

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.



Production Deployment of Hotstar Disney+ App on Kubernetes Eks | Monitor



sonarqube



🌐 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.

The screenshot shows the AWS EC2 Security Groups inbound rules configuration. There are six rules listed:

- Rule 1: HTTPS (TCP 443) from Anywhere (0.0.0.0/0) to Anywhere (0.0.0.0/0).
- Rule 2: Custom TCP (TCP 8080) from Anywhere (0.0.0.0/0) to Jenkins (0.0.0.0/0).
- Rule 3: Custom TCP (TCP 9000) from Anywhere (0.0.0.0/0) to Sonar (0.0.0.0/0).
- Rule 4: Custom TCP (TCP 587) from Anywhere (0.0.0.0/0) to Gmail-SMTP (0.0.0.0/0).
- Rule 5: Custom TCP (Custom Port) from Anywhere (0.0.0.0/0) to Trivy (0.0.0.0/0).

An "Add rule" button is at the bottom left.

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 Scripts . 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:

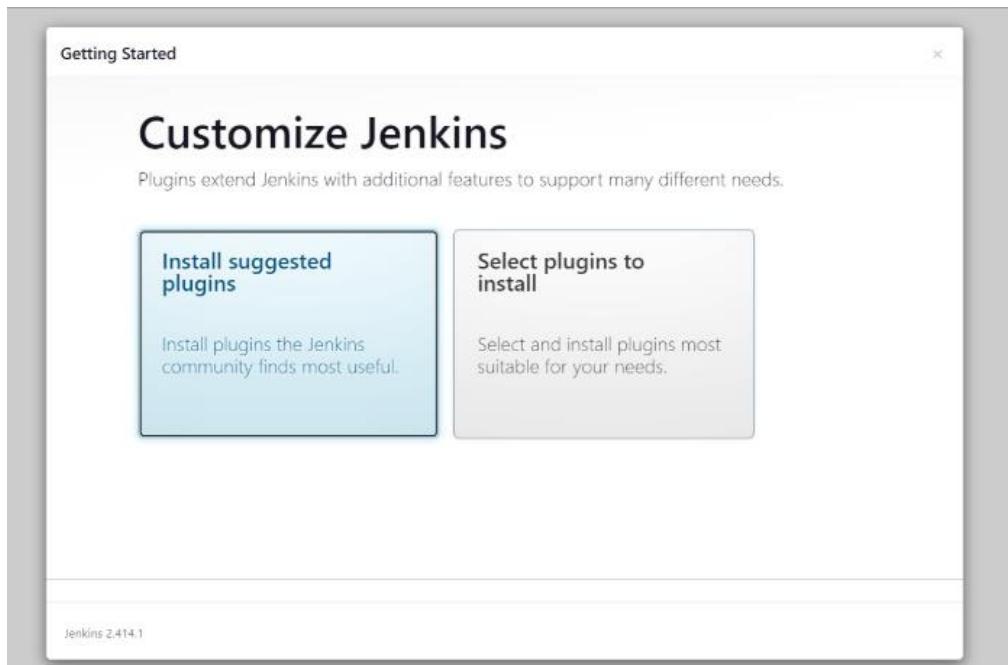
http://<PUBLIC_IP>:8080

The screenshot shows the Jenkins 'Getting Started' page. The title 'Unlock Jenkins' is prominently displayed. Below it, a message states: 'To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server: /var/lib/jenkins/secrets/initialAdminPassword'. A text input field labeled 'Administrator password' is provided for the user to enter the password. A 'Continue' button is located at the bottom right of the form.

- 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

The screenshot shows the Jenkins plugin manager interface under the CloudAseem dashboard. The left sidebar has links for Updates, Available plugins (which is selected), Installed plugins, Advanced settings, and Download progress. The main area lists available plugins with their names, versions, categories, and brief descriptions.

- NodeJS** 1.6.4 (npm): NodeJS Plugin executes NodeJS script as a build step.
- OWASP Dependency-Check** 5.6.0 (Security, DevOps, Build Tools, Build Reports): This plug-in can independently execute a Dependency-Check analysis and visualize results. Dependency-Check is a utility that identifies project dependencies and checks if there are any known, publicly disclosed, vulnerabilities.
- Pipeline: Stage View** 2.37 (User Interface): Pipeline Stage View Plugin.
- Blue Ocean** 1.27.17 (External Site/Tool Integrations, User Interface): BlueOcean Aggregator
- Docker** 1274.vc0203fdf2e74 (Cloud Providers, Cluster Management, docker): This plugin integrates Jenkins with Docker

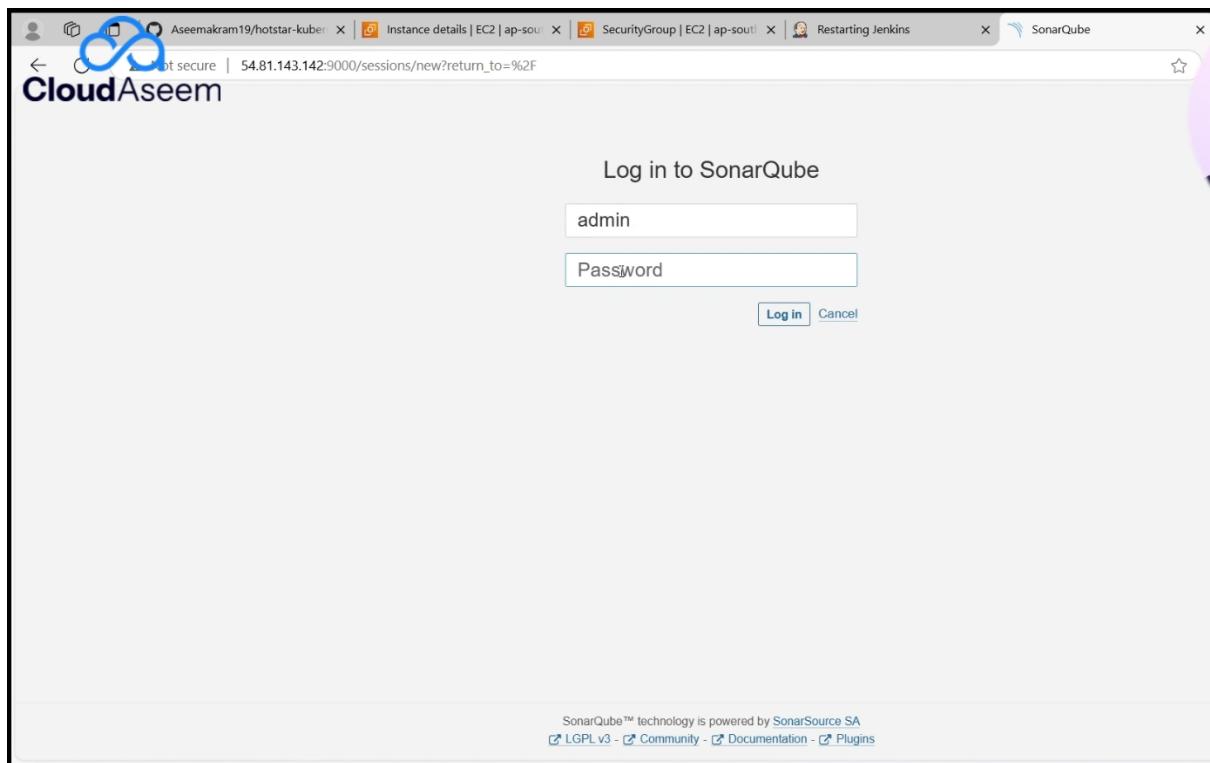
• Setup SonarQube Server

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 9000:9000 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

CloudAseem

Update your password

This account should not use the default password.

