



INDIAN INSTITUTE OF
INFORMATION
TECHNOLOGY

DevOps(DS457)

Assignment - 1

Jenkins CICD pipeline for Kubernetes

Submitted to
Dr.Uma.S

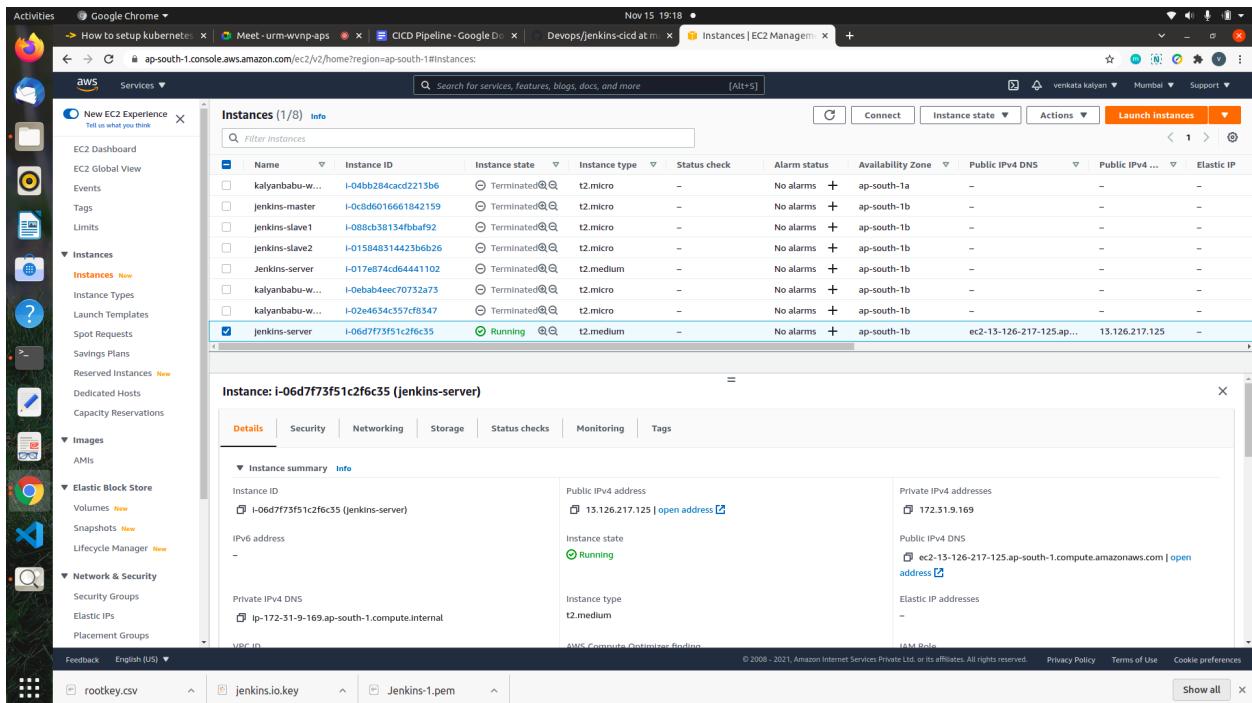
Submitted by
Team13

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.



```

Activities Terminal Nov 15 16:58 •
ubuntu@ip-172-31-8-110: ~
Adding debian:AffirmTrust_Networking.pem
Adding debian:ACVRAIZ1.pem
Adding debian:Hongkong_Post_Root_CA_3.pem
Adding debian:GTS_Root_R2.pem
Adding debian:GlobalSign_Root_CA_R5.pem
Adding debian:Trustwave_Global_ECC_P256_Certification_Authority.pem
Adding debian:Quovadis_Root_CA_1_G1.pem
Adding debian:COMODO_ECC_Certification_Authority.pem
Adding debian:SZAFIR_ROOT_CA2.pem
Adding debian:Sonera_Class_2_Root_CA.pem
Adding debian:GlobalSign_Root_GC_CA.pem
Adding debian:Go_Daddy_Root_Certificate_Authority_G2.pem
Adding debian:Quovadis_Root_CA_3.pem
Adding debian:Chambers_of_Commerce_Root_-2008.pem
Adding debian:SwissSign_Gold_CA_-G2.pem
Adding debian:cetnrtion_ROOT_CA.pem
Adding debian:DigiCert_RSA_Certification_Authority.pem
Adding debian:OpenDigiCert_Root_CA.pem
Adding debian:Entrust_Root_Certification_Authority.pem
Adding debian:EC-ACC.pem
Adding debian:Entrust_Root_Certification_Authority_G2.pem
Adding debian:CFCA_EV_ROOT.pem
Adding debian:GlobalSign_ECC_Root_CA_2009.pem
Adding debian:SSL.com_EV_Root_Certification_Authority_ECC.pem
Adding debian:SSL.com_EV_Root_Certification_Authority_RSA_B2.pem
Adding debian:Quovadis_Root_CA_3_G3.pem
Adding debian:Amazon_Root_CA_3.pem
Adding debian:DigiCert_Assured_ID_Root_CA.pem
Adding debian:OpenSSL_Light.pem
Adding debian:TWCAC_Global_Root_CA.pem
Adding debian:AC_RAIZ_FNWT-RCM.pem
done.
Processing triggers for libc-bin (2.31-0ubuntu9.2) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for ca-certificates (20210119-20.04.2) ...
Updating certificates in /etc/ssl/certs...
0 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...
done.
done.
Setting up openjdk-11-jre-headless:amd64 (11.0.11+9-Ubuntu2-28.04) ...
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/java to provide /usr/bin/java (java) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jjs to provide /usr/bin/jjs (jjs) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/keytool to provide /usr/bin/keytool (keytool) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/rmid to provide /usr/bin/rmid (rmid) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/rmiregistry to provide /usr/bin/rmiregistry (rmiregistry) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/pack200 to provide /usr/bin/pack200 (pack200) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/unpack200 to provide /usr/bin/unpack200 (unpack200) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jexec to provide /usr/bin/jexec (jexec) in auto mode
ubuntutip.172-31-8-110: ~ $ java --version
openjdk 11.0.11+9-Ubuntu2-28.04
OpenJDK 64-Bit Server VM (build 11.0.11+9-Ubuntu2-28.04, mixed mode, sharing)
ubuntutip.172-31-8-110: ~ $ 
```

