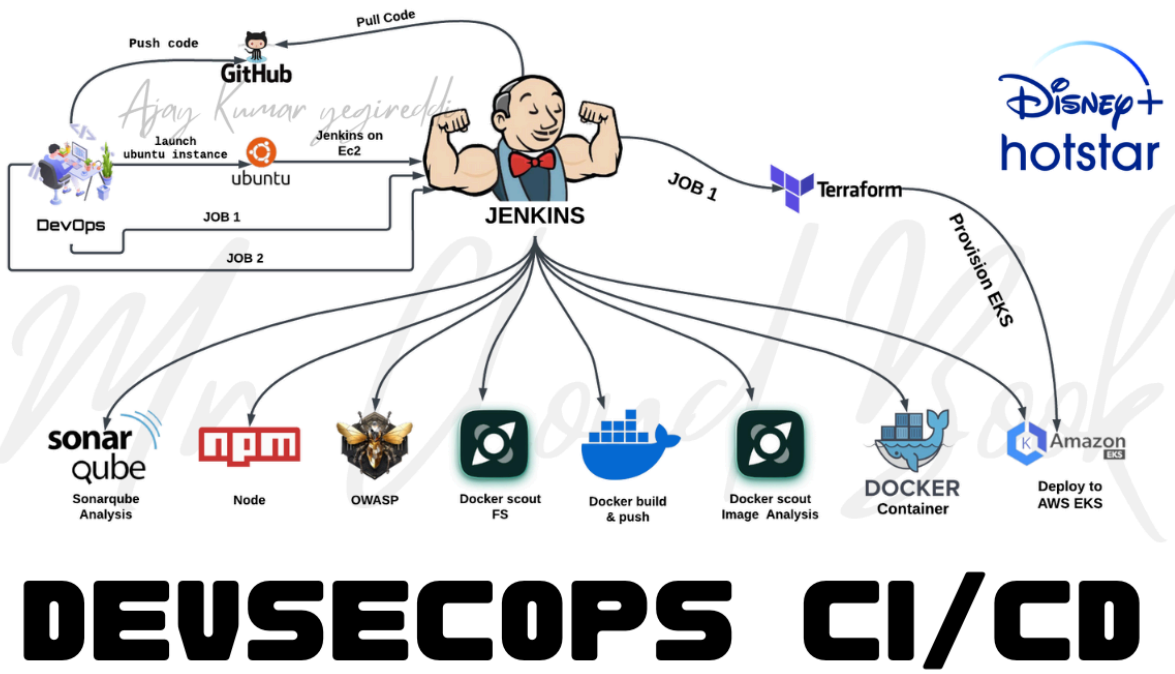
Uncategorized

DevSecOps CI/CD : Deploying a Secure Hotstar Clone (Even if You're Not a Pro)

mrcloudbook.com · 6 January 2024



In the ever-evolving landscape of software development and deployment, the integration of robust security practices into the development pipeline has become imperative. DevSecOps, an amalgamation of Development, Security, and Operations, emphasizes the integration of security measures throughout the software development lifecycle, promoting a proactive approach to mitigate potential vulnerabilities and threats.

This blog serves as a comprehensive guide to deploying a Hotstar clone—a popular streaming platform—using the principles of DevSecOps on Amazon Web Services (AWS). The deployment process encompasses the utilization of various tools and services, including Docker, Jenkins, Java, SonarQube, AWS CLI, Kubectl, and Terraform, to automate, secure, and streamline the deployment pipeline.

The journey begins with the setup of an AWS EC2 instance configured with Ubuntu, granting it a specific IAM role to facilitate the learning process. Subsequently, an automated script is crafted to install crucial tools and dependencies required for the deployment pipeline, ensuring efficiency and consistency in the setup process.

Central to this DevSecOps approach is the orchestration through Jenkins jobs, where stages are defined to execute tasks such as creating an Amazon EKS cluster, deploying the Hotstar clone application, and implementing security measures at various stages of the deployment.

The integration of security practices is a pivotal aspect of this process. Utilizing SonarQube, OWASP, and Docker Scout, the blog will elucidate how static code analysis, security checks, and container security scans are seamlessly embedded within the pipeline. These measures fortify the application against potential vulnerabilities, ensuring a robust and secure deployment.

GITHUB : https://github.com/Aj7Ay/Hotstar-Clone.git

Contents [hide]

Prerequisites

- AWS account setup
- Basic knowledge of AWS services
- Understanding of DevSecOps principles
- Familiarity with Docker, Jenkins, Java, SonarQube, AWS CLI, Kubectl, and Terraform,Docker Scout

Step-by-Step Deployment Process

Step 1: Setting up AWS EC2 Instance

- Creating an EC2 instance with Ubuntu AMI, t2.large, and 30 GB storage
- Assigning an IAM role with Admin access for learning purposes

Step 2: Installation of Required Tools on the Instance

- Writing a script to automate the installation of:
  - Docker
  - Jenkins
  - Java
  - SonarQube container
  - AWS CLI
  - Kubectl
  - Terraform

Step 3: Jenkins Job Configuration

- Creating Jenkins jobs for:
  - Creating an EKS cluster
  - Deploying the Hotstar clone application

- Configuring the Jenkins job stages:
  - Sending files to SonarQube for static code analysis
  - Running `npm install`
  - Implementing OWASP for security checks
  - Installing and running Docker Scout for container security
  - Scanning files and Docker images with Docker Scout
  - Building and pushing Docker images
  - Deploying the application to the EKS cluster

Step 4: Clean-Up Process

- Removing the EKS cluster
- Deleting the IAM role
- Terminating the Ubuntu instance

STEP 1A: Setting up AWS EC2 Instance and IAM Role

1. **Sign in to the AWS Management Console:** Access the AWS Management Console using your credentials
2. **Navigate to the EC2 Dashboard:** Click on the “Services” menu at the top of the page and select “EC2” under the “Compute” section. This will take you to the EC2 Dashboard.
3. **Launch Instance:** Click on the “Instances” link on the left sidebar and then click the “Launch Instance” button.
4. **Choose an Amazon Machine Image (AMI):** In the “Step 1: Choose an Amazon Machine Image (AMI)” section:
   - Select “AWS Marketplace” from the left-hand sidebar.
   - Search for “Ubuntu” in the search bar and choose the desired Ubuntu AMI (e.g., Ubuntu Server 22.04 LTS).
   - Click on “Select” to proceed.

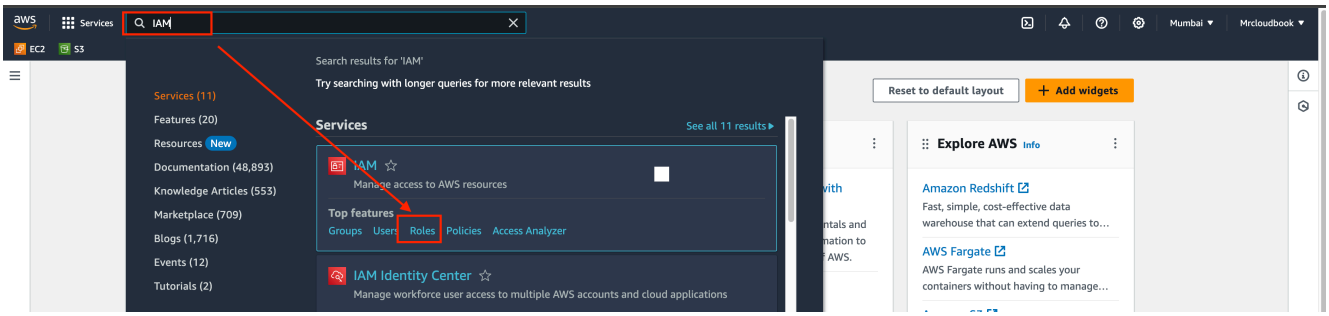
5. **Choose an Instance Type:** In the “Step 2: Choose an Instance Type” section:
    - Scroll through the instance types and select “t2.large” from the list.
    - Click on “Next: Configure Instance Details” at the bottom.
6. **Configure Instance Details:** In the “Step 3: Configure Instance Details” section, you can leave most settings as default for now. However, you can configure settings like the network, subnet, IAM role, etc., according to your requirements.
    - Once done, click on “Next: Add Storage.”
7. **Add Storage:** In the “Step 4: Add Storage” section:
    - You can set the size of the root volume (usually /dev/sda1) to 30 GB by specifying the desired size in the “Size (GiB)” field.
    - Customize other storage settings if needed.
    - Click on “Next: Add Tags” when finished.
8. **Add Tags (Optional):** In the “Step 5: Add Tags” section, you can add tags to your instance for better identification and management. This step is optional but recommended for organizational purposes.
    - Click on “Next: Configure Security Group” when done.
9. **Configure Security Group:** In the “Step 6: Configure Security Group” section:
    - Create a new security group or select an existing one.
    - Ensure that at least SSH (port 22) is open for inbound traffic to allow remote access.
    - You might also want to open other ports as needed for your application’s requirements.
    - Click on “Review and Launch” when finished.
10. **Review and Launch:** Review the configuration details of your instance. If everything looks good:
    - Click on “Launch” to proceed.
    - A pop-up will prompt you to select or create a key pair. Choose an existing key pair or create a new one.

- Finally, click on “Launch Instances.”

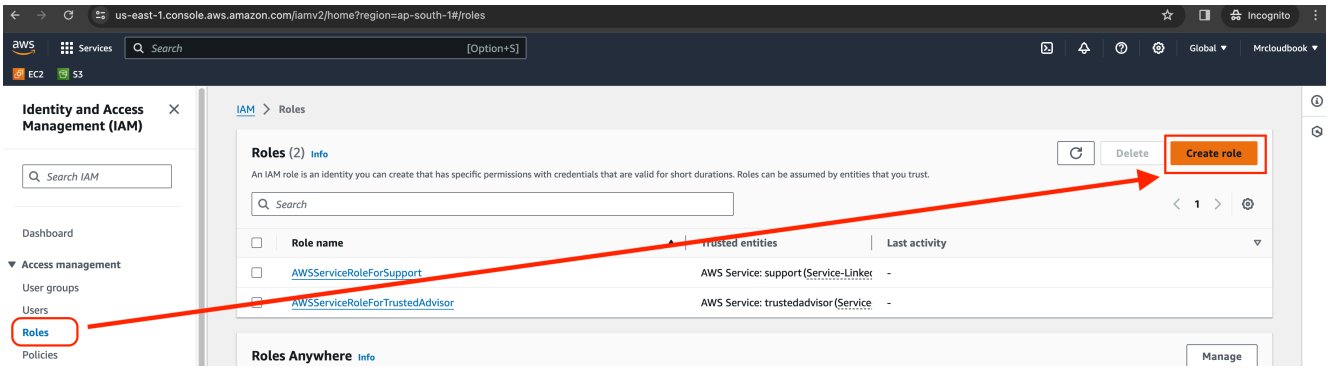
11. **Accessing the Instance:** Once the instance is launched, you can connect to it using SSH. Use the private key associated with the selected key pair to connect to the instance’s public IP or DNS address.

STEP 1B: IAM ROLE

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

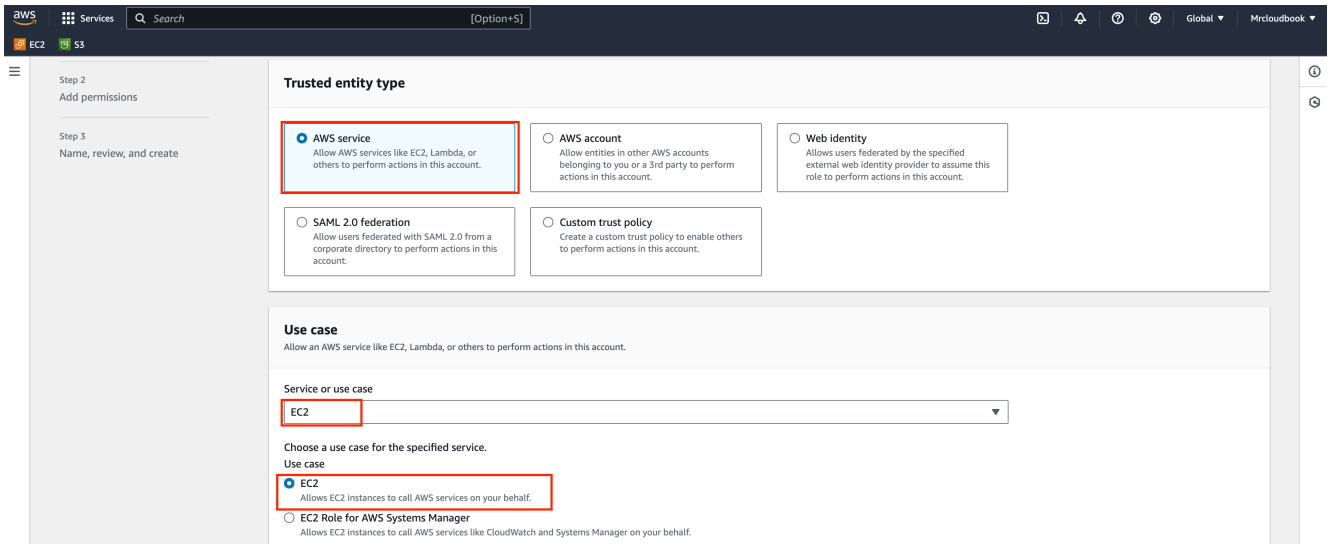


Click on Create Role

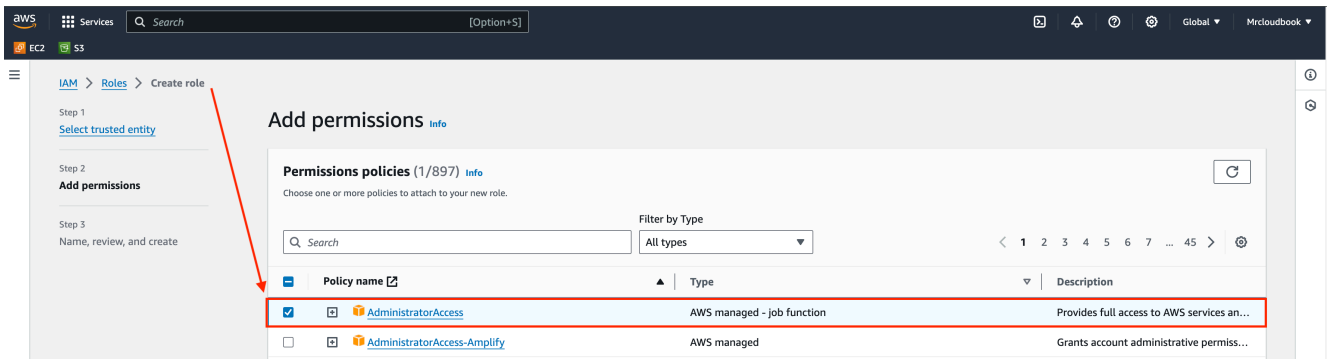


Select entity type as AWS service

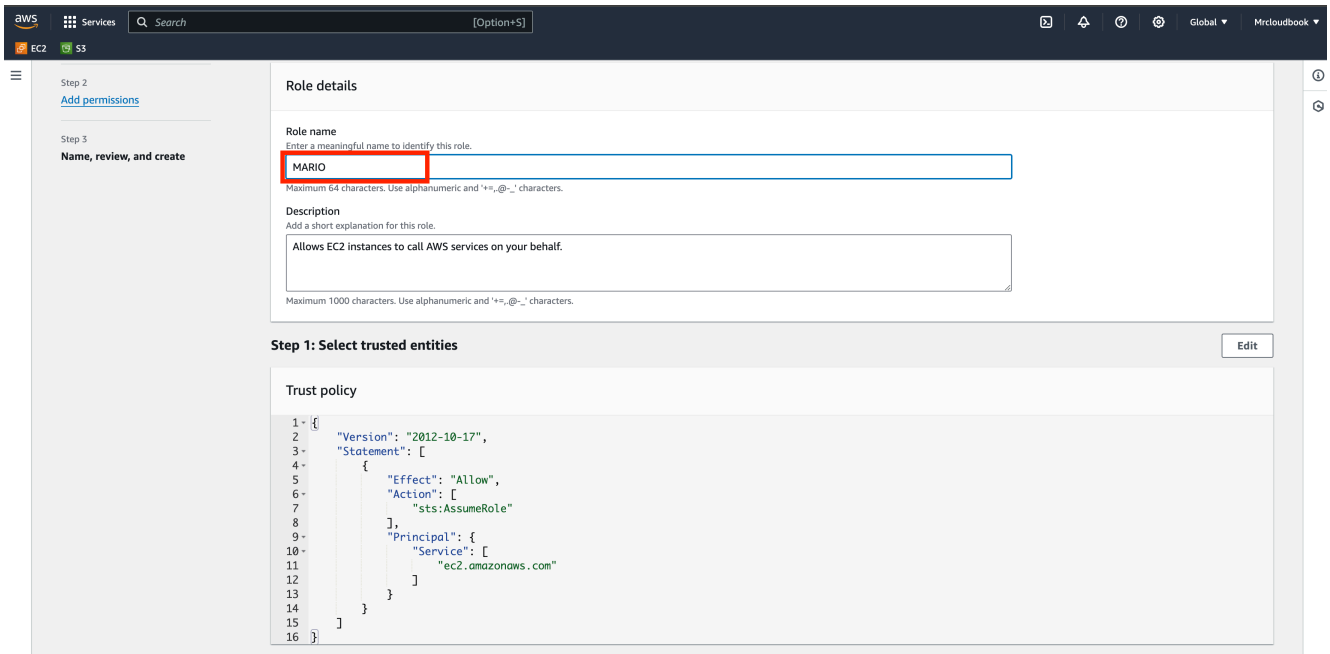
Use case as EC2 and click on Next.



