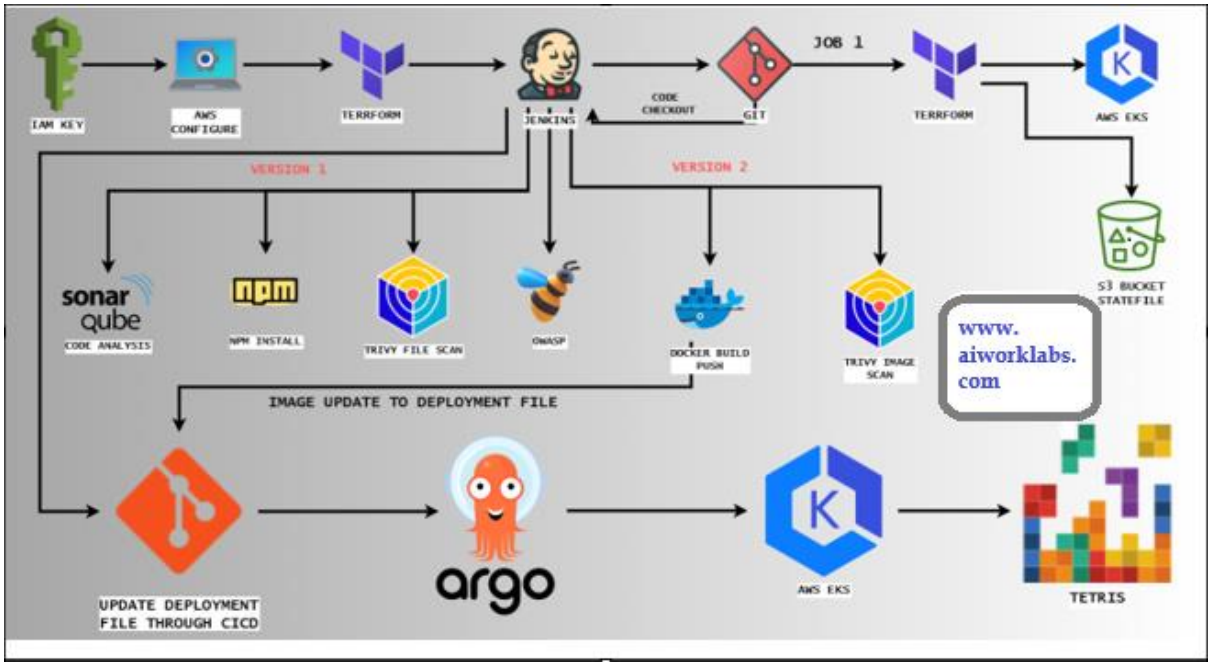
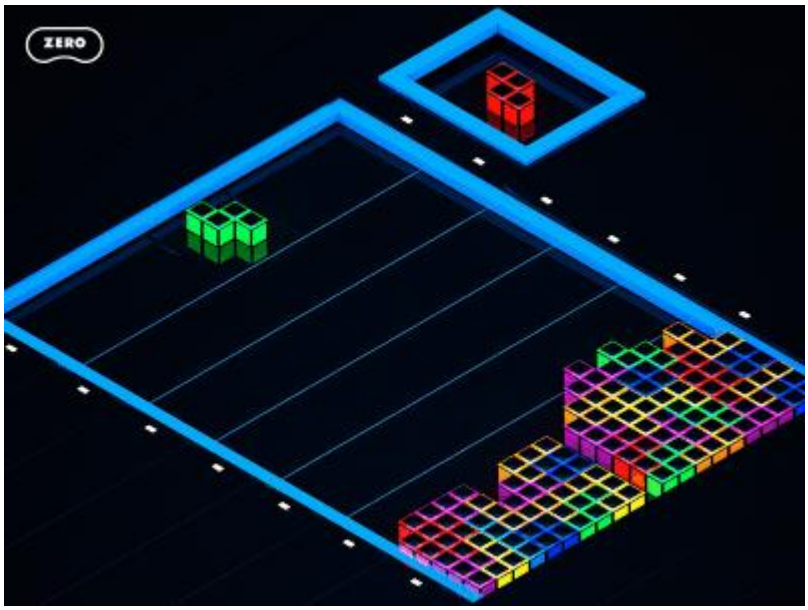
Rajinikanth DevSecOps Projects



Automating Tetris Deployments: DevSecOps with ArgoCD, Terraform, and Jenkins for Two Game Versions

Welcome to the world of DevSecOps automation, where cutting-edge

technologies merge to streamline the deployment of one of the most beloved classic games, Tetris. In this blog, we'll embark on a fascinating journey that unveils the power of ArgoCD, Terraform, and Jenkins in orchestrating a seamless and secure deployment pipeline for not one, but two distinct versions of the Tetris game. Imagine a scenario where your Tetris application effortlessly transitions through various stages of development, from infrastructure provisioning to continuous integration and delivery, all while maintaining the highest standards of security. Join us as we delve into the details of how these sophisticated tools work together to create an automated, efficient, and foolproof DevOps workflow, capable of handling multiple game versions, designed to impress even the most discerning tech enthusiasts.



GitHub REPOSITORIES

TETRIS-VERSION1

https://github.com/RajinikanthVadla/Tetris-V1.git

Prerequisites:

1. **AWS Account:** To get started, you'll need an active AWS account. Ensure that you have access and permission to create and manage AWS resources.
2. **AWS CLI:** Install the AWS Command Line Interface (CLI) on your local machine and configure it with your AWS credentials. This is essential for managing your AWS resources.
3. **IAM User and Key Pair:** Create an IAM (Identity and Access Management) user with the necessary permissions to provision resources on AWS. Additionally, generate an IAM Access Key and Secret Access Key for programmatic access. Ensure that you securely manage these credentials.
4. **S3 Bucket:** Set up an S3 bucket to store your Terraform state files. This bucket is crucial for maintaining the state of your infrastructure and enabling collaboration.
5. **Terraform:** Install Terraform on your local machine. Terraform is used for provisioning infrastructure as code and managing AWS resources. Make sure to configure Terraform to work with your AWS credentials and your S3 bucket for state

Step1: How to install and setup Terraform on Windows

Download Terraform:

Visit the official Terraform website: **terraform.io/downloads.html**

Verify the Installation:

Open a new Command Prompt or PowerShell window.

Type terraform –version and press Enter. This command should display the Terraform version, confirming that Terraform is installed and in your PATH.

```
PS C:\Users\Rajinikanth\Downloads> terraform -version
Terraform v1.7.0
on windows_386

Your version of Terraform is out of date! The latest version
is 1.7.1. You can update by downloading from https://www.terraform.io/downloads.html
PS C:\Users\Rajinikanth\Downloads> |
```

Step2: Download the AWS CLI Installer:

Visit the AWS CLI Downloads page: **aws.amazon.com/cli**

Under “Install the AWS CLI,” click on the “64-bit” link to download the AWS CLI installer for Windows

Verify the Installation:

Open a Command Prompt or PowerShell window.

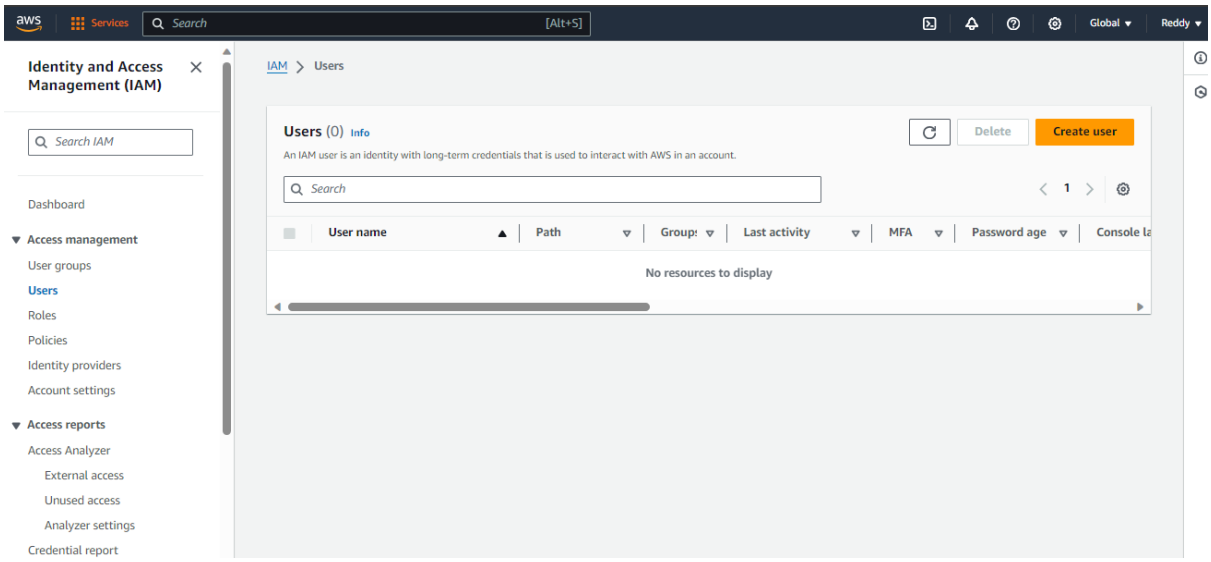
Type **aws –version** and press Enter. This command should display the AWS CLI version, confirming that the installation was successful.

```
PS C:\Users\Rajinikanth\Downloads> aws --version
aws-cli/2.9.0 Python/3.9.11 Windows/10 exe/AMD64 prompt/off
PS C:\Users\Rajinikanth\Downloads>
```

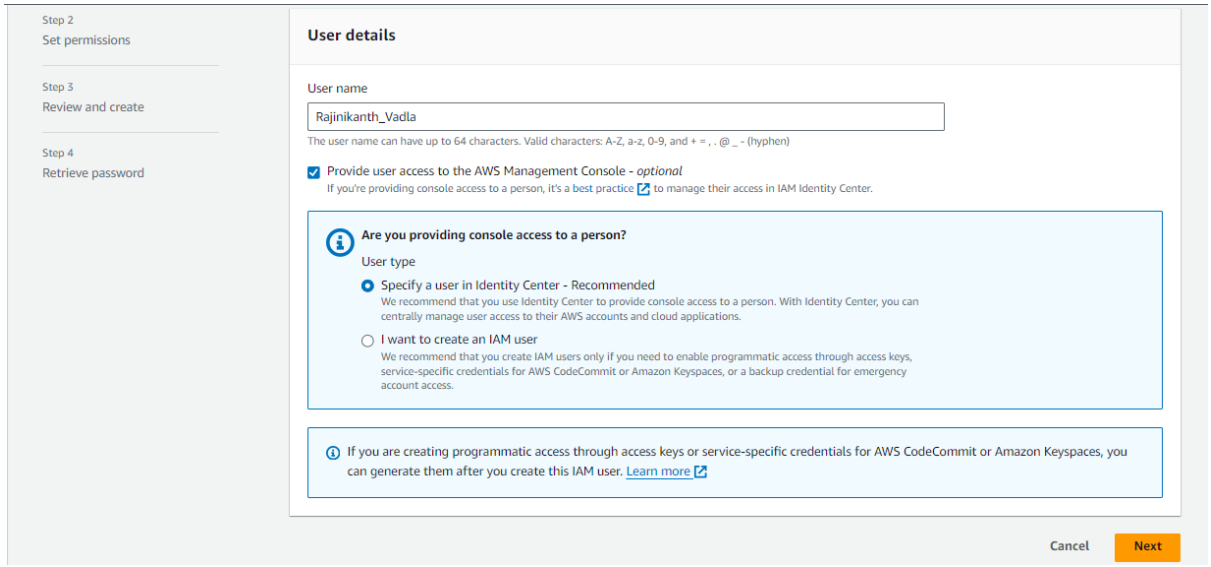
Step3: create an IAM user

Navigate to the **AWS console**

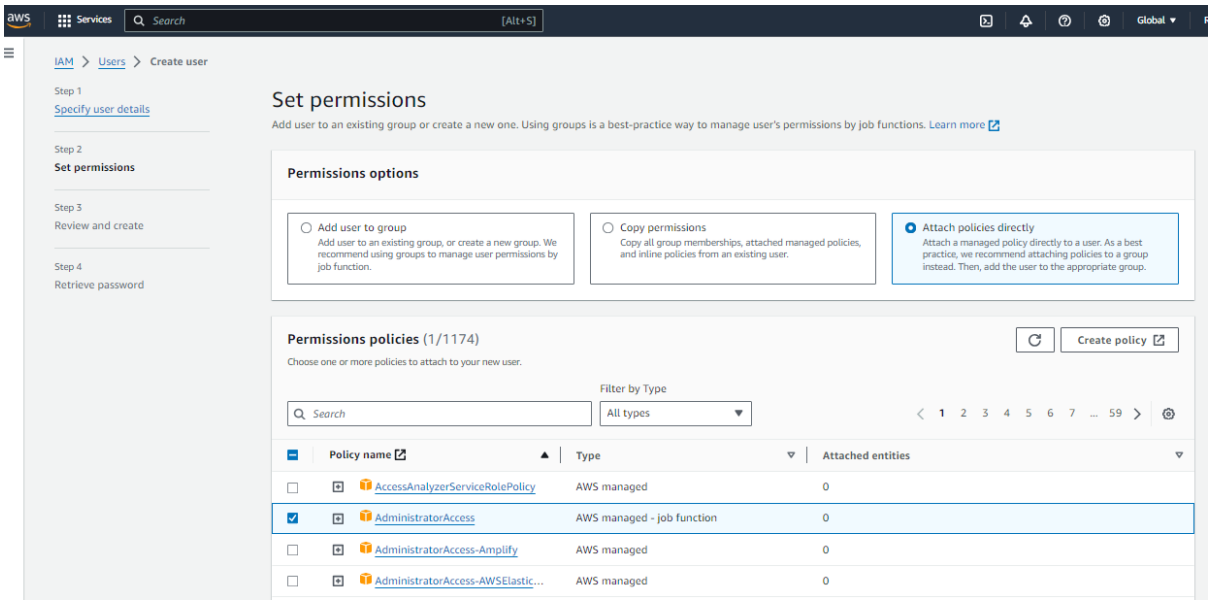
Click the “Search” field.



