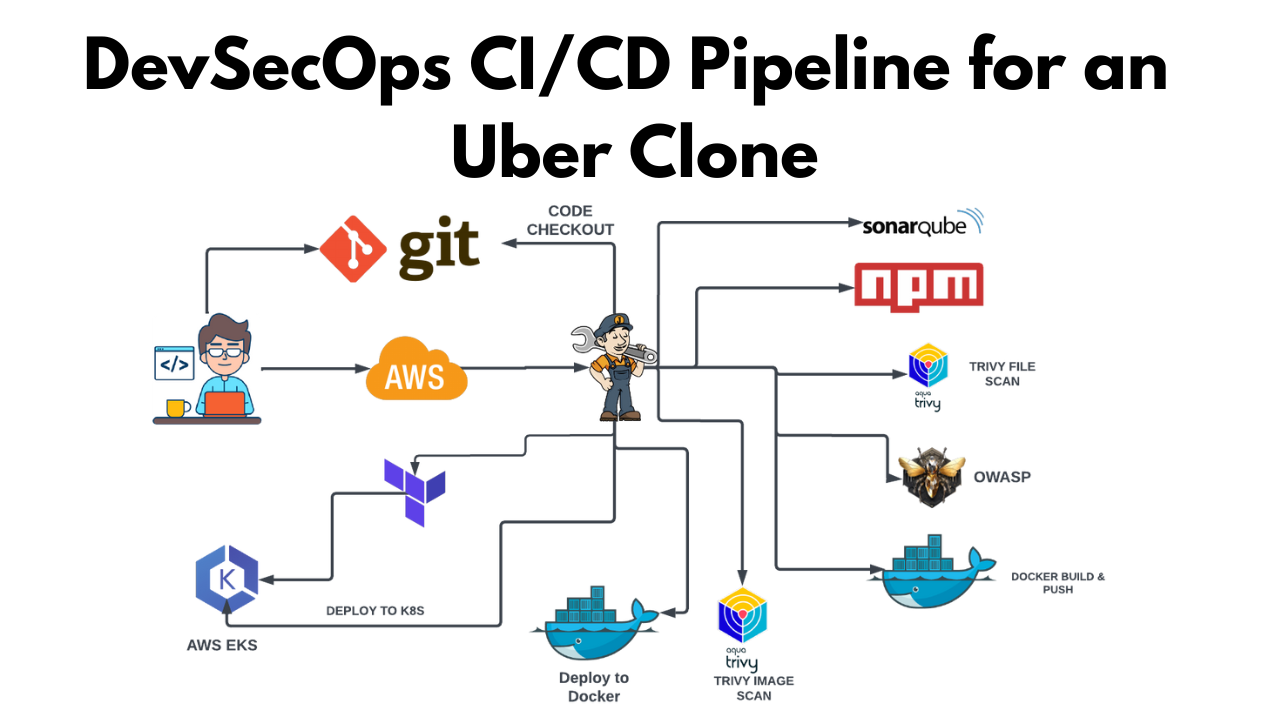
**\*\*DevSecOps CI/CD Pipeline for an Uber Clone\*\***



**GITHUB REPO:** [**https://github.com/kalyanvasu821/uber-clone.git**](https://github.com/kalyanvasu821/uber-clone.git)

**Step1: Launch an Ubuntu instance (**T2.large, 30GB storage**)**

**\*Note: Configure Security Group:**

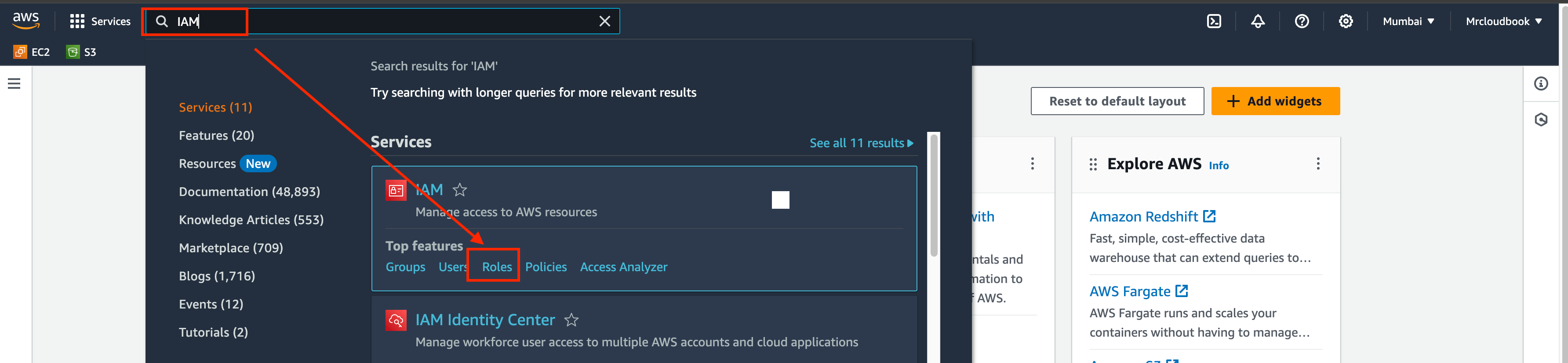
* Ensure the security group has the necessary inbound/outbound rules to allow access as required (Allow following ports: 8080 for jenkins, 9000 for sonarqube and 3000 for application)

>>Once the instance is launched, you can access it using the key pair and the instance’s public IP or DNS.

Ensure you have necessary permissions and follow best practices while configuring security groups and key pairs to maintain security for your EC2 instance.

**STEP 2: Create IAM role**

🡪 Search for IAM in the search bar of AWS and click on roles.



🡪Click on Create Role

A screenshot of a computer

Description automatically generated

🡪Select entity type as AWS service and Use case as EC2 and click on Next.

A screenshot of a computer

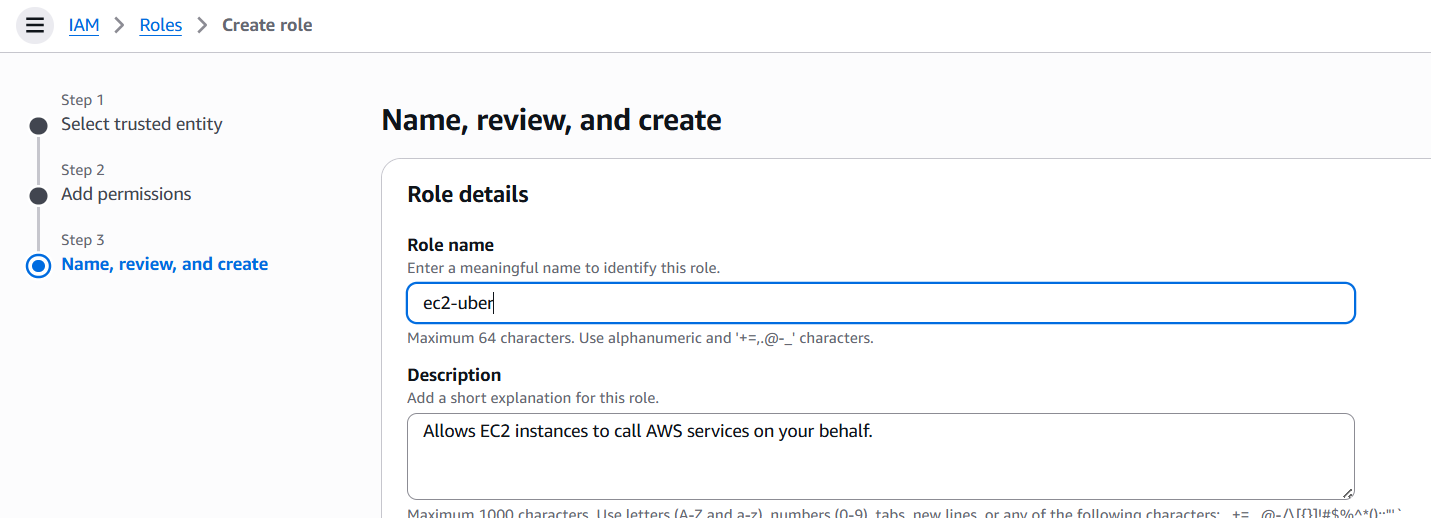
Description automatically generated

🡪For permission policy select Administrator Access (Just for learning purpose), click Next.

