

Name : S.Guhan Balaji

# Complete CICD implementation using Jenkins

## Introduction:

In this project we are going to build an application using Jenkins and deploy to the Kubernetes cluster. In this process allows for automated and streamlined software development and deployment, ensuring that your applications are built, tested, and deployed consistently and efficiently. So we are going to use AWS, Jenkins, Maven, SonarQube, Trivy, Docker and Kubernetes in this project.

Steps Involved:

### Step 1: AWS

- Login to your AWS account.
- Launch an ec2 instance with instance type as Large. So it can handle many resources easily.
- Here I am using Mobaxterm, to connect to my instance. Ensure that port is 22.
- Copy the public ip of your aws instance and launch it in your terminal using private key you have. Finally login as ec2-user.

### Step 2: INSTALLATION

Here we have to install necessary tools and resources to our prod server to work properly.

- Jenkins
  1. Ensure that your software packages are up to date on your instance by using the following command to perform a quick software update:  
`[ec2-user ~]$ sudo yum update -y`
  2. Add the Jenkins repo using the following command:  
`[ec2-user ~]$ sudo wget -O /etc/yum.repos.d/jenkins.repo \ https://pkg.jenkins.io/redhat-stable/jenkins.repo`
  3. Import a key file from Jenkins-CI to enable installation from the package:

```
[ec2-user ~]$ sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
```

```
[ec2-user ~]$ sudo yum upgrade
```

4. Install Java (Amazon Linux 2023):

```
[ec2-user ~]$ sudo dnf install java-17-amazon-corretto -y
```

5. Install Jenkins:

```
[ec2-user ~]$ sudo yum install jenkins -y
```

6. Enable the Jenkins service to start at boot:

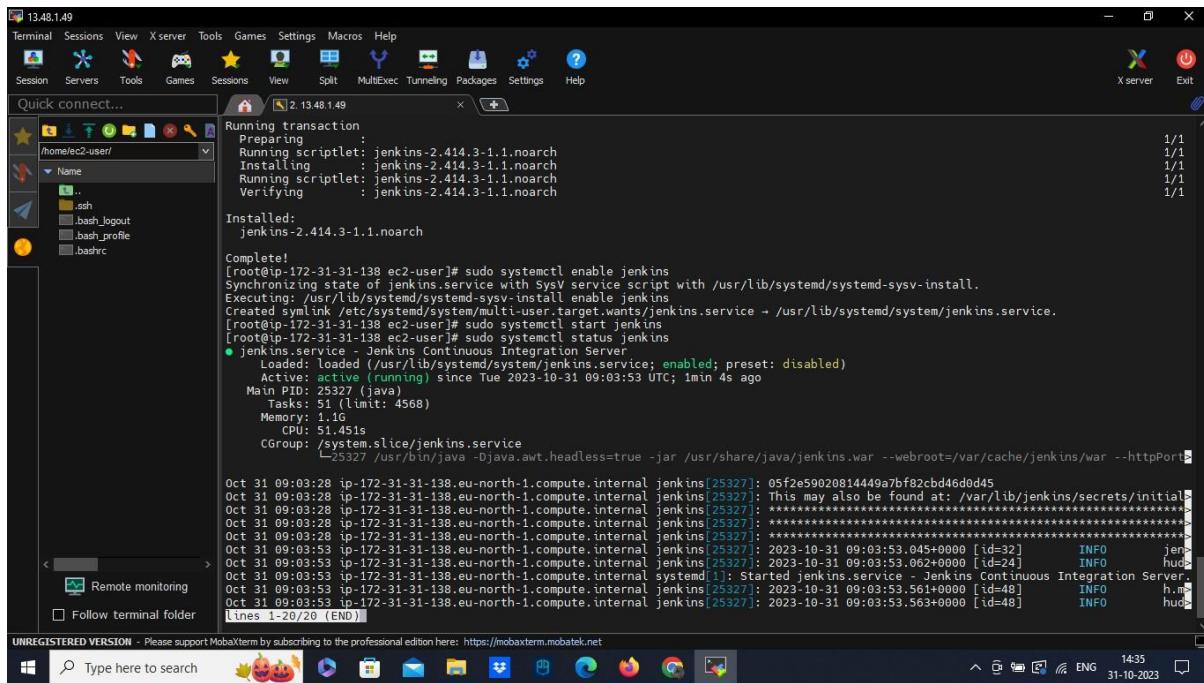
```
[ec2-user ~]$ sudo systemctl enable jenkins
```

7. Start Jenkins as a service:

```
[ec2-user ~]$ sudo systemctl start jenkins
```

8. You can check the status of the Jenkins service using the command:

```
[ec2-user ~]$ sudo systemctl status Jenkins
```



The screenshot shows a terminal session in MobaXterm with the title '2. 13.48.1.49'. The terminal window displays the output of several commands related to Jenkins installation:

```
Running transaction
Preparing...
  Running scriptlet: jenkins-2.414.3-1.1.noarch
Installing : jenkins-2.414.3-1.1.noarch
  Running scriptlet: jenkins-2.414.3-1.1.noarch
Verifying  : jenkins-2.414.3-1.1.noarch

Installed:
  jenkins-2.414.3-1.1.noarch

Complete!
[root@ip-172-31-31-138 ec2-user]# sudo systemctl enable jenkins
Synchronizing state of jenkins.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable jenkins
Created symlink /etc/systemd/system/multi-user.target.wants/jenkins.service → /usr/lib/systemd/system/jenkins.service.
[root@ip-172-31-31-138 ec2-user]# sudo systemctl start jenkins
[root@ip-172-31-31-138 ec2-user]# sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: disabled)
   Active: active (running) since Tue 2023-10-31 09:03:53 UTC; 1min 4s ago
     Main PID: 1 (java)
       Tasks: 51 (limit: 4568)
      Memory: 1.1G
        CPU: 51.45us
      CGroup: /system.slice/jenkins.service
             └─25327 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Oct 31 09:03:28 ip-172-31-31-138.eu-north-1.compute.internal jenkins[25327]: 05f2e59020814449a7bf82cbd46d0d45
Oct 31 09:03:28 ip-172-31-31-138.eu-north-1.compute.internal jenkins[25327]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Oct 31 09:03:28 ip-172-31-31-138.eu-north-1.compute.internal jenkins[25327]: ****
Oct 31 09:03:53 ip-172-31-31-138.eu-north-1.compute.internal jenkins[25327]: 2023-10-31 09:03:53.045+0000 [id=32]      INFO    jen>
Oct 31 09:03:53 ip-172-31-31-138.eu-north-1.compute.internal jenkins[25327]: 2023-10-31 09:03:53.062+0000 [id=24]      INFO    hu>
Oct 31 09:03:53 ip-172-31-31-138.eu-north-1.compute.internal jenkins[25327]: Started jenkins.service - Jenkins Continuous Integration Server.
Oct 31 09:03:53 ip-172-31-31-138.eu-north-1.compute.internal jenkins[25327]: 2023-10-31 09:03:53.561+0000 [id=48]      INFO    h.m>
Oct 31 09:03:53 ip-172-31-31-138.eu-north-1.compute.internal jenkins[25327]: 2023-10-31 09:03:53.563+0000 [id=48]      INFO    hu>
[lines 1-20/20 (END)]
```