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
```

```
Activities Terminal Nov 15 17:02 ● ubuntugui@ip-172-31-8-110:~
[1] https://pkg.jenkins.io/debian-stable binary/ Release
[1] https://security.ubuntu.com/ubuntu focal-security InRelease
[1] https://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal InRelease
[1] https://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal-updates InRelease
[1] https://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal-backports InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
 daemon net-tools
 The following packages will be installed:
 daemon Jenkins net-tools
 0 upgraded, 3 newly installed, 0 to remove and 24 not upgraded.
 Need to get 72.2 MB of additional disk space will be used.
Do you want to continue? [Y/n]
Get: https://pkg.jenkins.io/debian-stable binary/ jenkins 2.303.3 [71.9 kB]
Get: https://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal amd64 net-tools amd64 1:60-g1t20180626.aebd88e-1ubuntu1 [196 kB]
Fetched 72.2 MB in 1min 19s (917 kB/s)
Selecting previously unselected package daemon.
(Reading database ... 6428 files and directories currently installed.)
Unpacking daemon (0.6.4-1build2) ...
Selecting previously unselected package net-tools.
Preparing to unpack .../net-tools.1:60-g1t20180626.aebd88e-1ubuntu1_amd64.deb ...
Unpacking net-tools (1:60-g1t20180626.aebd88e-1ubuntu1) ...
Preparing to unpack .../jenkins_2.303.3_all.deb ...
Unpacking Jenkins (2.303.3) ...
Setting up net-tools (1:60-g1t20180626.aebd88e-1ubuntu1) ...
Setting up daemon (0.6.4-1build2) ...
Setting up Jenkins (2.303.3) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for systemd (245.4-4ubuntu3.13) ...
jenkins.service - LSB: Start Jenkins at boot time
 Loaded: loaded (/etc/init.d/jenkins; generated)
 Active: active (exited) since Mon 2021-11-15 11:32:04 UTC; 22s ago
 Docs: man/systemd-timedate-generator(8)
 Tasks: 0 (limit: 4760)
 Memory: 0B
 CGroup: /system.slice/jenkins.service

Nov 15 11:32:03 ip-172-31-8-110 systemd[1]: Starting LSB: Start Jenkins at boot time...
Nov 15 11:32:03 ip-172-31-8-110 jenkins[4079]: Correct Java version found
Nov 15 11:32:03 ip-172-31-8-110 jenkins[4079]: Jenkins v2.303.3 - The World's Automation Server jenkins
Nov 15 11:32:03 ip-172-31-8-110 su[4079]: (to jenkins) root on none
Nov 15 11:32:03 ip-172-31-8-110 su[4079]: pam_unix(su-session): session opened for user jenkins by (uid=0)
Nov 15 11:32:04 ip-172-31-8-110 su[4079]: pam_unix(su-session): session closed for user jenkins
Nov 15 11:32:04 ip-172-31-8-110 jenkins[4089]: ...done.
Nov 15 11:32:04 ip-172-31-8-110 systems[1]: Started LSB: Start Jenkins at boot time.