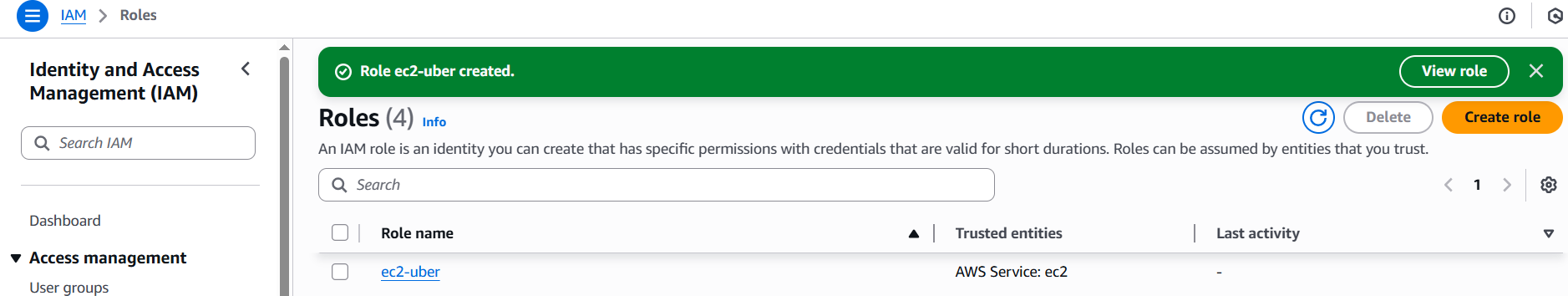
A screenshot of a computer

Description automatically generated

🡪Provide a Name for Role and click on Create role.



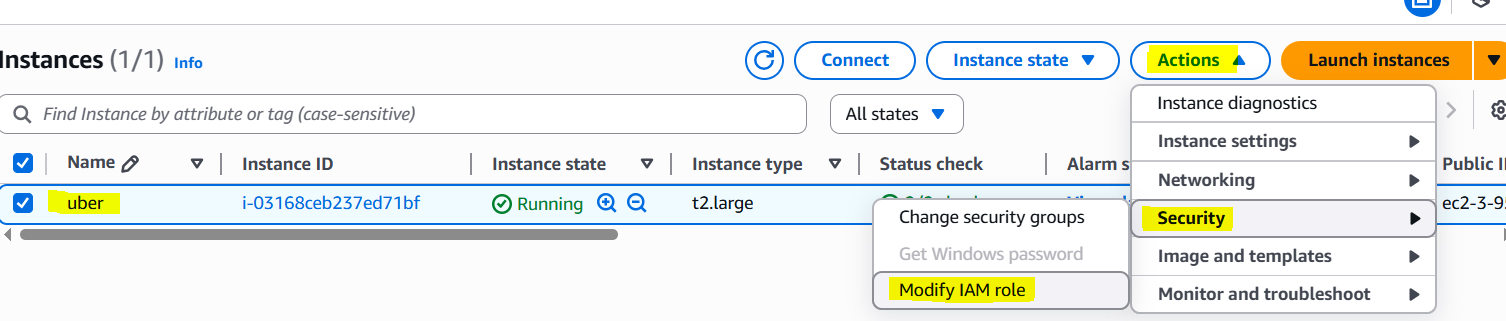
>>**Role is created.**



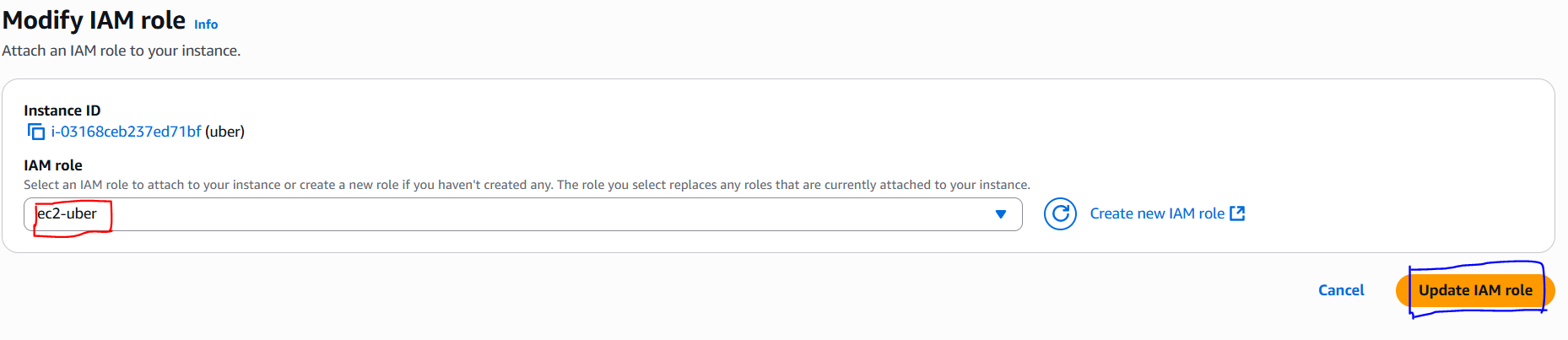
>>Now Attach this role to Ec2 instance that we created earlier, so we can provision cluster from that instance.

🡪Go to EC2 Dashboard and select the instance.

🡪Click on Actions –> Security –> Modify IAM role.



🡪Select the Role that created earlier and click on Update IAM role.



>>Connect the instance to Mobaxtreme or Putty or gitbash (we can connect directly form AWS console also)

**Step3: Installations of Packages**

create shell script in Ubuntu ec2 instance

sudo su #run from inside root or after connected to instance give sudo -i to become root user (learning purpose)

>> vi script1.sh (paste the below script in script1.sh and this script installs Jenkins, Docker)

#!/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

**#install docker**

# Add Docker's official GPG key:

sudo apt-get update

sudo apt-get install ca-certificates curl gnupg -y

sudo install -m 0755 -d /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

sudo chmod a+r /etc/apt/keyrings/docker.gpg

# Add the repository to Apt sources:

echo \

  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \

  $(. /etc/os-release && echo "$VERSION\_CODENAME") stable" | \

  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin -y

sudo usermod -aG docker ubuntu

newgrp docker

>>Now provide executable permissions to shell script and then run the script

--> chmod 777 script1.sh

--> sh script1.sh (./script1.sh)

>> vi script2.sh (paste the below script in script2.sh and this script installs trivy, Kubectl, Terraform, AWS Cli)

#!/bin/bash

# install trivy

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 on Jenkins

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

>>Now provide executable permissions to shell script and then run the script

--> chmod 777 script2.sh

--> sh script2.sh

>>Now check the versions of packages

docker --version

trivy --version

aws --version

terraform –version

kubectl version

A screenshot of a computer program

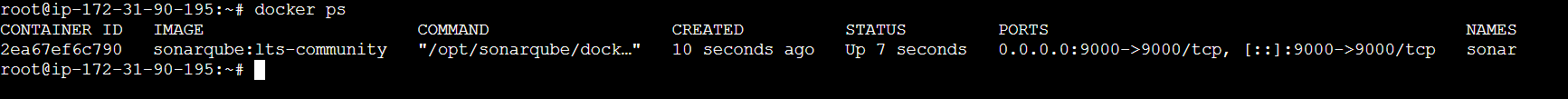
Description automatically generated

>>Install SonarQube: Provide executable permissions from Mobaxtreme/gitbash/putty

🡪sudo chmod 777 /var/run/docker.sock

🡪docker run -d --name sonar -p 9000:9000 sonarqube:lts-community (install sonarqube)

>>after above command, give docker ps in terminal



**Step4: Connect to Jenkins and Sonarqube**

>>Now copy the public IP address of ec2 and paste it into the browser shown like below

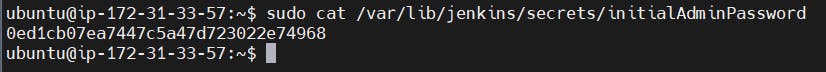
<Ec2-ip:8080> #you will see Jenkins login page

A screenshot of a computer

Description automatically generated

>>Connect your Instance to Putty or Mobaxtreme and provide the below command for the Administrator password

🡪sudo cat /var/lib/jenkins/secrets/initialAdminPassword



>>Now, install the suggested plugins.

A screenshot of a computer

Description automatically generated

Jenkins will now get installed and install all the libraries.

