Chat Bot deployment in Kubernetes

Launch a EC2 Ubuntu Instance:

Launch a Ubuntu(22.04) T2 Large Instance with 30 GB storage. Add HTTP and HTTPS settings in the security group and open the nessasary ports.

Create a IAM Role:

Search “IAM” in AWS console and click on “Rules”. Click on “Create rule” button.

Select “AWS service” in Trusted entity type.

Select “EC2” in Use case and click next.

Select “Administrator Access” and click next.

Give a name to the role as “chatbot” and create the role.

Attach the role to the EC2 instance:

Select the ec2 instance. Go to “Actions” -> “Security” -> “Modify IAM Role”.

Select the role that is newly created and click “Update IAM role” button.

Now connect to the EC2 instance using Mobaxterm.

Installation of Jenkins:

In the EC2 console, do the below:

```
vi jenkins.sh
```

```
#!/bin/bash

sudo apt update -y

wget -O - https://packages.adoptium.net/artifactory/api/gpg/key/public | tee /etc/apt/keyrings/adoptium.asc

echo "deb [signed-by=/etc/apt/keyrings/adoptium.asc] https://packages.adoptium.net/artifactory/deb $(awk -F= '/^VERSION_CODENAME/{print$2}' /etc/os-release) main" | tee /etc/apt/sources.list.d/adoptium.list

sudo apt update -y

sudo apt install temurin-17-jdk -y

/usr/bin/java --version

curl -fsSL https://pkg.jenkins.io/debian-stable/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-stable binary/ | sudo tee \
                /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update -y
sudo apt-get install jenkins -y
sudo systemctl start jenkins
```

Give execution permission to the jenkins.sh file:

```
sudo chmod 777 jenkins.sh
sudo su
./jenkins.
```

Install Docker:

```
sudo apt-get update
sudo apt-get install docker.io -y
sudo usermod -aG docker $USER   #my case is ubuntu
newgrp docker
sudo chmod 777 /var/run/docker.sock
```

Install Sonarqube using a docker container:

```
docker run -d --name sonar -p 9000:9000 sonarqube:lts-community
```

Install Trivy, Kubectl, Terraform:

```
vi script.sh
```

```
sudo apt-get install wget apt-transport-https gnupg lsb-release -y
wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | gpg --dearmor | sudo tee /usr/share/keyrings/trivy.gpg > /dev/null
echo "deb [signed-by=/usr/share/keyrings/trivy.gpg] https://aquasecurity.github.io/trivy-repo/deb $(lsb_release -sc) main" | sudo tee -a /etc/apt/sources.list.d/trivy.list
sudo apt-get update
sudo apt-get install trivy -y
# Install Terraform
sudo apt install wget -y
wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg
```

```
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg]
https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee
/etc/apt/sources.list.d/hashicorp.list

sudo apt update && sudo apt install terraform

# Install kubectl

sudo apt update

sudo apt install curl -y

curl -LO https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl

sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

kubectl version --client

# Install AWS CLI

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"

sudo apt-get install unzip -y

unzip awscliv2.zip

sudo ./aws/install
```

Give permission and run script:

```
sudo chmod 777 script.sh

./script.sh
```

Jenkins Dashboard:

Now login to the Jenkins Dashboard by using <ec2 public ip address>:8080 in a browser.

Select “Install suggested plugins”.

Create First Admin User and create a user click on save and continue.

Sonarqube Dashboard:

Enter username as “admin” & password as “admin”

In Jenkins Dashboard:

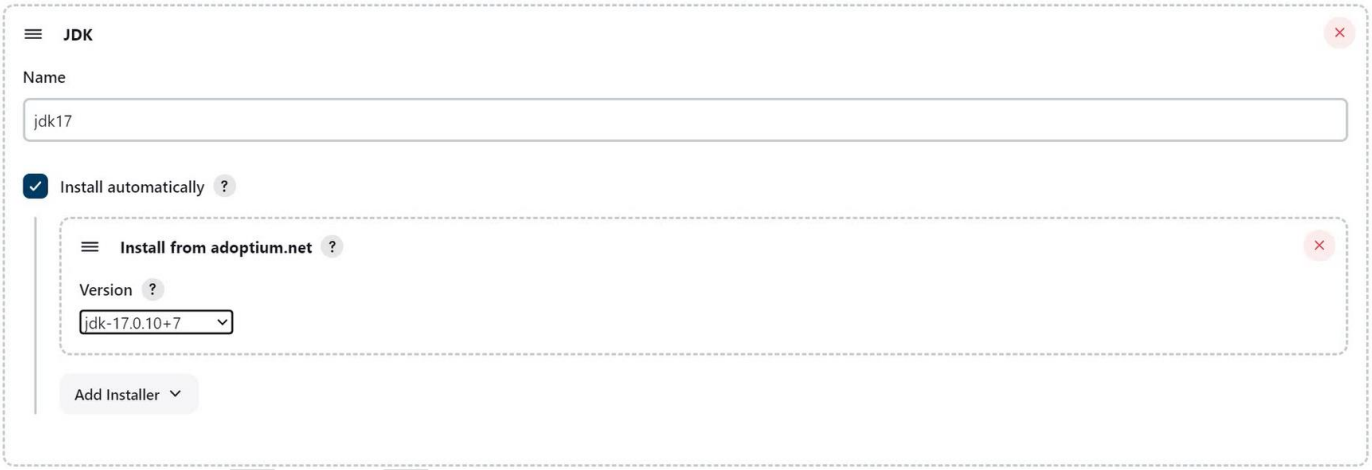
Install the below Plugins: Go to “Manage Jenkins” -> “Plugins” -> “Available Plugins

1. Eclipse Temurin Installer
2. SonarQube Scanner
3. Nodejs
4. Docker
5. Docker commons
6. Docker pipeline
7. Docker API
8. Docker build step
9. QWASP dependency check
10. Terraform
11. Kubernetes
12. Kubernetes CLI
13. Kubernetes client API
14. Kubernetes :: Pipeline :: Devops steps
15. Kubernetes Credentials
16. Kubernetes credentials provider
17. Blue Ocean

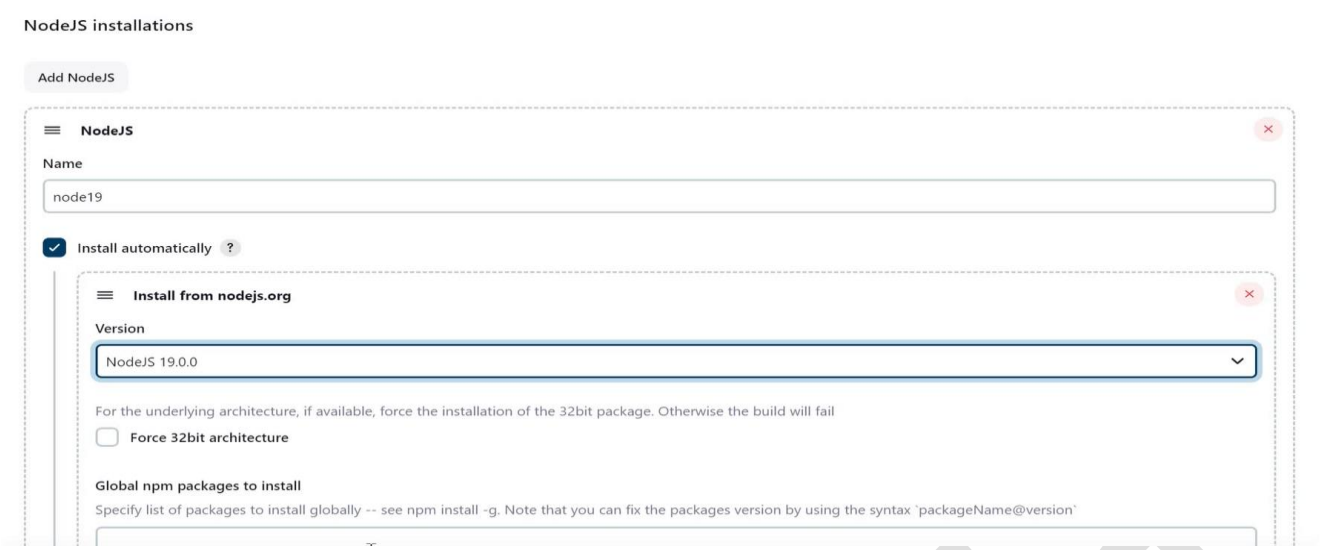
Configure Java and nodejs in Global Tool Configuration:

Go to “Manage Jenkins” -> “Tools”

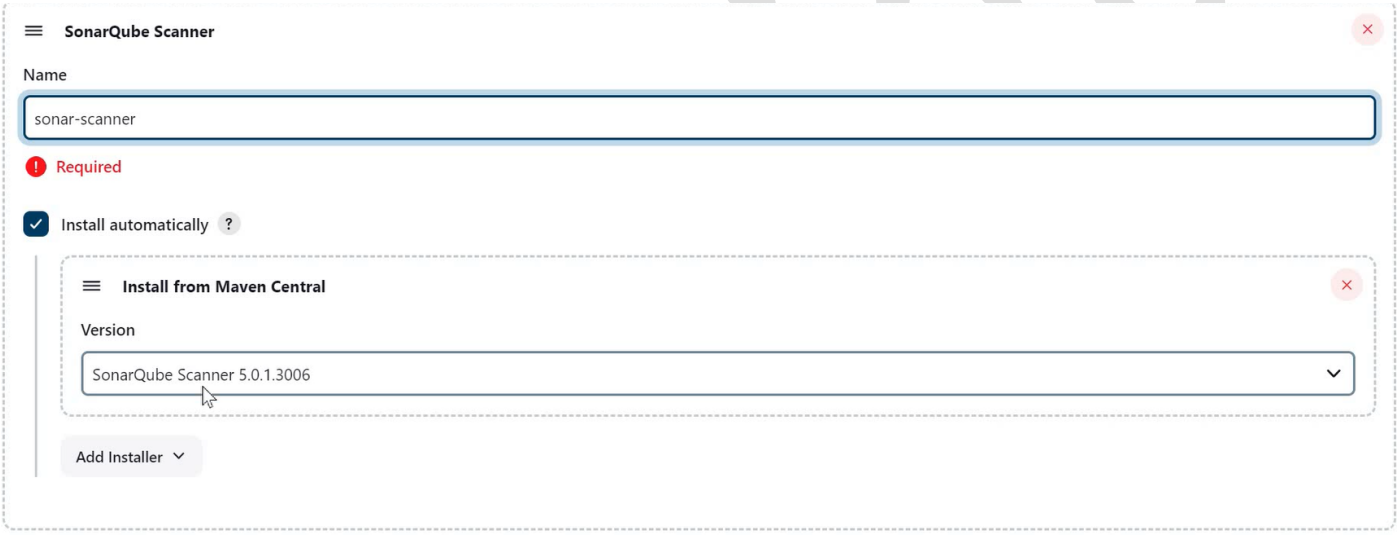
Click “**Add JDK**”



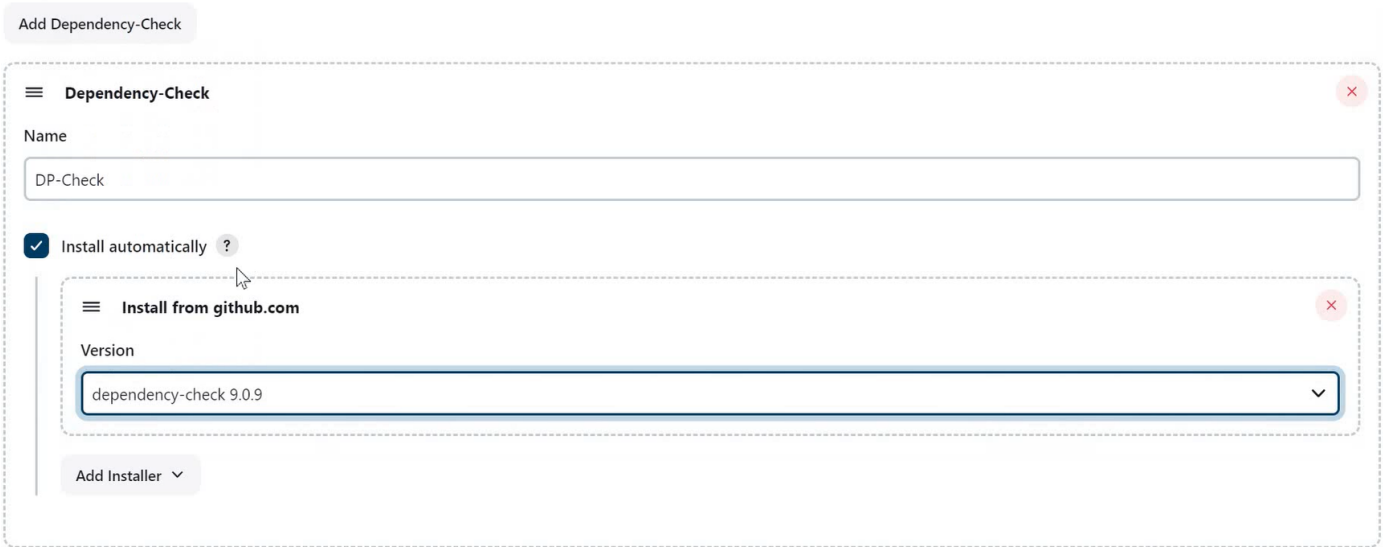
Click “**Add NodeJS**”



