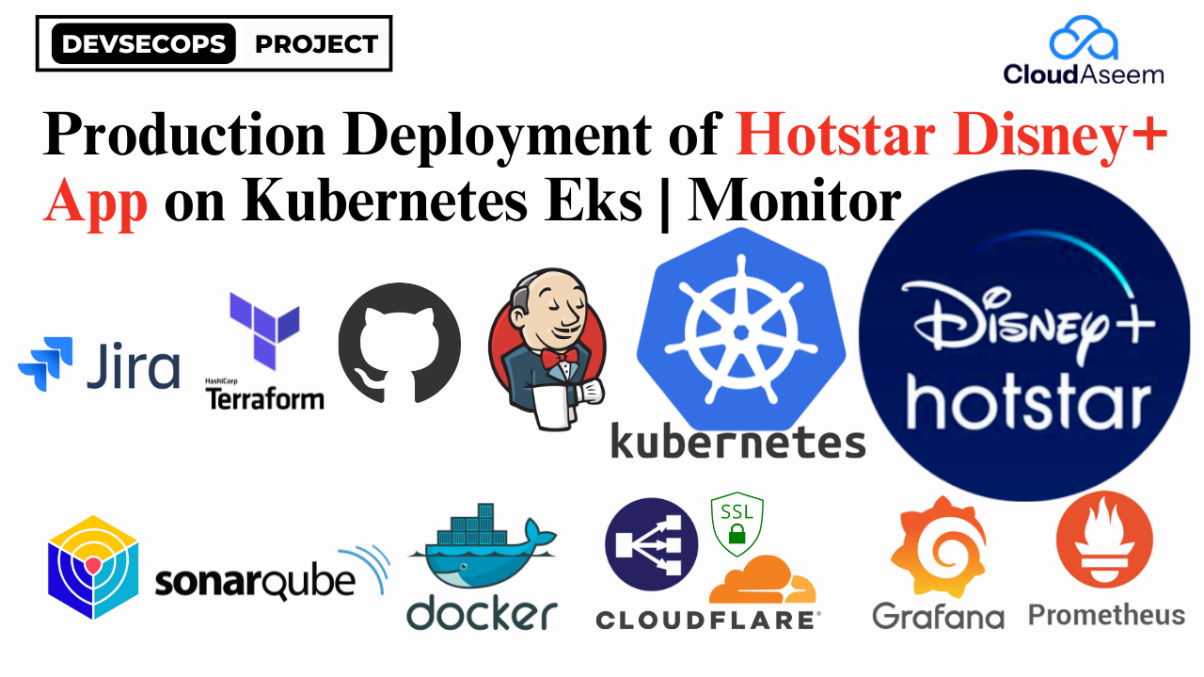
Disney+ Hotstar Clone App — End-to-End Secure & Scalable Deployment with DevSecOps & Jenkins !

Thrilled to showcase our latest project — a Hotstar **Clone Application** engineered with a **DevSecOps CI/CD pipeline** and powered by **Jenkins** Parameterise Build to deliver fully automated, secure, and scalable deployments.



🌎 **Follow me for projects Links :**
🔗 **YouTube:** https://youtu.be/VPJ4gesLXOc
🔗 **Project GitHub Repo:** https://github.com/Aseemakram19/hotstar-kubernetes.git
🔗 **LinkedIn:** Mohammed Aseem Akram
🔗 **GitHub Account:** Aseemakram19

📢 **If you found this tutorial helpful, don't forget to:**
👍 **Like** the video
💬 **Comment** your thoughts and questions
🔔 **Subscribe** to stay updated on **Cloud & DevOps tutorials!**

🔑 **Tech Stack** & **Key Integrations**:

✅ Kubernetes & Docker — for seamless, containerized, and scalable application deployments
✅ Jenkins — to enable reusable, standardized, and consistent CI/CD pipelines
✅ SonarQube, Trivy, OWASP Dependency-Check — ensuring robust security, vulnerability scanning, and code quality automation
✅ Prometheus & Grafana — for comprehensive real-time monitoring and alerting

✅ Gmail Email alerts and collaboration-driven notifications
✅ Parameterized environment orchestration — for on-demand infrastructure setup and teardown

🔥 **Key Highlights** & **Takeaways**:

🔐 Security-by-design with integrated DevSecOps practices
⚙️ Automated, reusable pipelines ensuring consistency and efficiency
📈 Production-grade scalability leveraging Kubernetes
🚀 Rapid deployment lifecycle driven by modern CI/CD automation

This solution represents a blueprint for secure, scalable, and production-ready application delivery, embodying best practices in DevSecOps, automation, and cloud-native architecture.

CLICK HERE FOR GITHUB REPO
https://github.com/Aseemakram19/hotstar-kubernetes.git

Now, let's get started and dig deeper into each of these steps: -

I. Configure Infrastructure In AWS Cloud

1. Launch an EC2 Instance Ubuntu (22.04) T3 X Large Instance

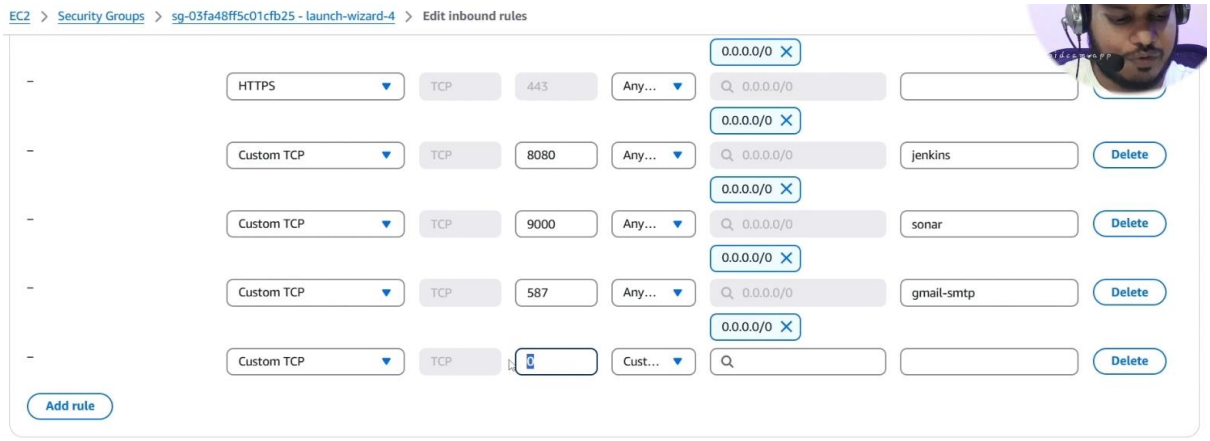
- Go to the **AWS Management Console** → **EC2** → **Instances**.
- Click **Launch Instance**.
- Set the following configurations:
  - **Name**: <Your_Instance_Name>
  - **AMI (Amazon Machine Image)**: Ubuntu Server 22.04 LTS (HVM), SSD Volume Type
  - **Instance Type**: t3.xlarge (4 vCPUs, 16 GB RAM)
  - **Key Pair**: Select an existing key pair or create a new one.
  - **Storage**: Default (e.g., 30GB GP3 SSD, adjust as needed).

2. Configure Security Group

Create or modify a security group to allow the following ports:

Port	Protocol	Description
22	TCP	SSH (for remote access)
80	TCP	HTTP (Web traffic)
443	TCP	HTTPS (Secure web traffic)
8080	TCP	Web applications (Tomcat, etc.)
587	TCP	SMTP (Email sending)
465	TCP	SMTP over SSL
3000	TCP	Web apps (Grafana, Node.js, etc.)
9000	TCP	SonarQube/Web apps

- Set the **Source** to **Anywhere (0.0.0.0/0, ::/0)** unless you want to restrict access.



3. Launch & Connect with Mobaxterm App

- Click **Launch Instance**.
- Once the instance is running, connect using:

- **Install Jenkins, Docker , awscli, terraform, kubectl , eksctl and Trivy,**
  Clone the GITHUB Project repositories https://github.com/Aseemakram19/hotstar-kubernetes.git

  cd **hotstar-kubernetes/scripts/**

- Install the TOOLS in the VM machine via Scrtipts . add executable permission to shell script
  chmod +x *.sh

```
echo "Making all .sh files executable..."
chmod +x *.sh

echo "Permissions updated successfully!"
ubuntu@ip-10-0-1-103:~/hotstar-kubernetes/scripts$ chmod +x *.sh
ubuntu@ip-10-0-1-103:~/hotstar-kubernetes/scripts$ ll
total 44
drwxrwxr-x 2 ubuntu ubuntu 4096 Mar 19 20:21 ./
drwxrwxr-x 8 ubuntu ubuntu 4096 Mar 19 20:21 ../
-rwxrwxr-x 1 ubuntu ubuntu  396 Mar 19 20:21 awscli.sh*
-rwxrwxr-x 1 ubuntu ubuntu  820 Mar 19 20:21 docker.sh*
-rwxrwxr-x 1 ubuntu ubuntu  393 Mar 19 20:21 eksctl.sh*
-rwxrwxr-x 1 ubuntu ubuntu  939 Mar 19 20:21 grafana.sh*
-rwxrwxr-x 1 ubuntu ubuntu  570 Mar 19 20:21 jenkins.sh*
-rwxrwxr-x 1 ubuntu ubuntu  460 Mar 19 20:21 kubectl.sh*
-rwxrwxr-x 1 ubuntu ubuntu  294 Mar 19 20:21 permissionexecute.sh*
-rwxrwxr-x 1 ubuntu ubuntu  846 Mar 19 20:21 terraform.sh*
-rwxrwxr-x 1 ubuntu ubuntu  637 Mar 19 20:21 trivy.sh*
ubuntu@ip-10-0-1-103:~/hotstar-kubernetes/scripts$ sh
```

Install Tools

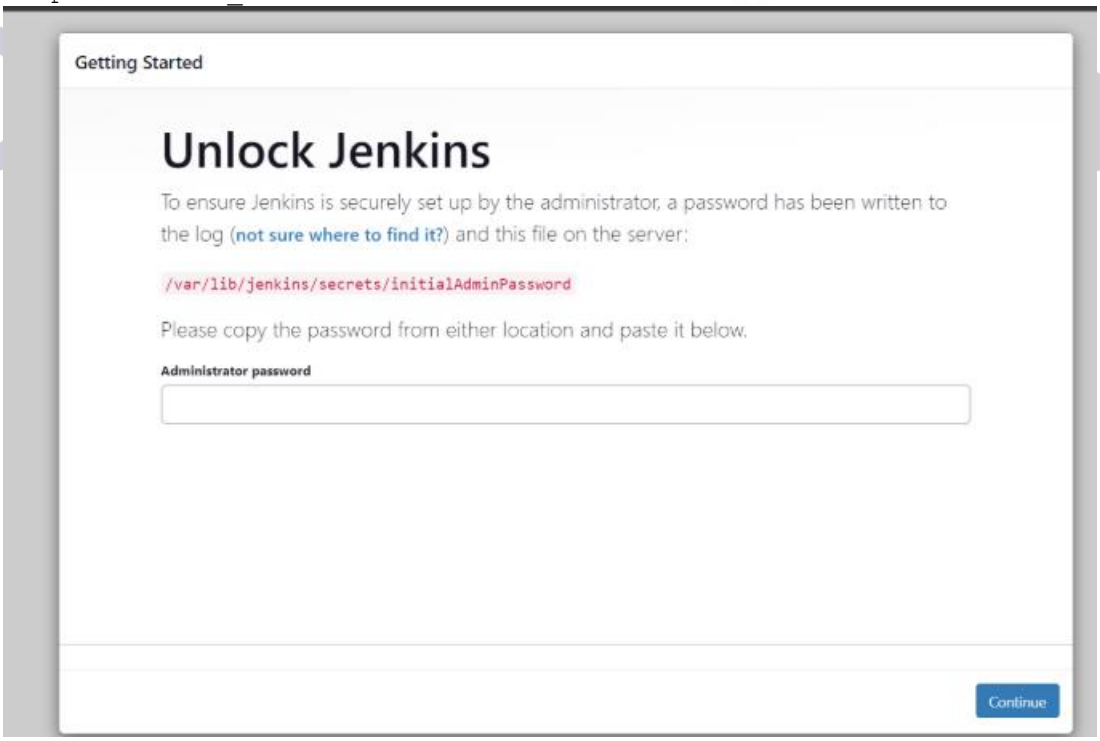
```
# Restart Docker service to apply changes
sudo systemctl restart docker

# Verify installation
docker -version
# Run SonarQube container in detached mode with port mapping
#docker run -d --name sonar -p 9000:9000 sonarqube:lts-community
ubuntu@ip-10-0-1-103:~/hotstar-kubernetes/scripts$ docker run -d --name sonar -p 90
00:9000 sonarqube:lts-community
Unable to find image 'sonarqube:lts-community' locally
lts-community: Pulling from library/sonarqube
9cb31e2e37ea: Pull complete
13876c96bdc5: Pull complete
25fdfc9faee8: Pull complete
b682cc54ed35: Pull complete
4615ed3a3407: Pull complete
c2e1e3bdd7bc: Extracting  150.4MB/300.4MB
b2dce0ce5cad: Download complete
4f4fb700ef54: Download complete
```

```
ubuntu@ip-10-0-1-103:~/hotstar-kubernetes/scripts$ trivy -v
Version: 0.60.0
```

```
Terraform v1.11.2
on linux_amd64
```

- Access Jenkins in your browser:

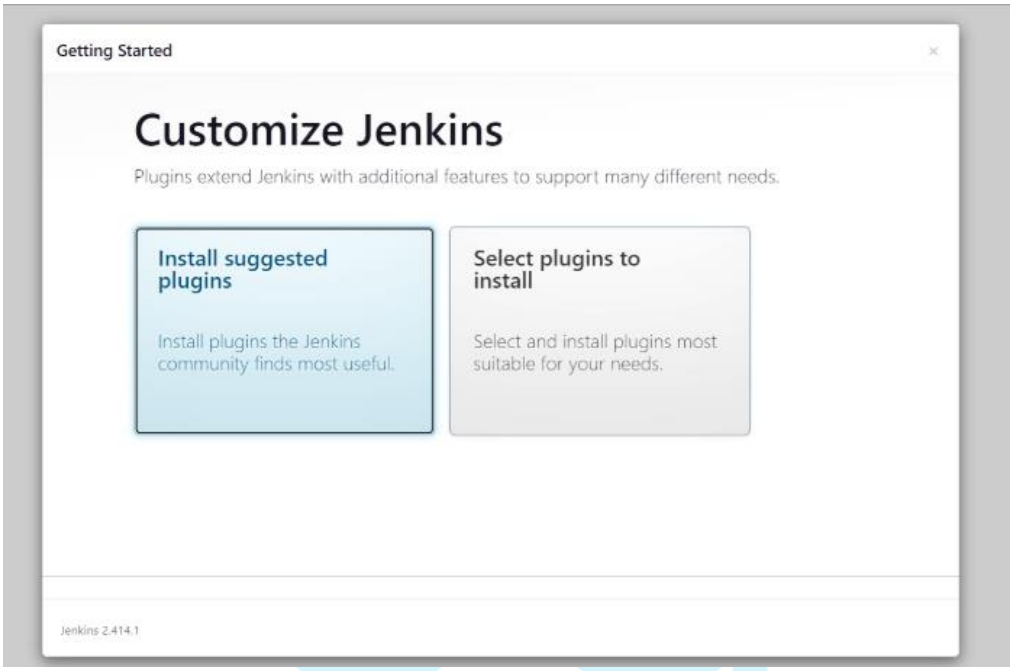
  `http://<PUBLIC_IP>:8080`



- Unlock Jenkins using an administrative password and install the suggested plugins. Retrieve the initial admin password:

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

Unlock Jenkins using an administrative password and install the suggested plugins.



- Create a user click on save and continue.
- Jenkins Getting Started Screen.
- Follow the setup wizard and install recommended plugins.

Install Plugins like JDK, SonarQube Scanner, NodeJs, OWASP Dependency Check

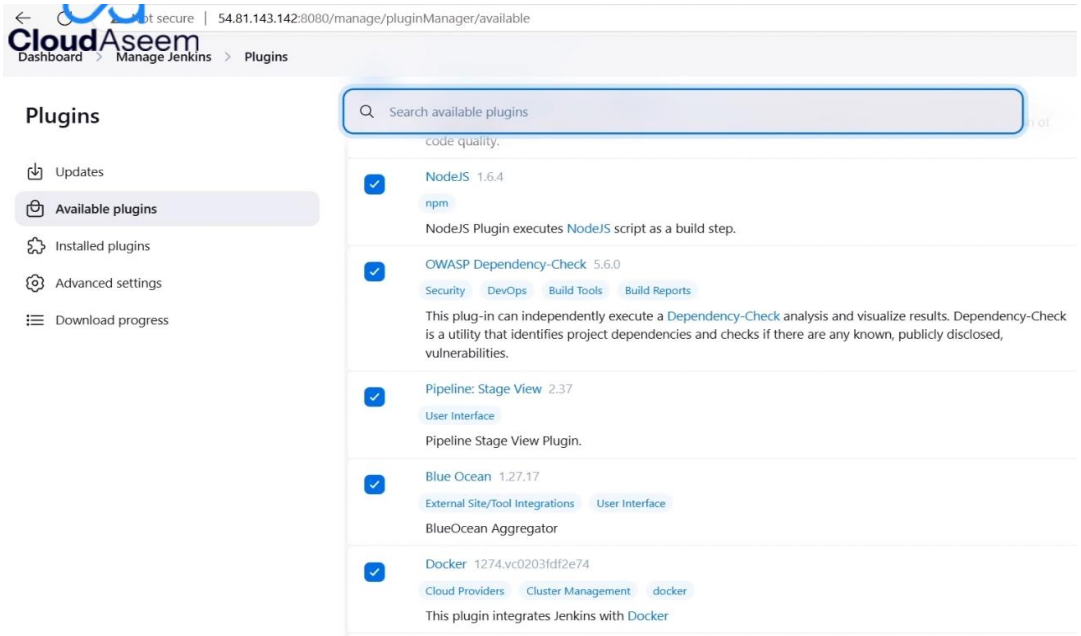
Goto Manage Jenkins →Plugins → Available Plugins →

Install below plugins

1. Eclipse Temurin Installer (Install without restart)
2. SonarQube Scanner (Install without restart)
3. NodeJs Plugin (Install Without restart) – 16.20.2
4. OWASP Dependency Check Plugins
5. Stage view
6. jdk