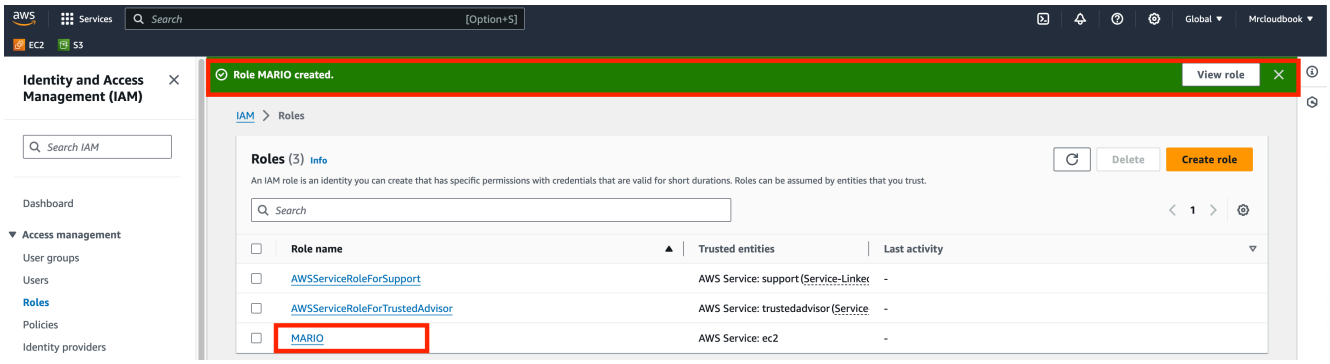
For permission policy select Administrator Access (Just for learning purpose), click Next.



Provide a Name for Role and click on Create role.



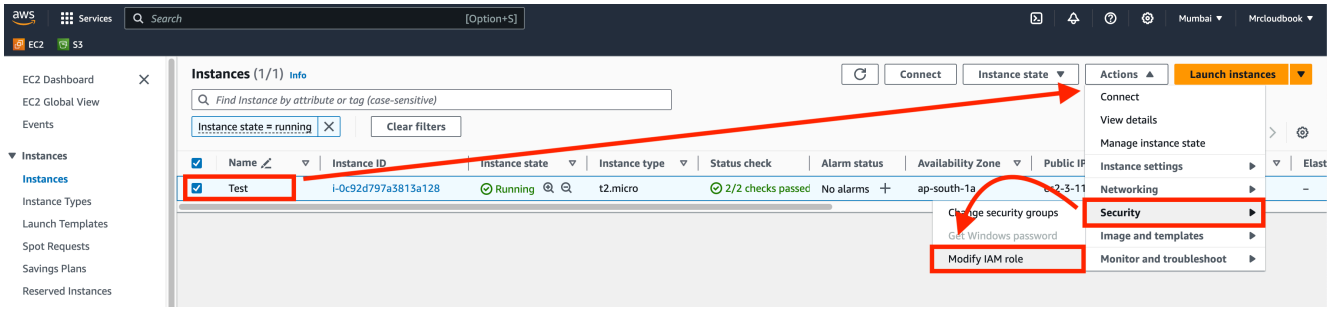
Role is created.



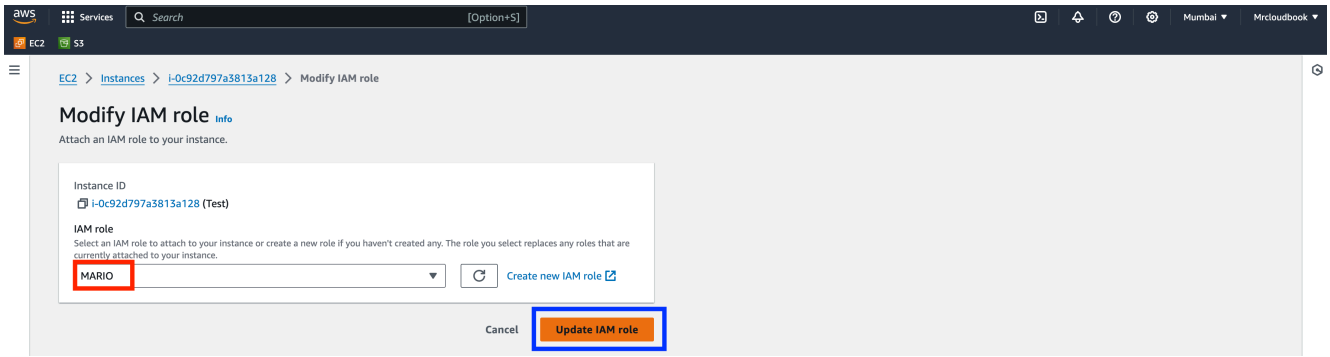
Now Attach this role to Ec2 instance that we created earlier, so we can provision cluster from that instance.

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.

Step 2: Installation of Required Tools on the Instance

Scripts to install Required tools

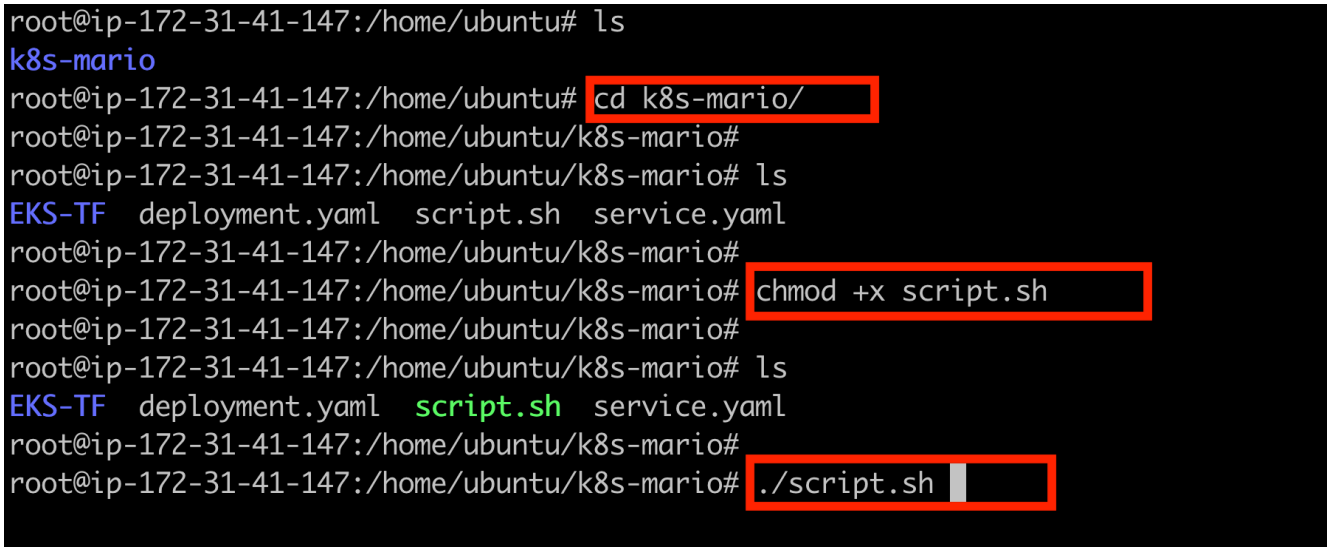
```
sudo su    #Into root
vi script1.sh
```

Script1 for Java,Jenkins,Docker

```
#!/bin/bash
sudo apt update -y
wget -O - https://packages.adoptium.net/artifactory/api/gpg/key/public
echo "deb [signed-by=/etc/apt/keyrings/adoptium.asc] https://packages.ad
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 | su
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg
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 --dea
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/do
  $(. /etc/os-release && echo "$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
sudo usermod -aG docker ubuntu
newgrp docker
```

Run script by providing permissions

```
sudo chmod 777 script1.sh
sh script1.sh
```



Script 2 for Terraform,kubectl,Aws cli

```
vi script2.sh
```

Script 2

```
#!/bin/bash
#install terraform
sudo apt install wget -y
wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg]
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/releas
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 "awsc
sudo apt-get install unzip -y
unzip awscliv2.zip
sudo ./aws/install
```

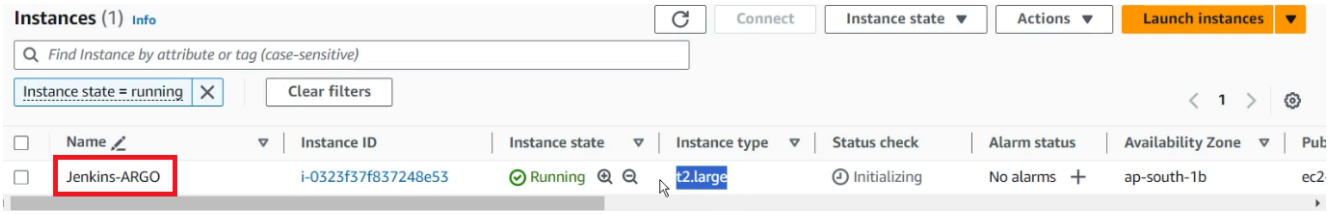
Give permissions

```
sudo chmod 777 script2.sh
sh script2.sh
```

Now Run sonarqube container

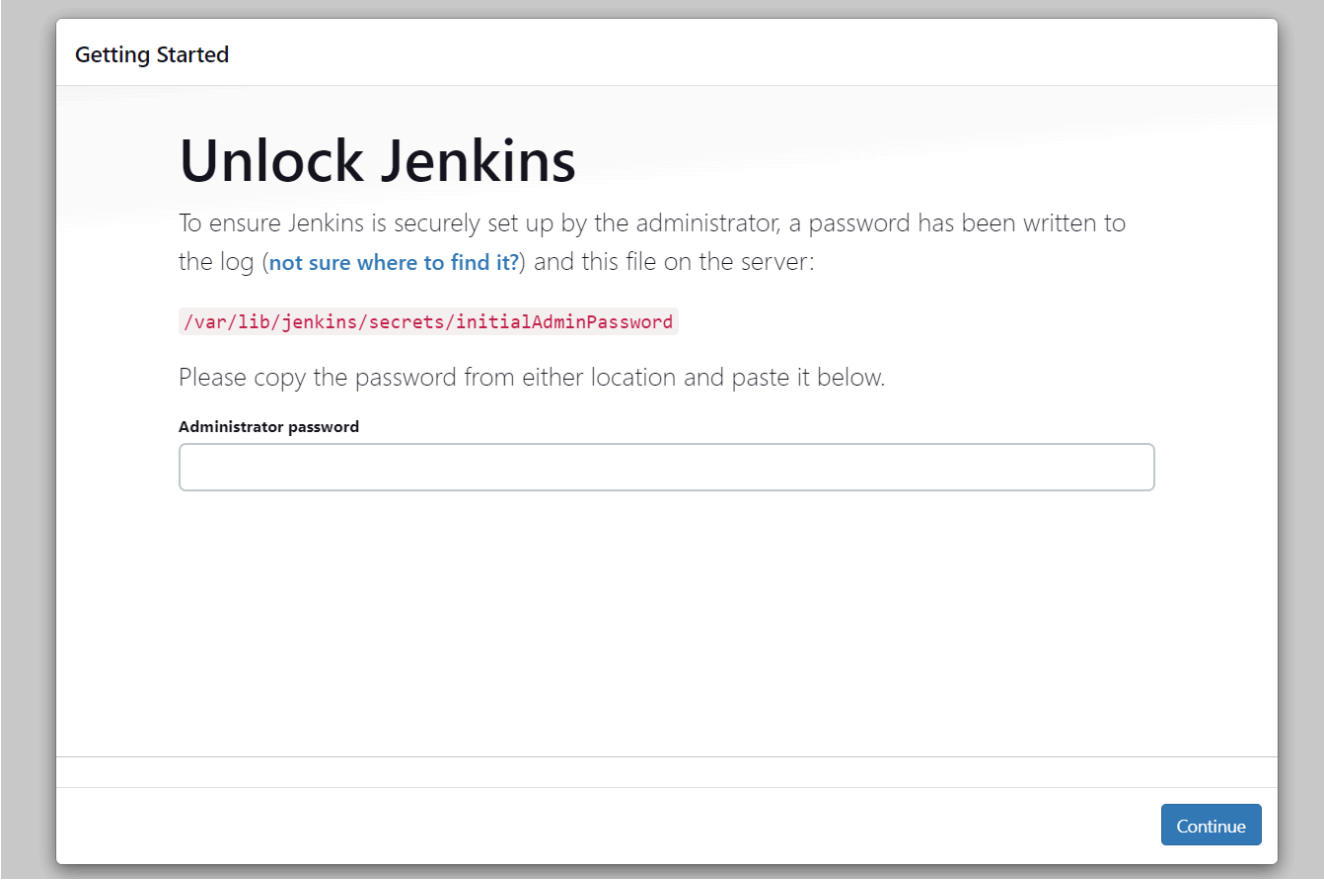
```
sudo chmod 777 /var/run/docker.sock
docker run -d --name sonar -p 9000:9000 sonarqube:lts-community
```

Ec2 is created in the Aws console



Now copy the public IP address of ec2 and paste it into the browser

```
Ec2-ip:8080 #you will Jenkins login page
```

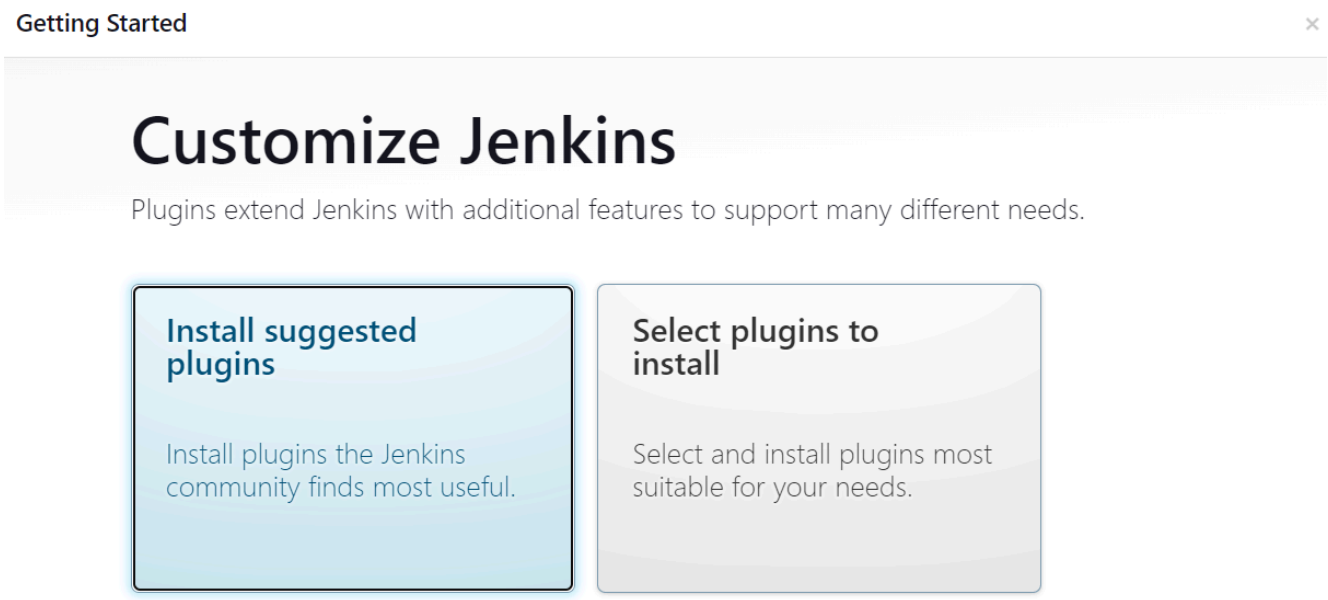


Connect your Instance to Putty or Mobaxtreme and provide the below command for the Administrator password