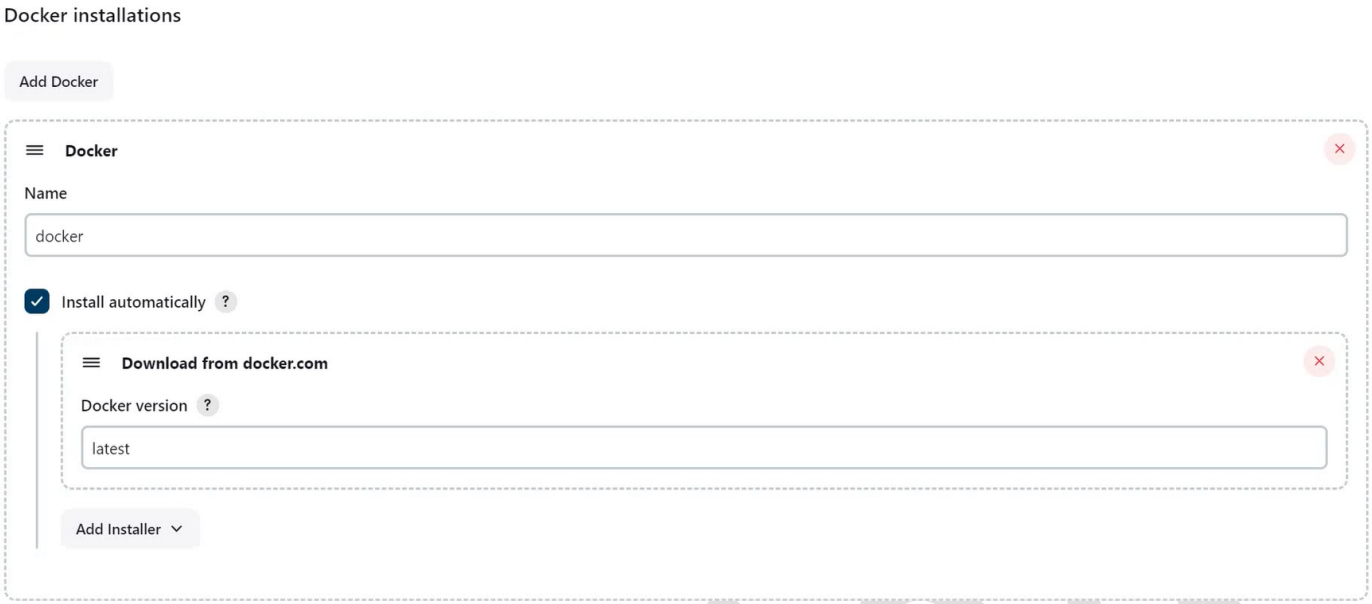
Click “**Add SonarQube**”



Click “**Add Dependency-Check**”



Click “**Add Docker**”

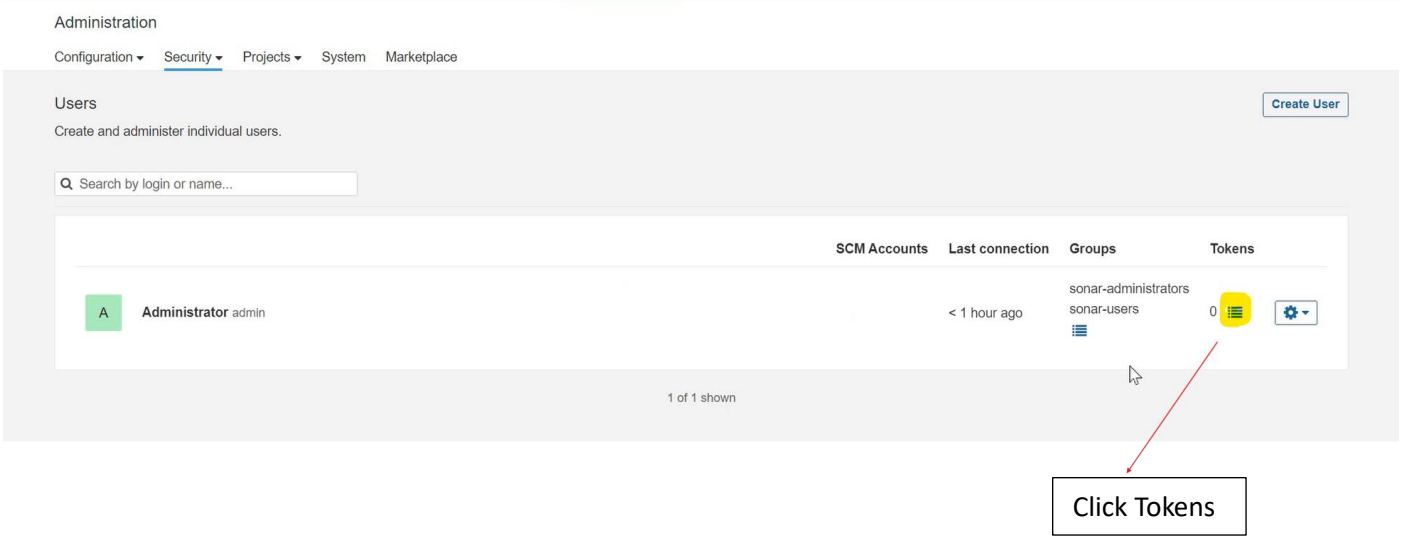


Click “**Add Terraform**”

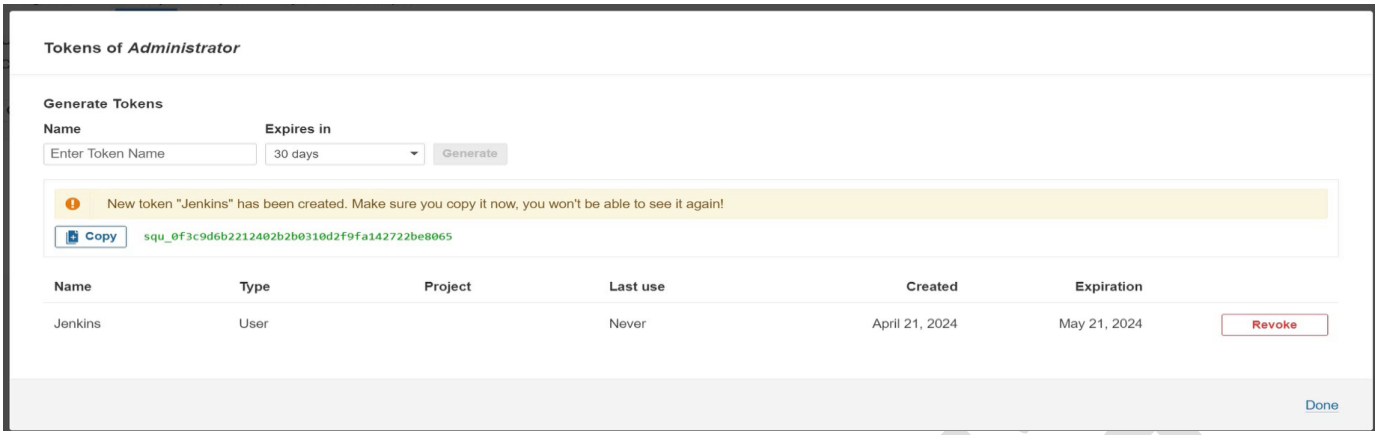
Click Apply and Save.

In Sonarqube Dashboard:

Click on “Administration” -> “Security” -> “Users”

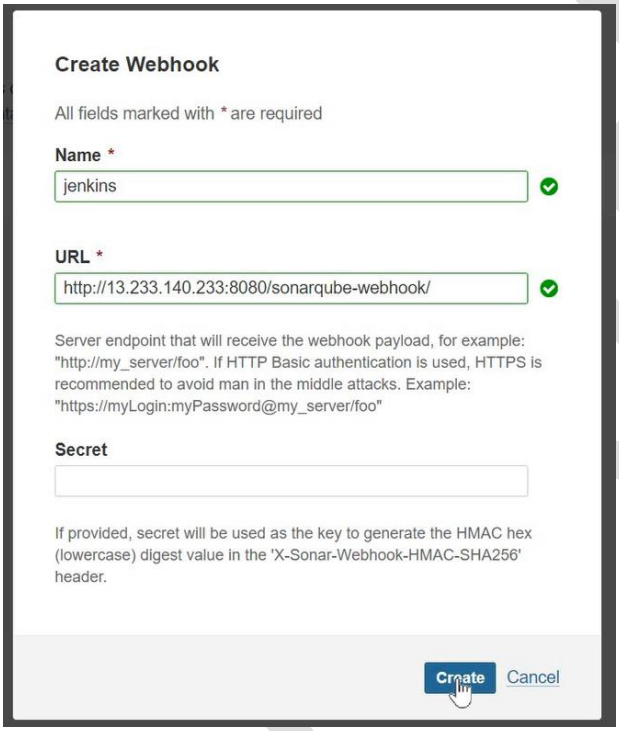


Give Token name and create. Save the token in a notepad.



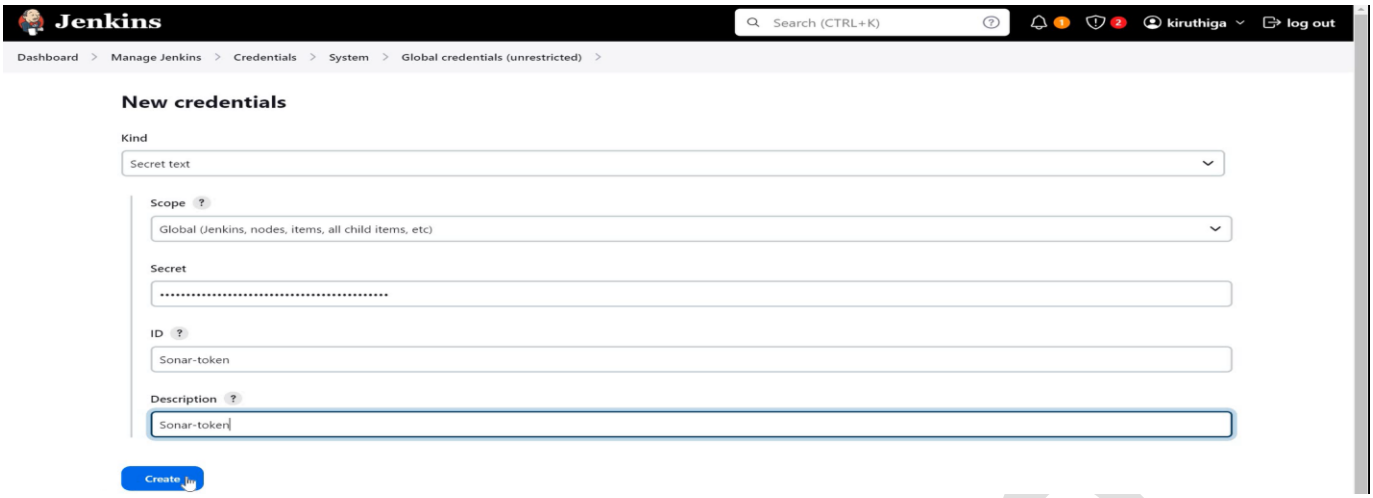
Add Quality Gate in Sonarqube Dashboard:

Go to “Administration” -> “Configuration” -> “Webhooks”. Add the below and click “Create”.



Add Sonarqube secret text in Jenkins Credentials:

Go to the Jenkins Dashboard -> “Manage Jenkins” -> “Credentials”. Add sonar token in credentials.



Add DockerHub Username and Password:

Configure system tools for sonarqube:

Go to "**Manage Jenkins**" -> "**System**". Click "**Add SonarQube**".

Click Apply and Save.

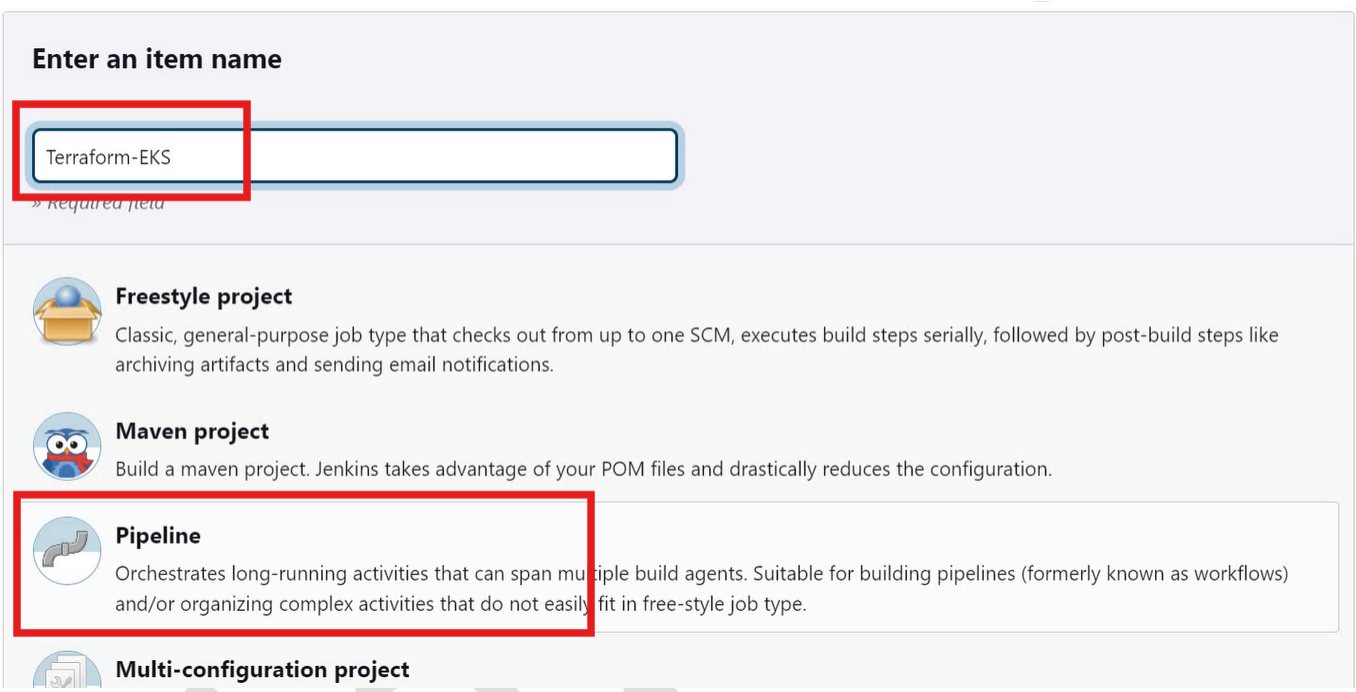
Create EKS Cluster from Jenkins using Terraform.

Go to legacy Branch in GitHub repository.

There is a folder named “Eks-terraform”. Here backend.tf file is available. (If needed: change the S3 bucket name as per the preference).