ubuntugui@ip-172-31-8-110:~]$
```

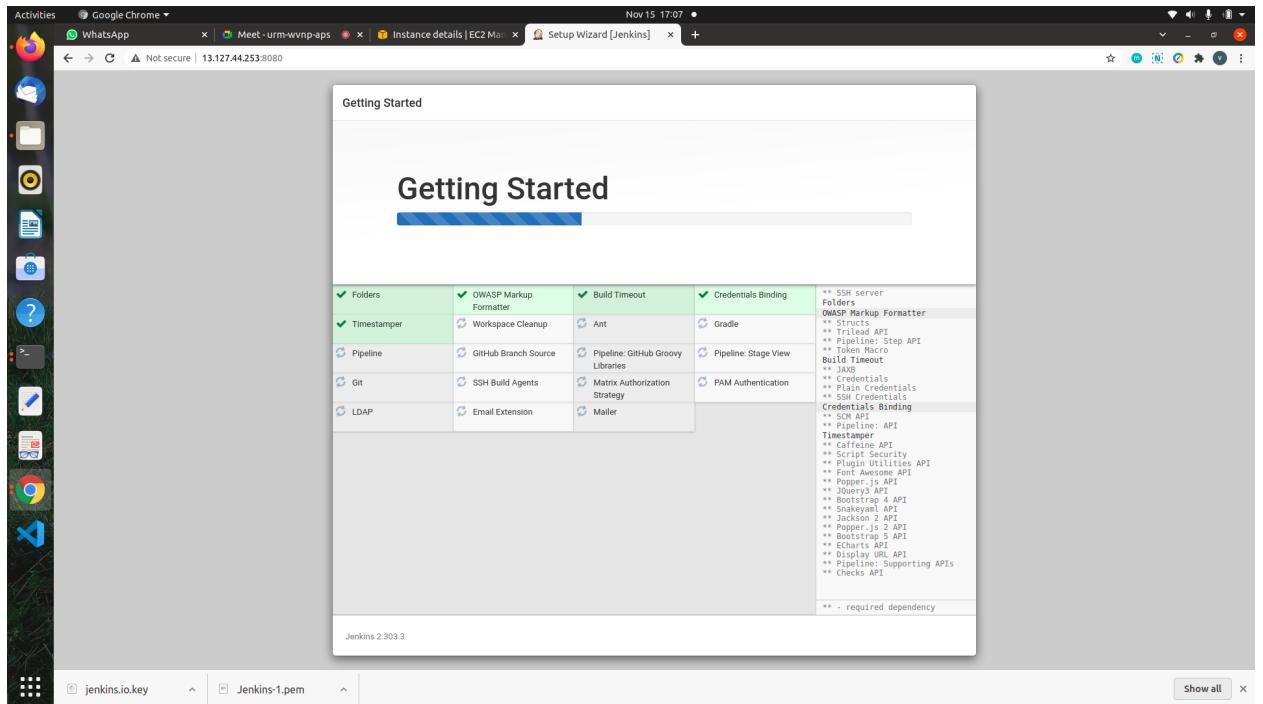
- Now we need to start using jenkins from the public ipv4 address of the instance created above. And jenkins by default runs on the port 8080.

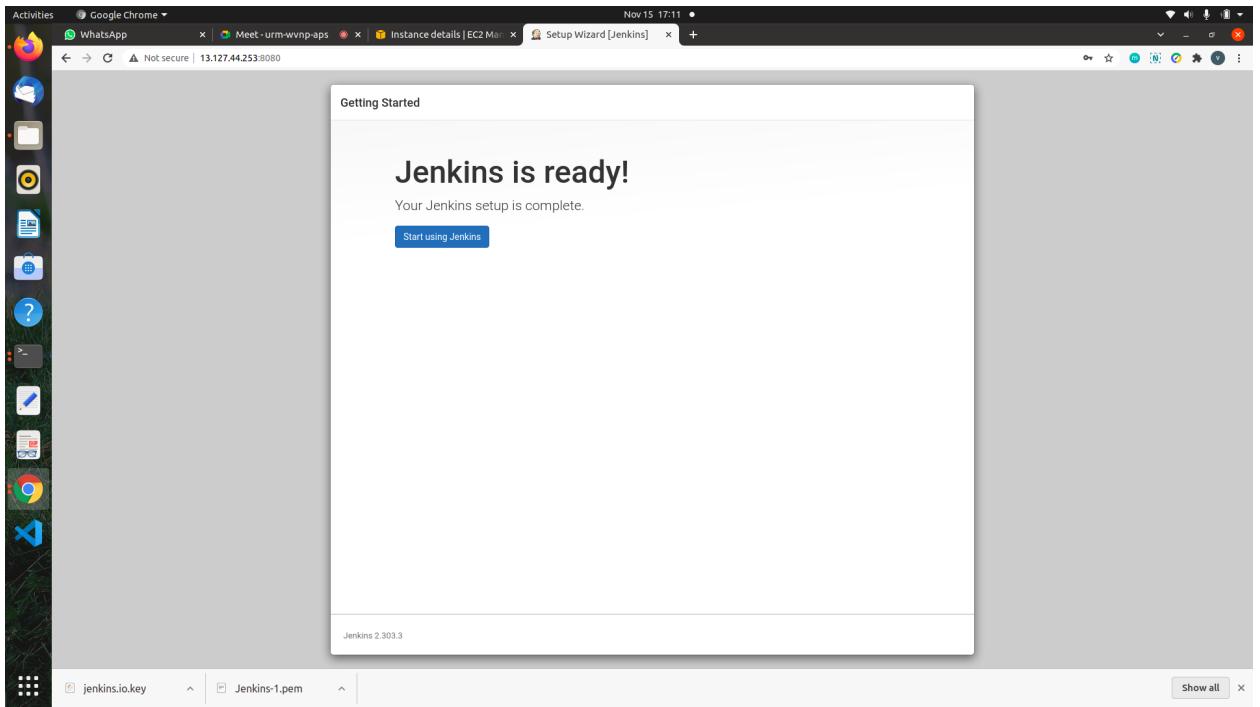
➤ <public-ipv4>:8080

- Now use the command to unlock jenkins to access it

➤ `sudo cat`

`/var/lib/jenkins/secrets/initialAdminPassword`

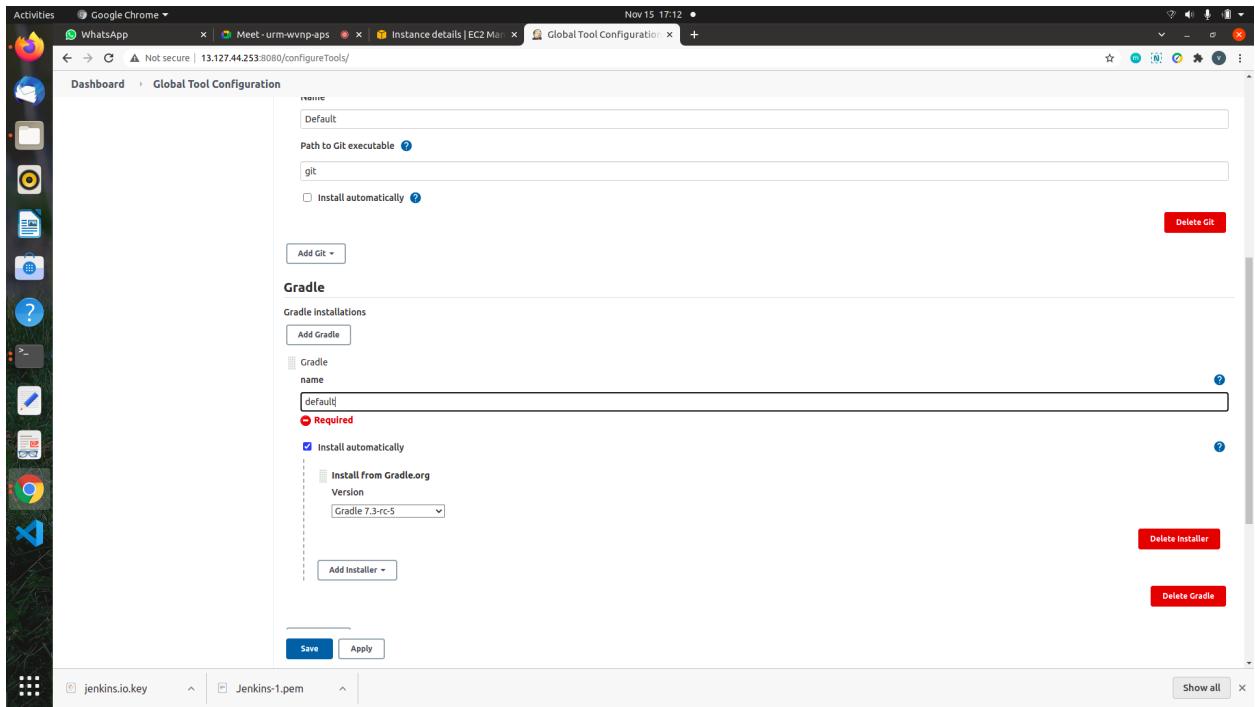




- Opt for install suggested plugin. After completing the installation of the suggested plugin you need to set the First Admin User for Jenkins. Also, check the instance configuration because it will be used for accessing the Jenkins. Now jenkins is ready to be used

## 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 beforehand. 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
#include /etc/sudoers.d/jenkins

38,26     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
#include /etc/sudoers.d/jenkins

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`