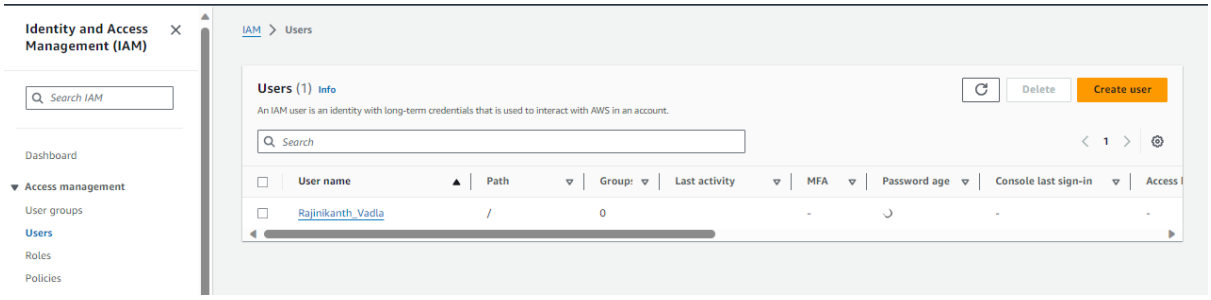
Create user Rajinikanth_Vadla and click next



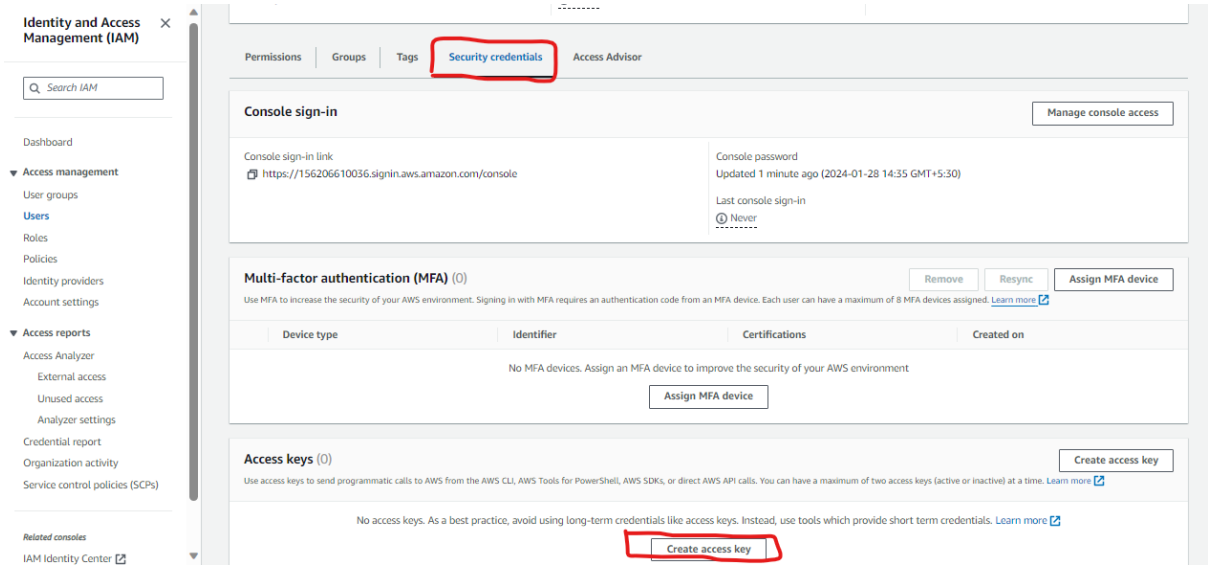
Then attach a policy as per below image

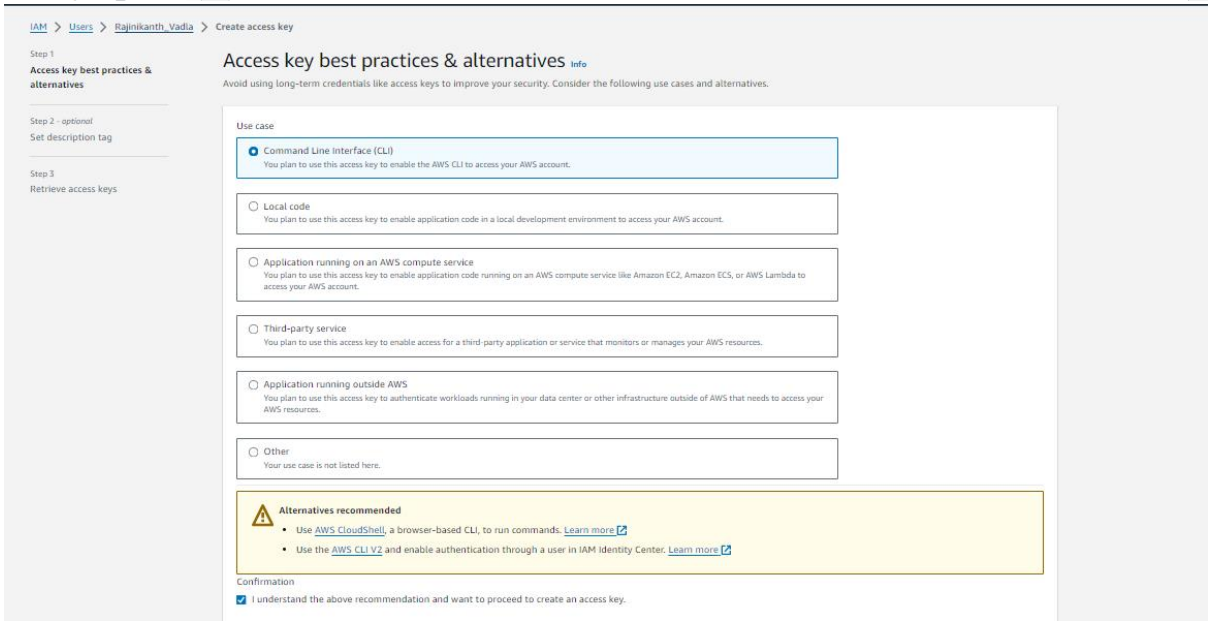


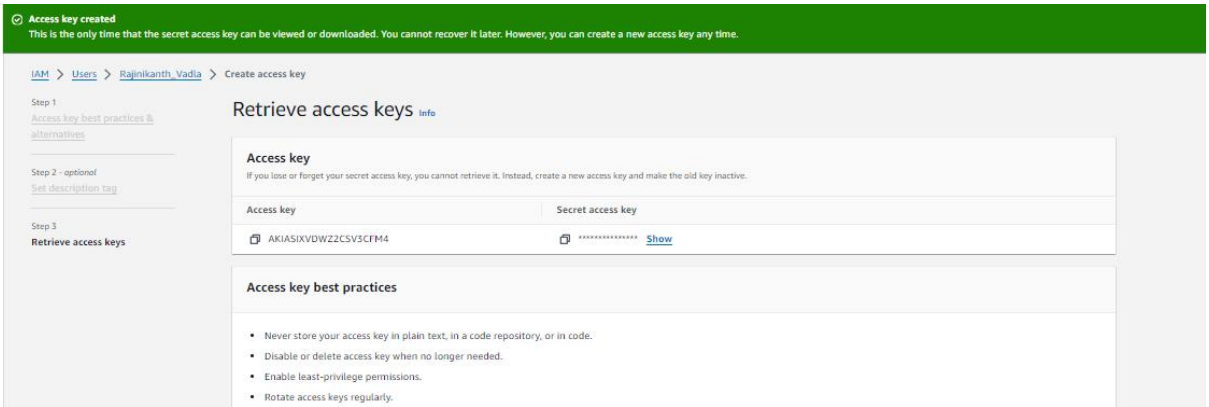
Once user created click on user



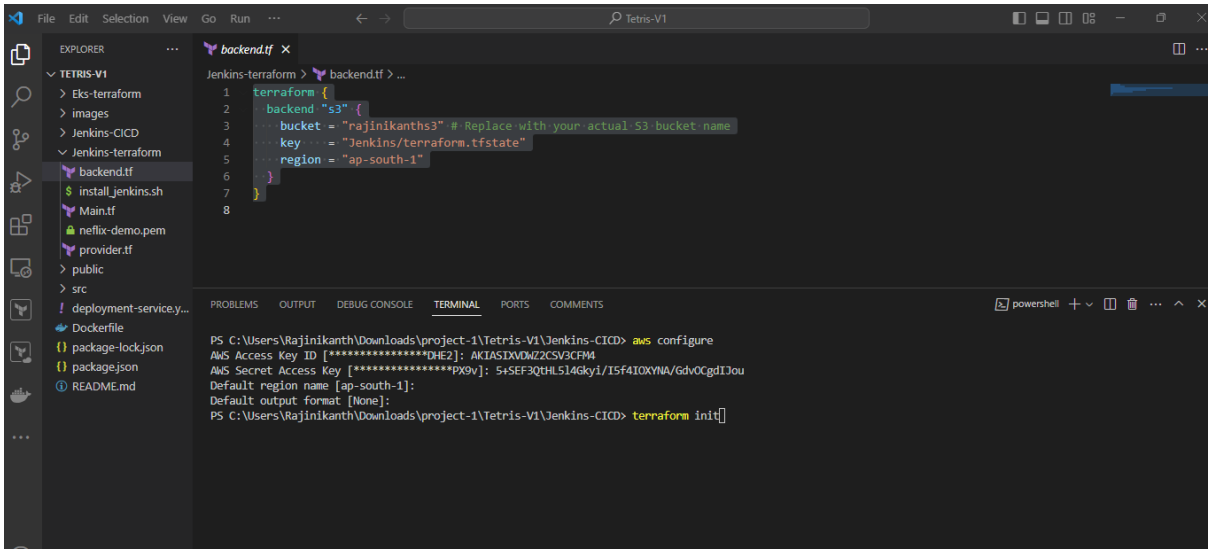
Create an access key for this user



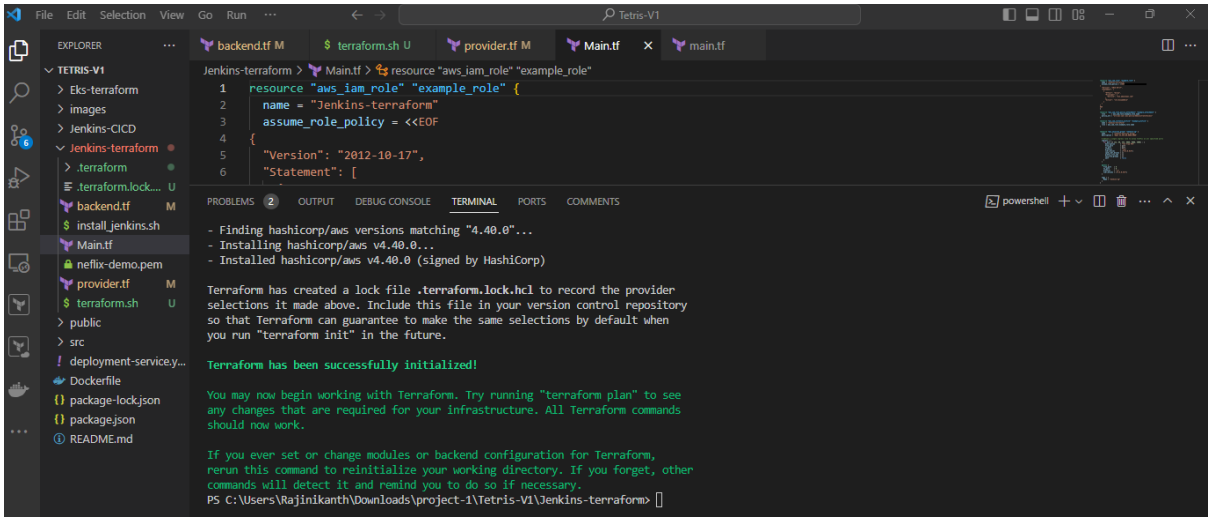




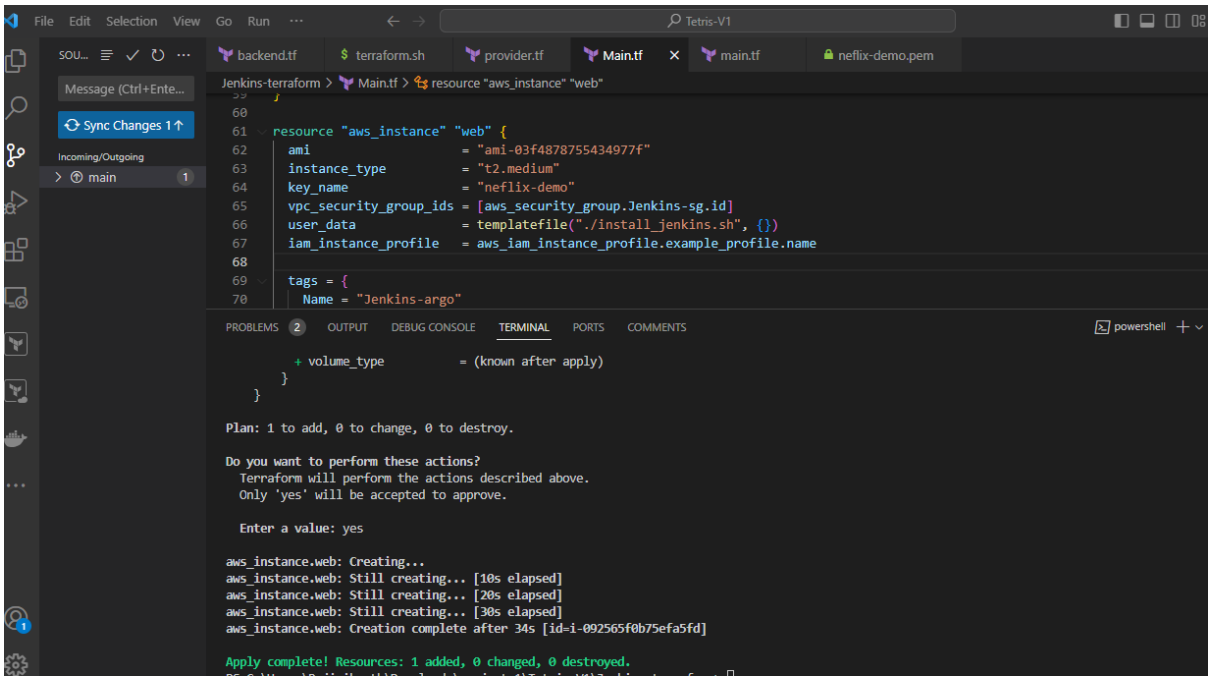
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

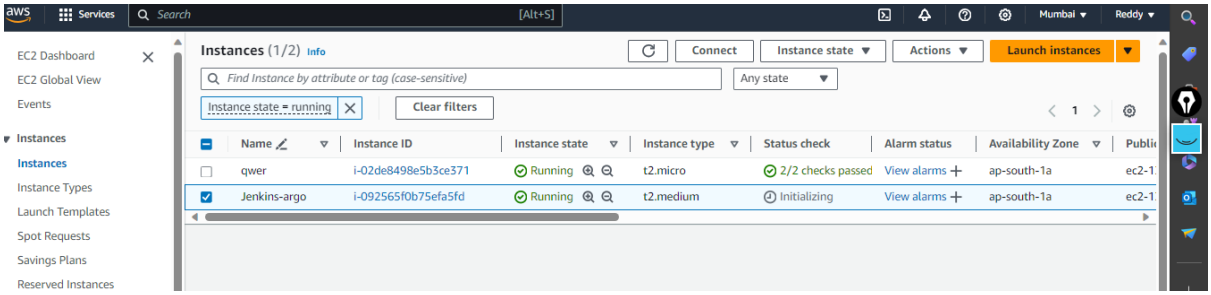


Clone repo and go to terraform code folder open terminal before thet you have to create S3 bucket in AWS



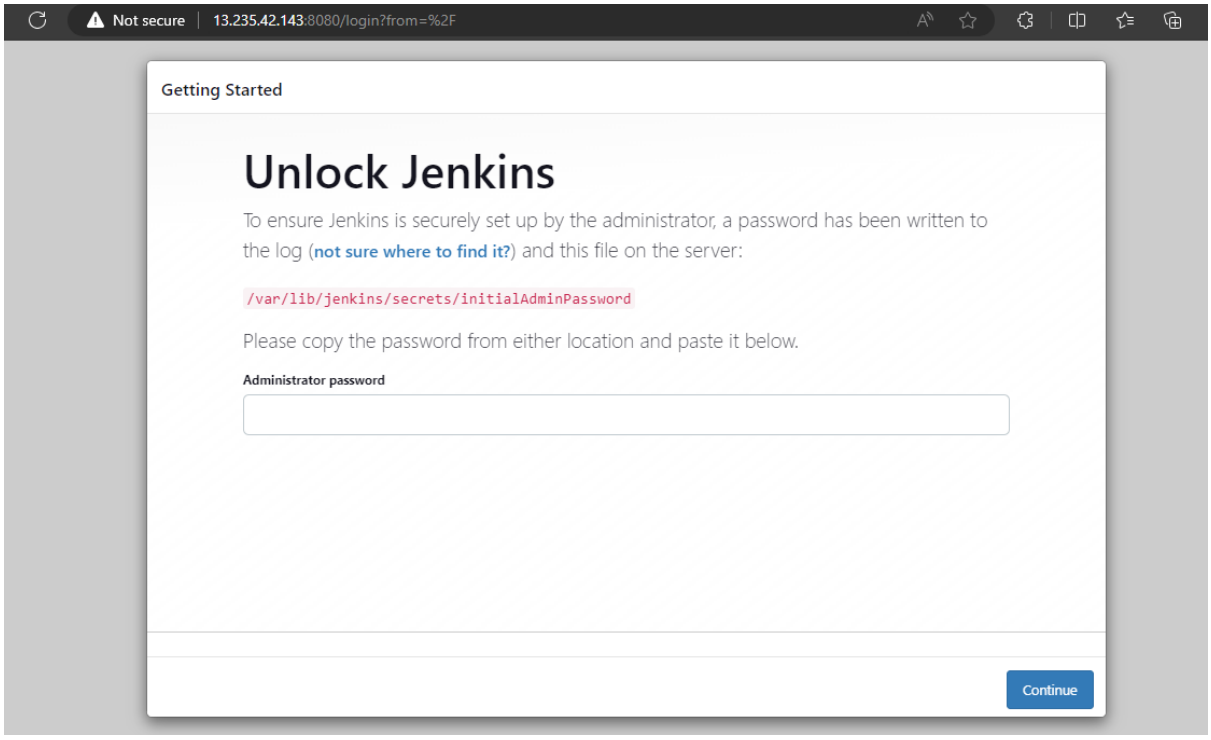
Then terraform apply it create a server in AWS console