Enter a new password

All fields marked with * are required

Old Password *

New Password *

Confirm Password *

Update

Update New password, This is Sonar Dashboard.

The screenshot shows the SonarQube dashboard with the following elements:

- Header:** Shows multiple browser tabs open, including "Aseemakram19/hoststar-kuber", "Instance details | EC2 | ap-sout", "SecurityGroup | EC2 | ap-sout", and "Restarting Jenkins".
- Breadcrumbs:** "CloudAseem" and "sonarqube".
- Notification Bar:** "There's a new version of SonarQube available. Upgrade to the latest active version to access new updates and features." with a "Learn More" link.
- Navigation:** "Projects" (highlighted), "Issues", "Rules", "Quality Profiles", "Quality Gates", "Administration".
- Search:** "Search for project" with a magnifying glass icon.
- Section: "How do you want to create your project?"**
 - From Azure DevOps:** Icon of a blue square with white lines, "From Azure DevOps", "Set up global configuration".
 - From Bitbucket Server:** Icon of a blue square with white lines, "From Bitbucket Server", "Set up global configuration".
 - From Bitbucket Cloud:** Icon of a blue square with white lines, "From Bitbucket Cloud", "Set up global configuration".
 - From GitHub:** Icon of a black octocat, "From GitHub", "Set up global configuration".
 - From GitLab:** Icon of a red and orange logo, "From GitLab", "Set up global configuration".
- Section: "Are you just testing or have an advanced use-case? Create a project manually."**
 - Manually:** Icon of two arrows, "Manually".
- Call-to-action:** "Get the most out of SonarQube!" with icons for SonarLint, SonarCloud, and SonarQube, and text: "Take advantage of the whole ecosystem by using SonarQube IDE plugin that helps you find and fix issues earlier in your development cycle. Connect SonarLint to SonarQube to sync rule sets and more..."

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

The screenshot shows the SonarQube administration interface. The top navigation bar includes links for sonarcube, Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, and a search bar. A user profile picture is visible in the top right. The main content area is titled 'Administration' and 'Users'. It says 'Create and administer individual users.' and includes a search bar. Below is a table with columns: SCM Accounts, Last connection, Groups, and Tokens. One row is shown for 'Administrator admin', which is part of the 'sonar-administrators' group and has 0 tokens. A 'Create User' button is at the top right of the user list.

Create a token with a name and generate

The screenshot shows the 'Tokens of Administrator' page. It has a 'Generate Tokens' section where a token name 'jenkins' is entered and set to expire in 30 days. A success message states 'New token "jenkins" has been created. Make sure you copy it now, you won't be able to see it again!' Below is a table of tokens. The first and only token listed is 'jenkins', which is a 'User' type token. It was created on March 20, 2025, and has never been used. There is a 'Revoke' button next to it. A 'Done' button is at the top right.

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

The screenshot shows the Jenkins 'Manage Jenkins' → 'System' configuration page. It displays a form for adding a SonarQube instance. The 'Name' field is filled with 'SonarQube'. The 'Server URL' field contains 'http://54.81.143.142:9000'. The 'Server authentication token' dropdown is set to '- none -'. There are 'Add' and 'Advanced' buttons at the bottom. A 'Done' button is visible on the right side of the main content area.

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

Jenkins Credentials Provider: Jenkins

Secret text	
Scope ? Global (Jenkins, nodes, items, all child items, etc)	
Secret	
ID ? Sonar-token	
Description ? token	
Saved info X	
token	
Cancel Add	

You will see this page once you click on create

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description	Actions
 Sonar-token	sonar	Secret text	sonar	

The **Configure System** option is used in Jenkins to configure different servers

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

≡ SonarQube Scanner

Name sonar-scanner	!
Required	
<input checked="" type="checkbox"/> Install automatically ?	x
≡ Install from Maven Central	
Version SonarQube Scanner 7.0.2.4839	x
Add Installer ▾	

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

The screenshot shows the Sonarqube Administration → Configuration → Webhooks page. A new webhook is being created with the following details:

- Name ***: jenkins
- URL ***: <http://54.81.143.142:8080/sonarqube-webhook/>
- Secret**: (empty field)

The "Create" button is visible at the bottom right of the modal.

Setup Jenkins GitHub token inorder to connect with Private Registry -

Generate Classic GitHub Token

Go to GitHub → Settings → Developer Settings → Personal Access Tokens.

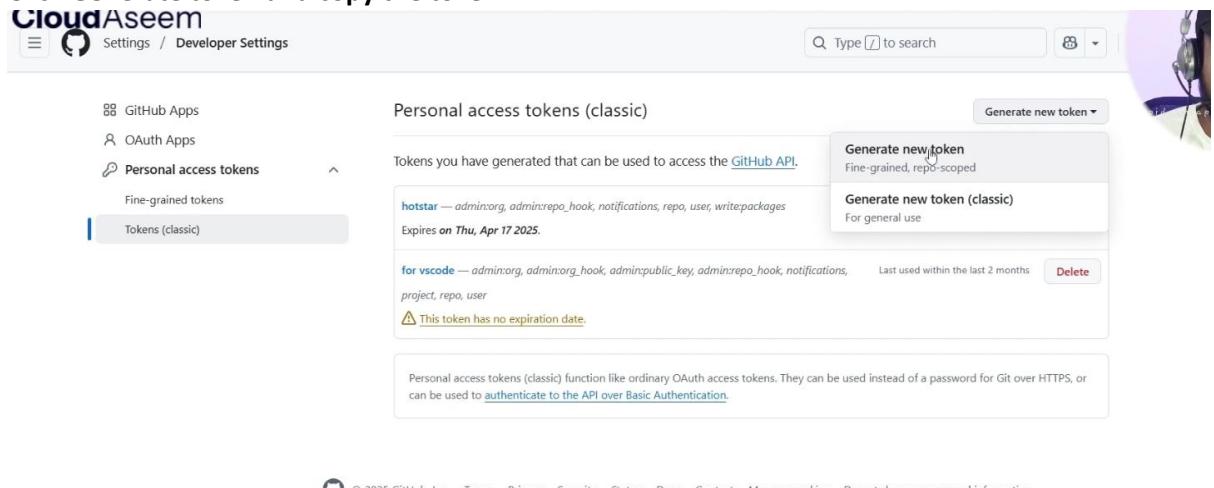
Click Generate new token (classic).

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.