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

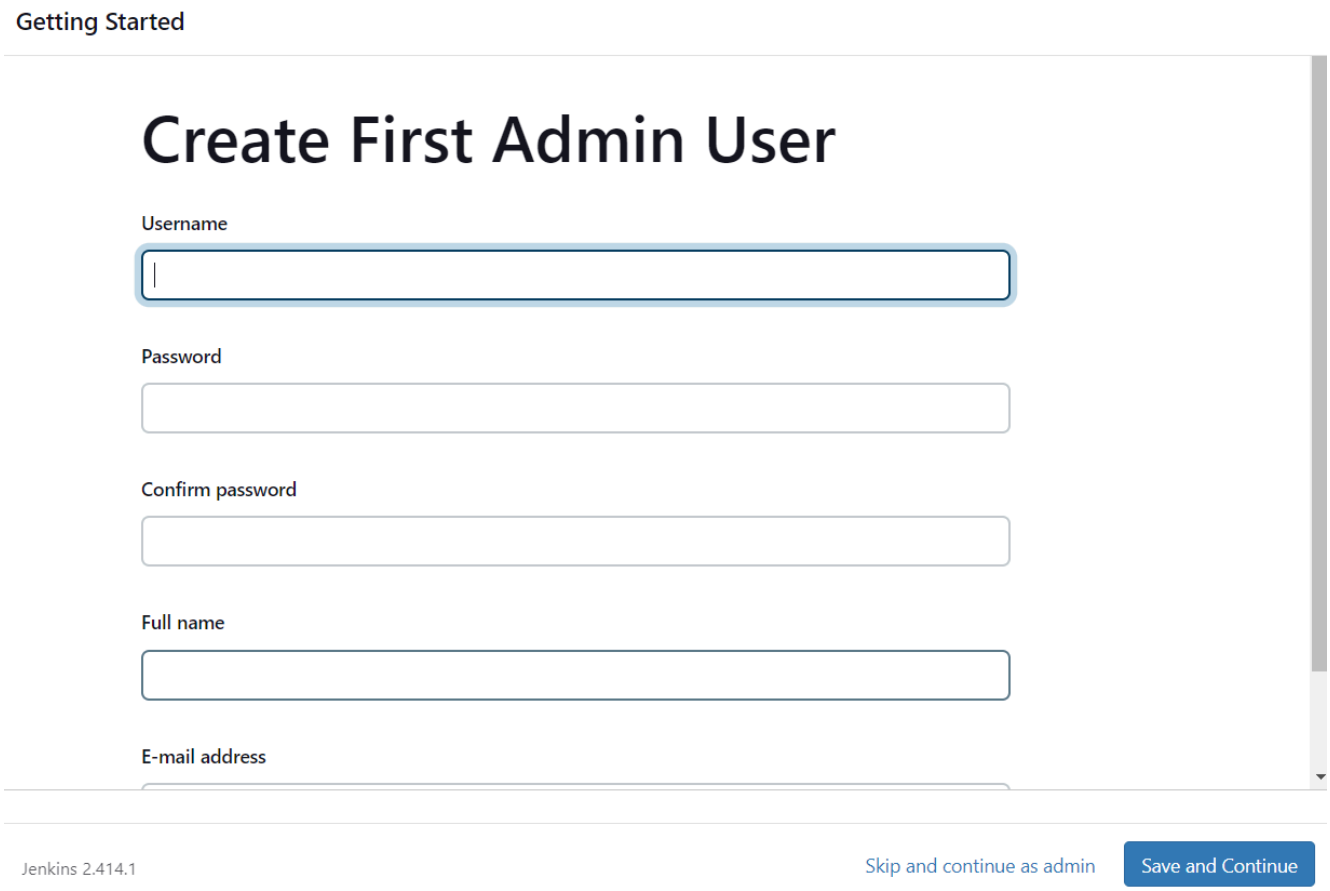
```
ubuntu@ip-172-31-33-57:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
0ed1cb07ea7447c5a47d723022e74968
ubuntu@ip-172-31-33-57:~$
```

Now, install the suggested plugins.



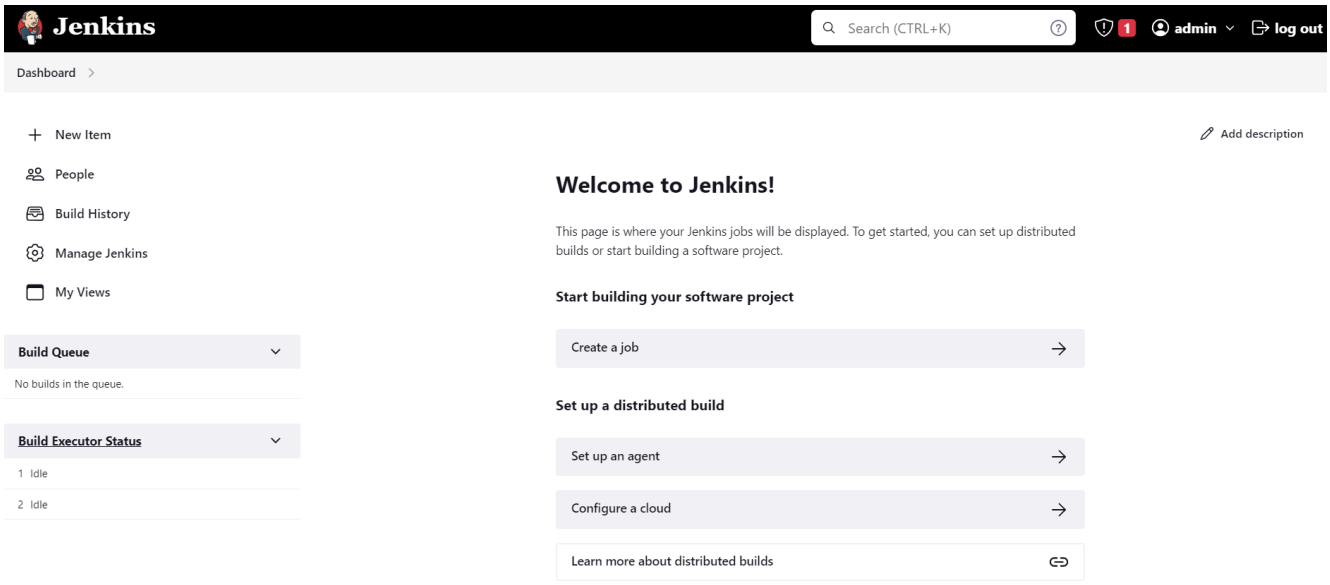
Jenkins will now get installed and install all the libraries.

Create an admin user



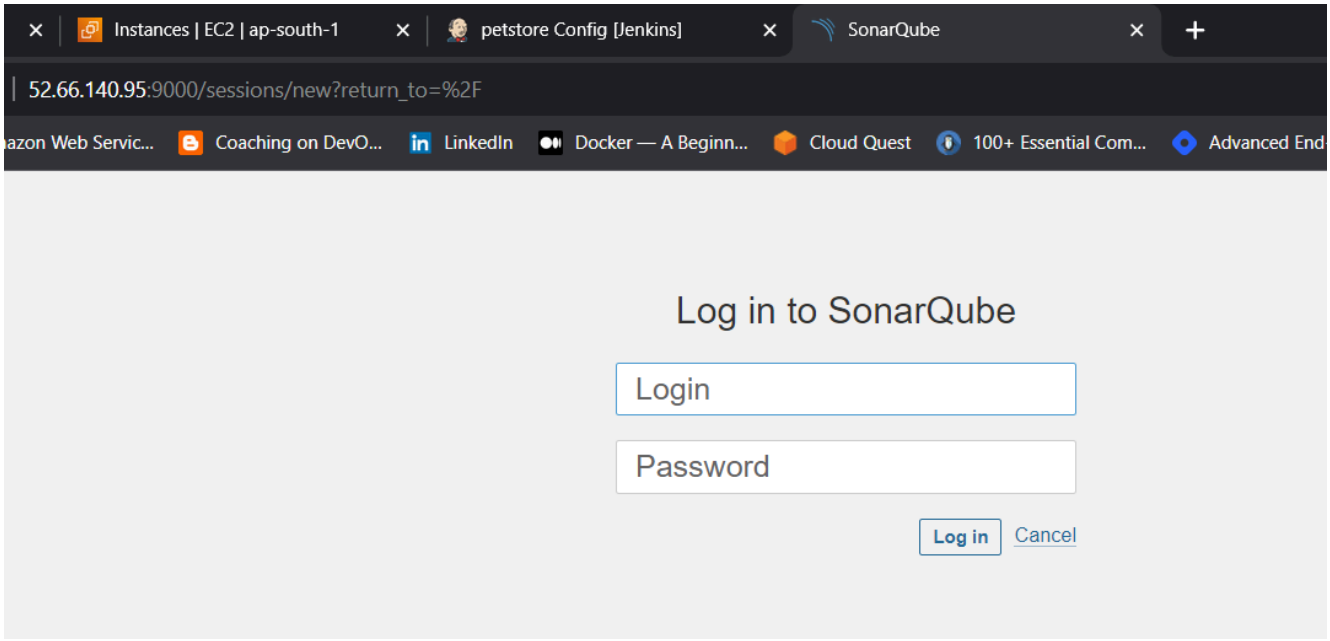
Click on save and continue.

Jenkins Dashboard



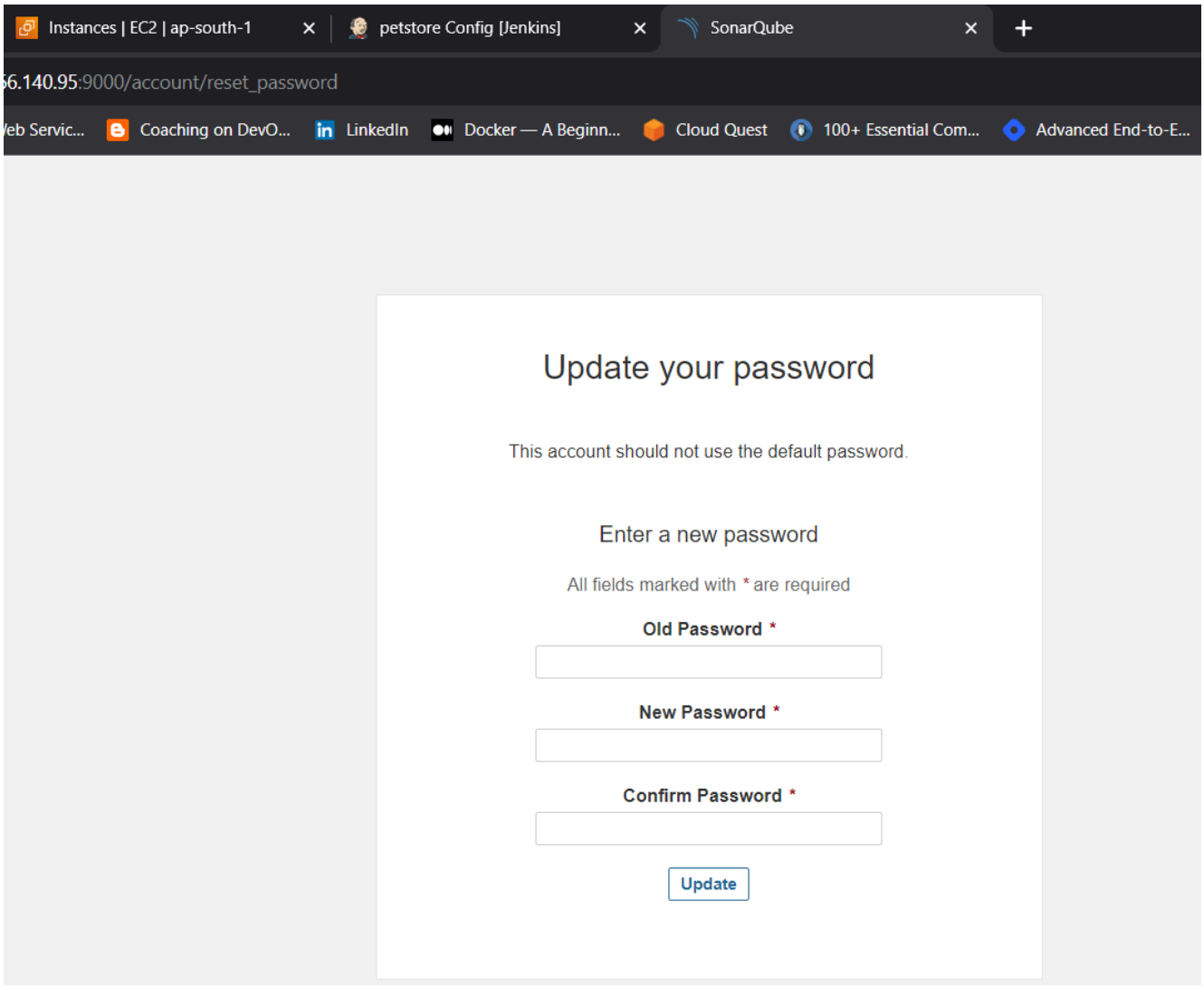
Now Copy the public IP again and paste it into a new tab in the browser with 9000

```
ec2-ip:9000  #runs sonar container
```

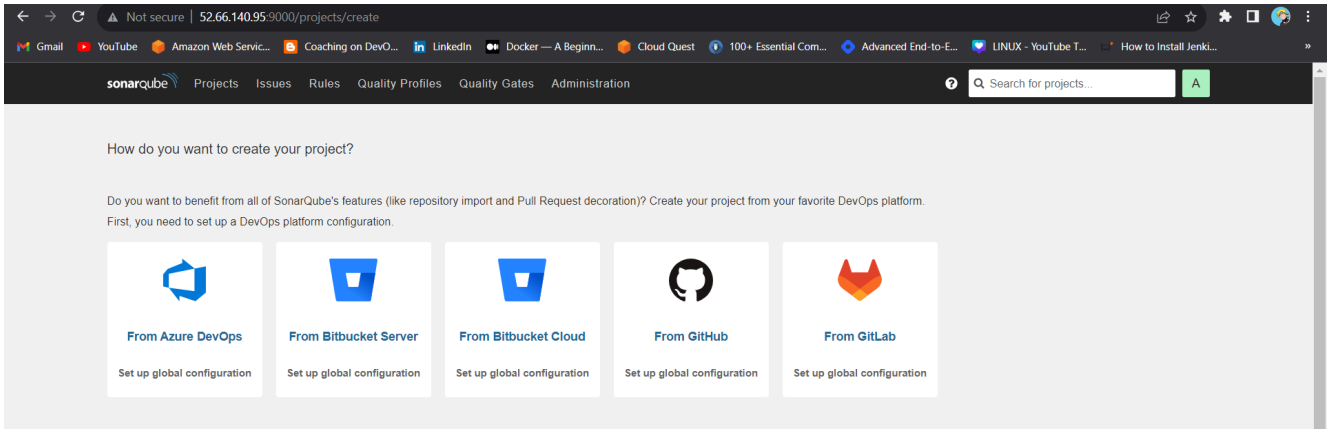


Enter username and password, click on login and change password

```
username admin
password admin
```

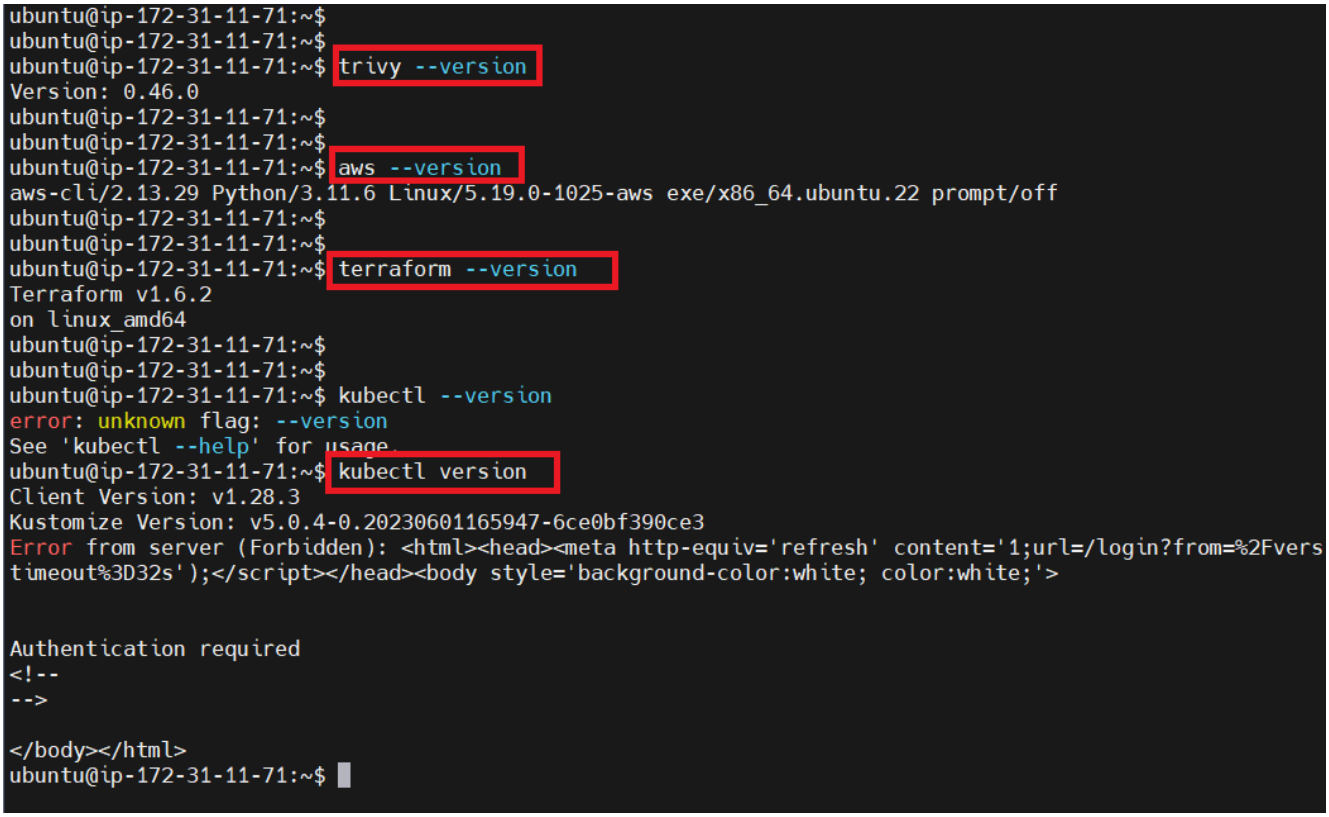


Update New password, This is Sonar Dashboard.