```
Activities Terminal Nov 15 17:35 ubuntu@ip-172-31-8-110:~  
  
# 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:~$ sudo su - jenkins  
jenkins@ip-172-31-8-110:~$ sudo apt install docker.io  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following additional packages will be installed:  
  bridge-utils containerd dns-root-data dnsmasq-base liblxdn1 pigz runc ubuntu-fan  
Suggested packages:  
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debbootstrap docker-doc rinse zfs-fuse | zfsutils  
The following NEW packages will be installed:  
  bridge-utils containerd dns-root-data dnsmasq-base docker.io liblxdn1 pigz runc ubuntu-fan  
0 upgraded, 9 newly installed, 0 to remove and 20 not upgraded.  
Need to get 74.5 MB of archives.  
After this operation, 361 MB of additional disk space will be used.  
Do you want to continue? [Y/n] y  
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal/universe amd64 pigz amd64 2.4-1 [57.4 kB]  
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal/main amd64 bridge-utils amd64 1.6-2ubuntu1 [30.5 kB]  
Get:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal-updates/main amd64 runc amd64 1.0.1-ubuntu2-20.04.1 [4155 kB]  
Get:4 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal-updates/main amd64 containerd amd64 1.5.5-ubuntu3-20.04.1 [33.0 kB]  
Get:5 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal/main amd64 dns-root-data all 2019052802 [5300 B]  
Get:6 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal/main amd64 liblxdn1 amd64 1.33-2.2ubuntu2 [46.2 kB]  
Get:7 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal-updates/main amd64 dnsmasq-base amd64 2.88-1.0ubuntu1.4 [315 kB]  
Get:8 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal-updates/universe amd64 docker.io amd64 20.10.7-0ubuntu5-20.04.2 [36.9 MB]  
Get:9 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal/main amd64 ubuntu-fan all 0.12.13 [34.5 kB]  
Fetched 74.5 MB in 8s (9599 kB/s)  
Preconfiguring package ...  
Selecting previously unselected package pigz.  
(Reading database ... 64349 files and directories currently installed.)  
Preparing to unpack .../0-pigz_2.4-1_amd64.deb ...  
Unpacking pigz (2.4-1)  
Selecting previously unselected package bridge-utils.  
Preparing to unpack .../1-bridge-utils_1.6-2ubuntu1_amd64.deb ...  
Unpacking bridge-utils (1.6-2ubuntu1)  
Selecting previously unselected package runc.  
Preparing to unpack .../2-runc_1.0.1-ubuntu2-20.04.1 ...  
Selecting previously unselected package containerd.  
Preparing to unpack .../3-containerd_1.5.5-ubuntu3-20.04.1_amd64.deb ...  
Unpacking containerd (1.5.5-ubuntu3-20.04.1) ...  
[Progress] [ 19% ] [##########################################]
```

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

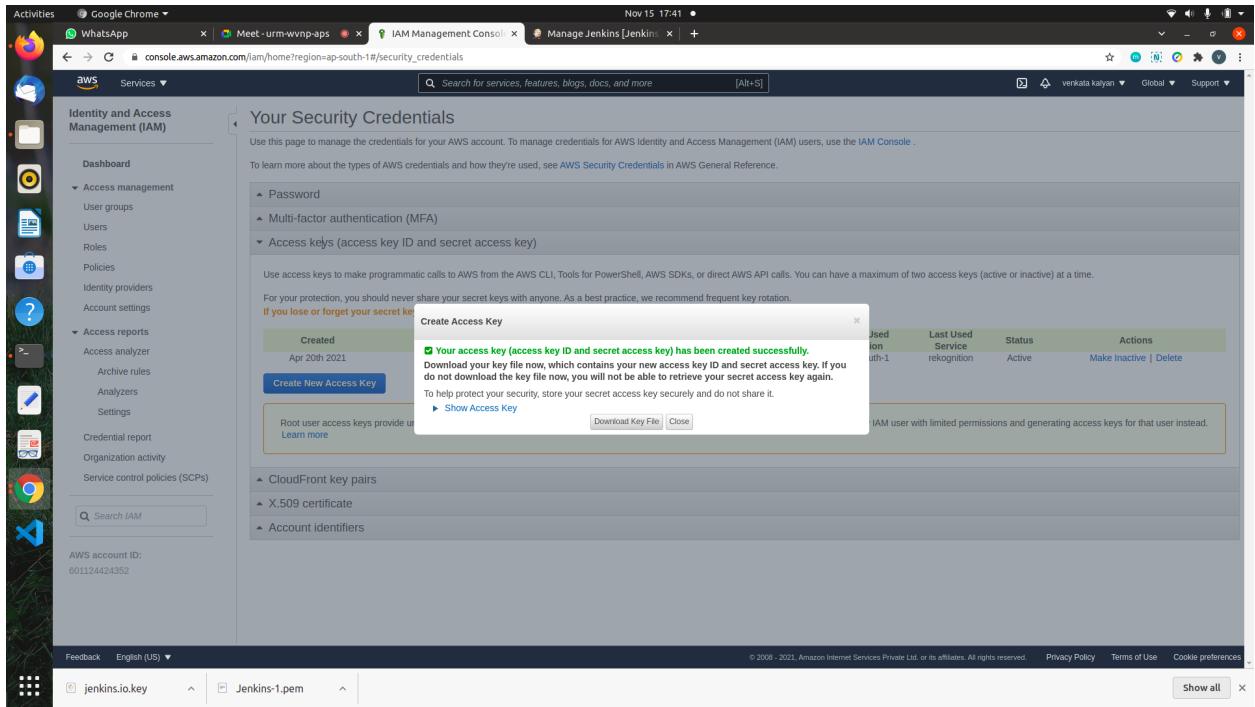
```
➤ sudo apt install awscli
```

- Verify your AWS CLI installation by running the following command
➤ `aws --version`

- It should return you with the version of CEF

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%3D3
  2s'/'><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/eksct
  l_$(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

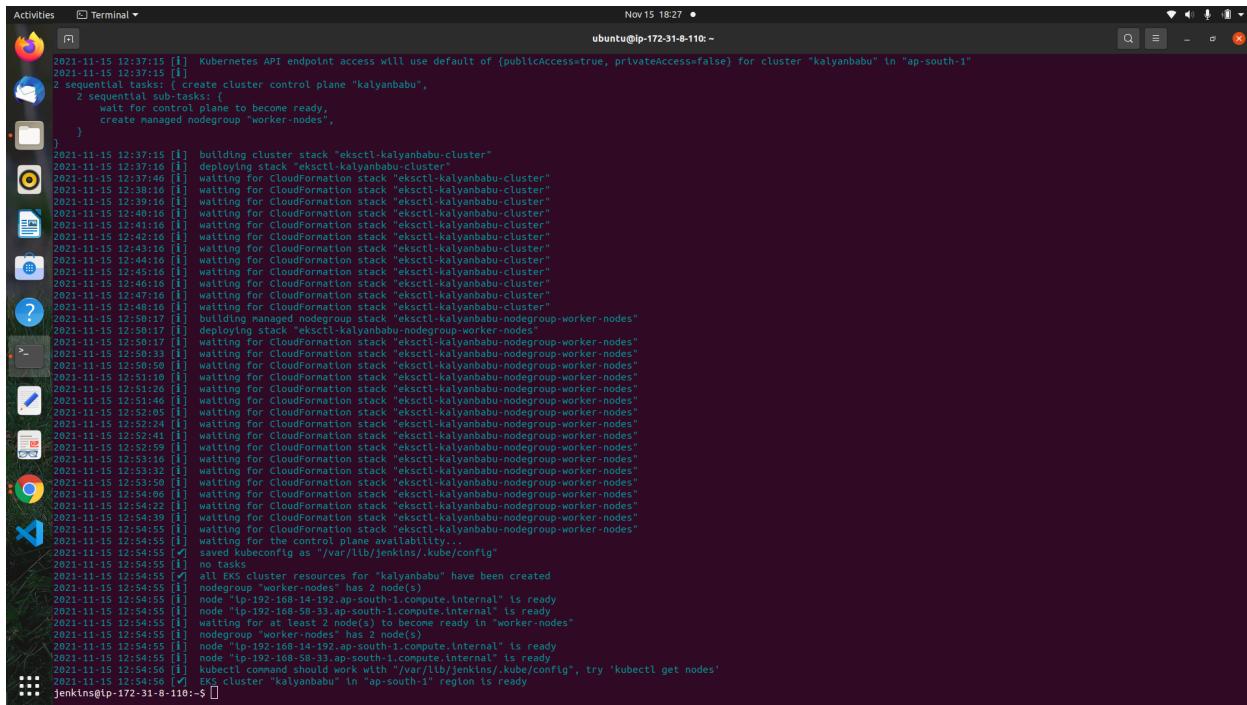
- Version of Kubernetes : –version 1.17
- Region : –name eu-central-1
- Nodegroup name/worker nodes : worker-nodes
- Node Type : t2.micro
- Number of nodes: -nodes 2
- Here is the eksctl command

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