Docker plugin

7. Docker
8. Docker Commons
9. Docker Pipeline
10. Docker API
11. docker-build-step



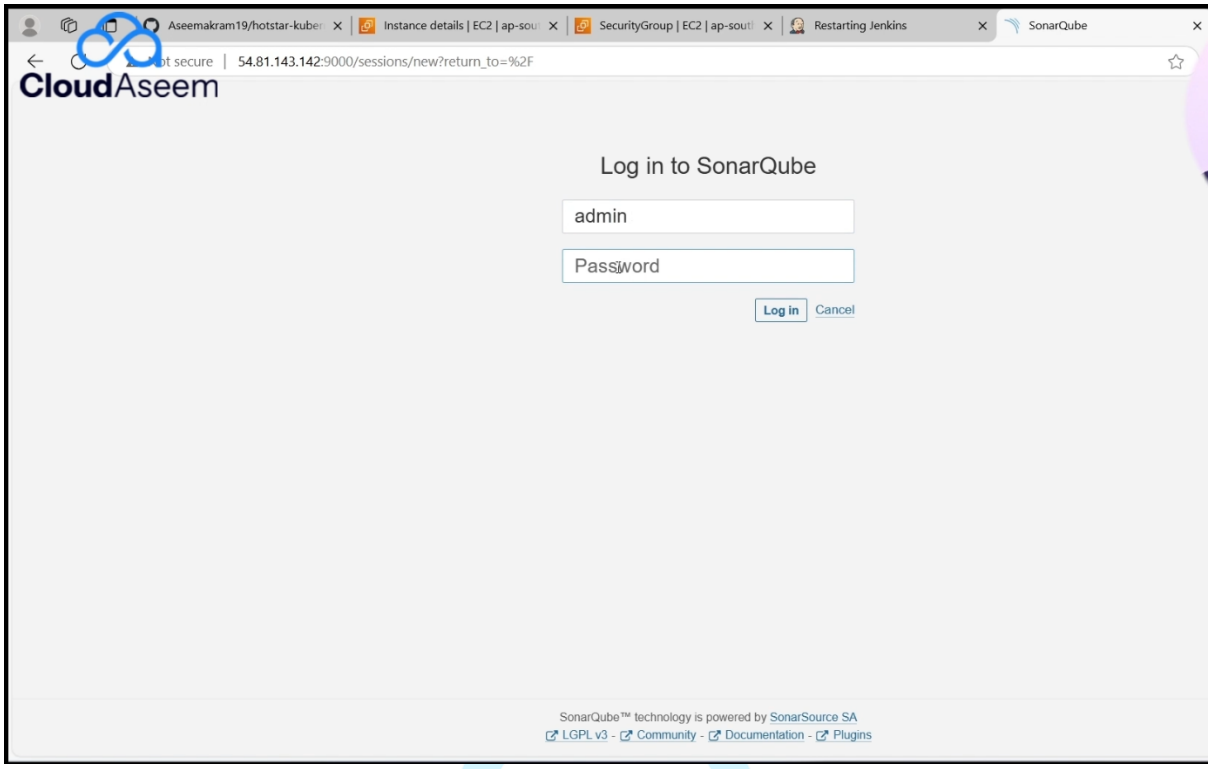
- **Setup SonarQube Server**

we create a sonarqube container

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

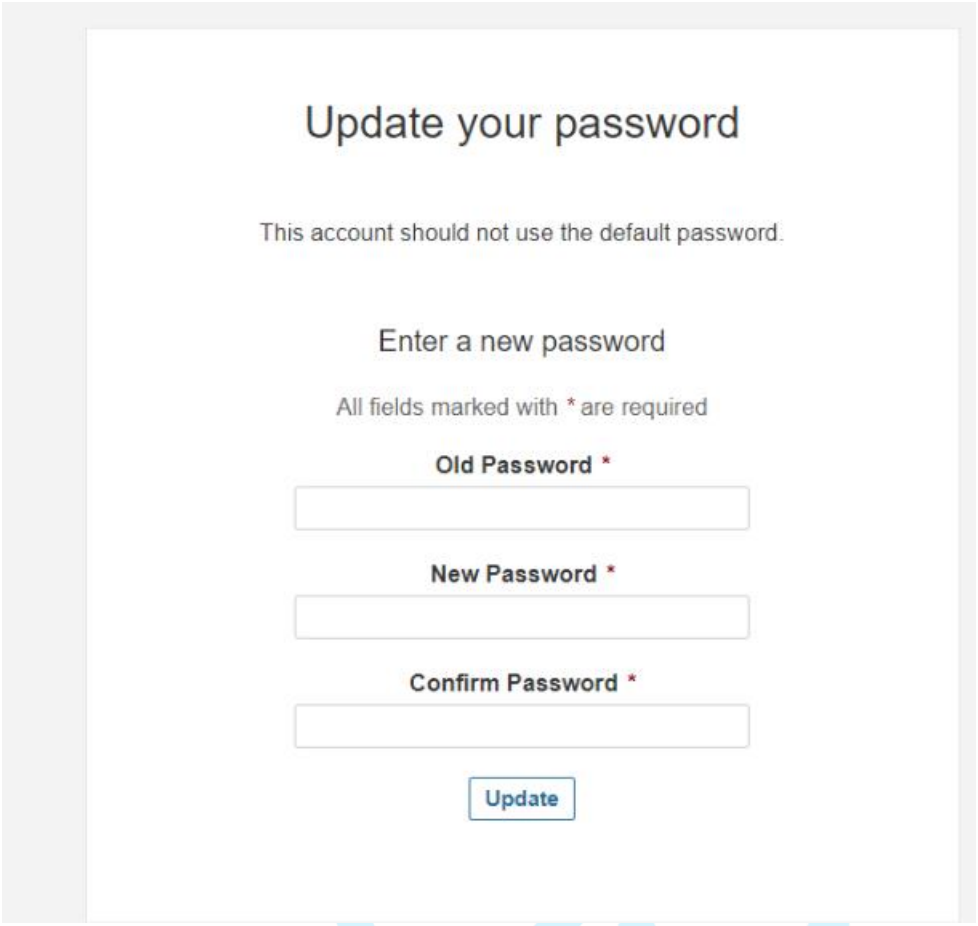
```
ubuntu@ip-10-0-1-103:~/hotstar-kubernetes/scripts$ docker run -d --name sonar -p 90
00:9000 sonarqube:lts-community
Unable to find image 'sonarqube:lts-community' locally
lts-community: Pulling from library/sonarqube
9cb31e2e37ea: Pull complete
13876c96bdc5: Pull complete
25fdfc9faee8: Pull complete
b682cc54ed35: Pull complete
4615ed3a3407: Pull complete
c2e1e3bdd7bc: Pull complete
b2dce0ce5cad: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:95b1826b68e606678882148e28c01c70dc0efe464a0d0c568570b17eedf1a9f9
Status: Downloaded newer image for sonarqube:lts-community
73467ceddea5dbb6cf2970ca1299b617974ec4276a33d9f95f796b73155589b6
ubuntu@ip-10-0-1-103:~/hotstar-kubernetes/scripts$
```

Now our Sonarqube is up and running

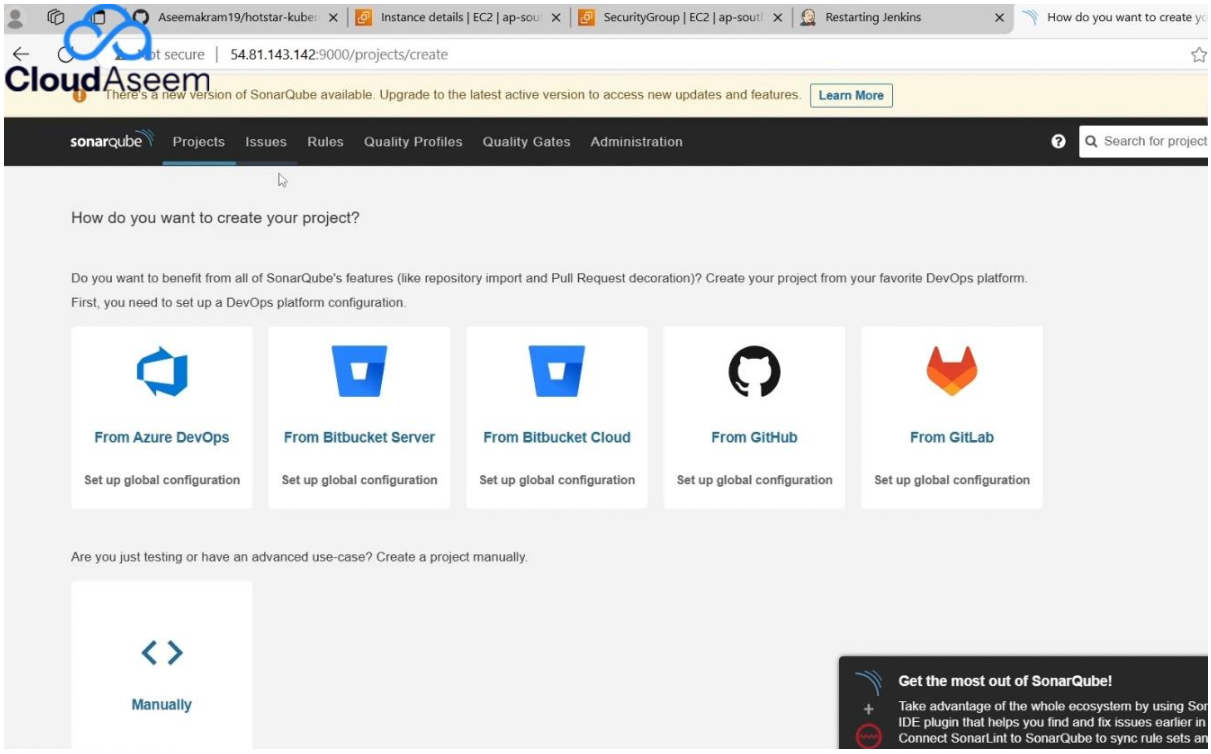


Enter username and password, click on login and change password

username admin
password admin

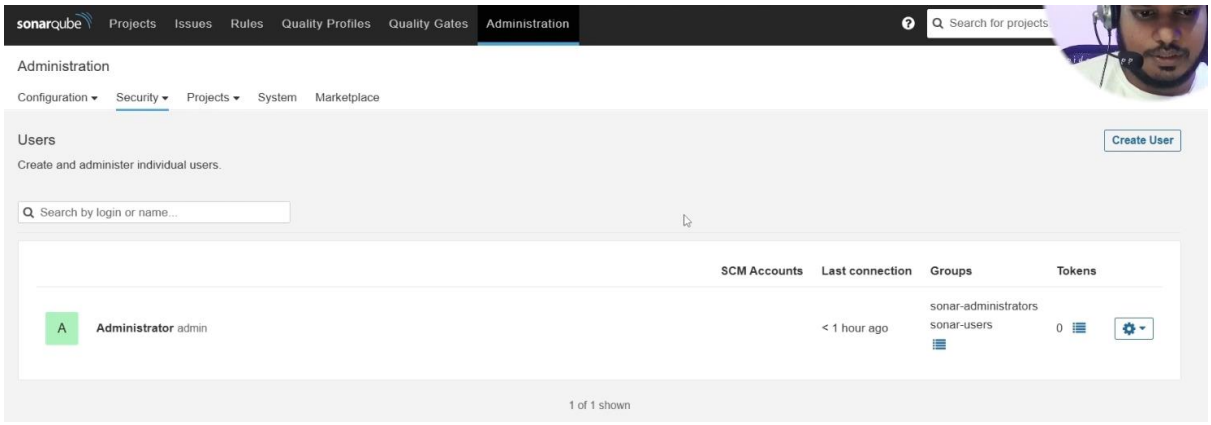


Update New password, This is Sonar Dashboard.

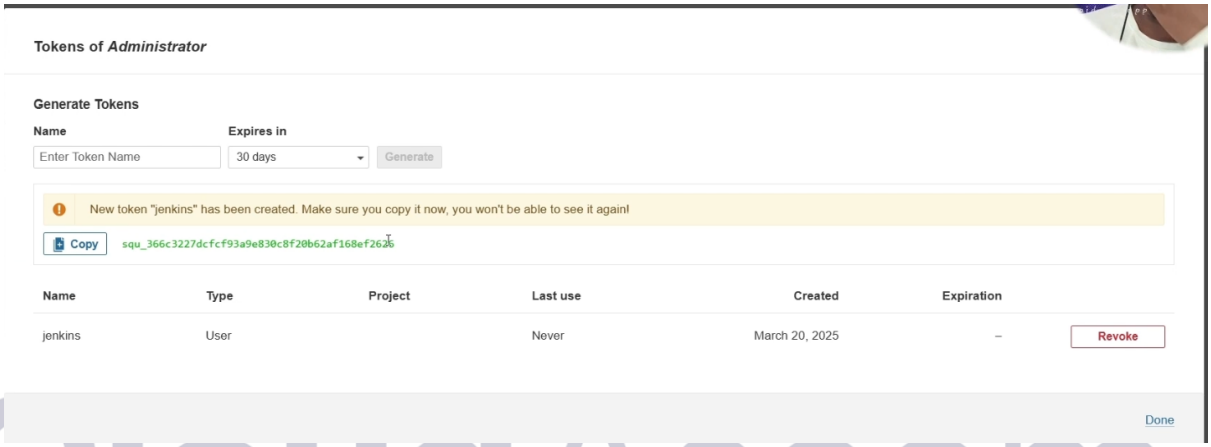


B. - Create Sonar token in order to connect with Jenkins

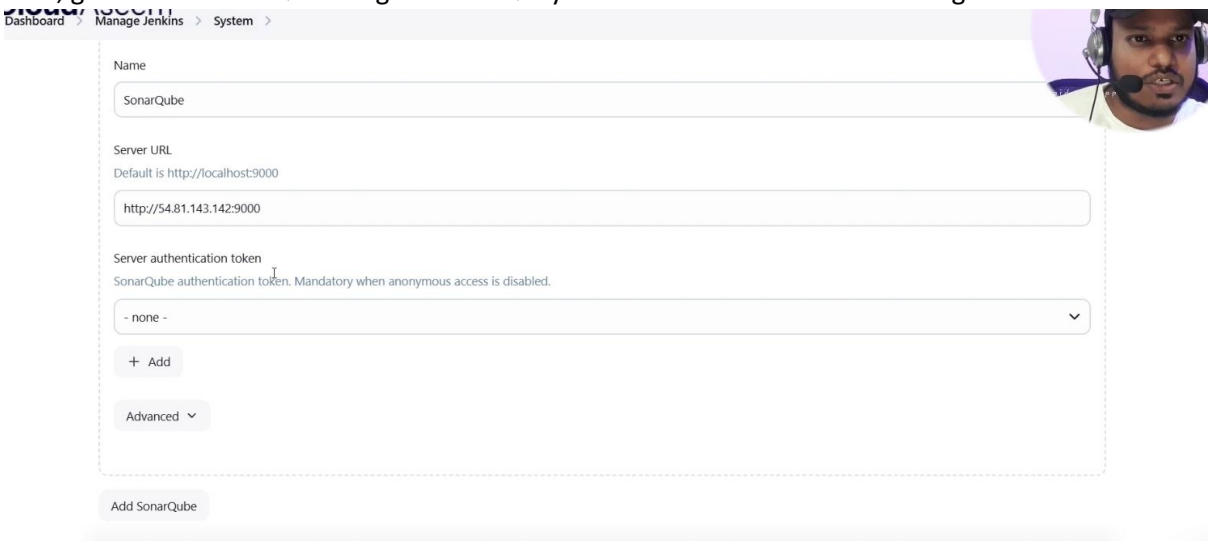
Click on Administration → Security → Users → Click on Tokens and Update Token → Give it a name → and click on Generate Token



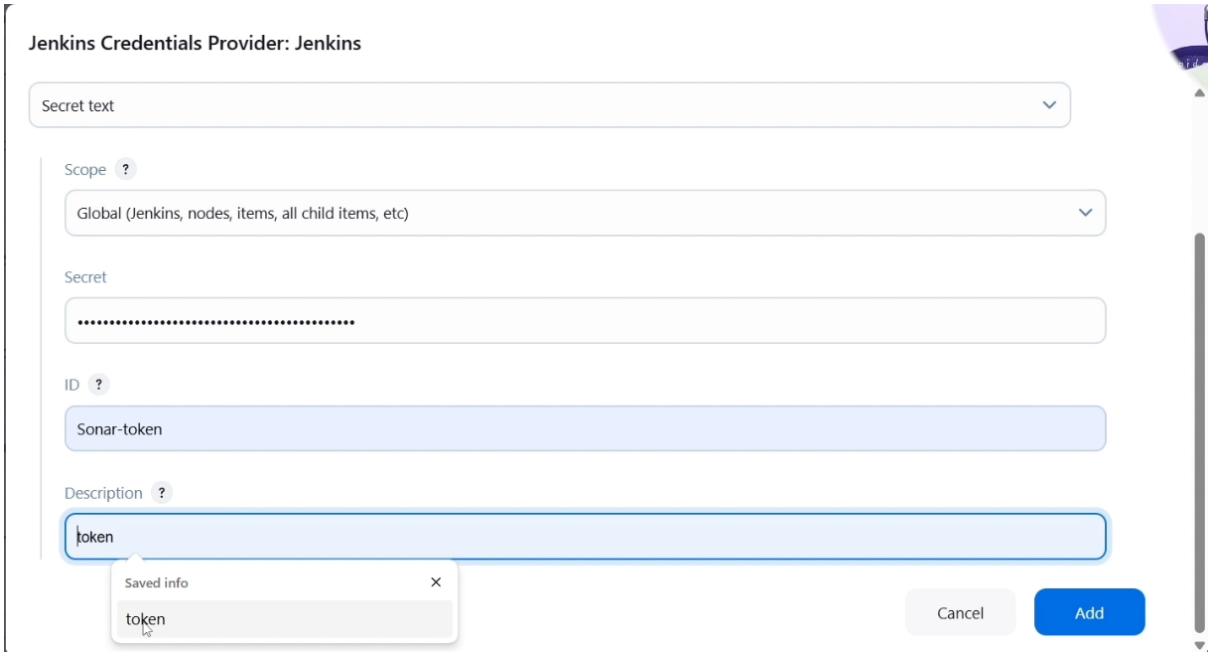
Create a token with a name and generate



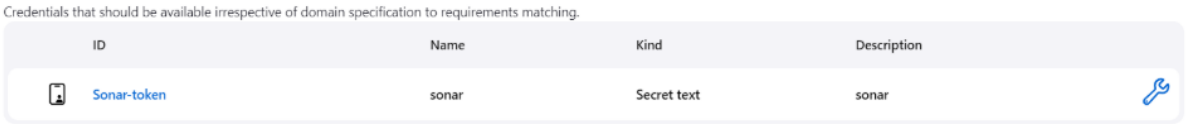
Now, go to Dashboard → Manage Jenkins → System and Add like the below image.



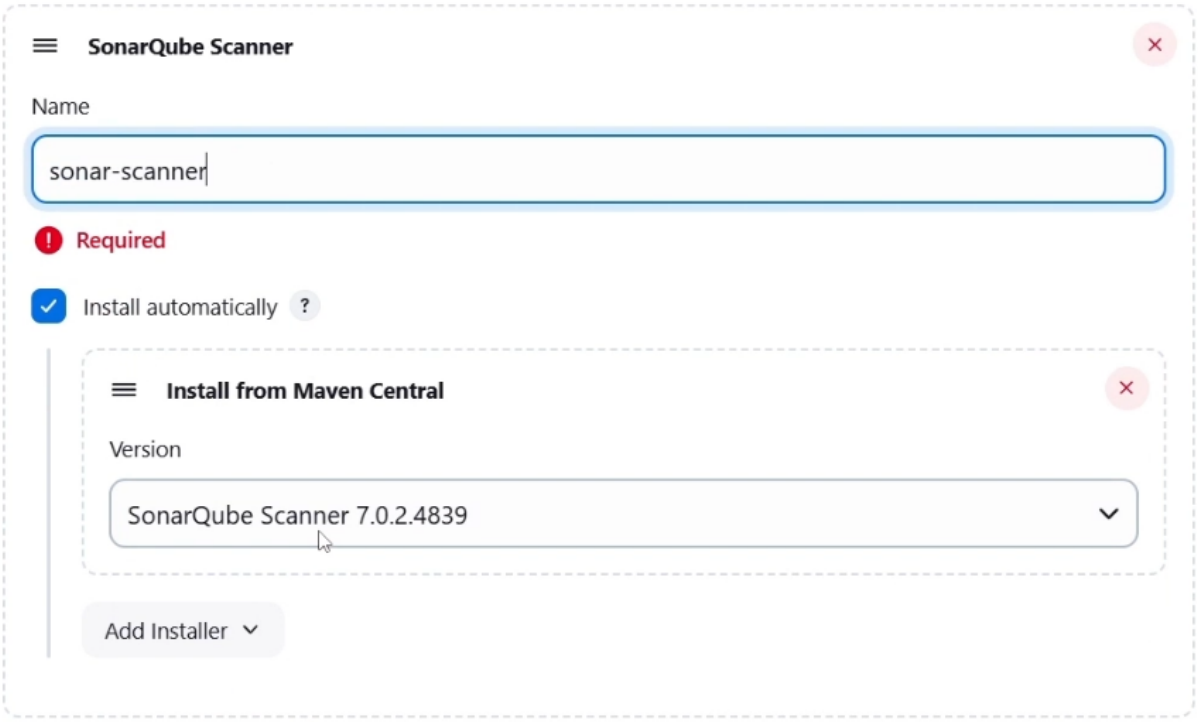
Add Credentials → Add Secret Text. It should look like this



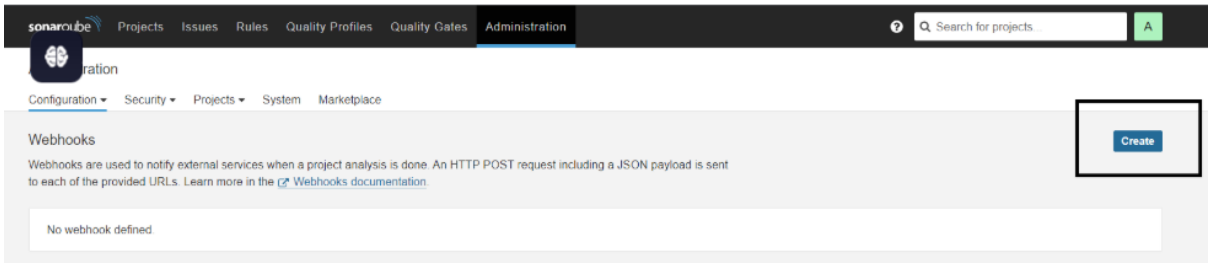
You will this page once you click on create



