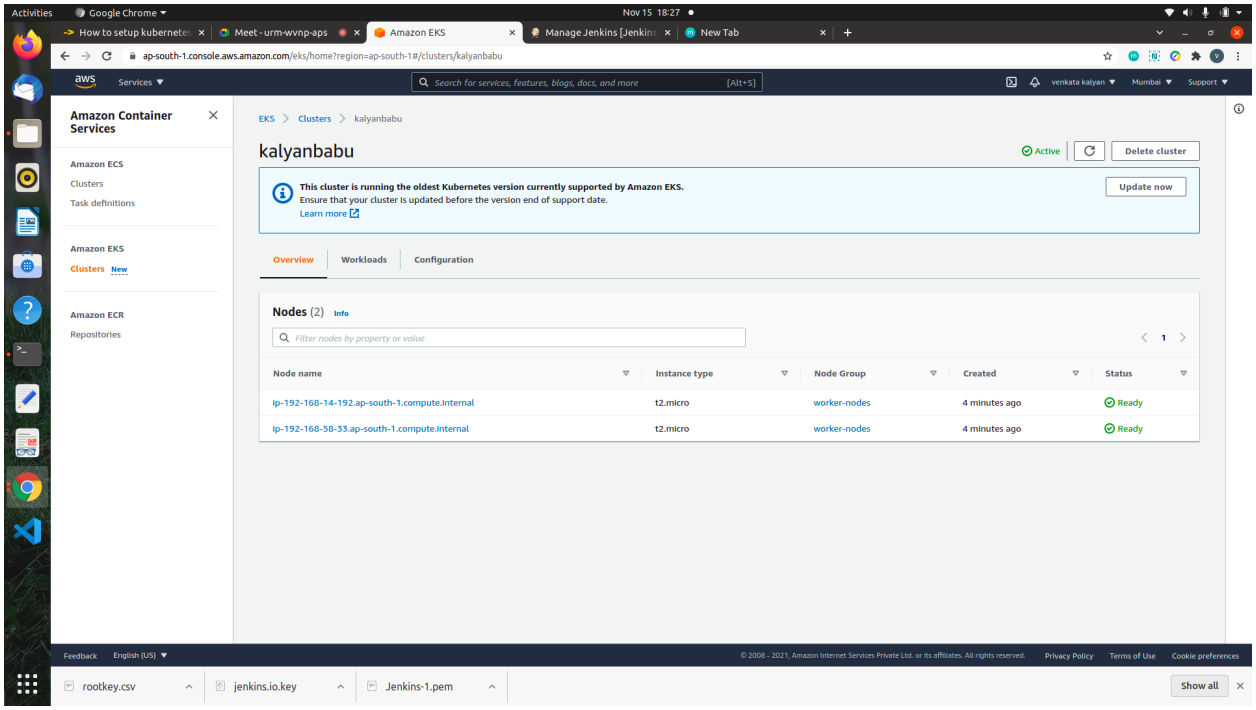
➢ eksctl create cluster --name jhooq-test-cluster --version 1.17 --region eu-central-1 --nodegroup-name worker-nodes --node-type t2.micro --nodes 2