🡪Create an admin user

A screenshot of a login page

Description automatically generated

>>Click on save and continue. 🡪finish 🡪start using Jenkins

**Jenkins Dashboard**

A screenshot of a computer

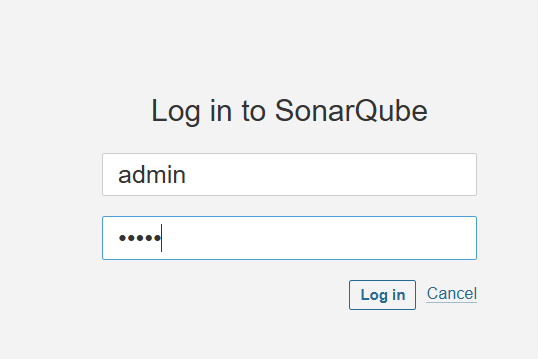
Description automatically generated

>>Now Copy the public IP again and paste it into a new tab in the browser with 9000

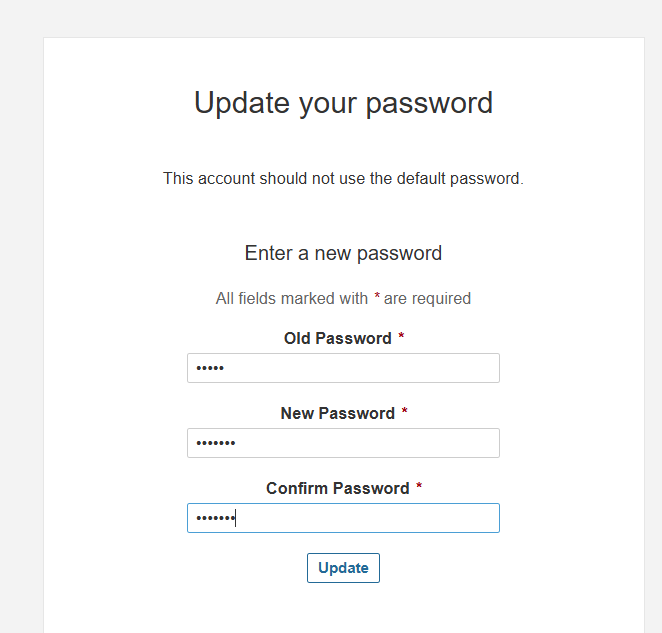
<ec2-ip:9000> #runs sonar container 🡪 Enter username and password, click on login and change password

Username: admin

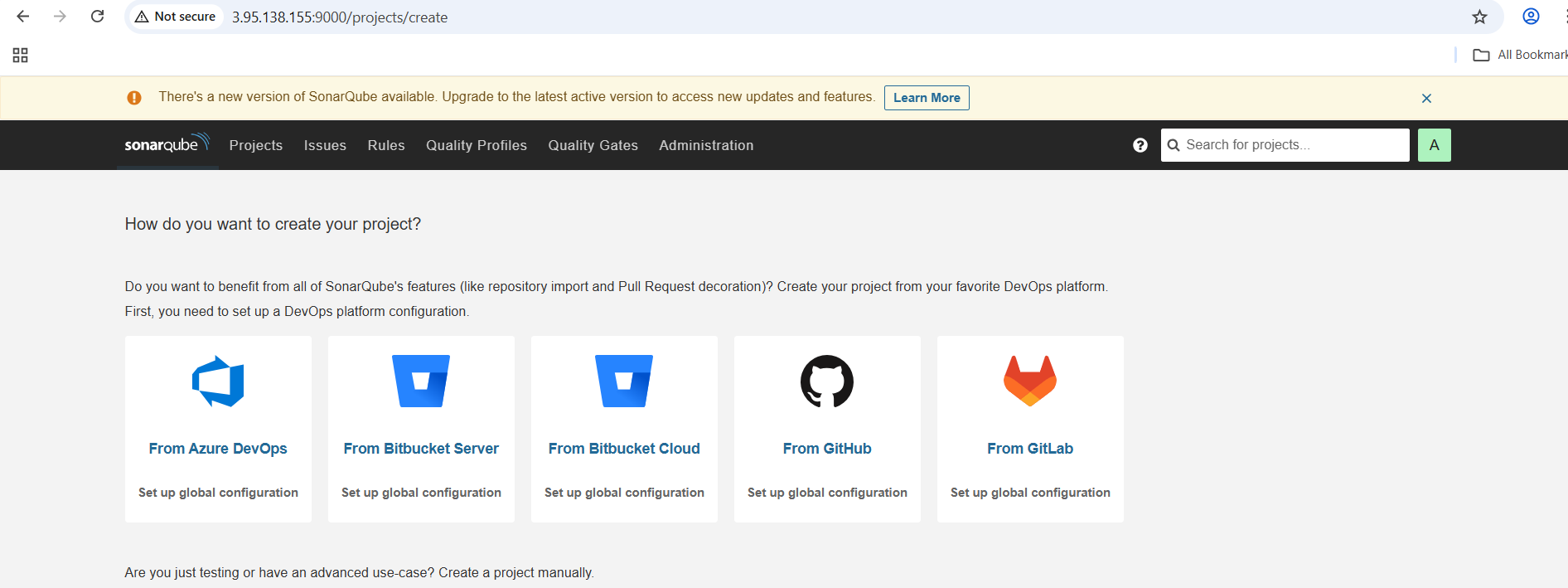
Password: admin



>> Update New password



**>>This is Sonar Dashboard**.



**Step5: Terraform plugin install and EKS provision**

>>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.

A screenshot of a computer

Description automatically generated

>>Go to moba/Putty/gitbash and use the below command to find the path to our Terraform (we will use it in the tools section of Terraform)

🡪which terraform

A screen shot of a computer

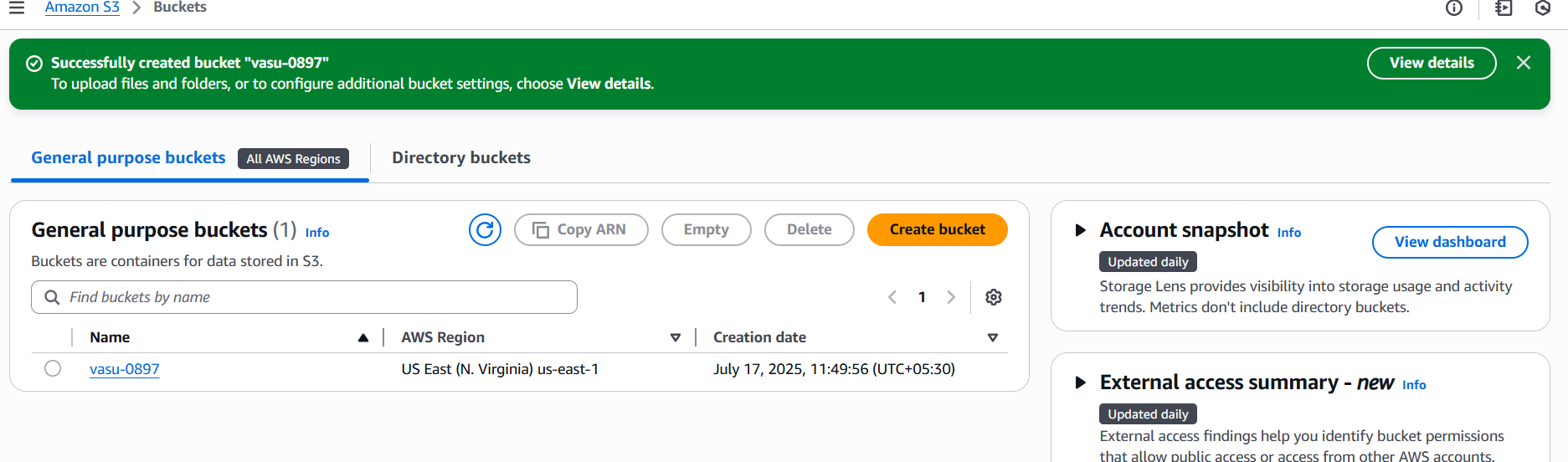
Description automatically generated

🡪Now come back to Manage Jenkins –> Tools (Add the terraform in Tools) 🡪Apply and save.