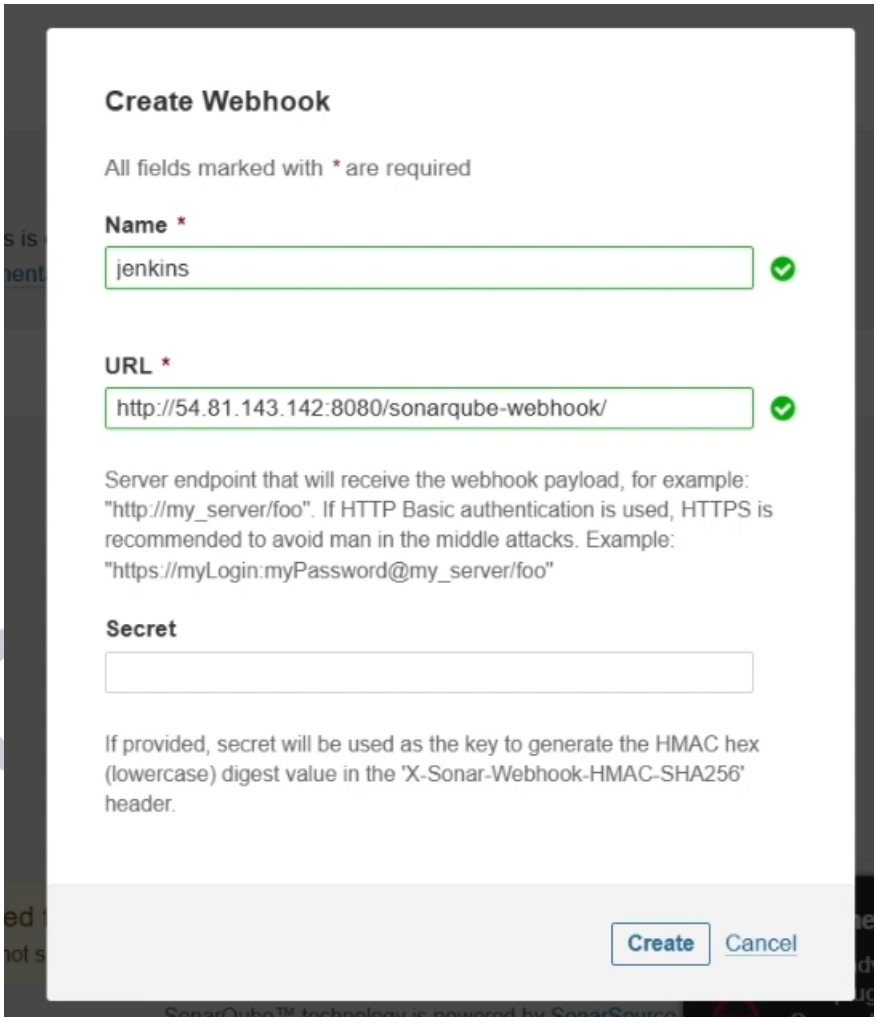
The Configure System option is used in Jenkins to configure different server
Global Tool Configuration is used to configure different tools that we install using Plugins
We will install a sonar scanner in the tools.
Manage Jenkins –> Tools –> SonarQube Scanner



In the Sonarqube Dashboard add a quality gate also
Administration–> Configuration–>Webhooks



- webhook - http://34.228.235.18:8080/sonarqube-webhook/



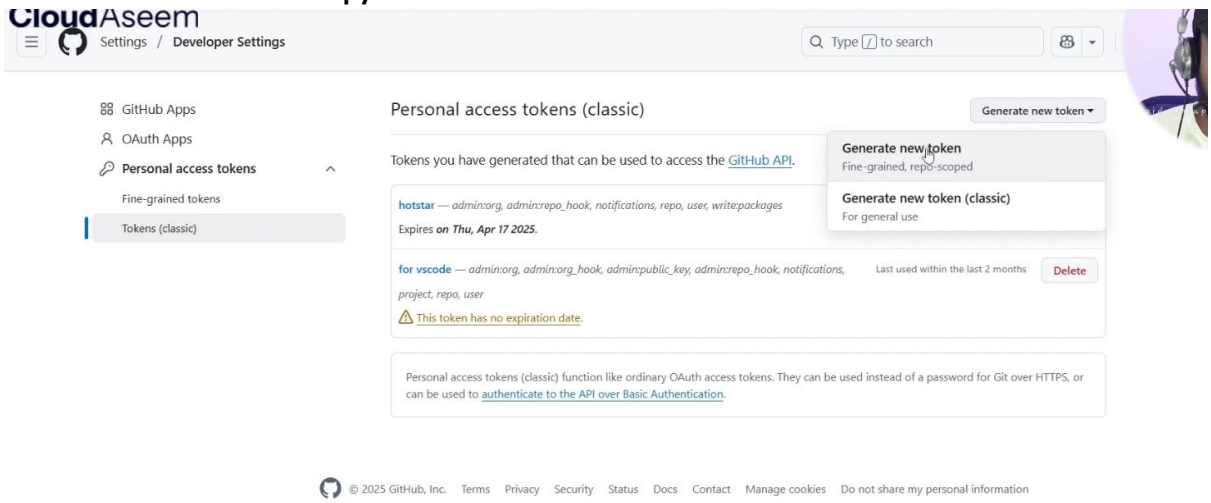
Setup Jenkins GitHub token inorder to connect with Private Registry -

Generate Classic GitHub Token

Go to GitHub → Settings → Developer Settings → Personal Access Tokens.
Click Generate new token (classic).
Set Expiration (or No Expiration if required).
Set Scopes:

- **repo → Full control of private repositories.**
- **admin:repo_hook → Manage repository webhooks.**
- **workflow → Required for GitHub Actions (optional).**

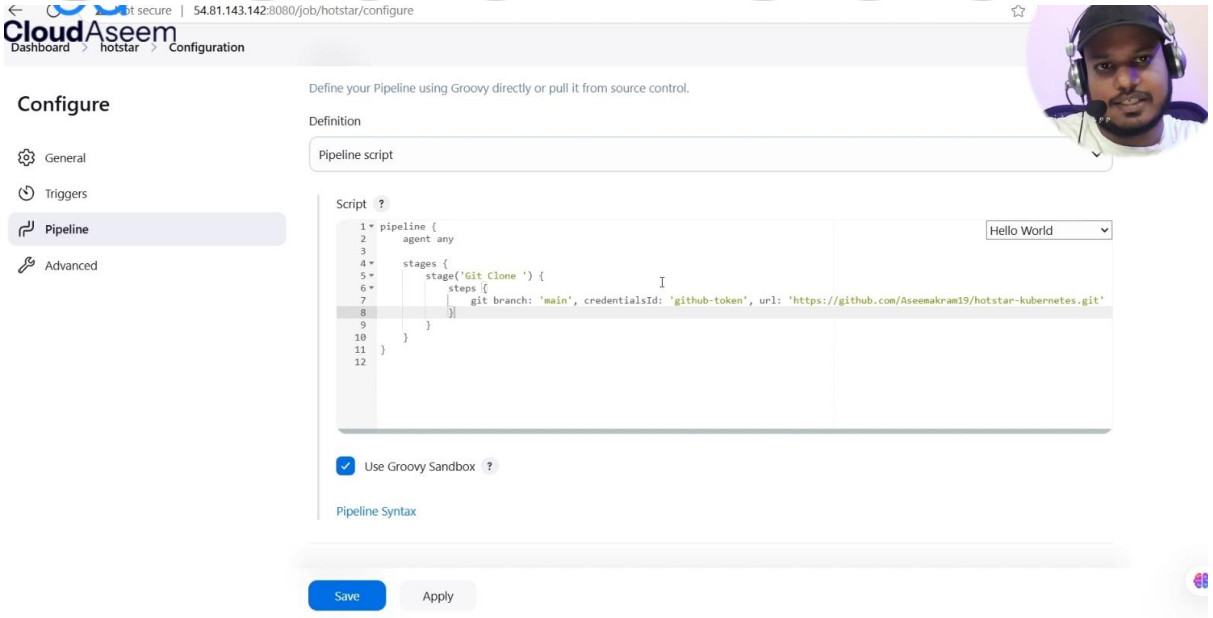
Click Generate token and copy the token.



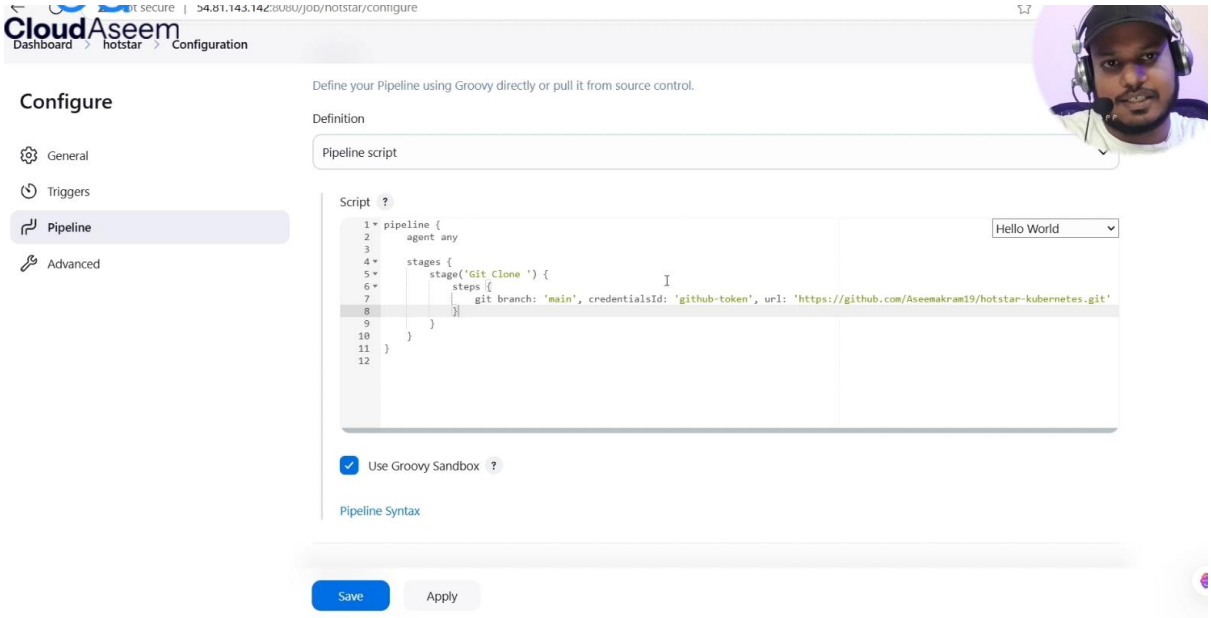
Configure Jenkins with GitHub Token

Add Credentials in Jenkins

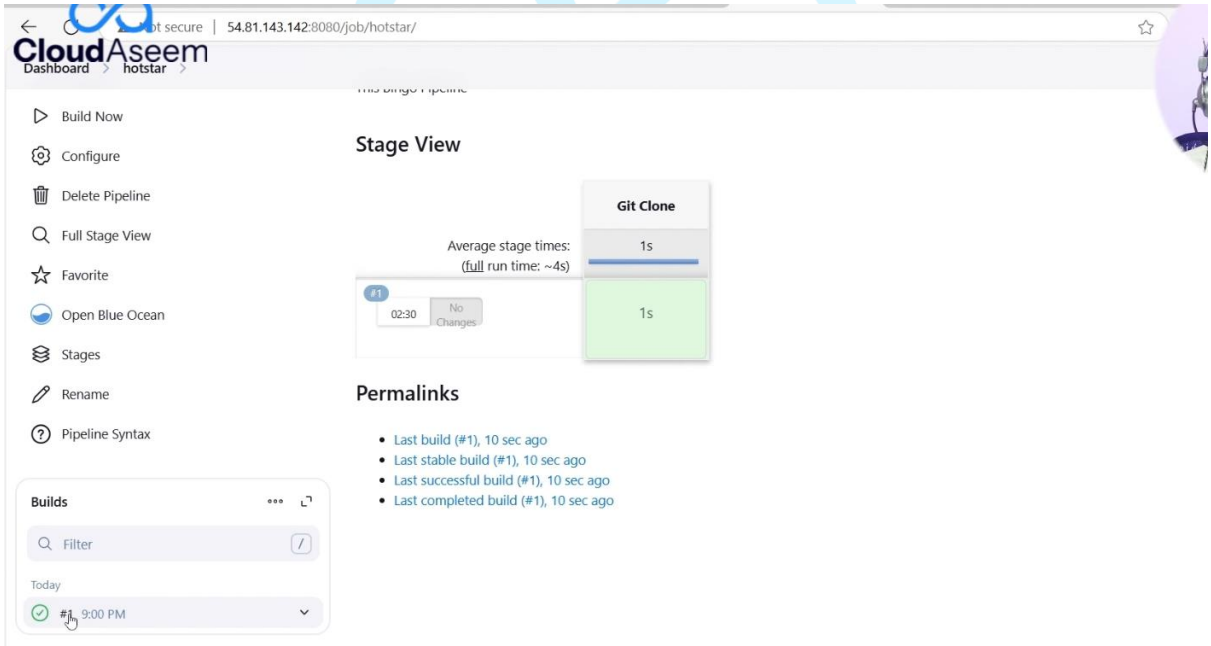
1. Go to **Jenkins Dashboard → Manage Jenkins → Manage Credentials**.
2. Click **Global Credentials (Unrestricted) → Add Credentials**.
3. Select:
   - **Kind**: Username and password
   - **Username**: Your GitHub username
   - **Password**: Paste your **GitHub Token**
   - **ID**: github-token
   - **Description**: GitHub Classic Token
4. Click **Save**.



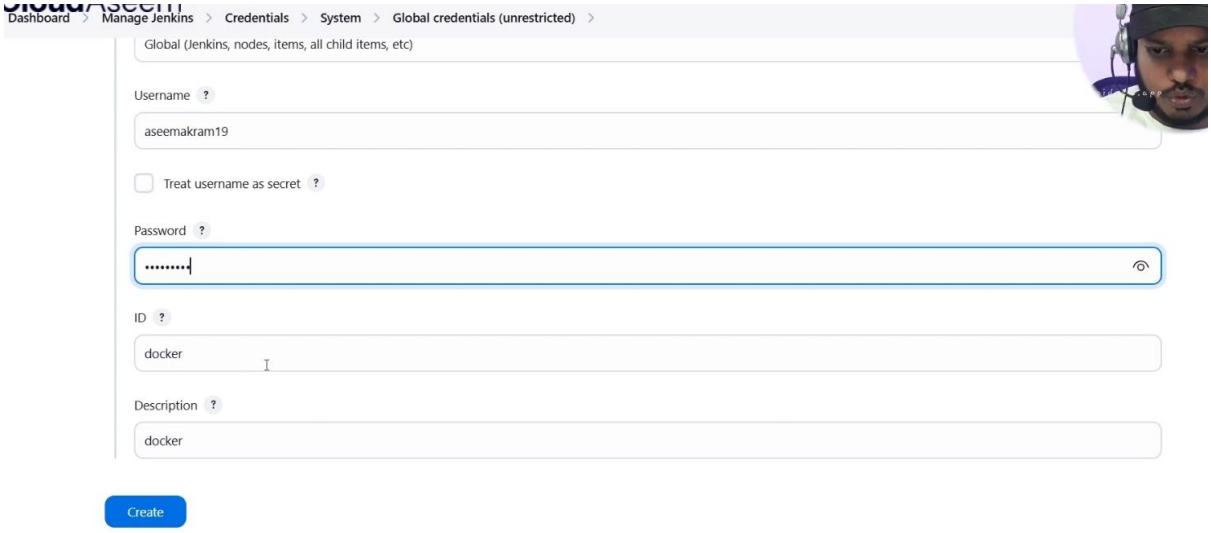
Create Job for Hotstar
Let’s add a pipeline , to test the Github Clone stage of Private Registry



Apply and Save and click on Build



Add docker Credentials



Create a Gmail SMTP App Password

An **App Password** is a 16-character password that allows third-party applications (like Jenkins) to send emails using **Gmail SMTP** securely.

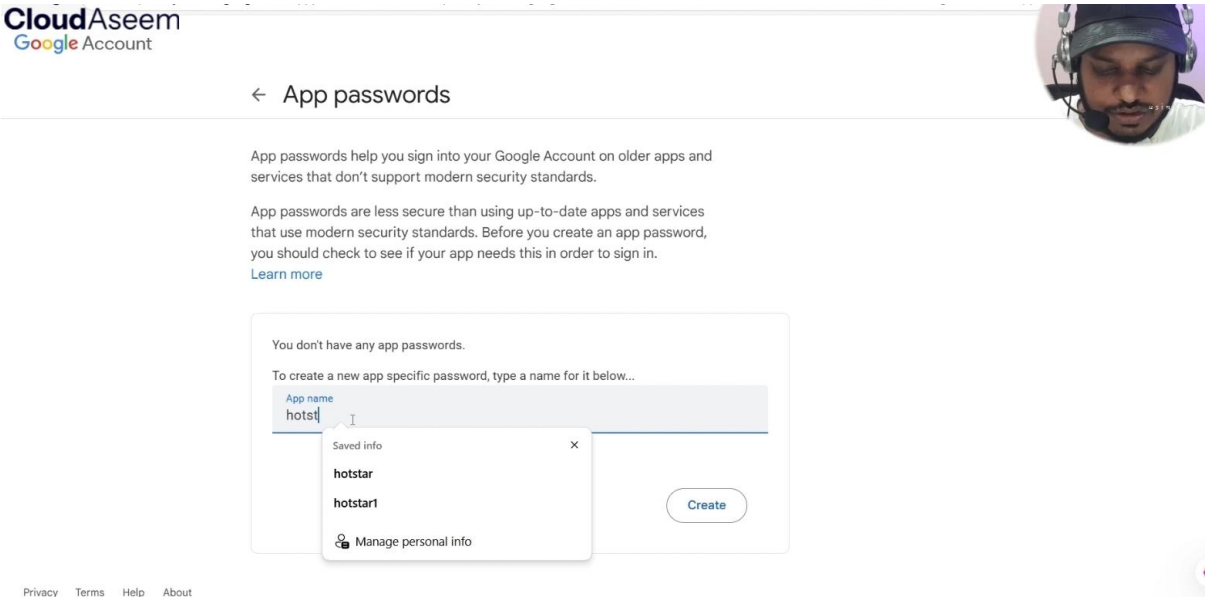
Step 1: Enable 2-Step Verification

Before generating an App Password, you **must** enable **2-Step Verification** in your Google Account.

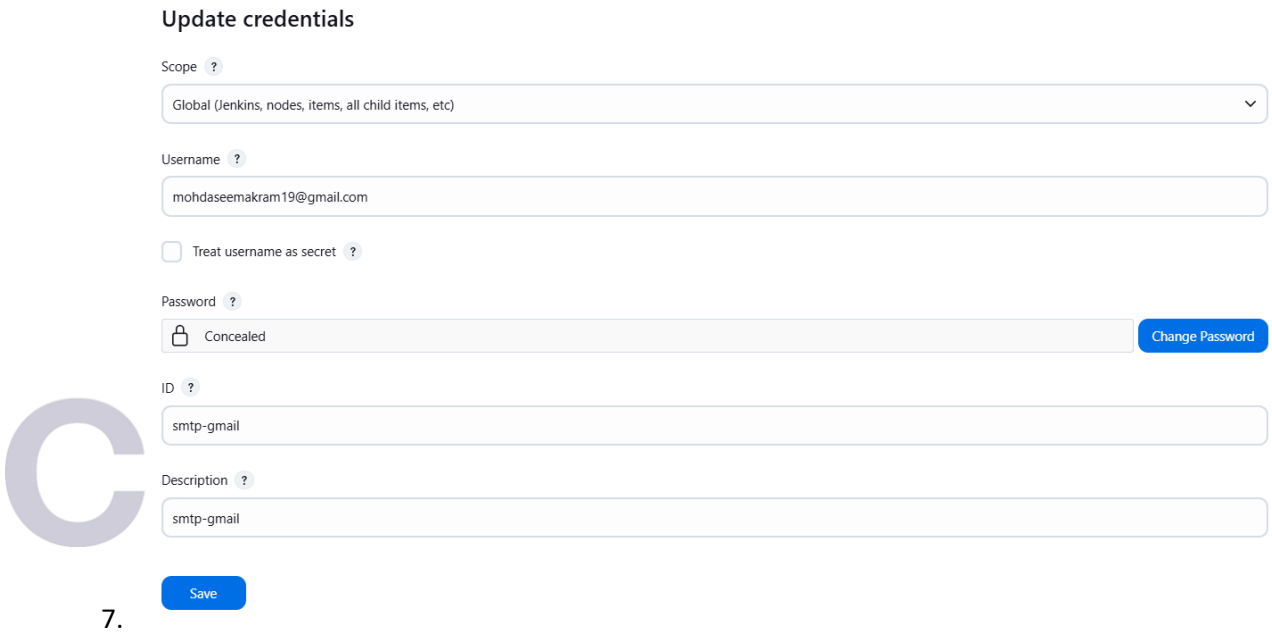
1. **Go to Google Account Security**:
   Google My Account
2. Scroll to **"Signing in to Google"**.
3. Click **"2-Step Verification"** → Click **"Get Started"**.
4. Follow the steps to set up **2-Step Verification** (via SMS or Authenticator App).