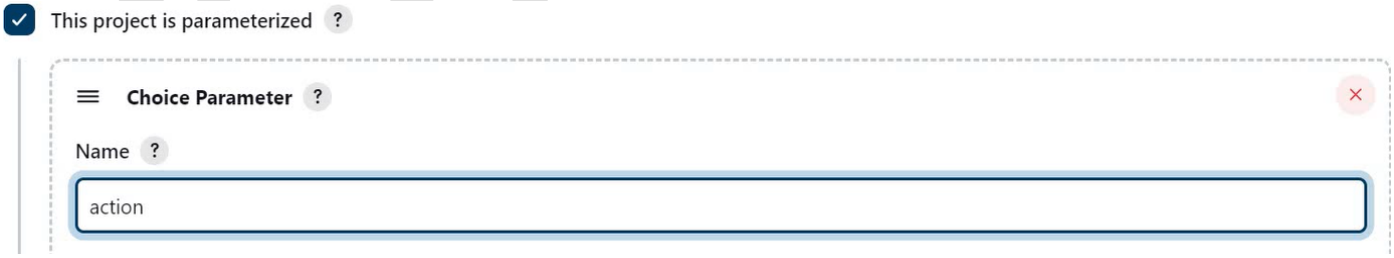
Create a Pipeline for EKS cluster creation:

Click **“+New Item**”. Enter the Name as “**Terraform-EKS**” and select as “**Pipeline**”. Click OK.



Select “**Discard Old Builds**” and mention **3** in “**Max # of builds to keep**”.

Select “**This project is parameterized**”. Click “**Add parameter**” button and Select “**Choice Parameter**”.



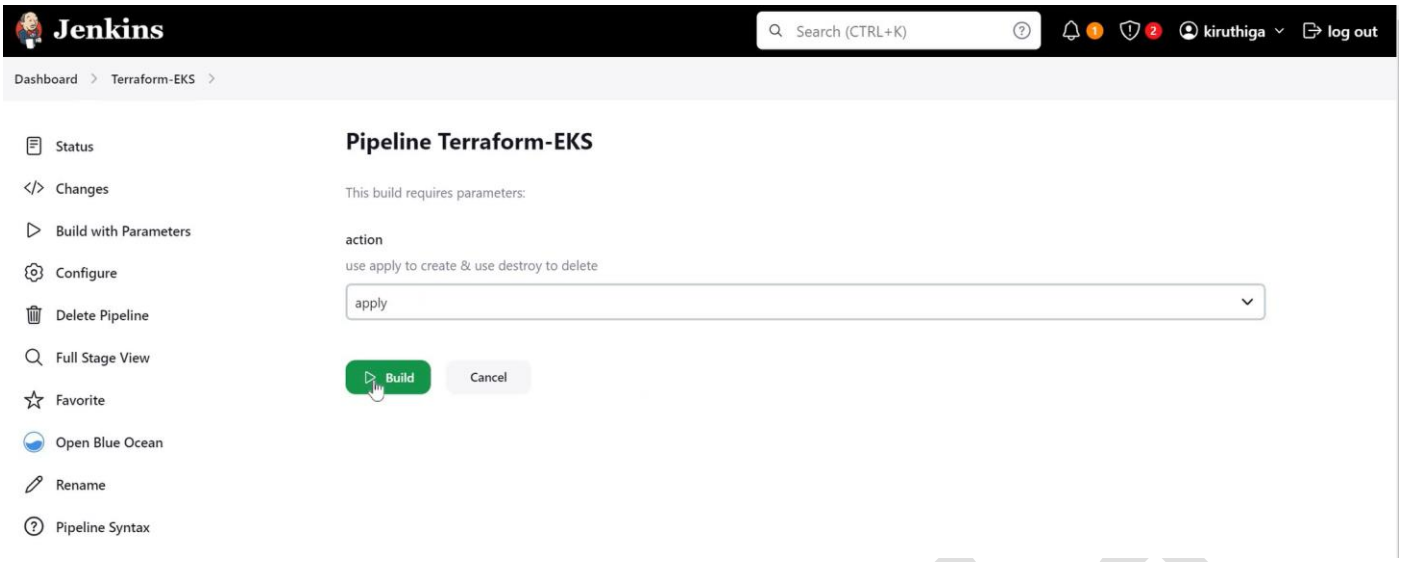
Add the below script in the pipeline. Click here to get the Script

```
pipeline{
   agent any
   stages {
      stage('Checkout from Git'){
         steps{
            git branch: 'legacy', url: ' https://github.com/Krishpluto/chatbot-ui.git'
         }
      }
      stage('Terraform version'){
         steps{
            sh 'terraform --version'
         }
      }
      stage('Terraform init'){
         steps{
            dir('Eks-terraform') {
               sh 'terraform init -reconfigure'
             }
         }
      }
      stage('Terraform validate'){
         steps{
            dir('Eks-terraform') {
```

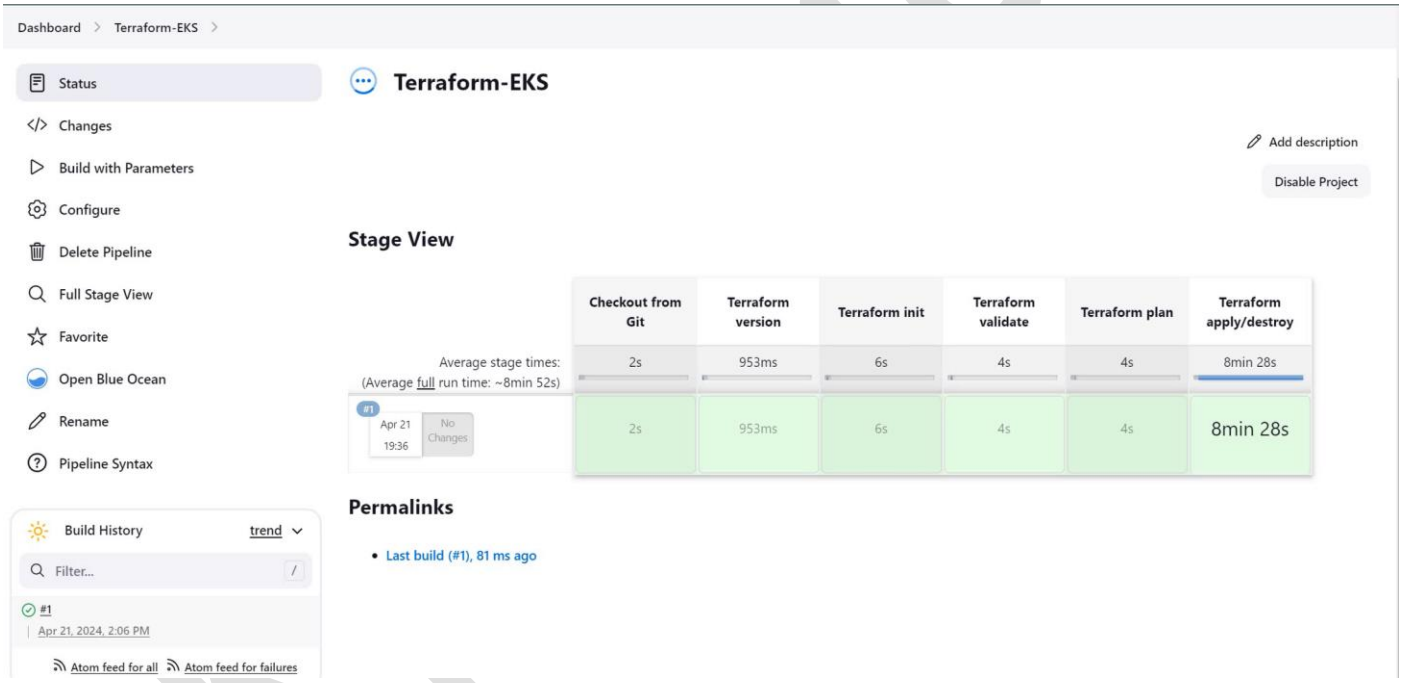
```
                sh 'terraform validate'
            }
        }
    }
    stage('Terraform plan'){
        steps{
            dir('Eks-terraform') {
                sh 'terraform plan'
            }
        }
    }
    stage('Terraform apply/destroy'){
        steps{
            dir('Eks-terraform') {
                sh 'terraform ${action} --auto-approve'
            }
        }
    }
  }
}
```

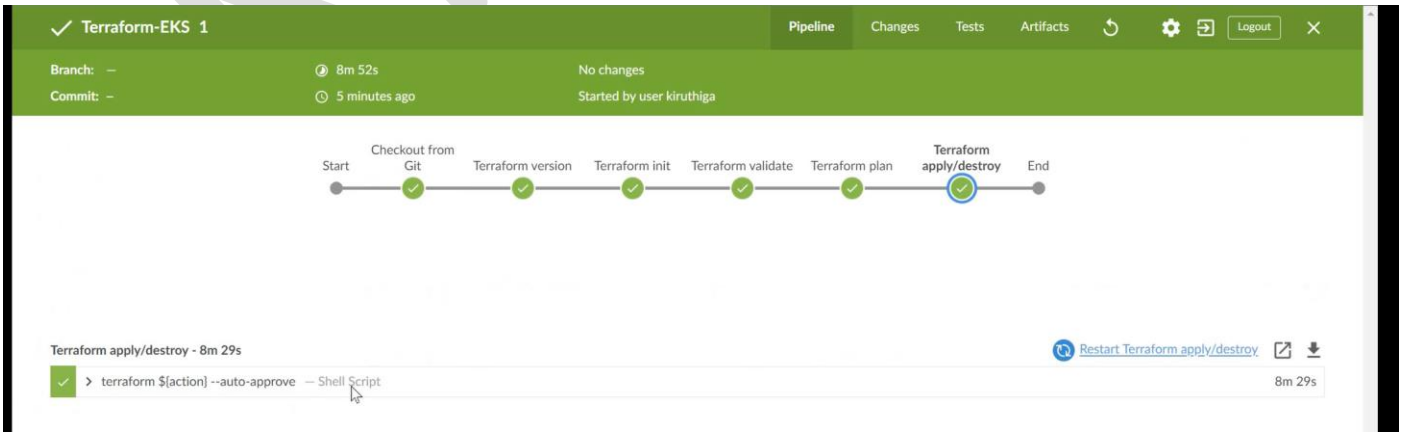
Click Apply and Save.

Now Select “**Build with Parameters**”, select “**apply**” and click “**Build**”.

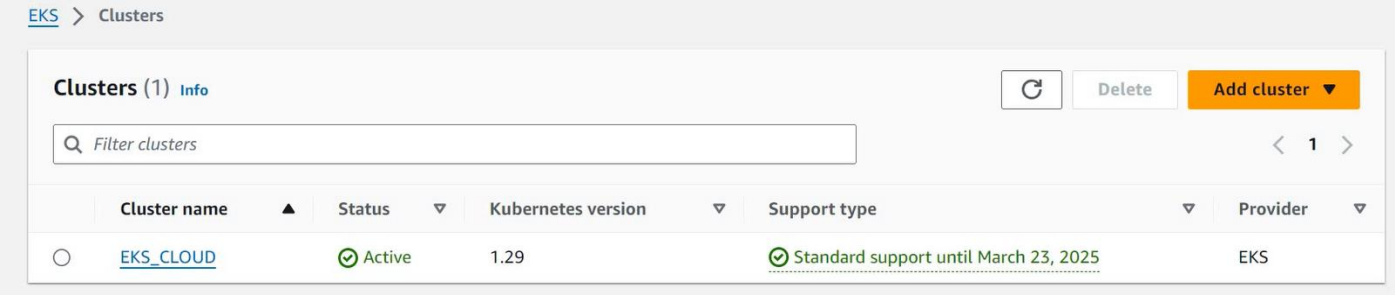

Jenkins
Search (CTRL+K)
kiruthiga
log out
Dashboard > Terraform-EKS >
Status
Changes
Build with Parameters
Configure
Delete Pipeline
Full Stage View
Favorite
Open Blue Ocean
Rename
Pipeline Syntax
Pipeline Terraform-EKS
This build requires parameters:
action
use apply to create & use destroy to delete
apply
Build
Cancel

Output:

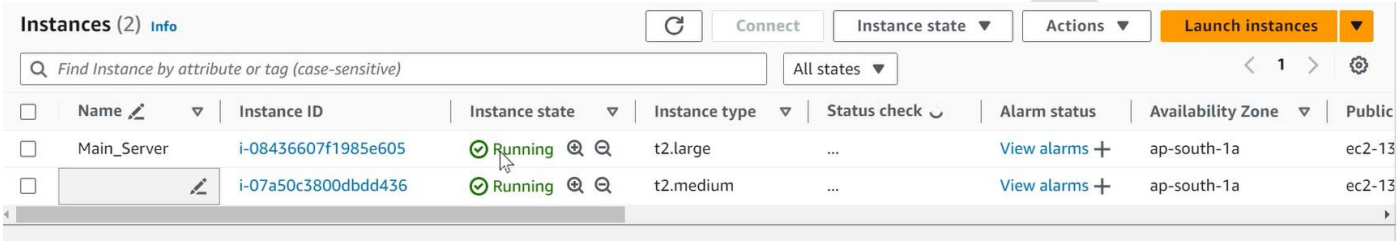

Dashboard > Terraform-EKS >
Status
Changes
Build with Parameters
Configure
Delete Pipeline
Full Stage View
Favorite
Open Blue Ocean
Rename
Pipeline Syntax
Terraform-EKS
Add description
Disable Project
Stage View
Checkout from Git
Terraform version
Terraform init
Terraform validate
Terraform plan
Terraform apply/destroy
Average stage times:
(Average full run time: ~8min 52s)
2s
953ms
6s
4s
4s
8min 28s
#1
Apr 21
19:36
No Changes
Permalinks
Last build (#1), 81 ms ago
Build History
trend
Filter...
#1
Apr 21, 2024, 2:06 PM
Atom feed for all
Atom feed for failures


Terraform-EKS 1
Pipeline
Changes
Tests
Artifacts
Logout
Branch: –
Commit: –
8m 52s
5 minutes ago
No changes
Started by user kiruthiga
Start
Checkout from Git
Terraform version
Terraform init
Terraform validate
Terraform plan
Terraform apply/destroy
End
Terraform apply/destroy - 8m 29s
Restart Terraform apply/destroy
terraform ${action} --auto-approve – Shell Script
8m 29s

Check in your AWS Console, if the EKS cluster is created or not.