```

Activities Terminal Nov 15 18:07 •
ubuntu@ip-172-31-8-110:~ [1]
AWS Secret Access Key [None]: ap-south-1
Default region name [None]: ap-south-1
Default output format [None]:
jenkins@ip-172-31-8-110:~$ curl -LO "https://storage.googleapis.com/kubernetes-release/release/5(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux/amd64/kubectl"
Default region name [None]: ap-south-1
Default output format [None]:
Total % Received % Xferd Average Speed Time Time Current
%: Total %: Received %: Xferred Average Speed Time: Time: Spent Left Speed
100 44.7M 100 44.7M 0 0 58.0M 0 --:--:--:--:--:--:--:--:--:--:--:58.0M
jenkins@ip-172-31-8-110:~$ chmod +x ./kubectl
jenkins@ip-172-31-8-110:~$ sudo mv ./kubectl /usr/local/bin
jenkins@ip-172-31-8-110:~$ kubectl --version
Error from server (Forbidden): kubectl: help for 'version'.
jenkins@ip-172-31-8-110:~$ kubectl version
Client Version: version.Info{Major:"1", Minor:"22", GitVersion:"v1.22.3", GitCommit:"c92036820499fedefec0f847e2054d824aea6cd1", GitTreeState:"clean", BuildDate:"2021-10-27T18:41:28Z", GoVersion:"go1.16.9", 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');</s
cript></head><body style='background-color:white; color:white;'>
</body></html>
Authentication required
<!--
-->
</html>
jenkins@ip-172-31-8-110:~$ curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp
jenkins@ip-172-31-8-110:~$ sudo mv /tmp/eksctl /usr/local/bin
jenkins@ip-172-31-8-110:~$ eksctl version
0.73.0
jenkins@ip-172-31-8-110:~$ eksctl create cluster --name kalyanbabu --version 1.17 --region ap-south-1 --nodegroup-name worker-nodes --node-type t2.micro --nodes 2
Error: --name=kalyanbabu --version and argument --1.17 cannot be used at the same time
jenkins@ip-172-31-8-110:~$ eksctl create cluster --name kalyanbabu --version 1.17 --region ap-south-1 --nodegroup-name worker-nodes --node-type t2.micro --nodes 2
2021-11-15 12:37:15 [i] eksctl version 0.73.0
2021-11-15 12:37:15 [i] using region ap-south-1
2021-11-15 12:37:15 [i] setting availability zones to [ap-south-1a ap-south-1b ap-south-1c]
2021-11-15 12:37:15 [i] subnets for ap-south-1a - public:192.168.96.0/19
2021-11-15 12:37:15 [i] subnets for ap-south-1b - public:192.168.32.0/19 private:192.168.128.0/19
2021-11-15 12:37:15 [i] subnets for ap-south-1c - public:192.168.64.0/19 private:192.168.160.0/19
2021-11-15 12:37:15 [i] nodegroup 'worker-nodes' will use "" [AmazonLinux2/1.17]
2021-11-15 12:37:15 [i] using Kubernetes version 1.17
2021-11-15 12:37:15 [i] creating EKS cluster "kalyanbabu" in "ap-south-1" region with managed nodes
2021-11-15 12:37:15 [i] will create 2 separate CloudFormation stacks for cluster itself and the initial managed nodegroup
2021-11-15 12:37:15 [i] If you encounter any issues, check CloudFormation console or try `eksctl utils describe-stacks --region=ap-south-1 --cluster=kalyanbabu`
2021-11-15 12:37:15 [i] CloudWatch logging will not be enabled for cluster "kalyanbabu" in "ap-south-1"
2021-11-15 12:37:15 [i] you can enable it with 'eksctl utils update-cluster-logging --enable-types=[SPECIFY-YOUR-LOG-TYPES-HERE (e.g. all)] --region=ap-south-1 --cluster=kalyanbabu'
2021-11-15 12:37:15 [i] Kubernetes API endpoint access will use default of [publicAccess=true, privateAccess=false] for cluster "kalyanbabu" in "ap-south-1"
2021-11-15 12:37:15 [i] 2 sequential tasks:
2 sequential tasks:
  2 sequential sub-tasks:
    wait for control plane to become ready,
    create managed nodegroup "worker-nodes",
  }
}
2021-11-15 12:37:15 [i] building cluster stack "eksctl-kalyanbabu-cluster"
2021-11-15 12:37:15 [i] deploying stack "eksctl-kalyanbabu-cluster"

```

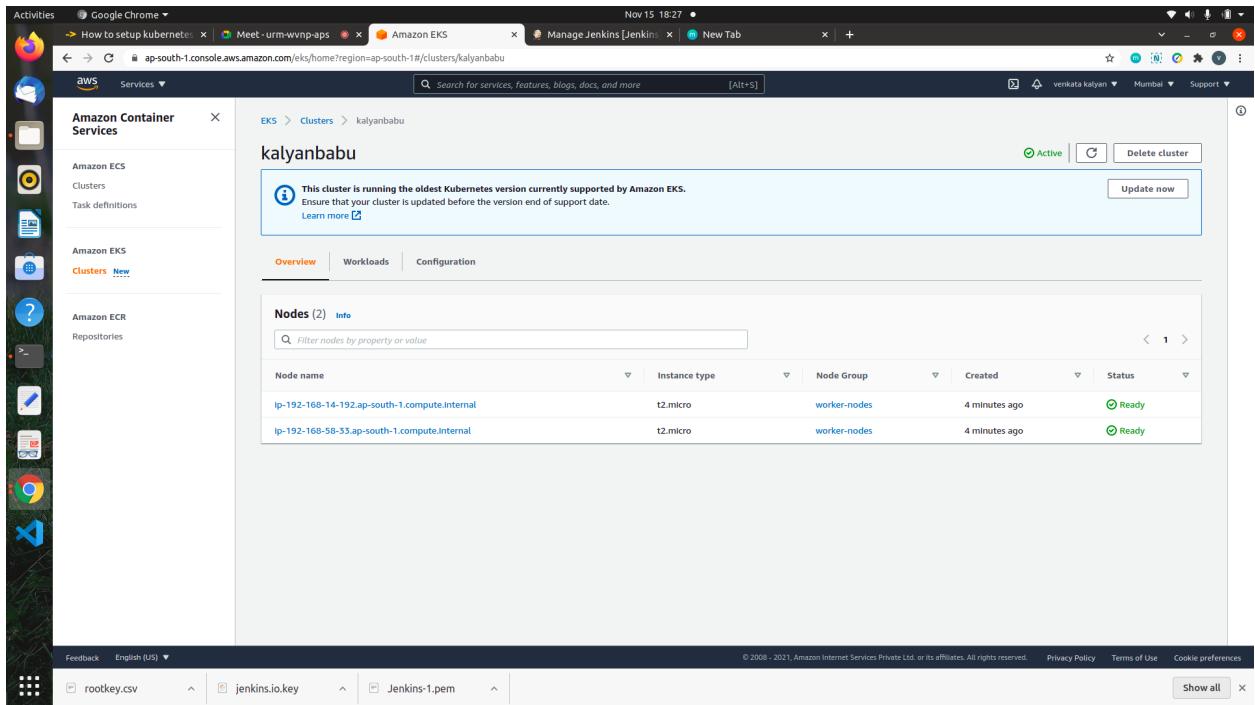


A screenshot of an Ubuntu terminal window titled "Terminal". The window shows a log of command-line output from Jenkins. The log details the creation of an EKS cluster named "kalyanbabu" in the "ap-south-1" region. It includes timestamped entries for building the cluster stack, deploying it, and waiting for various components like the control plane and worker nodes to become ready. The log concludes with a success message indicating the cluster is ready for use.