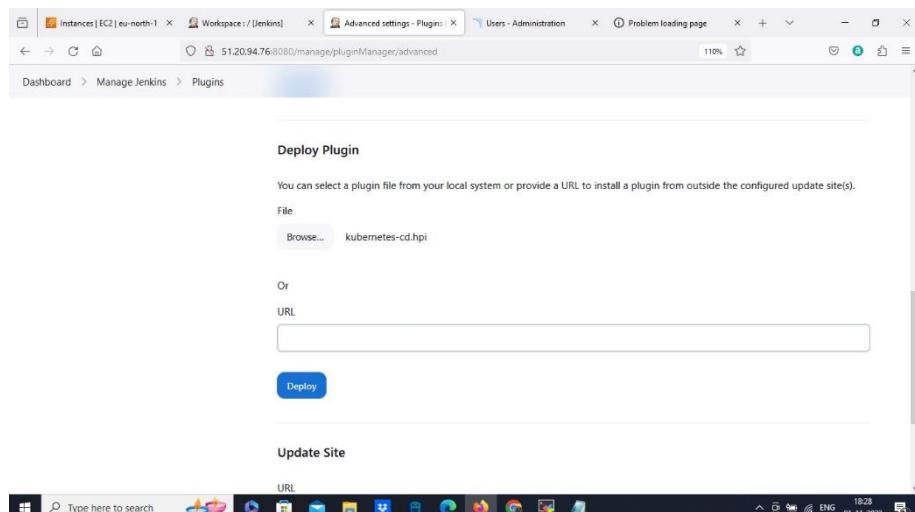
At the bottom of the terminal window, there is a message: "UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>".

- Jenkins is now installed and running on your EC2 instance. Now to open Jenkins.
- Copy the public ip and paste it in the browser as `http://<ip>:8080`.
- `[ec2-user ~]$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword` – use this command to get the password
- Login in with the initial password. Create user and new password.
- And you will get login. Now back to the server
- Install Git:
  1. `sudo yum install git -y`

- Install Trivy ( to secure a container) :
  1. Download the Trivy binary package for your architecture (either trivy\_0.12.0\_Linux-64bit.tar.gz or trivy\_0.12.0\_Linux-32bit.tar.gz) from the GitHub releases page: <https://github.com/aquasecurity/trivy/releases>
  2. Unpack the archive by running tar xvf trivy\_0.12.0\_Linux-64bit.tar.gz
  3. Move the binary to a directory in your PATH, for example /usr/local/bin/ by running mv trivy /usr/local/bin/
  4. Verify that trivy is installed by running trivy –version
- Install Docker:
  1. yum install docker -y
  2. systemctl start docker
  3. systemctl enable docker
- Pull Docker image from sonarqube and run the container:
  1. docker pull sonarqube:its-community
  2. docker run -d --name sonarqube -p 9000:9000 sonarqube:its-community
  3. docker ps to check the container status.
  4. To grant permission between Jenkins and docker use: sudo usermod -aG docker Jenkins
- now restart the Jenkins and login in to Jenkins in browser.

### Step 3: Installing plugins in Jenkins:

- Here we need to install additional plugin in our Jenkins
- Go to manage Jenkins and go to available plugins. The plugins are:
  1. Jdk from adoptium.net
  2. Sonarqube Scanner
  3. Docker
  4. Kubernetes
- Sometimes an important Kubernetes CD plugin wont be available. In that case we need to install it manually to our Jenkins:
  - a. Download Kubernetes cd.hpi (plugin) from browser



b. And install it manually in manage plugins

#### Step 4: Configure tools in Jenkins:

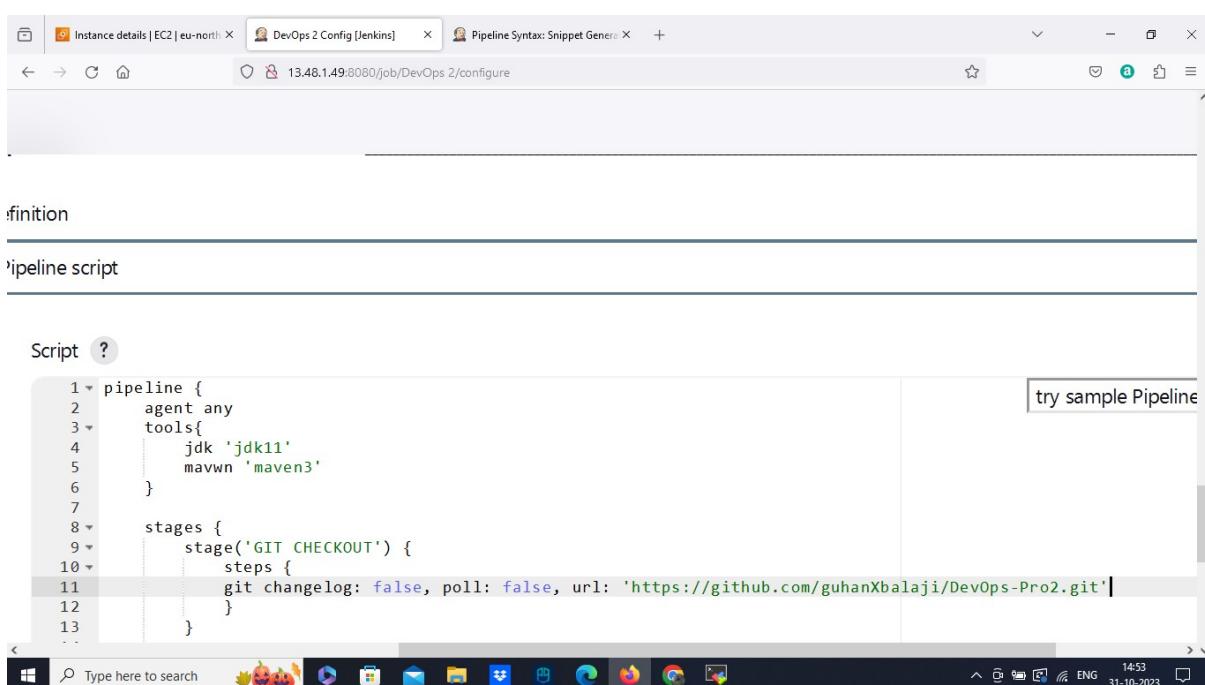
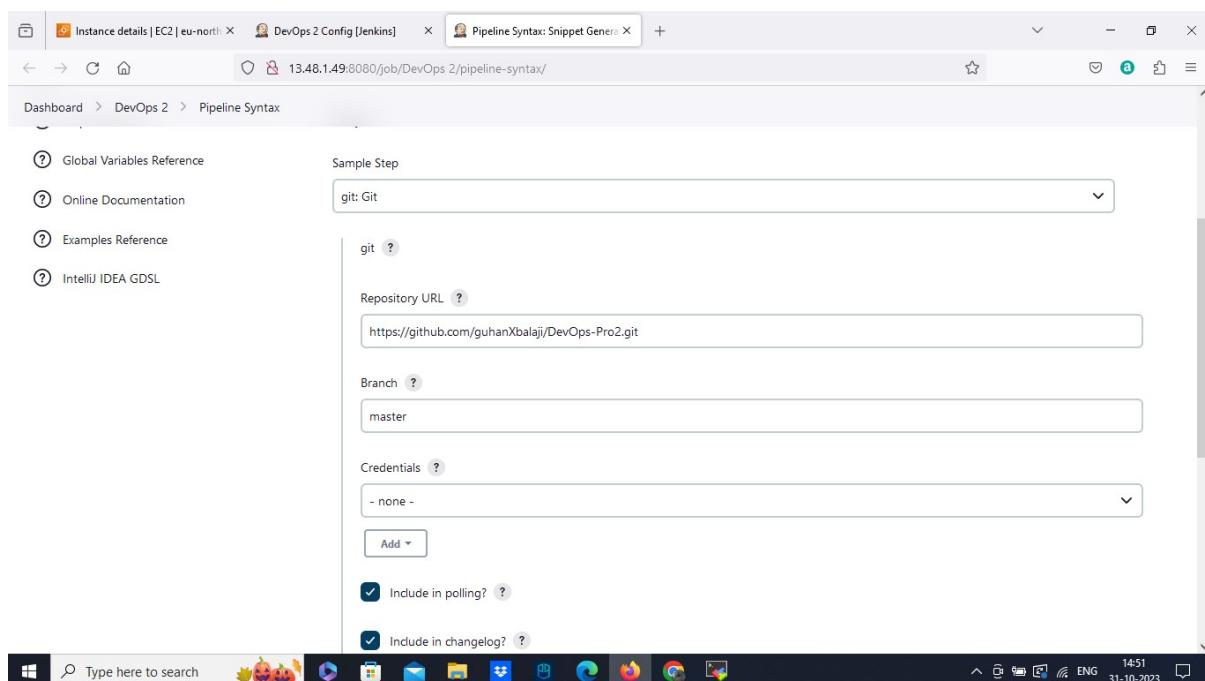
- Open configure tools in Jenkins
- Add jdk and name it jdk 11, with the installer adoptium.net and select the version
- Add maven and name it maven3, and select the version.
- Add docker with the latest version and add sonarqube as well.