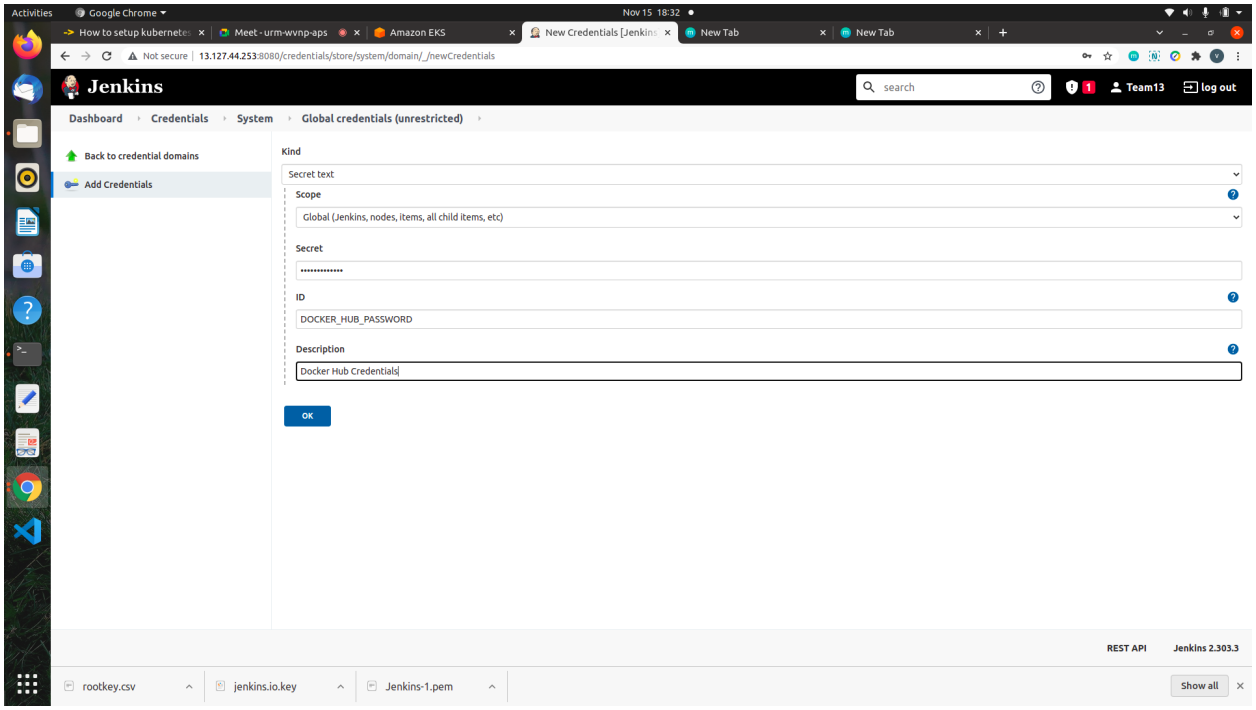
Verify the EKS kubernetes cluster from AWS

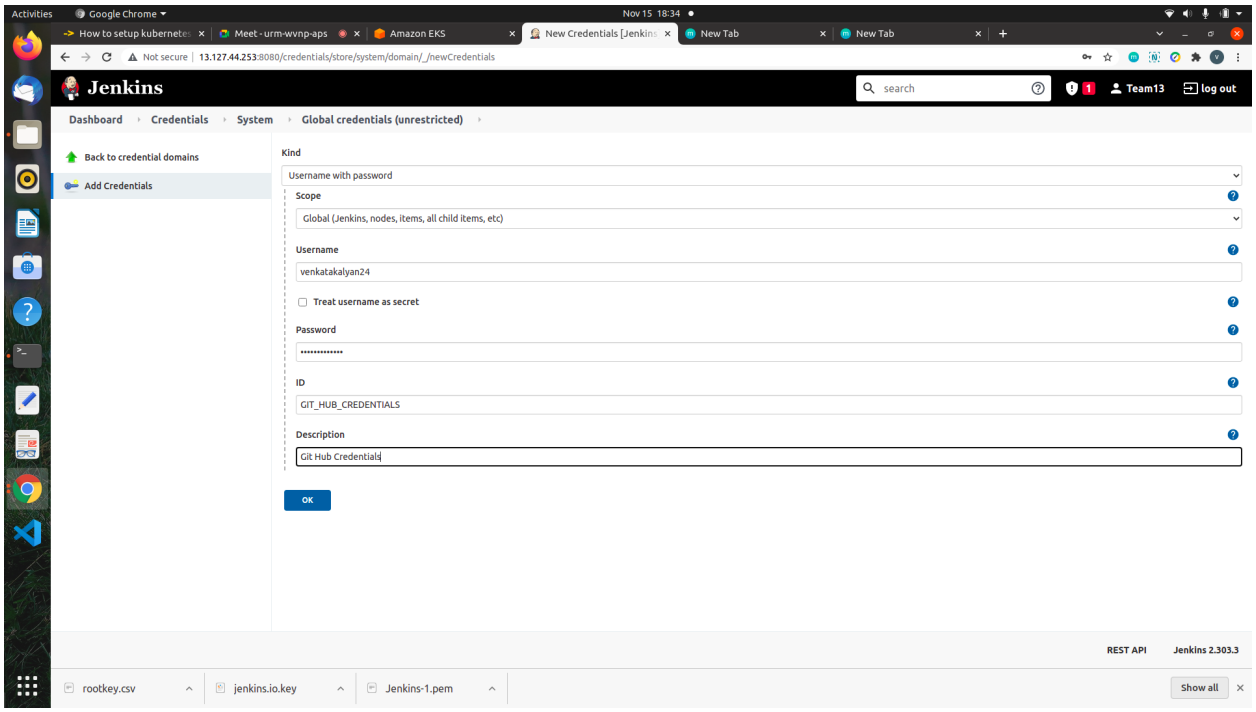
- You can go back to your AWS dashboard and look for Elastic Kubernetes Service -> Clusters. Click on the Cluster Name to verify the worker nodes.