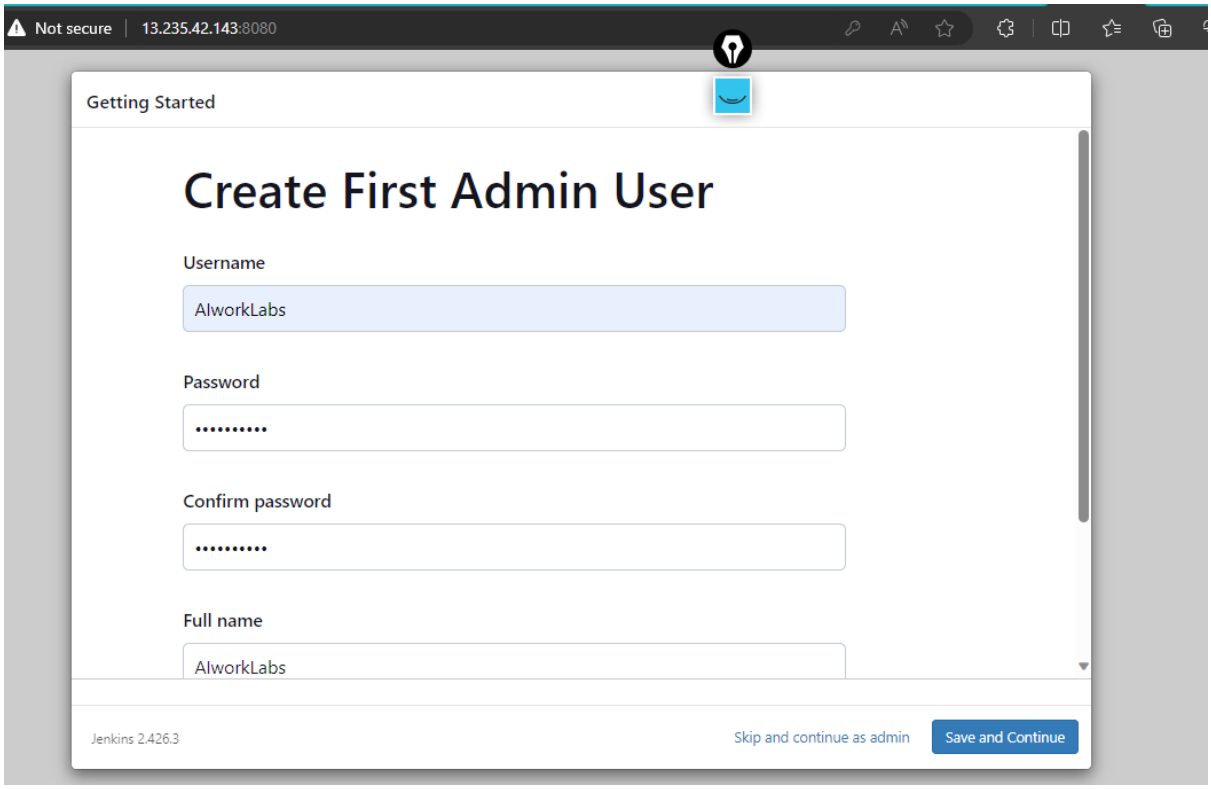




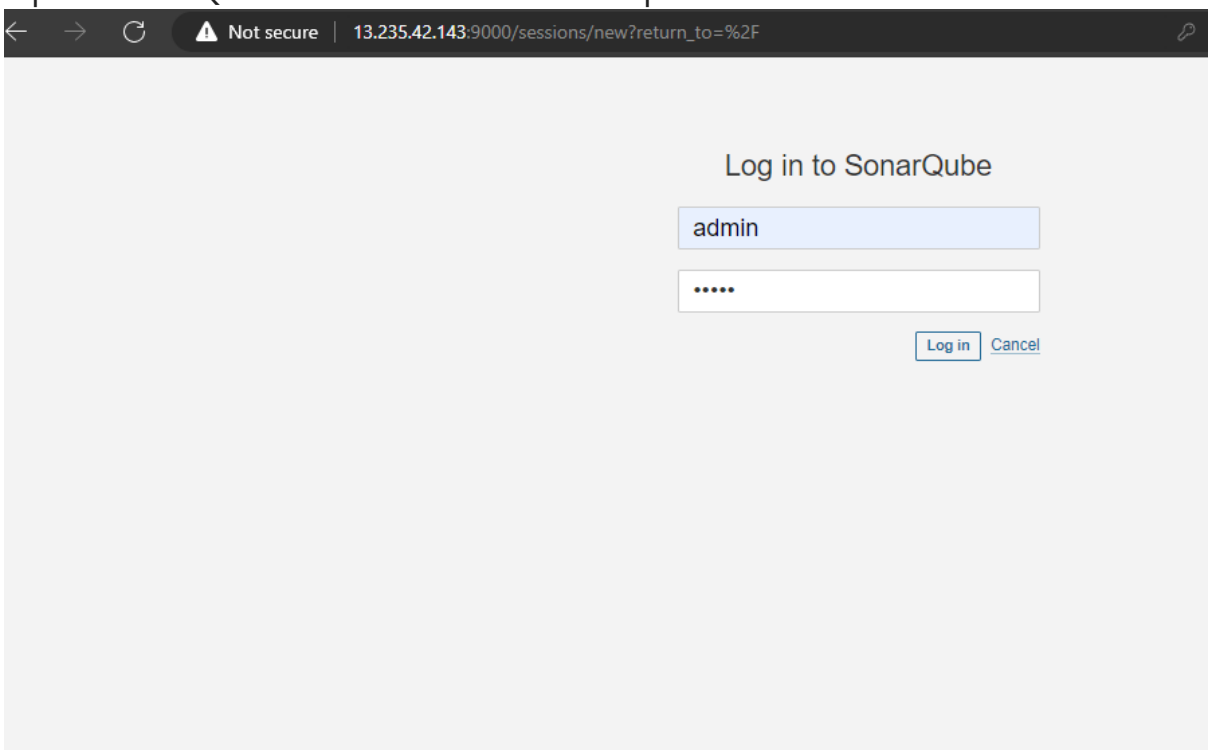
connect to Jenkins from this public IP

Connect to instance and get the Jenkins password





Open SonarQube same IP but different port number



Then update your password

The check the versions

```
root@ip-172-31-44-207:/home/ubuntu# trivy --version
aws --version
terraform --version
kubectl version
Version: 0.48.3
aws-cli/2.15.15 Python/3.11.6 Linux/6.2.0-1017-aws exe/x86_64.ubuntu.22 prompt/off
Terraform v1.7.1
on linux_amd64
Client Version: v1.29.1
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
Error from server (Forbidden): <html><head><meta http-equiv='refresh' content='1;url=/login?from=%2Fve
ion%3Ftimeout%3D32s'/><script>window.location.replace('/login?from=%2Fversion%3Ftimeout%3D32s');</scri
></head><body style='background-color:white; color:white;'>


Authentication required
<!--
-->

</body></html>
root@ip-172-31-44-207:/home/ubuntu#
```

That is done now go to Jenkins and add a terraform plugin to provision the AWS EKS using the Pipeline Job.

Go to Jenkins dashboard –> Manage Jenkins –> Plugins

Available Plugins, Search for Terraform and install it.

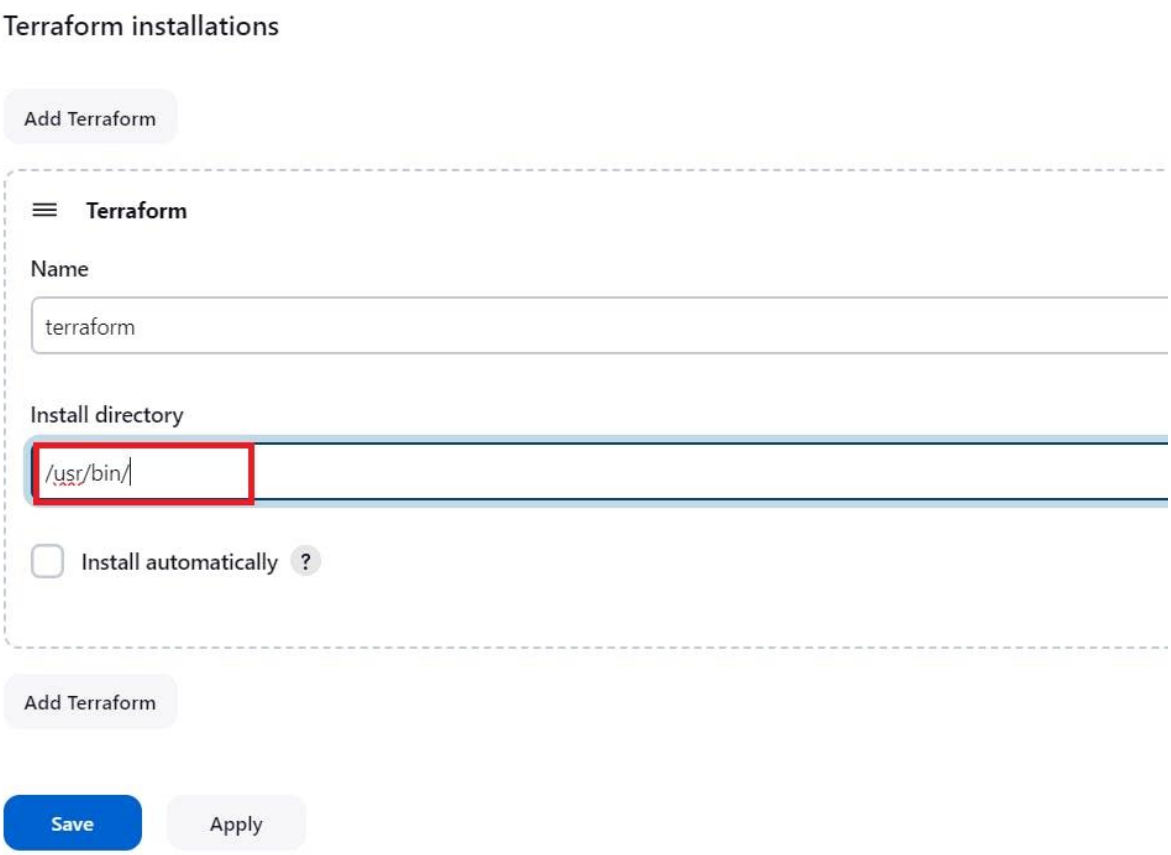
after installation
Now come back to Manage Jenkins –> Tools

Add the terraform in Tools

root@ip-172-31-44-207:/home/ubuntu# which terraform

/usr/bin/terraform

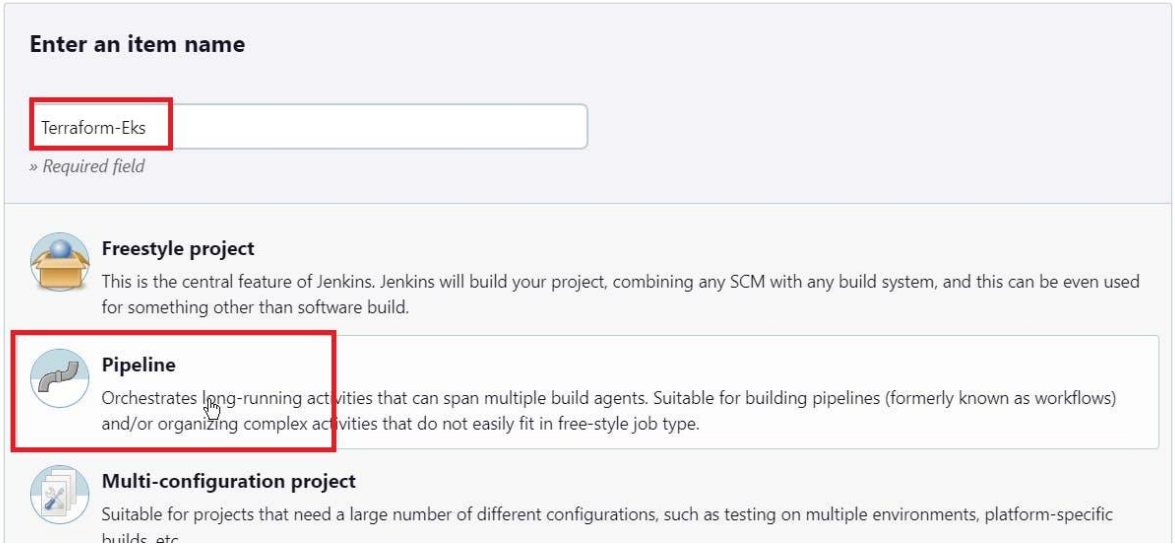
root@ip-172-31-44-207:/home/ubuntu#



Apply and save.

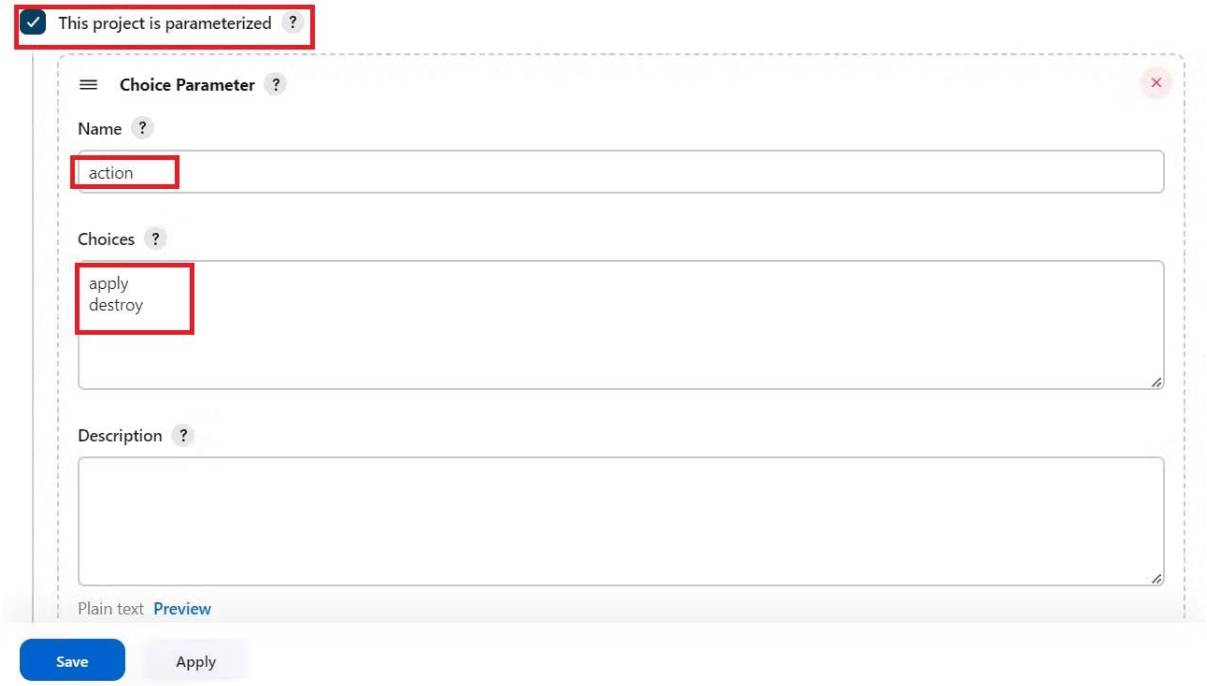
CHANGE YOUR S3 BUCKET NAME IN THE [BACKEND.TF]

Now create a new job for the Eks provision



I want to do this with build parameters to apply and destroy while building only.