The screenshot shows the Jenkins interface for managing tools. The URL is 13.51.107.243:8080/manage/configureTools/. The page is titled 'JDK installations'. A sub-section titled 'Add JDK' is active. It shows a 'JDK Name' field containing 'jdk11'. Below it is a checkbox 'Install automatically?' which is checked. Underneath is a sub-section 'Install from adoptium.net?' with a 'Version' dropdown set to 'jdk-11.0.19+7'. There is also a 'Add Installer' button. At the bottom are 'Save' and 'Apply' buttons. The background shows other tabs like 'Instance details | EC2 | eu-north-1' and 'EC2 | eu-north-1'.

The screenshot shows the Jenkins interface for managing tools. The URL is 13.51.107.243:8080/manage/configureTools/. The page is titled 'Ant installations' and 'Maven installations'. The 'Maven installations' section is active. A sub-section titled 'Add Maven' is active. It shows a 'Maven Name' field containing 'maven3'. Below it is a checkbox 'Install automatically?' which is checked. At the bottom are 'Save' and 'Apply' buttons. The background shows other tabs like 'Instance details | EC2 | eu-north-1' and 'EC2 | eu-north-1'.

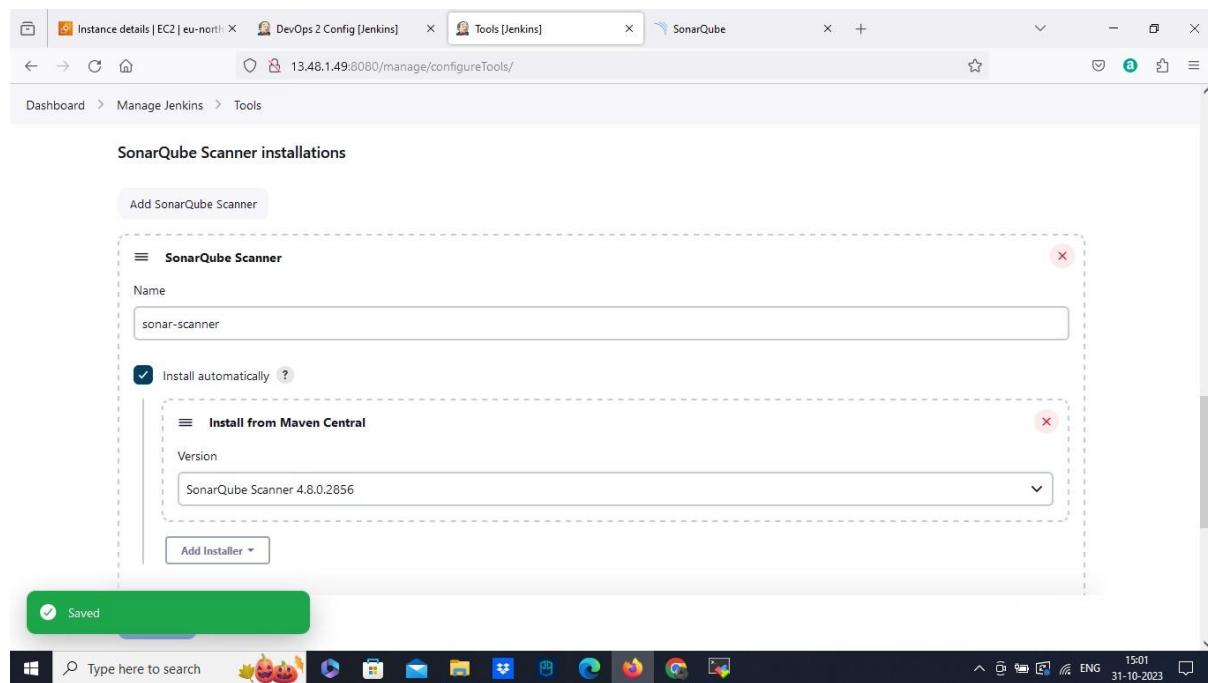
## Step 5: Create a Pipeline:

- Create a pipe line. Add a name and description.
- Write the script with any agent and use tools jdk11 and maven3.
- In the next stage of the script add the stage GIT CHECKOUT
- For this script use pipeline syntax for generate script
- Select git:Git as sample step and paste the repository URL. Then generate the script, copy it and paste it in the main script.
- In the next stage of the script will be code compile, and add the command sh “mvn clean compile”.



## Step 6: Sonarqube analysis:

- The next stage of the script will be sonar qube analysis
- We already started the sonarqube in our server using docker container. Copy the public ip and paste it in browser http:// :9000.
- Login the sonarqube and go to administration/security.
- Generate token.
- Copy the token. Create a credentials which is secretext. Add the token in secret field and save the credentials with id in your Jenkins.
- Go to configure system in Jenkins. Select sonarqube installations, add name, server URL and authentication token we created as a secret text.
- And now go back to the script we created. Using pipeline syntax again, generate a script .
- Add sonarqube env as sample step, and also add sonar token.
- Generate the script,copy it and paste it to our main script.



The screenshot shows the SonarQube Administration - Users page. At the top, there are tabs for Configuration, Security (which is selected), Projects, Rules, Quality Profiles, Quality Gates, System, and Marketplace. A search bar at the top right contains the placeholder "Search for projects...". Below the tabs, a sub-menu has options for Administration, Configuration, Security, Projects, System, and Marketplace. The main content area is titled "Users" with the sub-instruction "Create and administer individual users". A search bar labeled "Search by login or name..." is present. A table lists one user entry:

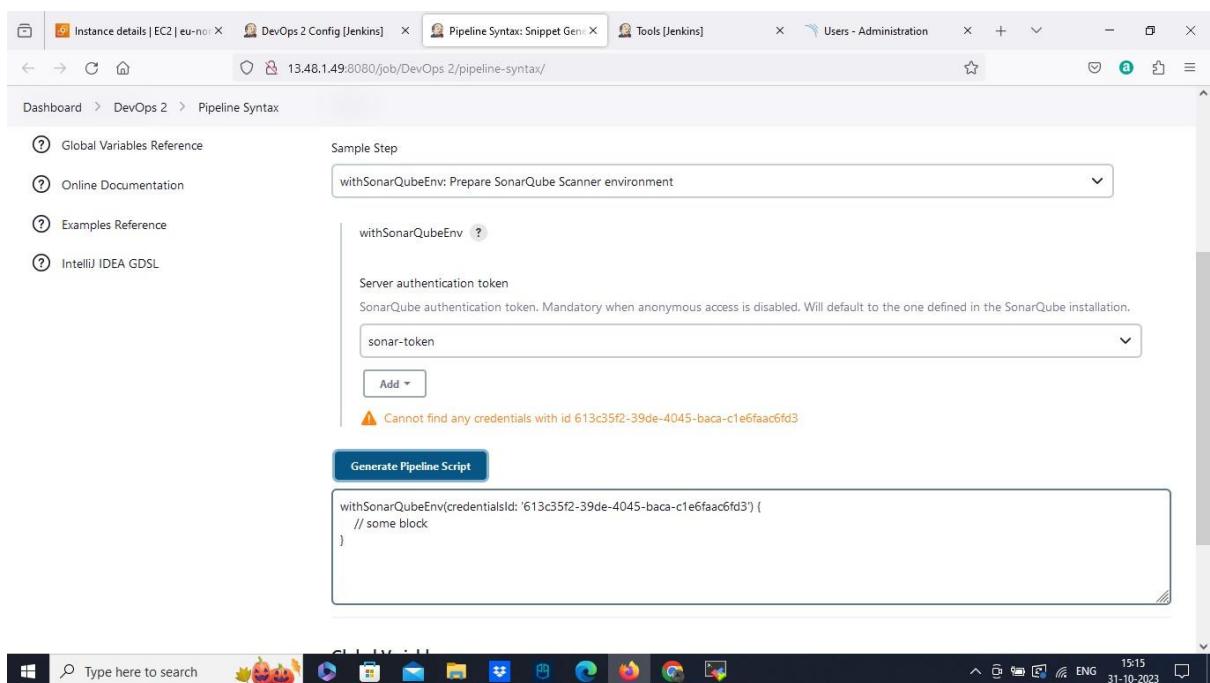
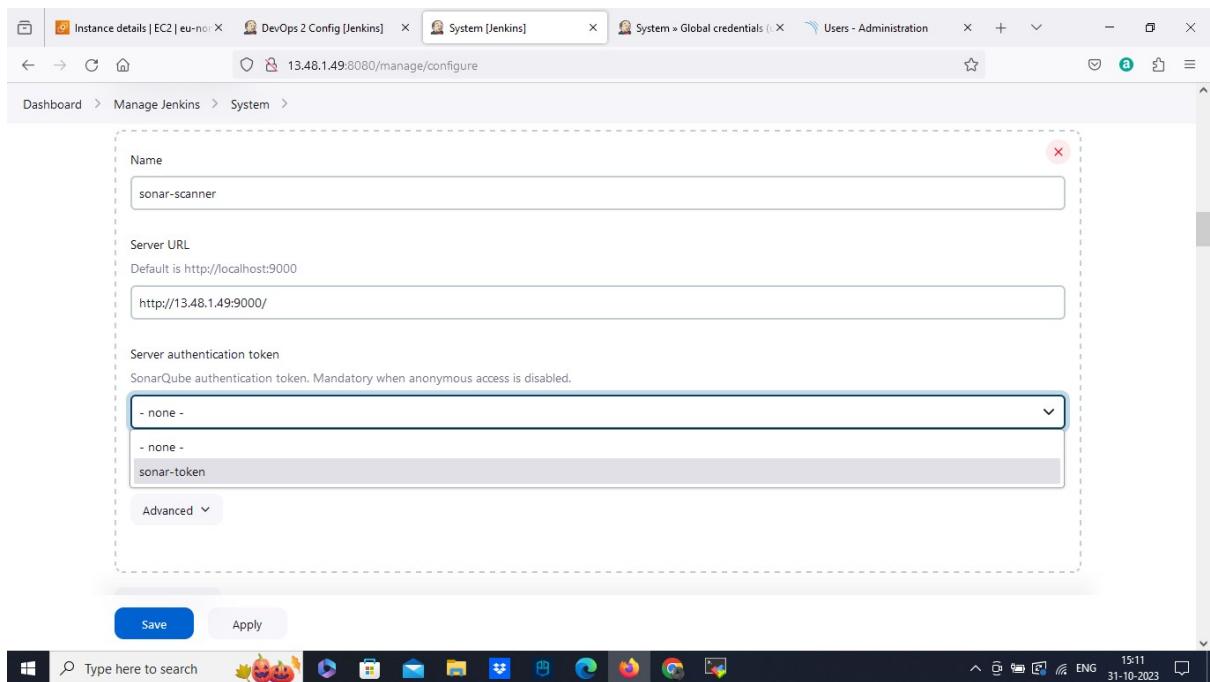
	SCM Accounts	Last connection	Groups	Tokens
A Administrator admin		< 1 hour ago	sonar-administrators sonar-users	0

Below the table, a message states: "Embedded database should be used for evaluation purposes only. The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine." The footer includes the SonarQube logo and version information: "Community Edition - Version 9.9.2 (build 77730) - LGPLv3 - Community - Documentation - Plugins - Web API". The taskbar at the bottom shows various pinned icons and the date/time: "15:04 31-10-2023".

The screenshot shows the Jenkins Manage Jenkins > Credentials > Global credentials (unrestricted) page. The URL is 13.48.1.49:8080/manage/credentials/store/system/domain/\_/newCredentials. The page displays a form for creating a new credential:

Kind	Secret text
Scope	Global (Jenkins, nodes, items, all child items, etc)
Secret	*****
ID	
Description	sonar-token

At the bottom left is a blue "Create" button. The taskbar at the bottom shows various pinned icons and the date/time: "15:09 31-10-2023".



- Save the script and select build now to build the script we done so far.
- After the built we can see that it got passed in sonarqube as you can see

The screenshot shows the Jenkins Pipeline DevOps 2 project page. On the left, there's a sidebar with options like Status, Changes, Build Now, Configure, Delete Pipeline, Full Stage View, Rename, and Pipeline Syntax. The main area is titled "Pipeline DevOps 2" and describes it as a "CICD Pipeline project". Below this is a "Stage View" section showing four stages: Declarative: Tool Install (31s), GIT CHECKOUT (7s), COMPILE DA CODE (26s), and SONARQUBE ANALYSIS (24s). A summary says "Average stage times: (Average full run time: ~1min 34s)". To the left of the stage view is a "Build History" card showing two builds: #3 (Oct 31, 15:32) and #2 (Oct 31, 15:28), both with "No Changes". The bottom of the screen shows a Windows taskbar with various icons and a search bar.

The screenshot shows the SonarQube Projects interface. The top navigation bar includes links for Instance details, DevOps 2 [Jenkins], Pipeline Syntax: Snippet Gen..., Tools [Jenkins], and Projects. The main content area shows a single project named "DevOps2" with a status of "Passed". It displays metrics for Bugs (0 A), Vulnerabilities (0 A), Hotspots Reviewed (0.0% E), Code Smells (1 A), Coverage (0.0% O), Duplications (0.0% G), and Lines (120 X XML, YAML...). On the left, there are filters for Quality Gate (Passed: 1, Failed: 0), Reliability (A rating: 1, B rating: 0, C rating: 0, D rating: 0, E rating: 0), and Security (A rating: 1, B rating: 0, C rating: 0, D rating: 0, E rating: 0). A note at the bottom states: "Embedded database should be used for evaluation purposes only. The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine." The bottom of the screen shows a Windows taskbar with various icons and a search bar.