Now go to Putty and see whether it's installed docker, Terraform, Aws cli, Kubectl or not.

```
docker --version
aws --version
terraform --version
kubectl version
```



Step 3: Jenkins Job Configuration

Step 3A: EKS Provision job

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

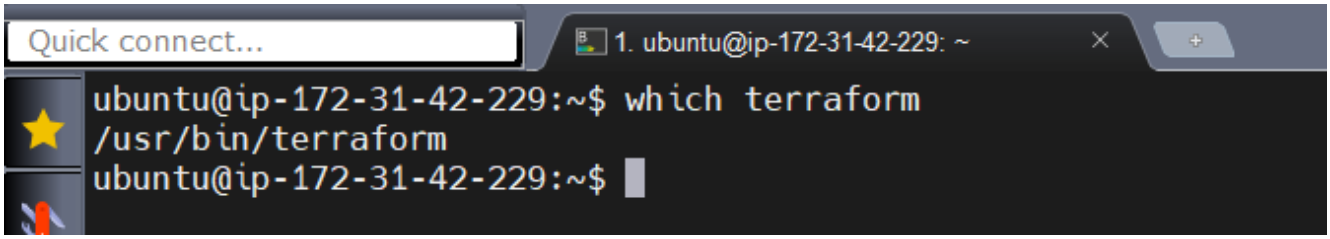
Go to Jenkins dashboard –> Manage Jenkins –> Plugins

Available Plugins, Search for Terraform and install it.

Go to Putty and use the below command

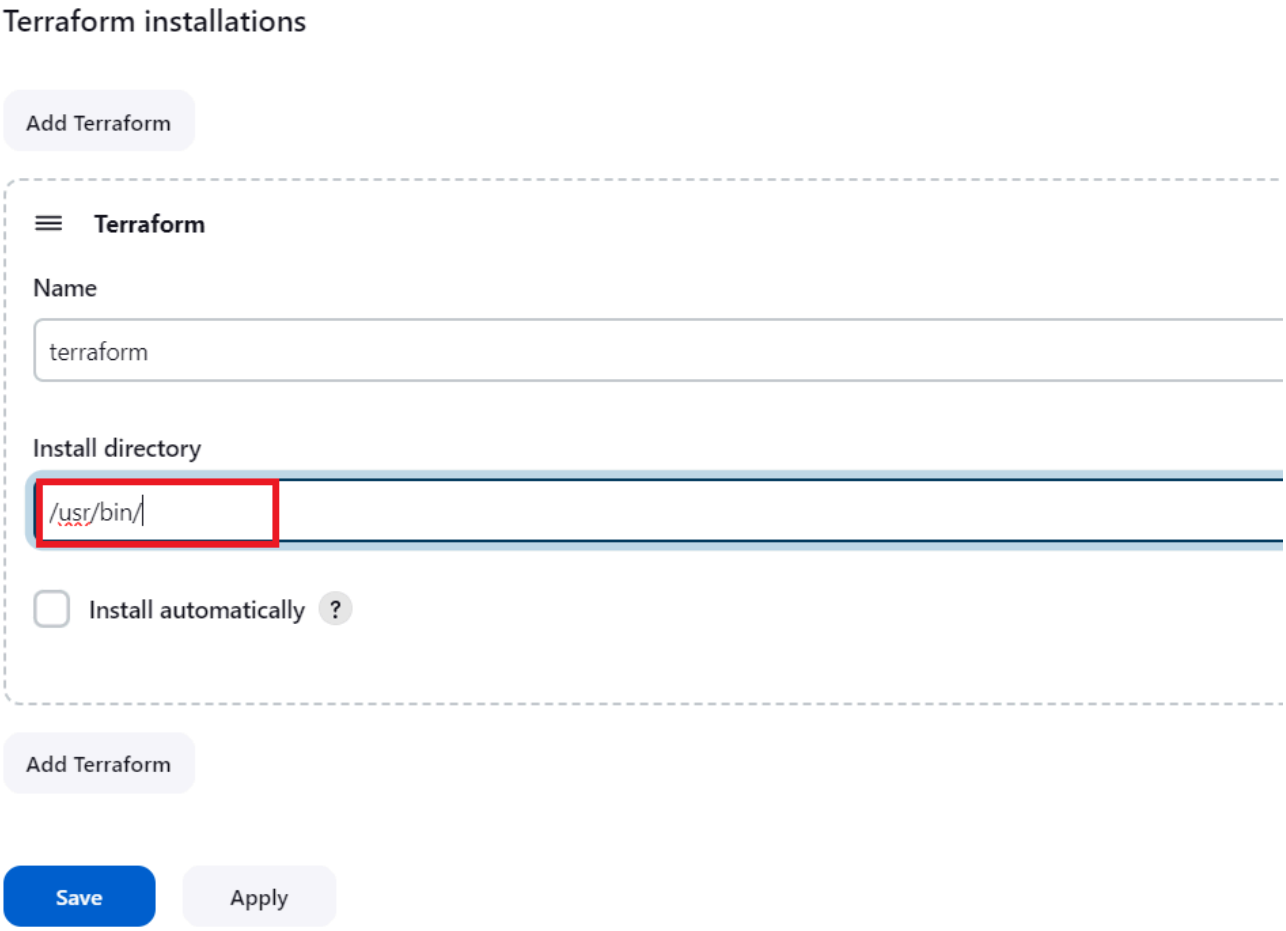
let's find the path to our Terraform (we will use it in the tools section of Terraform)

```
which terraform
```



Now come back to Manage Jenkins –> Tools

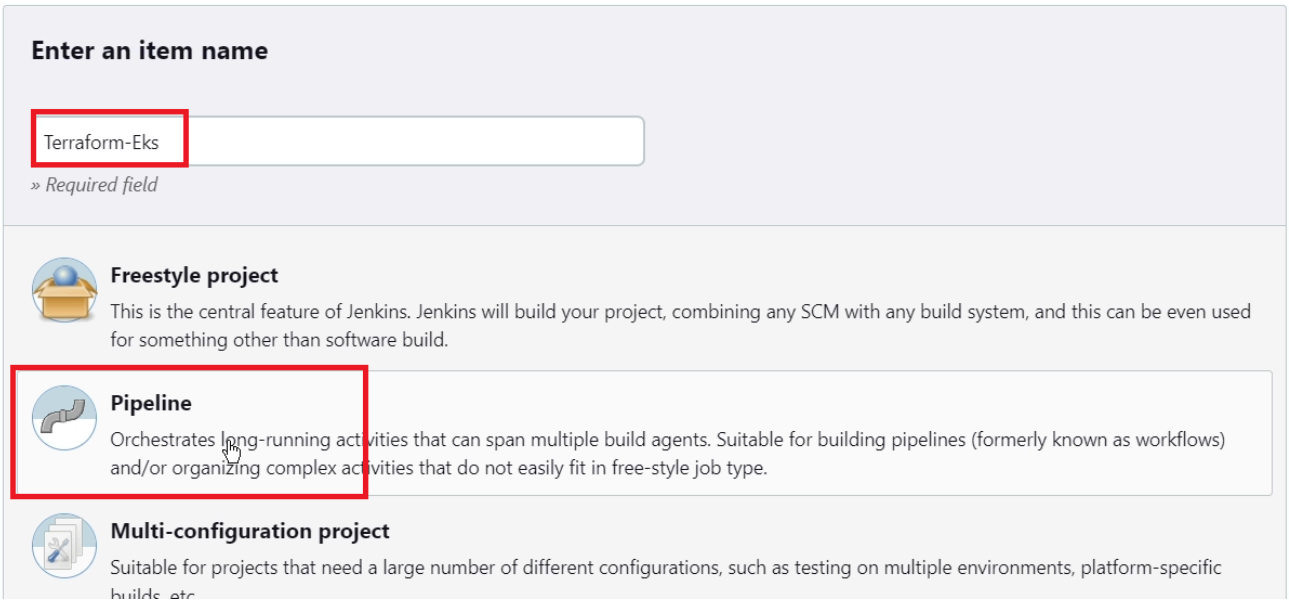
Add the terraform in Tools



Apply and save.

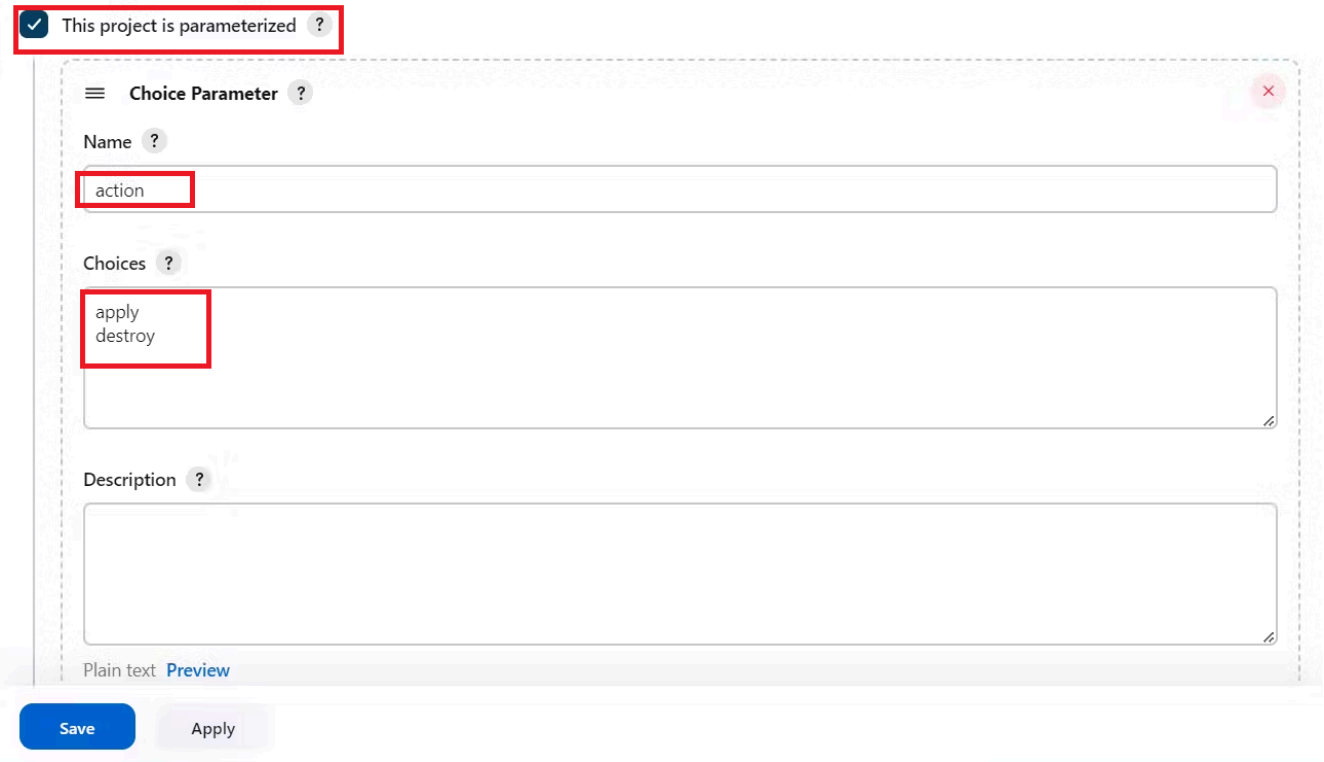
CHANGE YOUR S3 BUCKET NAME IN THE BACKEND.TF

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

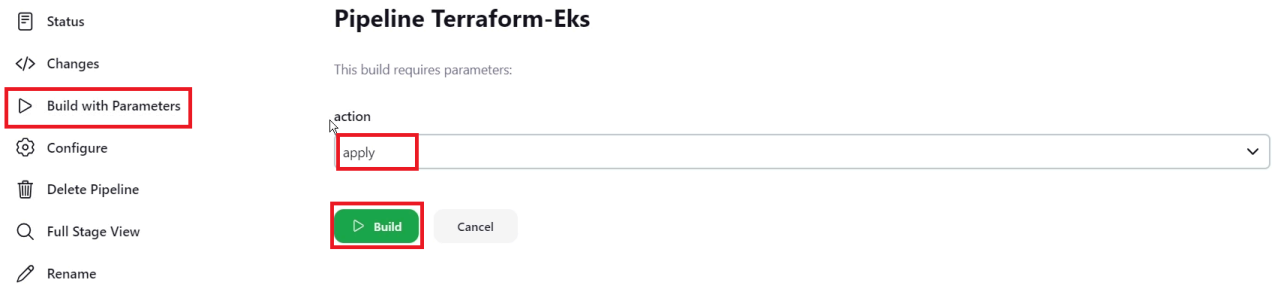


Let's add a pipeline

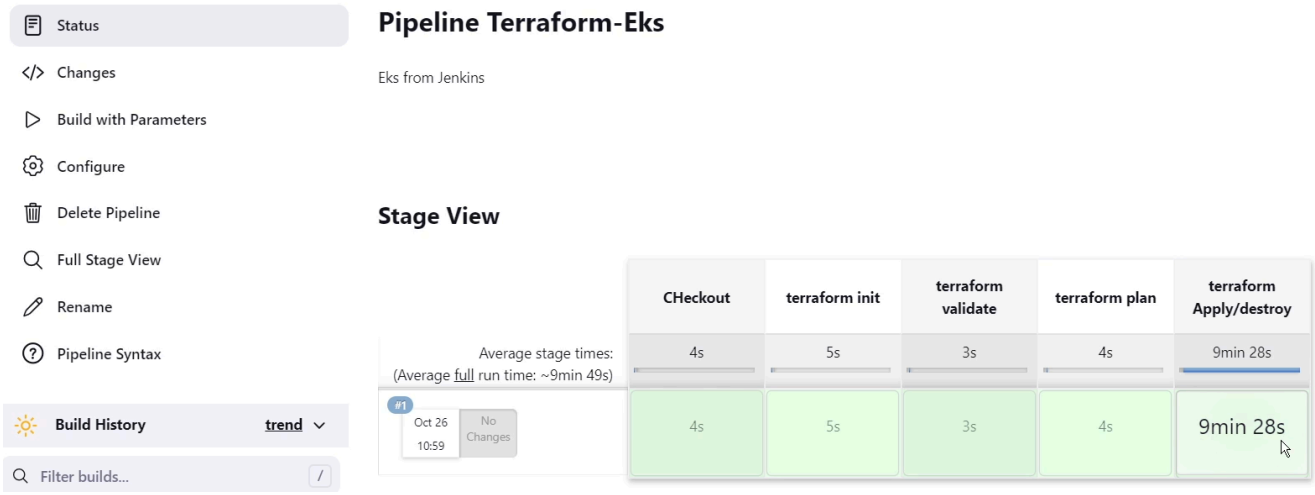
```
pipeline{
    agent any
    stages {
        stage('Checkout from Git'){
            steps{
                git branch: 'main', url: 'https://github.com/Aj7Ay/Hotst
            }
        }
        stage('Terraform version'){
             steps{
                 sh 'terraform --version'
             }
        }
        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'
                   }
             }
        }
    }
}
```

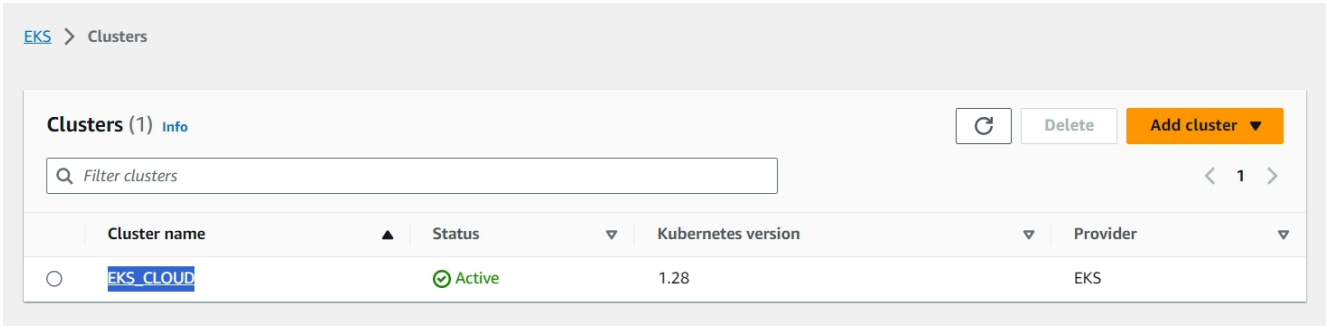
let's apply and save and Build with parameters and select action as apply



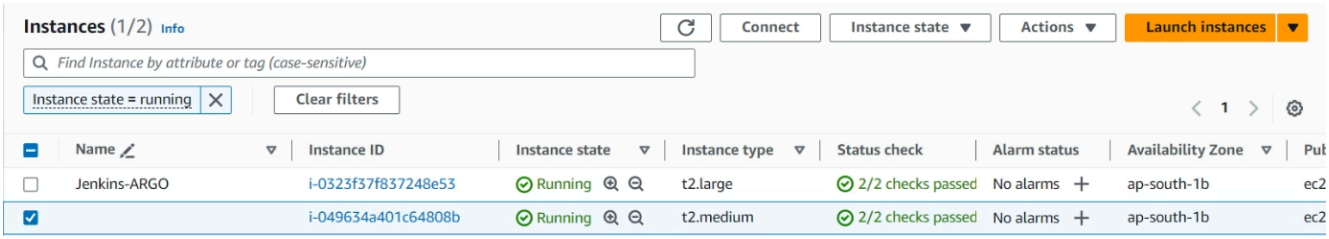
Stage view it will take max 10mins to provision