you have to add this inside job like the below image



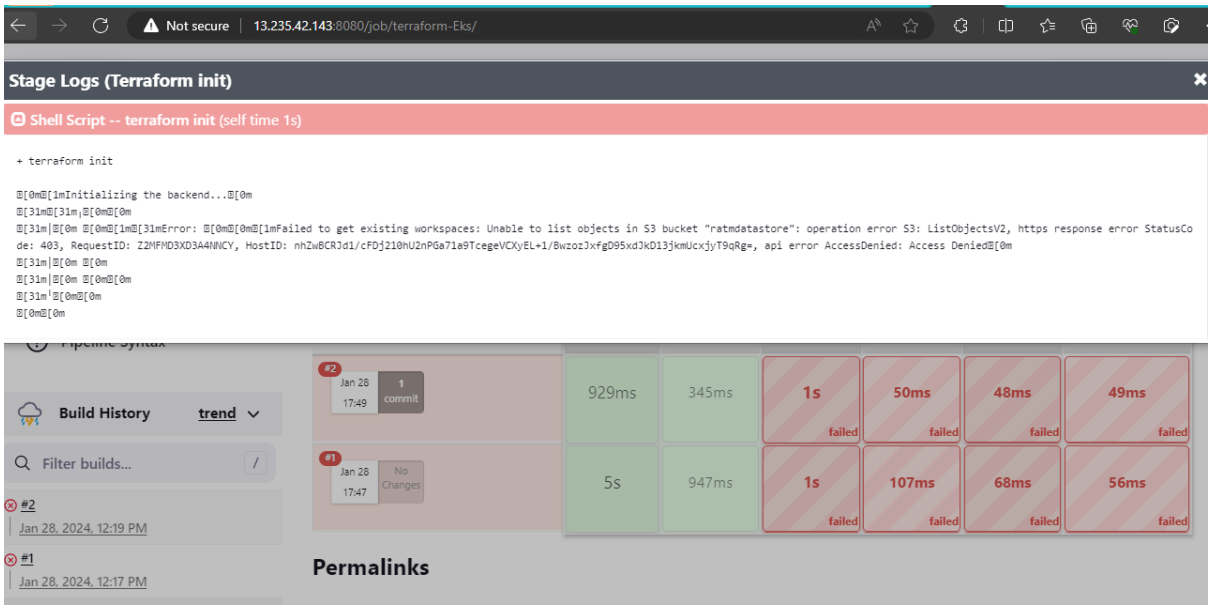
Let’s add a pipeline

add script

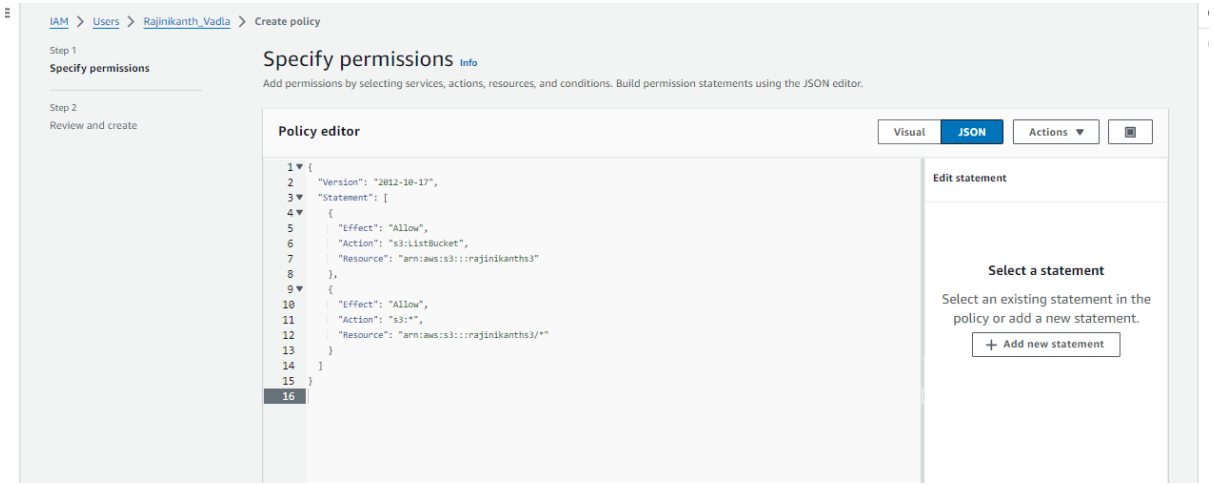
```
pipeline{
   agent any
   stages {
      stage('Checkout from Git'){
         steps{
            git branch: 'main', url:
'https://github.com/RajinikanthVadla/Tetris-V1.git'
         }
      }
```

```
stage('Terraform version'){
    steps{
        sh 'terraform --version'
    }
}
stage('Terraform init'){
    steps{
        dir('Eks-terraform') {
            sh 'terraform init'
          }
    }
}
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'
            }
        }
    }
  }
}
```

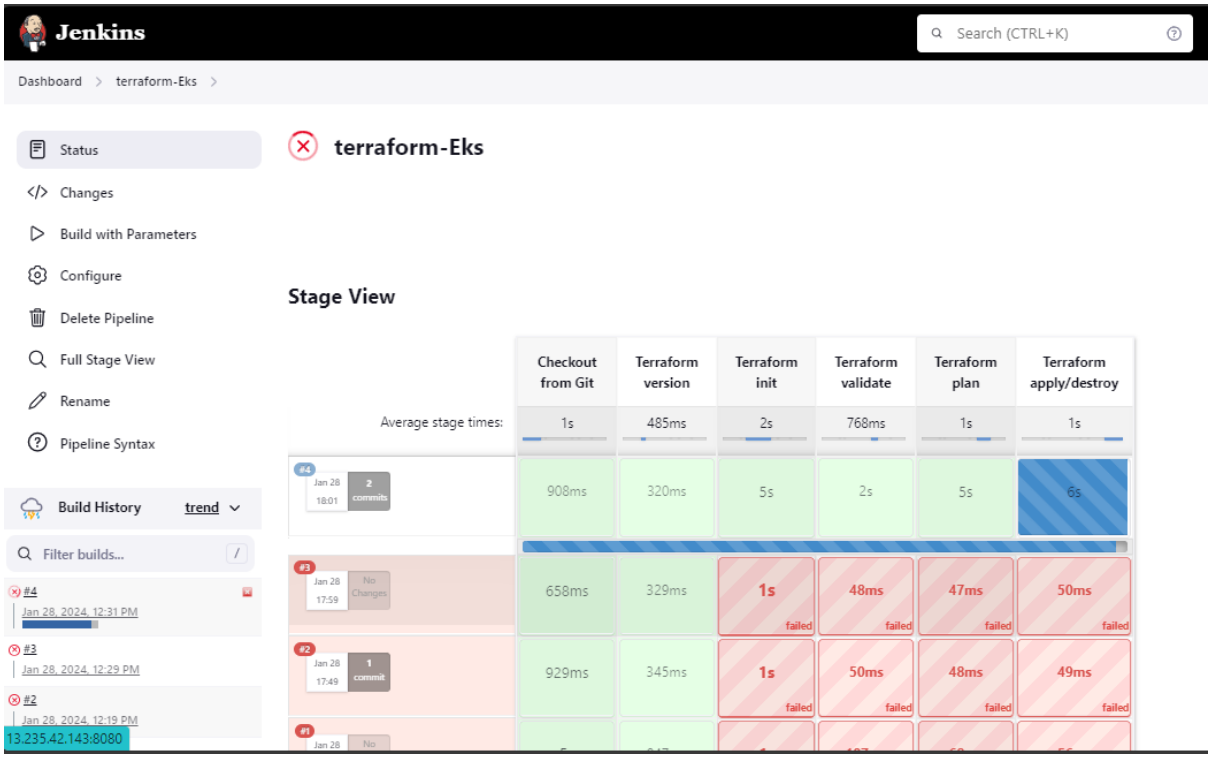


if its get failed ad in line policy in IAM user



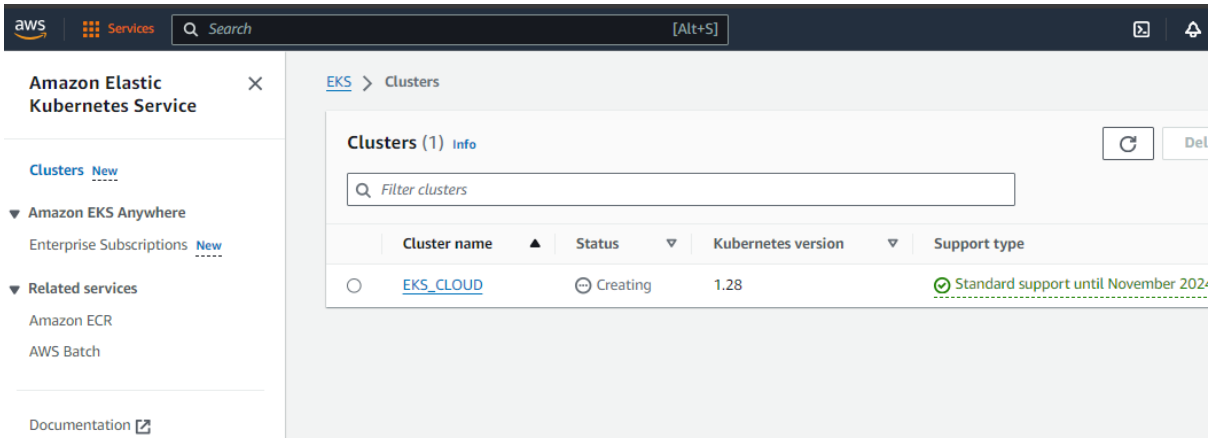

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::rajinikanths3"
    },
    {
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::rajinikanths3/*"
```

```
    }
  ]
}
```



it will take 10-15 mins time to create EKS cluster

Then we need to do dome configs in Jenkins



after created cluster go to Jenkins

Now let's build Tetris version 1

We need some plugins to complete this process

Go to Jenkins dashboard

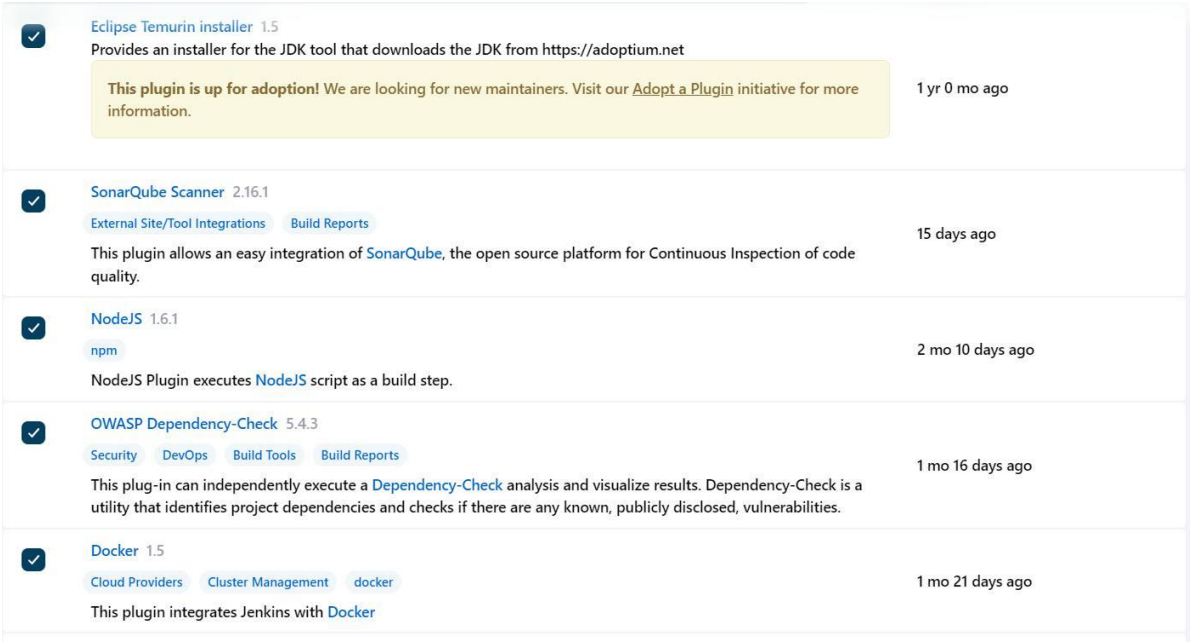
Manage Jenkins –> Plugins –> Available Plugins

Search for the Below Plugins