## Step 7: Trivy scan:

- The next stage of our script will be trivy scan.
- And add the command sh "trivy fs --security-checks vuln,config /var/lib/jenkins/workspace/DevOps\\ 2" in the script
- Then add the next stage as CODE BUILD and add the command sh "mvn clean install" in the step.
- Save it and build now.

The screenshot shows the Jenkins Pipeline configuration page. The pipeline script is defined as follows:

```
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
```

stage('SONARQUBE ANALYSIS') {  
 steps {  
 withSonarQubeEnv('sonar-scanner') {  
 sh '\$SONAR\_HOME/bin/sonar-scanner -Dsonar.projectName=DevOps2  
 -Dsonar.java.binaries=. \  
 -Dsonar.projectKey=DevOps2 ...'  
 }  
 }  
}  
stage('TRIVY SCAN') {  
 steps {  
 sh "trivy fs --security-checks vuln,config /var/lib/jenkins/workspace/DevOps 2"  
 }  
}

Below the script, there is a checkbox for "Use Groovy Sandbox". At the bottom, there are "Save" and "Apply" buttons. The status bar at the bottom right indicates Jenkins 2.414.3.

The screenshot shows the Jenkins Pipeline Stage View. The stages listed are Declarative: Tool Install, GIT CHECKOUT, COMPILE DA CODE, SONARQUBE ANALYSIS, TRIVY SCAN, and CODE BUILD. The TRIVY SCAN stage is highlighted in red, indicating a failure. The SonarQube Quality Gate section shows a green "Passed" status. The Permalinks section provides links for the pipeline and specific builds.

Declarative: Tool Install	GIT CHECKOUT	COMPILE DA CODE	SONARQUBE ANALYSIS	TRIVY SCAN	CODE BUILD
132ms	1s	9s	15s	7s	12s

Average stage times:  
(Average full run time: ~1min 1s)

#7	Oct 31 16:07	No Changes	155ms	691ms	10s	15s	8s	25s
#6	Oct 31 16:04	No Changes	110ms	1s	9s	15s	6s failed	46ms failed

SonarQube Quality Gate

DevOps2 | Passed

server-side processing | Success

Permalinks

## Step 8: Docker build and push:

- The next stage will be docker build in our script
- We can use pipeline syntax here with the dockerregistry, add the username and password of your docker hub. And generate the syntax.
- Copy to the main script and add the command sh "docker build -t cicddevops ." The image will be build
- The next stage will be docker push copy the same script and add docker tag and pushin the script.
- Apply it and build now

The screenshot shows the Jenkins Pipeline Syntax configuration screen. The left sidebar has tabs for General, Advanced Project Options, and Pipeline, with Pipeline selected. The main area contains Groovy code for a pipeline:

```
46 v
47 v
48 v
49 v
50 v
51 v
52 v
53 v
54 v
55 v
56 v
57 v
58 v
59 v
60 v
61 v
62 v
63 v

stage('DOCKER BUILD') {
    steps {
        script{
            withDockerRegistry(credentialsId: '1b34a6ca-98ec-40f7-816e-73c7d1928f82', toolName: 'docker')
            sh "docker build -t cicddevops ."
        }
    }
}

stage('DOCKER PUSH') {
    steps {
        script{
            withDockerRegistry(credentialsId: '1b34a6ca-98ec-40f7-816e-73c7d1928f82', toolName: 'docker')
            sh "docker tag cicddevops guhanxdocker/devopscid:$BUILD_ID"
            sh "docker push guhanxdocker/devopscid:$BUILD_ID"
        }
    }
}
```

Below the code, there is a checkbox for "Use Groovy Sandbox". At the bottom are "Save" and "Apply" buttons. The status bar at the bottom right shows "REST API Jenkins 2.414.3".

The screenshot shows the Jenkins Pipeline DevOps 2 project page. The left sidebar has tabs for status, Changes, Build Now, Configure, Delete Pipeline, and Pipeline Syntax, with Pipeline Syntax selected. The main area has a "Stage View" table and a "Build History" section.

**Stage View**

	Declarative: Tool Install	GIT CHECKOUT	COMPILE DA CODE	SONARQUBE ANALYSIS	TRIVY SCAN	CODE BUILD	DOCKER BUILD	DOCKER PUSH
Average stage times: (Average full run time: ~1min 12s)	202ms	942ms	8s	13s	7s	17s	8s	13s
#9 Oct 31 23:44 No Changes	173ms	753ms	8s	13s	7s	17s	4s	13s
#8 Oct 31 23:38 No Changes	333ms	1s	9s	14s	7s	17s	20s	
#7 Oct 31 23:32 No Changes	100ms	716ms	8s	13s	7s	17s	649ms	

**Build History**

- #9 Oct 31, 2023, 6:14 PM
- #8 Oct 31, 2023, 6:08 PM
- #7 Oct 31, 2023, 6:02 PM

The status bar at the bottom right shows "REST API Jenkins 2.414.3".

## Step 9: Deploy to Kubernetes:

- Now we are going to deploy to the Kubernetes.
- First we will create Kubernetes cluster using AWS.
- Go to AWS, launch 2 instance with AMI type UBUNTU and instance type as medium.
- Make sure to create in same subnet
- Then rename one instance as master and other as worker.
- Same here iam going to use Mobaxterm as my both master and worker terminal.
- And login user as Ubuntu and with our private key.

The screenshot shows the AWS EC2 Instances page with three instances listed:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
Devops Prod	i-0677682f046389b2b	Running	t3.large	2/2 checks passed	No alarms	eu-north-1a
Master k8	i-0824450491c6bbd4e	Running	t3.medium	2/2 checks passed	No alarms	eu-north-1a
worker k8	i-0d8f311eee024442	Running	t3.medium	2/2 checks passed	No alarms	eu-north-1a

Below the instances, there is a monitoring section showing CPU utilization and status check failed metrics for the three instances.

- Kubernetes cluster installation (need to run these commands in both master and worker instance).
  - Switch to root user  
sudo su -
  - Update the System  
apt-get update
  - Install http package  
apt-get install apt-transport-https
- Install Docker
  - apt install docker.io -y
  - docker --version
  - systemctl start docker
  - systemctl enable docker
- Setup open GPG Key
  - sudo curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add
- Edit source list file  
nano /etc/apt/sources.list.d/kubernetes.list