```
Nov 15 18:27 • ubuntu@ip-172-31-8-110:~  
2021-11-15 12:37:15 [i] Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "kalyanbabu" in "ap-south-1"  
2021-11-15 12:37:15 [i] 2 sequential tasks: { create cluster control plane "kalyanbabu",  
2021-11-15 12:37:16 [i]     2 sequential sub-tasks: {  
2021-11-15 12:37:16 [i]         wait for control plane to become ready,  
2021-11-15 12:37:16 [i]         create managed nodegroup "worker-nodes",  
2021-11-15 12:37:16 [i]     }  
2021-11-15 12:37:16 [i]     building cluster stack "eksctl-kalyanbabu-cluster"  
2021-11-15 12:37:16 [i]     deploying stack "eksctl-kalyanbabu-cluster"  
2021-11-15 12:38:06 [i]     waiting for CloudFormation stack "eksctl-kalyanbabu-cluster"  
2021-11-15 12:38:16 [i]     waiting for CloudFormation stack "eksctl-kalyanbabu-cluster"  
2021-11-15 12:39:16 [i]     waiting for CloudFormation stack "eksctl-kalyanbabu-cluster"  
2021-11-15 12:40:16 [i]     waiting for CloudFormation stack "eksctl-kalyanbabu-cluster"  
2021-11-15 12:41:16 [i]     waiting for CloudFormation stack "eksctl-kalyanbabu-cluster"  
2021-11-15 12:42:16 [i]     waiting for CloudFormation stack "eksctl-kalyanbabu-cluster"  
2021-11-15 12:43:16 [i]     waiting for CloudFormation stack "eksctl-kalyanbabu-cluster"  
2021-11-15 12:44:16 [i]     waiting for CloudFormation stack "eksctl-kalyanbabu-cluster"  
2021-11-15 12:45:16 [i]     waiting for CloudFormation stack "eksctl-kalyanbabu-cluster"  
2021-11-15 12:46:16 [i]     waiting for CloudFormation stack "eksctl-kalyanbabu-cluster"  
2021-11-15 12:47:16 [i]     waiting for CloudFormation stack "eksctl-kalyanbabu-cluster"  
2021-11-15 12:48:16 [i]     waiting for CloudFormation stack "eksctl-kalyanbabu-cluster"  
2021-11-15 12:49:16 [i]     building cluster stack "eksctl-kalyanbabu-nodegroup-worker-nodes"  
2021-11-15 12:50:17 [i]     deploying stack "eksctl-kalyanbabu-nodegroup-worker-nodes"  
2021-11-15 12:50:17 [i]     waiting for CloudFormation stack "eksctl-kalyanbabu-nodegroup-worker-nodes"  
2021-11-15 12:50:33 [i]     waiting for CloudFormation stack "eksctl-kalyanbabu-nodegroup-worker-nodes"  
2021-11-15 12:50:50 [i]     waiting for CloudFormation stack "eksctl-kalyanbabu-nodegroup-worker-nodes"  
2021-11-15 12:51:16 [i]     waiting for CloudFormation stack "eksctl-kalyanbabu-nodegroup-worker-nodes"  
2021-11-15 12:51:32 [i]     waiting for CloudFormation stack "eksctl-kalyanbabu-nodegroup-worker-nodes"  
2021-11-15 12:51:46 [i]     waiting for CloudFormation stack "eksctl-kalyanbabu-nodegroup-worker-nodes"  
2021-11-15 12:52:05 [i]     waiting for CloudFormation stack "eksctl-kalyanbabu-nodegroup-worker-nodes"  
2021-11-15 12:52:24 [i]     waiting for CloudFormation stack "eksctl-kalyanbabu-nodegroup-worker-nodes"  
2021-11-15 12:52:41 [i]     waiting for CloudFormation stack "eksctl-kalyanbabu-nodegroup-worker-nodes"  
2021-11-15 12:52:59 [i]     waiting for CloudFormation stack "eksctl-kalyanbabu-nodegroup-worker-nodes"  
2021-11-15 12:53:16 [i]     waiting for CloudFormation stack "eksctl-kalyanbabu-nodegroup-worker-nodes"  
2021-11-15 12:53:32 [i]     waiting for CloudFormation stack "eksctl-kalyanbabu-nodegroup-worker-nodes"  
2021-11-15 12:53:50 [i]     waiting for CloudFormation stack "eksctl-kalyanbabu-nodegroup-worker-nodes"  
2021-11-15 12:54:06 [i]     waiting for CloudFormation stack "eksctl-kalyanbabu-nodegroup-worker-nodes"  
2021-11-15 12:54:22 [i]     waiting for CloudFormation stack "eksctl-kalyanbabu-nodegroup-worker-nodes"  
2021-11-15 12:54:39 [i]     waiting for CloudFormation stack "eksctl-kalyanbabu-nodegroup-worker-nodes"  
2021-11-15 12:54:55 [i]     waiting for CloudFormation stack "eksctl-kalyanbabu-nodegroup-worker-nodes"  
2021-11-15 12:54:55 [i]     waiting for the control plane availability  
2021-11-15 12:54:55 [✓]     saved kubeconfig as "/var/lib/jenkins/.kube/config"  
2021-11-15 12:54:55 [i]     no tasks  
2021-11-15 12:54:55 [✓]     all EKS cluster resources for "kalyanbabu" have been created  
nodegroup "worker-nodes" has 2 node(s)  
2021-11-15 12:54:55 [i]     node "ip-192-168-58-33.ap-south-1.compute.internal" is ready  
2021-11-15 12:54:55 [i]     node "ip-192-168-58-33.ap-south-1.compute.internal" is ready  
2021-11-15 12:54:55 [i]     waiting for at least 2 node(s) to become ready in "worker-nodes"  
nodegroup "worker-nodes" has 2 node(s)  
2021-11-15 12:54:55 [i]     node "ip-192-168-14-192.ap-south-1.compute.internal" is ready  
2021-11-15 12:54:55 [i]     node "ip-192-168-58-33.ap-south-1.compute.internal" is ready  
2021-11-15 12:54:55 [i]     kubectl command should work with "/var/lib/jenkins/.kube/config", try "kubectl get nodes"  
2021-11-15 12:54:56 [✓]     EKS cluster "kalyanbabu" in "ap-south-1" region is ready  
jenkins@ip-172-31-8-110:~
```

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-ap... | Amazon EKS | New Credentials [Jenkins] | New Tab | New Tab

Not secure | 13.127.44.253:8080/credentials/store/system/domain/_/newCredentials

Jenkins

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 Credential

OK

REST API Jenkins 2.303.3

rootkey.csv jenkins.io.key Jenkins-1.pem Show all

This screenshot shows the Jenkins Global credentials (unrestricted) configuration page. A new credential is being added for Docker Hub. The Kind is set to 'Secret text' with a scope of 'Global'. The ID is 'DOCKER_HUB_PASSWORD' and the description is 'Docker Hub Credential'. The secret field contains a masked password.

Activities Google Chrome Nov 15 18:34

How to setup kubernetes | Meet - urm-wvnp-ap... | Amazon EKS | New Credentials [Jenkins] | New Tab | New Tab

Not secure | 13.127.44.253:8080/credentials/store/system/domain/_/newCredentials

Jenkins

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 Credential

OK

REST API Jenkins 2.303.3

rootkey.csv jenkins.io.key Jenkins-1.pem Show all

This screenshot shows the Jenkins Global credentials (unrestricted) configuration page. A new credential is being added for GitHub. The Kind is set to 'Username with password' with a scope of 'Global'. The ID is 'GIT_HUB_CREDENTIALS' and the description is 'Git Hub Credential'. The username is 'venkatakalyan24' and the password is a masked value.

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'
> }
```

