

Name: S.Guhan Balaji

CI/CD Implementation using Jenkins, Kubernetes, ArgoCD, Sonarqube, Maven

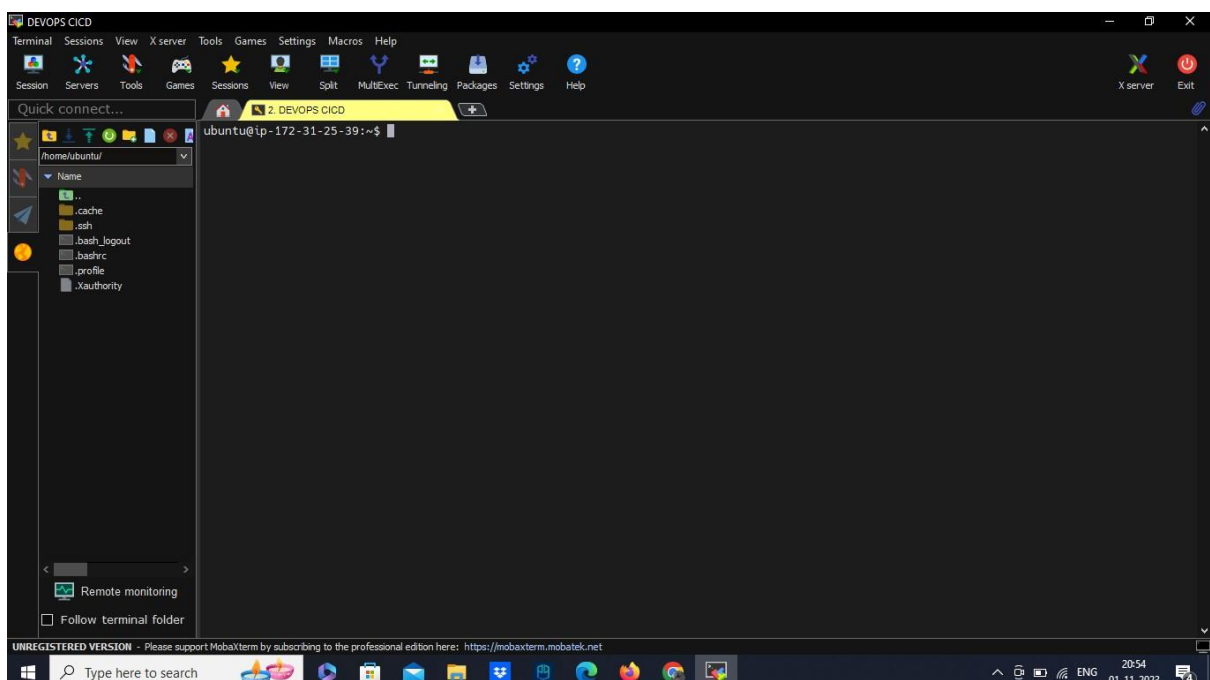
Introduction:

In this project we are going to build and deploy to the Kubernetes cluster using Jenkins, Kubernetes, ArgoCD. This process allows for automated and streamlined software development and deployment, ensuring that your applications are built, tested, and deployed consistently and efficiently.

Steps Involved:

Step 1: AWS

- Launch an instance. Select the ami as Ubuntu.
- Select the instance type as large, to handle heavy work.
- Here I am using MobaXterm, to connect to my instance. Ensure that port is 22.

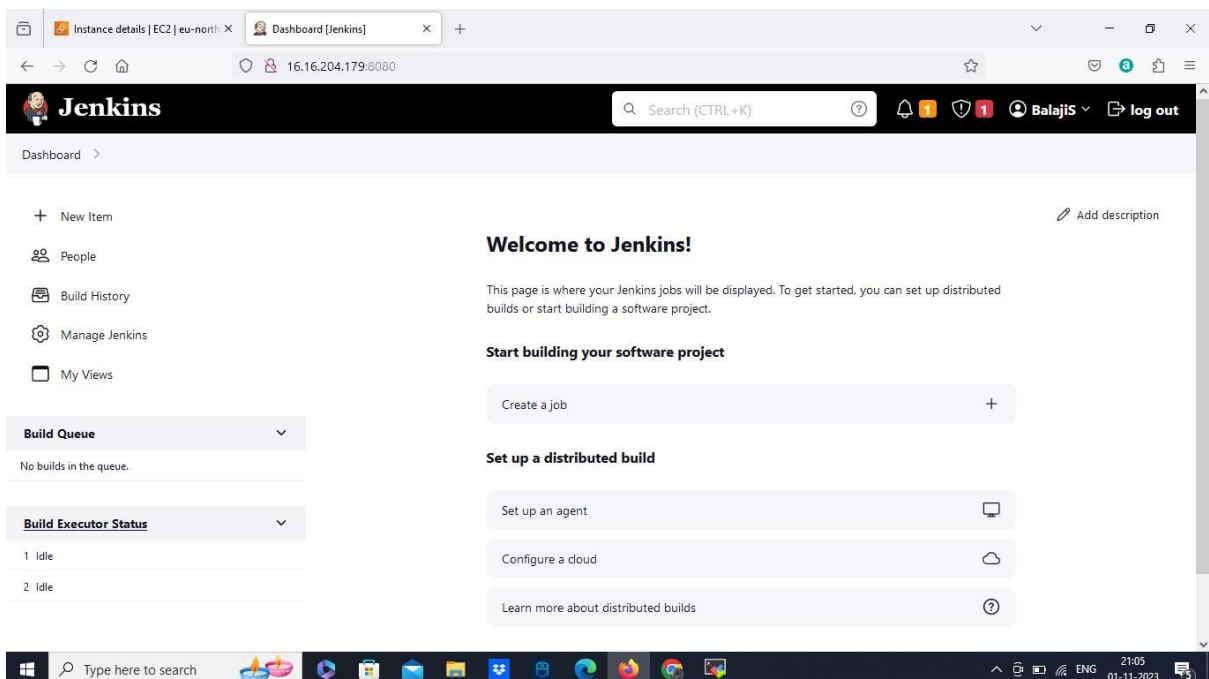


- Copy the public ip of your aws instance and launch it in your terminal using private key you have. Finally login as Ubuntu.

Step 2: Installation/Login

Here we need to install Jenkins for our server.