- Add below line in the above file  
deb http://apt.kubernetes.io/ kubernetes-xenial main
- Install the Kubernetes packages
  - apt-get update
  - apt-get install -y kubelet kubeadm kubectl kubernetes-cni
- BOOTSTRAPPING THE MASTER NODE (Only in MASTER Node) - kubeadm init
- Copy the steps of mkdir.kube which wil be above and run it in master instance.
- Deploy flannel node network - kubectl apply -f  
<https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml>
- Configure Worker Nodes - kubeadm join 172.31.14.32:6443 --token  
mcvd2n.aqvyp59vq3inhtks --discovery-token-ca-cert-hash  
sha256:5ab6e6d695d31c05a4e4cde8f6e7b14ff0d0b450383e9b255270be90904de12  
8 ( copy this in the worker instance only).
- Go to master and run the command kubectl get nodes

```

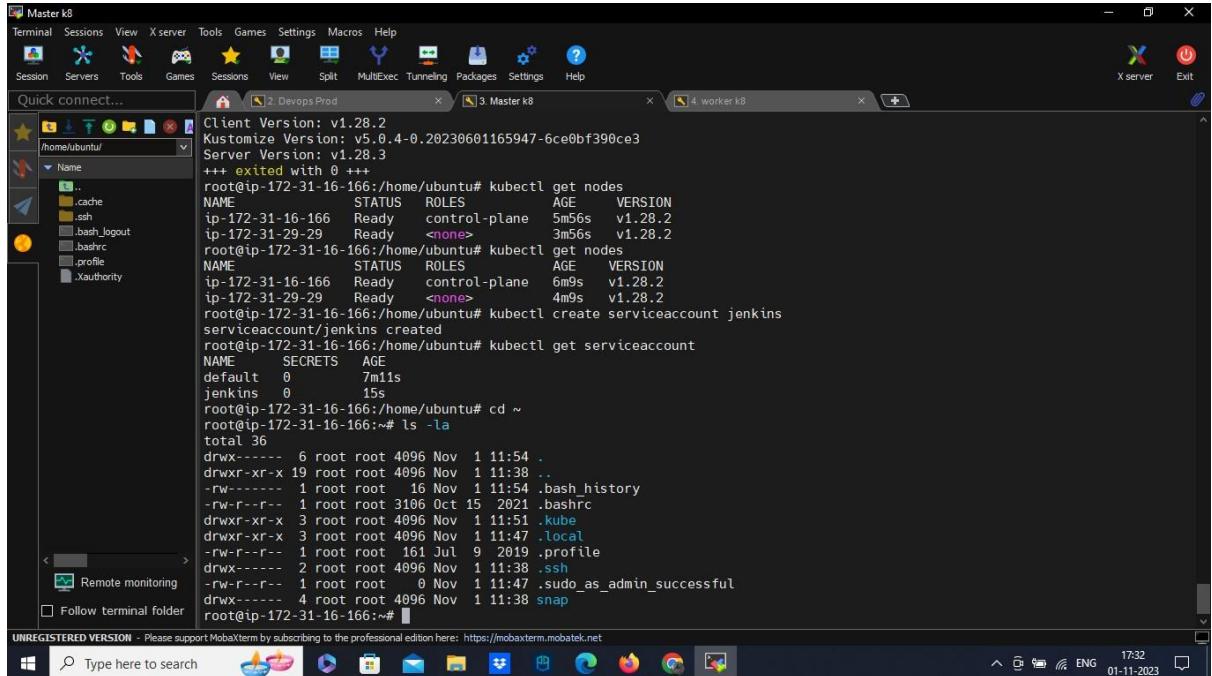
Master k8
Terminal Sessions View Xserver Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
Quick connect...
[Session] [Servers] [Tools] [Games] [Sessions] [View] [Split] [MultiExec] [Tunneling] [Packages] [Settings] [Help]
[?]
[?] [X] [Exit]
[?] [X] [Exit]

2 Devops Prod < 3. Master k8 > 4. worker k8
--- SIGURG {si_signo=SIGURG, si_code=SI_TKILL, si_pid=7594, si_uid=0} ---
openat(AT_FDCWD, "/root/.kube/config", O_RDONLY|O_CLOEXEC) = 3
Client Version: v1.28.2
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf990ce3
Server Version: v1.28.3
+++ exited with 0 +++
root@ip-172-31-16-166:/home/ubuntu# kubectl get nodes
NAME           STATUS   ROLES      AGE    VERSION
ip-172-31-16-166 Ready    control-plane   5m56s   v1.28.2
ip-172-31-29-29 Ready    <none>    3m56s   v1.28.2
root@ip-172-31-16-166:/home/ubuntu# kubectl get nodes
NAME           STATUS   ROLES      AGE    VERSION
ip-172-31-16-166 Ready    control-plane   6m9s    v1.28.2
ip-172-31-29-29 Ready    <none>    4m9s    v1.28.2
root@ip-172-31-16-166:/home/ubuntu# kubectl create serviceaccount jenkins
serviceaccount/jenkins created
root@ip-172-31-16-166:/home/ubuntu# kubectl get serviceaccount
NAME        SECRETS   AGE
default     0          7m11s
jenkins    0          15s
root@ip-172-31-16-166:/home/ubuntu# 

UNREGISTERED VERSION - Please support MobaTerm by subscribing to the professional edition here: https://mobaterm.mobatek.net
Windows Type here to search 17:28
ENG 01-11-2023
```

- Create a service account named Jenkins. With the command kubectl create serviceaccount Jenkins, and kubectl get serviceaccount to check it.
- Now go to our scriptline configure. And add Deploy to k8 stage.
- With the help of pipeline syntax. We are using Kubernetes Deploy.
- Here we are going to add kubeconfig file as a credential format.
- Go to our master instance view the config file from .kube/config, copy that and paste it in the kubeconfig credentials content and save it. Generate the pipeline script.

- Copy the generated script to the main script under the stage deploy to k8 and save it.
- Need to change a thing, that is open your git repository, locate the deploymentservice.yaml.
- Open the file from git and change the docker image to our docker image we created and pushed to the hub.
- That's it save it. And build now.

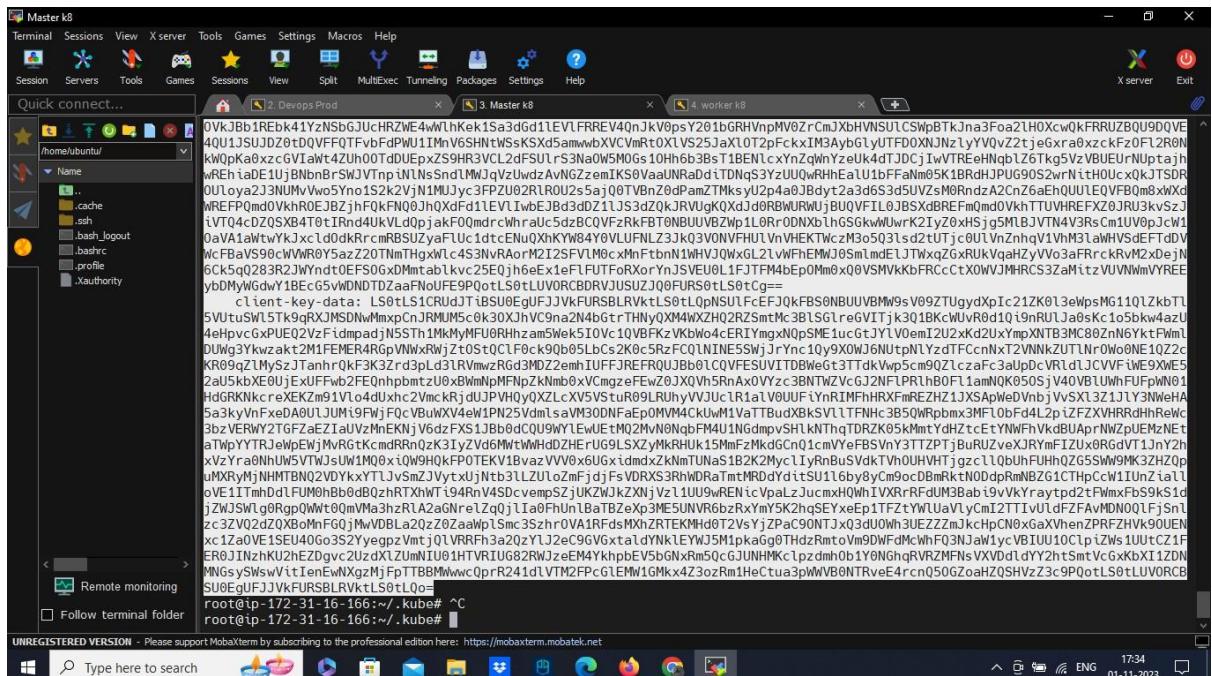


```

Client Version: v1.28.2
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
Server Version: v1.28.3
+++ exited with 0 +++
root@ip-172-31-16-166:/home/ubuntu# kubectl get nodes
NAME           STATUS   ROLES      AGE   VERSION
ip-172-31-16-166 Ready    control-plane   5m56s   v1.28.2
ip-172-31-29-29 Ready    <none>     3m56s   v1.28.2
root@ip-172-31-16-166:/home/ubuntu# kubectl get nodes
NAME           STATUS   ROLES      AGE   VERSION
ip-172-31-16-166 Ready    control-plane   6m9s    v1.28.2
ip-172-31-29-29 Ready    <none>     4m9s    v1.28.2
root@ip-172-31-16-166:/home/ubuntu# kubectl create serviceaccount jenkins
serviceaccount/jenkins created
root@ip-172-31-16-166:/home/ubuntu# kubectl get serviceaccount
NAME      SECRETS   AGE
default   0          7m1s
jenkins   0          15s
root@ip-172-31-16-166:/home/ubuntu# cd ~
root@ip-172-31-16-166:# ls -la
total 36
drwx----- 6 root root 4096 Nov  1 11:54 .
drwxr-xr-x 19 root root 4096 Nov  1 11:38 ..
-rw----- 1 root root 16 Nov  1 11:54 .bash_history
-rw-r--r-- 1 root root 3106 Oct 15 2021 .bashrc
drwxr-xr-x 3 root root 4096 Nov  1 11:51 .kube
drwxr-xr-x 3 root root 4096 Nov  1 11:47 .local
-rw-r--r-- 1 root root 161 Jul  9 2019 .profile
drwxr-xr-x 2 root root 4096 Nov  1 11:38 .ssh
-rw-r--r-- 1 root root 0 Nov  1 11:47 .sudo_as_admin_successful
drwxr-xr-x 4 root root 4096 Nov  1 11:38 snap
root@ip-172-31-16-166:# 