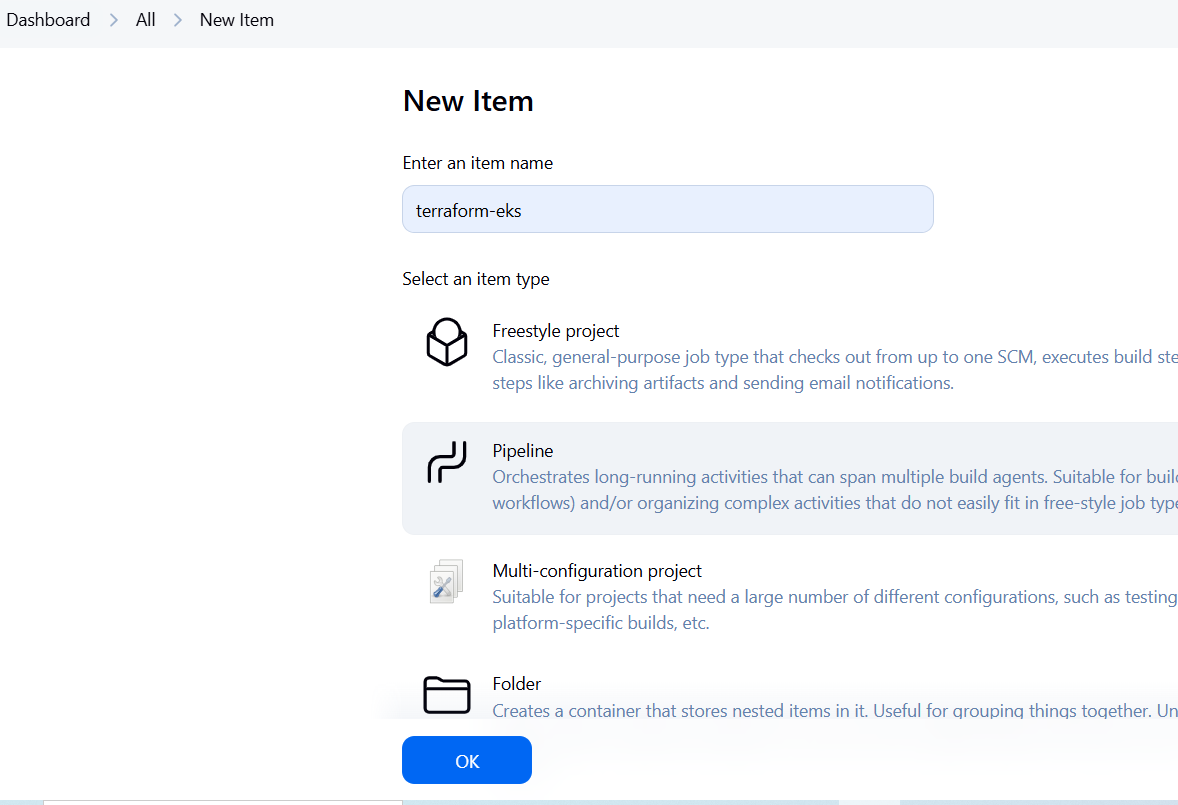
A screenshot of a computer

Description automatically generated

>>CHANGE YOUR S3 BUCKET NAME IN THE backend.tf (create a s3 bucket if you don’t have one, and change the bucket name in **backend.tf** in GitHub🡪 EKS\_ TERRAFORM folder)



>>Now create a new job in Jenkins for the EKS provision



>>I want to do this with build parameters to **apply** and **destroy** while building only. So, we have to add this inside job like the below image(add parameter 🡪 choice parameter)

A screenshot of a computer

Description automatically generated

>> **Let’s add a pipeline**

pipeline{

    agent any

    stages {

        stage('Checkout from Git'){

            steps{

                git branch: 'master', url: 'https://github.com/ kalyanvasu821/uberclone.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'

                   }

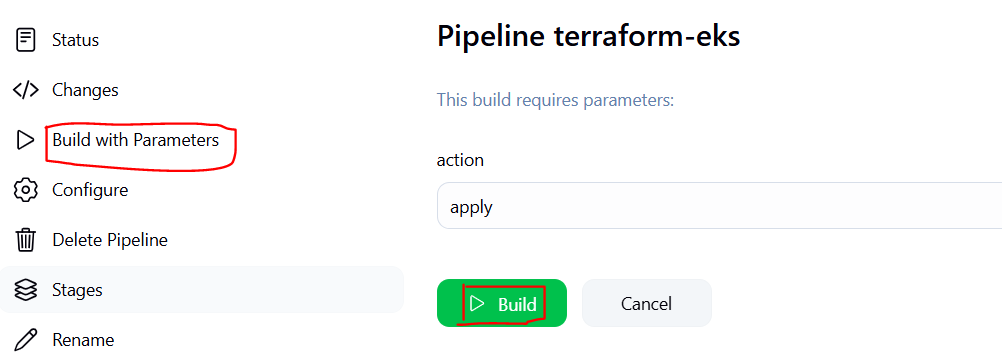
             }

        }

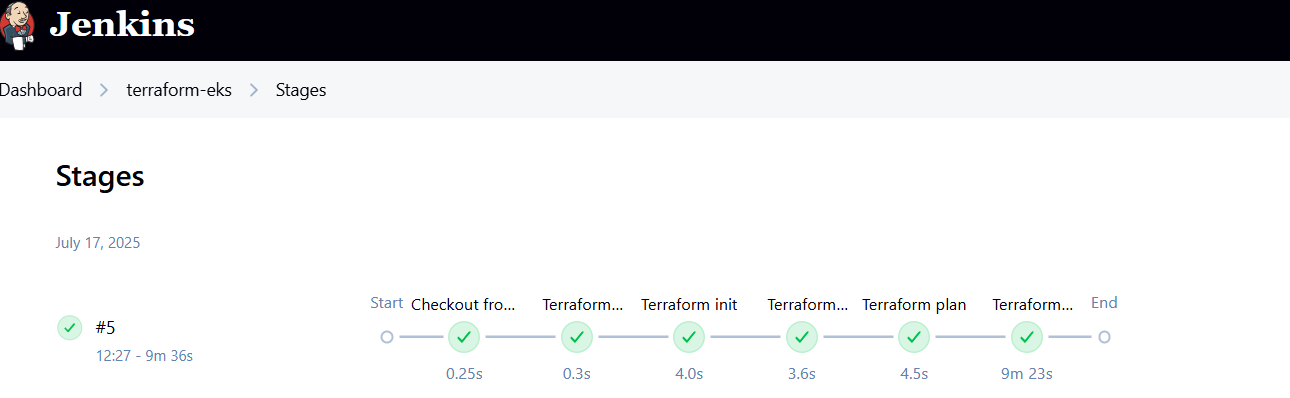
    }

}

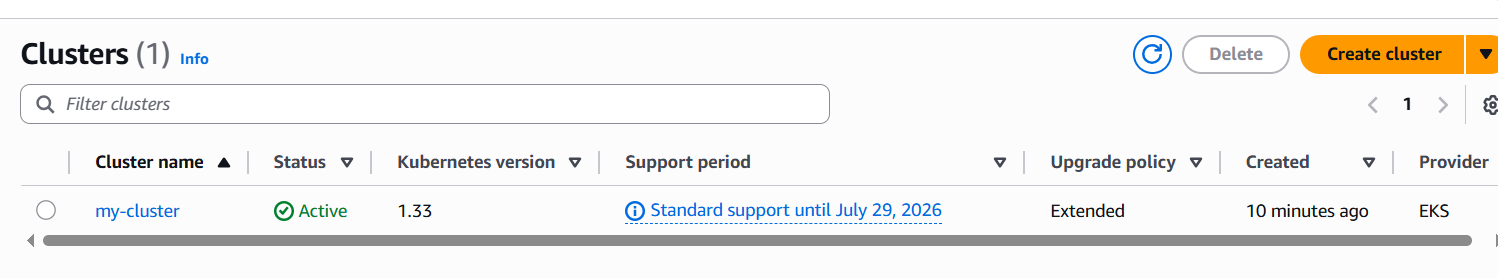
>>Let’s **apply and save** and Build with parameters and select action as apply