The screenshot shows the GitHub Developer Settings page under Personal access tokens (classic). It lists two tokens:

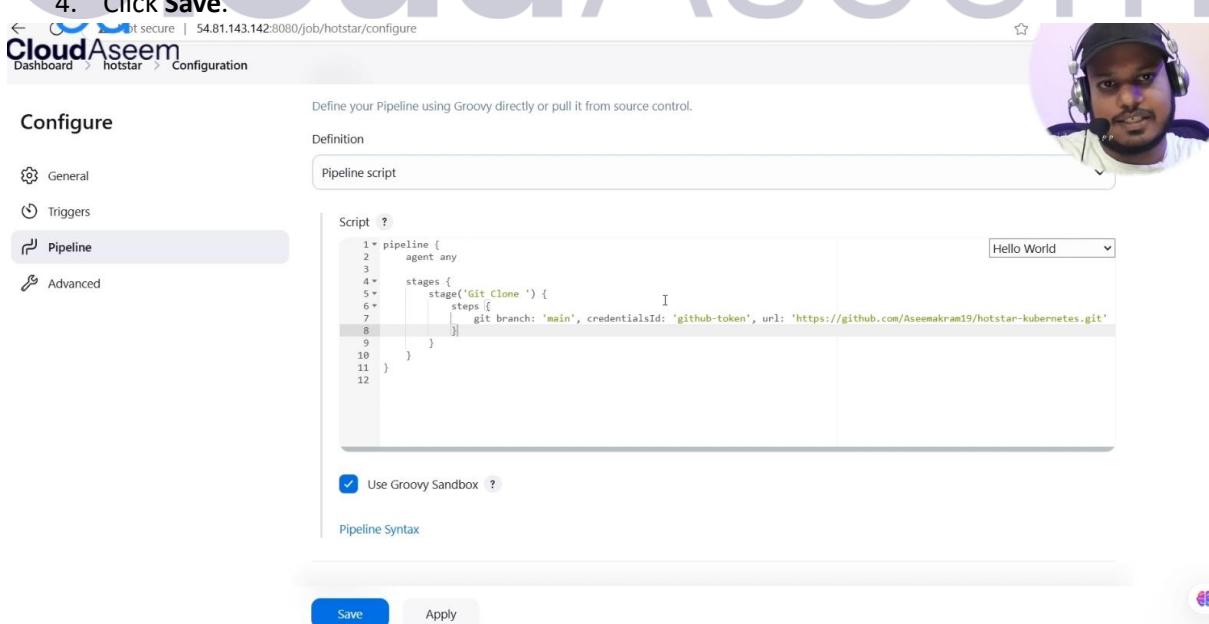
- hotstar** — admin:org, admin:repo_hook, notifications, repo, user, write_packages. Expires on Thu, Apr 17 2025.
- for vscode** — admin:org, admin:org_hook, admin:public_key, admin:repo_hook, notifications, project, repo, user. Last used within the last 2 months. Delete button.

A note at the bottom states: "Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication."

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.



The screenshot shows the Jenkins Pipeline configuration screen for a job named 'hotstar'. The pipeline script is defined as follows:

```

1 * pipeline {
2     agent any
3
4     stages {
5         stage('Git Clone') {
6             steps [
7                 git branch: 'main', credentialsId: 'github-token', url: 'https://github.com/Aseemakram19/hotstar-kubernetes.git'
8             ]
9         }
10    }
11 }
12

```

The 'Pipeline' tab is selected in the left sidebar. The pipeline script is pasted into the 'Pipeline script' text area. A 'Hello World' dropdown is visible on the right. At the bottom, there are 'Save' and 'Apply' buttons.

Create Job for Hotstar

Let's add a pipeline , to test the Github Clone stage of Private Registry

The screenshot shows the CloudAseem Pipeline Configuration interface. On the left, a sidebar lists 'General', 'Triggers', 'Pipeline' (which is selected), and 'Advanced'. The main area has a heading 'Configure' and a 'Definition' section titled 'Pipeline script'. A code editor contains the following Groovy script:

```
1 * pipeline {
2     agent any
3
4     stages {
5         stage('Git Clone') {
6             steps [
7                 git branch: 'main', credentialsId: 'github-token', url: 'https://github.com/Aseemakram19/hotstar-kubernetes.git'
8             ]
9         }
10    }
11 }
```

Below the code editor is a checkbox 'Use Groovy Sandbox ?' which is checked. At the bottom are 'Save' and 'Apply' buttons.

Apply and Save and click on Build

The screenshot shows the CloudAseem Stage View for the pipeline. The left sidebar includes options like 'Build Now', 'Configure', 'Delete Pipeline', 'Full Stage View', 'Favorite', 'Open Blue Ocean', 'Stages', 'Rename', and 'Pipeline Syntax'. The main area is titled 'Stage View' and shows a 'Git Clone' stage. It indicates an average stage time of 1s and a full run time of ~4s. A build log entry for build #1 at 02:30 shows 'No Changes'. Below this is a 'Permalinks' section with a list of recent builds:

- Last build (#1), 10 sec ago
- Last stable build (#1), 10 sec ago
- Last successful build (#1), 10 sec ago
- Last completed build (#1), 10 sec ago

A 'Builds' section at the bottom shows a dropdown menu with 'Today' and '#1 9:00 PM' selected.

Add docker Credentials

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

Global (Jenkins, nodes, items, all child items, etc)

Username ?
aseemakram19

Treat username as secret ?

Password ?
.....

ID ?
docker

Description ?
docker

Create

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).

← App passwords



App passwords help you sign into your Google Account on older apps and services that don't support modern security standards.

App passwords are less secure than using up-to-date apps and services that use modern security standards. Before you create an app password, you should check to see if your app needs this in order to sign in.

[Learn more](#)

You don't have any app passwords.

To create a new app specific password, type a name for it below...

App name:

Saved info:

- hotstar
- hotstar1

[Manage personal info](#)

[Create](#)

[Privacy](#) [Terms](#) [Help](#) [About](#)

Add credentials as Username and password in jenkins

Update credentials

Scope [?](#)

Global (Jenkins, nodes, items, all child items, etc)

Username [?](#)

mohdaseemakram19@gmail.com

Treat username as secret [?](#)

Password [?](#)

Concealed

[Change Password](#)

ID [?](#)

smtp-gmail

Description [?](#)

smtp-gmail

[Save](#)

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:
 - o **SMTP Server:** smtp.gmail.com
 - o **Use SMTP Authentication:** Checked
 - o **User Name:** Your Gmail ID (your-email@gmail.com)
 - o **Password:** Paste the **App Password**
 - o **SMTP Port:** 587
 - o **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

E-mail Notification

SMTP server

smtp.gmail.com

Default user e-mail suffix ?

@gmail.com

Advanced ^  Edited

Use SMTP Authentication ?

User Name

mohdaseemakram19@gmail.com

Password

 Concealed

Use SSL ?

Use TLS

SMTP Port ?

465

Reply-To Address

mohdaseemakram19@gmail.com

Charset

UTF-8

Test configuration by sending test e-mail



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

JDK installations ^ Edited

Add JDK

JDK

Name

jdk

Install automatically ?

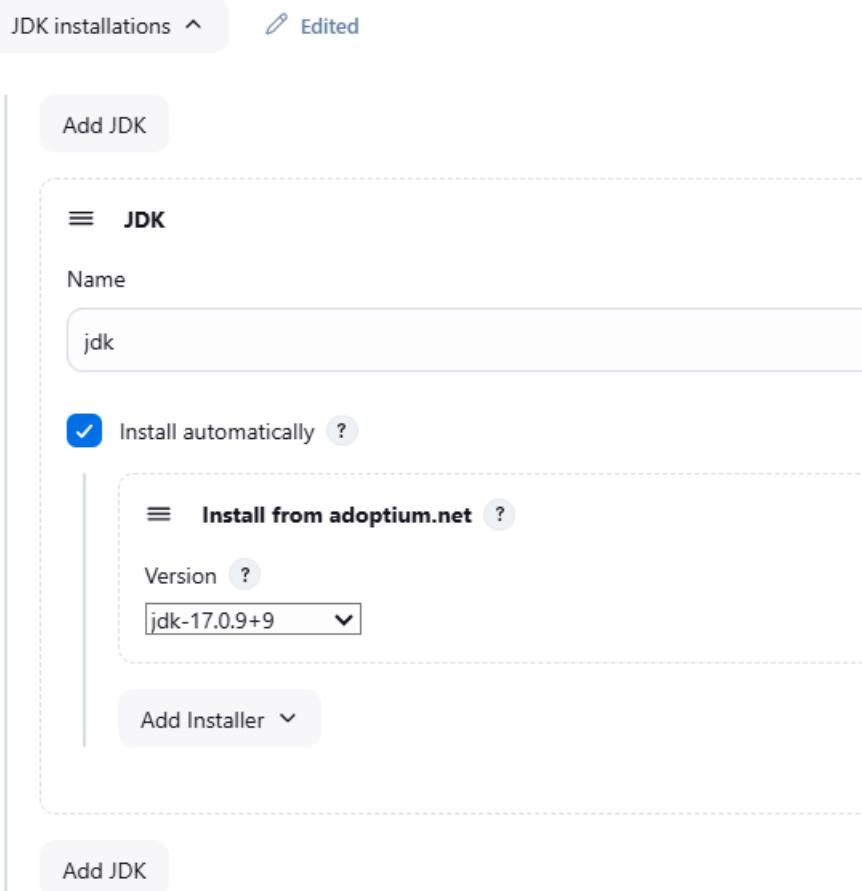
Install from adoptium.net ?