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



Ec2 instance is created for the Node group



Step 3B: Hotstar job

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

Go to Jenkins dashboard

Manage Jenkins –> Plugins –> Available Plugins

Search for the Below Plugins

```
Eclipse Temurin installer
```

Sonarqube Scanner

NodeJs

Owasp Dependency-Check

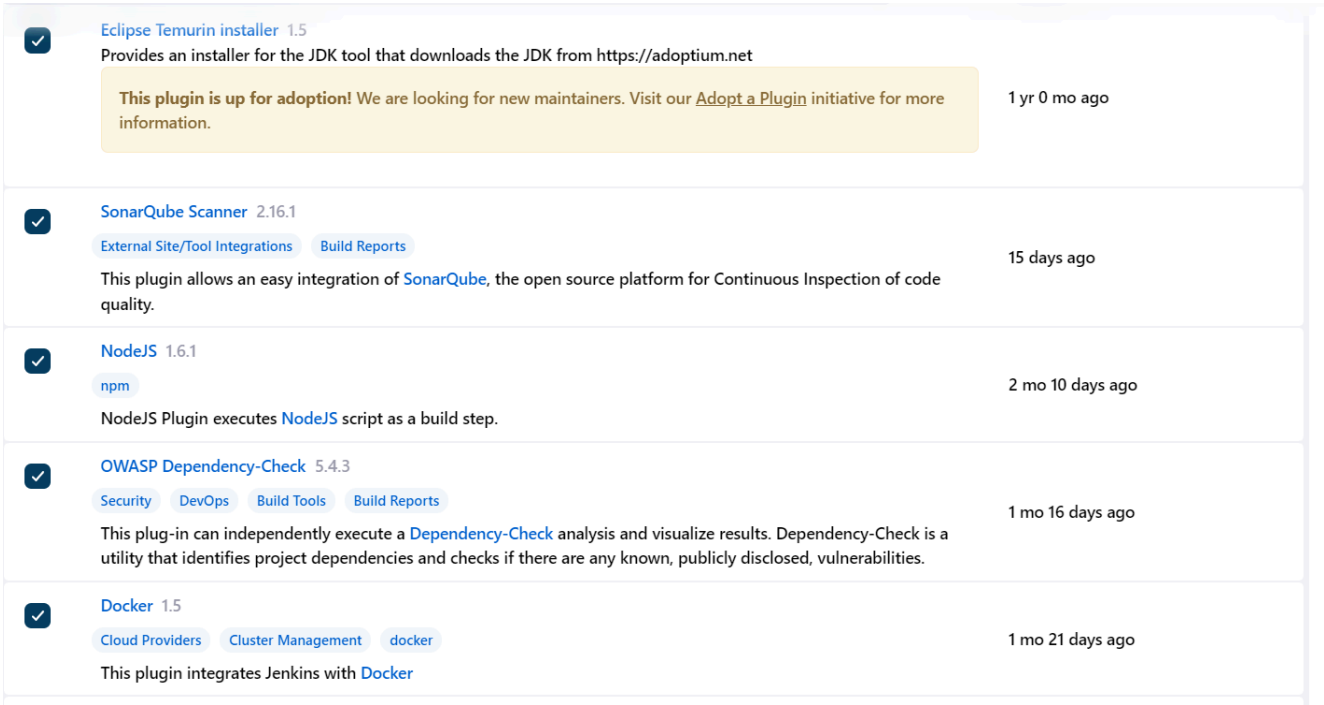
Docker

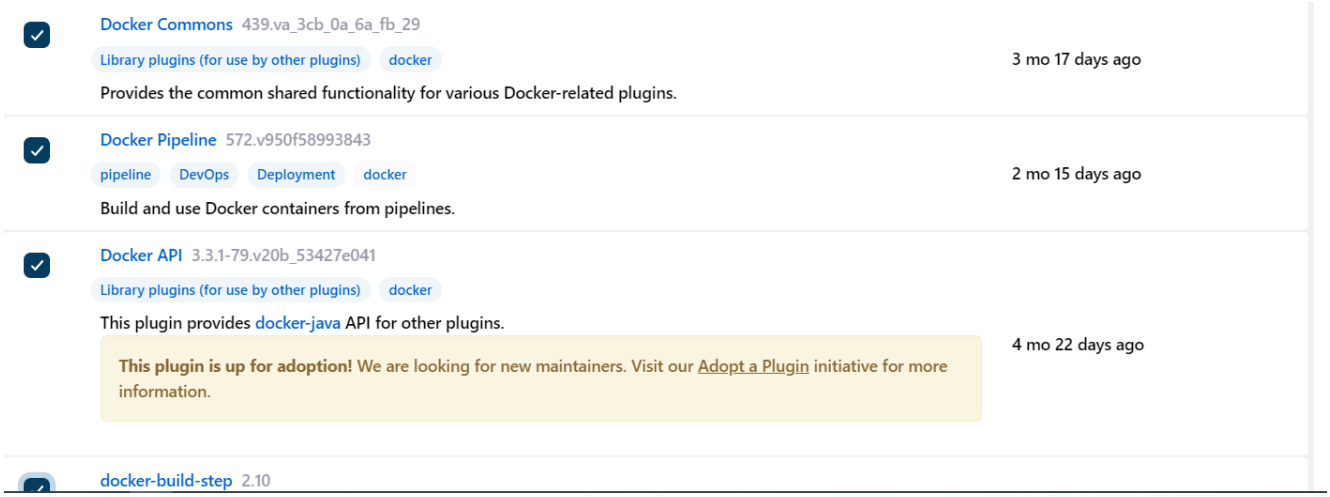
Docker Commons

Docker Pipeline

Docker API

Docker-build-step

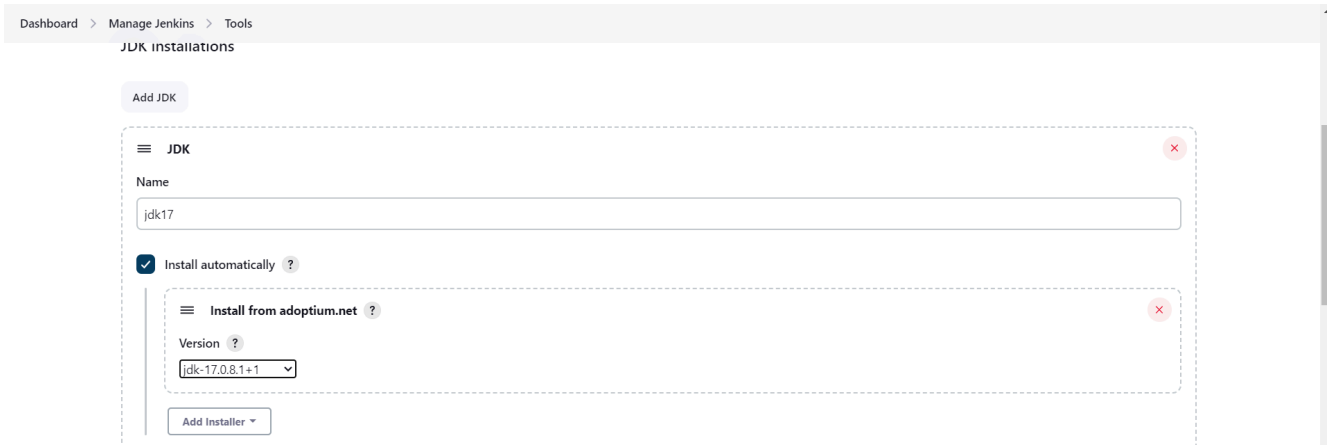




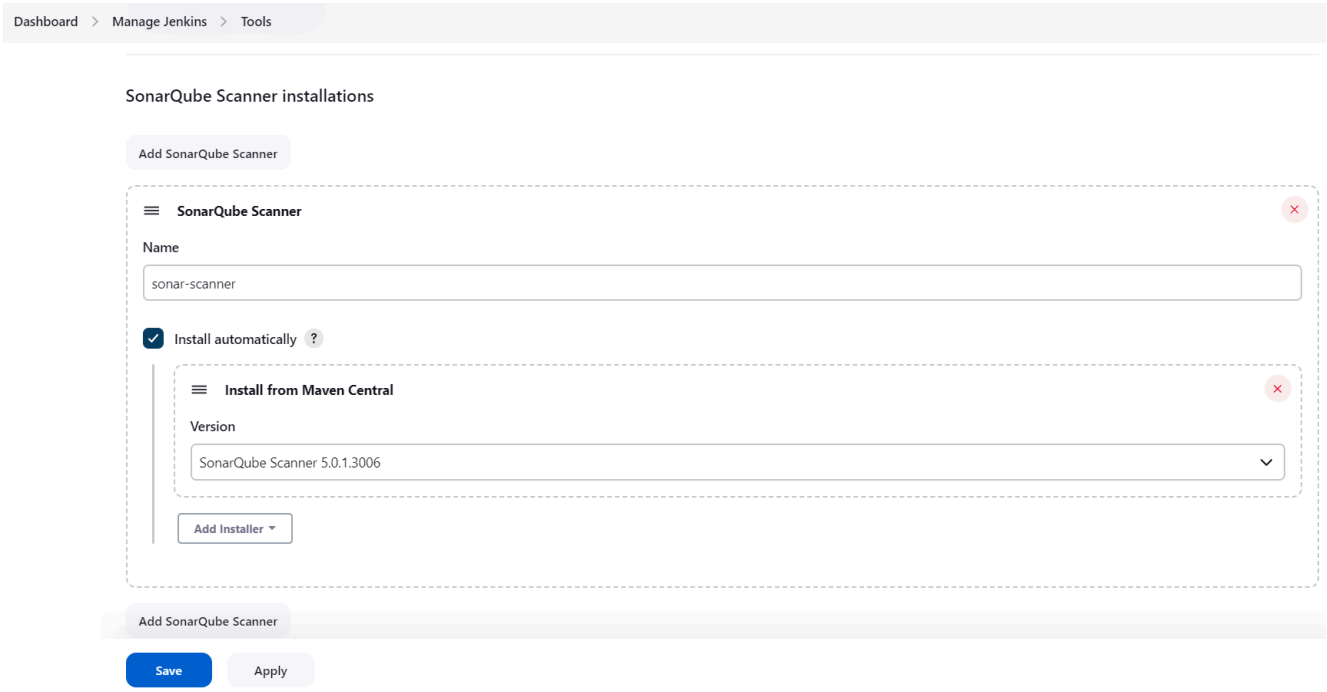
Configure in Global Tool Configuration

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

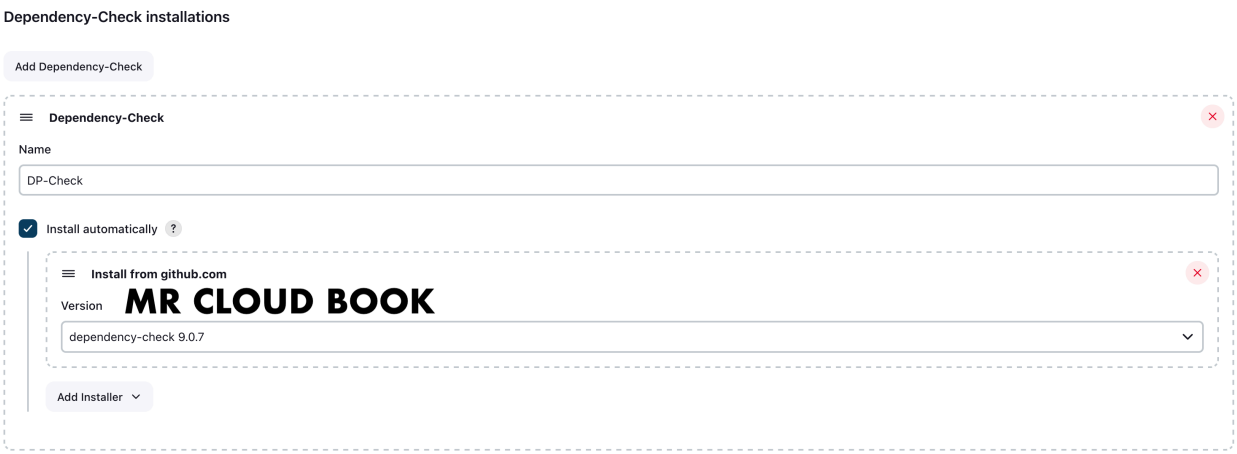
> **NOTE: USE ONLY NODE JS 16**



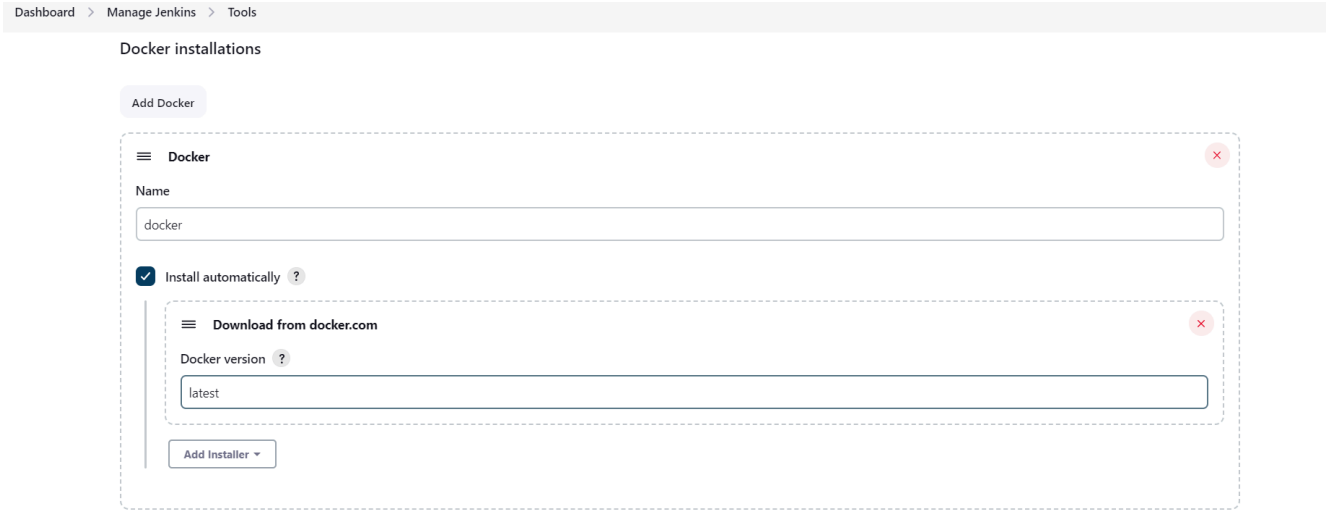
For Sonarqube use the latest version



For Owasp use the 9.0.7 version



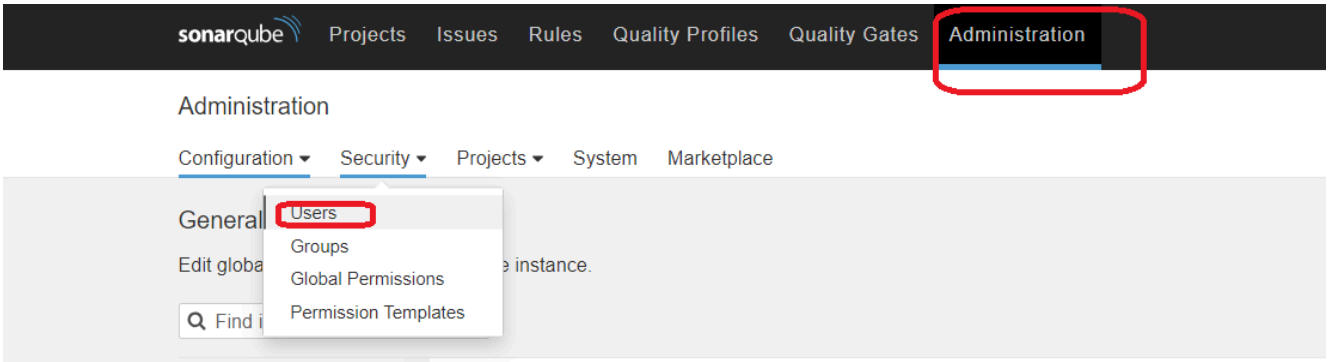
Use the latest version of Docker



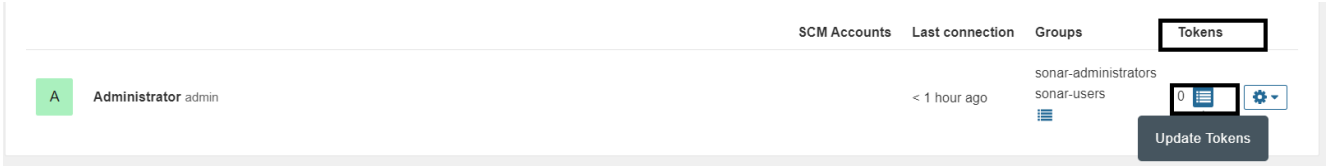
Click apply and save.