Step 2: Generate an App Password

1. **Go to App Passwords Page**:
   Google App Passwords
2. Sign in with your **Google Account**.
3. Under **"Select app"**, choose **"Mail"**.
4. Under **"Select device"**, choose **"Other (Custom Name)"** and enter **"Jenkins SMTP"**.
5. Click **"Generate"**.
6. **Copy the 16-character App Password** (e.g., abcd efgh ijkl mnop).



Add credentials as Username and password in jenkins



7.

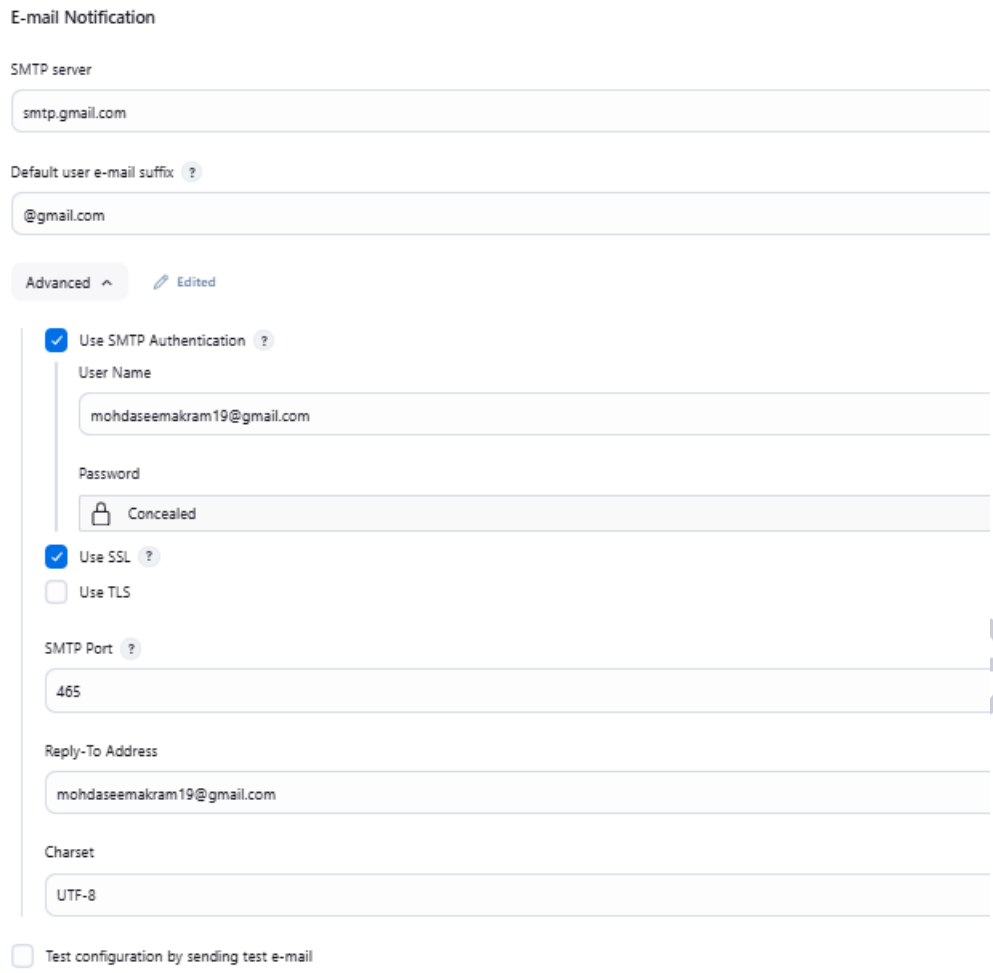
Step 3: Configure Gmail SMTP in Jenkins

1. **Go to Jenkins Dashboard → Manage Jenkins → Configure System**.
2. Scroll to **"E-mail Notification"**.
3. Set the following:
   - **SMTP Server**: smtp.gmail.com
   - **Use SMTP Authentication**: ✅ Checked
   - **User Name**: Your Gmail ID (your-email@gmail.com)
   - **Password**: Paste the **App Password**
   - **SMTP Port**: 587
   - **Use TLS**: ✅ Checked
4. Click **Save**.

Email Extension Plugin

xsuc kxeb xcvk xqkf

1. Basic Email Notification
SMTP Server: smtp.gmail.com
Email Suffix: @gmail.com (default user email domain)
SMTP Authentication: Enabled
Username: Your Gmail address
Password: Your Gmail password or App Password (for 2-factor authentication)
Use TLS: Checked
SMTP Port: 587
Reply-To Address: Your email address
Charset: UTF-8



2. Extended Email Notification
SMTP Server: smtp.gmail.com
SMTP Port: 465 (for tls)
Use SSL: Checked

Credentials: Select from the

Extended E-mail Notification

SMTP server

smtp.gmail.com

SMTP Port

587

Advanced Edited

Default user e-mail suffix ?

@gmail.com

Advanced Edited

Default Content Type ?

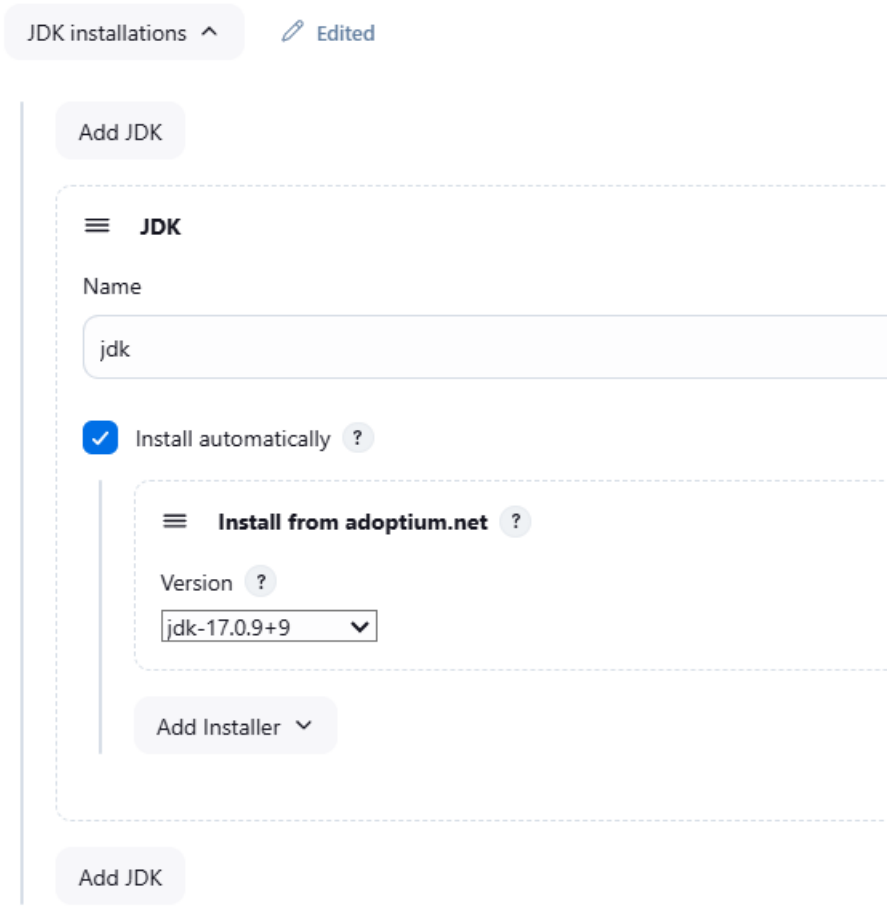
Plain Text (text/plain)

List ID ?

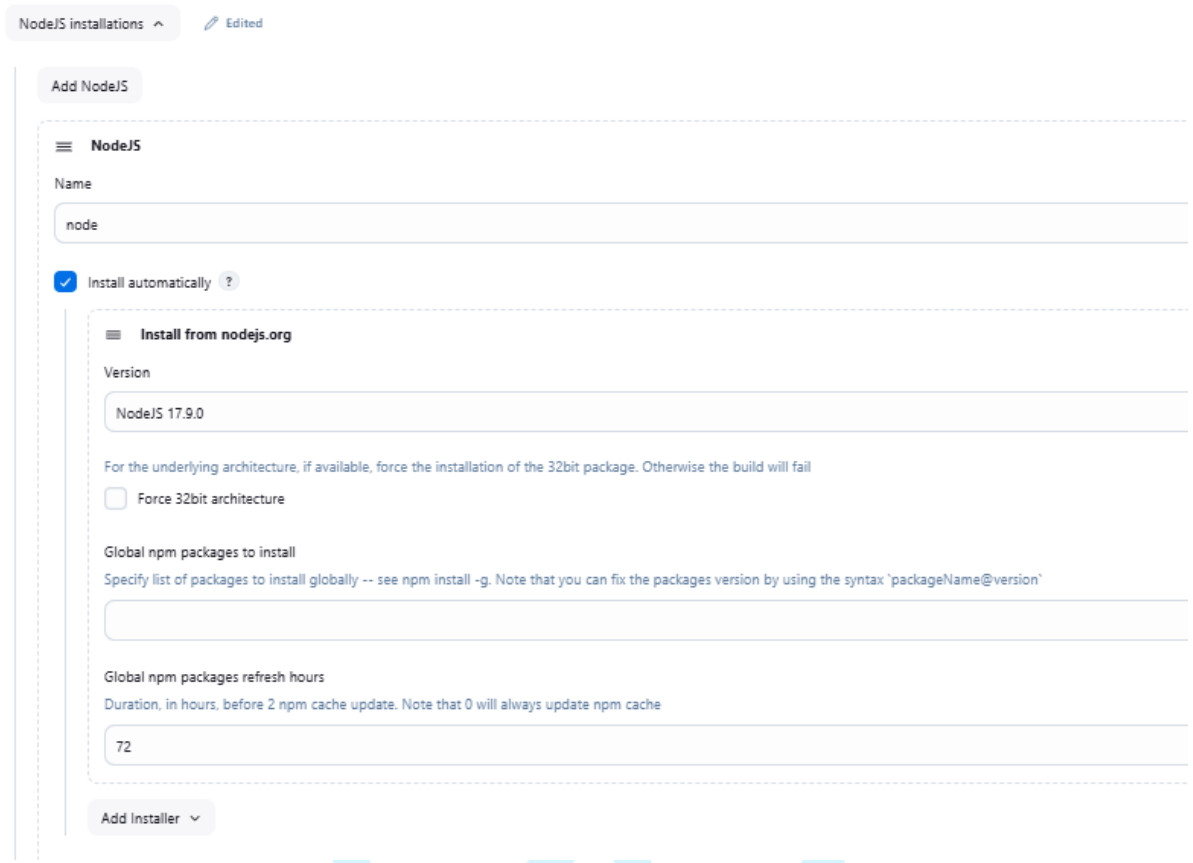
Add 'Precedence: bulk' E-mail Header ?

Default Recipients ?

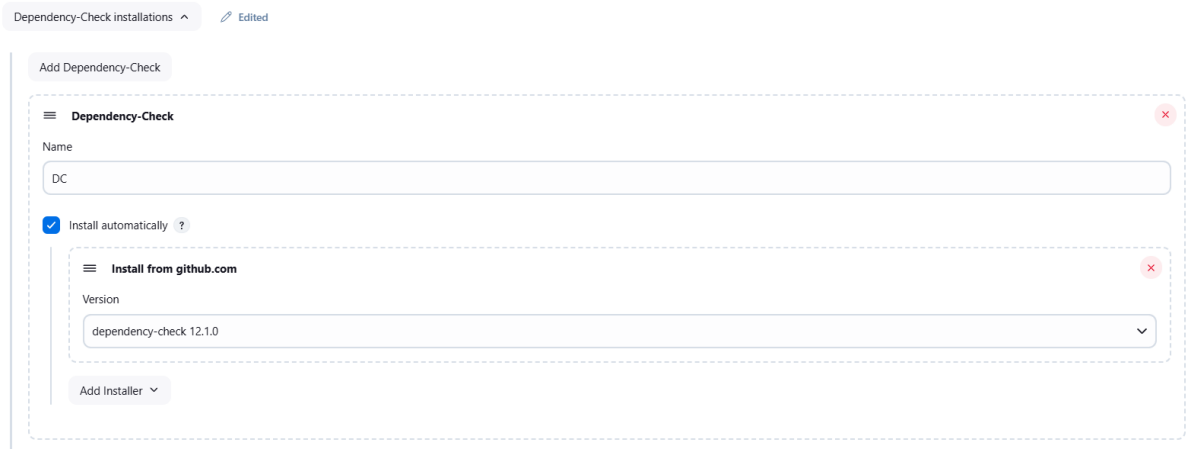
JDK installations



NodeJS installations



Dependency-Check installations



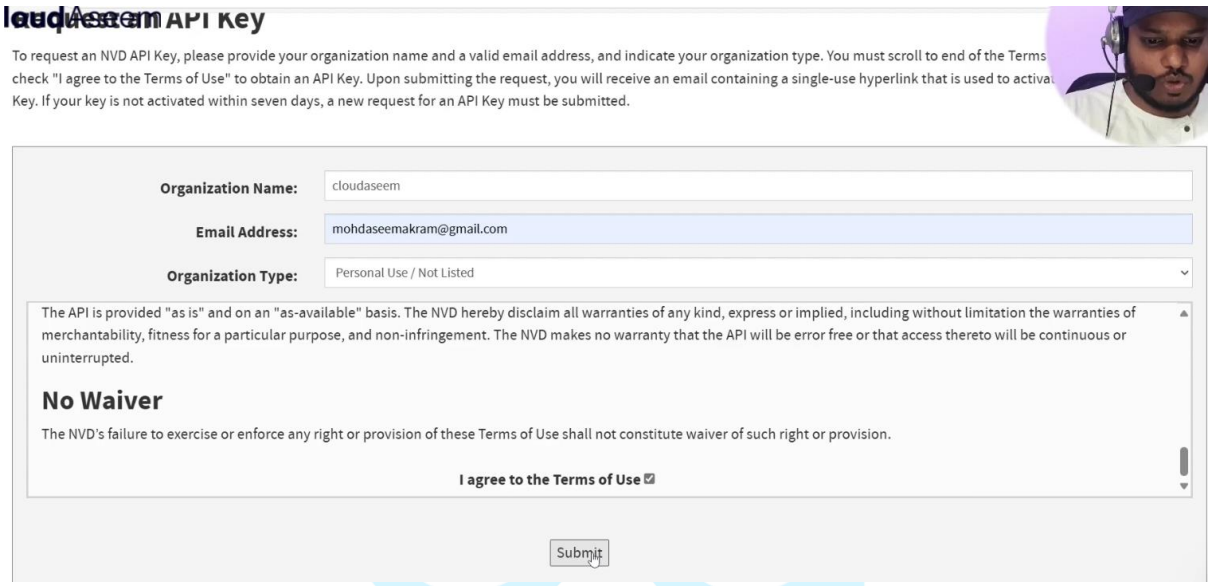
And last

Register for NVD API for Dependency Check

The **National Vulnerability Database (NVD) API** provides access to security vulnerabilities (CVEs) and is often used with tools like **OWASP Dependency-Check** to identify security risks in software dependencies.

Step 1: Create an NVD API Key

1. **Go to the NVD API Registration Page:**
    - Open: [NVD API Registration]
2. **Sign In or Create an Account:**
    - Click **Sign In** (or create an account if you don’t have one).
3. **Request an API Key:**
    - Provide your details and agree to the terms.
    - Click **Submit**.
4. **Receive API Key via Email:**
    - Once approved, you'll receive an API key.



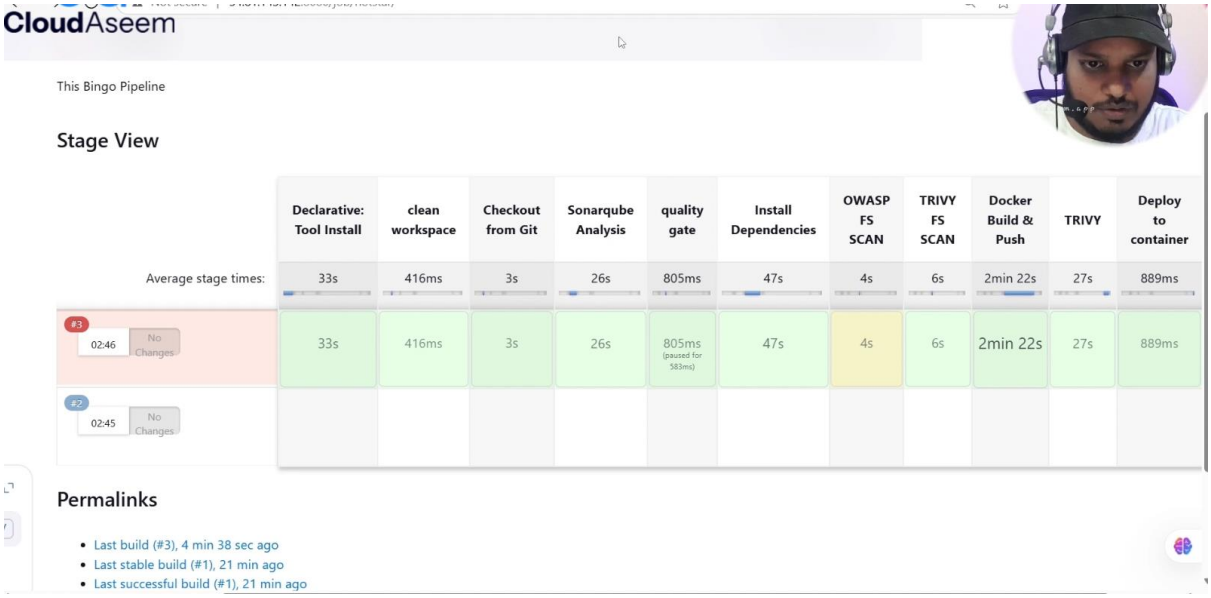
Let’s add a pipeline of Project

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

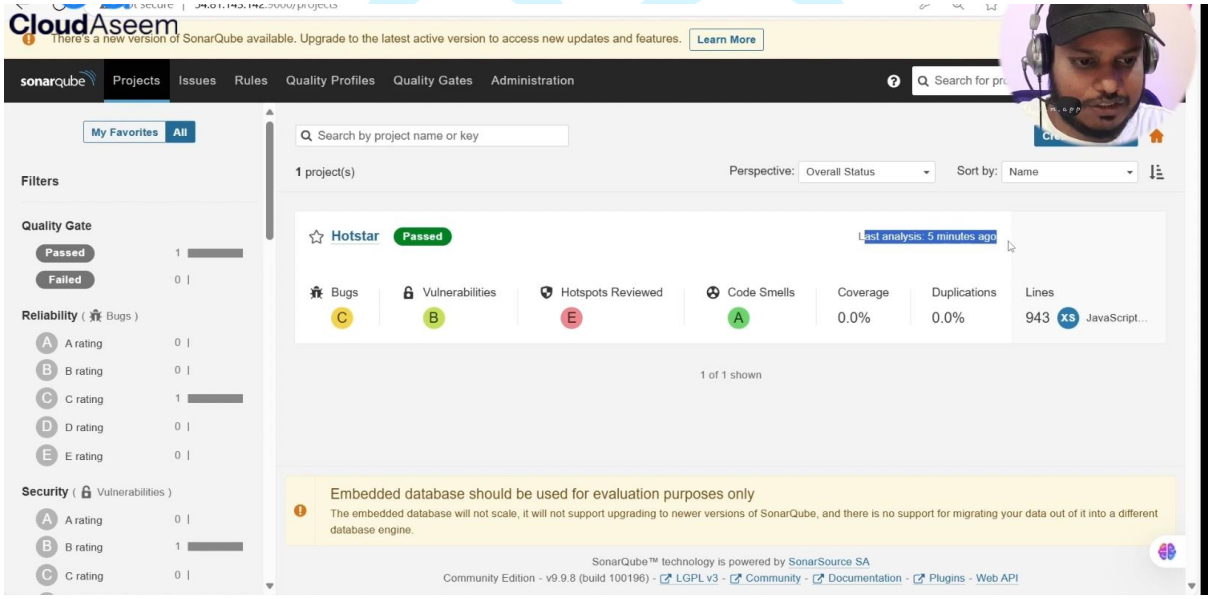
```
            git branch: 'main', credentialsId: 'github-token', url:
'https://github.com/Aseemakram19/hotstar-kubernetes.git'
          }
        }
        stage("Sonarqube Analysis "){
          steps{
            withSonarQubeEnv('SonarQube') {
              sh ''' $SCANNER_HOME/bin/sonar-scanner -Dsonar.projectName=Hotstar \
              -Dsonar.projectKey=Hotstar '''
            }
          }
        }
        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 --
nvdApiKey d7e8c629-7da9-4f96-8a4a-a45fd3f213ba', odcInstallation: 'DC'
            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 hotstar ."
                sh "docker tag hotstar aseemakram19/hotstar:latest "
                sh "docker push aseemakram19/hotstar:latest "
              }
            }
          }
        }
        stage("TRIVY"){
```

```
        steps{
            sh "trivy image aseemakram19/hotstar:latest > trivyimage.txt"
        }
      }
      stage('Deploy to container'){
        steps{
            sh 'docker run -d --name hotstar -p 3000:3000 aseemakram19/hotstar:latest'
        }
      }

   }
   post {
   always {
      script {
         def buildStatus = currentBuild.currentResult
         def buildUser = currentBuild.getBuildCauses('hudson.model.Cause$UserIdCause')[0]?.userId
?: 'Github User'

         emailext (
            subject: "Pipeline ${buildStatus}: ${env.JOB_NAME} #${env.BUILD_NUMBER}",
            body: """
               <p>This is a Jenkins HOTSTAR CICD pipeline status.</p>
               <p>Project: ${env.JOB_NAME}</p>
               <p>Build Number: ${env.BUILD_NUMBER}</p>
               <p>Build Status: ${buildStatus}</p>
               <p>Started by: ${buildUser}</p>
               <p>Build URL: <a href="${env.BUILD_URL}">${env.BUILD_URL}</a></p>
            """,
            to: 'mohdaseemakram19@gmail.com',
            from: 'mohdaseemakram19@gmail.com',
            replyTo: 'mohdaseemakram19@gmail.com',
            mimeType: 'text/html',
            attachmentsPattern: 'trivyfs.txt,trivyimage.txt'
         )
        }
     }

   }

}
```