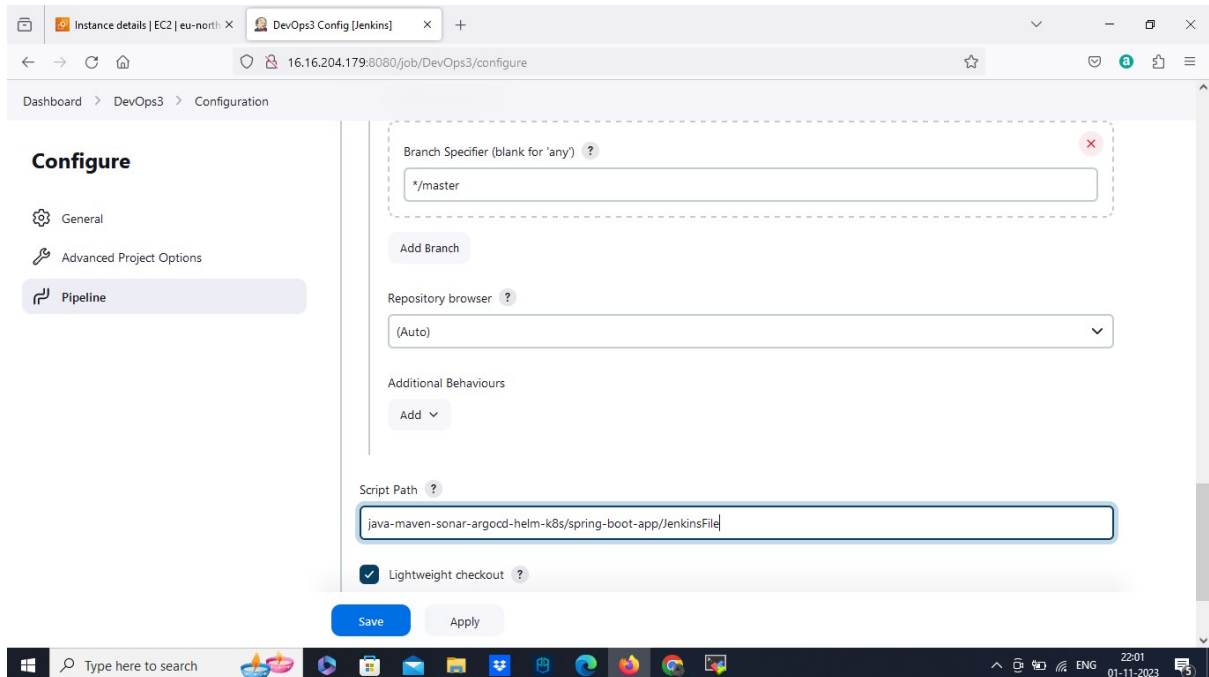
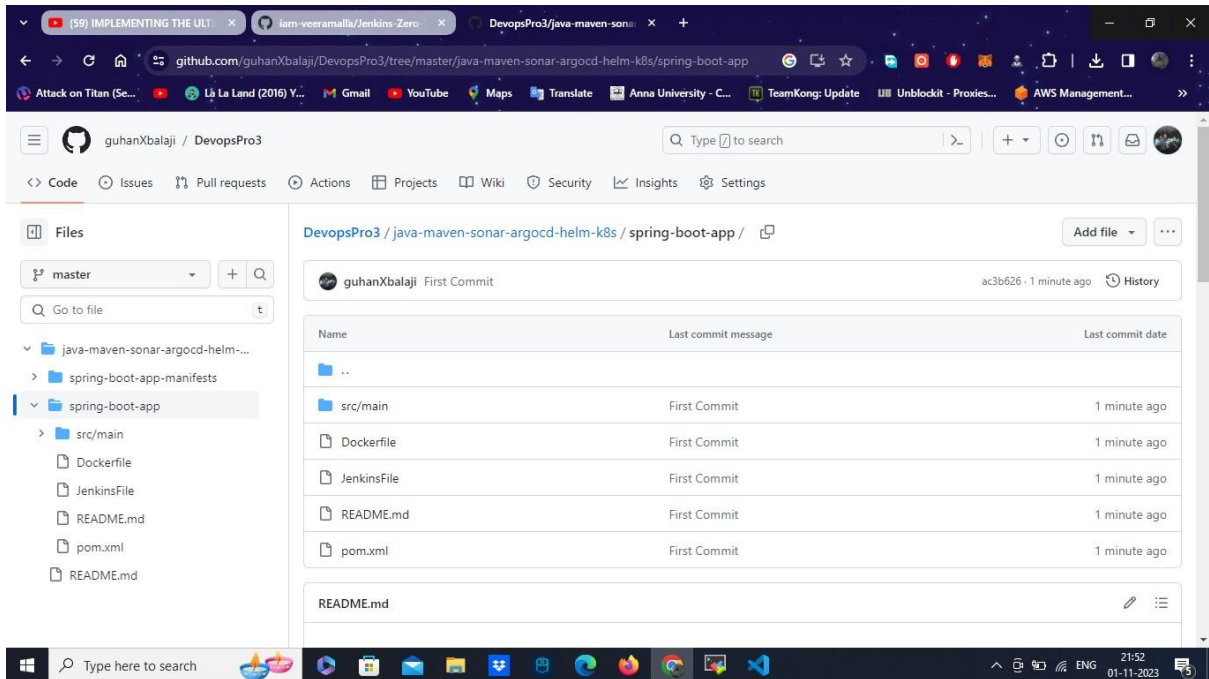
- Jenkins
 - `sudo apt update`
 - `sudo apt install openjdk-11-jre`
 - `java -version`
 - `curl -fsSL https://pkg.jenkins.io/debian/jenkins.io-2023.key | sudo tee \`
`/usr/share/keyrings/jenkins-keyring.asc > /dev/null`
 - `echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \`
`https://pkg.jenkins.io/debian binary/ | sudo tee \`
`/etc/apt/sources.list.d/jenkins.list > /dev/null`
 - `sudo apt-get update`
 - `sudo apt-get install Jenkins`
 - For the login password - `sudo cat /var/lib/jenkins/secrets/initialAdminPassword`
- Copy the public IP and hit in the browser `http:// :8080`
- Now Login to the Jenkins with the user name and password. Also install necessary plugins.



Step 3: Create the pipeline:

- Enter a name to the pipeline

- Here we are going to use pipeline script from SCM. Where we are going to add predefined Jenkins file contains predefined script.
- Select scm as Git. Provide the git repository URL where our Application present.
- Select branch as master/main.
- Add the path to the jenkinsfile in our git repository. And save it.