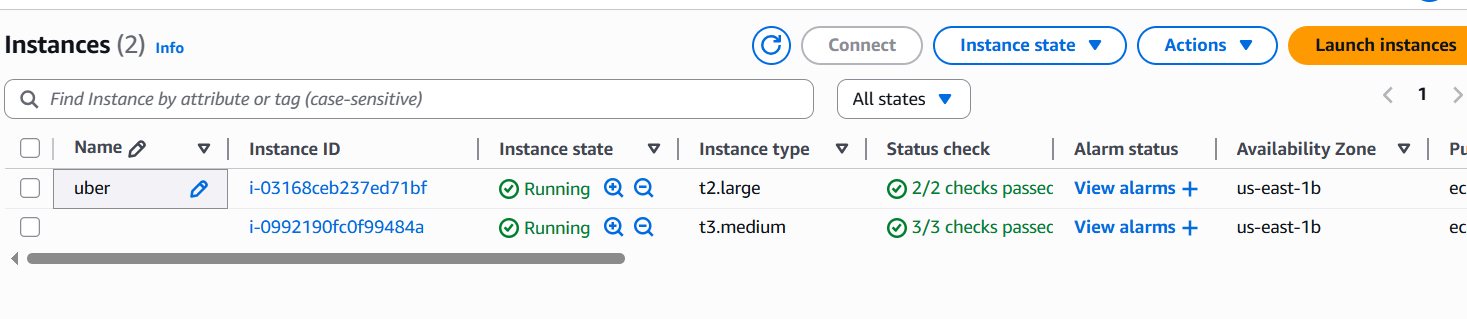
>>Stage view it will take max 10mins to provision



>>Check in Your Aws console whether it created EKS or not.



>>Ec2 instance is created for the Node group



**Step6: Plugins installation & setup (Java, Sonar, Nodejs, owasp, Docker)**

🡪Go to Jenkins dashboard 🡪Manage Jenkins –> Plugins –> Available Plugins

>>Search for the Below Plugins and install them

Eclipse Temurin installer

SonarQube Scanner

NodeJs

Owasp Dependency-Check

Docker

Docker Commons

Docker Pipeline

Docker API

Docker-build-step

A screenshot of a computer

Description automatically generated

A screenshot of a chat

Description automatically generated

**Step7: Configure in Global Tool Configuration**

🡪Goto Manage Jenkins → Tools → Install JDK(17) and NodeJs(16)→ Click on Apply and Save (if have issues with jdk, uncheck install automatically button and add java home path, for that go to connected instace and give command 🡪 **readlink -f $(which java)** and will get something like this /usr/lib/jvm/temurin-17-jdk-amd64/bin/java, remove bin/java from here that is our path)

>>path: usr/lib/jvm/temurin-17-jdk-amd64

A screenshot of a phone

Description automatically generated

A screenshot of a computer

Description automatically generated

>>For Sonarqube use the latest version

A screenshot of a computer

Description automatically generated

>>For Owasp use the 6.5.1 version

A screenshot of a computer

Description automatically generated

>>Use the latest version of Docker

A screenshot of a computer

Description automatically generated

>>Click on apply and save.

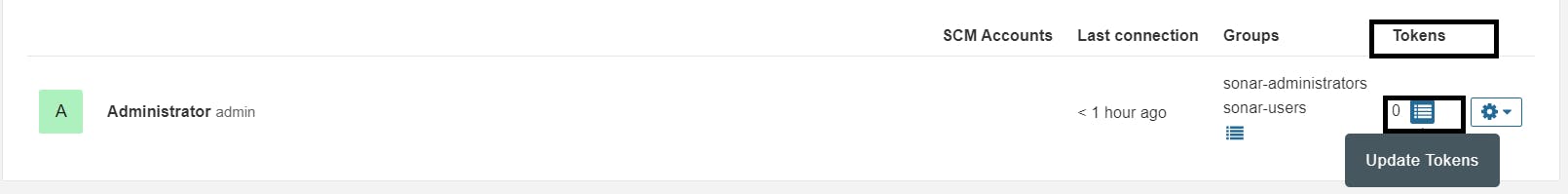
**Step8: Configure Sonar Server in Manage Jenkins**

🡪Goto your Sonarqube Server. Click on Administration → Security → Users → Click on Tokens and Update Token → Give it a name → and click on Generate Token

A screenshot of a computer

Description automatically generated

>>click on update Token



>>Create a token with a name and generate

A screenshot of a computer

Description automatically generated

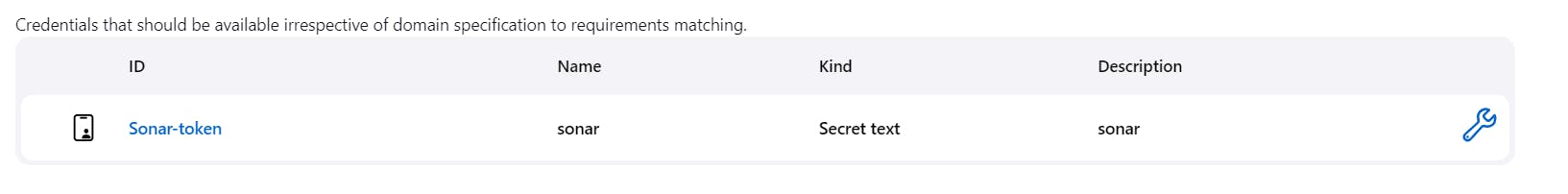
>>now copy Token

🡪Goto Jenkins Dashboard → Manage Jenkins → Credentials → global 🡪Add credential 🡪Add Secret Text. It should look like this (paste copied token in the secret field)

A screenshot of a computer

Description automatically generated

>>You will this page once you click on create



>>Now, go to Dashboard → Manage Jenkins → System and Add like the below image. (Grab the Public IP Address of your EC2 Instance, Sonarqube works on Port 9000, so <Public IP>:9000.)

A screenshot of a computer

Description automatically generated

>>Click on Apply and Save

>>In the Sonarqube Dashboard add a quality gate also

🡪Administration–> Configuration–>Webhooks

A screenshot of a computer

Description automatically generated

🡪Click on Create

A screenshot of a computer

Description automatically generated

Add details

>>in url section of quality gate, add like below

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

A screenshot of a computer

Description automatically generated

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

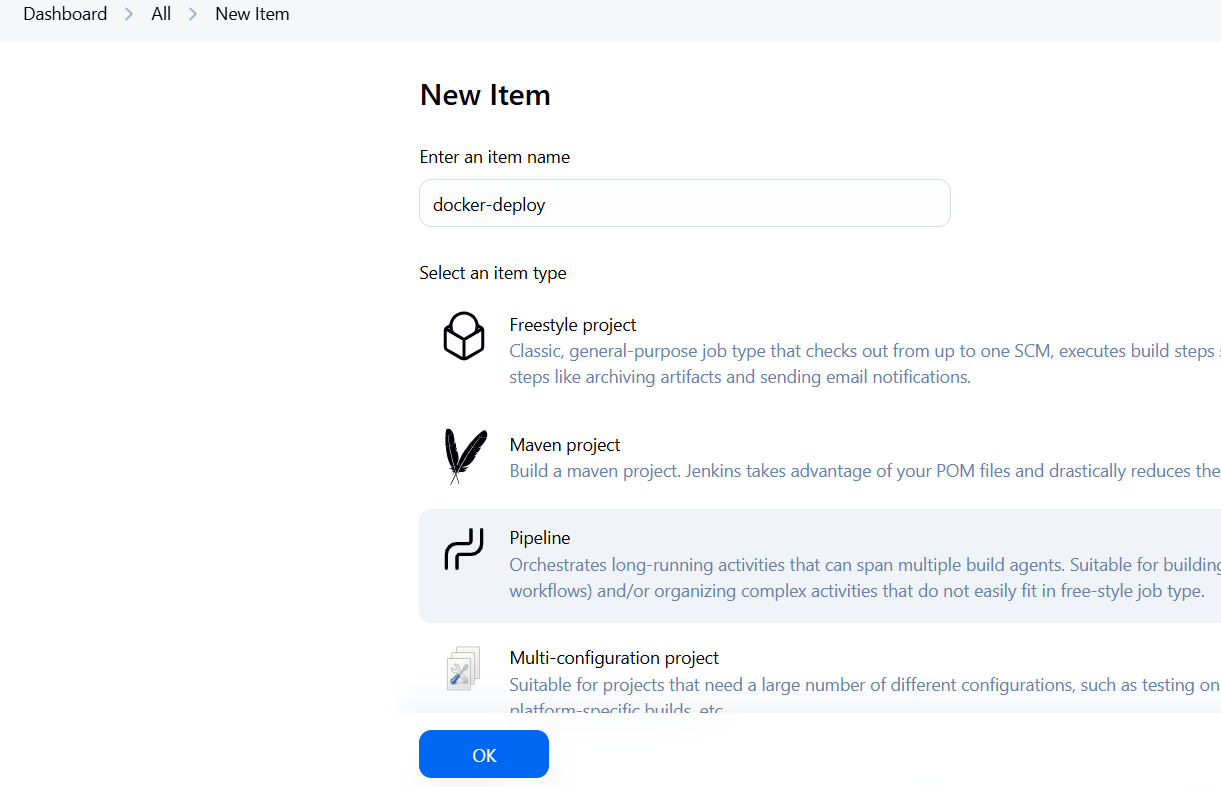
🡪Manage Jenkins –> Credentials –> global –> add credential (Add DockerHub Username and Password under Global Credentials) 🡪 Create