Version ?

jdk-17.0.9+9

Add Installer ▾

Add JDK



CloudAseem

NodeJS installations

NodeJS installations ^ Edited

Add NodeJS

NodeJS

Name
node

Install automatically ?

Install from nodejs.org

Version
NodeJS 17.9.0

For the underlying architecture, if available, force the installation of the 32bit package. Otherwise the build will fail
 Force 32bit architecture

Global npm packages to install
Specify list of packages to install globally -- see npm install -g. Note that you can fix the packages version by using the syntax 'packageName@version'

Global npm packages refresh hours
Duration, in hours, before 2 npm cache update. Note that 0 will always update npm cache
72

Add Installer ▾



Dependency-Check installations

Dependency-Check installations ^ Edited

Add Dependency-Check

Dependency-Check

Name
DC

Install automatically ?

Install from github.com

Version
dependency-check 12.1.0

Add Installer ▾



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

Request an API Key

To request an NVD API Key, please provide your organization name and a valid email address, and indicate your organization type. You must scroll to end of the Terms & Conditions and check "I agree to the Terms of Use" to obtain an API Key. Upon submitting the request, you will receive an email containing a single-use hyperlink that is used to activate your API Key. If your key is not activated within seven days, a new request for an API Key must be submitted.



Organization Name:

Email Address:

Organization Type:

The API is provided "as is" and on an "as-available" basis. The NVD hereby disclaim all warranties of any kind, express or implied, including without limitation the warranties of merchantability, fitness for a particular purpose, and non-infringement. The NVD makes no warranty that the API will be error free or that access thereto will be continuous or uninterrupted.

No Waiver

The NVD's failure to exercise or enforce any right or provision of these Terms of Use shall not constitute waiver of such right or provision.

I agree to the Terms of Use

CloudAseem

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

CloudAseem

This Bingo Pipeline



Stage View

	Declarative: Tool Install	clean workspace	Checkout from Git	Sonarqube Analysis	quality gate	Install Dependencies	OWASP FS SCAN	TRIVY FS SCAN	Docker Build & Push	TRIVY	Deploy to container
Average stage times:	33s	416ms	3s	26s	805ms	47s	4s	6s	2min 22s	27s	889ms
#3 02:46 No Changes	33s	416ms	3s	26s	805ms (passed for 583ms)	47s	4s	6s	2min 22s	27s	889ms
#2 02:45 No Changes											

Permalinks

- Last build (#3), 4 min 38 sec ago
- Last stable build (#1), 21 min ago
- Last successful build (#1), 21 min ago

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.

CloudAseem

There's a new version of SonarQube available. Upgrade to the latest active version to access new updates and features. [Learn More](#)



sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

My Favorites All

Search for project

1 project(s) Perspective: Overall Status Sort by: Name

Quality Gate: Hotstar Passed Last analysis: 5 minutes ago

Reliability (Bugs)

- A rating: 0
- B rating: 0
- C rating: 1
- D rating: 0
- E rating: 0

Security (Vulnerabilities)

- A rating: 0
- B rating: 1
- C rating: 0

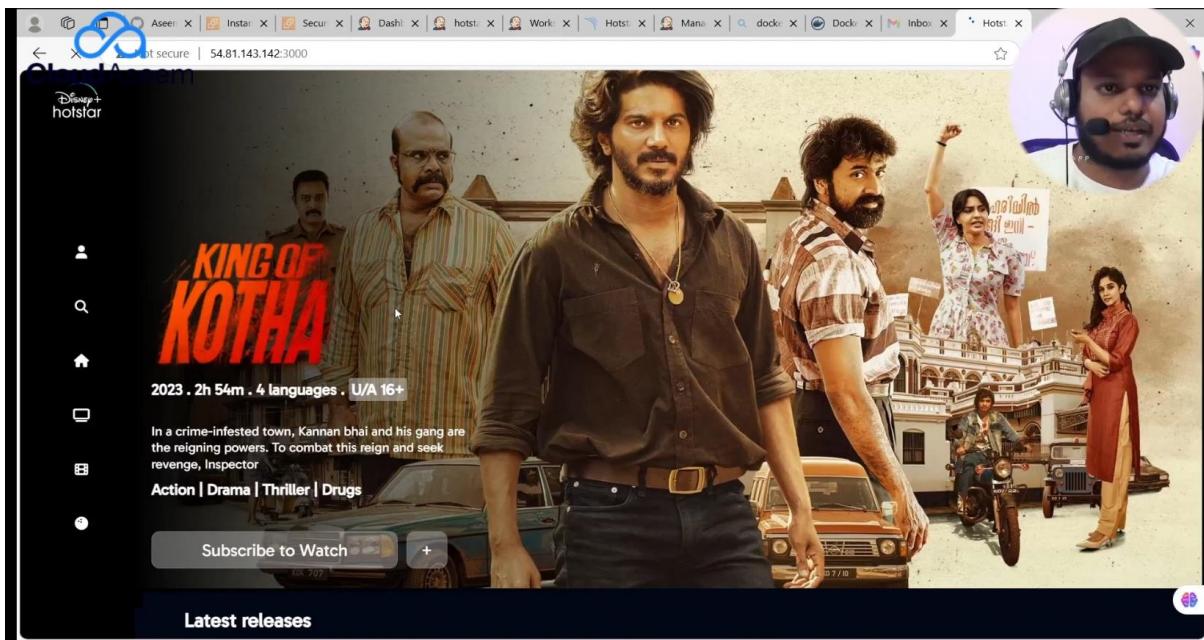
Bugs: C Vulnerabilities: B Hotspots Reviewed: E Code Smells: A Coverage: 0.0% Duplications: 0.0% Lines: 943 JavaScript...

Embedded database should be used for evaluation purposes only
The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.