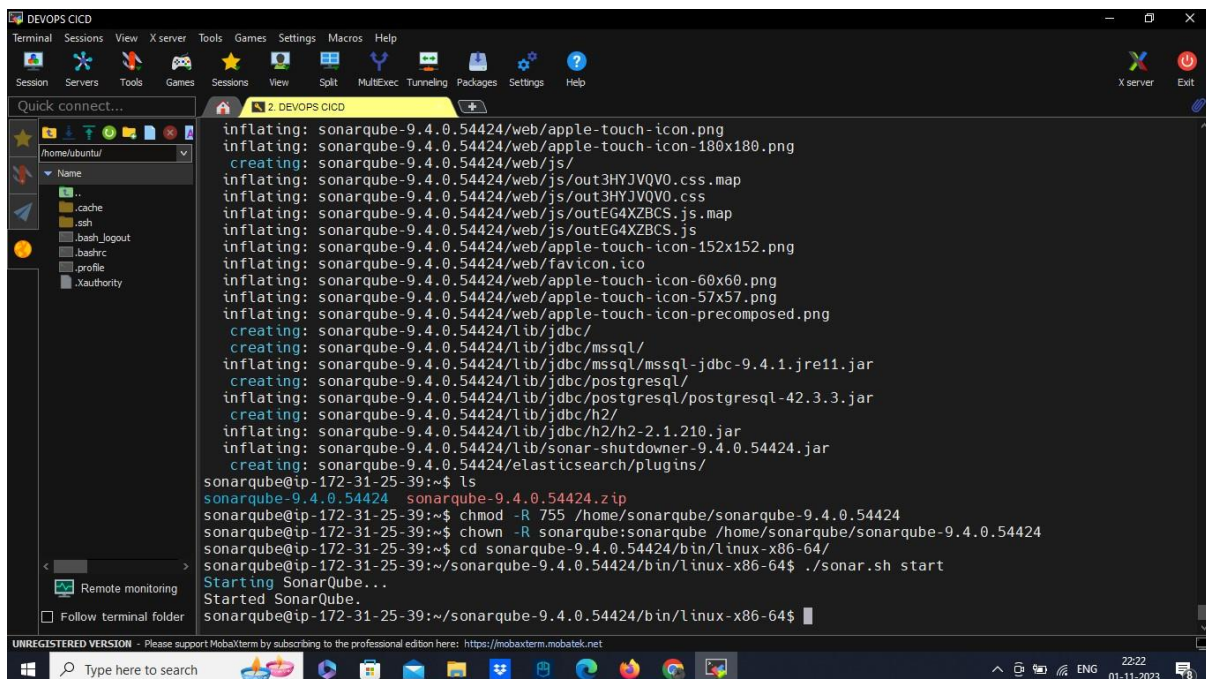


Step 4: Plugins

- Now go to manage plugins in your Jenkins.
- Locate available plugins, search docker pipeline plugin and install it.
- Install another plugin called sonar qube scanner in your instance. And restart the Jenkins.

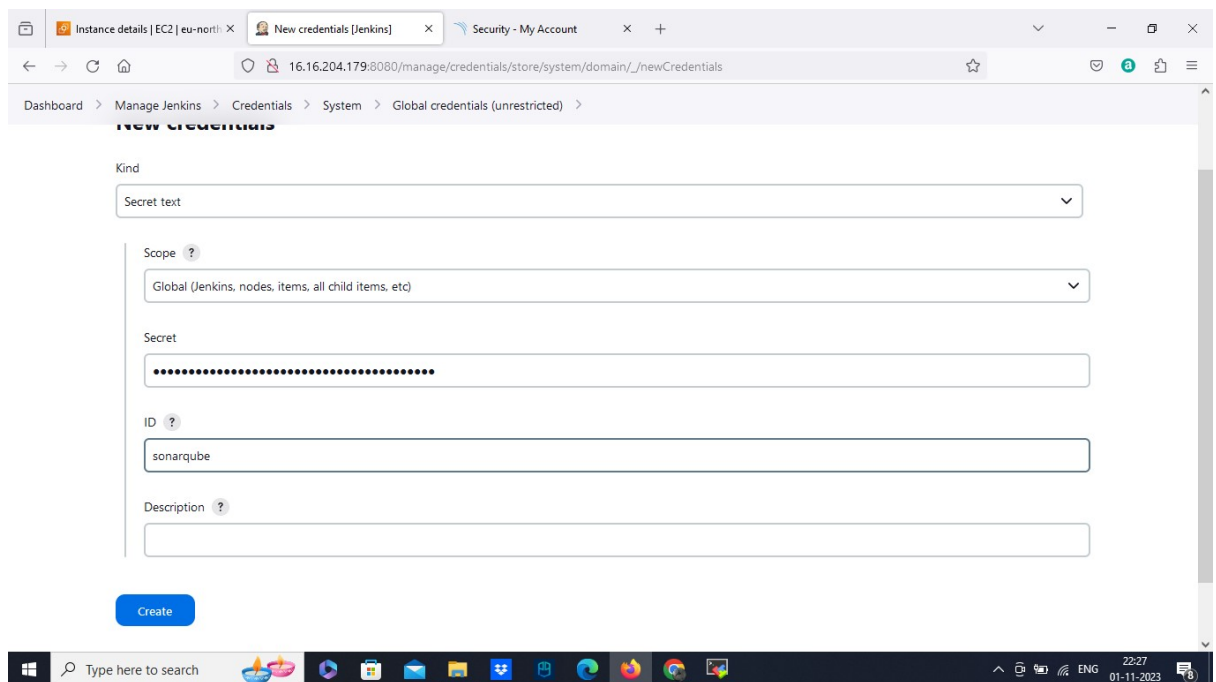
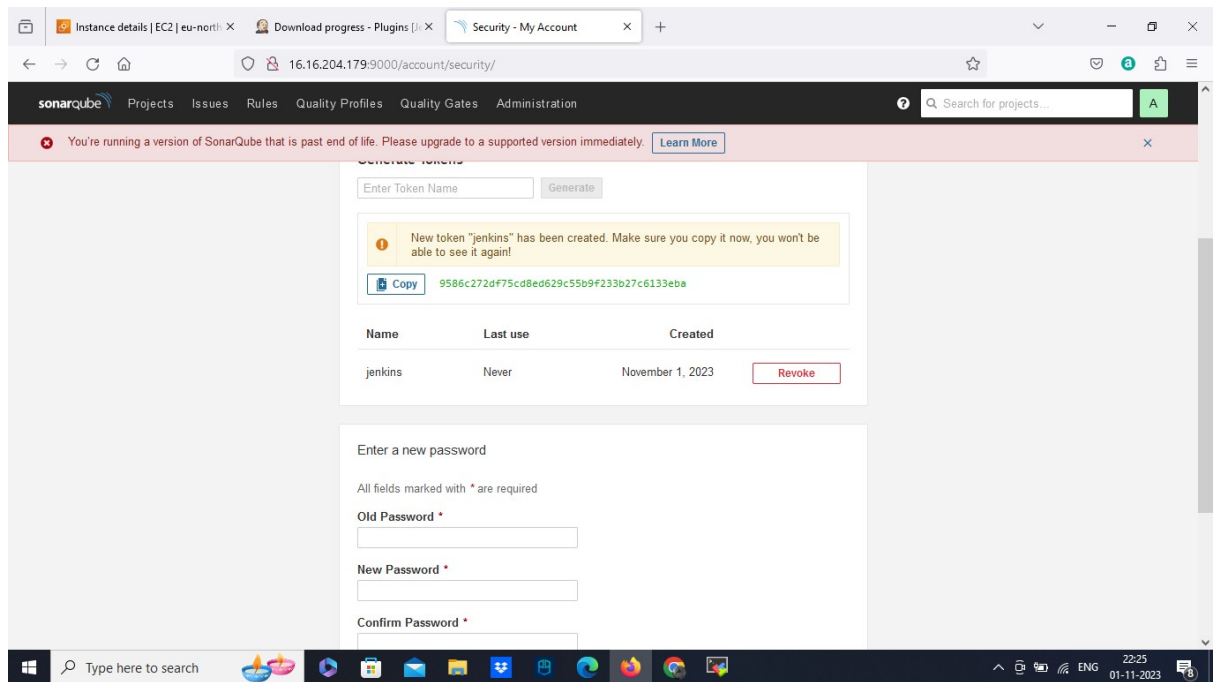
Step 5: Install Sonarqube and login