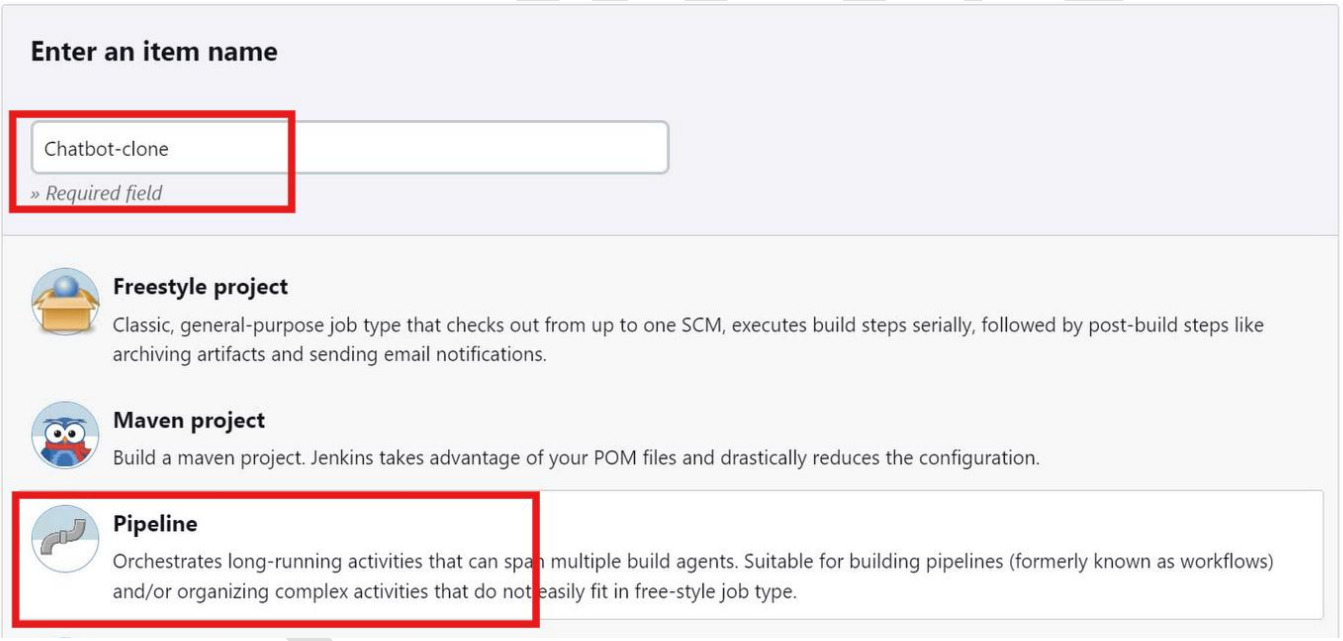


Check if EC2 instance is created or not.



Create New pipeline for deploying application.

Go to Jenkins Dashboard -> click “+New Item”.



Select “**Discard Old Builds**” and mention **3** in “**Max # of builds to keep**”.

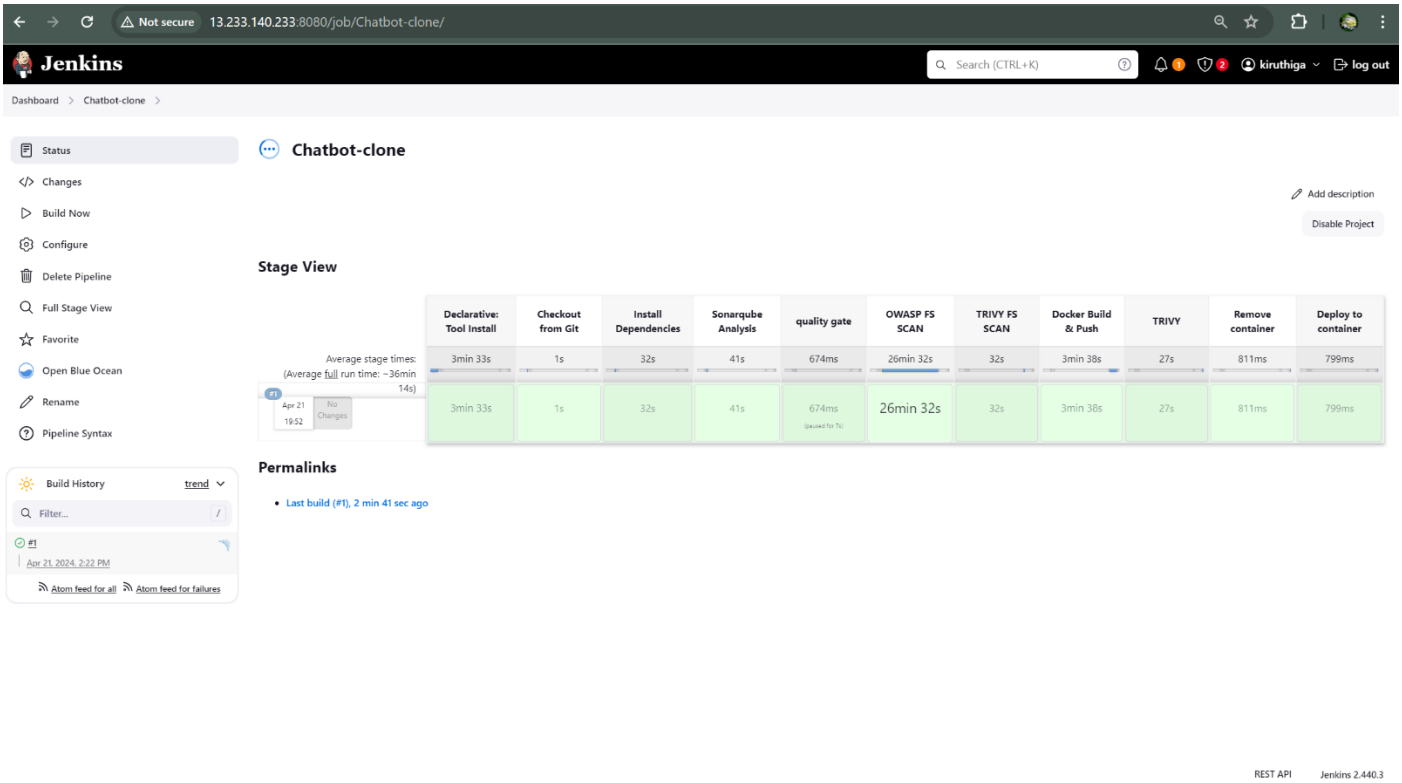
Add the below script in the pipeline script. [(SCRIPT)]

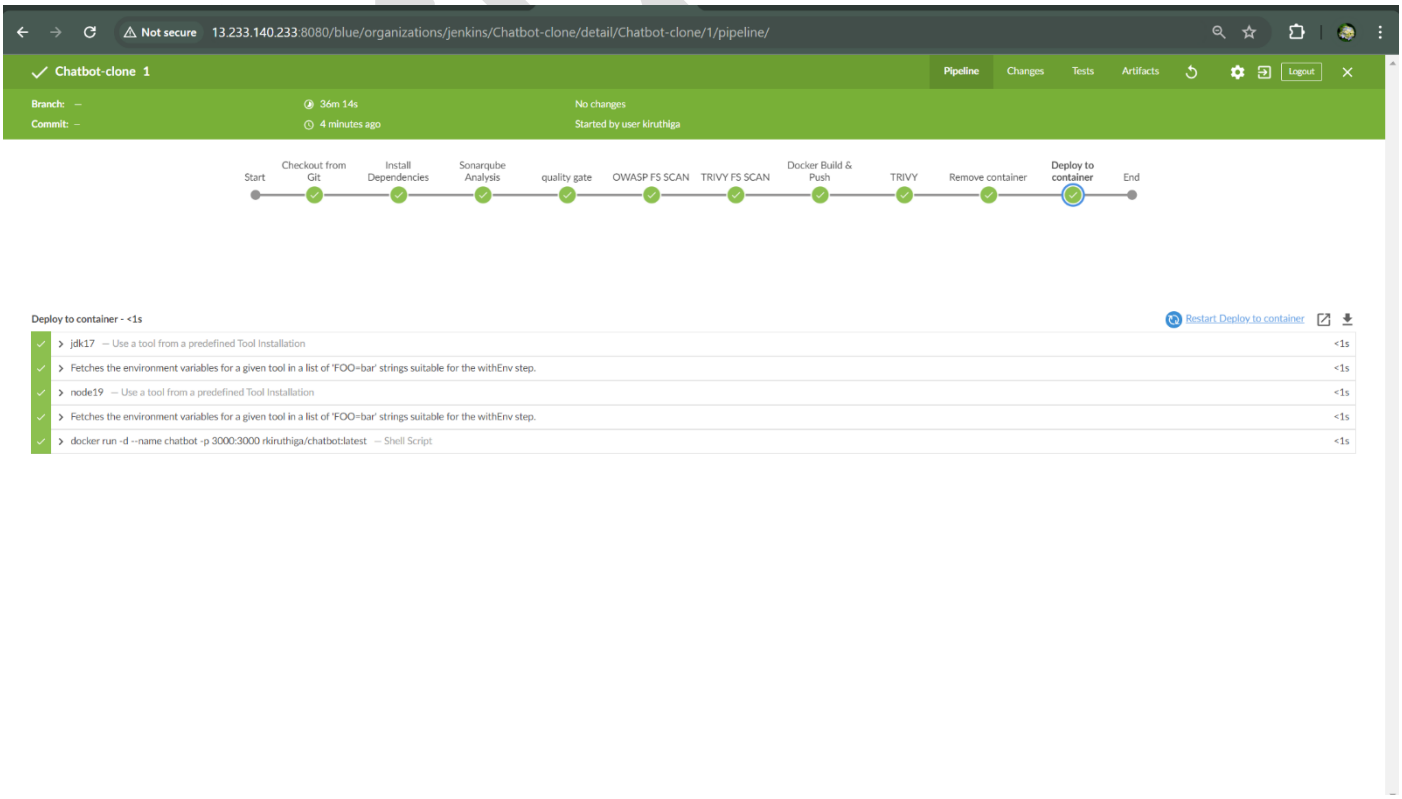
```
pipeline{
   agent any
   tools{
      jdk 'jdk17'
      nodejs 'node19'
```

```
}
environment {
   SCANNER_HOME=tool 'sonar-scanner'
}
stages {
   stage('Checkout from Git'){
      steps{
         git branch: 'legacy', url: ' https://github.com/Krishpluto/chatbot-ui.git'
      }
   }
   stage('Install Dependencies') {
      steps {
         sh "npm install"
      }
   }
   stage("Sonarqube Analysis "){
      steps{
         withSonarQubeEnv('sonar-server') {
            sh ''' $SCANNER_HOME/bin/sonar-scanner -Dsonar.projectName=Chatbot \
            -Dsonar.projectKey=Chatbot '''
         }
      }
   }
   stage("quality gate"){
     steps {
         script {
            waitForQualityGate abortPipeline: false, credentialsId: 'Sonar-token'
         }
      }
   }
   stage('OWASP FS SCAN') {
      steps {
```

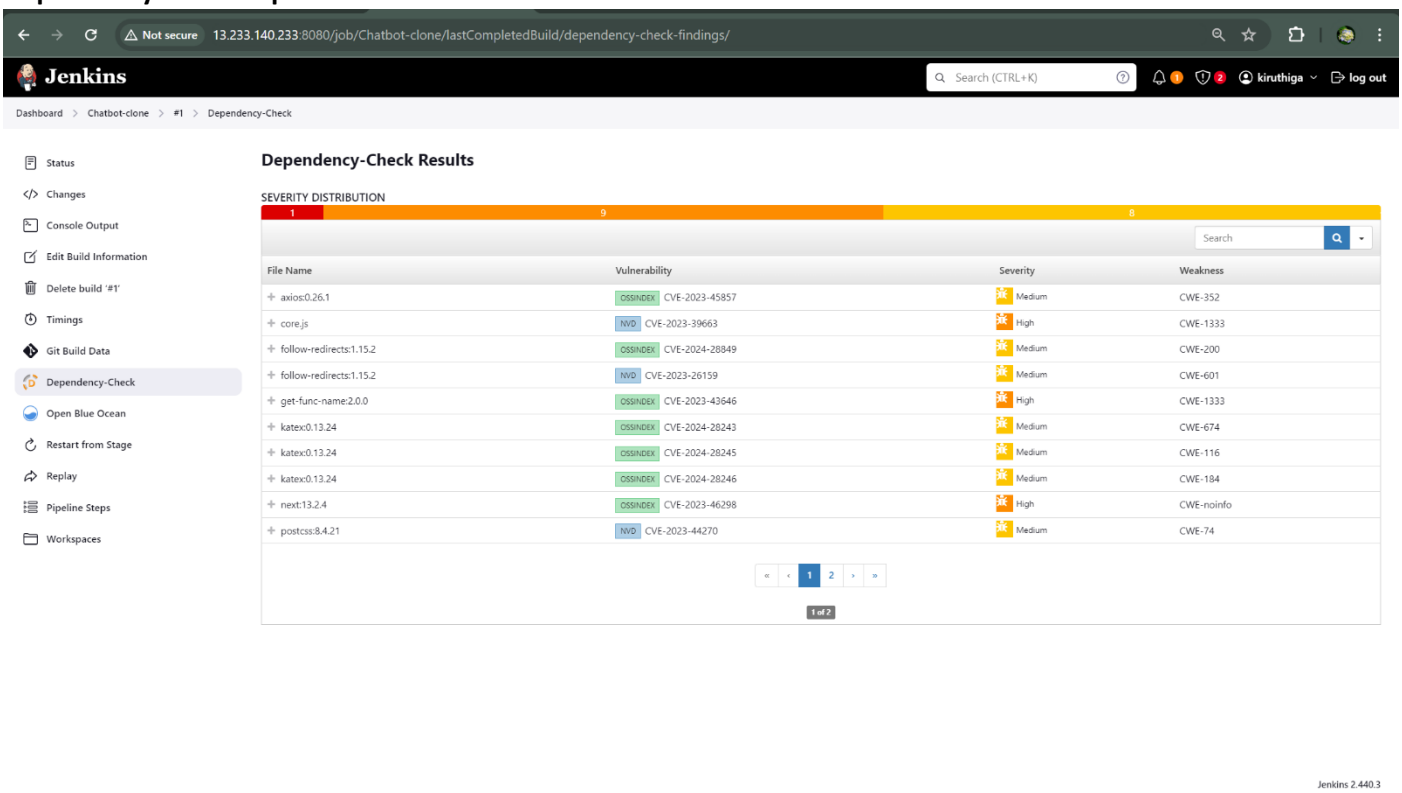
```
            dependencyCheck additionalArguments: '--scan ./ --disableYarnAudit --disableNodeAudit', odcInstallation:
'DP-Check'
            dependencyCheckPublisher pattern: '**/dependency-check-report.xml'
        }
    }
    stage('TRIVY FS SCAN') {
        steps {
            sh "trivy fs . > trivyfs.json"
        }
    }
    stage("Docker Build & Push"){
        steps{
            script{
              withDockerRegistry(credentialsId: 'docker', toolName: 'docker'){
                sh "docker build -t chatbot ."
                sh "docker tag chatbot rkiruthiga/chatbot:latest "
                sh "docker push rkiruthiga/chatbot:latest "
              }
            }
        }
    }
    stage("TRIVY"){
        steps{
            sh "trivy image rkiruthiga/chatbot:latest > trivy.json"
        }
    }
    stage ("Remove container") {
        steps{
            sh "docker stop chatbot | true"
            sh "docker rm chatbot | true"
         }
    }
    stage('Deploy to container'){
        steps{
            sh 'docker run -d --name chatbot -p 3000:3000 rkiruthiga/chatbot:latest'
        }
```

```
        }
    }
}
```