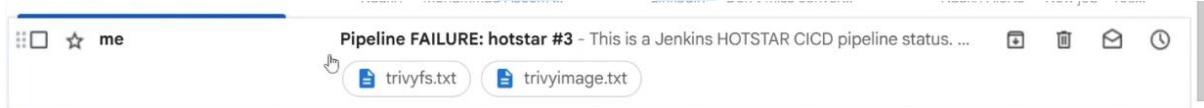
SonarQube™ technology is powered by SonarSource SA
Community Edition - v9.9.8 (build 100196) - [LGPLv3](#) - [Community](#) - [Documentation](#) - [Plugins](#) - [Web API](#)

<public-ip:3000>

Our Application 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			7.20.1	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				7.20.0	7.29.2, 8.0.0-alpha.4	Babel: arbitrary code execution https://avd.aquasec.com/nvd/cve-2023-45133
@babel/traverse	CVE-2023-45133	CRITICAL		1.5.1	1.7.4	axios: axios: Server-Side Request Forgery https://avd.aquasec.com/nvd/cve-2024-39338
axios	CVE-2024-39338	HIGH			1.8.2	axios: Possible SSRF and Credential Leakage via Absolute URL in axios Requests... https://avd.aquasec.com/nvd/cve-2024-27152
	CVE-2025-27152				1.6.0, 0.28.0	axios: exposure of confidential data stored in cookies https://avd.aquasec.com/nvd/cve-2023-45857
	CVE-2023-45857	MEDIUM		1.20.1	1.20.3	body-parser: Denial of Service Vulnerability in body-parser https://avd.aquasec.com/nvd/cve-2024-45590
body-parser	CVE-2024-45590	HIGH		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
braces	CVE-2024-4068			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
cookie	CVE-2024-47764	LOW		7.0.5	7.0.5, 6.0.6	cross-spawn: regular expression denial of service https://avd.aquasec.com/nvd/cve-2024-21538
cross-spawn	CVE-2024-21538	HIGH		3.1.9	3.1.10	The ejs (aka Embedded JavaScript Templates) package before 3.1.10 for, https://avd.aquasec.com/nvd/cve-2024-39883
ejs	CVE-2024-39883	MEDIUM		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
express	CVE-2024-29041				4.20.0, 5.0.0	express: Improper Input Handling in Express Redirects https://avd.aquasec.com/nvd/cve-2024-45736
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

Stage View



Permanent links



Dependency-Check Results

CRITICAL: 1 (2.2%)		HIGH: 20 (44.4%)		SEARCH	
SEVERITY DISTRIBUTION					
File Name	Vulnerability	Severity	Weakness		
+	async:3.2.4 OSSINDEX CVE-2024-39249	Medium	CWE-1333		
+	axios.js NVD CVE-2024-39338	High	CWE-918		
+	axios.js RETIREJS CVE-2025-27152	High			
+	axios.js NVD CVE-2023-45857	Medium	CWE-352		
+	axios.js RETIREJS Versions before 1.6.8 depends on follow-redirects before 1.15.6 which could leak the proxy authentication credentials	Medium			
+	axios.js NVD CVE-2024-39338	High	CWE-918		
+	axios.js RETIREJS CVE-2025-27152	High			
+	axios.js NVD CVE-2023-45857	Medium	CWE-352		
+	axios.js RETIREJS Versions before 1.6.8 depends on follow-redirects before 1.15.6 which could leak the proxy authentication credentials	Medium			

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 #

The screenshot shows the AWS IAM 'Users' section. On the left, there's a sidebar with navigation links like 'Identity and Access Management (IAM)', 'Dashboard', 'Access management', 'Users', 'Roles', 'Policies', 'Identity providers', 'Account settings', and 'Root access management'. The main area is titled 'hotstar-user' with a 'Summary' card. It shows the ARN (arn:aws:iam::050451391050:user/hotstar-user), console access status (Disabled), and creation date (March 20, 2025, 06:14 (UTC+05:30)). Below the summary is a 'Permissions' tab, which lists 'Permissions policies (5)'. It shows two policies attached directly: 'AdministratorAccess' (AWS managed - job function) and 'AmazonEC2FullAccess' (AWS managed). There are also tabs for 'Groups', 'Tags', 'Security credentials', and 'Last Accessed'. A watermark for 'CloudASSET' is overlaid across the bottom of the screenshot.

1) Create New Role using IAM service (Select Usecase - ec2)

2) Add below permissions for the role

- IAM - fullaccess

- VPC - fullaccess

- EC2 - fullaccess

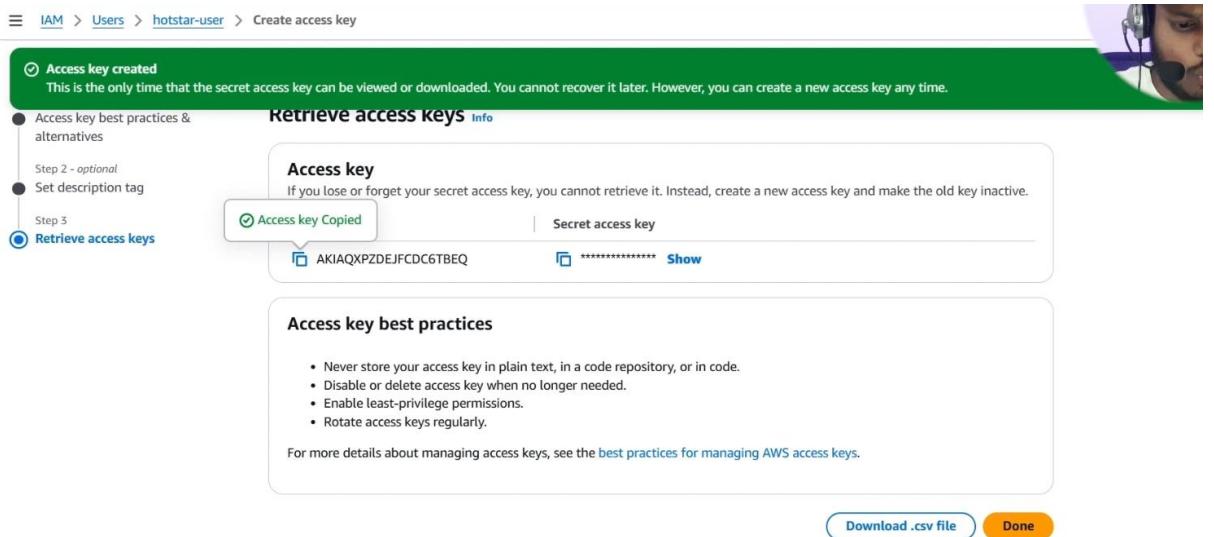
- CloudFormation - fullaccess

- Administrator - acces

3) Create credentials to connect with your aws account

AWS_ACCESS_KEY_ID

AWS_SECRET_ACCESS_KEY



A screenshot of the AWS IAM 'Create access key' page. The top navigation bar shows 'IAM > Users > hotstar-user > Create access key'. A green banner at the top says 'Access key created' with the message: 'This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.' Below the banner, a sidebar lists steps: 'Access key best practices & alternatives', 'Step 2 - optional Set description tag', 'Step 3 Retrieve access keys' (which is selected). The main area is titled 'Retrieve access keys' with an 'Info' link. It contains an 'Access key' section with a note: 'If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.' A button labeled 'Access key Copied' with a checkmark is shown. To its right is a 'Secret access key' field containing 'AKIAQXPZDEJFCDC6TBEQ' with a 'Show' link. Below this is a '*****' placeholder for the secret key. A large callout bubble points to the copied access key. The bottom section is titled 'Access key best practices' with a bulleted list: 'Never store your access key in plain text, in a code repository, or in code.', 'Disable or delete access key when no longer needed.', 'Enable least-privilege permissions.', and 'Rotate access keys regularly.' A link 'For more details about managing access keys, see the best practices for managing AWS access keys.' is provided. At the bottom right are 'Download .csv file' and 'Done' buttons.