Configure Sonar Server in Manage Jenkins

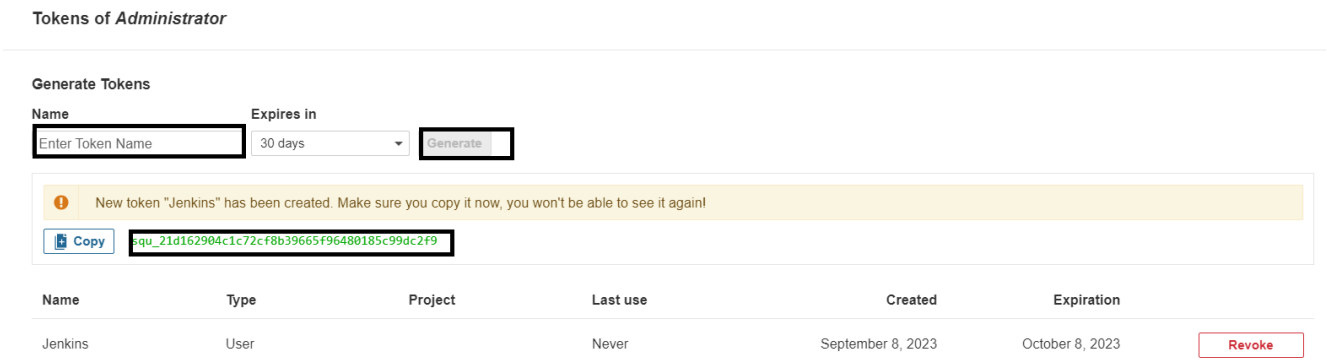
Grab the Public IP Address of your EC2 Instance, Sonarqube works on Port 9000, so <Public IP>:9000. Goto your Sonarqube Server. Click on Administration → Security → Users → Click on Tokens and Update Token → Give it a name → and click on Generate Token



click on update Token

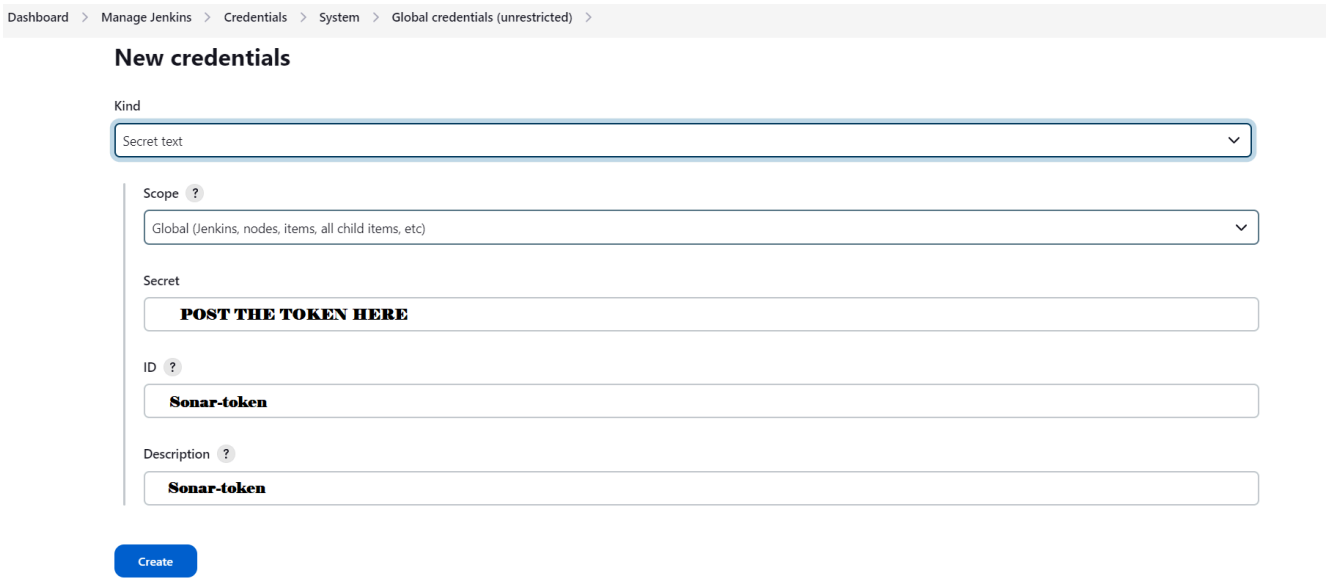


Create a token with a name and generate



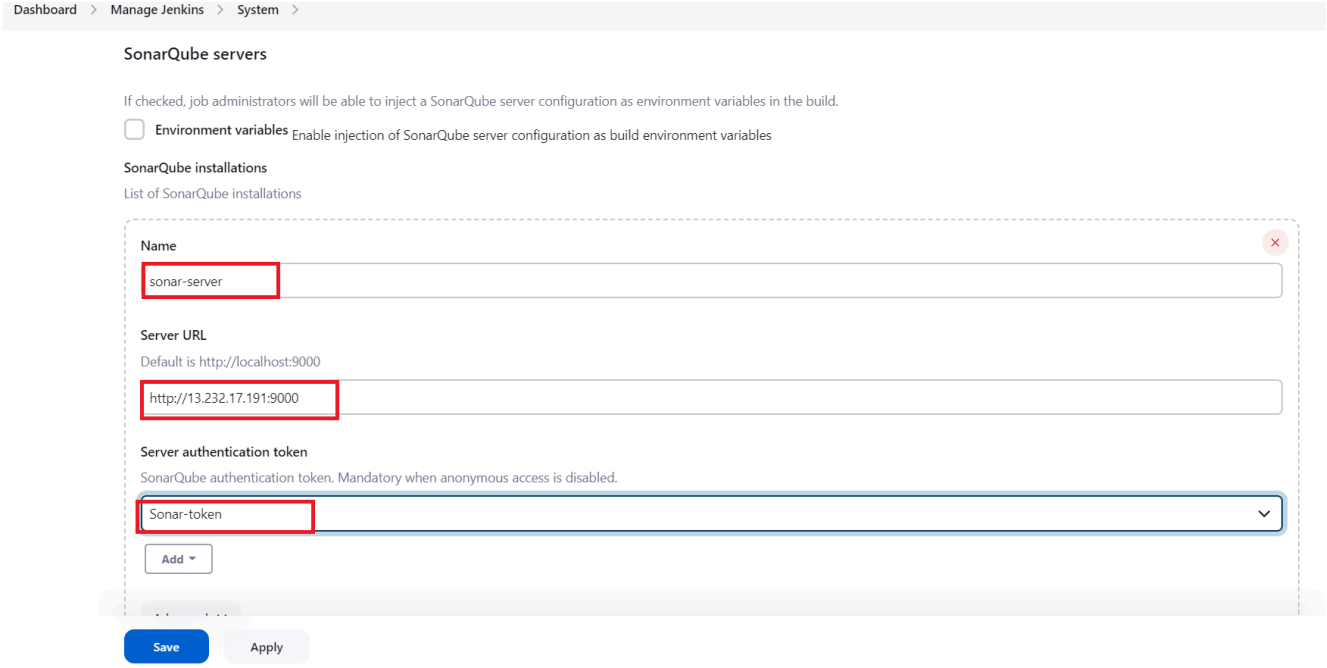
copy Token

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



You will this page once you click on create

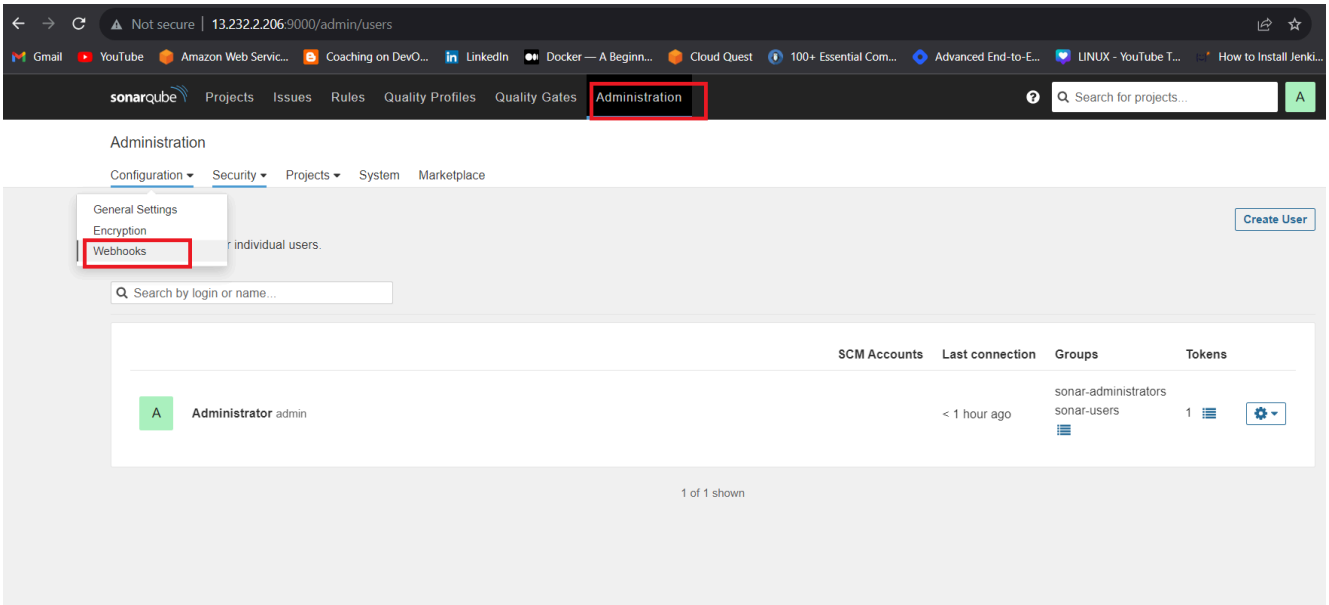
Now, go to Dashboard → Manage Jenkins → System and Add like the below image.



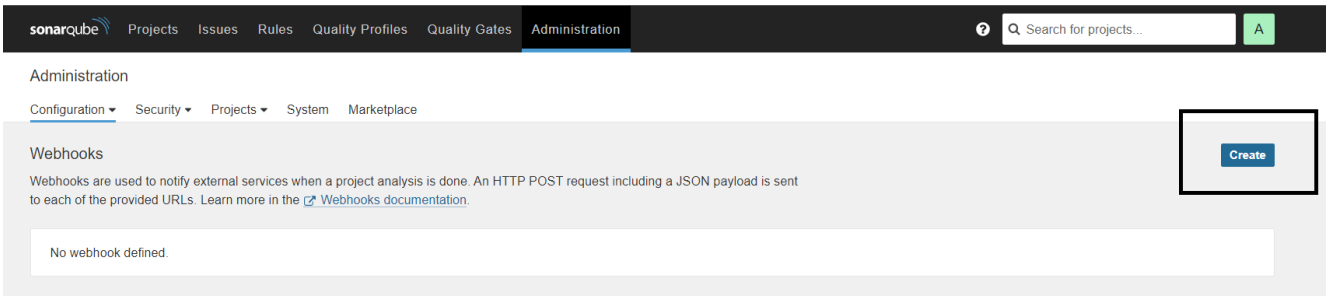
Click on Apply and Save

In the Sonarqube Dashboard add a quality gate also

Administration–> Configuration–>Webhooks

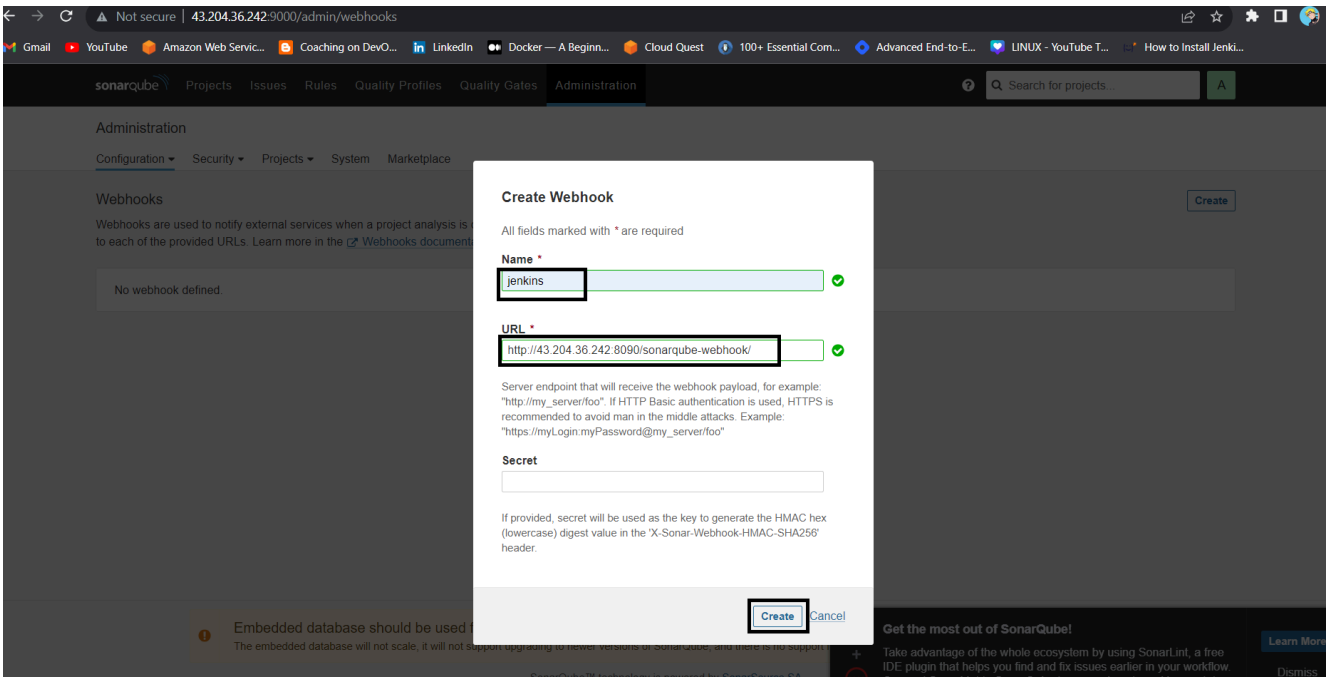


Click on Create



Add details

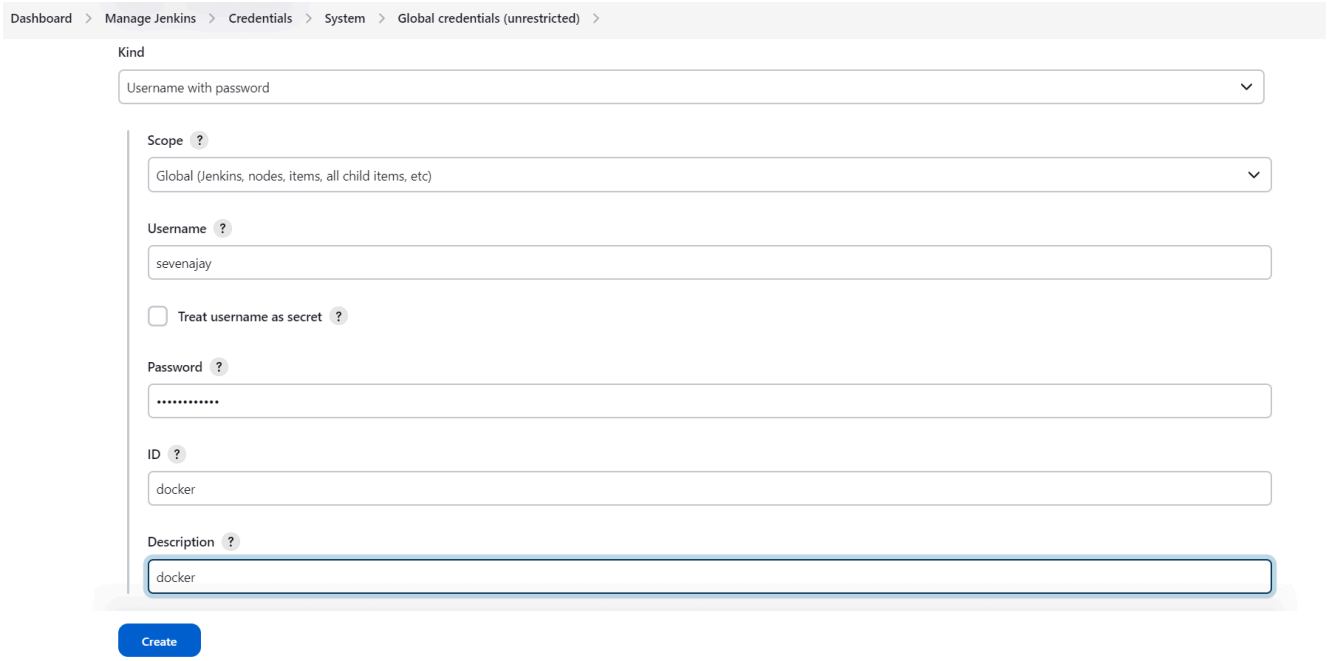
```
#in url section of quality gate
http://jenkins-public-ip:8080&gt;/sonarqube-webhook/
```



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

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

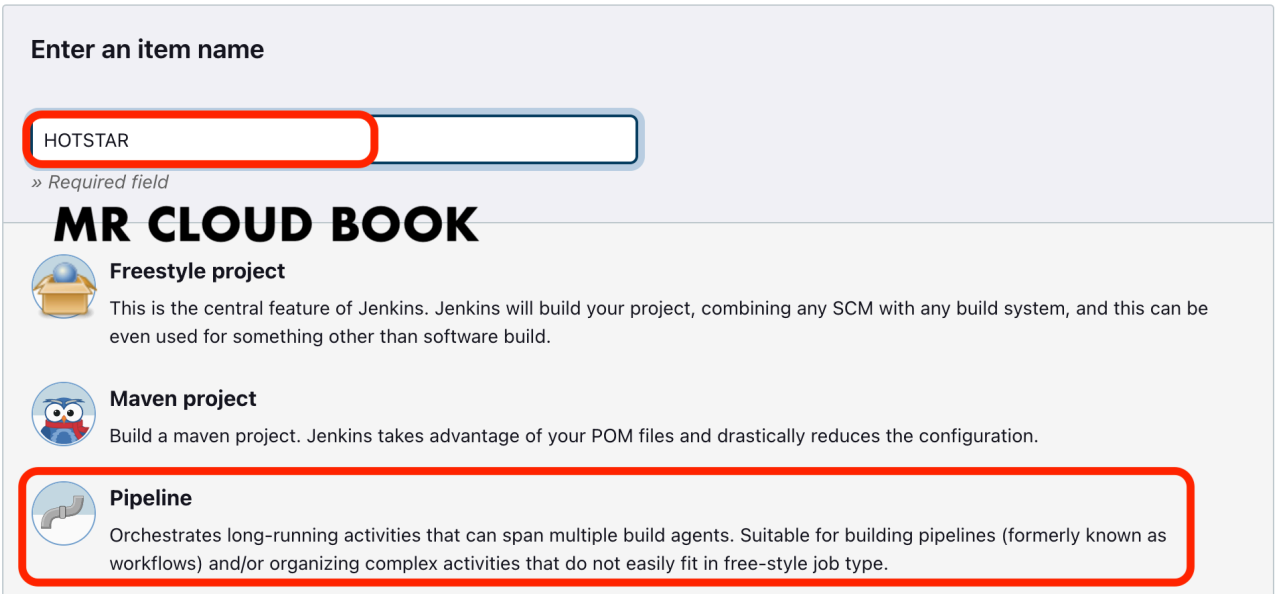
Add DockerHub Username and Password under Global Credentials



Create.