- Now go to the terminal of our instance and install sonarqube
 1. apt install unzip
 2. adduser sonarqube
 3. wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.4.0.54424.zip
 4. unzip *
 5. chmod -R 755 /home/sonarqube/sonarqube-9.4.0.54424
 6. chown -R sonarqube:sonarqube /home/sonarqube/sonarqube-9.4.0.54424
 7. cd sonarqube-9.4.0.54424/bin/linux-x86-64/
 8. ./sonar.sh start



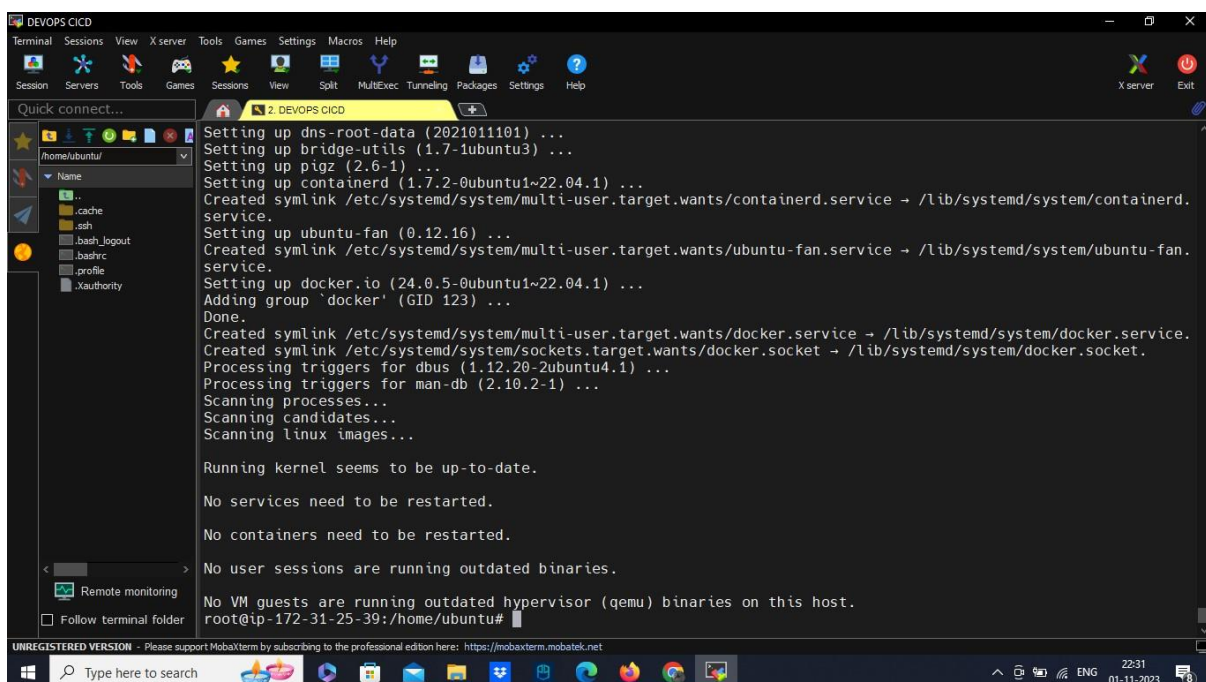
```
DEVOPS CICD
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultExec Tunneling Packages Settings Help
Quick connect...
/home/ubuntu/
Name
..
.cache
.ssh
.bash_logout
.bashrc
.profile
.xauthority
inflatng: sonarqube-9.4.0.54424/web/apple-touch-icon.png
inflatng: sonarqube-9.4.0.54424/web/apple-touch-icon-180x180.png
creatng: sonarqube-9.4.0.54424/web/js/
inflatng: sonarqube-9.4.0.54424/web/js/out3HYJVQV0.css.map
inflatng: sonarqube-9.4.0.54424/web/js/out3HYJVQV0.css
inflatng: sonarqube-9.4.0.54424/web/js/outEG4XZBCS.js.map
inflatng: sonarqube-9.4.0.54424/web/js/outEG4XZBCS.js
inflatng: sonarqube-9.4.0.54424/web/apple-touch-icon-152x152.png
inflatng: sonarqube-9.4.0.54424/web/favicon.ico
inflatng: sonarqube-9.4.0.54424/web/apple-touch-icon-60x60.png
inflatng: sonarqube-9.4.0.54424/web/apple-touch-icon-57x57.png
inflatng: sonarqube-9.4.0.54424/web/apple-touch-icon-precomposed.png
creatng: sonarqube-9.4.0.54424/lib/jdbc/
creatng: sonarqube-9.4.0.54424/lib/jdbc/mssql/
inflatng: sonarqube-9.4.0.54424/lib/jdbc/mssql/mssql-jdbc-9.4.1.jre11.jar
creatng: sonarqube-9.4.0.54424/lib/jdbc/postgresql/
inflatng: sonarqube-9.4.0.54424/lib/jdbc/postgresql/postgresql-42.3.3.jar
creatng: sonarqube-9.4.0.54424/lib/jdbc/h2/
inflatng: sonarqube-9.4.0.54424/lib/jdbc/h2/h2-2.1.210.jar
inflatng: sonarqube-9.4.0.54424/lib/sonar-shutdowner-9.4.0.54424.jar
creatng: sonarqube-9.4.0.54424/elasticsearch/plugins/
sonarqube@ip-172-31-25-39:~$ ls
sonarqube-9.4.0.54424 sonarqube-9.4.0.54424.zip
sonarqube@ip-172-31-25-39:~$ chmod -R 755 /home/sonarqube/sonarqube-9.4.0.54424
sonarqube@ip-172-31-25-39:~$ chown -R sonarqube:sonarqube /home/sonarqube/sonarqube-9.4.0.54424
sonarqube@ip-172-31-25-39:~$ cd sonarqube-9.4.0.54424/bin/linux-x86-64/
sonarqube@ip-172-31-25-39:~/sonarqube-9.4.0.54424/bin/linux-x86-64$ ./sonar.sh start
Starting SonarQube...
Started SonarQube.
sonarqube@ip-172-31-25-39:~/sonarqube-9.4.0.54424/bin/linux-x86-64$
```