A screenshot of a computer

Description automatically generated

**Step09: Pipeline upto Docker**

Now let’s create a new job for our pipeline



>>Add below script to Pipeline

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: 'master', url: 'https://github.com/ kalyanvasu821/uberclone.git'

            }

        }

        stage("Sonarqube Analysis "){

            steps{

                withSonarQubeEnv('sonar-server') {

                    sh ''' $SCANNER\_HOME/bin/sonar-scanner -Dsonar.projectName=Uber \

                    -Dsonar.projectKey=Uber'''

                }

            }

        }

        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 uber ."

                       sh "docker tag uber kalyanvasu08/uber:latest "

                       sh "docker push kalyanvasu08/uber:latest "

                    }

                }

            }

        }

        stage("TRIVY"){

            steps{

                sh "trivy image kalyanvasu08/uber:latest > trivyimage.txt"

            }

        }

        stage("deploy\_docker"){

            steps{

                sh "docker run -d --name uber -p 3000:3000 kalyanvasu08/uber:latest"

            }

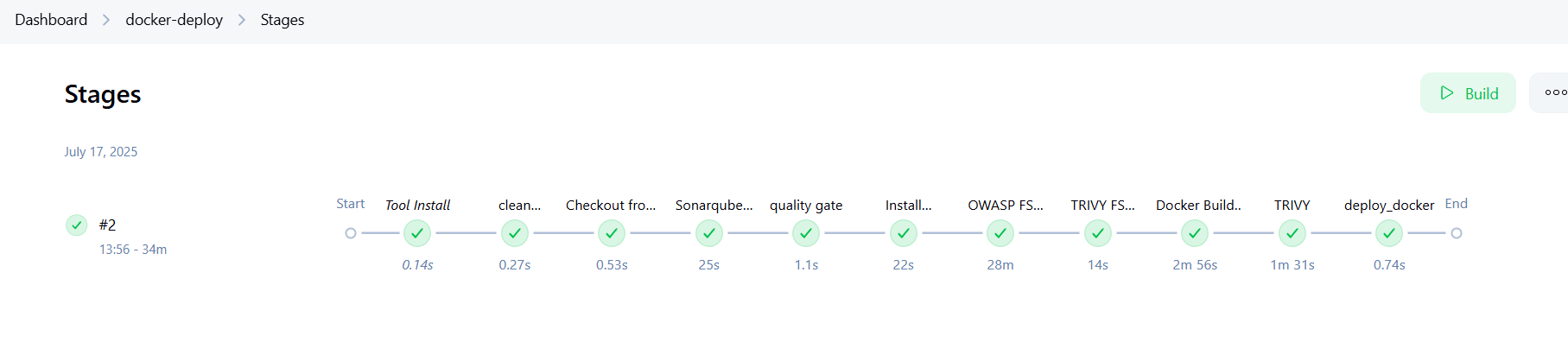
        }

    }

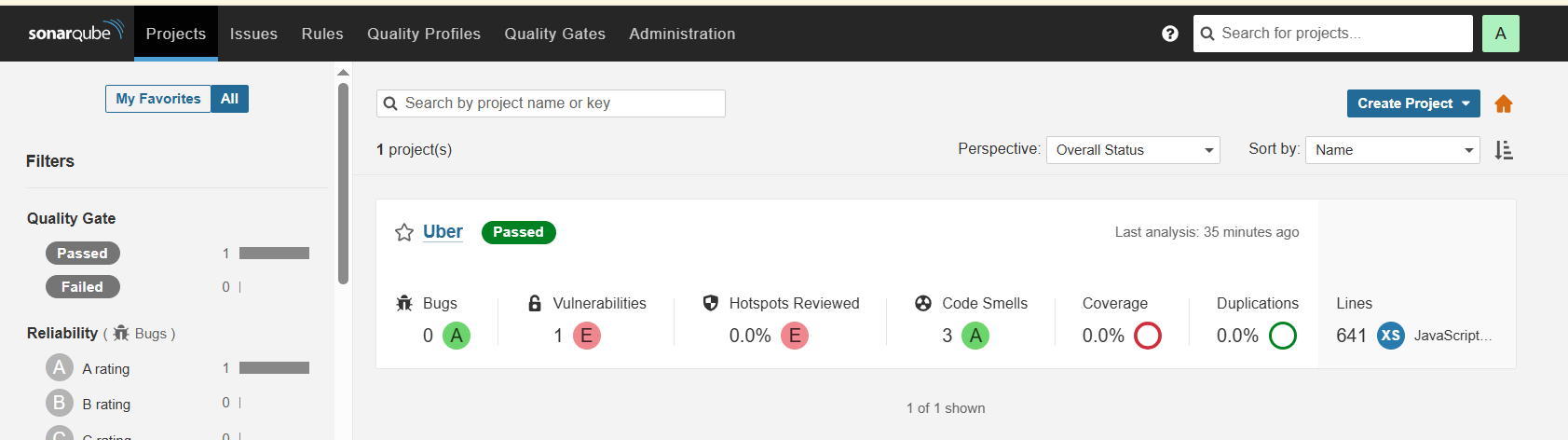
}

🡪Click on Apply and save 🡪Build now

**>>Stage view**



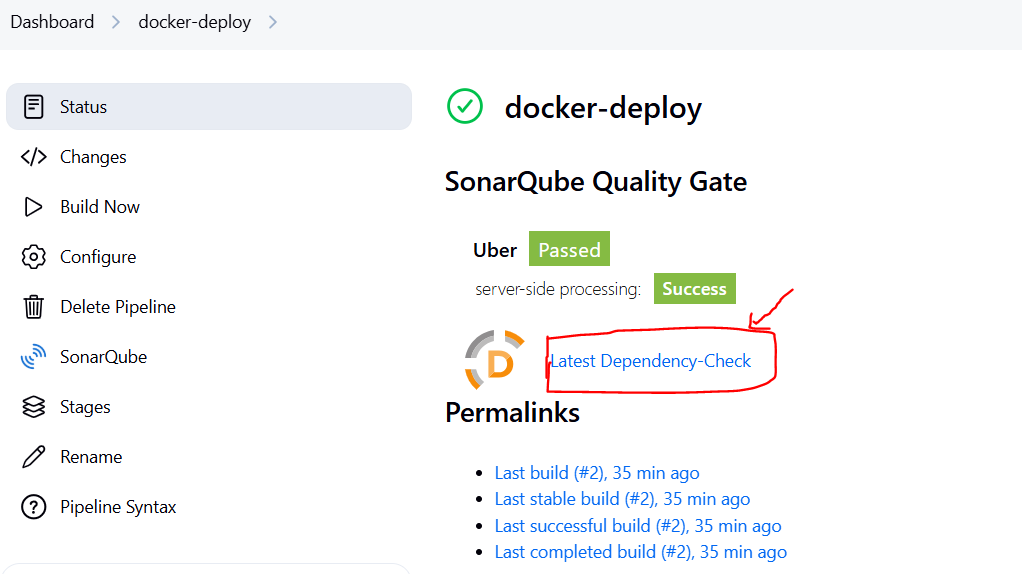
>>To see the report, you can go to Sonarqube Server and go to Projects.