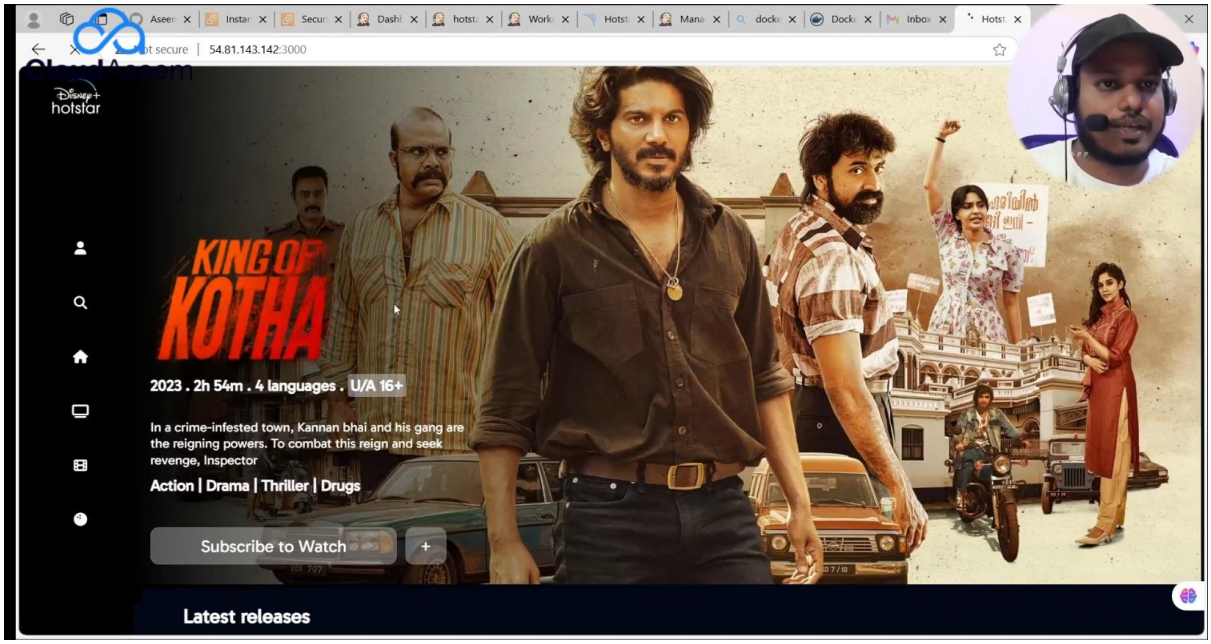
Apply and Save and click on Build
stage view



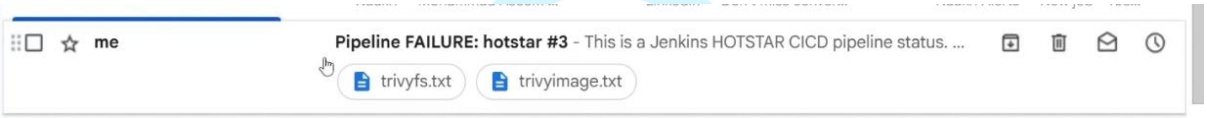
You can see the report has been generated and the status shows as passed. You can see that there are 943 lines it scanned. To see a detailed report, you can go to issues.
You will see that in status, a graph will also be generated and Vulnerabilities.



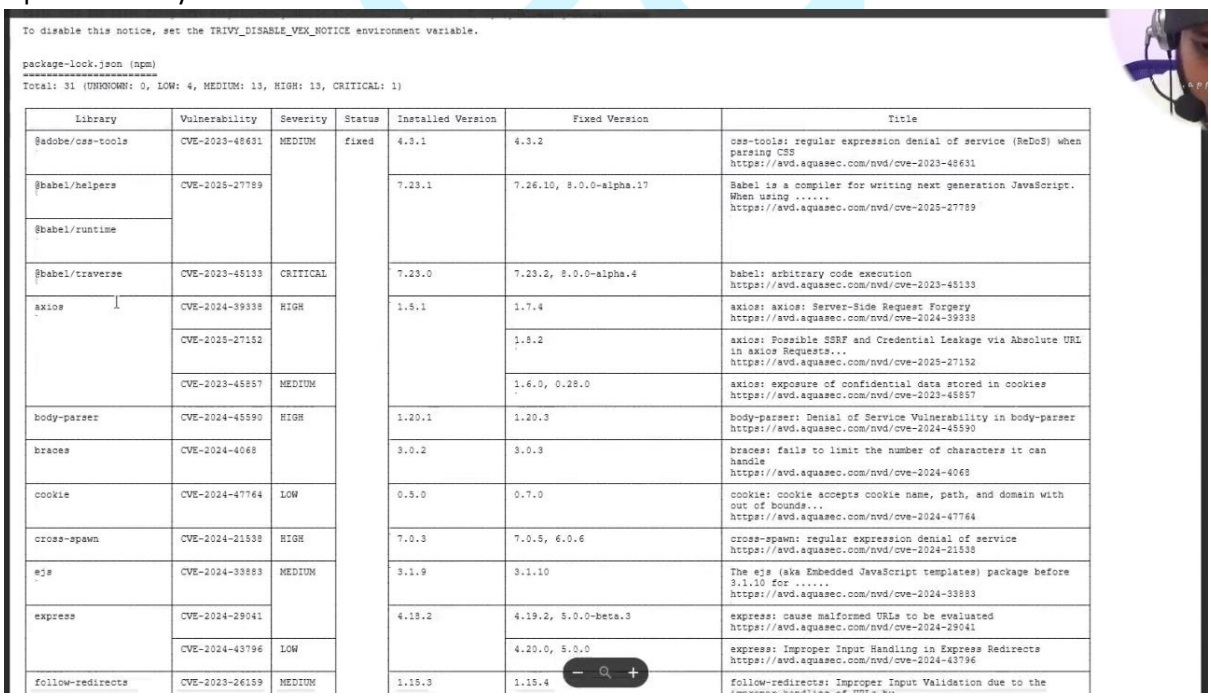
<public-ip:3000>
Our Applciation is live with this output



Email alert with Post build



Open Files trivy as attached to view vulnerabilities



To disable this notice, set the TRIVY_DISABLE_VEX_NOTICE environment variable.

package-lock.json (npm)

Total: 31 (UNKNOWN: 0, LOW: 4, MEDIUM: 13, HIGH: 13, CRITICAL: 1)

Library	Vulnerability	Severity	Status	Installed Version	Fixed Version	Title
@adobe/css-tools	CVE-2023-48631	MEDIUM	fixed	4.3.1	4.3.2	css-tools: regular expression denial of service (ReDoS) when parsing CSS https://avd.aquasec.com/nvd/cve-2023-48631
@babel/helpers	CVE-2025-27789			7.23.1	7.26.10, 8.0.0-alpha.17	Babel is a compiler for writing next generation JavaScript. When using https://avd.aquasec.com/nvd/cve-2025-27789
@babel/runtime						
@babel/traverse	CVE-2023-45133	CRITICAL		7.23.0	7.23.2, 8.0.0-alpha.4	babel: arbitrary code execution https://avd.aquasec.com/nvd/cve-2023-45133
axios	CVE-2024-39338	HIGH		1.5.1	1.7.4	axios: axios: Server-Side Request Forgery https://avd.aquasec.com/nvd/cve-2024-39338
	CVE-2025-27152				1.8.2	axios: Possible SSRF and Credential Leakage via Absolute URL in axios Requests... https://avd.aquasec.com/nvd/cve-2025-27152
	CVE-2023-45857	MEDIUM			1.6.0, 0.28.0	axios: exposure of confidential data stored in cookies https://avd.aquasec.com/nvd/cve-2023-45857
body-parser	CVE-2024-45590	HIGH		1.20.1	1.20.3	body-parser: Denial of Service Vulnerability in body-parser https://avd.aquasec.com/nvd/cve-2024-45590
braces	CVE-2024-4068			3.0.2	3.0.3	braces: fails to limit the number of characters it can handle https://avd.aquasec.com/nvd/cve-2024-4068
cookie	CVE-2024-47764	LOW		0.5.0	0.7.0	cookie: cookie accepts cookie name, path, and domain with out of bounds... https://avd.aquasec.com/nvd/cve-2024-47764
cross-spawn	CVE-2024-21538	HIGH		7.0.3	7.0.5, 6.0.6	cross-spawn: regular expression denial of service https://avd.aquasec.com/nvd/cve-2024-21538
ejs	CVE-2024-33883	MEDIUM		3.1.9	3.1.10	The ejs (aka Embedded JavaScript templates) package before 3.1.10 for https://avd.aquasec.com/nvd/cve-2024-33883
express	CVE-2024-29041			4.18.2	4.19.2, 5.0.0-beta.3	express: cause malformed URLs to be evaluated https://avd.aquasec.com/nvd/cve-2024-29041
	CVE-2024-43796	LOW			4.20.0, 5.0.0	express: Improper Input Handling in Express Redirects https://avd.aquasec.com/nvd/cve-2024-43796
follow-redirects	CVE-2023-26159	MEDIUM		1.15.3	1.15.4	follow-redirects: Improper Input Validation due to the

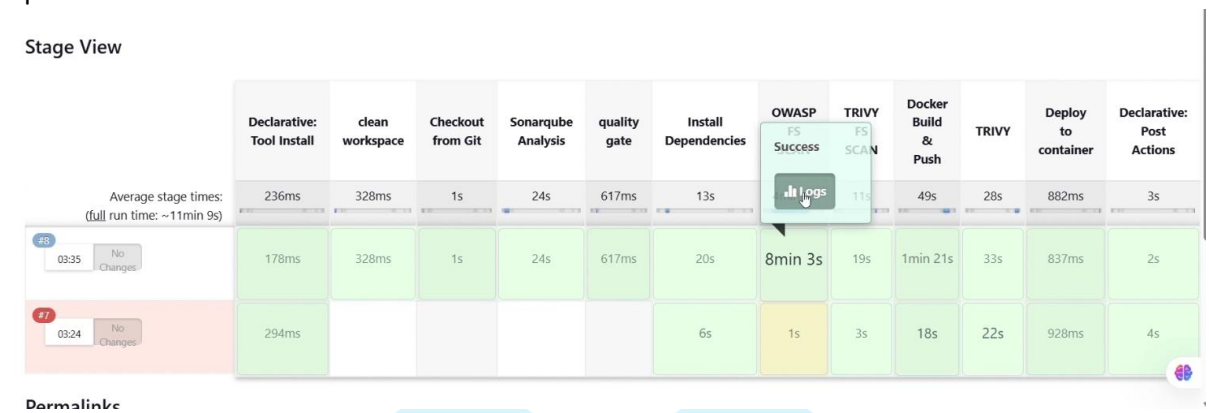
Lets fix the Dependency DP stage now

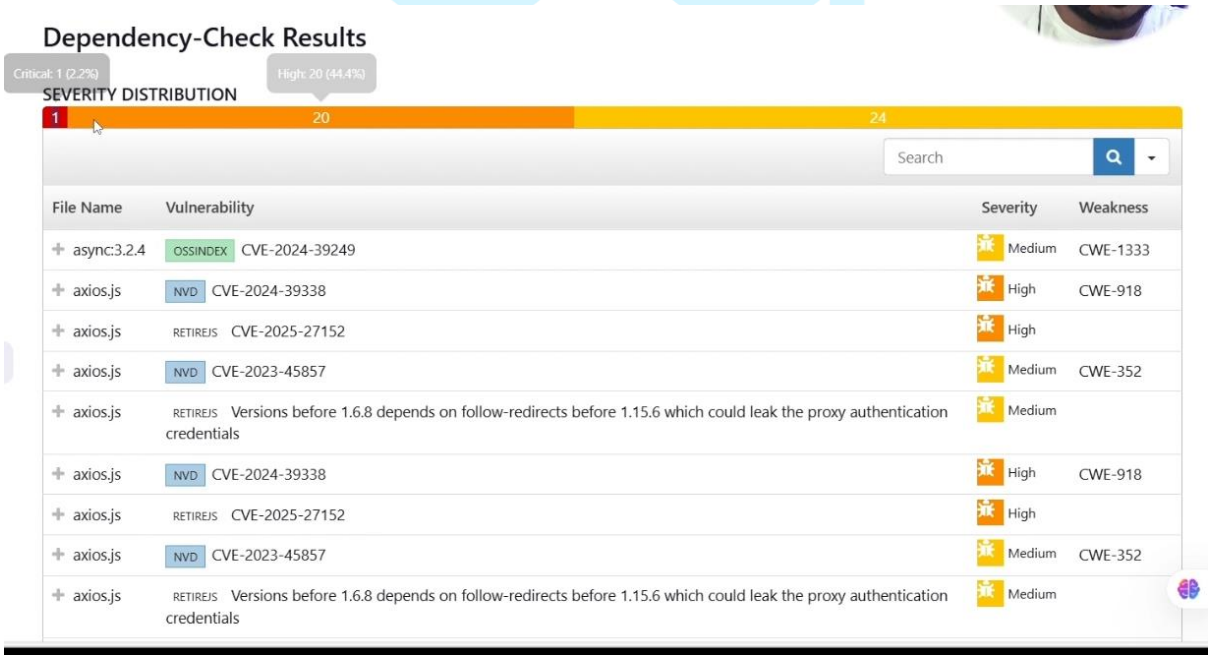
fix with __nvdApikey syntax error , use api to

```
    }
    stage('OWASP FS SCAN') {
        steps {
            dependencyCheck additionalArguments: '--scan ./ --disableYarnAudit --disableNodeAudit --nvdApiKey d7e8c629-7da9
            dependencyCheckPublisher pattern: '**/dependency-check-report.xml'
        }
    }
```

Rerun the Pipeline with above fix

Pipeline is successful





EKS cluster Step on aws

How to create an EKS cluster using AWS Console | Create node group | Configure Kubernetes cluster

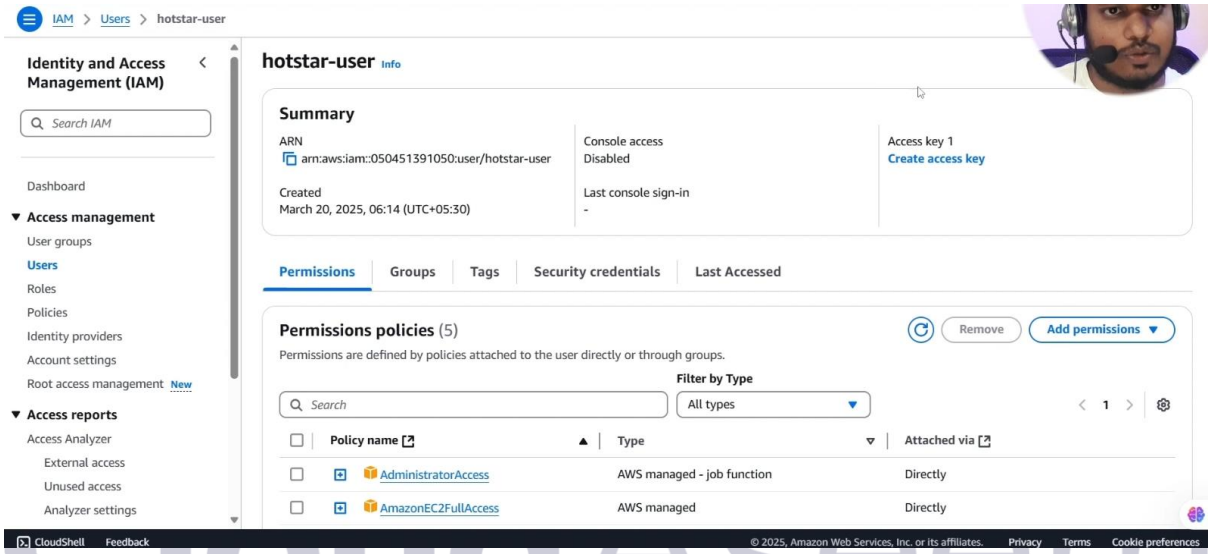
Step - 1 : Create EKS Management Host in AWS

Prerequisites is done already

1. AWS Account: Make sure you have an AWS account and configure one ec2 instance
2. kubectl: Install kubectl.
3. AWS CLI: Install and configure the AWS CLI.
4. eksctl: Install eksctl.