Pipeline upto Docker

Now let's create a new job for our pipeline



Before Adding pipeline install Docker Scout

```
docker login    #use credentials to login
```

```
curl -sSfL https://raw.githubusercontent.com/docker/scout-cli/main/insta
```

Add this 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: 'main', url: 'https://github.com/Aj7Ay/Hotst
            }
        }
        stage("Sonarqube Analysis "){
            steps{
                withSonarQubeEnv('sonar-server') {
                    sh ''' $SCANNER_HOME/bin/sonar-scanner -Dsonar.proje
                    -Dsonar.projectKey=Hotstar'''
                }
            }
        }
        stage("quality gate"){
           steps {
                script {
                    waitForQualityGate abortPipeline: false, credentials
                }
            }
        }
```

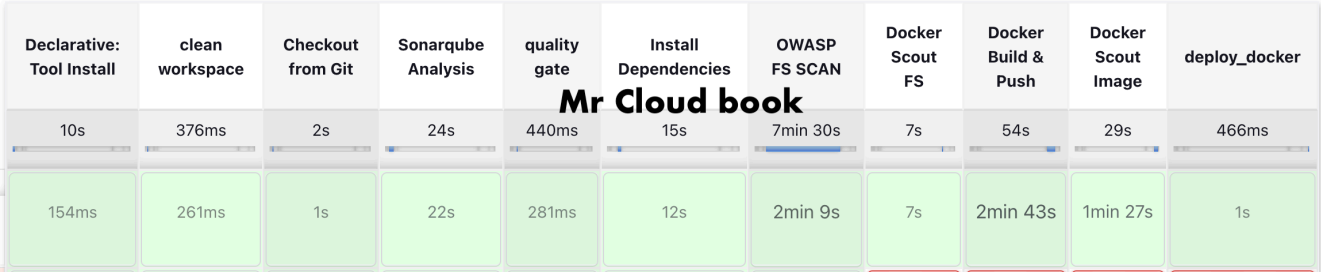
```
stage('Install Dependencies') {
    steps {
        sh "npm install"
    }
}
stage('OWASP FS SCAN') {
    steps {
        dependencyCheck additionalArguments: '--scan ./ --disabl
        dependencyCheckPublisher pattern: '**/dependency-check-r
    }
}
stage('Docker Scout FS') {
    steps {
        script{
           withDockerRegistry(credentialsId: 'docker', toolName
               sh 'docker-scout quickview fs://.'
               sh 'docker-scout cves fs://.'
           }
        }
    }
}
stage("Docker Build &amp;amp; Push"){
    steps{
        script{
           withDockerRegistry(credentialsId: 'docker', toolName
               sh "docker build -t hotstar ."
               sh "docker tag hotstar sevenajay/hotstar:latest '
               sh "docker push sevenajay/hotstar:latest"
            }
        }
    }
}
stage('Docker Scout Image') {
    steps {
        script{
           withDockerRegistry(credentialsId: 'docker', toolName
               sh 'docker-scout quickview sevenajay/hotstar:late
               sh 'docker-scout cves sevenajay/hotstar:latest'
               sh 'docker-scout recommendations sevenajay/hotsta
           }
        }
    }
}
stage("deploy_docker"){
    steps{
```

```
                sh "docker run -d --name hotstar -p 3000:3000 sevenajay/
            }
        }
    }
}
```

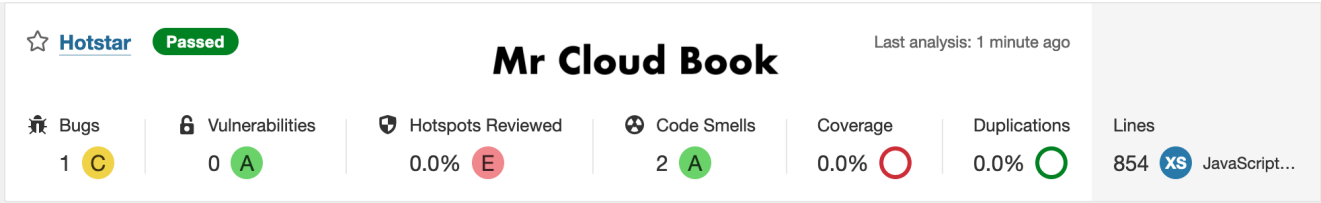
Click on Apply and save.

Build now

Stage view

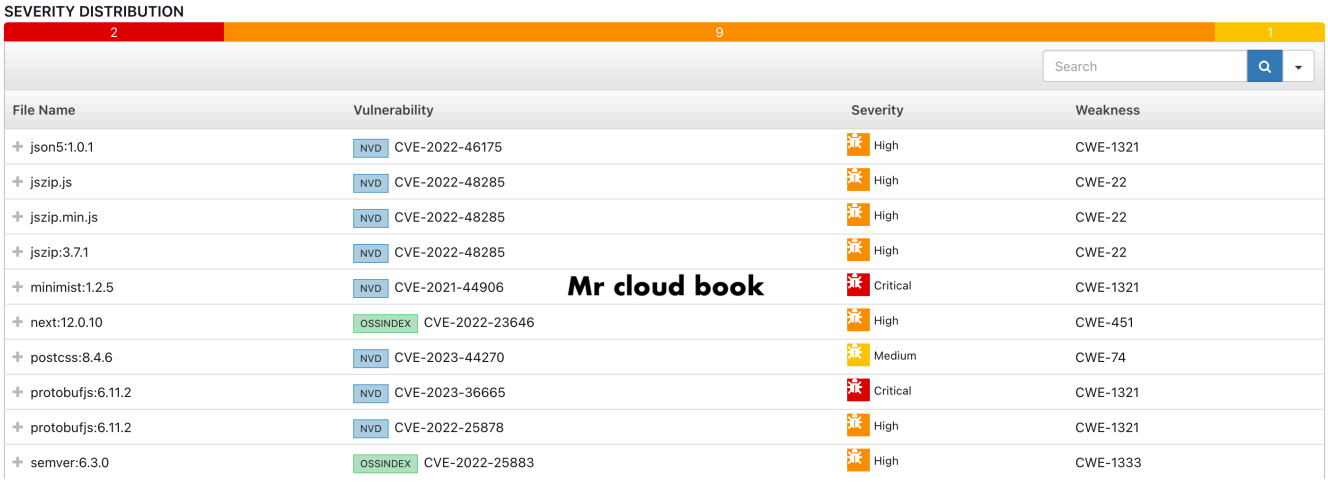


To see the report, you can go to Sonarqube Server and go to Projects.



You can see the report has been generated and the status shows as passed. You can see that there are 854 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.



Let's See Docker Scout File scan report

```
[Pipeline] sh
+ docker-scout quickview fs://.
    ...Reading file system
    ✓ File system read
    ...Indexing
    ✓ Indexed 1257 packages

  Target │  fs://.  │    1C     1H     3M     0L

What's Next?
  View vulnerabilities → docker scout cves fs://.

[Pipeline] sh
+ docker-scout cves fs://.
    ...Reading file system
    ✓ File system read
    ...Indexing
    ✓ Indexed 1257 packages
    ✗ Detected 5 vulnerable packages with a total of 5 vulnerabilities


## Overview

                    │        Analyzed path
────────────────────┼──────────────────────────────
  Target            │  fs://.
    vulnerabilities │    1C     1H     3M     0L


## Packages and Vulnerabilities

   1C     0H     0M     0L  @babel/traverse 7.23.0
pkg:npm/%40babel/traverse@7.23.0

    ✗ CRITICAL CVE-2023-45133 [Incomplete List of Disallowed Inputs]
      https://scout.docker.com/v/CVE-2023-45133
      Affected range : <7.23.2
      Fixed version  : 7.23.2
      CVSS Score     : 9.3
      CVSS Vector    : CVSS:3.1/AV:L/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H
```


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

Let's See Docker Scout Image analysis

Quickview

```
+ docker-scout quickview sevenajay/hotstar:latest
    ...Storing image for indexing
    ✓ Image stored for indexing
    ...Indexing
    ✓ Indexed 1456 packages

  Target             |  sevenajay/hotstar:latest  |    1C     1H     3M     0L
    digest           |  90df4be4e344              |
  Base image         |  node:21-alpine            |    0C     0H     0M     0L
  Updated base image |  node:20-alpine            |    0C     0H     0M     0L
                     |                            |
```

Cves

```
+ docker-scout cves sevenajay/hotstar:latest
    ✓ SBOM of image already cached, 1456 packages indexed
    ✗ Detected 5 vulnerable packages with a total of 5 vulnerabilities


## Overview

                    |       Analyzed Image
--------------------|---------------------------------
  Target            |  sevenajay/hotstar:latest
    digest          |  90df4be4e344
    platform        | linux/amd64
    vulnerabilities |    1C     1H     3M     0L
    size            | 236 MB
    packages        | 1456


## Packages and Vulnerabilities

   1C     0H     0M     0L  @babel/traverse 7.23.0
pkg:npm/%40babel/traverse@7.23.0

    ✗ CRITICAL CVE-2023-45133 [Incomplete List of Disallowed Inputs]
      https://scout.docker.com/v/CVE-2023-45133
      Affected range : <7.23.2
      Fixed version  : 7.23.2
      CVSS Score     : 9.3
      CVSS Vector    : CVSS:3.1/AV:L/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H
```

Recommendations

```
+ docker-scout recommendations sevenajay/hotstar:latest
    ✓ SBOM of image already cached, 1456 packages indexed

  Target   |  sevenajay/hotstar:latest
    digest |  90df4be4e344

## Recommended fixes

  Base image is  node:21-alpine

  Name            |  21-alpine
  Digest          |  sha256:9b54d010b382f0ef176dc93cd829bd4f2a905092b260746b3999aa824c9b7121
  Vulnerabilities |    0C     0H     0M     0L
  Pushed          | 1 week ago
  Size            | 49 MB
  Packages        | 240
  Flavor          | alpine
  OS              | 3.19
  Runtime         | 19
```

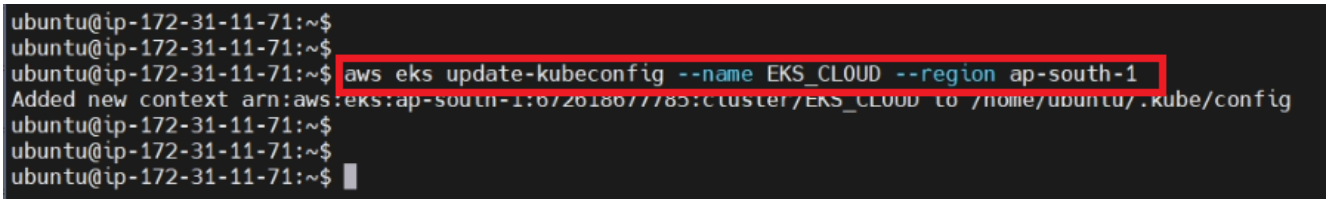
Deploy to Container

```
ec2-ip:3000
```

Output

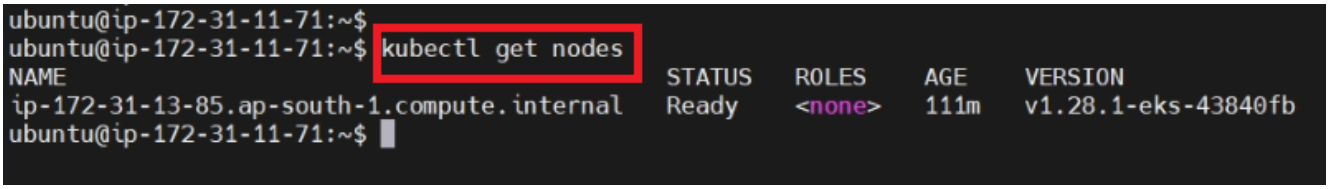
Go to Putty of your Jenkins instance SSH and enter the below command

```
aws eks update-kubeconfig --name CLUSTER NAME --region CLUSTER REGION
aws eks update-kubeconfig --name EKS_CLOUD --region ap-south-1
```



Let's see the nodes

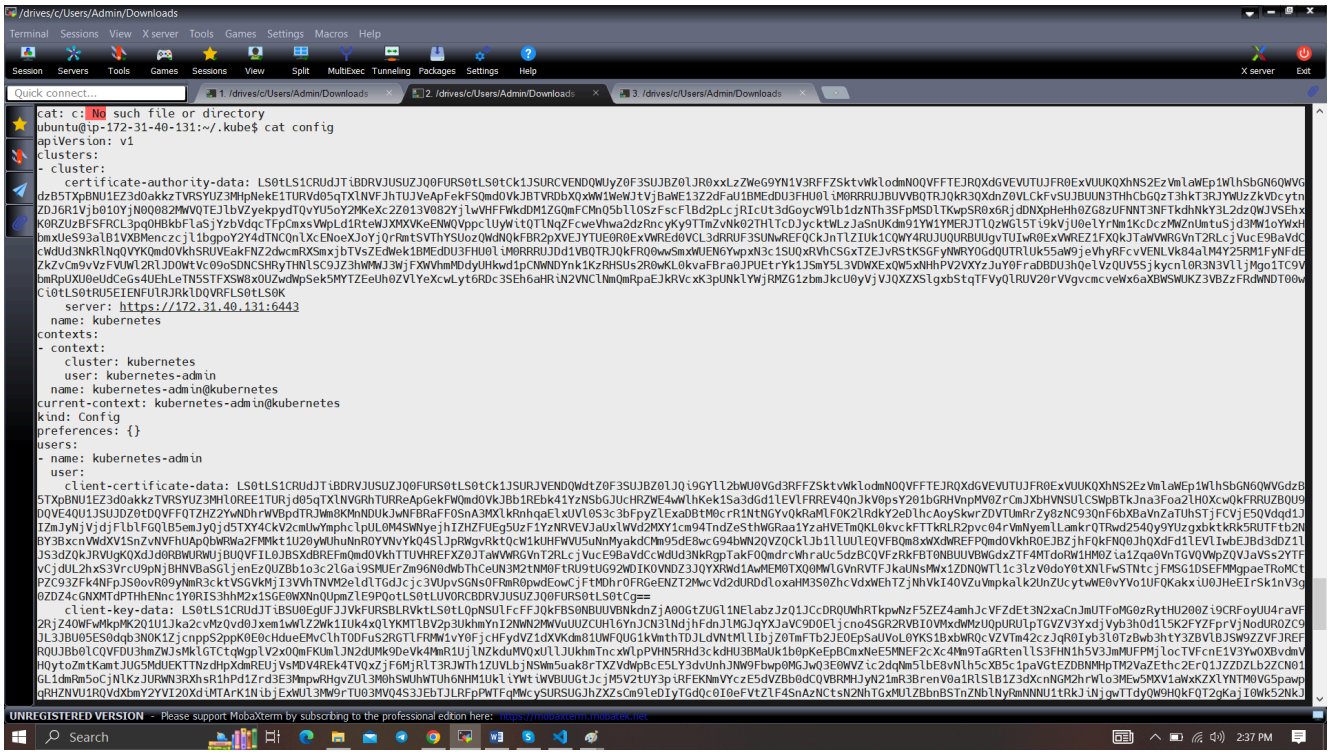
```
kubectl get nodes
```



Now Give this command in CLI