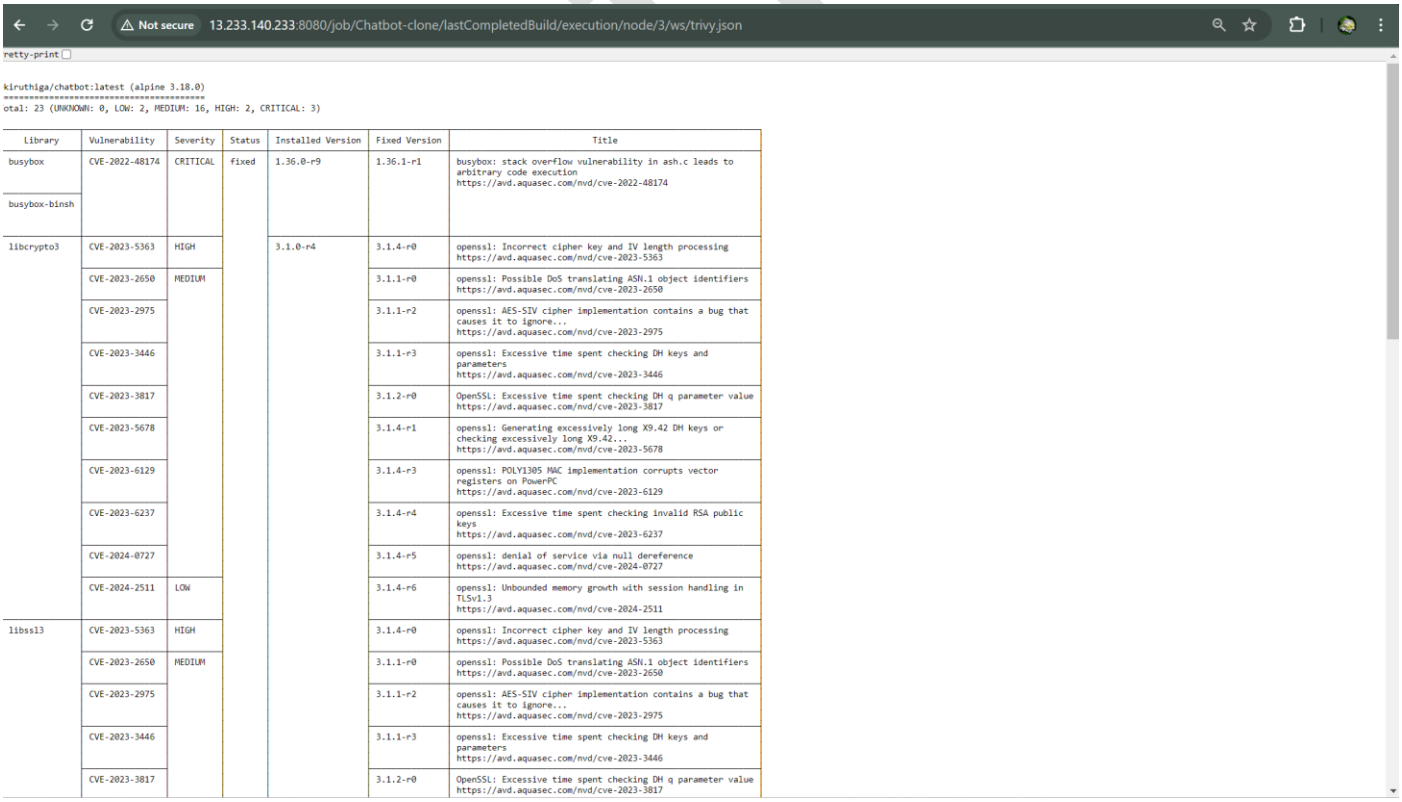

Not secure 13.233.140.233:8080/job/Chatbot-clone/
Jenkins
Search (CTRL+K)
kiruthiga
log out
Dashboard > Chatbot-clone >
Status
Changes
Build Now
Configure
Delete Pipeline
Full Stage View
Favorite
Open Blue Ocean
Rename
Pipeline Syntax
Build History
trend
Filter...
#1
Apr 21, 2024, 2:22 PM
Atom feed for all
Atom feed for failures
Chatbot-clone
Add description
Disable Project
Stage View
Declarative: Tool Install
Checkout from Git
Install Dependencies
Sonarqube Analysis
quality gate
OWASP FS SCAN
TRIVY FS SCAN
Docker Build & Push
TRIVY
Remove container
Deploy to container
Average stage times:
(Average full run time: ~36min 14s)
3min 33s
1s
32s
41s
674ms
26min 32s
32s
3min 38s
27s
811ms
799ms
Apr 21 19:52
No Changes
Permalinks
Last build (#1), 2 min 41 sec ago
REST API
Jenkins 2.440.3


Not secure 13.233.140.233:8080/blue/organizations/jenkins/Chatbot-clone/detail/Chatbot-clone/1/pipeline/
Chatbot-clone 1
Pipeline
Changes
Tests
Artifacts
Logout
Branch: –
Commit: –
36m 14s
4 minutes ago
No changes
Started by user kiruthiga
Start
Checkout from Git
Install Dependencies
Sonarqube Analysis
quality gate
OWASP FS SCAN
TRIVY FS SCAN
Docker Build & Push
TRIVY
Remove container
Deploy to container
End
Deploy to container - <1s
Restart Deploy to container
jdk17 – Use a tool from a predefined Tool Installation
Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step.
node19 – Use a tool from a predefined Tool Installation
Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step.
docker run -d --name chatbot -p 3000:3000 rkiruthiga/chatbot:latest – Shell Script

Dependency Check Reports:



Trivy Reports:



Library	Vulnerability	Severity	Status	Installed Version	Fixed Version	Title
busybox	CVE-2022-48174	CRITICAL	fixed	1.36.0-r9	1.36.1-r1	busybox: stack overflow vulnerability in ash.c leads to arbitrary code execution https://avd.aquasec.com/nvd/cve-2022-48174
busybox-binsh						
libcrypto3	CVE-2023-5363	HIGH		3.1.0-r4	3.1.4-r0	openssl: Incorrect cipher key and IV length processing https://avd.aquasec.com/nvd/cve-2023-5363
	CVE-2023-2650	MEDIUM			3.1.1-r0	openssl: Possible DoS translating ASN.1 object identifiers https://avd.aquasec.com/nvd/cve-2023-2650
	CVE-2023-2975				3.1.1-r2	openssl: AES-SIV cipher implementation contains a bug that causes it to ignore... https://avd.aquasec.com/nvd/cve-2023-2975
	CVE-2023-3446				3.1.1-r3	openssl: Excessive time spent checking DH keys and parameters https://avd.aquasec.com/nvd/cve-2023-3446
	CVE-2023-3817				3.1.2-r0	OpenSSL: Excessive time spent checking DH q parameter value https://avd.aquasec.com/nvd/cve-2023-3817
	CVE-2023-5678				3.1.4-r1	openssl: Generating excessively long X9.42 DH keys or checking excessively long X9.42... https://avd.aquasec.com/nvd/cve-2023-5678
	CVE-2023-6129				3.1.4-r3	openssl: POLY1305 MAC implementation corrupts vector registers on PowerPC https://avd.aquasec.com/nvd/cve-2023-6129
	CVE-2023-6237				3.1.4-r4	openssl: Excessive time spent checking invalid RSA public keys https://avd.aquasec.com/nvd/cve-2023-6237
	CVE-2024-0727				3.1.4-r5	openssl: denial of service via null dereference https://avd.aquasec.com/nvd/cve-2024-0727
	CVE-2024-2511	LOW			3.1.4-r6	openssl: Unbounded memory growth with session handling in TLSv1.3 https://avd.aquasec.com/nvd/cve-2024-2511
libssl3	CVE-2023-5363	HIGH			3.1.4-r0	openssl: Incorrect cipher key and IV length processing https://avd.aquasec.com/nvd/cve-2023-5363
	CVE-2023-2650	MEDIUM			3.1.1-r0	openssl: Possible DoS translating ASN.1 object identifiers https://avd.aquasec.com/nvd/cve-2023-2650
	CVE-2023-2975				3.1.1-r2	openssl: AES-SIV cipher implementation contains a bug that causes it to ignore... https://avd.aquasec.com/nvd/cve-2023-2975
	CVE-2023-3446				3.1.1-r3	openssl: Excessive time spent checking DH keys and parameters https://avd.aquasec.com/nvd/cve-2023-3446
	CVE-2023-3817				3.1.2-r0	OpenSSL: Excessive time spent checking DH q parameter value https://avd.aquasec.com/nvd/cve-2023-3817

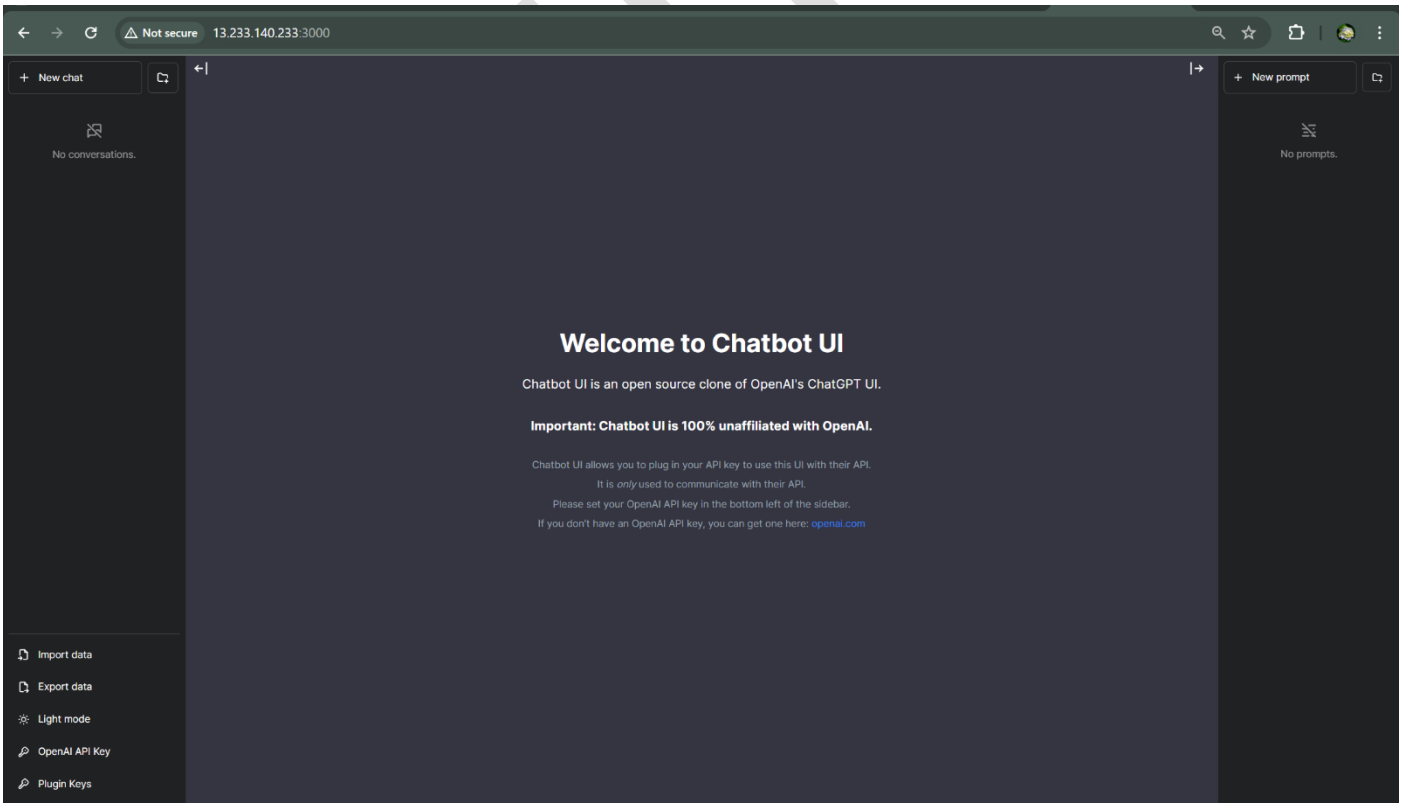
Not secure 13.233.140.233:8080/job/Chatbot-clone/lastCompletedBuild/execution/node/3/ws/trivyfs.json