```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>



```

OKvKb1REBk41YzNsBGJUcHRZWE4wLhKek1Sa3ddGd11EVUFRREV40nJKv0psY201bGRHlVnpMV02rCmjxblvNSU1CSiwpBTkJna3foa21HOxw0KFPRUZBQ9D0QE
4QU1JSUJDZ0tDQVFFQTfBpFd51IMntW5HnSksXdf5amwvbXCVnrt0XlVS25ja10t2pFkxIM3AybGlyUTFD0XWjNz1yYVqVz2tje6xra0xzckFz0F12R0N
kwQpKa0xz2gVlaWt42Uh00tdUepxzS9Hr3VCL2dFSU1rs3Na0w5M0gs10Hn6b3bsT1BEN1xrnqWyeuk4dTDJc1jVtVtRehNqbZ6Tkg5VzVBUEUUrUp0tajh
wREniaDE1BnBnBrSWJTpnsLNTNsndTlMwjqVzUwdzAvNGZzemIKS0VaaUnRaDidtNDNgS3YzU0wRHneAu1bFaWn5K1BRhdJPUG9052wvNtHOUcqxJtSDR
0U1oya2j3NUMVw05y0n152k2VjN1M0Jy3PFZU02RL0U2s5a1q0TBnZ0dPamTMsjyU2p4a0Bdyt2a3dS3d5UVzMsM0RndzA2Cn26AenQ0U1EQVFb0mWxhxD
wREFPQd0Vkh0RjBZjhnF0Qj0xdF1iELV1wbeJ3d3d211s3dZQkRVuQx0djxdrBwURWUjB0u0V1l0J0BSxBRFEmd0vKhTTUHREFXZ0jR03kvSzJ
iVT04cDZ0SX8B4t0IRn04kVlD0pjakF00mdrcwharUc5d2BC0VzRkFBT0BUUWbzW0LDRr0Dnxh1gSGkyWUwRk21yZ0xH15g5MLBJVtN4V3RsCm1V0pJcw1
0a0Vw1tWxjxcl0d0krCmRBSLRV1d0tCn1EPUt4tEnU0XhKYw84Yv0LUFNLZ33k03VONFHULvWHEKTWczM053lsd2tU7c0ULvZnhqV1vhM31awHVSdEFTdV
wcfBaV890cwVWR0y5azZ207NmTHgxlw453NmRaorM2125FVU0mcxMnFtbnn1WhJ3W0wxG21vwFEMWj0Sm1nElEJTwxqZgxRukVqaHzyVw03afRcrkRvM2xDejN
6Ck5q0283R2JWYndt0EFS0GxMmtab1kv25E0jhe6Ex1eFLUTFoRxOrJnSVEU0L1JTFM4bEp0m6x0v8SMVkbFRCccxt0WVJMRCS3ZaMitZv0UNwInVYRE
ybDMy6Gdw1YBEc5VwDNDT0zaFn0uFE9P0tLS0tL0UvORCBDRJUSUZjQ0fURS05t0L0tCg=
client-key-data:
LS0tLS1CrUDjtIiBSU0EgUfJJKvFURSBLRVktLs0tL0qNSU1fcEFQjkFB50NB0UUVBMw9sV09ZTU7UgydXpIc21Zk013ewpsMG1101ZkbTl
5VUtus51Tk9gRQXMSDwMlxpcJRMU5c0t30XjhVC9na2N4BgrTRHNYQXM4WXZQ2RzSmc3B1SGLre6VITjK301BKcWUvR0d1q19nRULja0skc1o5bkv4azU
4eHpvGxPUE02VzFidmpadN5StHr0Kf3MzK3r3d3LrVmzwRgd3MDZ2emhIUfFJRFERQUjBb0lCqVfESU1TDBw6gt3TdkWp5cm90ZLzcafc3alp0cVRl1dJCvVFtWE5
DUw9qzL3Kwzakt21EMERGpWWxRjZt0s0tC1F0cK9Qb05L1bc52K0c5R2F0C1NINE55WjJyRnc10y9X0WJ6NUtPnYzdtFCcnNxz2WNKzUTTLn0w0NE1022c
KR09qzL3Kwzakt21EMERGpWWxRjZt0s0tC1F0cK9Qb05L1bc52K0c5R2F0C1NINE55WjJyRnc10y9X0WJ6NUtPnYzdtFCcnNxz2WNKzUTTLn0w0NE1022c
2au5kbXE0UjExUffw2F0EqnhpbmtzU0xBlmnpMFpnZKnb0xVmcmgzeFwz0JXqVh5RnAx0Vyzc3BNTwzVcG2J2NFPLPrhB0fL1amNQk05jv40Vb1UwhFUfpwN01
HdGRKnKrcxEKZm91v1Qd0dxdh2VmcKrjdV0QJPH0tQzLcXV5tsur09LRuhvYJUc1atV09UfUyTr1MphhRxfmREZHZ1JXSApWeDwrbjVvSx13z1JLY3NwHA
5a3kyVnFxe0AU0IJUm1Fw0FVbUw4W1P25Vdm1sVaM300nfAep0MM4cKuWm1VattBudXBksVl1TfNn3B50WPbmoxMFl0bFd4L2pZfZKvHRRdHnReWc
3bzVERy2TGF2aeZ1auVzMnENKjV6dzFxs1JBb6dCQJU9WYLeWUetM0Q2M0N8qbFM4U1N6ndmVsHlkNhqtDZR205KmYdH2tctEYNWFhVkdBuaPrNWZpUMenNet
aTlpYYTRjewpEWjMvRgtKcmdRnQzK31yZvd6MwtWmhdZHeurU9LSzXyMkRhuk15MmfzMcGdnQ1cmeYYFB5vNzY3TzPTjBuRuZveJXRymfIZUxORGdVT1JnYzH
xVzYra0NhUw5VTKWjsU1M0Qx10W9HQkFPOTEK1BvazVV0x6UGx1dmdzxKnmTuNaS1B2K2MycIlyRnBuSVdkTvh0UHtjgzcl1lObuhFUH0QZG55Wn9MK32H2Op
uMxRyMjNHMTBNQ2DYkxXTlJvSmzJYvtJxJntb3LzU1UzmfjzdfjFsVDRXs3RhrDaTmtMrd0dYdtSIU16by8Cm9ocDBmRktN0dprmNB2G1CThpCc1Iu2z1a1l
oVe117mhd1FJM0hBb0B02hRTxhM194RnV4SdcvempSzjUK2WjLzXnjVz11Uu9wREN1cVpaLzJucmxh0WbIVXRFRdU5Mbab19vVKraytpd2tFwmxFbS9ks1d
jZWjSwLg0Rgp0Wt0QmMa3hzR1A2aGnrelZqj1la0FhUnlBaTBZexp3ME5UNVR6bzRyXmY5K2hqsEYx0pe1TfZtYwluavlyCm12TT1vUldFZFAvMDN0Q1fjsnl
zc3ZV2dZQ2BxMnFGQjhwVDBLz2Qz02adw1Sm3S3zrhvVA1Rfd5MxhZTREKmhd0T2VsYjZPaC90NTjx3d0uWh3z2ZmJckhPcN0xGaxWhenZPRFZHvK90UN
xc12a0E1SEU40Go3S2YyegpVmJ0lVRFHf3a2QzYLJ2eC9GVxta1dYNYkEWYJ5M1pkag0tHdzRmtoV9m90WfMcWhF03Njw1ycVBIU01C1Lzws1UUcZ1F
ER0JINzKu2HeZdgvc2uzLzUmNI01HTVR1U682RNjzeEM4YkhpBv5BGNxRm5QGJUNHMKcLpzdm0b1YONGhqrVRZMFNsVXV0dldYy2htSmvCxGkbX12ZD
MNgsySwsw1tenewXqzAjMpfTTBBMwqc0prR241d1Vtm2FpcG1emW16Mkx4Z3ozRm1HeCtau3pwB0NTRvE4rcn050GzoaH0zSH0ShVz3c9P0otLS0tLUV0RCB
SU0EgUFJJVKvFURSBLRVktLs0tL0o=
root@ip-172-31-16-166:~/ . kube# ^c
root@ip-172-31-16-166:~/ . kube#