```
cat /root/.kube/config
```

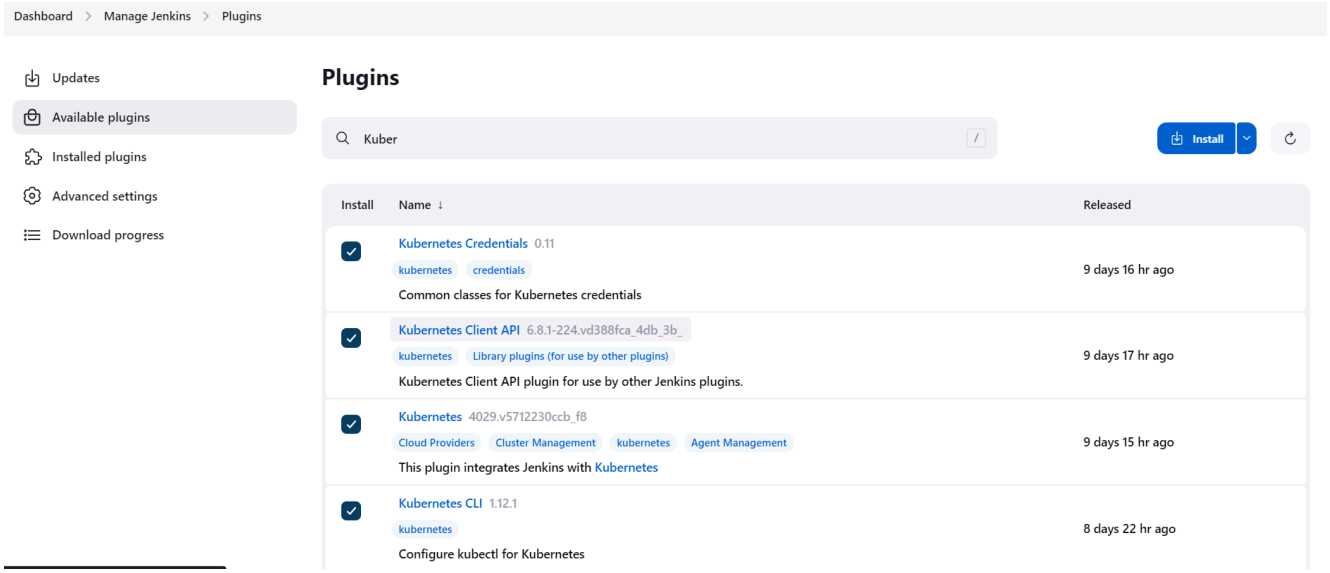
Copy the config file to Jenkins master or the local file manager and save it



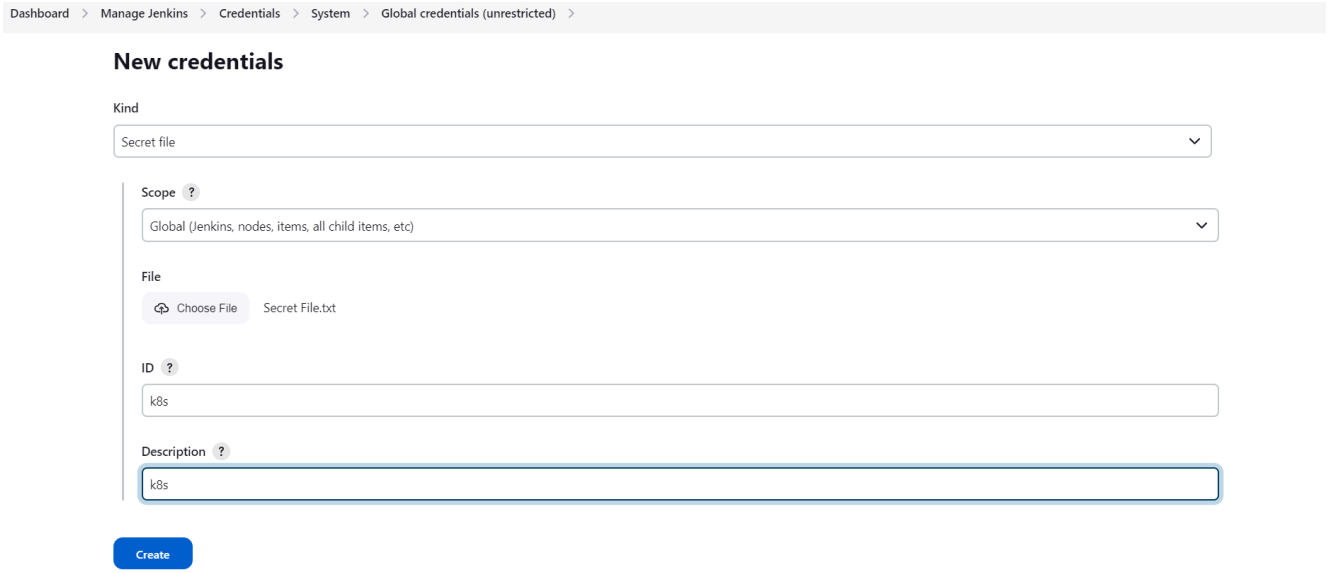
copy it and save it in documents or another folder save it as secret-file.txt

> 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



goto manage Jenkins –> manage credentials –> Click on Jenkins global –> add credentials



final step to deploy on the Kubernetes cluster

```
stage('Deploy to kubernets'){
            steps{
                script{
                    dir('K8S') {
                        withKubeConfig(caCertificate: '', clusterName:
                                sh 'kubectl apply -f deployment.yml'
                                sh 'kubectl apply -f service.yml'
                        }
                    }
                }
            }
        }
```

Give the command after pipeline success

```
kubectl get all
```

> Add Load balancer IP address to cluster ec2 instance security group and copy load balancer Link and open in a browser

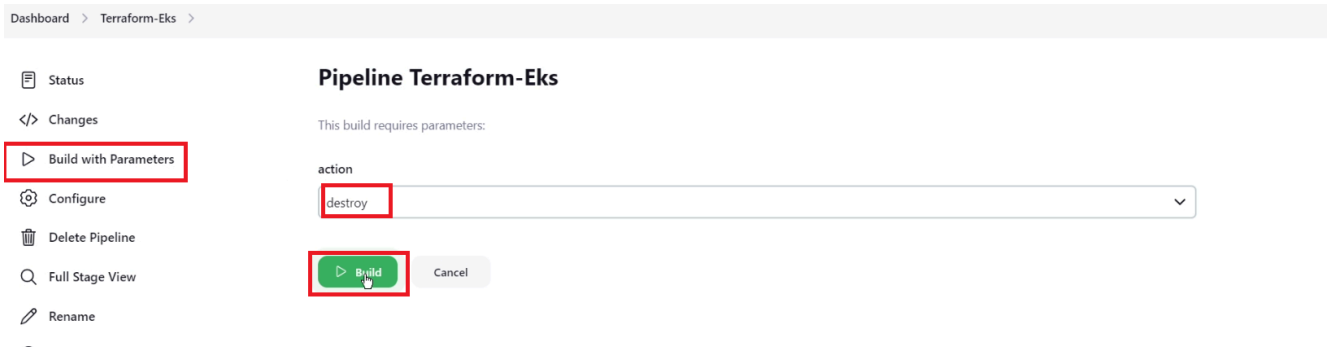
You will see output like this.

Step 4: Destruction

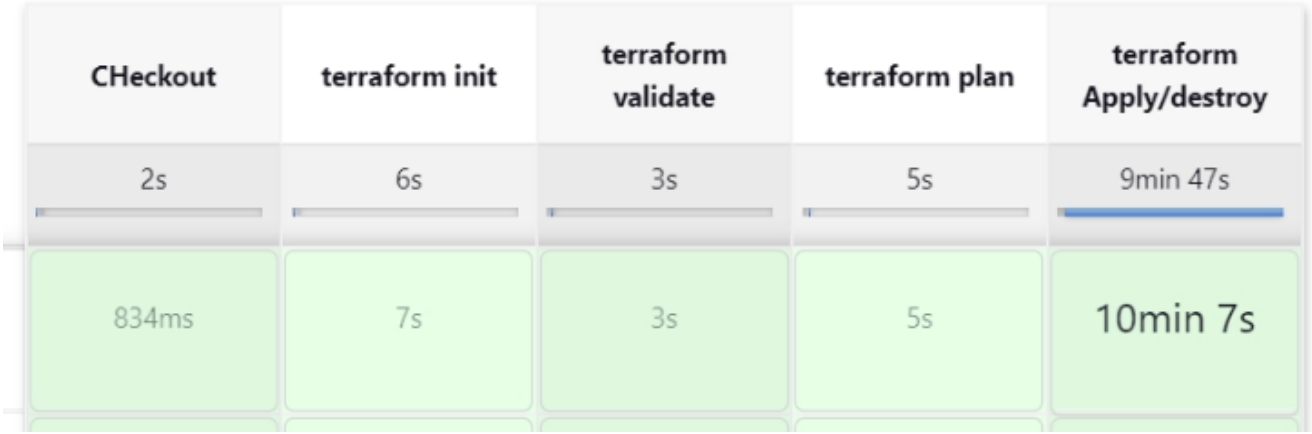
Now Go to Jenkins Dashboard and click on Terraform-Eks job

And build with parameters and destroy action

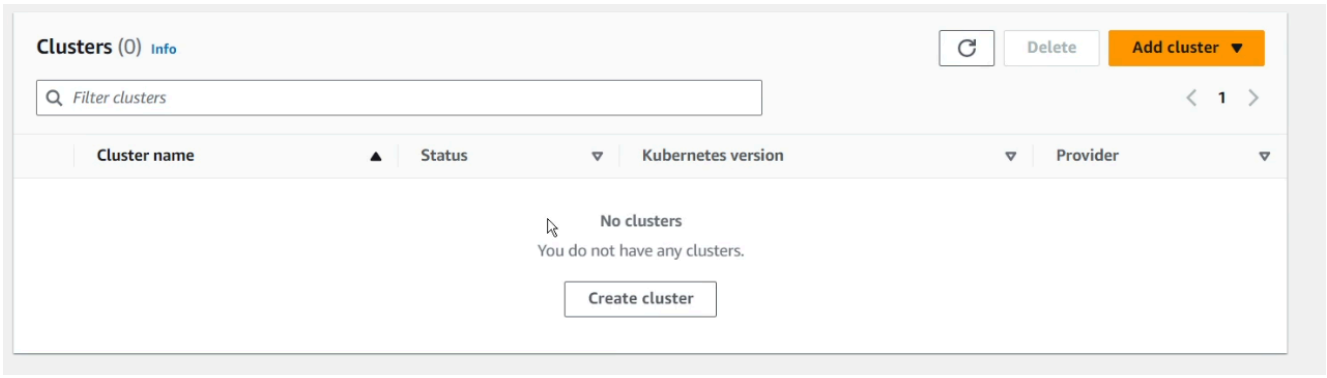
It will delete the EKS cluster that provisioned



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



Cluster deleted



Delete the Ec2 instance & IAM role.

Check the load balancer also if it is deleted or not.

Congratulations on completing the journey of deploying your Hotstar clone using DevSecOps practices on AWS! This process has highlighted the power of integrating security measures seamlessly into the deployment pipeline, ensuring not only efficiency but also a robust shield against potential threats.

Key Highlights:

- Leveraging AWS services, Docker, Jenkins, and security tools, we orchestrated a secure and automated deployment pipeline.
- Implementing DevSecOps principles helped fortify the application against vulnerabilities through continuous security checks.
- The seamless integration of static code analysis, container security, and automated deployment showcases the strength of DevSecOps methodologies.

What's Next? Continue exploring the realm of DevSecOps and cloud technologies. Experiment with new tools, refine your deployment pipelines, and delve deeper into securing your applications effectively.

Stay Connected: We hope this guide has been insightful and valuable for your learning journey. Don't hesitate to reach out with questions, feedback, or requests for further topics. Follow us on [Your Social Media Handles] for more tech tutorials, guides, and updates.

Thank you for embarking on this DevSecOps journey with us. Keep innovating, securing, and deploying your applications with confidence!

ansible Aws ci/cd devops DevSecOps Docker Git

GitHubActions GitLab Jenkins kuberenetes snyk Terraform Trivy

Ajay Kumar Yegireddi is a DevSecOps Engineer and System Administrator, with a passion for sharing real-world DevSecOps projects and tasks. **Mr. Cloud Book**, provides hands-on tutorials and practical insights to help others master DevSecOps tools and workflows. Content is designed to bridge the gap between development, security, and operations, making complex concepts easy to understand for both beginners and professionals.

Comments

24 responses to "DevSecOps CI/CD : Deploying a Secure Hotstar Clone (Even if You're Not a Pro)"

Abdul Salam
6 January 2024

Thank You for Creating Such valuable content project it really helps us to upgrade our skills and Thank you once again Mr. Cloud Book and keep doing it its our humble request from us.

Reply

mrcloudbook.com
7 January 2024

Thanks bro
will do

Reply

Rajesh
7 January 2024

Thank you Ajay for explaining such wonderfull ,in detail manner, the way you have presented is just awesome, keep up good work . I love to see you contributing so much to the community .

Reply

mrcloudbook.com
7 January 2024

Thanks for support brother

Reply

Srinivas
8 January 2024

Thanks Ajay. Wonderful demonstration. I have also implemented this project and learnt lot of things. If possible please do videos on GCP DevOps too. Thanks!

Reply

mrcloudbook.com
8 January 2024

will try bro

Reply

Jay
9 January 2024

You're the EYE. Great job brother!

Reply

mrcloudbook.com
9 January 2024

Thanks brother

Reply

Hieu Luong
13 January 2024

Thanks for your great share. Keep your journey, bro 🙂

Reply

mrcloudbook.com
14 January 2024

definitely bro

Reply

Shivasantosh
14 January 2024

This is very much helpful... Thank you very much...Please provide details of different projects also.. Advance happy Sankranti to everyone...

Reply

mrcloudbook.com
14 January 2024

okay bro happy sankranthi

Reply

Nagaraj
14 January 2024

Hi Bro,

Fantastic information and explaination,

Regards,
Nagaraj CH

Reply

mrcloudbook.com
14 January 2024

Thanks bro

Reply

Ved
11 February 2024

sh ‘terraform ${action} –auto-approve’

i think this command should be

sh ‘terraform apply ${action} –auto-approve’

Awesome Work Bro,Keep Going your journey.

Reply

Ved
11 February 2024

sh ‘terraform ${action} –auto-approve’

i think this command should be

sh ‘terraform apply ${action} –auto-approve’

Awesome Work Bro,Keep goin your journey.

Reply

mrcloudbook.com
12 February 2024

It will automcatically take from build
no need for apply in command

Reply

Alfiya
21 February 2024

NIce!

Reply

mrcloudbook.com
6 March 2024

Thanks

Reply

Pardeep
13 March 2024

Thank you Ajay. i have completed the entire project after no. of tries. but in the end still after applying kubernete stage. i got some error. in last stage what will be the serverurl? i added. stage('Deploy to kubernets'){
steps{
script{
dir('K8S') {
withKubeConfig(caCertificate: '', clusterName: 'EKS_Cloud', contextName: '', credentialsId: 'K8s', namespace: 'default', restrictKubeConfigAccess: false, serverUrl: 'https://5A0A1F4D004A601558FEE65B5F7CF0F0.gr7.ca-central-1.eks.amazonaws.com') {
sh 'kubectl apply -f deployment.yml –validate=false'
sh 'kubectl apply -f service.yml –validate=false'
}
}
}
}
}

i got error :

+ kubectl apply -f deployment.yml
error: error validating "deployment.yml": error validating data: failed to download openapi: Get "https://2D9148EDEB83A1C2147C731BBA6D7104.gr7.ca-central-1.eks.amazonaws.com/openapi/v2?timeout=32s": dial tcp: lookup 2D9148EDEB83A1C2147C731BBA6D7104.gr7.ca-central-1.eks.amazonaws.com on 127.0.0.53:53: no such host; if you choose to ignore these errors, turn validation off with –validate=false
[Pipeline] }
[kubernetes-cli] kubectl configuration cleaned up
[Pipeline] // withKubeConfig

[Pipeline] }
[Pipeline] // dir
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
ERROR: script returned exit code 1
Finished: FAILURE
it shows this error

later on anyhow i overcome this error but when i deploy on kuberentes the loadbalancer ip address doesnot work.

Reply

Saketh
17 March 2024

HI mr cloud book , can YOU suggest me some projects to keep in my resume

Reply

mrcloudbook.com
17 March 2024

Do your own projects with the above process Brother
everyone uses same projects

Reply

Umer Asif
5 February 2025

Hi Bro,

I want to do these projects. However , I was curious if creating Ec2 etc would cost me money or it can be done in free aws account?

Reply

36.01hw34k15gesbqggjtzbn6zd79@mail4u.life
12 May 2024

velit error pariatur placeat natus. veritatis assumenda laborum quasi incidunt esse quia aut facilis. aut maxime ipsa a placeat quasi corporis. qui magnam saepe est id aliquam ipsa quis in deserunt la

Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment *

Name *

Email *

Website

Save my name, email, and website in this browser for the next time I comment.

I'm not a robot

reCAPTCHA
Privacy - Terms

Post Comment

Uncategorized

How to Automate Incident Response : How Q Developer Helped Me Automate a Daily Pain Point

22 July 2025

AI

How to Run Docker Model Runner on Ubuntu 24.04

11 July 2025

AI, DevOps

How to Install docker-ai on Ubuntu 24.04

15 June 2025

Upskill with Ajay: DevSecOps Mastery

Join Mr Cloud book to master DevSecOps through real-world projects. Learn CI/CD, security integration, automation, and more, gaining hands-on skills for industry-level challenges.

Important Links

Privacy Policy

Terms & Conditions

Contact

Resources

Blog

YouTube Channel

Follow Us on YouTube