Pretty-print

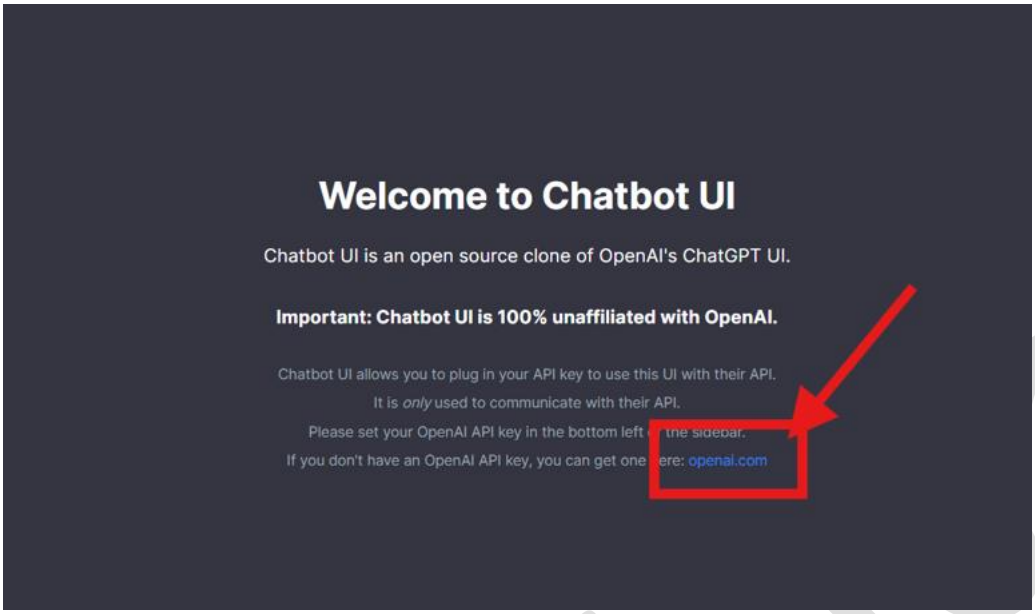
package-lock.json (npm)
==========
Total: 10 (UNKNOWN: 0, LOW: 1, MEDIUM: 9, HIGH: 0, CRITICAL: 0)

Library	Vulnerability	Severity	Status	Installed Version	Fixed Version	Title
axios	CVE-2023-45857	MEDIUM	fixed	0.26.1	1.6.0, 0.28.0	axios: exposure of confidential data stored in cookies https://avd.aquasec.com/nvd/cve-2023-45857
follow-redirects	CVE-2023-26159			1.15.2	1.15.4	follow-redirects: Improper Input Validation due to the improper handling of URLs by... https://avd.aquasec.com/nvd/cve-2023-26159
	CVE-2024-28849				1.15.6	follow-redirects: Possible credential leak https://avd.aquasec.com/nvd/cve-2024-28849
katex	CVE-2024-28243			0.13.24	0.16.10	KaTeX is a JavaScript library for TeX math rendering on the web.... https://avd.aquasec.com/nvd/cve-2024-28243
	CVE-2024-28245					KaTeX is a JavaScript library for TeX math rendering on the web.... https://avd.aquasec.com/nvd/cve-2024-28245
	CVE-2024-28246					KaTeX is a JavaScript library for TeX math rendering on the web.... https://avd.aquasec.com/nvd/cve-2024-28246
next	CVE-2023-46298	LOW		13.2.4	13.4.20-canary.13	Next.js missing cache-control header may lead to CDN caching empty reply https://avd.aquasec.com/nvd/cve-2023-46298
postcss	CVE-2023-44270	MEDIUM		8.4.14	8.4.31	An issue was discovered in PostCSS before 8.4.31. The vulnerability af https://avd.aquasec.com/nvd/cve-2023-44270
tough-cookie	CVE-2023-26136			4.1.2	4.1.3	tough-cookie: prototype pollution in cookie memstore https://avd.aquasec.com/nvd/cve-2023-26136
word-wrap	CVE-2023-26115			1.2.3	1.2.4	word-wrap: ReDoS https://avd.aquasec.com/nvd/cve-2023-26115

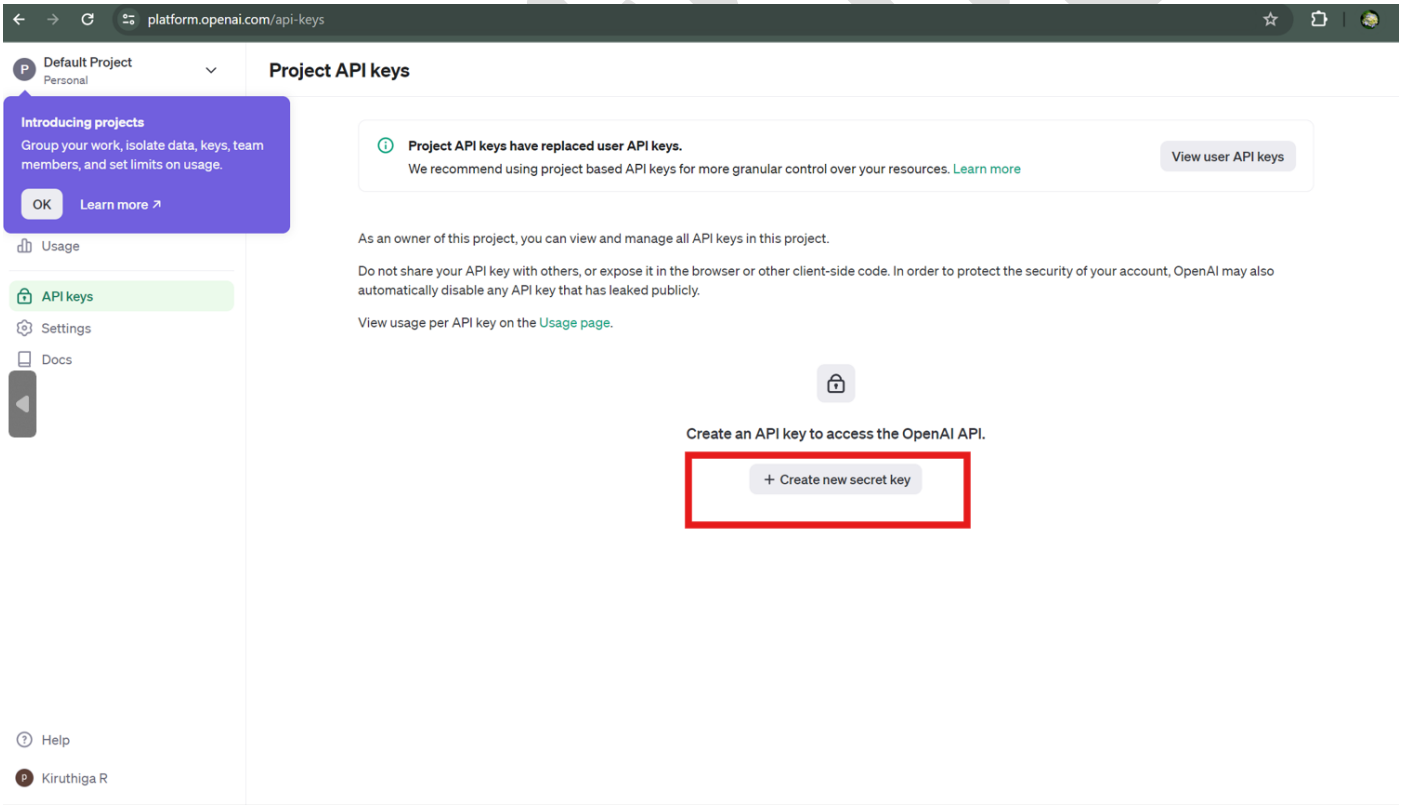
Now go to browser and enter <public ip of jenkins server ip>:3000. You will see the below output



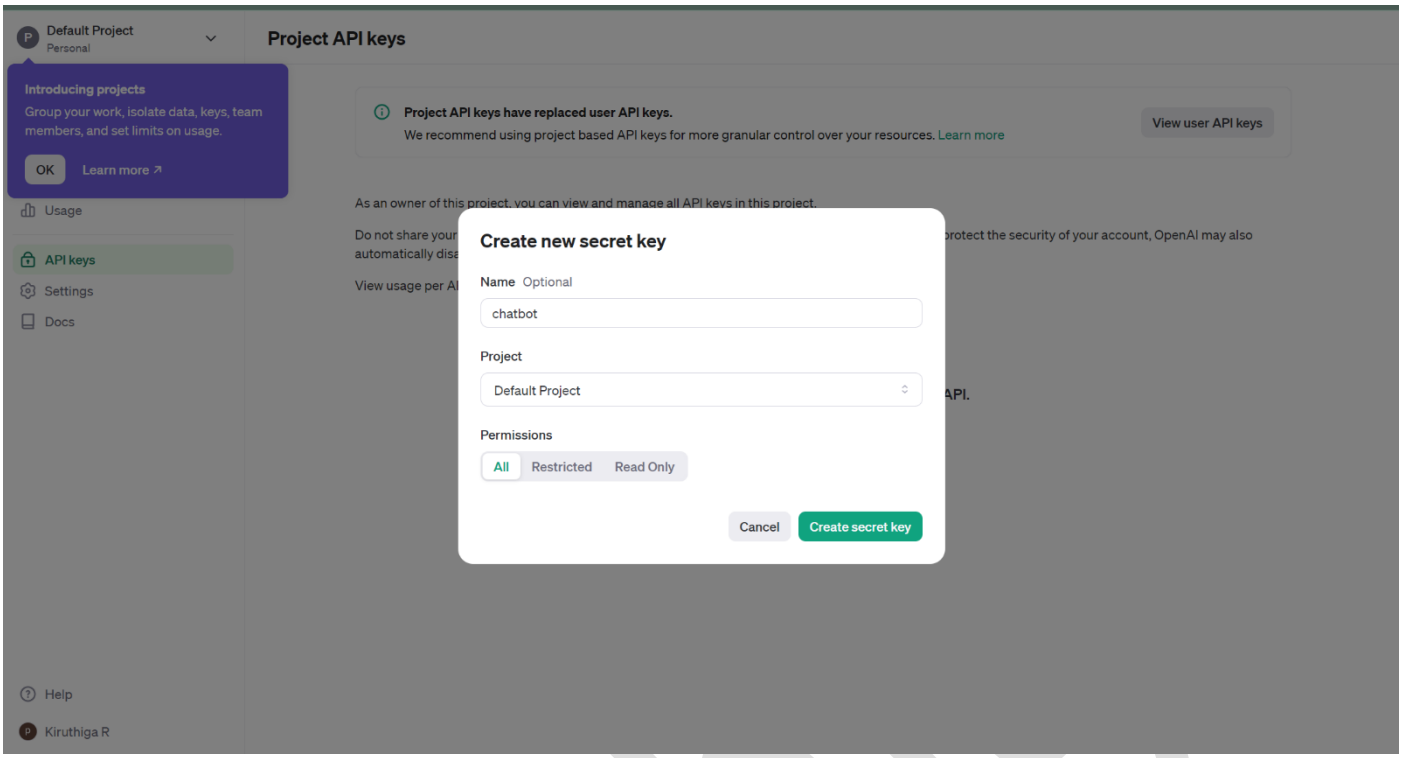
Click on openai.com to generate the API token.



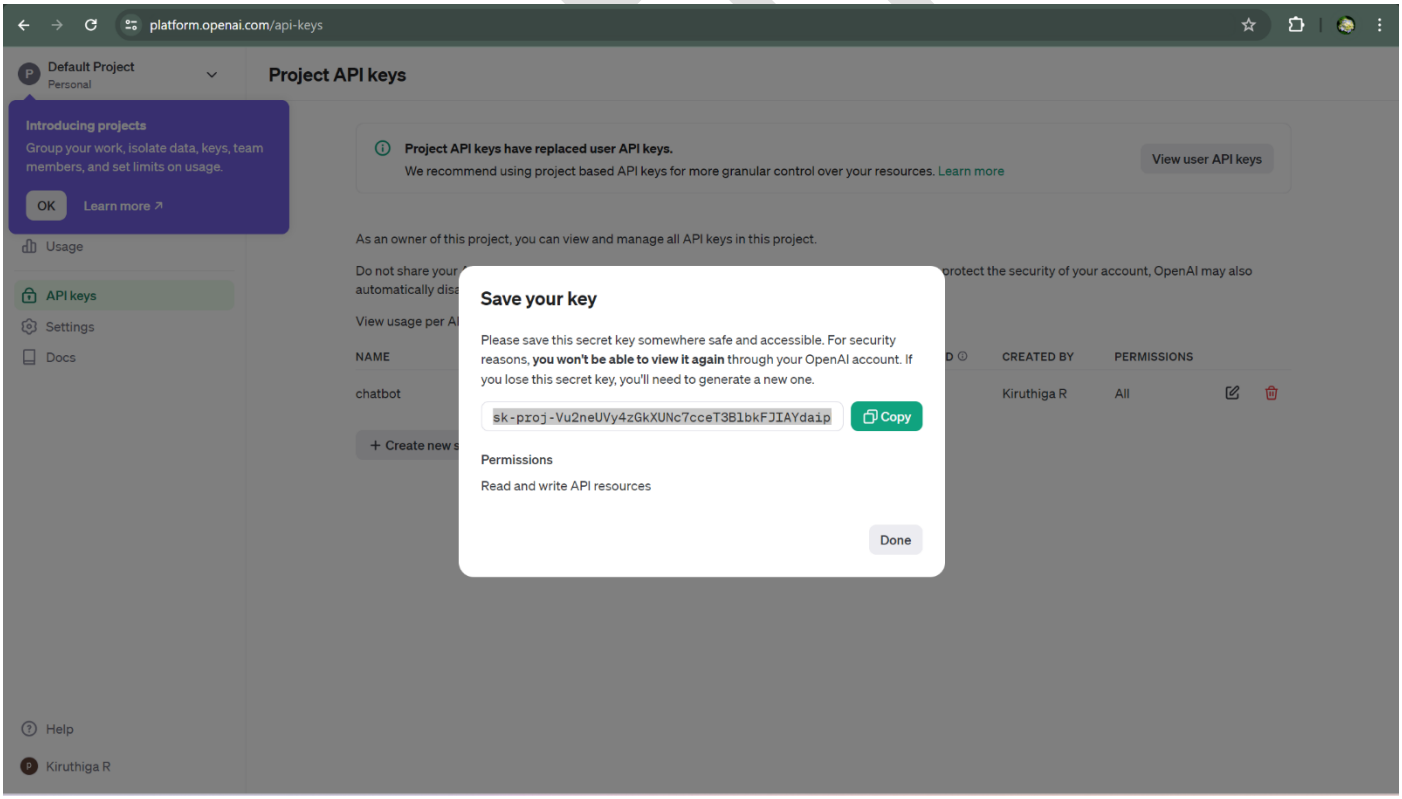
Click on Create new secret key.