- Copy the public ip of our instance and hit the browser http:// :9000.
- Go to administration/ security. Where we can generate a sonarqube token for our Jenkins.
- Go back to Jenkins, Locate credentials and add new credentials as secret text.
- Paste the token from sonarqube in secret field and provide the name.



Step 6: Docker

- Install Docker in our Instance and granting access which contains Jenkins.
 - `sudo apt update`
 - `sudo apt install docker.io`
 - `sudo su -`
 - `usermod -aG docker jenkins`
 - `usermod -aG docker ubuntu`
 - `systemctl restart docker`



```
Setting up dns-root-data (2021011101) ...
Setting up bridge-utils (1.7-1ubuntu3) ...
Setting up pigz (2.6-1) ...
Setting up containerd (1.7.2-0ubuntu1~22.04.1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /lib/systemd/system/containerd.service.
Setting up ubuntu-fan (0.12.16) ...
Created symlink /etc/systemd/system/multi-user.target.wants/ubuntu-fan.service → /lib/systemd/system/ubuntu-fan.service.
Setting up docker.io (24.0.5-0ubuntu1~22.04.1) ...
Adding group `docker' (GID 123) ...
Done.
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /lib/systemd/system/docker.socket.
Processing triggers for dbus (1.12.20-2ubuntu4.1) ...
Processing triggers for man-db (2.10.2-1) ...
Scanning processes...
Scanning candidates...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

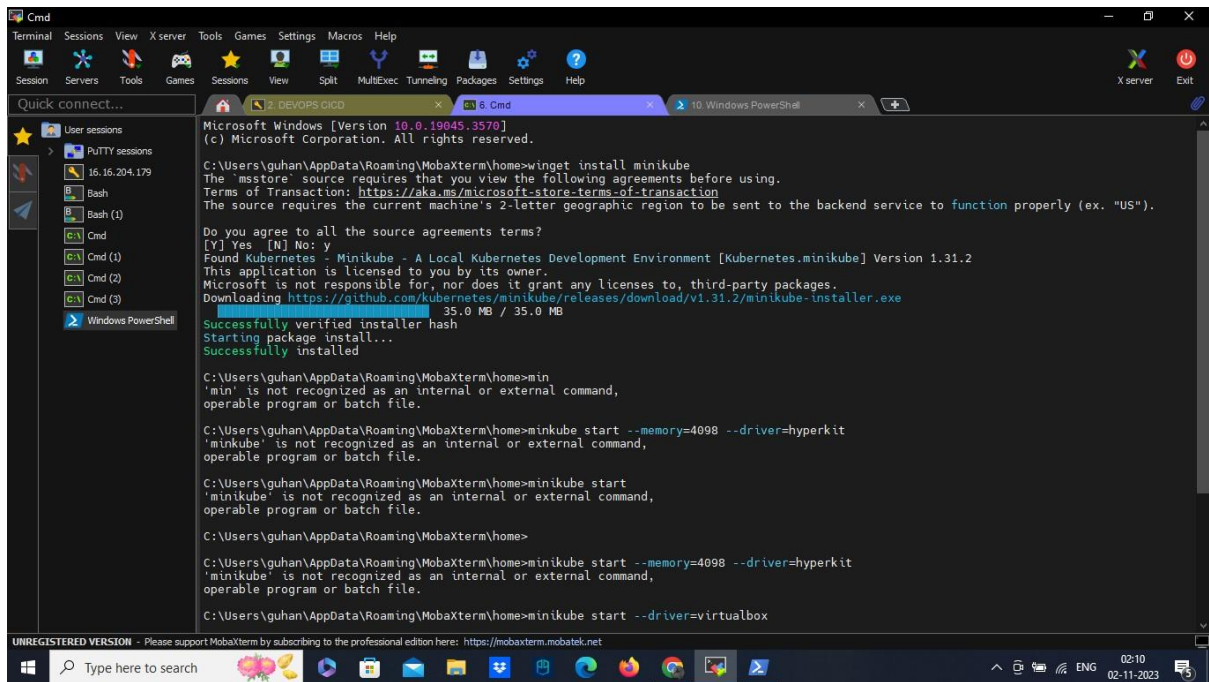
No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@ip-172-31-25-39:/home/ubuntu#
```