Step - 1 : Create User & add policies as below

Create USER with IAM role & attach to EKS Management Host #



Create user

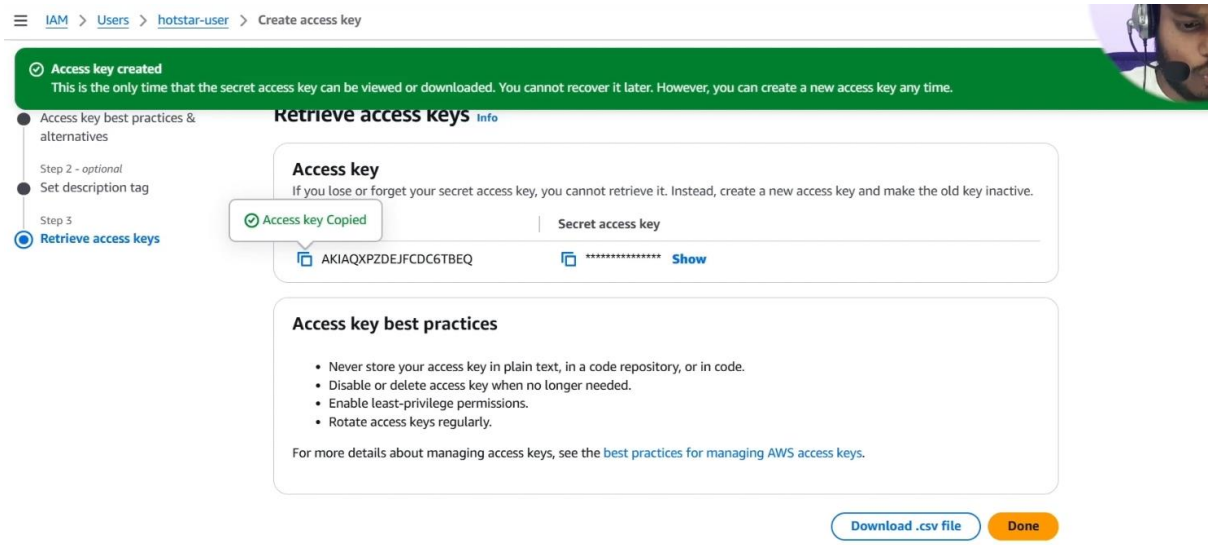
bingouser

1) Create New Role using IAM service (Select Usecase - ec2)
2) Add below permissions for the role
   - IAM - fullaccess
   - VPC - fullaccess
   - EC2 - fullaccess
   - CloudFomration - fullaccess
   - Administrator - acces

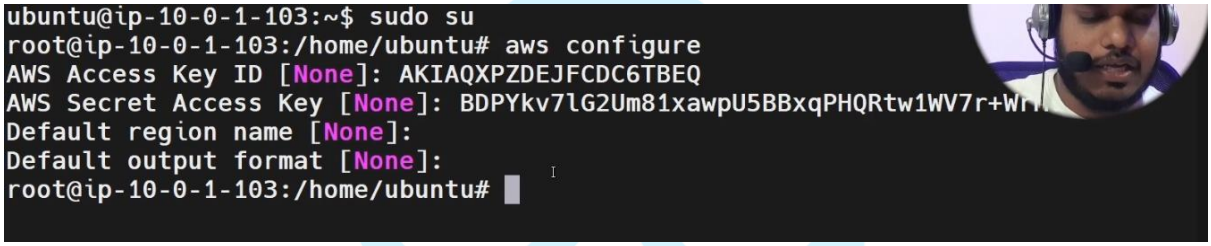
3) Create credentials to connect with your aws account

AWS_ACCESS_KEY_ID

AWS_SECRET_ACCESS_KEY



Aws configure in VM



Step - 3 : Create EKS Cluster using eksctl
Syntax:

```
eksctl create cluster --name cluster-name  \
--region region-name \
--node-type instance-type \
--nodes-min 2 \
--nodes-max 2 \
--zones <AZ-1>,<AZ-2>
```

`

Option 1 for Asian Region ## Mumbai:
`

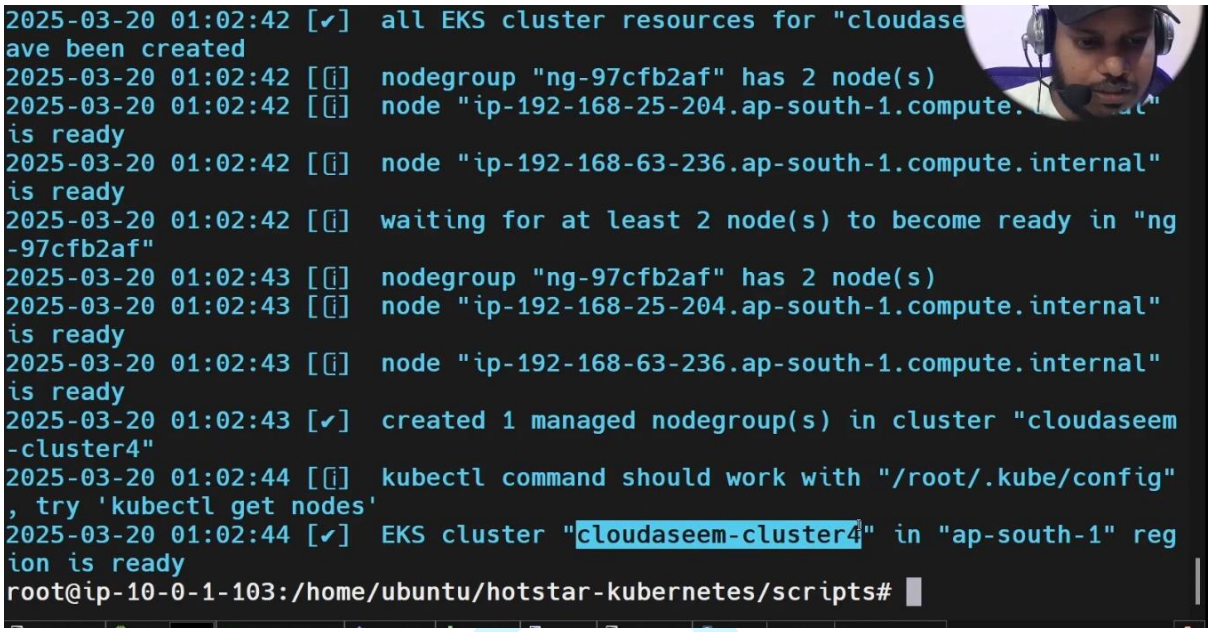
```
eksctl create cluster --name cloudaseem-cluster4 --region ap-south-1 --node-type t2.medium  --zones ap-south-1a,ap-south-1b
```

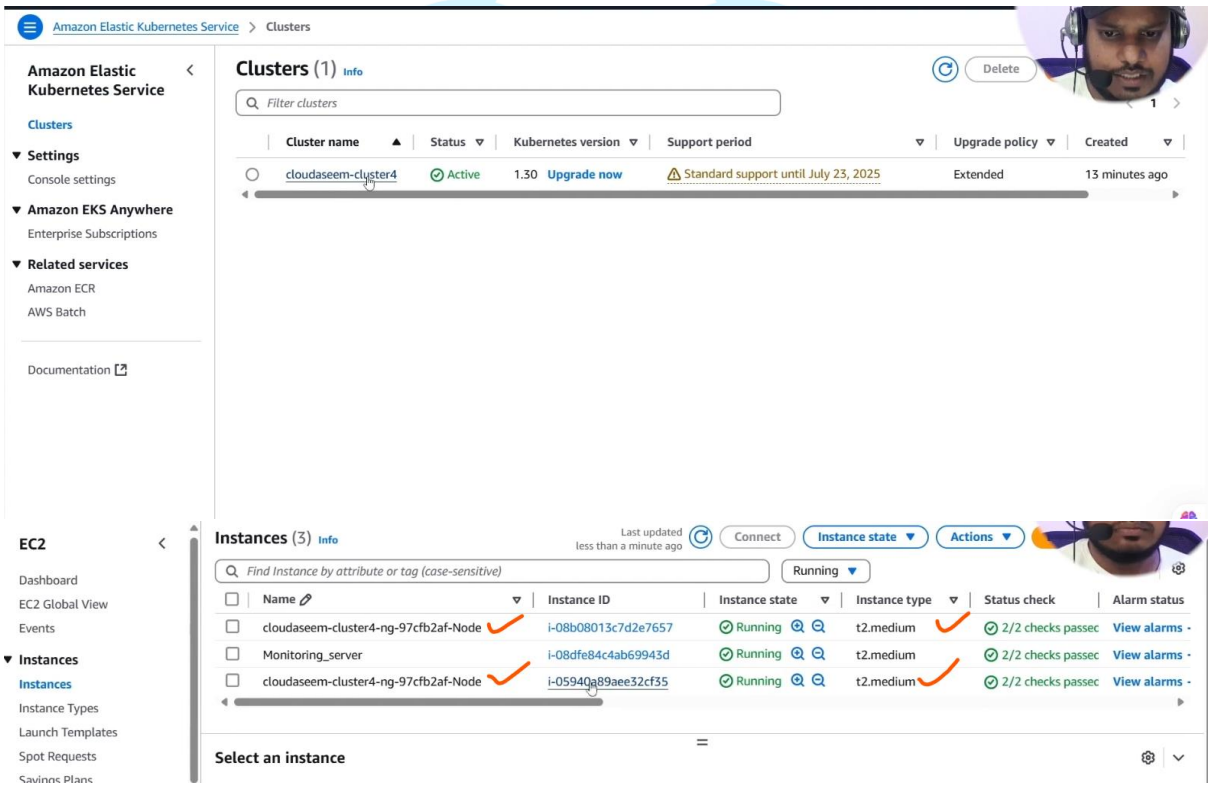
`

Option 2 for US ## N. Virgina:
`

```
eksctl create cluster --name cloudaseem-cluster4 --region us-east-1 --node-type t2.medium  --zones us-east-1a,us-east-1b
```

Note: Cluster creation will take 5 to 10 mins of time (we have to wait). After cluster created we can check nodes using below command.





Got to
project dir
cd hotstar-kubernetes/K8S/

File exist ,

Note to update this manifest.yml file with your Docker images name and apply

manifest.yml file

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hotstar-deployment
spec:
  replicas: 2
  strategy:
    type: RollingUpdate
  selector:
    matchLabels:
      app: hotstar
  template:
    metadata:
      labels:
        app: hotstar
    spec:
      containers:
      - name: hotstar-container
        image: aseemakram19/hotstar
        ports:
        - containerPort: 3000
---
apiVersion: v1
kind: Service
metadata:
  name: hotstar-service
spec:
  type: LoadBalancer
  selector:
    app: hotstar
  ports:
  - port: 80
    targetPort: 3000
```

and execute this command

kubectl apply -f manifest.yml

```
root@ip-10-0-1-103:/home/ubuntu/hotstar-kubernetes/K8S# kubectl apply -f manifest.yml
deployment.apps/hotstar-deployment created
service/hotstar-service created
root@ip-10-0-1-103:/home/ubuntu/hotstar-kubernetes/K8S# kubectl get all
NAME                                    READY   STATUS              RESTARTS   AGE
pod/hotstar-deployment-5d956cf469-h2g6r   0/1     ContainerCreating   0          19s
pod/hotstar-deployment-5d956cf469-mbx8n   0/1     ContainerCreating   0          19s

NAME                      TYPE           CLUSTER-IP       EXTERNAL-IP
                                         PORT(S)        AGE
service/hotstar-service   LoadBalancer   10.100.231.174   a2dd246cc8dd24fdf8c44114d3cb7ade-1
076843819.ap-south-1.elb.amazonaws.com   80:32078/TCP   18s
service/kubernetes        ClusterIP      10.100.0.1       <none>
                                         443/TCP        11m

NAME                                  READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/hotstar-deployment    0/2     2            0           20s

NAME                                             DESIRED   CURRENT   READY   AGE
replicaset.apps/hotstar-deployment-5d956cf469    2         2         0       20s
root@ip-10-0-1-103:/home/ubuntu/hotstar-kubernetes/K8S# kubectl get all
```

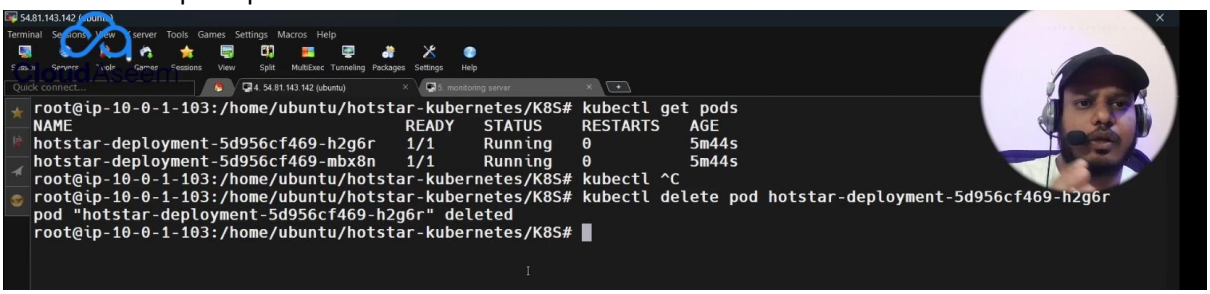
kubectl get all

```
NAME                                  READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/hotstar-deployment    0/2     2            0           20s

NAME                                             DESIRED   CURRENT   READY   AGE
replicaset.apps/hotstar-deployment-5d956cf469    2         2         0       20s
root@ip-10-0-1-103:/home/ubuntu/hotstar-kubernetes/K8S# kubectl get all
NAME                                    READY   STATUS              RESTARTS   AGE
pod/hotstar-deployment-5d956cf469-h2g6r   1/1     Running             0          27s
pod/hotstar-deployment-5d956cf469-mbx8n   0/1     ContainerCreating   0          27s

NAME                      TYPE           CLUSTER-IP       EXTERNAL-IP
                  PORT(S)        AGE
service/hotstar-service   LoadBalancer   10.100.231.174   a2dd246cc8dd24fdf8c44114d3cb7ade-1076843819.ap-south-1
.elb.amazonaws.com   80:32078/TCP   26s
service/kubernetes        ClusterIP      10.100.0.1       <none>
                  443/TCP        11m

NAME                                  READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/hotstar-deployment    1/2     2            1           28s

NAME                                             DESIRED   CURRENT   READY   AGE
replicaset.apps/hotstar-deployment-5d956cf469    2         2         1       28s
root@ip-10-0-1-103:/home/ubuntu/hotstar-kubernetes/K8S# kubectl get all
NAME                                    READY   STATUS    RESTARTS   AGE
pod/hotstar-deployment-5d956cf469-h2g6r   1/1     Running   0          37s
pod/hotstar-deployment-5d956cf469-mbx8n   1/1     Running   0          37s

NAME                      TYPE           CLUSTER-IP       EXTERNAL-IP                                                                PORT(S)        AGE
service/hotstar-service   LoadBalancer   10.100.231.174   a2dd246cc8dd24fdf8c44114d3cb7ade-1076843819.ap-south-1.elb.amazonaws.com   80:32078/TCP   36s
service/kubernetes        ClusterIP      10.100.0.1       <none>                                                                     443/TCP        11m

NAME                                  READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/hotstar-deployment    2/2     2            2           38s

NAME                                             DESIRED   CURRENT   READY   AGE
replicaset.apps/hotstar-deployment-5d956cf469    2         2         2       38s
root@ip-10-0-1-103:/home/ubuntu/hotstar-kubernetes/K8S#
```


TEST Kubernetes Auto Healing Function

Kubernetes has a built-in self-healing mechanism that automatically replaces failed or deleted pods when they are managed by a Deployment, ReplicaSet, or StatefulSet.

kubectl delete pod <pod-name>



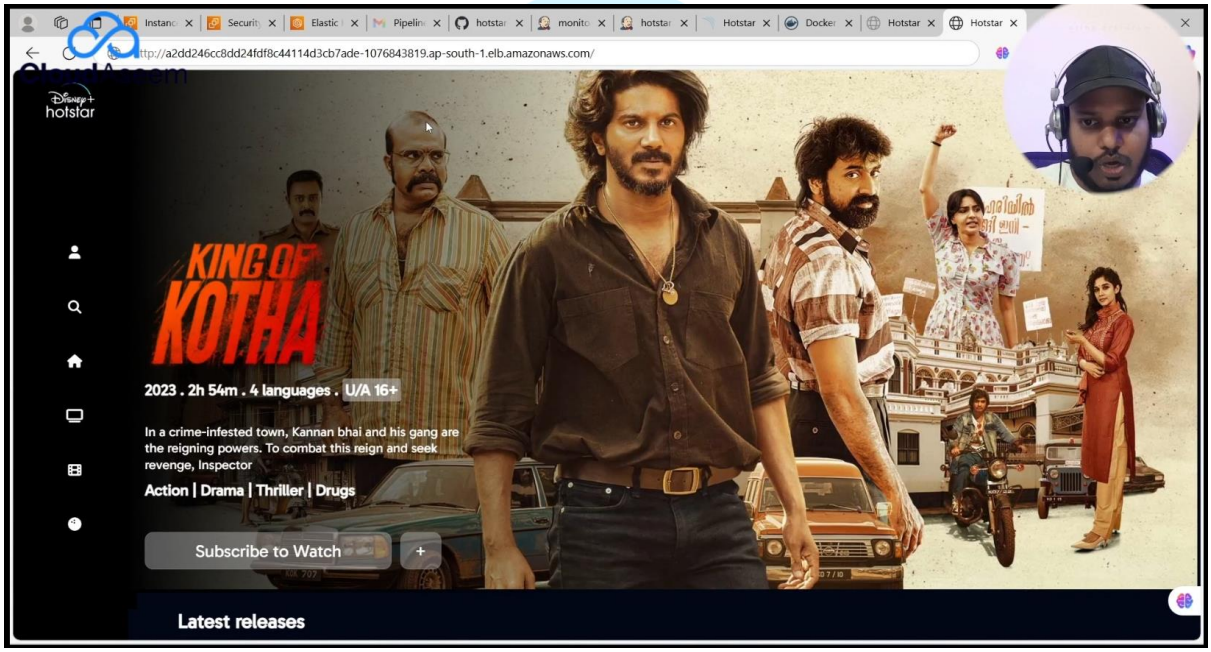
Look for a new pod with a different name but managed by the same **Deployment/ReplicaSet**.

```
root@ip-10-0-1-103:/home/ubuntu/hotstar-kubernetes/K8S# kubectl get pods
NAME                                    READY   STATUS    RESTARTS   AGE
hotstar-deployment-5d956cf469-crmqc     1/1     Running   0          37s
hotstar-deployment-5d956cf469-mbx8n     1/1     Running   0          6m52s
root@ip-10-0-1-103:/home/ubuntu/hotstar-kubernetes/K8S#
```

Application is accessible with Loadbalancer