Add Docker and GitHub Credentials into Jenkins

- Kubernetes is a container orchestration tool and container management we are using docker. so if you are reading this line then I am assuming you have a DockerHub Account and GitHub Account. Here is the link of GitHub Repository for this project. You can set the docker credentials by going into : Goto -> Jenkins -> Manage Jenkins -> Manage Credentials -> Stored scoped to jenkins -> global -> Add Credentials.
- Now we add one more username and password for GitHub. Goto -> Jenkins -> Manage Jenkins -> Manage Credentials -> Stored scoped to jenkins -> global -> Add Credentials.


Activities
Google Chrome
Nov 15 18:32
How to setup kubernetes
Meet - urm-wvnp-aps
Amazon EKS
New Credentials [Jenkins
New Tab
New Tab
Not secure | 13.127.44.253:8080/credentials/store/system/domain/_/newCredentials
Jenkins
search
Team13
log out
Dashboard
Credentials
System
Global credentials (unrestricted)
Back to credential domains
Add Credentials
Kind
Secret text
Scope
Global (Jenkins, nodes, items, all child items, etc)
Secret
ID
DOCKER_HUB_PASSWORD
Description
Docker Hub Credentials
OK
REST API
Jenkins 2.303.3
rootkey.csv
jenkins.io.key
Jenkins-1.pem
Show all


Activities
Google Chrome
Nov 15 18:34
How to setup kubernetes
Meet - urm-wvnp-aps
Amazon EKS
New Credentials [Jenkins
New Tab
New Tab
Not secure | 13.127.44.253:8080/credentials/store/system/domain/_/newCredentials
Jenkins
search
Team13
log out
Dashboard
Credentials
System
Global credentials (unrestricted)
Back to credential domains
Add Credentials
Kind
Username with password
Scope
Global (Jenkins, nodes, items, all child items, etc)
Username
venkatakalyan24
Treat username as secret
Password
ID
GIT_HUB_CREDENTIALS
Description
Git Hub Credentials
OK
REST API
Jenkins 2.303.3
rootkey.csv
jenkins.io.key
Jenkins-1.pem
Show all

Add jenkins stages

Okay, now we can start writing out the Jenkins pipeline for deploying the Spring Boot Application into the Kubernetes Cluster. Jenkins stage-1 : Checkout the GitHub Repository.

➢ *stage("Git Clone"){*
➢
➢ *git credentialsId: 'GIT_HUB_CREDENTIALS', url: 'https://github.com/rahulwagh/k8s-jenkins-aws'*
➢ *}*

- Jenkins stage-2 : Gradle compilation and build

➢ *stage('Gradle Build') {*
➢ *sh './gradlew build'*
➢ *}*

- Jenkins stage-3 : Create Docker Container and push to Docker Hub. After successful compilation and build let's create a Docker image and push to the docker hub.

➢ *stage("Docker build"){*
➢ *sh 'docker version'*
➢ *sh 'docker build -t jhooq-docker-demo .'*
➢ *sh 'docker image list'*
➢ *sh 'docker tag jhooq-docker-demo rahulwagh17/jhooq-docker-demo:jhooq-docker-demo'*
➢ *}*
➢
➢ *stage("Push Image to Docker Hub"){*
➢ *sh 'docker push rahulwagh17/jhooq-docker-demo:jhooq-docker-demo'*
➢ *}*