Step 7: Install Minikube

- Now we have to install minikube in our computer.
- I am Using MobaXterm cmd to install the minikube in my computer.
 - `winget install minikube`
- Open Docker desktop and run docker engine.
- Use other cmd prompt starting minikube cluster (I am using Git Cmd here)
 - `minikube start`



```
Microsoft Windows [Version 10.0.19045.3570]
(c) Microsoft Corporation. All rights reserved.

C:\Users\guhan\AppData\Roaming\MobaXterm\home>winget install minikube
The 'msstore' source requires that you view the following agreements before using.
Terms of Transaction: https://aka.ms/microsoft-store-terms-of-transaction
The source requires the current machine's 2-letter geographic region to be sent to the backend service to function properly (ex. "US").

Do you agree to all the source agreements terms?
[Y] Yes [N] No: y
Found Kubernetes - Minikube - A Local Kubernetes Development Environment [Kubernetes.minikube] Version 1.31.2
This application is licensed to you by its owner.
Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.
Downloading https://github.com/kubernetes/minikube/releases/download/v1.31.2/minikube-installer.exe
35.0 MB / 35.0 MB
Successfully verified installer hash
Starting package install...
Successfully installed

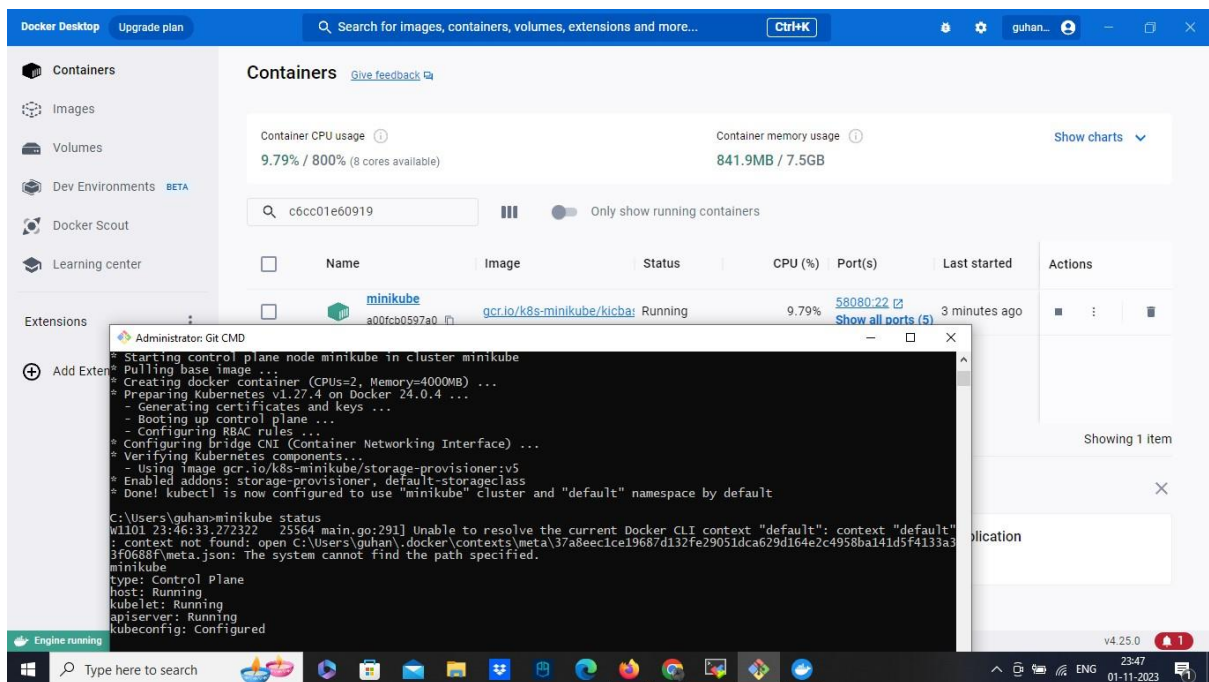
C:\Users\guhan\AppData\Roaming\MobaXterm\home>min
'min' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\guhan\AppData\Roaming\MobaXterm\home>minikube start --memory=4098 --driver=hyperkit
'minikube' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\guhan\AppData\Roaming\MobaXterm\home>minikube start
'minikube' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\guhan\AppData\Roaming\MobaXterm\home>minikube start --memory=4098 --driver=hyperkit
'minikube' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\guhan\AppData\Roaming\MobaXterm\home>minikube start --driver=virtualbox
```



Step 8: Argo CD

- Install Argo cd by using powershell in your computer (here iam using mobaxterm powershell). Also create namespace as argocd. [?](#)
 - `kubectl create namespace argocd`
 - `kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml`

```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\guhan\AppData\Roaming\MobaXterm\home> kubectl get all
NAME                                TYPE                CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
service/kubernetes                  ClusterIP           10.96.0.1     <none>         443/TCP    46m
PS C:\Users\guhan\AppData\Roaming\MobaXterm\home> kubectl create namespace argocd
namespace/argocd created
PS C:\Users\guhan\AppData\Roaming\MobaXterm\home> kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml
customresourcedefinition.apiextensions.k8s.io/applications.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/appprojects.argoproj.io created
serviceaccount/argocd-application-controller created
serviceaccount/argocd-applicationset-controller created
serviceaccount/argocd-dex-server created
serviceaccount/argocd-notifications-controller created
serviceaccount/argocd-repo-server created
serviceaccount/argocd-server created
role.rbac.authorization.k8s.io/argocd-application-controller created
role.rbac.authorization.k8s.io/argocd-applicationset-controller created
role.rbac.authorization.k8s.io/argocd-dex-server created
role.rbac.authorization.k8s.io/argocd-notifications-controller created
role.rbac.authorization.k8s.io/argocd-server created
clusterrole.rbac.authorization.k8s.io/argocd-application-controller created
clusterrole.rbac.authorization.k8s.io/argocd-server created
rolebinding.rbac.authorization.k8s.io/argocd-application-controller created
rolebinding.rbac.authorization.k8s.io/argocd-applicationset-controller created
rolebinding.rbac.authorization.k8s.io/argocd-dex-server created
rolebinding.rbac.authorization.k8s.io/argocd-notifications-controller created
rolebinding.rbac.authorization.k8s.io/argocd-server created
clusterrolebinding.rbac.authorization.k8s.io/argocd-application-controller created
clusterrolebinding.rbac.authorization.k8s.io/argocd-server created
configmap/argocd-cm created
configmap/argocd-cm-params-cm created
  
```

Step 9: Credentials

- We have to add credentials for tools to get accessed from the Jenkins file
- Go to manage credentials in Jenkins, add credentials for docker hub.
- Enter the username and password of docker hub. And add the name in the id, the name you provided in Jenkins file.
- Next one for Git Hub credentials. For that choose secret text.

Jenkins

Dashboard > Manage Jenkins > Credentials

Global credentials

Credentials that should be available to all builds:

ID	Description	Kind
sonarqube	sonarqube	Secret text
docker-cred	guhanxdocker/*****	Username with password
github	github	Secret text

Save password for http://16.16.204.179:8080/?

Username: No username

Password: *****

Kind: Secret text

Buttons: Save, Don't save, Add Credentials

- Add the git hub token you generated in the secret field and give the id name, the name you provided in script.
- Save it and restart it.

Step 10: Build

- Before start the build, we need to do a minor change.
- Go to our application git repository, locate the Jenkins file and add our sonarqube URL in that file.
- Commit and save it.
- Now go back to Jenkins and build now. And will get built.

The screenshot shows the Jenkins web interface for a pipeline named 'DevOps3'. The 'Stage View' is active, displaying a table of stages and their execution times for two builds.