```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

The screenshot shows the Jenkins Pipeline Syntax Snippet Generator interface. The URL in the address bar is [51.20.94.76:8080/job/DevOps%202/pipeline-syntax/](http://51.20.94.76:8080/job/DevOps%202/pipeline-syntax/). The page displays a configuration for a 'Kubernetes configuration (kubeconfig)' step. The 'Scope' is set to 'Global (Jenkins, nodes, items, all child items, etc.)'. The 'ID' is 'kubernetes'. The 'Content' field contains a large, obfuscated string representing a kubeconfig file, starting with 'Yraytbd2tFWmxFbS9kS1djZWJSWlg0RgpQWWl0QmVMa3hzRIA2aG...'. A checkbox for 'Enable Variable Substitution in Config' is checked.

The screenshot shows the Jenkins Pipeline Syntax Snippet Generator interface. The URL in the address bar is [51.20.94.76:8080/job/DevOps%202/pipeline-syntax/](http://51.20.94.76:8080/job/DevOps%202/pipeline-syntax/). The page displays a configuration for a 'Steps' section. Under 'Steps', there is a 'Sample Step' labeled 'kubernetesDeploy: Deploy to Kubernetes'. Below it, under 'kubernetesDeploy', is a 'Kubeconfig' section with an 'ID' of 'kubernetes'. A 'Config Files' field is present, and a checkbox for 'Enable Variable Substitution in Config' is checked.

The screenshot shows the Jenkins Pipeline configuration page. The pipeline script is defined as follows:

```
57+     withDockerRegistry(credentialsId: '33fbffea-4c21-415d-8f92-5c25467156b9', toolName: 'docker-')
58+     sh "docker tag cicddevops guhanxdocker/devopscicdx:$BUILD_ID"
59+     sh "docker push guhanxdocker/devopscicdx:$BUILD_ID"
60+
61    }
62  }
63 }
64 stage('DEPLOY TO K8s') {
65   steps {
66     script{
67       kubernetesDeploy (configs: 'deploymentservice.yaml', kubeconfigId: 'kubernetes')
68     }
69   }
70 }
71
72 }
73
```

Below the script, there is a checkbox labeled "Use Groovy Sandbox". At the bottom, there are "Save" and "Apply" buttons.

The screenshot shows the GitHub repository interface for the "DevOps-Pro2" branch. The "deploymentservice.yaml" file is being edited. The content of the file is as follows:

```
11  metadata:
12    labels:
13      app: spring-boot-k8s
14    spec:
15      containers:
16        - name: spring-boot-k8s
17          image: guhanxdocker/devopscicdx:7 # Image that will be used to containers in the cluster
18          imagePullPolicy: IfNotPresent
19        ports:
20          - containerPort: 8080 # The port that the container is running on in the cluster
21
22
23  ---
24
25  apiVersion: v1 # Kubernetes API version
26  kind: Service # Kubernetes resource kind we are creating
27  metadata: # Metadata of the resource kind we are creating
28  ...
29
```

At the top right of the editor, there are "Cancel changes" and "Commit changes..." buttons. The GitHub interface includes a sidebar with project navigation and a search bar at the top.

The screenshot shows a Windows desktop environment. At the top, there is a taskbar with various icons. An open browser window displays a Jenkins pipeline log for job DevOps 2, build #8. The log output is as follows:

```
k8s), sessionAffinity=NONE, sessionAffinityConfig=null, type=NODEPORT, additionalProperties={clusterIPs=[10.98.114.117], ipFamilies=[IPv4], ipFamilyPolicy=SingleStack, internalTrafficPolicy=Cluster}), status=ServiceStatus(loadBalancer=LoadBalancerStatus(ingress=[], additionalProperties={})), additionalProperties={}, additionalProperties={})  
Finished Kubernetes deployment  
[Pipeline] }  
[Pipeline] // script  
[Pipeline] }  
[Pipeline] // withEnv  
[Pipeline] }  
[Pipeline] // stage  
[Pipeline] }  
[Pipeline] // withEnv  
[Pipeline] }  
[Pipeline] // withEnv  
[Pipeline] }  
[Pipeline] // node  
[Pipeline] End of Pipeline  
Finished: SUCCESS
```

At the bottom right of the screen, system tray icons show the date as 01-11-2023 and the time as 17:47.

The screenshot shows a Windows desktop environment. At the top, there is a taskbar with various icons. An open browser window displays a Jenkins build summary for build #8, dated Nov 1, 2023, at 12:15:11 PM. The summary includes:

- Status: Build #8 (Nov 1, 2023, 12:15:11 PM)
- Started by user GuhanS
- Started 1 min 58 sec ago
- Took 1 min 2 sec
- Console Output link
- Edit Build Information link
- Delete build '#8' link
- Timings link
- Git Build Data link
- Restart from Stage link
- Replay link
- Pipeline Steps link

At the bottom right of the screen, system tray icons show the date as 01-11-2023 and the time as 17:47.

Now we can see our Application got build and successfully deployed to Kubernetes cluster. You can also copy the ip and paste it in the browser for checking the Application. This is how the complete CICD implementation works in DevOps.