```
Eclipse Temurin installer

Sonarqube Scanner

NodeJs

Owasp Dependency-Check

Docker

Docker Commons

Docker Pipeline
```

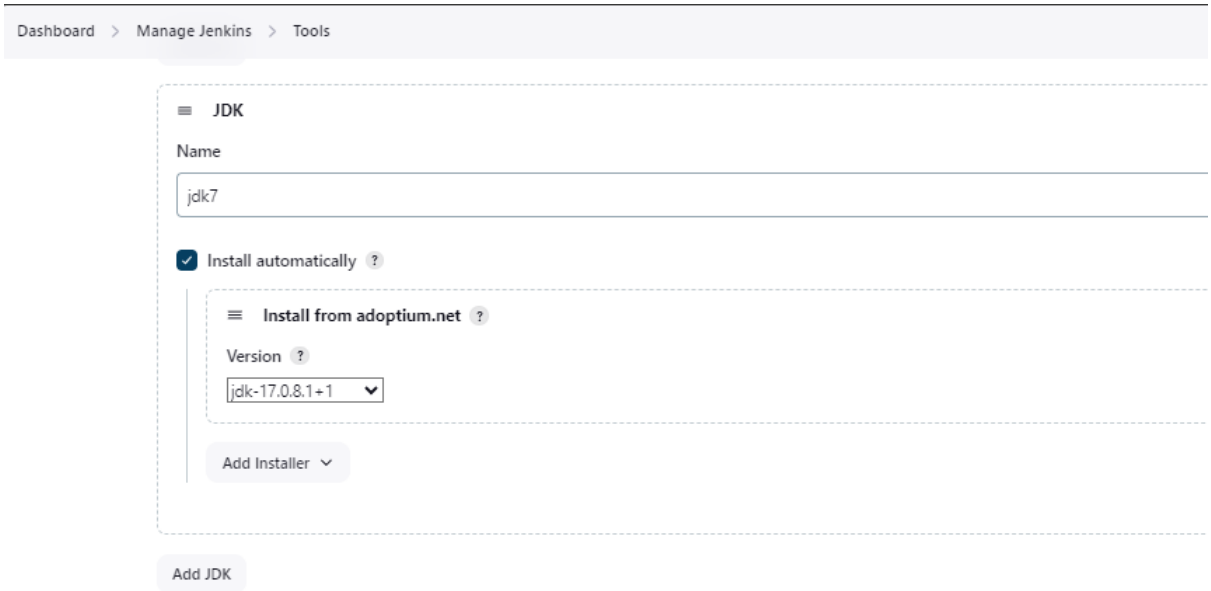
Docker API

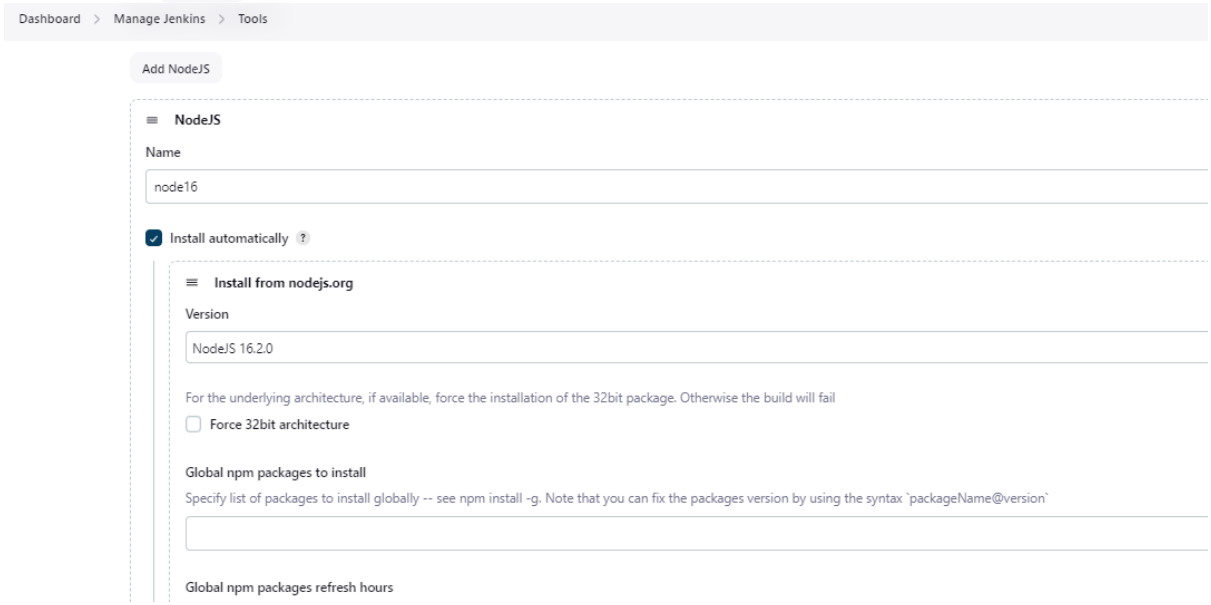
Docker-build-step



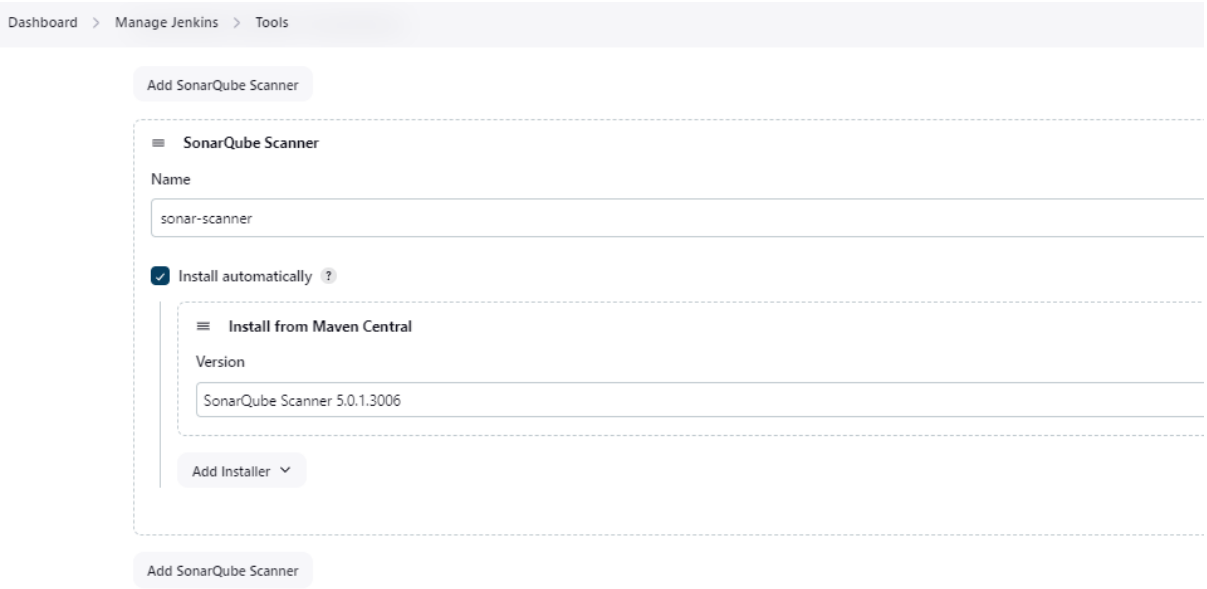
Configure in Global Tool Configuration

Goto Manage Jenkins → Tools → Install JDK(17) and NodeJs(16)→ Click on Apply and Save

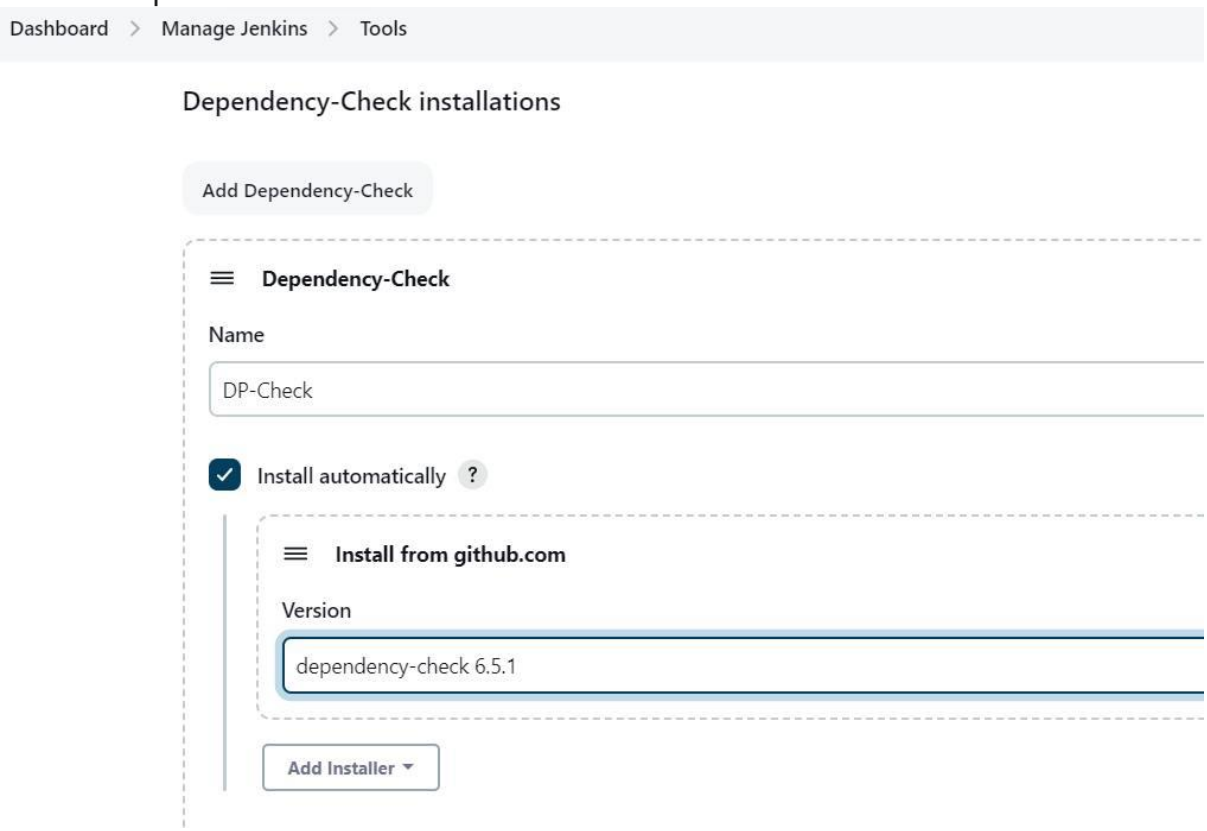




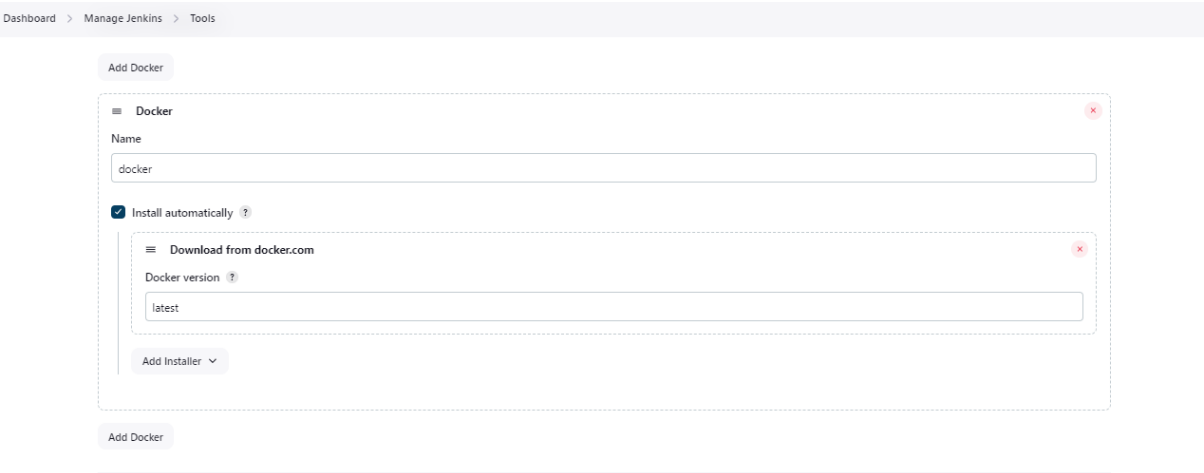
For Sonarqube use the latest version



For Owasp use the 6.5.1 version



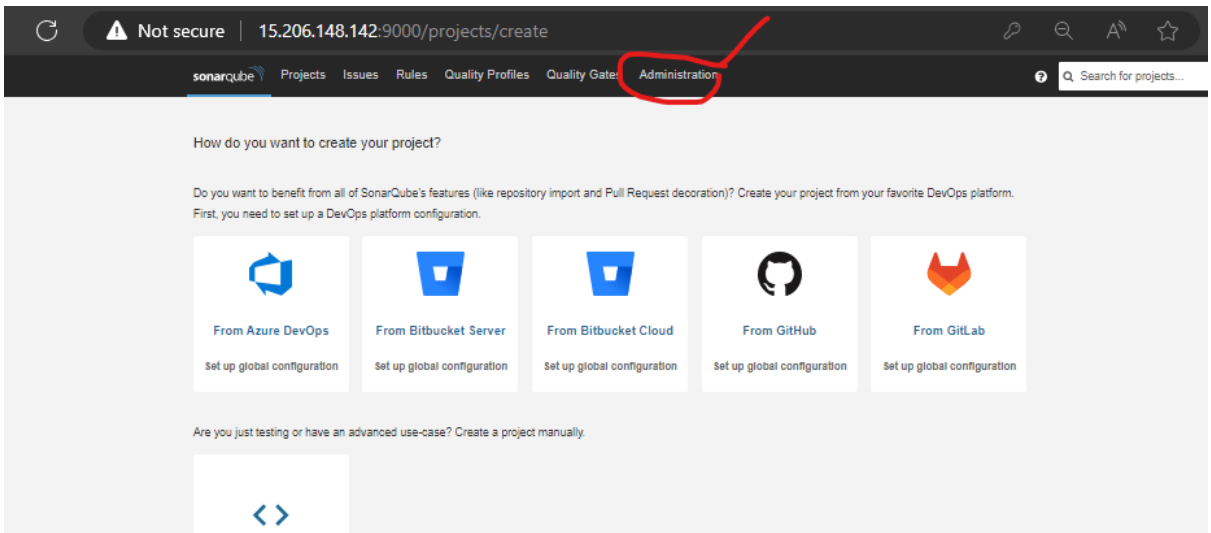
Use the latest version of Docker

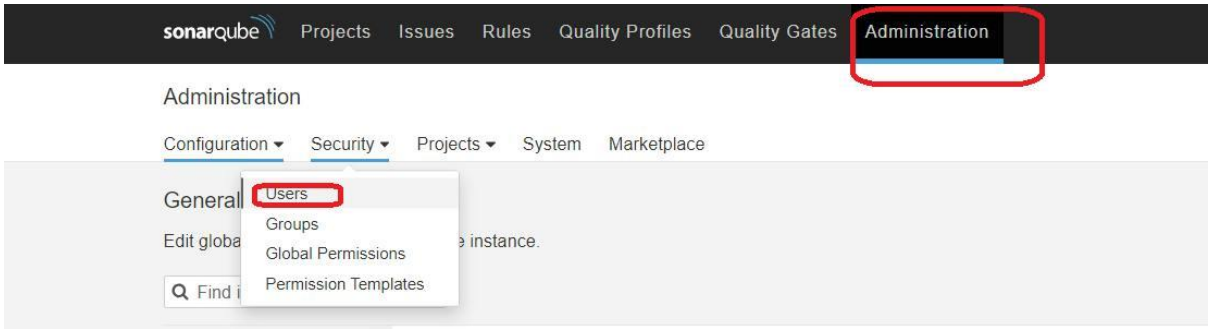


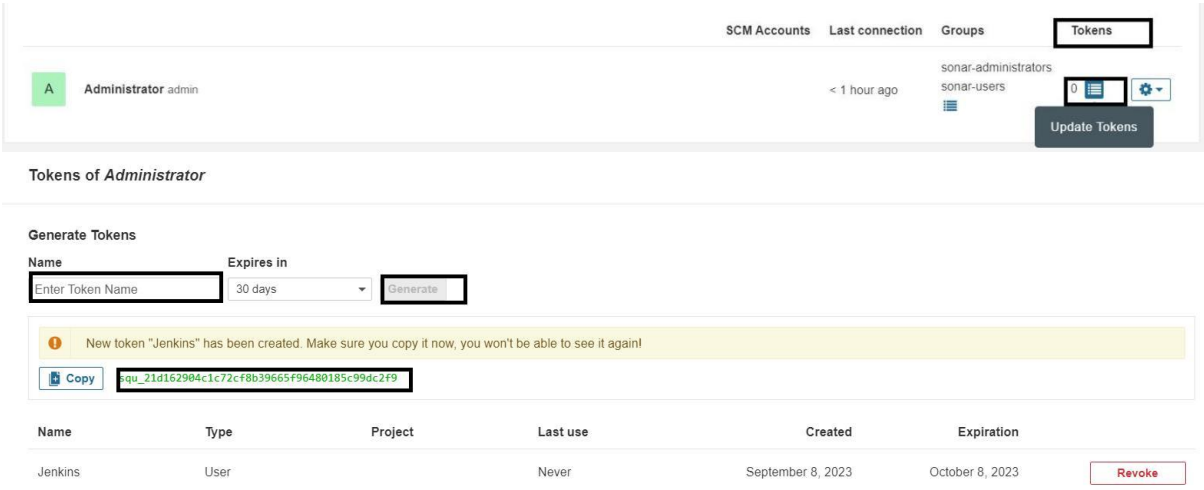
Click apply and save.

Configure Sonar Server in Manage Jenkins

Grab the Public IP Address of your EC2 Instance, Sonarqube works on Port 9000, so <Public IP>:9000. Goto your Sonarqube Server. Click on Administration → Security → Users → Click on Tokens and Update Token → Give it a name → and click on Generate Token

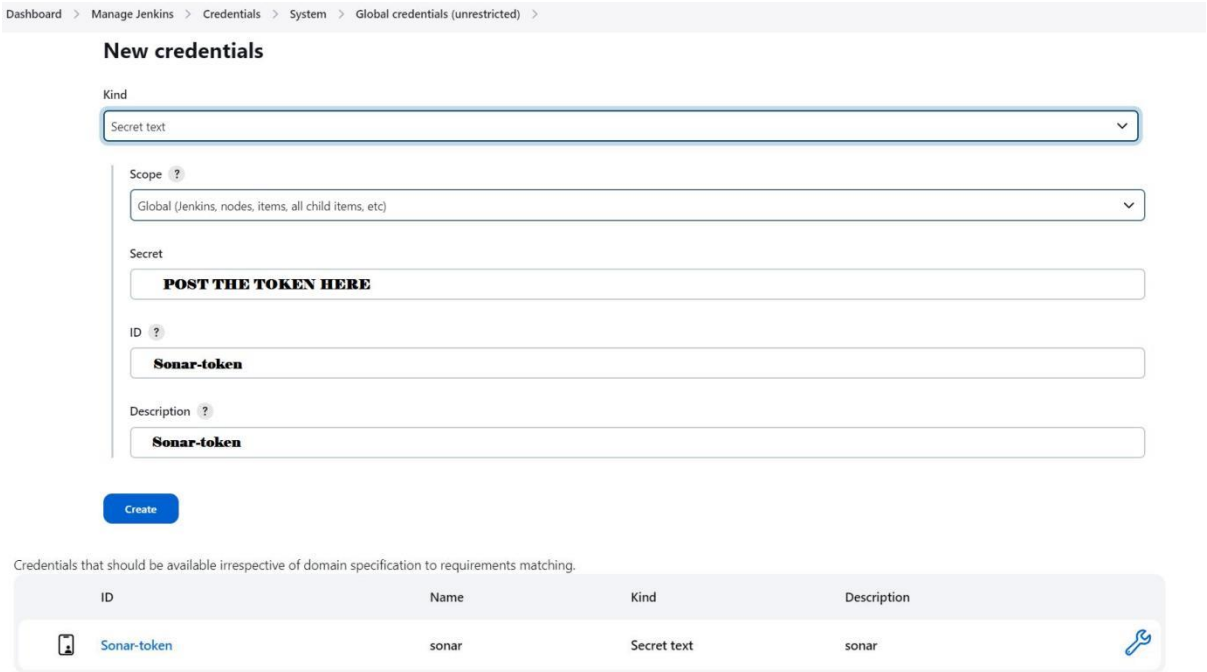




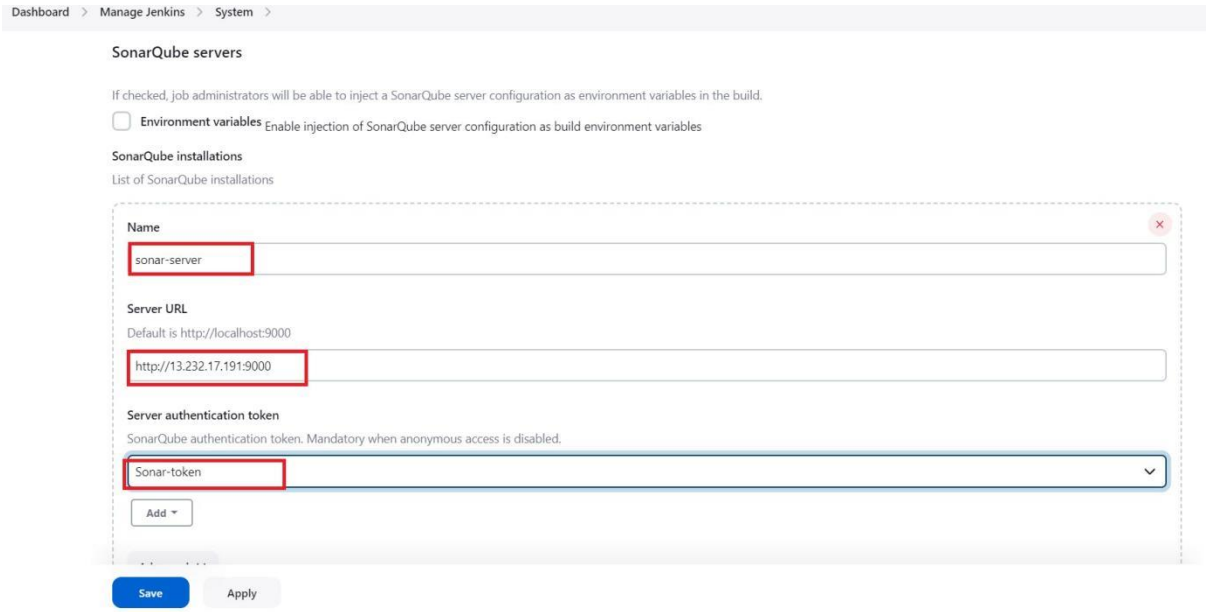


copy Token

Goto Jenkins Dashboard → Manage Jenkins → Credentials → Add Secret Text. It should look like this



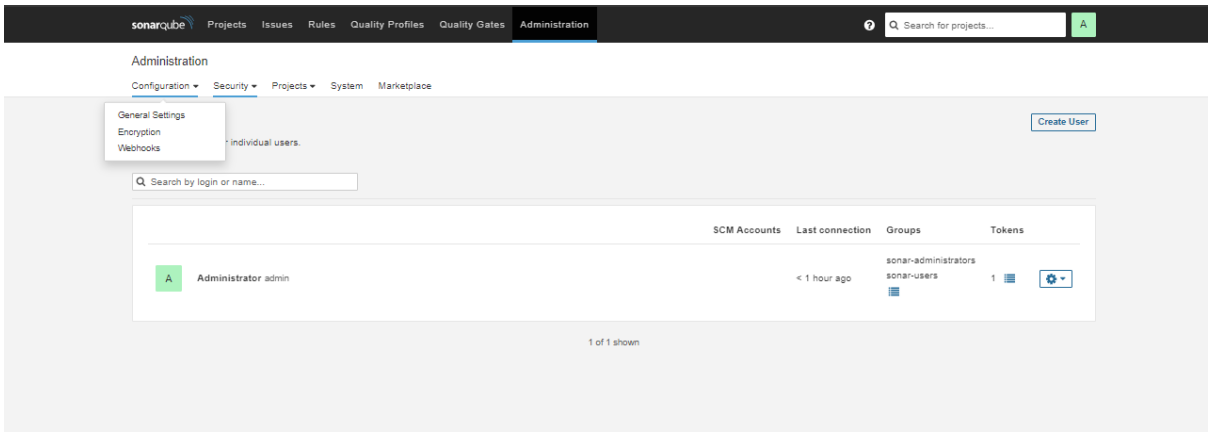
Now, go to Dashboard → Manage Jenkins → System and Add like the below image.

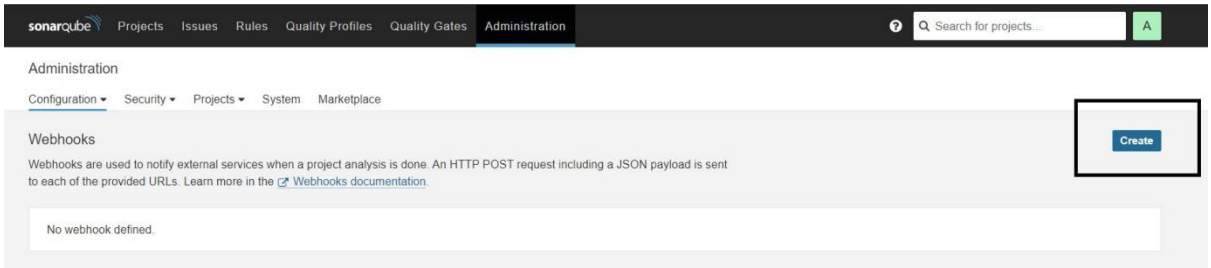


Click on Apply and Save

In the Sonarqube Dashboard add a quality gate also

Administration–> Configuration–>Webhooks

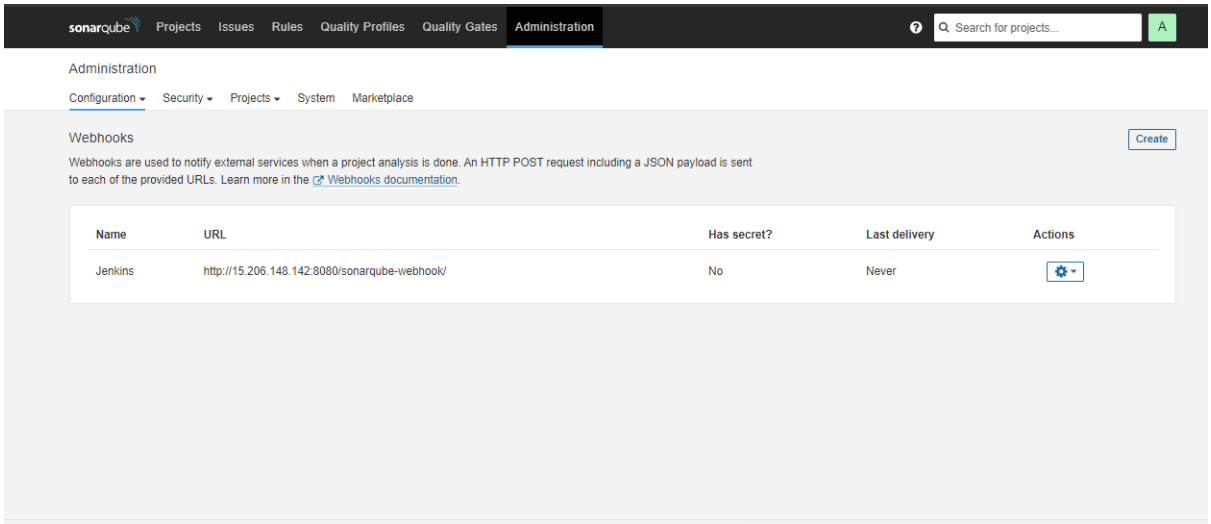




Add details

#in url section of quality gate

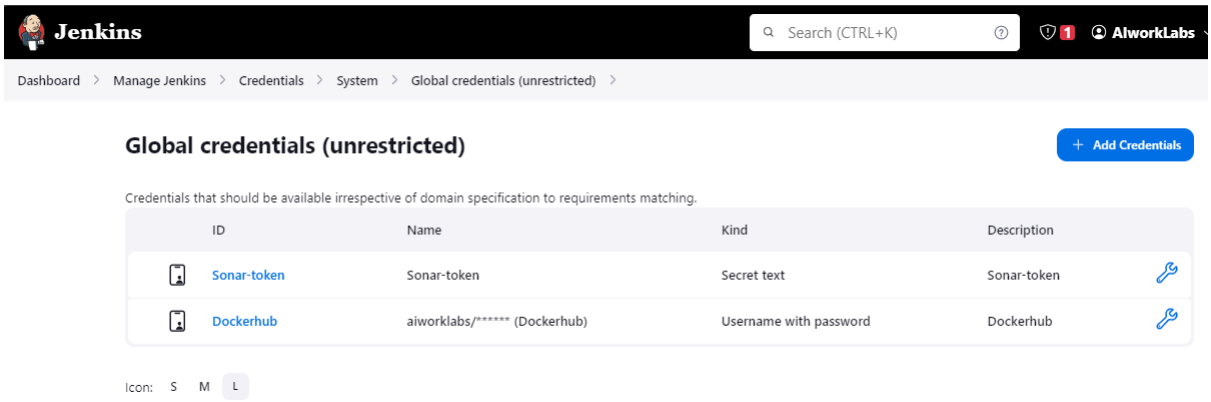
<http://jenkins-public-ip:8080>/sonarqube-webhook/



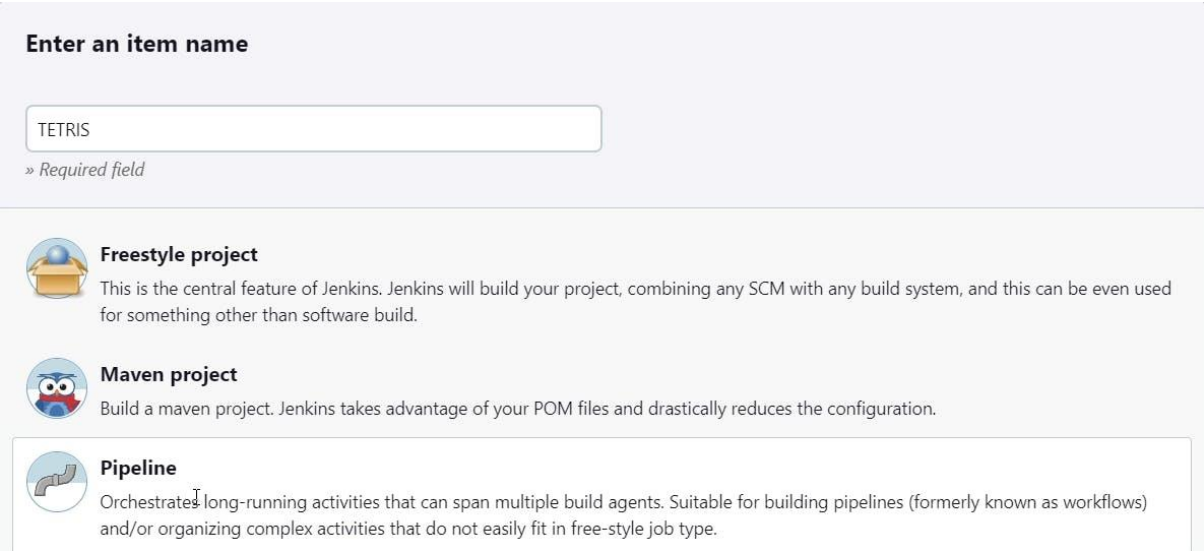
Now add Docker credentials to the Jenkins to log in and push the image

Manage Jenkins –> Credentials –> global –> add credential

Add DockerHub Username and Password under Global Credentials



Now let's create a new job for our pipeline create new one with name "docker"



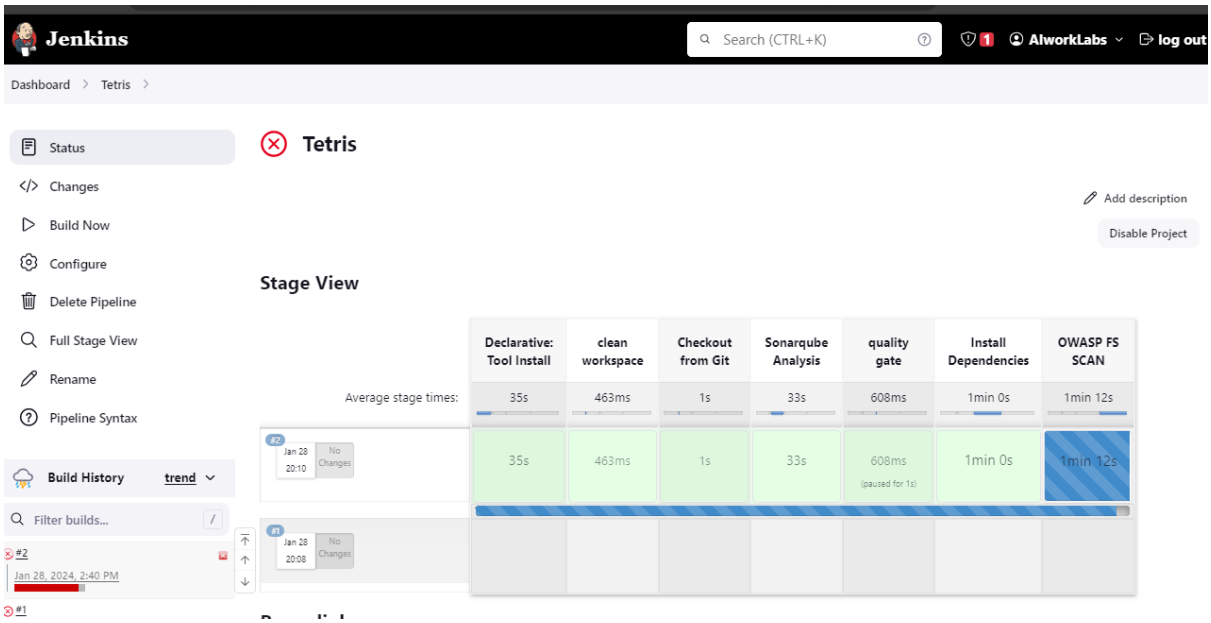
add this script