Stage	Build #2 (Nov 02 01:23)	Build #1 (Nov 02 01:19)
Declarative: Checkout SCM	919ms	757ms
Checkout	414ms	368ms
Build and Test	19s	19s
Static Code Analysis	25s	23s
Build and Push Docker Image	17s	13s
Update Deployment File	1s	1s

Average stage times: 919ms, 414ms, 19s, 25s, 17s, 1s. (Average full run time: ~1min 3s)

The screenshot shows the SonarQube web interface. A message at the top indicates that the current version is past its end of life and should be upgraded. The 'Projects' tab is active, showing a list of projects. The 'spring-boot-demo' project is highlighted, showing a 'Passed' status and a last analysis time of 1 minute ago.

Project: spring-boot-demo (Passed)

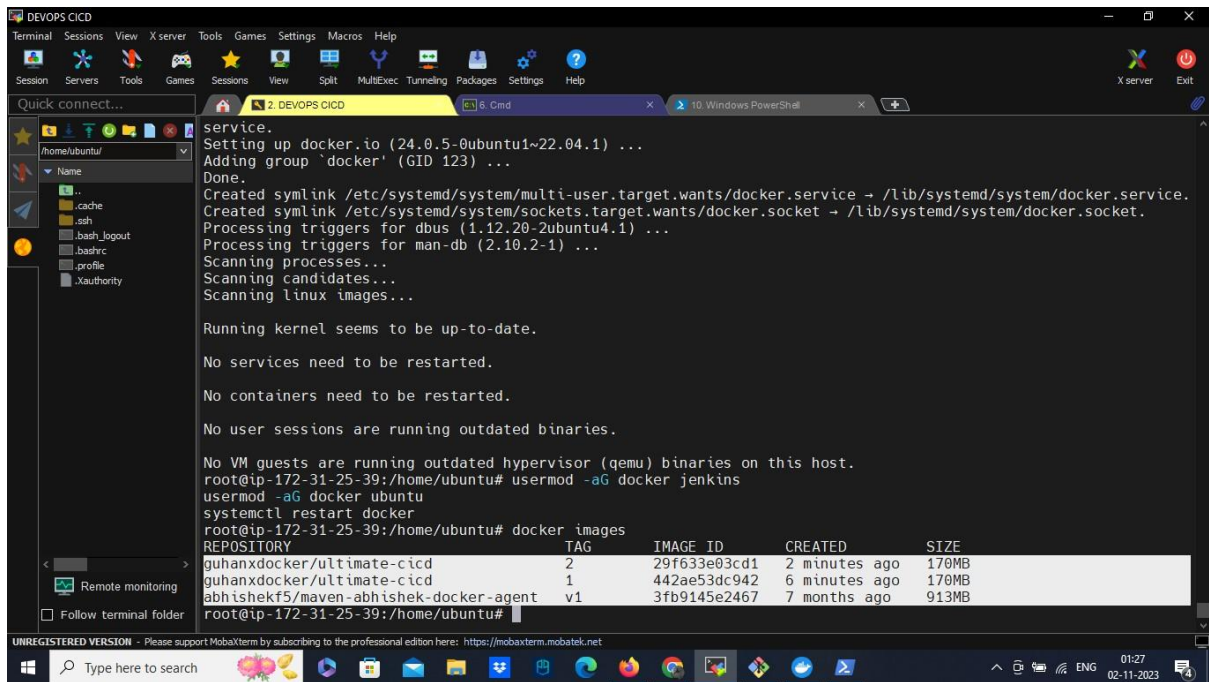
Last analysis: 1 minute ago

Metric	Value	Quality Gate
Bugs	0	A
Vulnerabilities	0	A
Hotspots Reviewed	0	A
Code Smells	0	A
Coverage	0.0%	F
Duplications	0.0%	F
Lines	79	A

1 of 1 shown

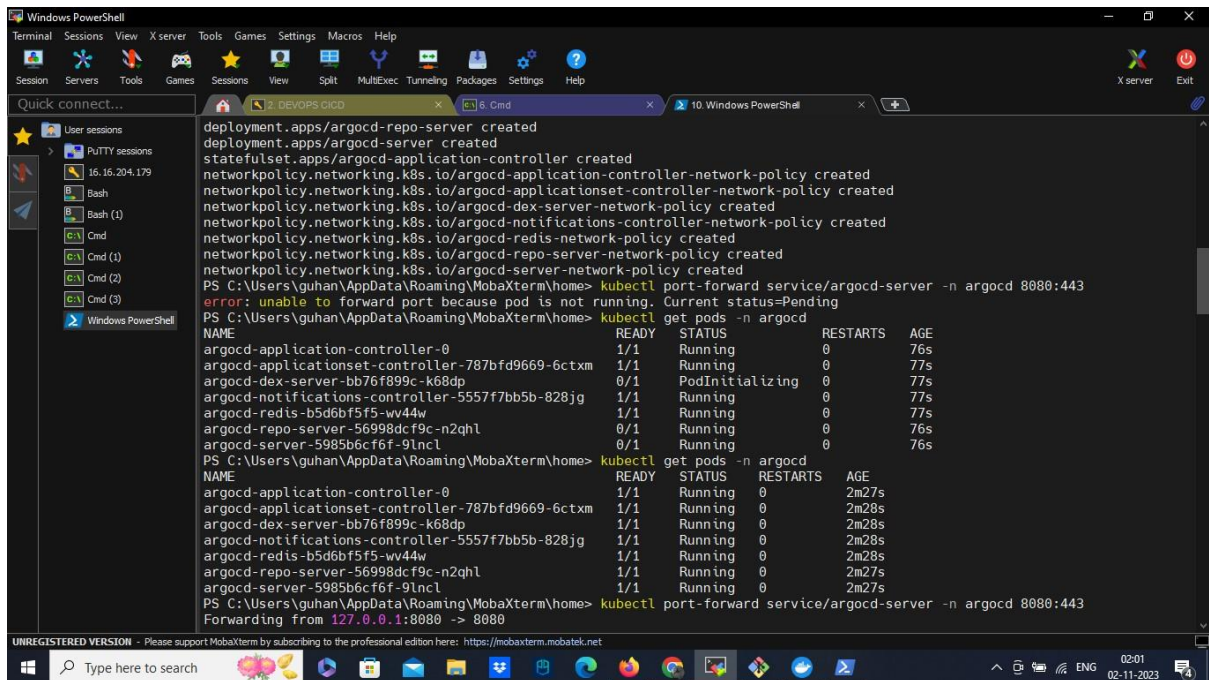
Embedded database should be used for evaluation purposes only
The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.