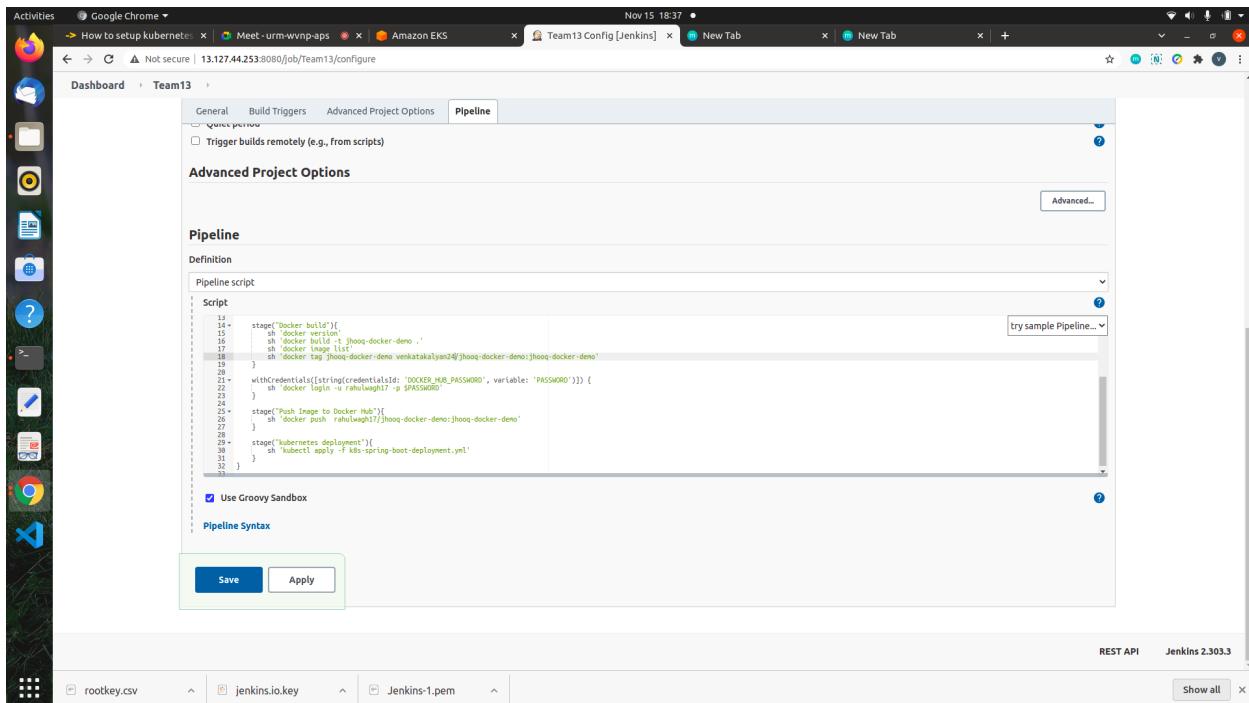
- Here is the complete final script for Jenkins pipeline

```
> > 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'
➤     }

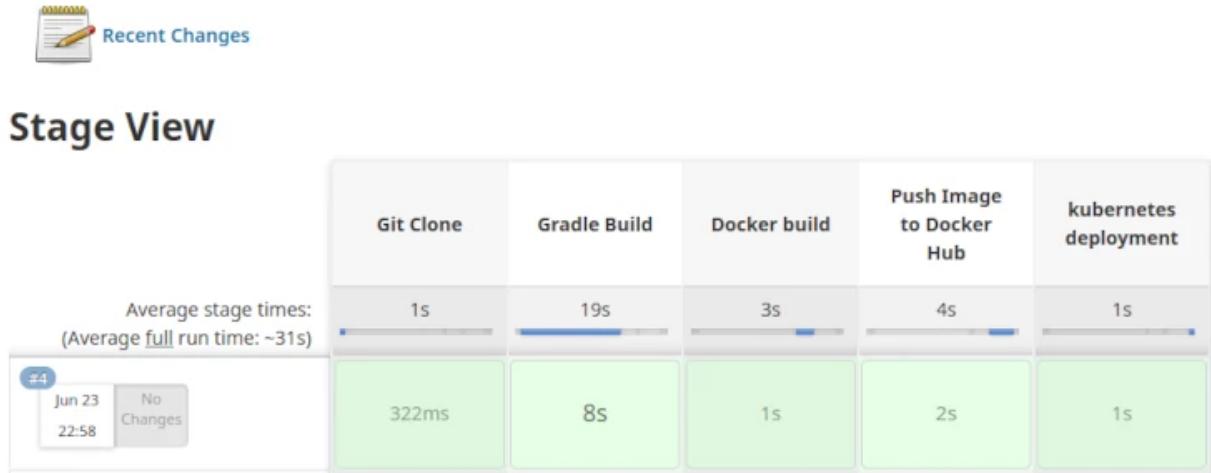
```



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



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.