```
pipeline{
    agent any
    tools{
        jdk 'jdk17'
        nodejs 'node16'
    }
    environment {
        SCANNER_HOME=tool 'sonar-scanner'
    }
    stages {
        stage('clean workspace'){
            steps{
                cleanWs()
            }
```

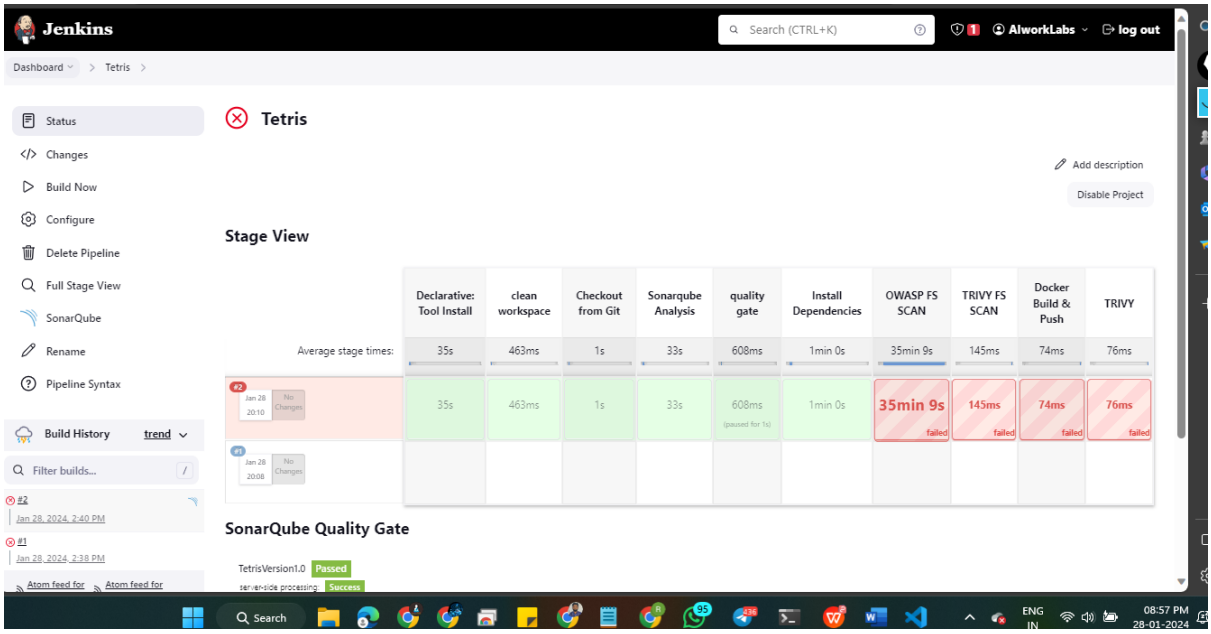
```
        }
        stage('Checkout from Git'){
            steps{
                git branch: 'main', url:
'https://github.com/RajinikanthVadla/Tetris-V1.git'
            }
        }
        stage("Sonarqube Analysis "){
            steps{
                withSonarQubeEnv('sonar-server') {
                    sh ''' $SCANNER_HOME/bin/sonar-scanner -
Dsonar.projectName=TetrisVersion1.0 \
                    -Dsonar.projectKey=TetrisVersion1.0 '''
                }
            }
        }
        stage("quality gate"){
           steps {
                script {
                    waitForQualityGate abortPipeline: false, credentialsId: 'Sonar-
token'
                }
            }
        }
        stage('Install Dependencies') {
            steps {
```