Aws configure in VM

```
ubuntu@ip-10-0-1-103:~$ sudo su
root@ip-10-0-1-103:/home/ubuntu# aws configure
AWS Access Key ID [None]: AKIAQXPZDEJFCDC6TBEQ
AWS Secret Access Key [None]: BDPYkv7lG2Um81xawpU5BBxqPHQRtw1WV7r+Wri
Default region name [None]:
Default output format [None]:
root@ip-10-0-1-103:/home/ubuntu#
```

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:

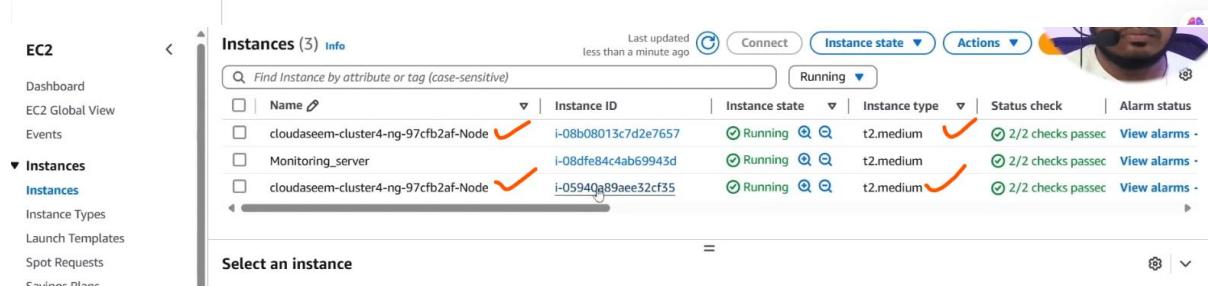
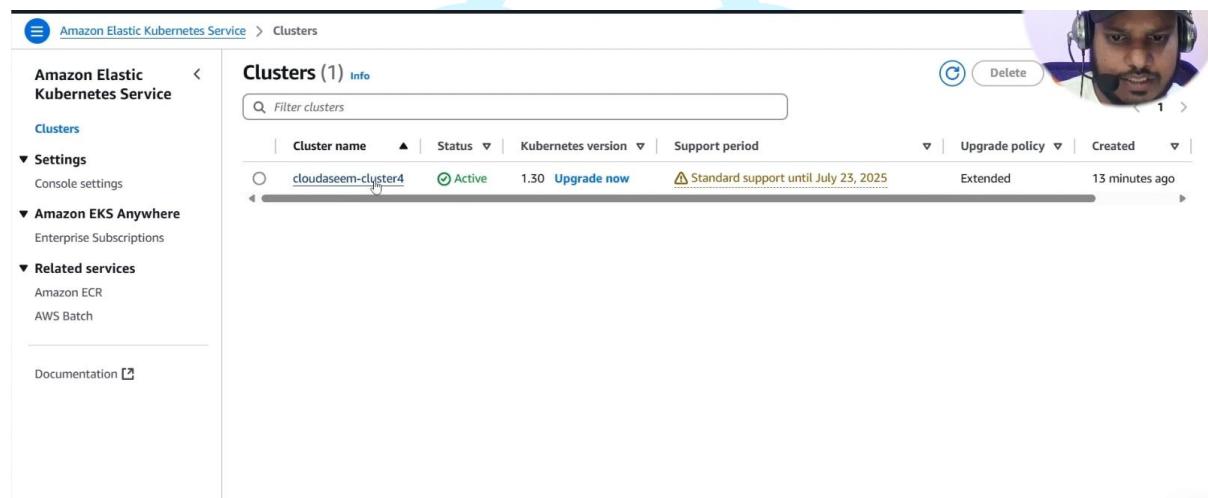
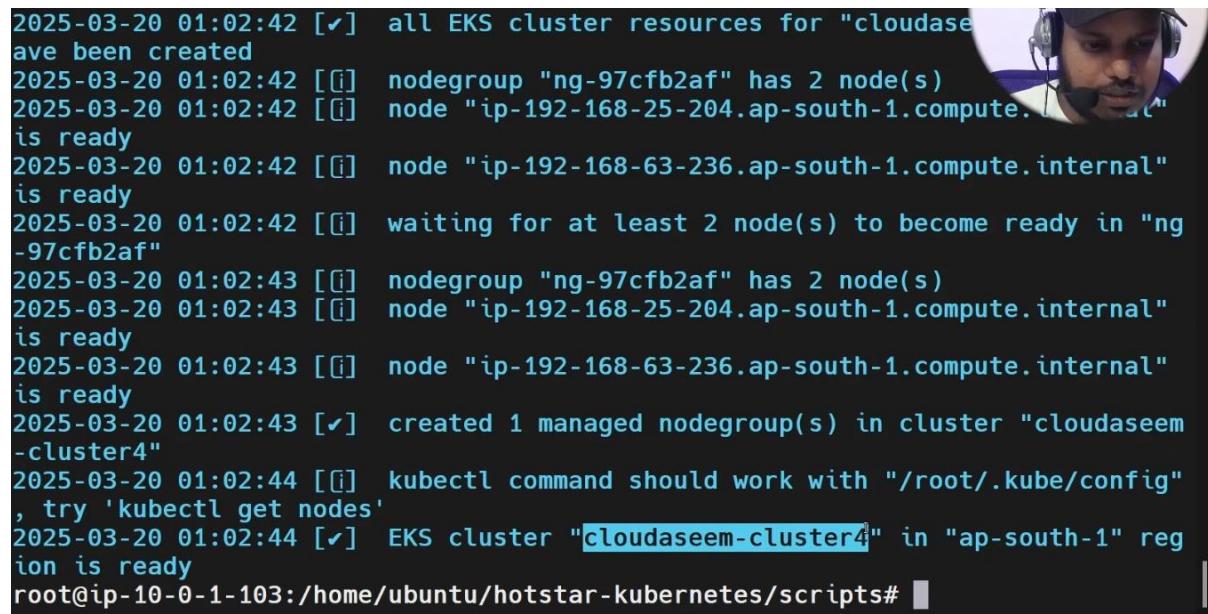

```
eksctl create cluster --name cloudaseem-cluster4 --region ap-south-1 --node-type t2.medium --
zones ap-south-1a,ap-south-1b
```

Option 2 for US ## N. Virginia:


```
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.

```
2025-03-20 01:02:42 [✓] all EKS cluster resources for "cloudaseem-cluster4" have been created
2025-03-20 01:02:42 [i] nodegroup "ng-97cfb2af" has 2 node(s)
2025-03-20 01:02:42 [i] node "ip-192-168-25-204.ap-south-1.compute.internal" is ready
2025-03-20 01:02:42 [i] node "ip-192-168-63-236.ap-south-1.compute.internal" is ready
2025-03-20 01:02:42 [i] waiting for at least 2 node(s) to become ready in "ng-97cfb2af"
2025-03-20 01:02:43 [i] nodegroup "ng-97cfb2af" has 2 node(s)
2025-03-20 01:02:43 [i] node "ip-192-168-25-204.ap-south-1.compute.internal" is ready
2025-03-20 01:02:43 [i] node "ip-192-168-63-236.ap-south-1.compute.internal" is ready
2025-03-20 01:02:43 [✓] created 1 managed nodegroup(s) in cluster "cloudaseem-cluster4"
2025-03-20 01:02:44 [i] kubectl command should work with "/root/.kube/config", try 'kubectl get nodes'
2025-03-20 01:02:44 [✓] EKS cluster "cloudaseem-cluster4" in "ap-south-1" region is ready
root@ip-10-0-1-103:/home/ubuntu/hotstar-kubernetes/scripts#
```



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

The logo features the word "cloudAseem" in a large, lowercase, sans-serif font. The letters "cloud" are in a light blue color, while "Aseem" is in a darker grey. Above the text, there is a stylized graphic element consisting of three light blue, rounded, horizontal shapes that resemble clouds or a stylized infinity symbol.