SonarQube™ technology is powered by SonarSource SA
Community Edition - Version 9.4 (build 54424) - LGPL v3 - Community - Documentation - Plugins - Web API

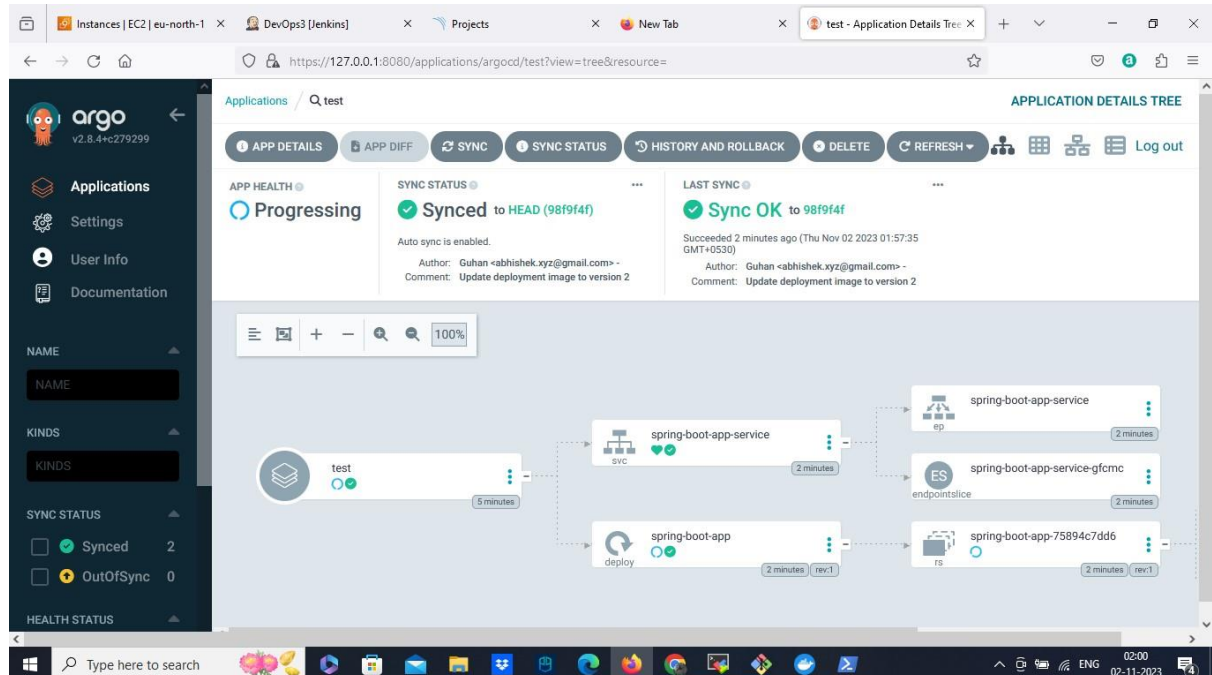
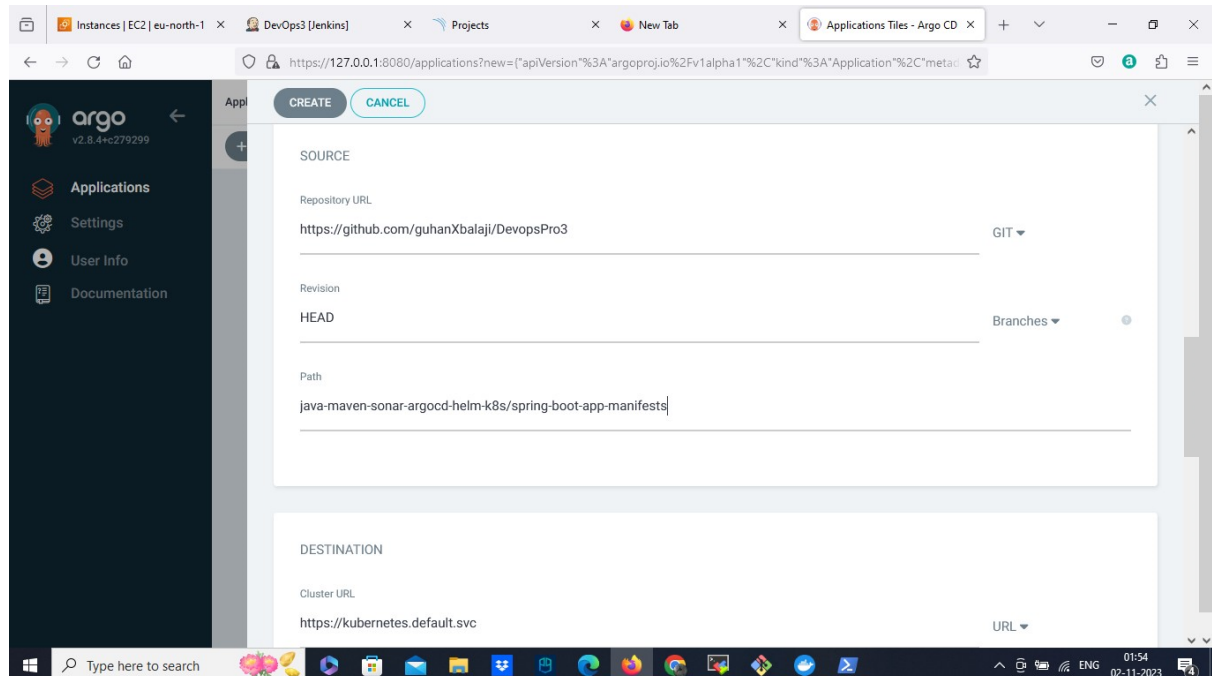


Step 11: ArgoCD

- First we need to change the argocd to run in our browser.
- To do that use kubectl get svc and edit the argocd server with kubectl edit svc name.
- It get opened with notepad, change the type from cluster ip to node port. Save it.
- Now use minikube service name and minikube service list, You can get URL for argo CD now. And also check the pods using kubectl get pods.



- After login Argocd. Select create application.
- Provide the application name, add default as project name. make sync policy automatic.
- Add the application git hub repo URL, add the path of yaml file from the repository.
- Select the cluster url and add the namespace as default. And it will get created. You can also check with kubectl get pods.



In summary, our implementation of CI/CD with Jenkins, Maven, SonarQube, Docker, Kubernetes, and ArgoCD has had a profound impact on our project. This is how the CICD implementation works. It has made our development and deployment processes more efficient, elevated the quality of our applications, and significantly improved their reliability.