Give Name and click on create.



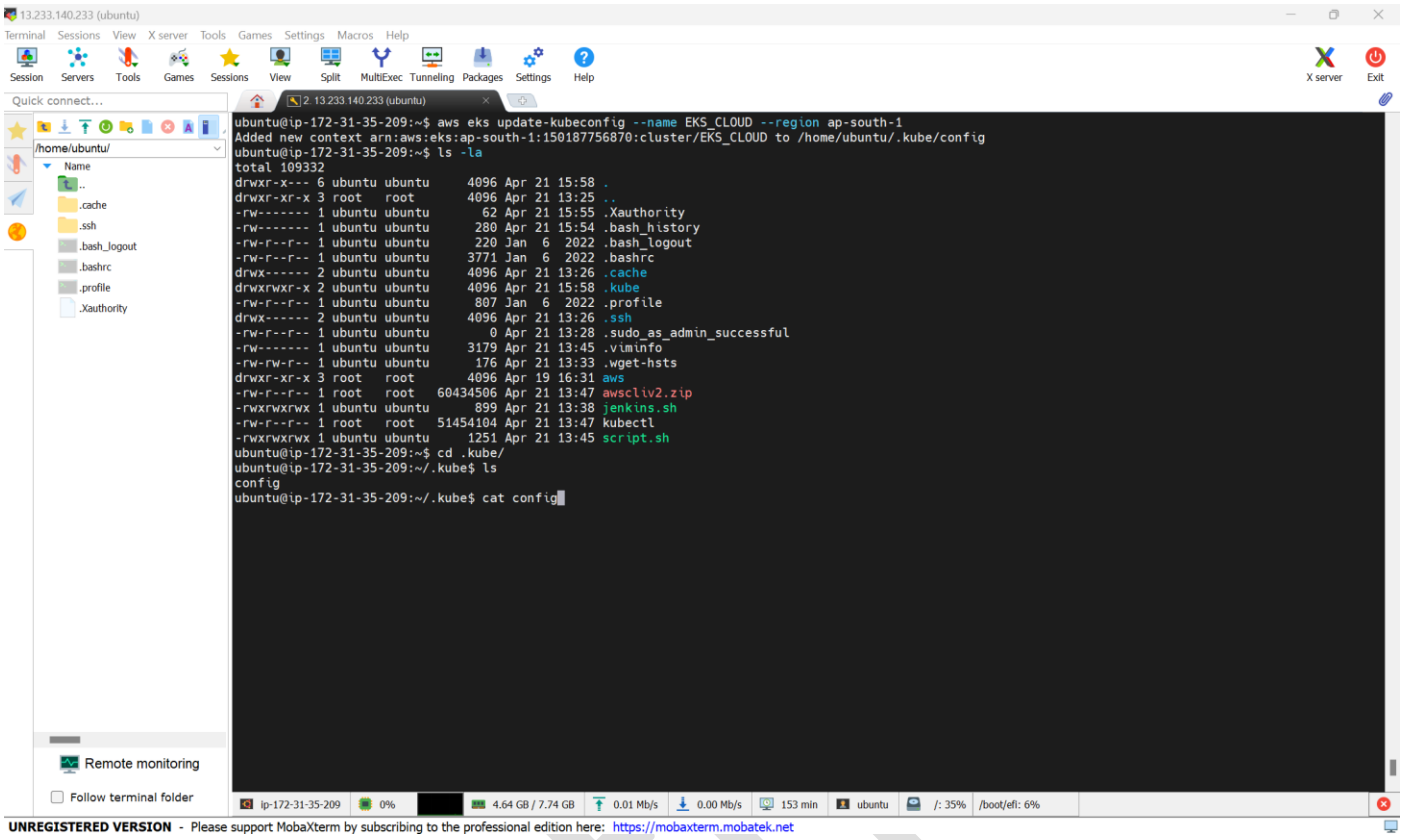
Save the Token and use.



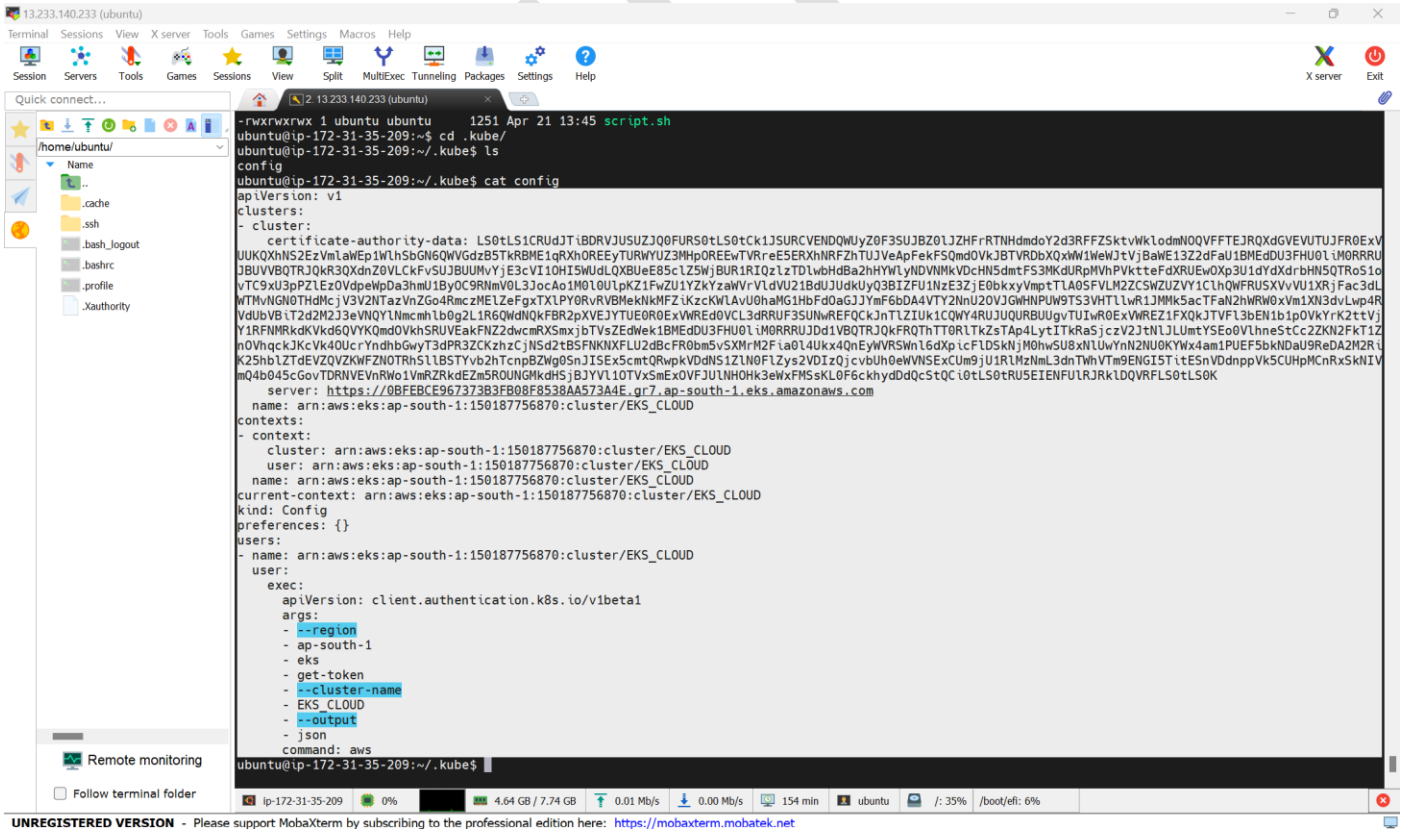
Navigate back to Chatbot UI that we deployed and bottom left you will see “OpenAI API Key”. Click that and give the generated key and save.

Now in Jenkins Mobaxterm instance, Give the below command:

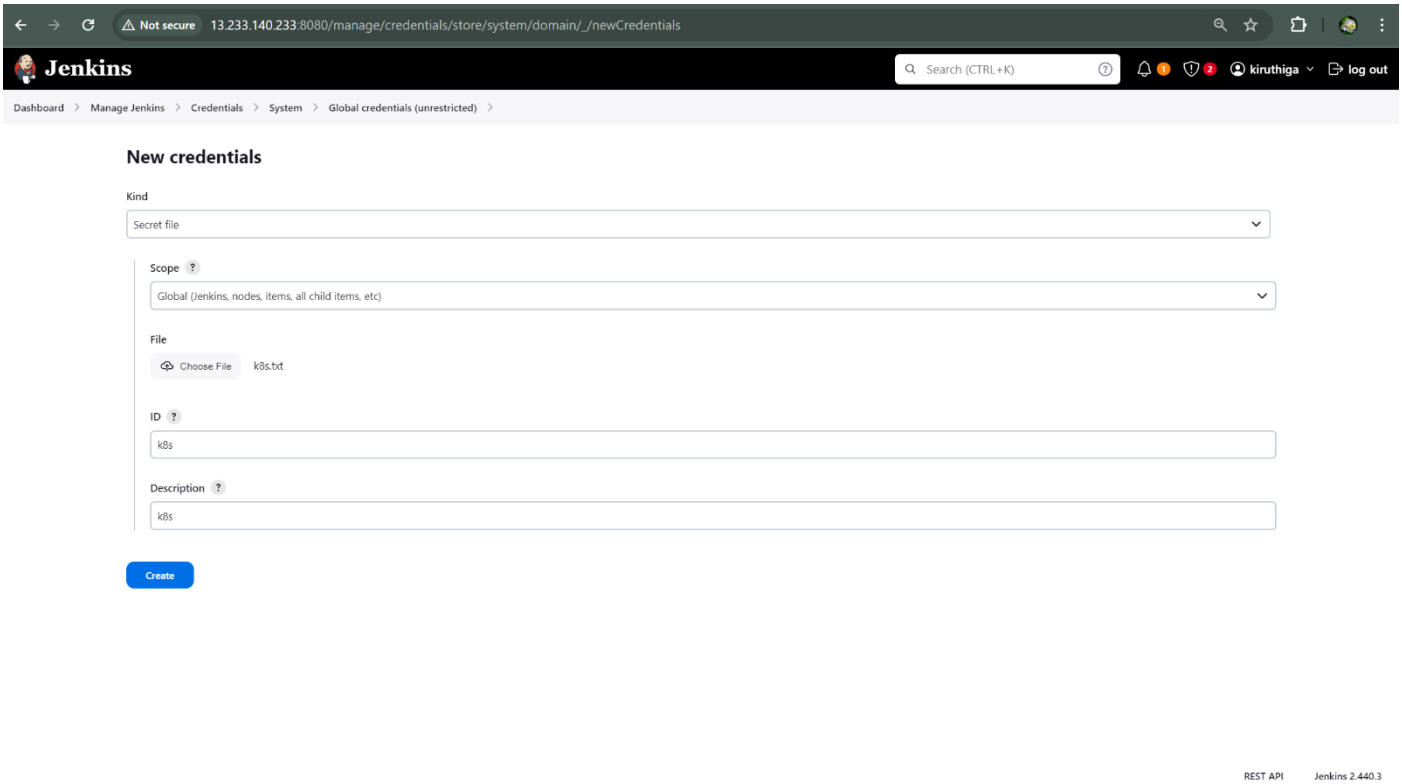
aws eks update-kubeconfig --name <clustername> --region <region>



Copy the contents of the config file and save it in text file in local machine.

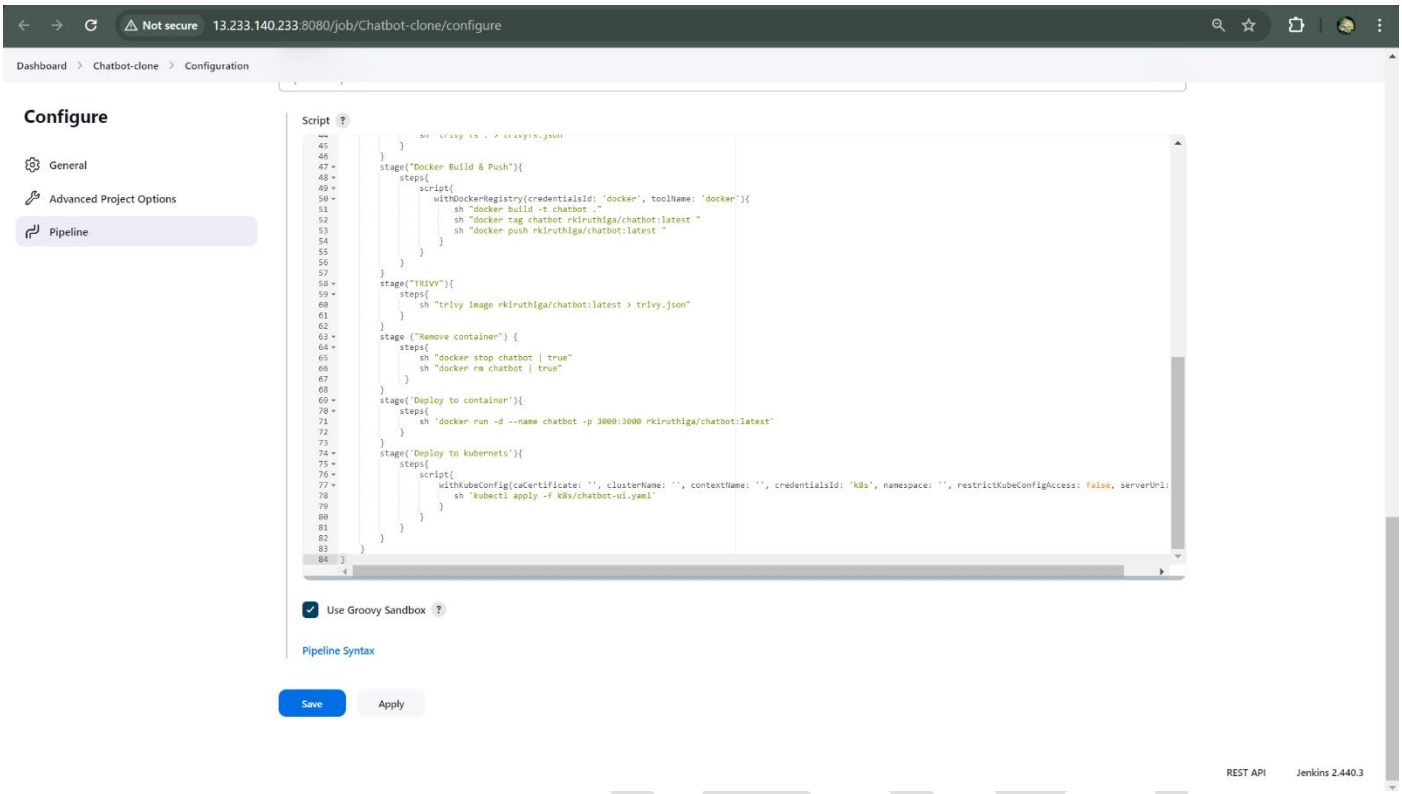


Now save this in a file and use this file to create a “**Secret file**” in “**Manage Jenkins**” -> “**Credentials**”