```
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
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
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

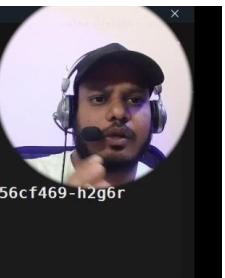
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/hotstar-deployment   2/2      2           2           38s

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>
```



```
root@ip-10-0-1-103:/home/ubuntu/hotstar-kubernetes/K8S# kubectl get pods
NAME                                         READY   STATUS    RESTARTS   AGE
hotstar-deployment-5d956cf469-h2g6r   1/1    Running   0      5m44s
hotstar-deployment-5d956cf469-mbx8n   1/1    Running   0      5m44s
root@ip-10-0-1-103:/home/ubuntu/hotstar-kubernetes/K8S# kubectl ^C
root@ip-10-0-1-103:/home/ubuntu/hotstar-kubernetes/K8S# kubectl delete pod hotstar-deployment-5d956cf469-h2g6r
pod "hotstar-deployment-5d956cf469-h2g6r" deleted
root@ip-10-0-1-103:/home/ubuntu/hotstar-kubernetes/K8S#
```

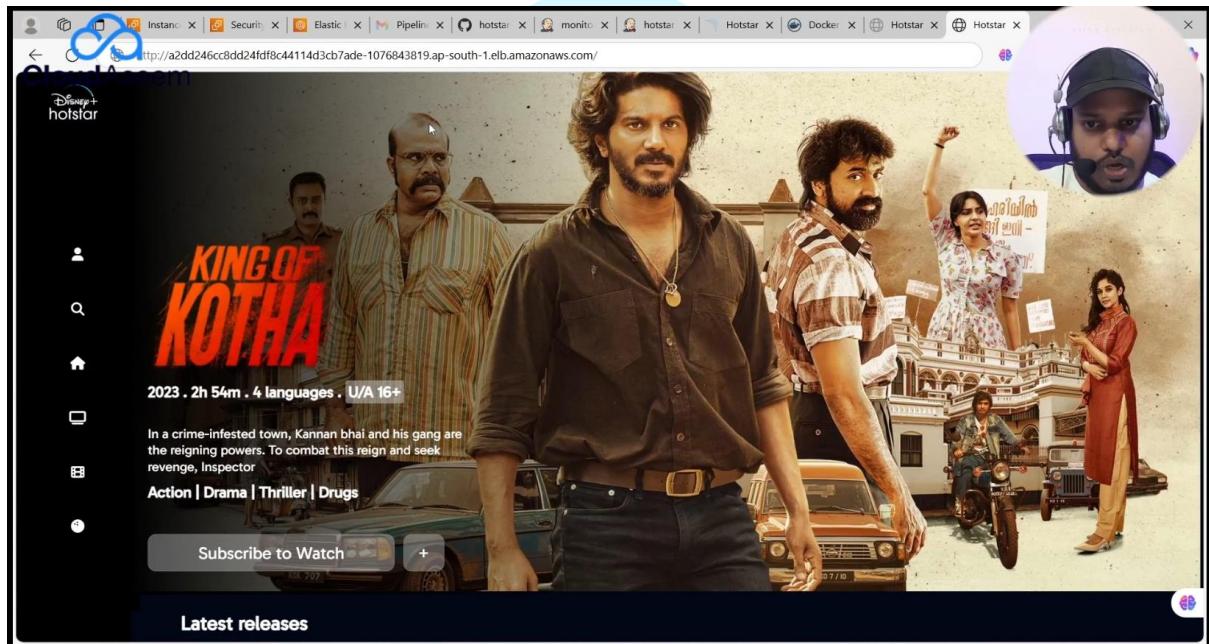
Look for a new pod with a different name but managed by the same Deployment/ReplicaSet.

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

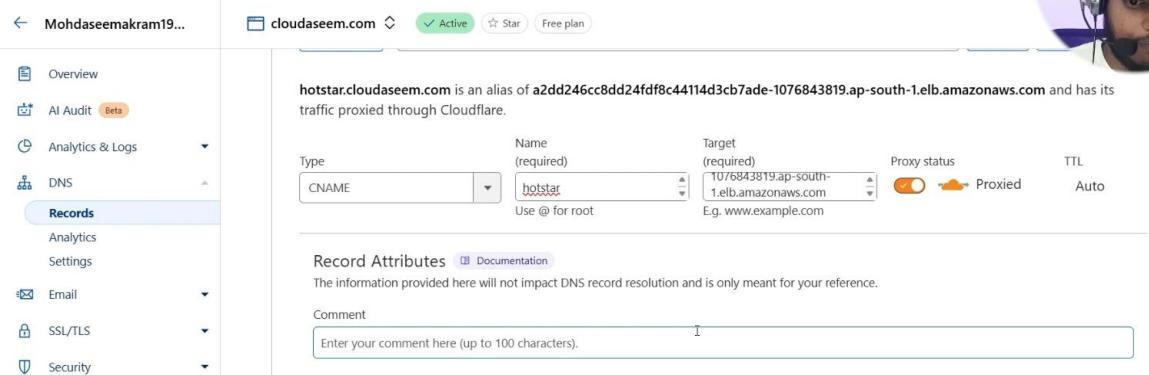
You have successfully Deployed a Hotstar on Kubernetes with Loadbalancer Enabled with AutoHealing.



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

Configure CNAME for Client Access

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



The screenshot shows the Cloudflare DNS interface for the domain `hotstar.cloudaseem.com`. The left sidebar has a 'Records' section selected. A CNAME record is being edited with the name `hotstar` and target `T076843819.ap-south-1.elb.amazonaws.com`. The record is marked as 'Active' and 'Proxied'. A comment field is present but empty.

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

Global credentials (unrestricted)

+ Add Credentials

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description	Actions
Sonar-token	Sonar-token	Secret text	Sonar-token	
github-token	Aseemakram19/******** (github-token)	Username with password	github-token	
docker	aseemakram19/******** (docker)	Username with password	docker	
smtp-gmail	mohdaseemakram19@gmail.com/******** (smtp-gmail)	Username with password	smtp-gmail	
<u>AWS_ACCESS_KEY_ID</u>	AWS_ACCESS_KEY_ID ✓	Secret text ✓	AWS_ACCESS_KEY_ID ✓	
<u>AWS_SECRET_ACCESS_KEY</u>	AWS_SECRET_ACCESS_KEY ✓	Secret text ✓	AWS_SECRET_ACCESS_KEY ✓	

Icon: S M L

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

This project is parameterized ?

Choice Parameter ?

Name ?

Choices ?

Description ?

Plain text [Preview](#)

[Save](#) [Apply](#)

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

Status

Changes

[Build with Parameters](#)

Configure

Delete Pipeline

Full Stage View

Rename

Pipeline Terraform-Eks

This build requires parameters:

[Build](#) [Cancel](#)

Note: Create Key pair to access monitoring server

[EC2](#) > [Key pairs](#) > Create key pair

Create key pair [Info](#)

Key pair

A key pair, consisting of a private key and a public key, is a set of security credentials that you use to prove your identity when connecting to an instance.

Name

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type [Info](#)

RSA

Elliptic Curve Cryptography (ECC)

[Manage personal info](#)

Private key file format

.pem

For use with OpenSSH

.ppk

For use with PuTTY

Tags - optional

No tags associated with the resource.

[Add new tag](#)

You can add up to 50 more tags.

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') {
```

The logo features a stylized blue infinity symbol or cloud-like shape composed of three curved segments. To the left of the symbol, the word "Cloud" is written in a light gray sans-serif font. To the right, the name "Aseem" is written in a larger, bold, light gray sans-serif font.

```
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.'
    }
}
```



The screenshot shows a CI/CD pipeline interface. On the left, there's a sidebar with various options like Status, Changes, Build with Parameters, Configure, Delete Pipeline, Full Stage View, Favorite, Open Blue Ocean, Stages, Rename, and Pipeline Syntax. The main area is titled "monitoringserver" and "monitoring server". It shows a "Stage View" with a timeline and a breakdown of stage times. The timeline has segments for Checkout from Git (764ms), Terraform version (374ms), Terraform init (8s), Terraform validate (6s), Terraform plan (8s), Terraform apply/destroy (27s), and Declarative: Post Actions (107ms). Below the timeline, a box indicates "Average stage times: (full run time: ~54s)" and shows a build status of "#1 06:27 No Changes". A "Permalinks" section shows a link to "Last build (#1), 50 sec ago". At the bottom, there's a "Builds" section with a filter and a timestamp of "Today #1 12:57 AM".