```
NAME                        TYPE           CLUSTER-IP       EXTERNAL-IP
                    PORT(S)        AGE
service/hotstar-service     LoadBalancer   10.100.231.174   a2dd246cc8dd24fdf8c44114d3cb7ade-1076843819.ap-south-1
.elb.amazonaws.com   80:32078/TCP   26s
service/kubernetes          ClusterIP      10.100.0.1       <none>
                    443/TCP        11m
```

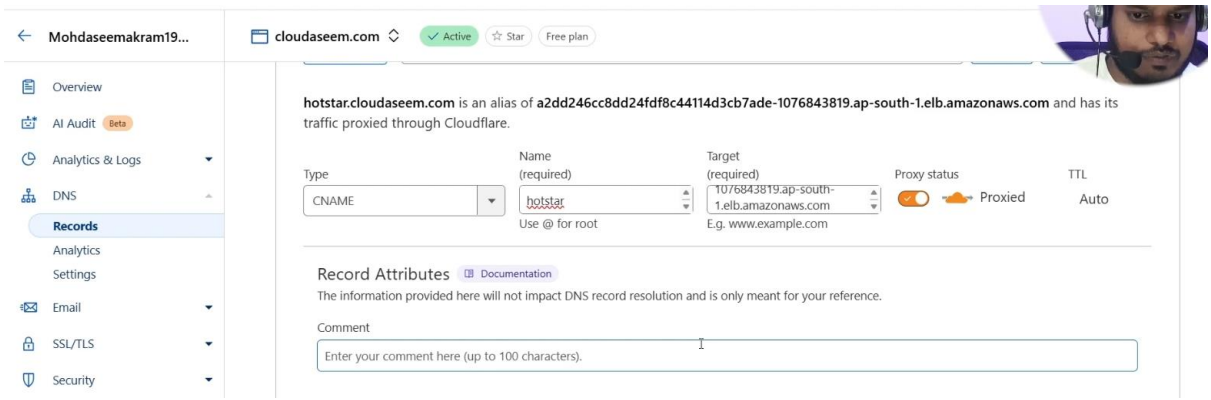
You have successfully Deployed a Hotstar on Kubernetes with Loadbalancer Enabled with AutoHealing.



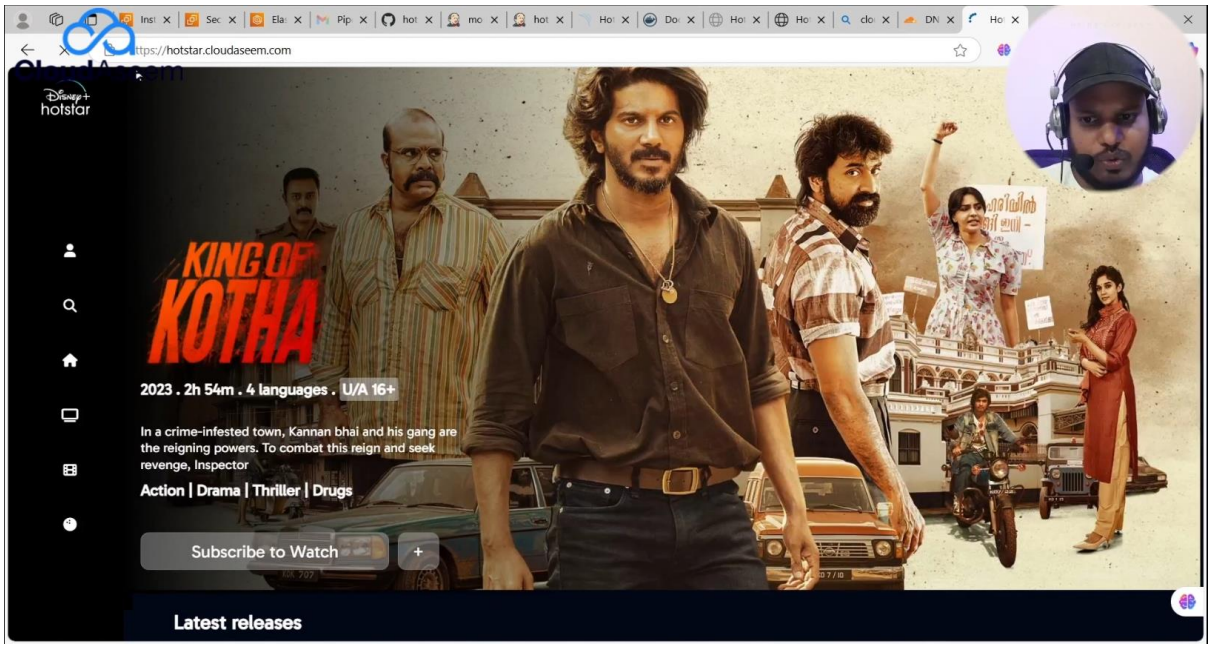
Add Loadbalance in Cloudflare to apply domain and ssl to access by clinet with cname entry

Configure CNAME for Client Access

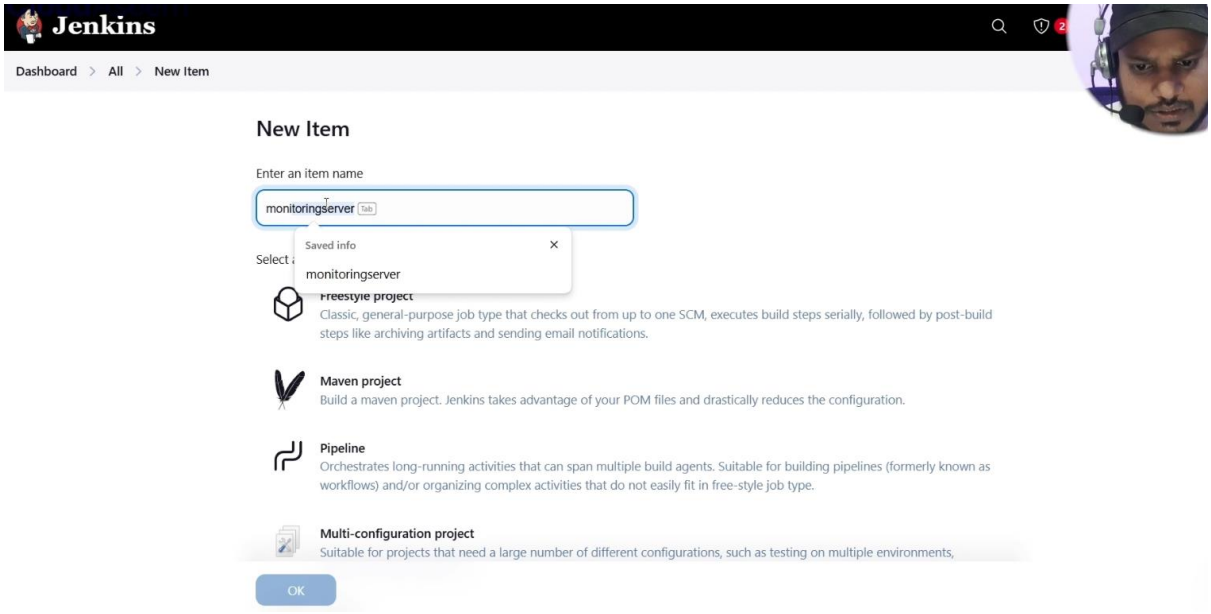
1. **Go to** Cloudflare **DNS Settings**.
2. Add a **CNAME Record**:
    - **Name**: app.example.com
    - **Target**: Load Balancer domain (e.g., lb.example.com)
    - **Proxy Status**: **Proxied (Orange Cloud)** for Cloudflare SSL.
3. Save and Test.



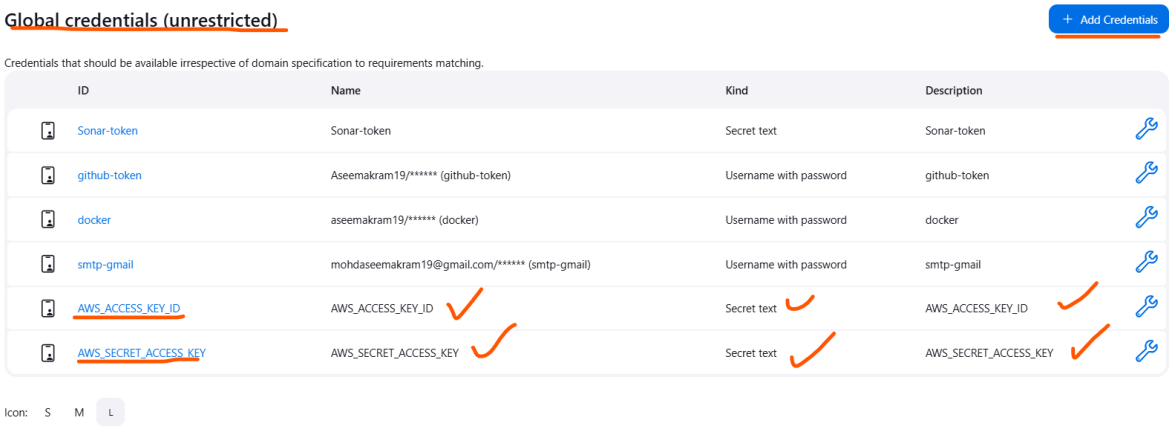
You have successfully Deployed a Hotstar on Kubernetes with Loadbalancer Enabled and SSL certificated



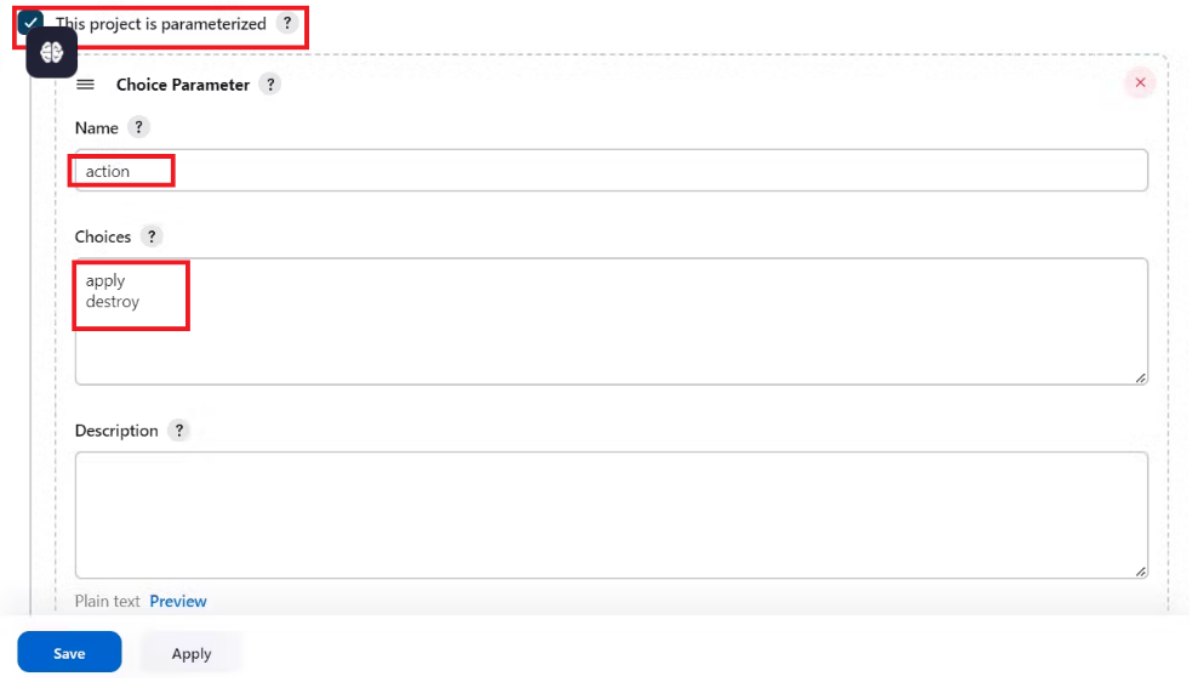
Monitoring Server setup with Jenkins + Terraform



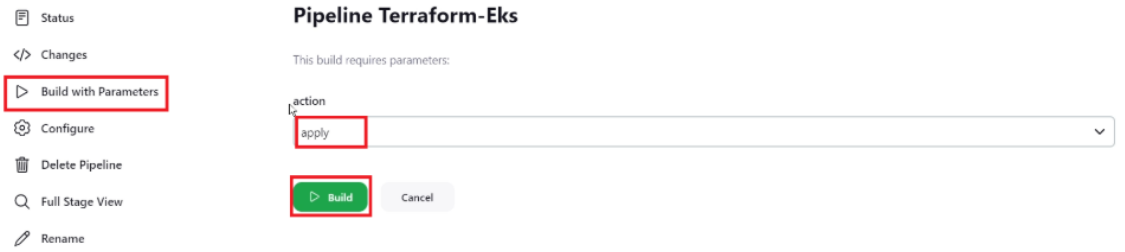
add Secret in AWS credentials



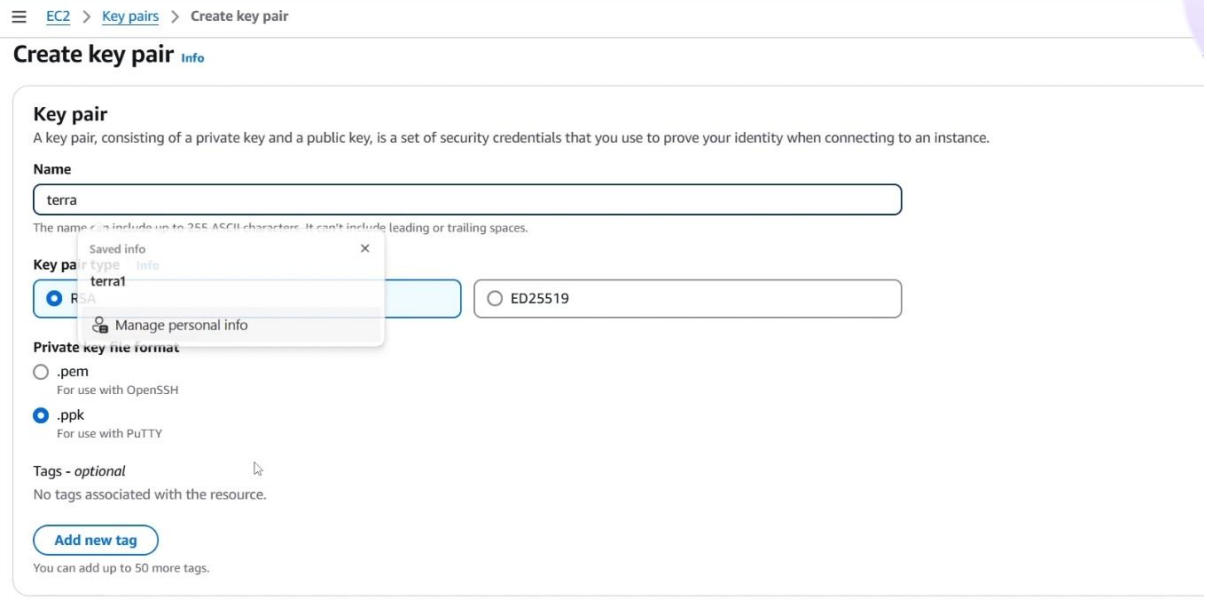
I want to do this with build parameters to apply and destroy while building only.
you have to add this inside job like the below image



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



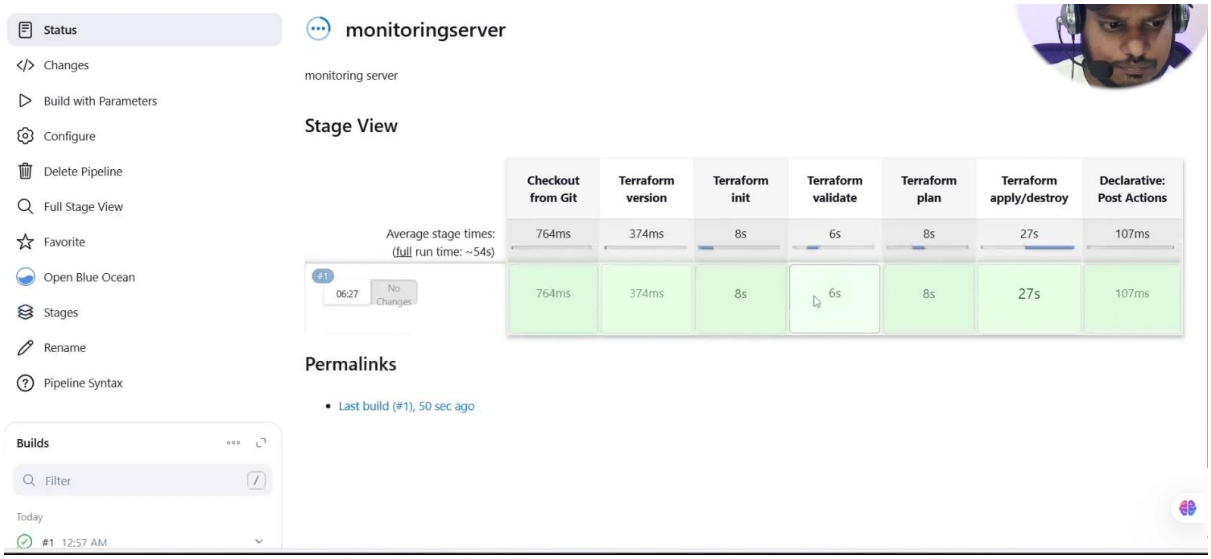
Note: Create Key pair to access monitoring server



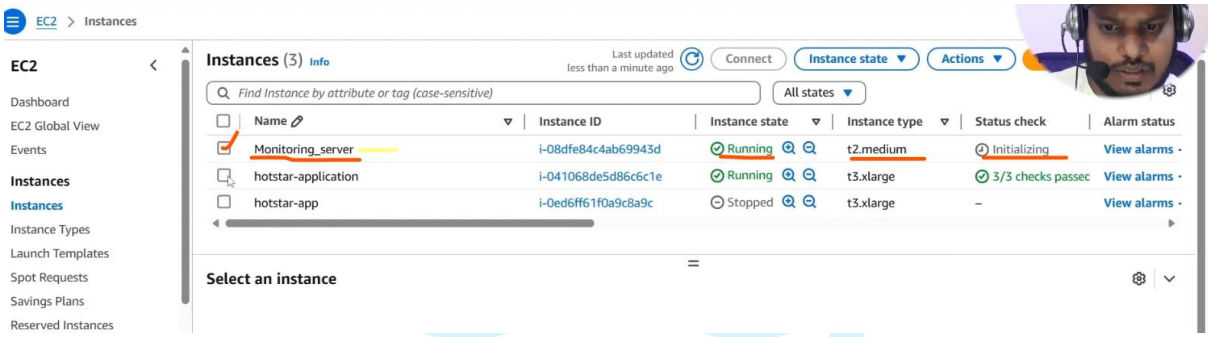
Add script

```
pipeline {
    agent any

    environment {
        AWS_ACCESS_KEY_ID     = credentials('AWS_ACCESS_KEY_ID')
        AWS_SECRET_ACCESS_KEY = credentials('AWS_SECRET_ACCESS_KEY')
    }

    parameters {
        string(name: 'action', defaultValue: 'apply', description: 'Terraform action: apply or destroy')
    }

    stages {
        stage('Checkout from Git') {
            steps {
                git branch: 'main', credentialsId: 'github-token', url:
'https://github.com/Aseemakram19/hotstar-kubernetes.git'
            }
        }

        stage('Terraform version') {
            steps {
                sh 'terraform --version'
            }
        }

        stage('Terraform init') {
            steps {
                dir('Terraform') {
                    sh '''
                    terraform init \
                    -backend-config="access_key=$AWS_ACCESS_KEY_ID" \
                    -backend-config="secret_key=$AWS_SECRET_ACCESS_KEY"
                    '''
                }
            }
        }

        stage('Terraform validate') {
            steps {
                dir('Terraform') {
                    sh 'terraform validate'
                }
            }
        }

        stage('Terraform plan') {
```

```
        steps {
            dir('Terraform') {
                sh '''
                terraform plan \
                -var="access_key=$AWS_ACCESS_KEY_ID" \
                -var="secret_key=$AWS_SECRET_ACCESS_KEY"
                '''
            }
        }
    }

    stage('Terraform apply/destroy') {
        steps {
            dir('Terraform') {
                sh '''
                terraform ${action} --auto-approve \
                -var="access_key=$AWS_ACCESS_KEY_ID" \
                -var="secret_key=$AWS_SECRET_ACCESS_KEY"
                '''
            }
        }
    }
  }

  post {
    success {
        echo '✅ Terraform execution completed successfully!'
    }
    failure {
        echo '❌ Terraform execution failed! Check the logs.'
    }
  }
}
```



Verify the Monitoring server



Installing Grafana and Prometheus for Monitoring

Grafana and Prometheus are commonly used for monitoring Kubernetes clusters, EC2 instances, and other infrastructure components. Follow these steps to install them on an **Ubuntu** server.

1. **Grafana installation**

   Access and create a garafan.sh , add permission , and xecute it

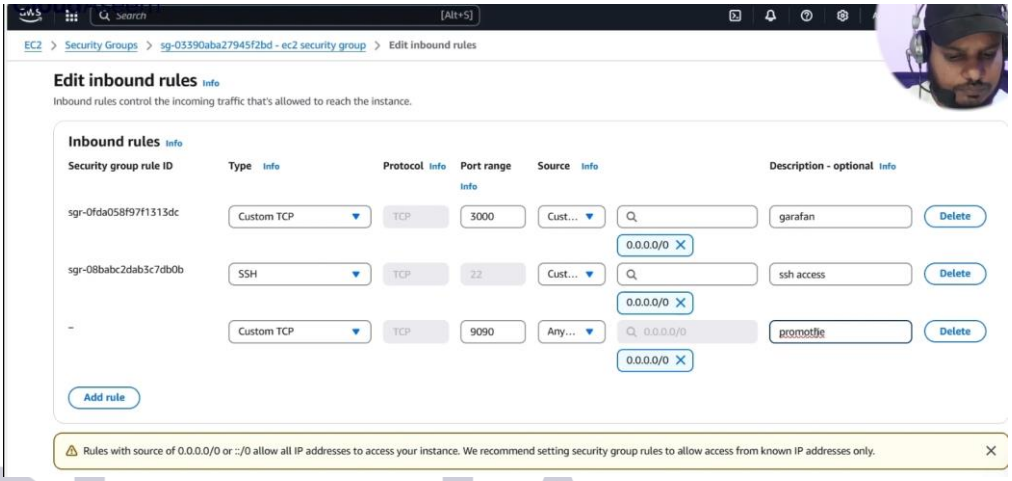
   ```
   #!/bin/bash
   # Script to install Grafana on a Linux instance

   # Update package list and install dependencies
   sudo apt-get install -y apt-transport-https software-properties-common wget