Now add the below stage in the pipeline as a last stage:

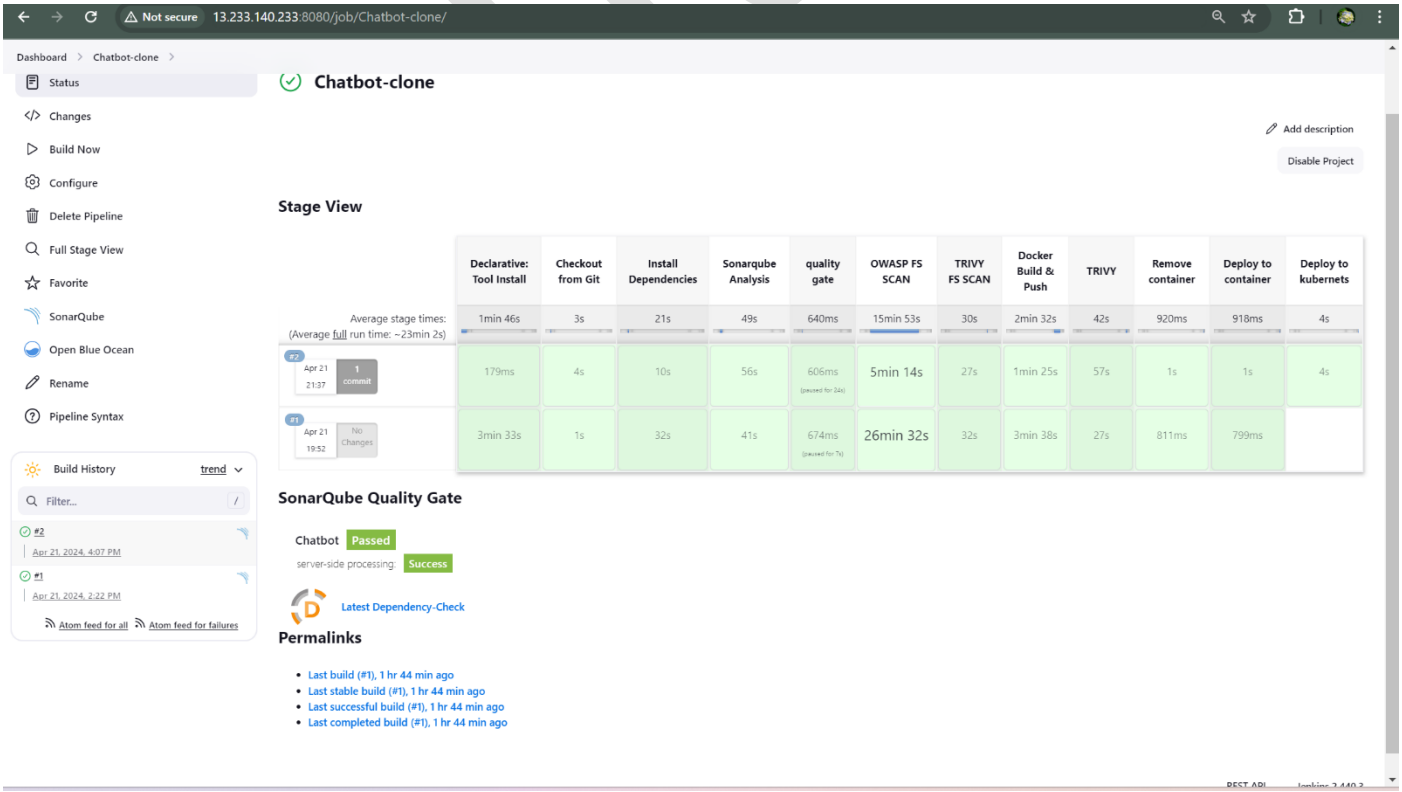
```
stage('Deploy to kubernets'){
        steps{
          script{
             withKubeConfig(caCertificate: '', clusterName: '', contextName: '', credentialsId: 'k8s', namespace: '', restrictKubeConfigAccess: false, serverUrl: '') {
               sh 'kubectl apply -f k8s/chatbot-ui.yaml'
           }
          }
        }
    }
```



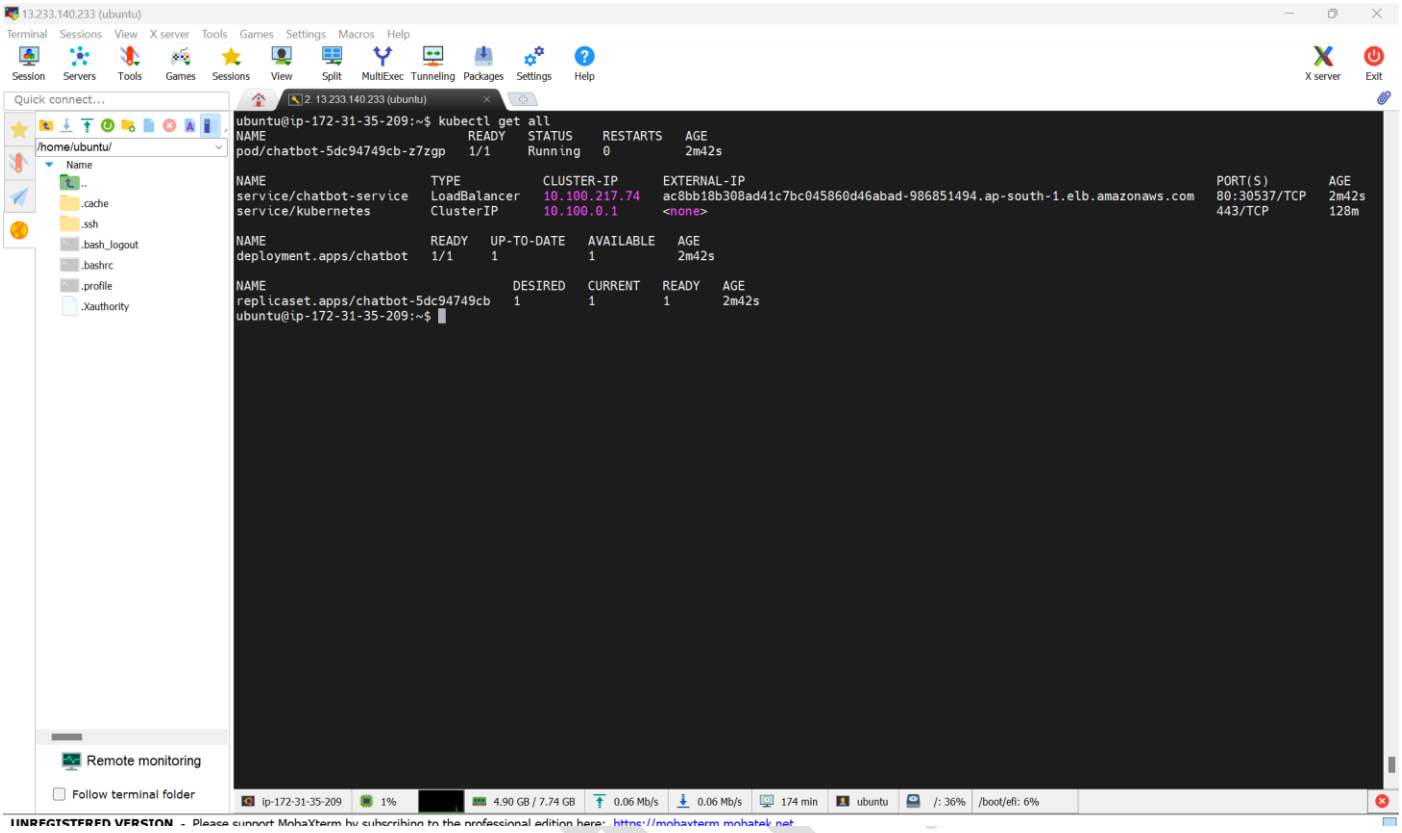
[CLICK THIS LINK TO GET THE FULL SCRIPT]

Update the Image in the chatbot-ui yml file in GitHub repository.

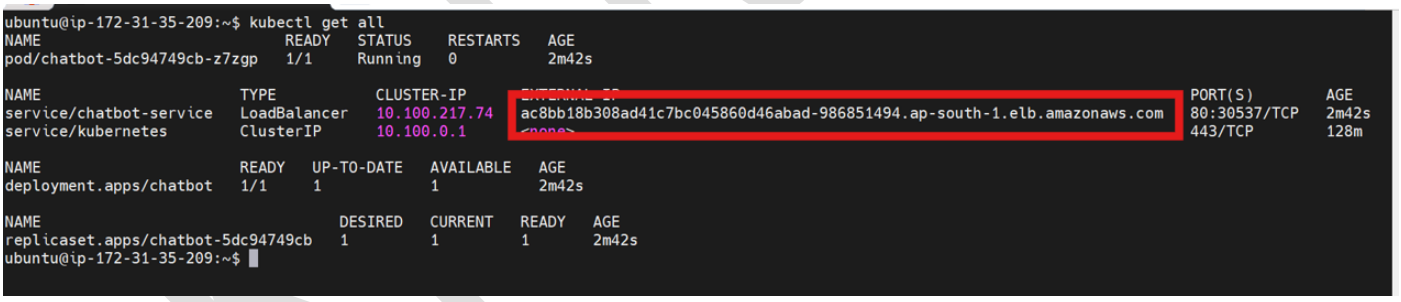
Click Apply and Save, Run the build to deploy to EKS.

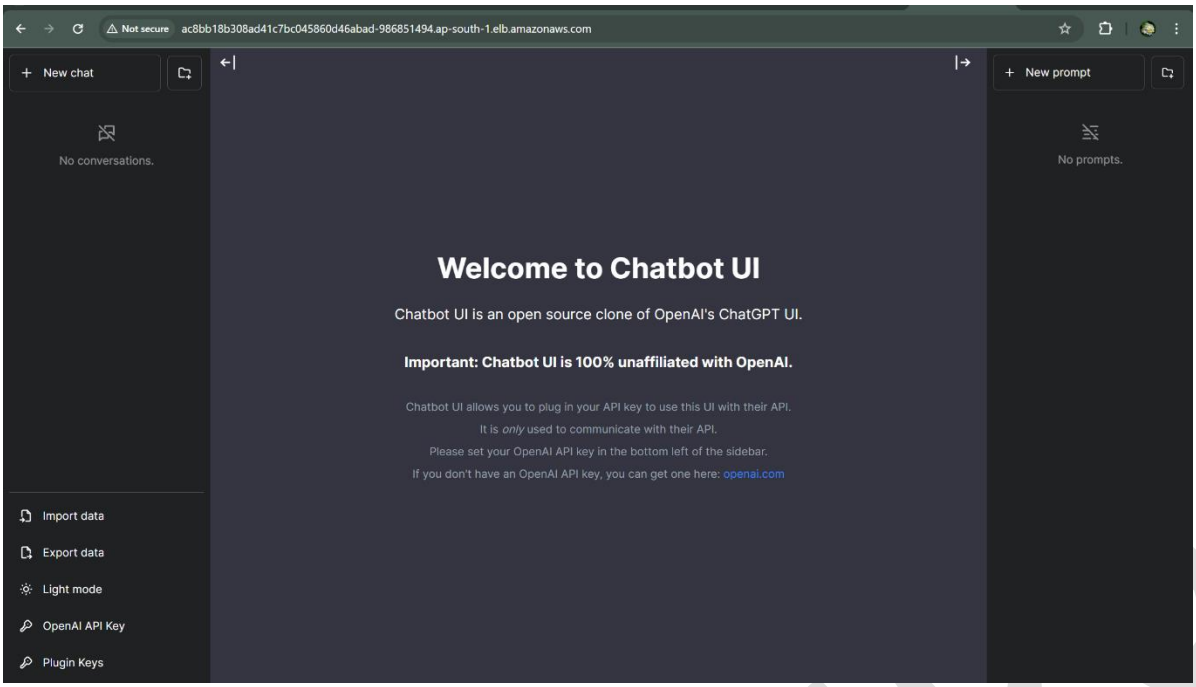


Now in Jenkins Mobaxterm instance, Give the below command:



Copy the Loadbalancer link and paste it in browser to see the output.





Delete the Resources:

Delete the Deployment by running the below script in the same pipeline (edit the script in the same pipeline)

(CLICK HERE)

```
pipeline{
   agent any
   stages {
      stage ("Remove container") {
         steps{
            sh "docker stop chatbot | true"
            sh "docker rm chatbot | true"
          }
      }
      stage('Delete the Deployment in kubernets'){
         steps{
            script{
                withKubeConfig(caCertificate: '', clusterName: '', contextName: '', credentialsId: 'k8s', namespace: '', restrictKubeConfigAccess: false, serverUrl: '') {
                  sh 'kubectl delete -f k8s/chatbot-ui.yaml'
              }
            }
         }
      }
```

```
    stage('scan cluster'){
      steps{
        script{
          withKubeConfig(caCertificate: '', clusterName: '', contextName: '', credentialsId: 'k8s', namespace: '', restrictKubeConfigAccess: false, serverUrl: '') {
            sh 'trivy k8s --report summary cluster'
          }
        }
      }
    }
  }
```

This Script will remove the Loadbalancer from EC2 and removes the deployment.

Go to EKS pipeline to destroy the cluster:

Select the “Build with Parameters” and select “destroy” and build

Delete the EC2 instance used to install jenkins server and other packages.