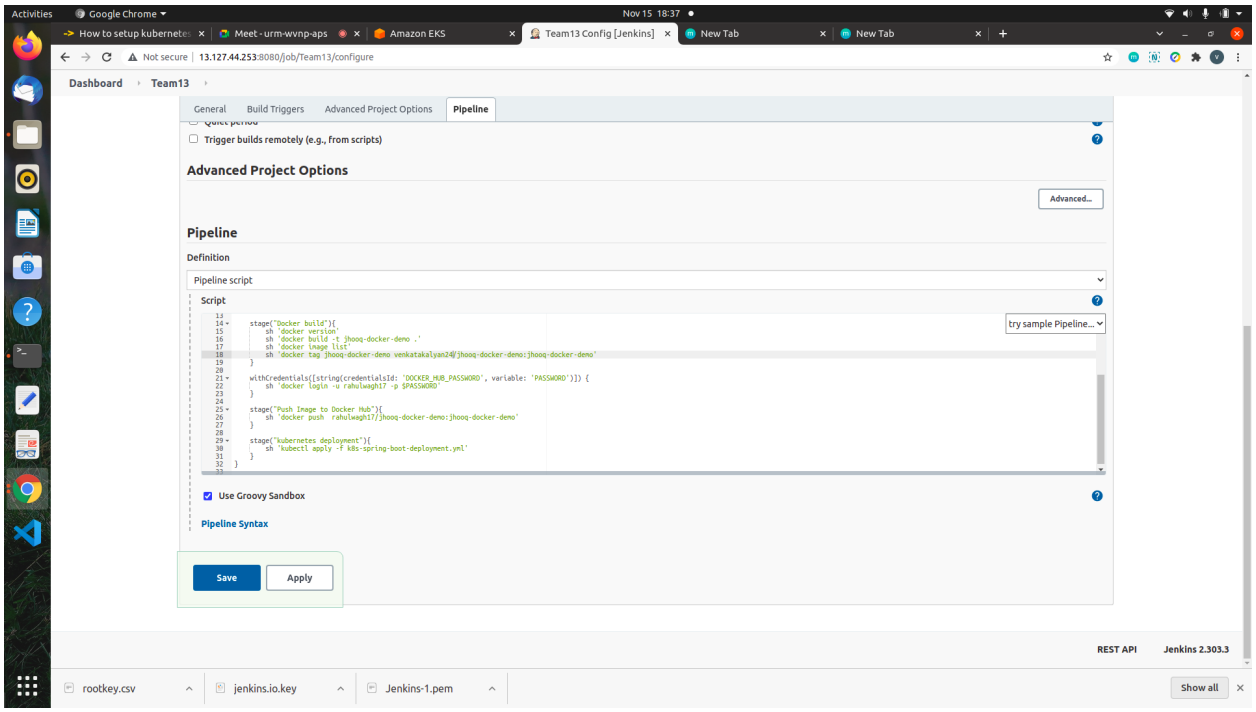
- Jenkins stage-4 : Kubernetes deployment
- Finally, do the Kubernetes deployment

```
➢ stage("kubernetes deployment"){
➢   sh 'kubectl apply -f k8s-spring-boot-deployment.yml'
➢ }
```

- Here is the complete final script for Jenkins pipeline

```
➢ node {
➢
➢     stage("Git Clone"){
➢
➢         git credentialsId: 'GIT_HUB_CREDENTIALS', url:
    'https://github.com/rahulwagh/k8s-jenkins-aws'
➢     }
➢
➢      stage('Gradle Build') {
➢
➢        sh './gradlew build'
➢
➢     }
➢
➢     stage("Docker build"){
➢         sh 'docker version'
➢         sh 'docker build -t jhooq-docker-demo .'
➢         sh 'docker image list'
➢         sh 'docker tag jhooq-docker-demo
    rahulwagh17/jhooq-docker-demo:jhooq-docker-demo'
➢     }
➢
➢     withCredentials([string(credentialsId:
    'DOCKER_HUB_PASSWORD', variable: 'PASSWORD')]) {
➢         sh 'docker login -u rahulwagh17 -p $PASSWORD'
➢     }
```

➢
➢ *stage("Push Image to Docker Hub"){*
➢ *sh 'docker push rahulwagh17/jhooq-docker-demo:jhooq-docker-demo'*
➢ *}*



Build, deploy and test CI/CD pipeline

- Create new Pipeline: Goto Jenkins Dashboard or Jenkins home page click on New Item.
- Pipeline Name: Now enter Jenkins pipeline name and select Pipeline
- Add pipeline script: Goto -> Configure and then pipeline section. Copy the Jenkins script from Step 12 and paste it there. Build and Run Pipeline: Now goto pipeline and click on build now. Verify the
- build status:

Pipeline test-pipeline

Recent Changes

Stage View

	Git Clone	Gradle Build	Docker build	Push Image to Docker Hub	kubernetes deployment
Average stage times: (Average full run time: ~31s)	1s	19s	3s	4s	1s
#4 Jun 23 22:58 No Changes	322ms	8s	1s	2s	1s

Verify using kubectl commands

- You can also verify the Kubernetes deployment and service with kubectl command .e.g kubectl get deployments, kubectl get service. You can access the rest end point from browser using the EXTERNAL-IP address.