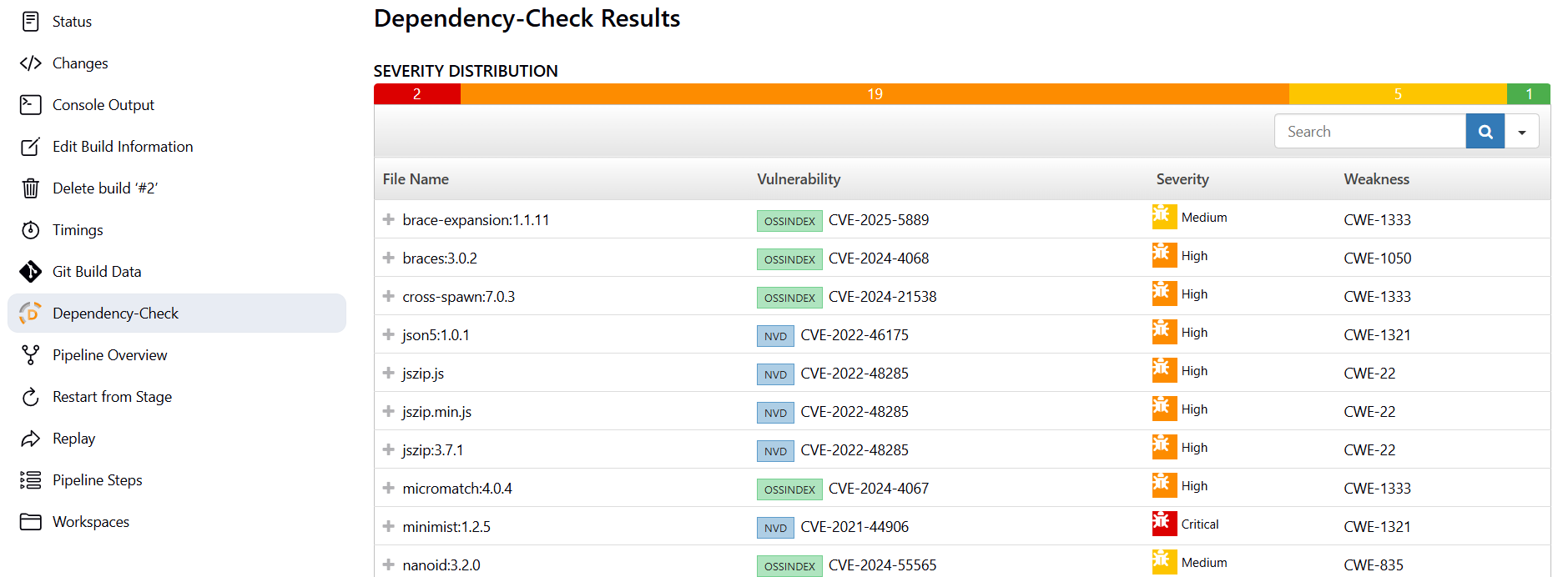


>>You can see the report has been generated and the status shows as passed. You can see that there are k lines it scanned. To see a detailed report, you can go to issues.

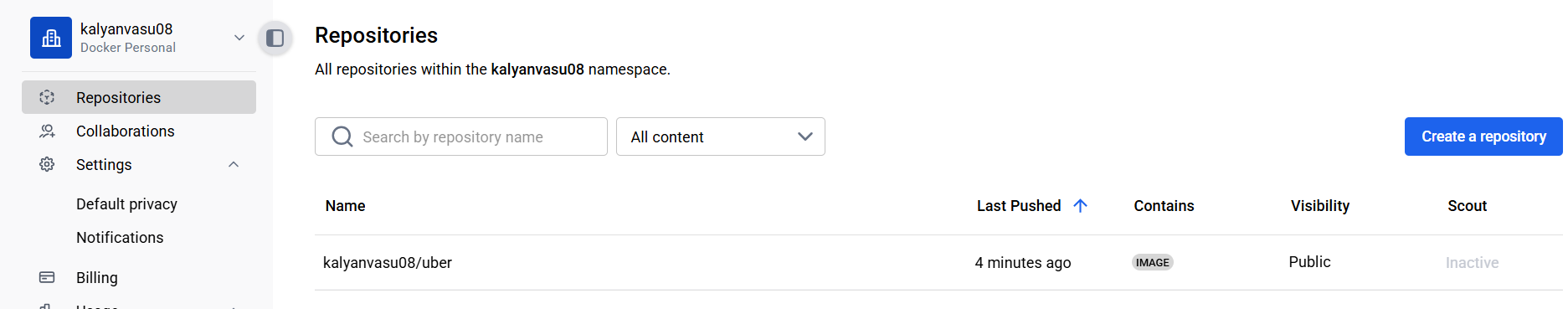
>>OWASP, you will see that in status, a graph will also be generated and Vulnerabilities.

>>Click on Latest Dependency Check



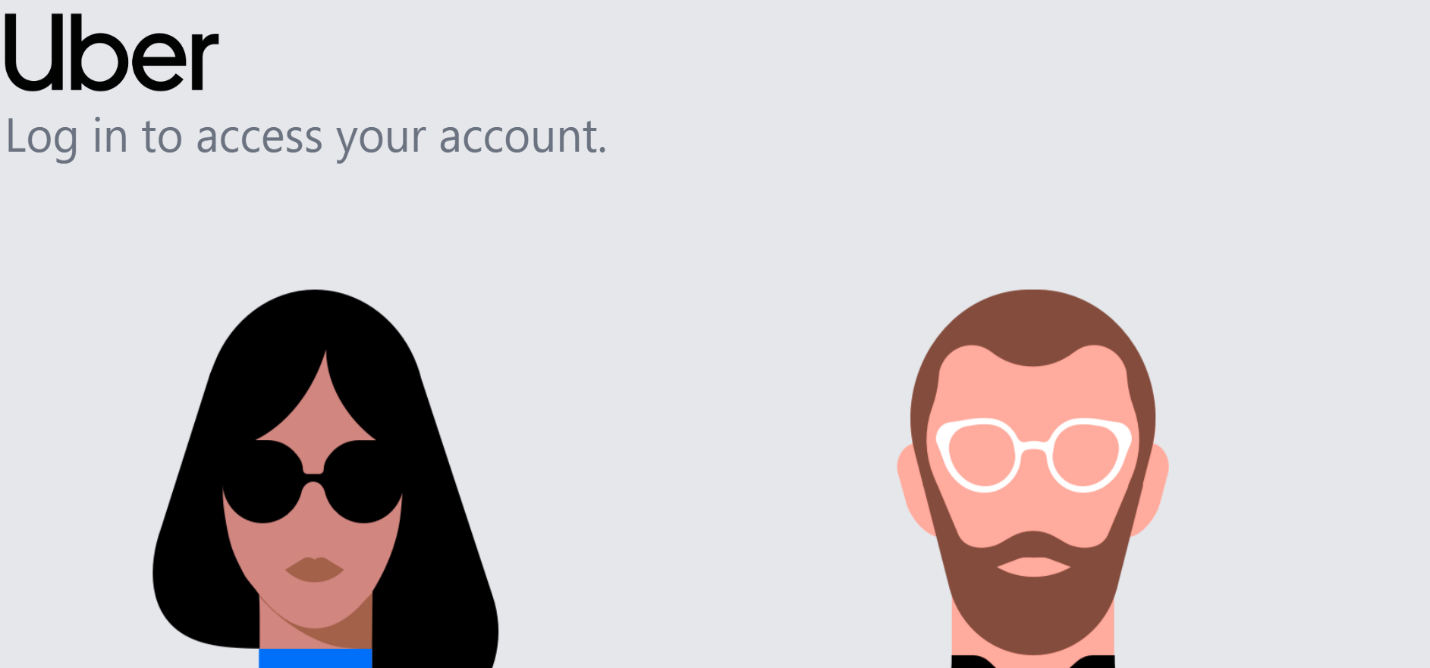


>>When you log in to Dockerhub, you will see a new image is created



>>Now we can check app by taking public IP of instance with port 3000   
<public-IP:300>

**>>output looks like below**

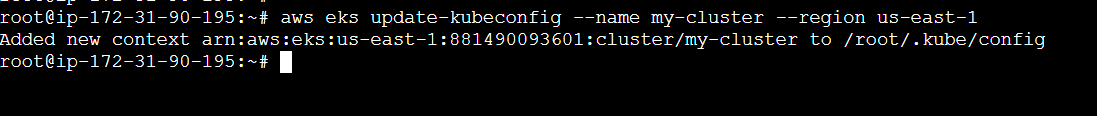


**Step10: Kubernetes Deployment**

>>Go to Putty/Moba/GitBash of your Jenkins instance SSH and enter the below command