```
            sh "npm install"
        }
    }
    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.txt"
        }
    }
    stage("Docker Build & Push"){
        steps{
            script{
              withDockerRegistry(credentialsId: 'docker', toolName: 'docker'){
                  sh "docker build -t tetrisv1 ."
                  sh "docker tag tetrisv1 aiworklabs/workshop:latest "
                  sh "docker push aiworklabs/workshop:latest "
               }
            }
        }
```

```
        }
        stage("TRIVY"){
            steps{
                sh "trivy image sevenajay/tetrisv1:latest > trivyimage.txt"
            }
        }
    }
}
```

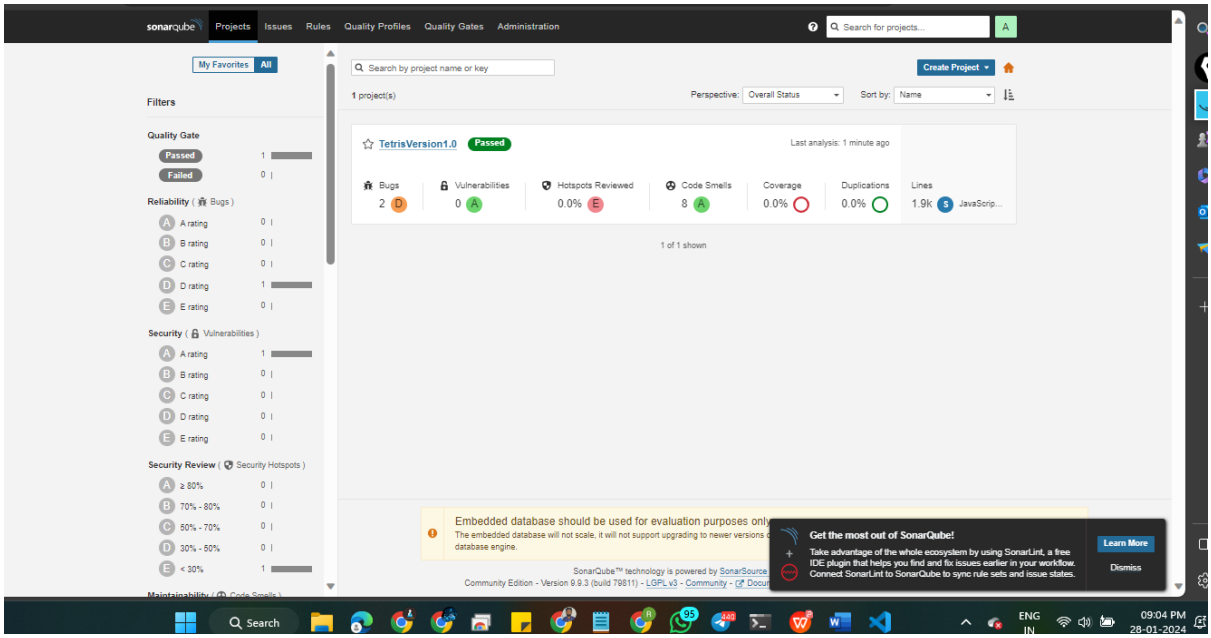


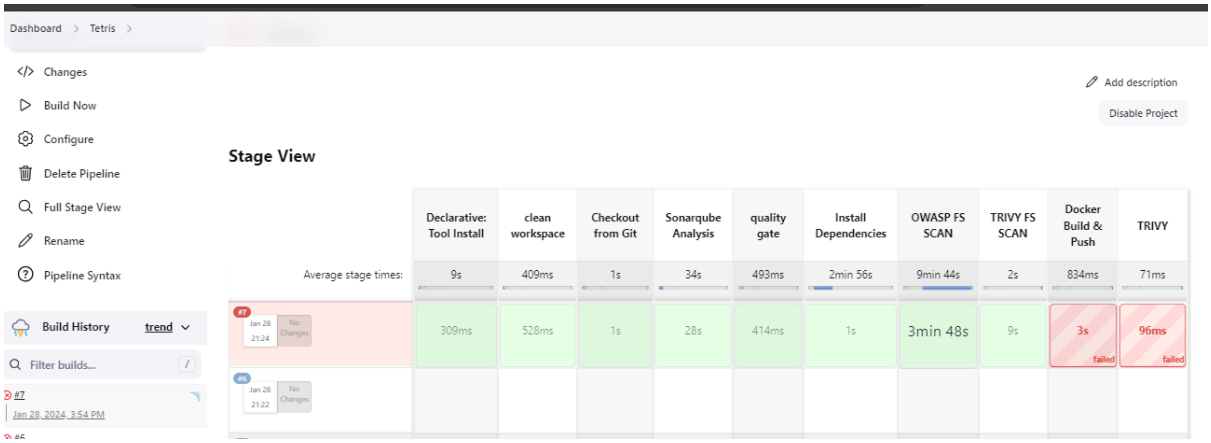
Let this pipeline complete
but I have a error



lets re-run this pipeline
squ_388c7c2a0a9e5141c0662e926667e43728c2ccef

Then after the you can see sonarqube dashboard





you have to get failure if not you doing copy paste or simply not doing anything ---- request try yourself to solve errors

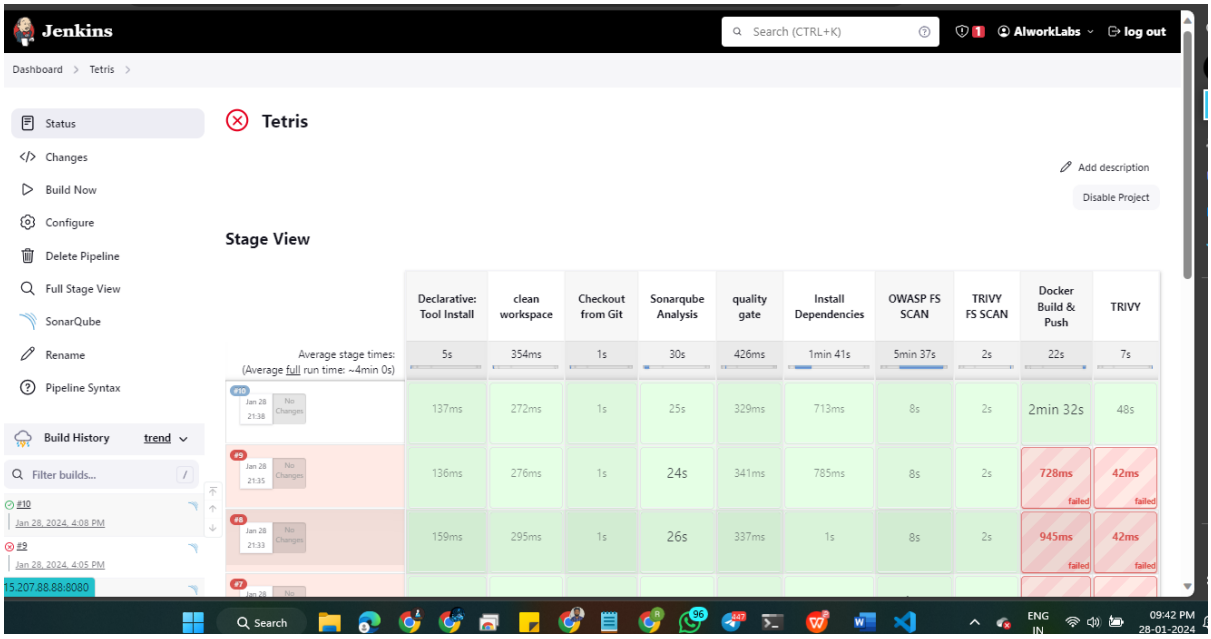
---

sudo usermod -aG docker ubuntu

newgrp docker
sudo chmod 777 /var/run/docker.sock

run above above commands if you are getting error

-



Let’s add the Image Updater stage to the Pipeline

Go to Terminal of your Jenkins instance SSH and enter the below command

aws eks update-kubeconfig --name <CLUSTER NAME> --region <CLUSTER REGION>

aws eks update-kubeconfig--name EKS_CLOUD--region ap-south-1

then see kubectl get nodes in server jenkis-argo server

kubectl get nodes

ARGO CD SETUP

Let's install ArgoCD

ARGOCD INSTALLATION LINK

You will redirected to this page

Then run this commands

kubectl create namespace argocd

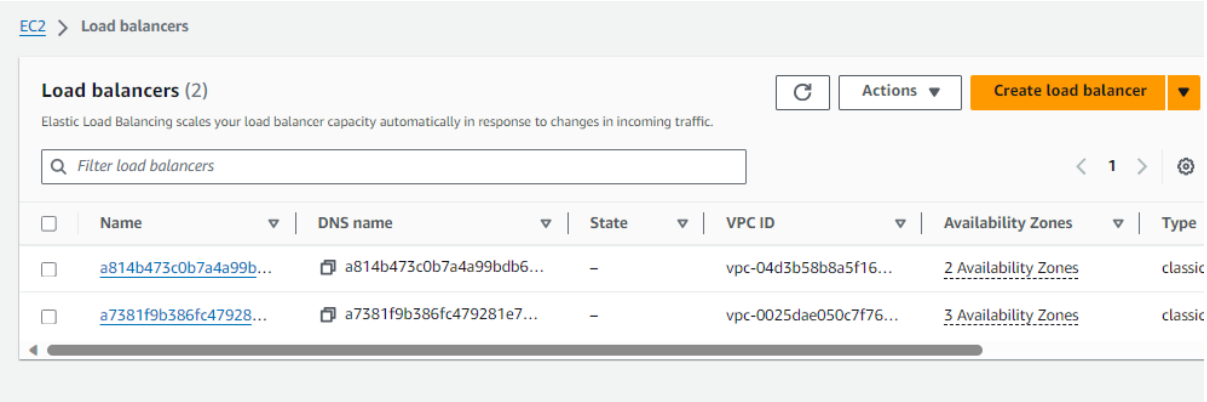
kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/v2.4.7/manifests/install.yaml

COMMANDS ARGOCD

By default, argocd-server is not publicly exposed. For this project, we will use a Load Balancer to make it usable:

Run This command mandatory

kubectl patch svc argocd-server -n argocd -p '{"spec": {"type": "LoadBalancer"}}'



load balancer will create automatically

&& patched with above command

Wait about 2 minutes for the LoadBalancer creation

sudo apt install jq -y

export ARGOCD_SERVER=`kubectl get svc argocd-server -n argocd -o json | jq --raw-output '.status.loadBalancer.ingress[0].hostname'`

when you run this command, it will export the hostname of the ArgoCD server's load balancer and store it in the ARGOCD_SERVER environment variable, which you can then use in other commands or scripts to interact with the ArgoCD server. This can be useful when you need to access the ArgoCD web UI or interact with the server programmatically.

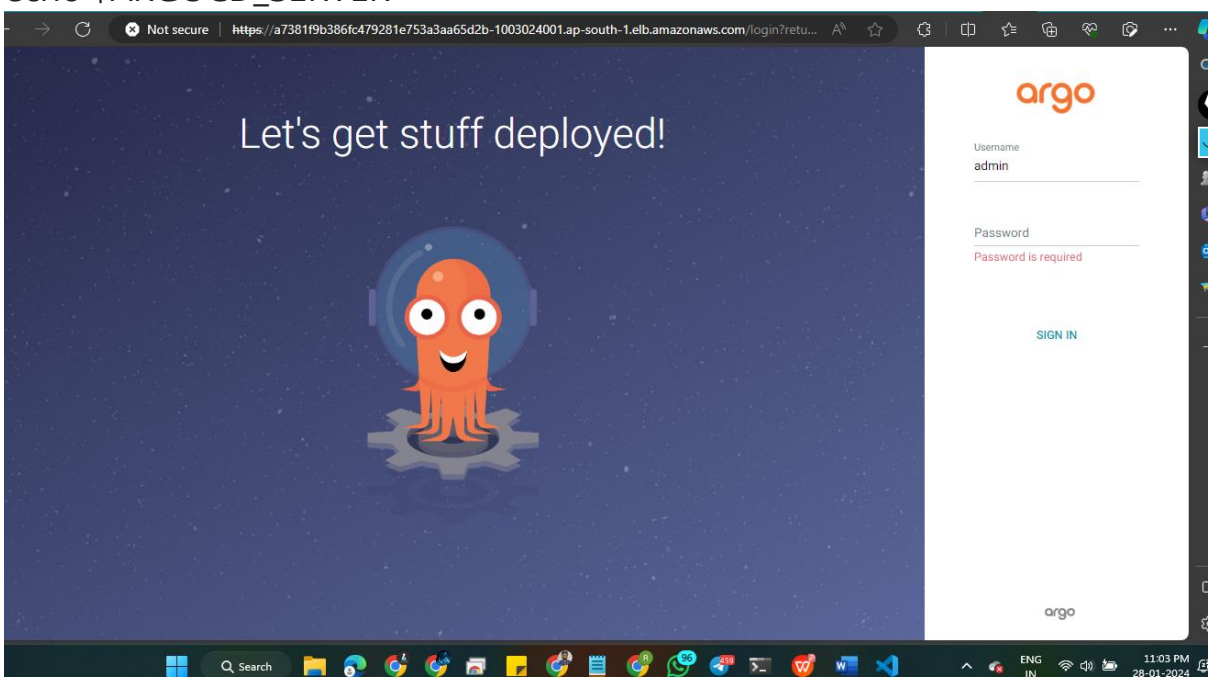
If run this command you will get the load balancer external IP

If you want to see your password provide the below command

echo $ARGO_PWD

Now copy the load balancer IP and paste it into the browser

echo $ARGOCD_SERVER

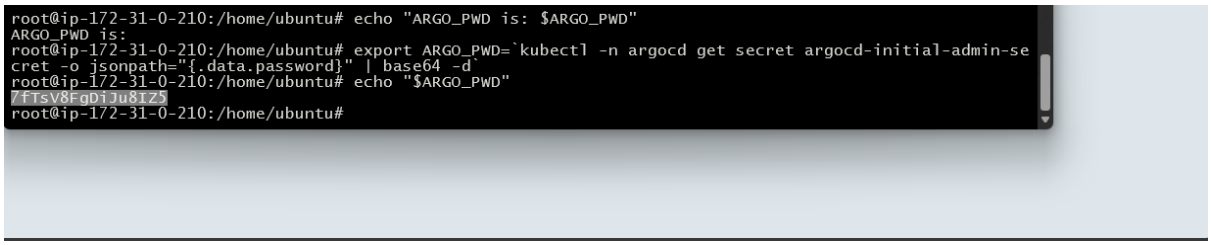


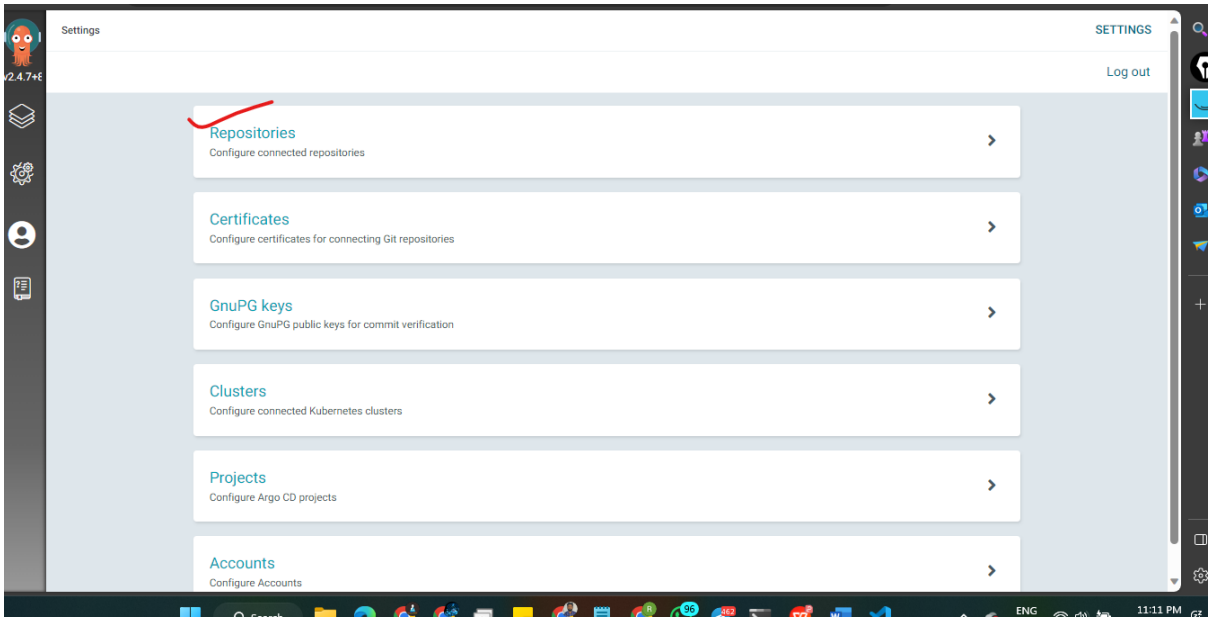
Username is admin

For the password, you have to provide the below command and copy it

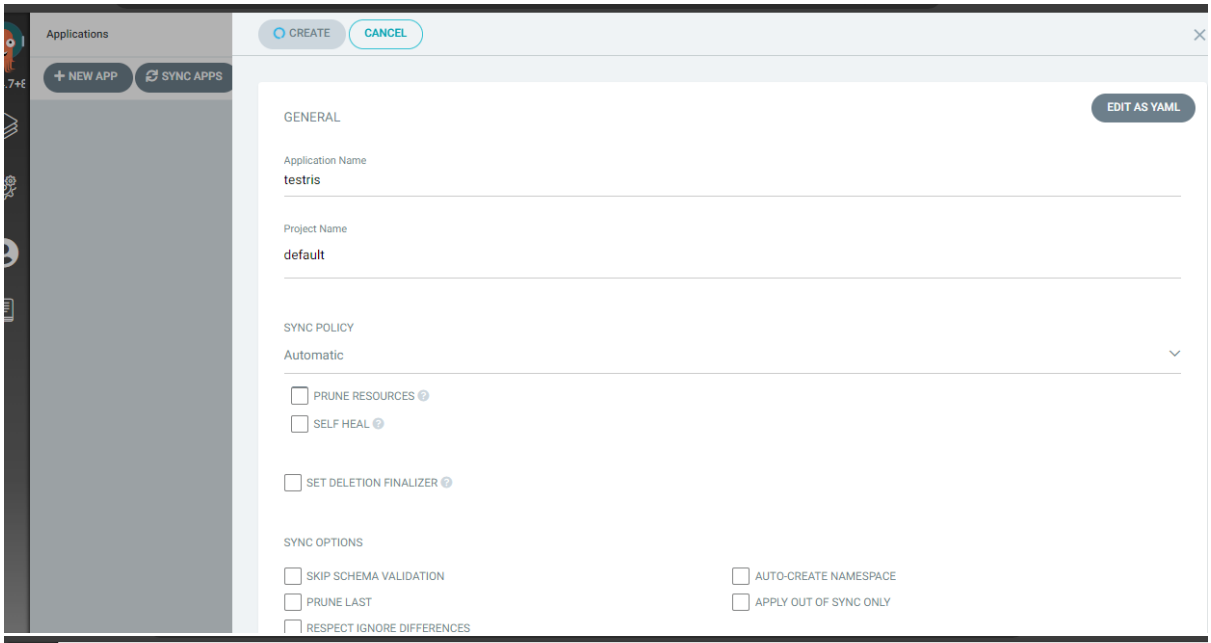
export ARGO_PWD=`kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}" | base64 -d`

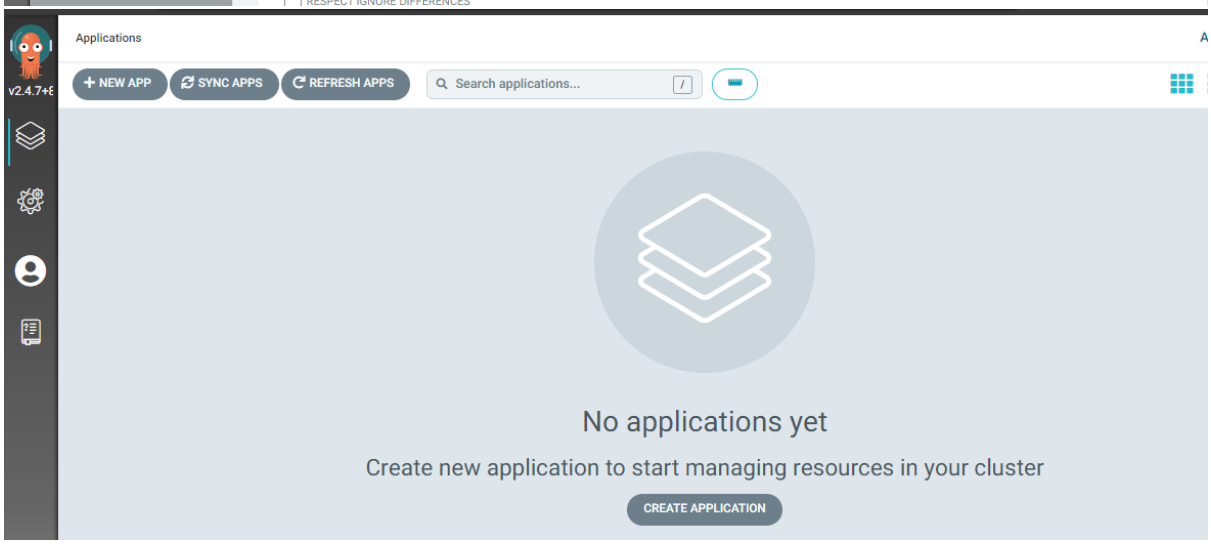
echo $ARGO_PWD

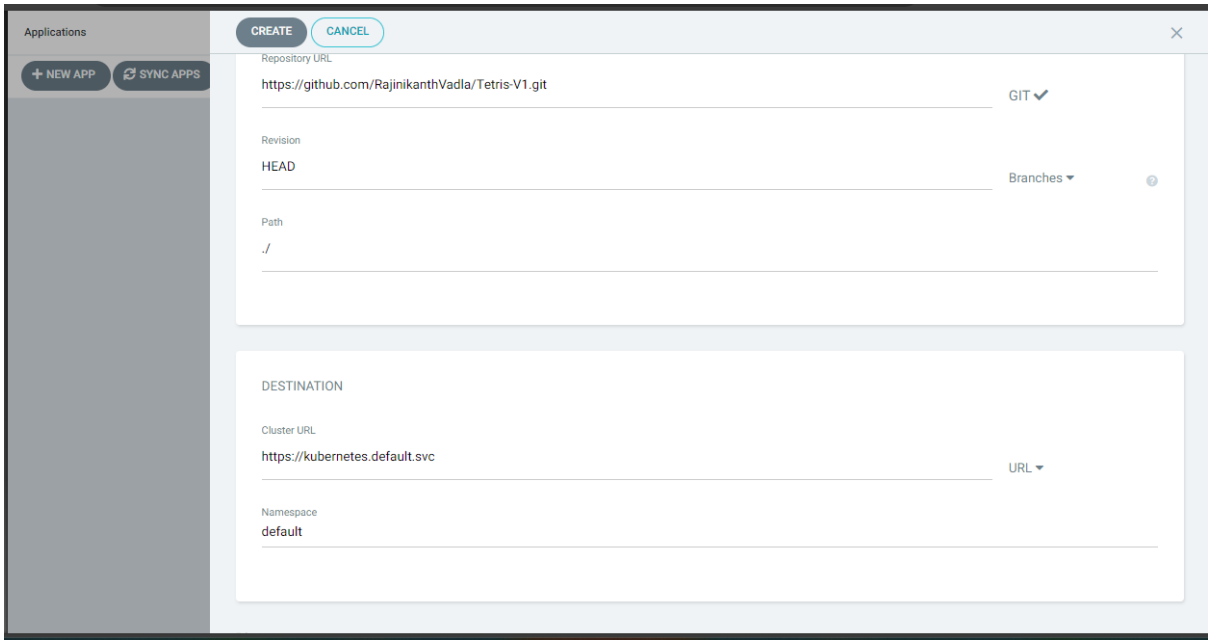




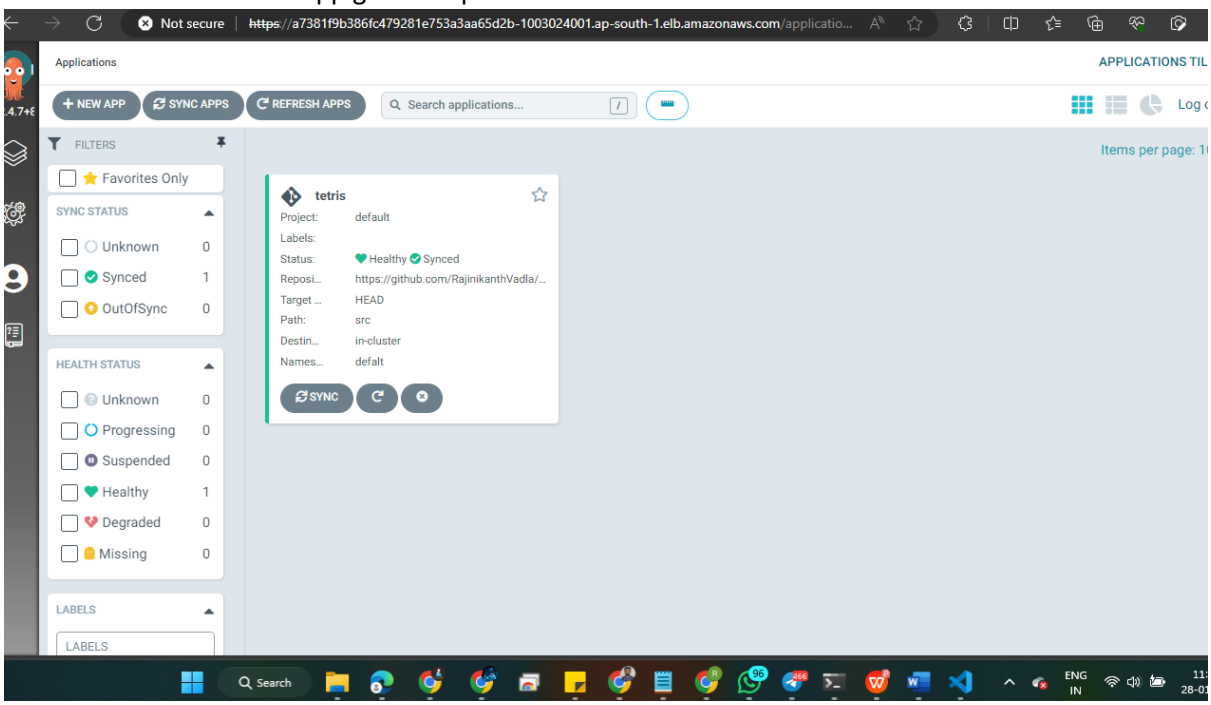
after configuring the git repo things

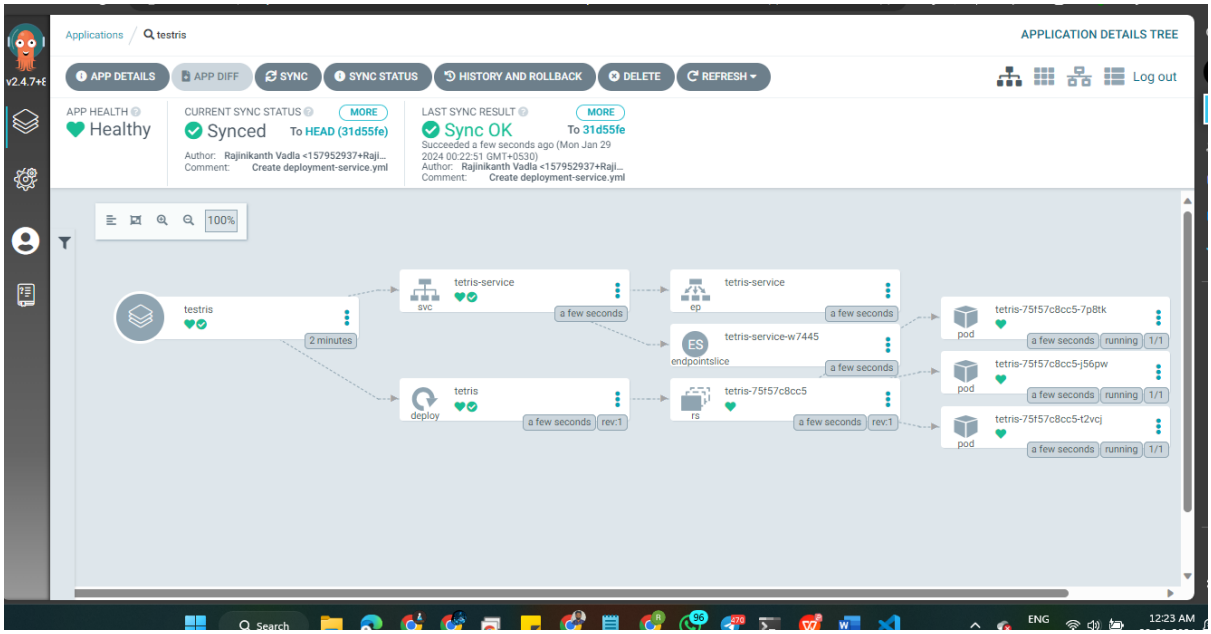

Applications
CREATE
CANCEL
NEW APP
SYNC APPS
GENERAL
EDIT AS YAML
Application Name
testris
Project Name
default
SYNC POLICY
Automatic
PRUNE RESOURCES
SELF HEAL
SET DELETION FINALIZER
SYNC OPTIONS
SKIP SCHEMA VALIDATION
AUTO-CREATE NAMESPACE
PRUNE LAST
APPLY OUT OF SYNC ONLY
RESPECT IGNORE DIFFERENCES


Applications
v2.4.7+8
NEW APP
SYNC APPS
REFRESH APPS
Search applications...
No applications yet
Create new application to start managing resources in your cluster
CREATE APPLICATION

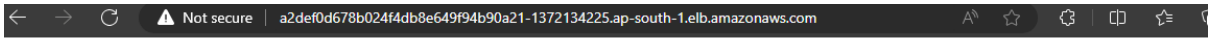


Then its will connect the app github repo





take LoadBalancer dns and map with dns



helm repo add grafana https://grafana.github.io/helm-charts

helm repo update