Verify the Monitoring server

The screenshot shows the AWS EC2 Instances page. The left sidebar includes EC2, Dashboard, EC2 Global View, Events, Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, and Reserved Instances. The main area displays "Instances (3) Info" with a table of instance details. The table columns include Name, Instance ID, Instance state, Instance type, Status check, and Alarm status. The instances listed are: Monitoring_server (i-08dfe84c4ab69943d, Running, t2.medium, Initializing, View alarms), hotstar-application (i-041068de5d86c6c1e, Running, t3.xlarge, 3/3 checks passed, View alarms), and hotstar-app (i-0ed6ff61f0a9c8a9c, Stopped, t3.xlarge, -, View alarms).

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 `grafana.sh`, add permission, and execute 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

Inbound rules	Type	Protocol	Port range	Source	Description - optional
sgr-0fda058f97f1313dc	Custom TCP	TCP	3000	Cust... ▾	garafan
sgr-08babbc2dab3c7db0b	SSH	TCP	22	Cust... ▾	ssh access
-	Custom TCP	TCP	9090	Any... ▾	promofile

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

2. Install 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	e73ecf2cc3ecc4bba8f2f82f82c8c51d190283217c6007b41bfe2478be482
prometheus-3.2.1.darwin-arm64.tar.gz	darwin	arm64	106.84 MiB	f71d933be340486e27625e9b80c7be8282a5c8dbb041594ba2e52a4788dcceb
prometheus-3.2.1.linux-amd64.tar.gz	linux	amd64	108.87 MiB	a622e3007c9109a7f470e1433cbd29bf392596715cf7ea8b81b37fa9d26b7be
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

```
v_0.26.0_linux-amd64.tar.gz
--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-45e1-9674-99ed1137b43
b?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=releaseassetproduction%2F20250320%2Fsus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20
250320T012924Z&X-Amz-Expires=300&X-Amz-Signature=2d54029d913c128ab7b481bfaba38b65fbef005b05aa20b986af70d939579869GX-Amz-SignedHeade
rs=host&response-content-disposition=attachment%3B%20filename%3Dblackbox_exporter-0.26.0.linux-amd64.tar.gz&response-content-type=ap
plication%2Foctet-stream [following]
--2025-03-20 01:29:24-- https://objects.githubusercontent.com/github-production-release-asset-2e65be/41964498/37e20444-c65c-45e1-9
674-99ed1137b43b?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=releaseassetproduction%2F20250320%2Fsus-east-1%2Fs3%2Faws4_reques
t&X-Amz-Date=20250320T012924Z&X-Amz-Expires=300&X-Amz-Signature=2d54029d913c128ab7b481bfaba38b65fbef005b05aa20b986af70d939579869GX-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.cloudseem.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: 13.232.124.65:9115 # 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=WARNING 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.

The screenshot shows the Grafana interface with the left sidebar open. Under 'Connections', 'Data sources' is selected. A new connection is being configured with the name 'prometheus'. The 'Connection' section contains the 'Prometheus server URL' input field, which has the value 'http://13.233.124.65:9090/'. The 'Authentication' section shows 'No Authentication' selected. A large blue cloud icon is overlaid on the bottom right of the screen.

Add

Go to Dashboards → Import.

Enter Dashboard ID: 13659 (Prometheus Blackbox Exporter).

Click Load.

Select Prometheus as the Data Source.

Click Import.

The screenshot shows the Grafana interface displaying the 'HTTP Probe Overview' dashboard. The table lists three probe results:

Instance	Status	Code	SSL	TLS Version	SSL Cert Expiry (days)	Probe Duration (s)	DNS Lookup Duration (s)
http://54.81.143.142:3000	UP	200	OK	TLS 1.3	57	0.37	0.000
http://prometheus.io	UP	200	OK	TLS 1.3	70	0.29	0.001
https://hotstar.cloudaseem.com:443	UP	200	OK	TLS 1.3	400 ms	0.04	0.001

Below the table, a chart titled 'HTTP Probe Duration' shows the distribution of probe durations for the three targets. The x-axis represents the duration in milliseconds, and the y-axis represents the count of probes. The legend indicates the colors for each target: green for http://54.81.143.142:3000, yellow for http://prometheus.io, and blue for https://hotstar.cloudaseem.com:443.

Step - 4 :

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

Pipeline monitoringserver

This build requires parameters:

action

this for monitoring server setup

destroy

Build Cancel

Status Changes Build with Parameters Configure Delete Pipeline Full Stage View Favorite Open Blue Ocean Stages Rename Pipeline Syntax

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 --region 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-cluster4"
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 of Kind Service or Ingress
```

```
2025-03-20 01:56:53 [i] waiting for CloudFormation stack "eksctl-cloudaseem-cluster4-nodegroup-97cfb2af"
2025-03-20 01:57:23 [i] waiting for CloudFormation stack "eksctl-cloudaseem-cluster4-nodegroup-97cfb2af"
2025-03-20 01:58:10 [i] waiting for CloudFormation stack "eksctl-cloudaseem-cluster4-nodegroup-97cfb2af"
2025-03-20 01:59:15 [i] waiting for CloudFormation stack "eksctl-cloudaseem-cluster4-nodegroup-97cfb2af"
2025-03-20 02:00:45 [i] waiting for CloudFormation stack "eksctl-cloudaseem-cluster4-nodegroup-97cfb2af"
2025-03-20 02:02:27 [i] waiting for CloudFormation stack "eksctl-cloudaseem-cluster4-nodegroup-97cfb2af"
2025-03-20 02:03:41 [i] waiting for CloudFormation stack "eksctl-cloudaseem-cluster4-nodegroup-97cfb2af"
2025-03-20 02:04:40 [i] waiting for CloudFormation stack "eksctl-cloudaseem-cluster4-nodegroup-97cfb2af"
2025-03-20 02:04:41 [i] will delete stack "eksctl-cloudaseem-cluster4-cluster"
2025-03-20 02:04:42 [✓] all cluster resources were deleted
root@ip-10-0-1-103:/home/ubuntu#
```

✓ 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) - <https://www.youtube.com/@clouddevopswithaseem>

🔗 LinkedIn: [Mohammed Aseem Akram](https://www.linkedin.com/in/mohammed-aseem-akram) - www.linkedin.com/in/mohammed-aseem-akram

🔗 GitHub: [Aseemakram19](https://github.com/Aseemakram19)

👉 Stay tuned for more DevOps insights! 🚀

#Cloud #DevOps #Technology #Kubernetes