   # Create a directory for Grafana's GPG key
   sudo mkdir -p /etc/apt/keyrings/

   # Add Grafana's GPG key
   wget -q -O - https://apt.grafana.com/gpg.key | gpg --dearmor | sudo tee
   /etc/apt/keyrings/grafana.gpg > /dev/null
   ```

Add Grafana's repository to the sources list
echo "deb [signed-by=/etc/apt/keyrings/grafana.gpg] https://apt.grafana.com stable main" | sudo tee -a /etc/apt/sources.list.d/grafana.list

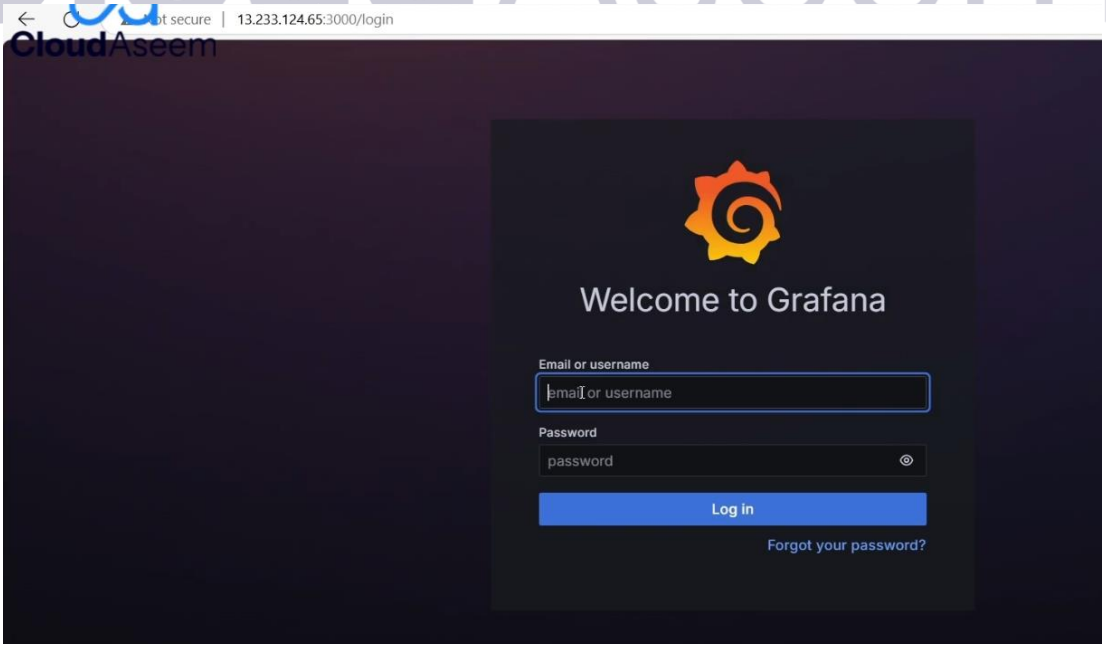
# Update package lists
sudo apt-get update -y
# Install the latest OSS release of Grafana
sudo apt-get install grafana -y

# Start and enable Grafana service
sudo systemctl start grafana-server
sudo systemctl enable grafana-server

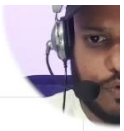
Open ports in SG group



After installation, you can access Grafana at:
http://your-server-ip:3000 (default user: admin, password: admin)



2. **Install Prometheus**



prometheus

The Prometheus monitoring system and time series database. prometheus/prometheus

3.2.1 / 2025-02-25 Release notes

File name	OS	Arch	Size	SHA256 Checksum
prometheus-3.2.1.darwin-amd64.tar.gz	darwin	amd64	110.48 MiB	e73ecf2ccd3ecca4bba8f2f82f82c8c51d190283217c6007b41bfe2478ebe482
prometheus-3.2.1.darwin-arm64.tar.gz	darwin	arm64	106.84 MiB	f71d933be340486e27625a9b80c7be8282a5c8dbb041594ba2e52a4788dc6ebb
prometheus-3.2.1.linux-amd64.tar.gz	linux	amd64	108.87 MiB	a622e3007c9109a7f470e1433cbd29bf392596715cf7eea8b81b37fa9d26b7be
prometheus-3.2.1.windows-amd64.zip	windows	amd64	112.51 MiB	2bd93938b1de339f562321b08e6642582a148c237602272b64d33ce92ba29b71

2.53.4 / 2025-03-18 LTS Release notes

File name	OS	Arch	Size	SHA256 Checksum

Install Blackbox exporter

```
--2025-03-20 01:29:24--  https://github.com/prometheus/blackbox_exporter/releases/download/v0.26.0/blackbox_exporter-0.26.0.linux-amd64.tar.gz
Resolving github.com (github.com)... 20.207.73.82
Connecting to github.com (github.com)|20.207.73.82|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/41964498/37e20444-c65c-45e1-9674-99ed1137b43b?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=releaseassetproduction%2F20250320%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20250320T012924Z&X-Amz-Expires=300&X-Amz-Signature=2d54029d913c128ab7b481bfaba38b65fbef005b05aa20b986af70d9395798 69&X-Amz-SignedHeaders=host&response-content-disposition=attachment%3B%20filename%3Dblackbox_exporter-0.26.0.linux-amd64.tar.gz&response-content-type=application%2Foctet-stream [following]
--2025-03-20 01:29:24--  https://objects.githubusercontent.com/github-production-release-asset-2e65be/41964498/37e20444-c65c-45e1-9674-99ed1137b43b?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=releaseassetproduction%2F20250320%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20250320T012924Z&X-Amz-Expires=300&X-Amz-Signature=2d54029d913c128ab7b481bfaba38b65fbef005b05aa20b986af70d9395798 69&X-Amz-SignedHeaders=host&response-content-disposition=attachment%3B%20filename%3Dblackbox_exporter-0.26.0.linux-amd64.tar.gz&response-content-type=application%2Foctet-stream
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.111.133, 185.199.108.133, 185.199.109.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.111.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 12370868 (12M) [application/octet-stream]
Saving to: 'blackbox_exporter-0.26.0.linux-amd64.tar.gz'

blackbox_exporter-0.26.0.linux-a 100%[=====================================================>]  11.80M  40.0MB/s    in 0.3s

2025-03-20 01:29:26 (40.0 MB/s) - 'blackbox_exporter-0.26.0.linux-amd64.tar.gz' saved [12370868/12370868]

root@ip-172-31-38-246:/home/ubuntu#
```

Step 1: Edit prometheus.yml

Open the **Prometheus configuration file**:

Add the following **scrape jobs** at the end of the file:

```
- job_name: 'blackbox'
   metrics_path: /probe
   params:
     module: [http_2xx]  # Look for a HTTP 200 response.
   static_configs:
     - targets:
         - http://prometheus.io      # Target to probe with HTTP.
         - http://IP:3000 # Target to probe with HTTPS.
   relabel_configs:
     - source_labels: [__address__]
       target_label: __param_target
```

```
    - source_labels: [__param_target]
      target_label: instance
    - target_label: __address__
      replacement: 13.232.214.2:9115  # The blackbox exporter's real hostname.


- job_name: node_exporter
  static_configs:
    - targets:
        - 'IP:9100'
```

Save the file (CTRL + X, then Y, and Enter).

```
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: "prometheus"

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: ["localhost:9090"]

  - job_name: 'blackbox'
    metrics_path: /probe
    params:
      module: [http_2xx]  # Look for a HTTP 200 response.
    static_configs:
      - targets:
          - http://prometheus.io      # Target to probe with HTTP.
          - http://54.81.143.142:3000 # Target to probe with HTTPS.
          - https://hotstar.cloudaseem.com:443 # Target to probe with HTTPS.
    relabel_configs:
      - source_labels: [__address__]
        target_label: __param_target
      - source_labels: [__param_target]
        target_label: instance
      - target_label: __address__
        replacement: 13.233.124.65:9115  # The blackbox exporter's real hostname.
```

Step 2: Restart Prometheus to Apply Changes

pgrep Prometheus and kill PID

```
root@ip-172-31-38-246:/home/ubuntu/prometheus-3.2.1.linux-amd64# nano prometheus.yml
root@ip-172-31-38-246:/home/ubuntu/prometheus-3.2.1.linux-amd64# pgrep prometheus
6166
root@ip-172-31-38-246:/home/ubuntu/prometheus-3.2.1.linux-amd64# kill 6166
root@ip-172-31-38-246:/home/ubuntu/prometheus-3.2.1.linux-amd64# time=2025-03-20T01:35:15.401Z level=WARN source=main.go:1015 msg="
Received an OS signal, exiting gracefully..." signal=terminated
time=2025-03-20T01:35:15.401Z level=INFO source=main.go:1040 msg="Stopping scrape discovery manager..."
time=2025-03-20T01:35:15.401Z level=INFO source=main.go:1054 msg="Stopping notify discovery manager..."
time=2025-03-20T01:35:15.401Z level=INFO source=manager.go:189 msg="Stopping rule manager..." component="rule manager"
time=2025-03-20T01:35:15.401Z level=INFO source=manager.go:205 msg="Rule manager stopped" component="rule manager"
time=2025-03-20T01:35:15.401Z level=INFO source=main.go:1091 msg="Stopping scrape manager..."
time=2025-03-20T01:35:15.401Z level=INFO source=main.go:1036 msg="Scrape discovery manager stopped"
time=2025-03-20T01:35:15.401Z level=INFO source=main.go:1050 msg="Notify discovery manager stopped"
time=2025-03-20T01:35:15.401Z level=INFO source=main.go:1083 msg="Scrape manager stopped"
time=2025-03-20T01:35:15.405Z level=INFO source=notifier.go:702 msg="Stopping notification manager..." component=notifier
time=2025-03-20T01:35:15.405Z level=INFO source=notifier.go:409 msg="Draining any remaining notifications..." component=notifier
time=2025-03-20T01:35:15.405Z level=INFO source=notifier.go:415 msg="Remaining notifications drained" component=notifier
time=2025-03-20T01:35:15.405Z level=INFO source=notifier.go:345 msg="Notification manager stopped" component=notifier
time=2025-03-20T01:35:15.405Z level=INFO source=main.go:1361 msg="Notifier manager stopped"
time=2025-03-20T01:35:15.406Z level=INFO source=main.go:1375 msg="See you next time!"

[1]-  Done                    ./prometheus
root@ip-172-31-38-246:/home/ubuntu/prometheus-3.2.1.linux-amd64#
```

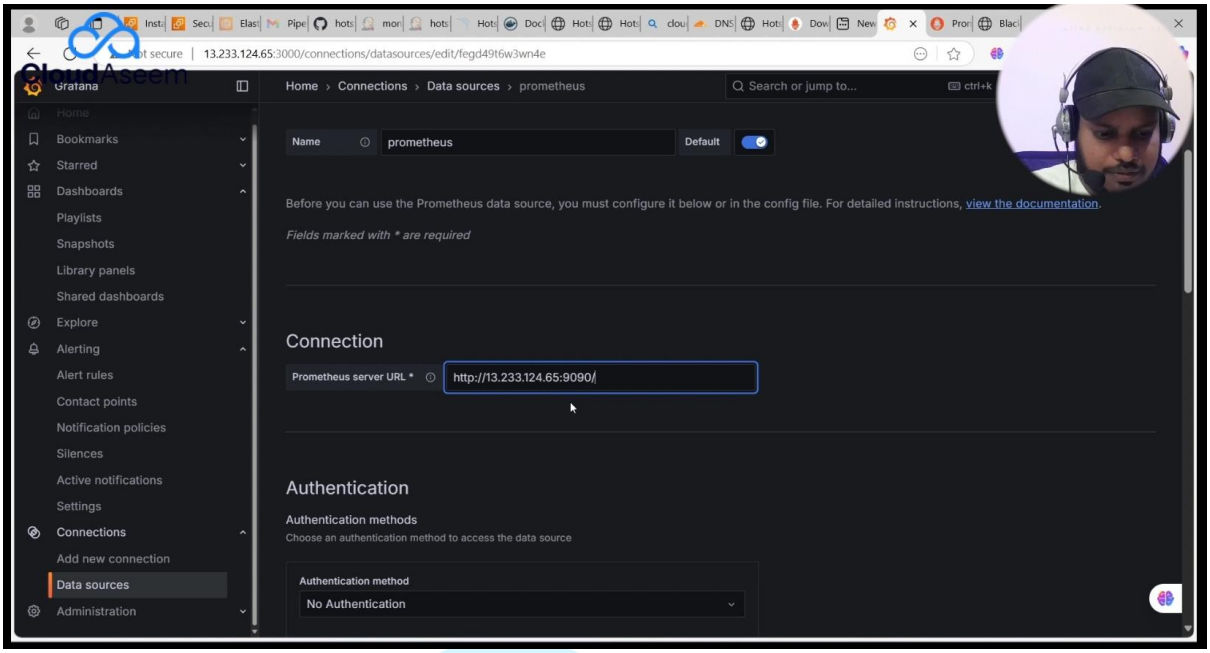
restart

./Prometheus &

Step 3: Connect Prometheus to Grafana

1. **Login to Grafana (http://<your-server-ip>:3000).**
2. **Go to Configuration → Data Sources → Add Data Source.**
3. **Select Prometheus.**
4. **Set Prometheus URL: http://localhost:9090.**

5. **Click Save & Test.**



Add

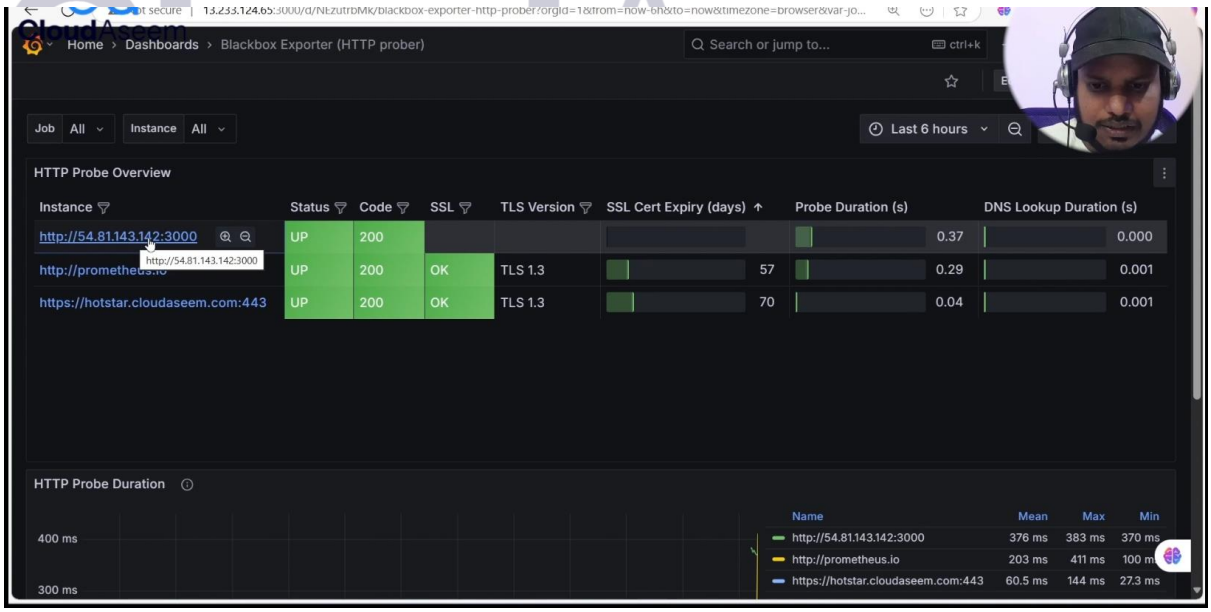
Go to **Dashboards → Import**.

Enter **Dashboard ID: 13659** (Prometheus Blackbox Exporter).

Click **Load**.

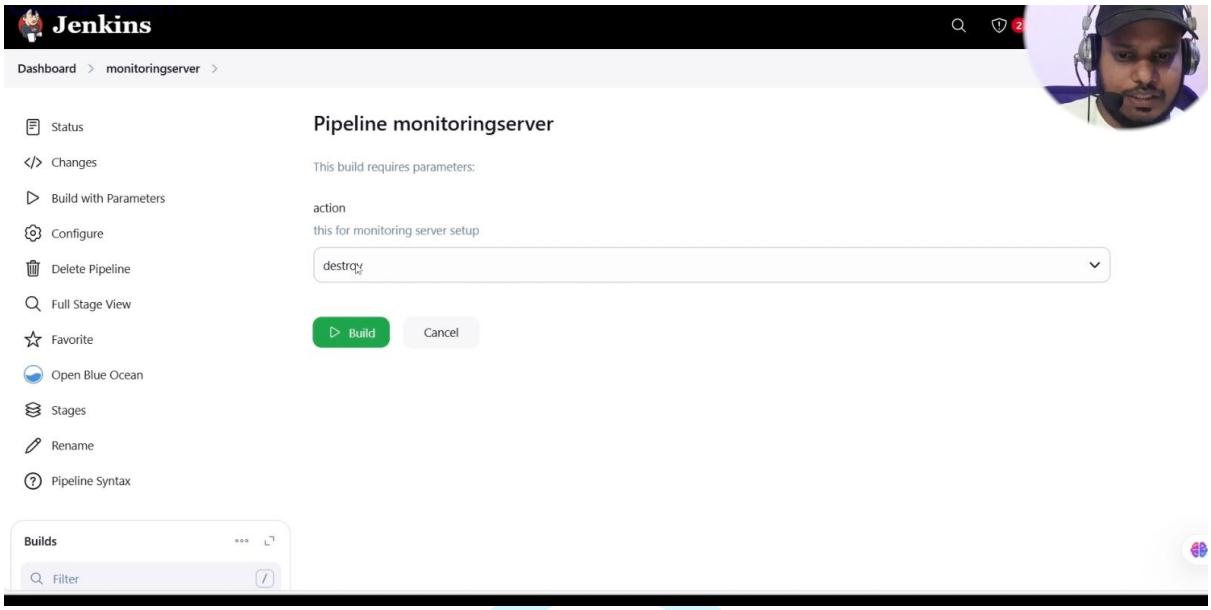
Select **Prometheus** as the **Data Source**.

Click **Import**.



Step - 4 :

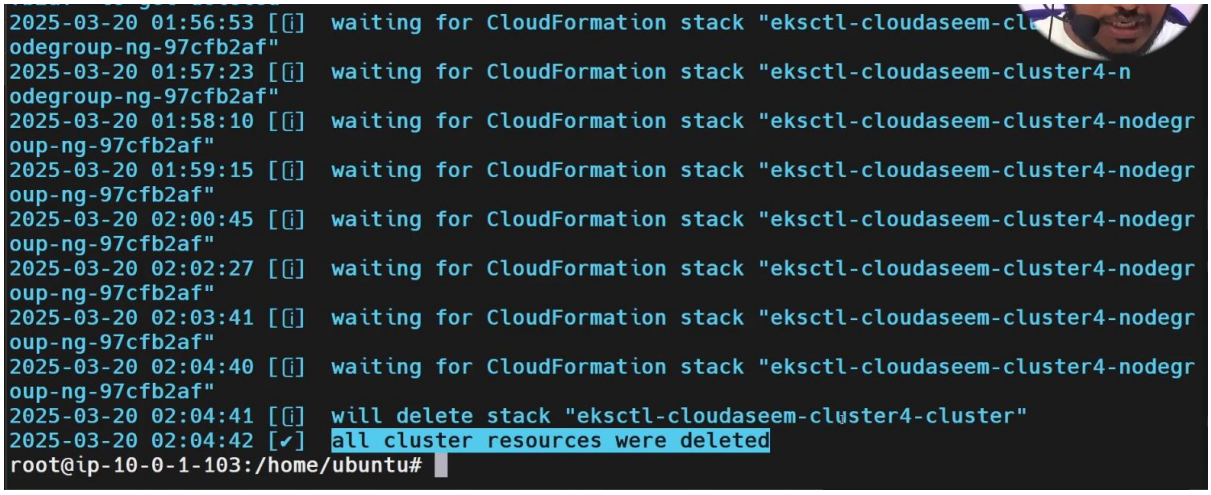
1. Delete Monitoring Server with Jenkins pipeline with action as destroy



2. delete Cluster and other resources we have used in AWS Cloud to avoid billing ##

eksctl delete cluster --name cloudaseem-cluster4 --region ap-south-1

```
root@ip-10-0-1-103:/home/ubuntu# eksctl delete cluster --name cloudaseem-cluster4 --reg
ion ap-south-1
2025-03-20 01:56:07 [ℹ]  deleting EKS cluster "cloudaseem-cluster4"
2025-03-20 01:56:08 [ℹ]  will drain 0 unmanaged nodegroup(s) in cluster "cloudaseem-clu
ster4"
2025-03-20 01:56:08 [ℹ]  starting parallel draining, max in-flight of 1
2025-03-20 01:56:10 [ℹ]  deleted 0 Fargate profile(s)
2025-03-20 01:56:11 [✔]  kubeconfig has been updated
2025-03-20 01:56:11 [ℹ]  cleaning up AWS load balancers created by Kubernetes objects o
f Kind Service or Ingress
```



✅ **Congratulations!** You have successfully deployed **Hotstar on Kubernetes** with:

- 🔹 **Load Balancer Enabled**
- 🔹 **SSL Certificates Configured**
- 🔹 **Monitoring Setup Complete**

📢 **If you found this tutorial helpful, don't forget to:**

👍 **Like** the video

💬 **Comment** your thoughts and questions

🔔 **Subscribe** to stay updated on **Cloud & DevOps tutorials!**

🌍 **Follow me for more updates on Cloud & DevOps:**

🔗 **YouTube:** Cloud DevOps with Aseem - https://www.youtube.com/@clouddevopswithaseem

🔗 **LinkedIn:** Mohammed Aseem Akram - www.linkedin.com/in/mohammed-aseem-akram

🔗 **GitHub:** Aseemakram19

📌 **Stay tuned for more DevOps insights!** 🚀

#Cloud #DevOps #Technology #Kubernetes