```
helm install grafana grafana/grafana \
  --namespace monitoring --create-namespace
```

```
kubectl get secret --namespace monitoring grafana -o jsonpath="{.data.admin-password}" | base64 --decode ; echo
```

```
kubectl expose deployment grafana --namespace monitoring --type=NodePort --name=grafana-service
```

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
```

```
helm install prometheus prometheus-community/kube-prometheus-stack --namespace monitoring
```

Add Prometheus Data Source to Grafana

Log in to Grafana.

Navigate to Configuration > Data Sources.

Add a new Prometheus data source:

URL: http://prometheus-server:80

8. Import Kubernetes Monitoring Dashboard

Grafana has pre-built dashboards to monitor Kubernetes nodes and pods.

Go to Create > Import.

Use the following dashboard IDs (examples):

6417 (Kubernetes Monitoring)

12740 (Kubernetes All-In-One)

```
helm install grafana grafana/grafana \
    --namespace monitoring \
    --set adminPassword='EKS!sAWSome' \
    --values grafana.yaml \
```

```
  --set service.type=LoadBalancer

kubectl get all -n Grafana
```

----------or----------

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
helm repo add grafana https://grafana.github.io/helm-charts
helm repo update

helm install prometheus prometheus-community/kube-prometheus-stack --namespace devops
helm install grafana grafana/grafana \
  --namespace devops \
  --set adminPassword='YourAdminPassword' \
  --set service.type=LoadBalancer


kubectl get svc --namespace devops grafana -o jsonpath='{.status.loadBalancer.ingress[0].ip}'

kubectl get svc  --namespace devops
kubectl edit svc prometheus1-kube-prometheu-prometheus -n monitoring
#to edit Clusert Ip to Load balancer
```