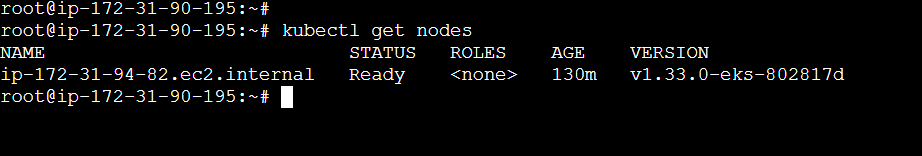
🡪aws eks update-kubeconfig --name <CLUSTER NAME> --region <CLUSTER REGION>

aws eks update-kubeconfig --name my-cluster --region us-east-1



>>Let’s see the nodes

🡪kubectl get nodes



>>Copy the config file to Jenkins master or the local file manager and save it

A computer screen with text on it

Description automatically generated

>> cat ~/.kube/config to view config (copy the content and save in text file. Will use it as credentials in jenkins)

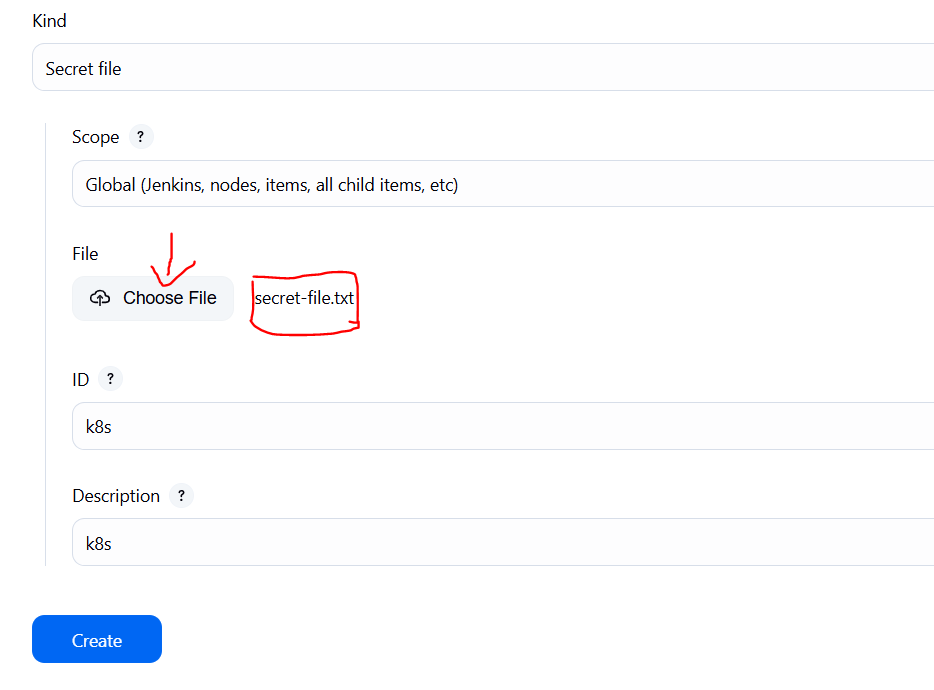
\*Note: create a secret-file.txt in your file explorer save the config in it and use this at the kubernetes credential section.

>>Install Kubernetes Plugin, once it’s installed successfully

A screenshot of a computer

Description automatically generated

🡪go to manage Jenkins –> credentials –> global –> add credentials 🡪add below details and then Create



**>>final step to deploy on the Kubernetes cluster (**add below stage in our existing docker-deploy pipeline**)**

stage('Deploy to kubernets'){

            steps{

                script{

                    dir('K8S') {

                        withKubeConfig(caCertificate: '', clusterName: '', contextName: '', credentialsId: 'k8s', namespace: '', restrictKubeConfigAccess: false, serverUrl: '') {

                                sh 'kubectl apply -f deployment.yml'

                                sh 'kubectl apply -f service.yml'

                        }

                    }

                }

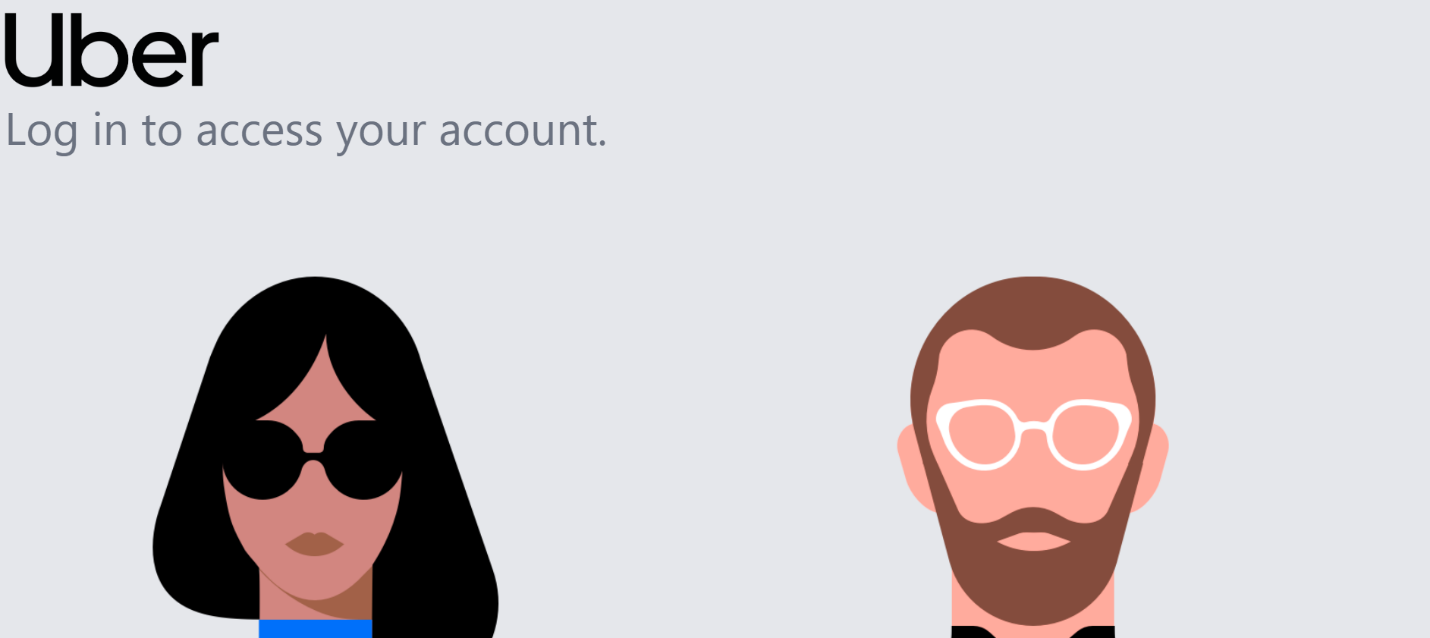
            }

        }

>>to check app use below command to get External IP

kubectl get service uber-service

**>>**copy the external IP and paste in browser to see the output



**Step11: Destruction**

>>Now Go to Jenkins Dashboard and click on Terraform-Eks job and build with parameters and destroy action. It will delete the EKS cluster that provisioned

A screenshot of a computer

Description automatically generated

>>After 10 minutes cluster will delete and wait for it. Don’t remove ec2 instance till that time.

>>Cluster deleted

A screenshot of a computer

Description automatically generated

>>Finally delete the Ec2 instance.

**\*\*NOTE\*\*:** Few observations during implementation

>>Find Real Path to Java Binary (when issue with jdk installation in Jenkins)

🡪readlink -f $(which java) ---> /usr/lib/jvm/java-17-temurin/bin/java

From here, remove /bin/java to get the JAVA\_HOME:

--> /usr/lib/jvm/java-17-temurin

🡪cat ~/.kube/config to view config (save in text file and give it as credentials in jenkins)

**\*\*edit trust policy (**when using same arn for both cluster and node group**)**

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Principal": {

"Service": [

"ec2.amazonaws.com",

"eks.amazonaws.com"

]

},

"Action": "sts:AssumeRole"

}

]

}

🡪kubectl get service uber-service (expose external IP)