

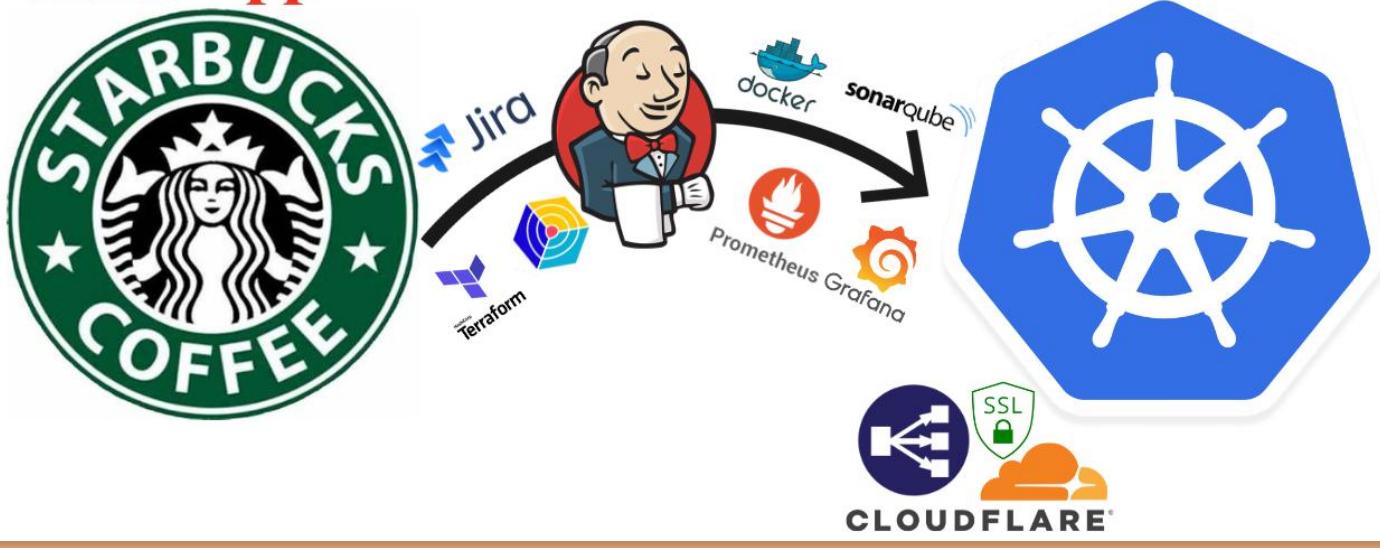
StarbucksClone App — End-to-End Secure & Scalable Deployment with DevSecOps & Jenkins !

Thrilled to showcase our latest project — a StarbucksClone Application engineered Starbucks Clone App 🔒 End-to-End Secure & ⚖️ Scalable Deployment with DevSecOps with a DevSecOps CI/CD pipeline and powered by Jenkins Parameterise Build to deliver fully automated, secure, and scalable deployments.

DevSecOps Project



Production Deployment of StarBucks CloneApp on Kubernetes



 Follow me for projects Links :

 YouTube: <https://youtu.be/1FNhSda0sV4> - @clouddevopswithaseem

 Project GitHub Repo: <https://github.com/Aseemakram19/starbucks-kubernetes.git>

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

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



Project Agenda

□ Infrastructure Setup

- Jenkins Server — For CI/CD pipeline automation
- SonarQube Server — For code quality and security checks
- Monitoring Server — For system observability and alerting

□ Core Components & Toolchain

- Docker & Kubernetes
→ Containerized, scalable application deployment with orchestration
- Jenkins CI/CD Pipelines
→ Reusable, standardized, and consistent deployment pipelines
- SonarQube & Trivy
→ Code quality analysis & vulnerability scanning for secure releases
- Prometheus & Grafana
→ Real-time metrics monitoring and visual dashboards
- Gmail Email Alerts
→ Collaboration-driven notifications and incident alerts
- Parameterized Environment Orchestration
→ On-demand infrastructure setup and teardown (Dev/Test/Prod)

| 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 |

We are going to cover this 3 responsibilities in this project :

Roles & Responsibilities

Platform Engineer

- Designs and builds reusable infrastructure as code (IaC)
- Integrates CI/CD tools, secrets management, and container registries
- Enables self-service environments and developer portals
- Focuses on security, compliance, and platform governance

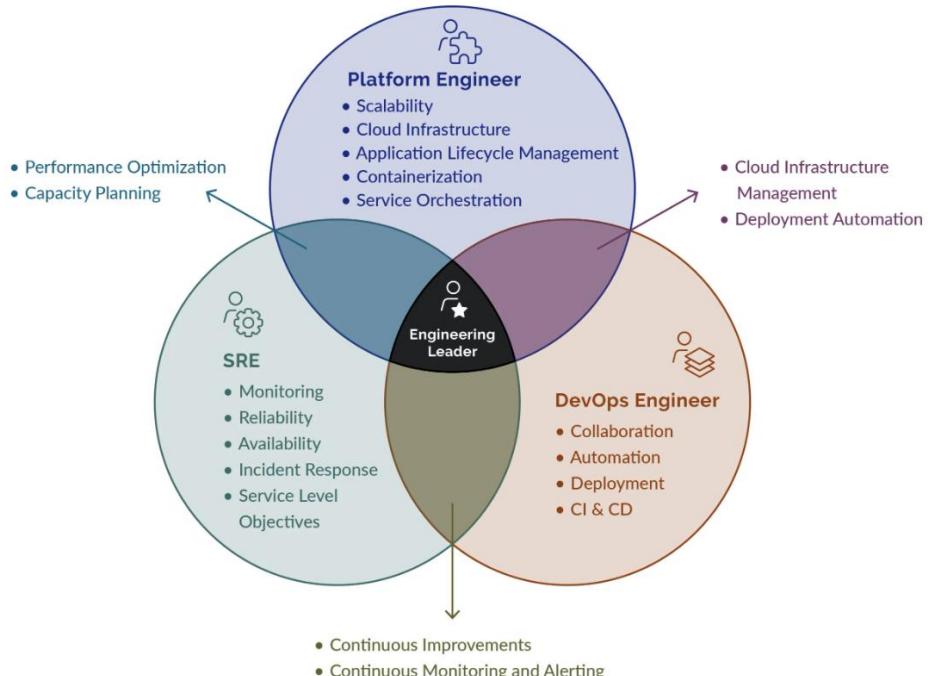
SRE (Site Reliability Engineer)

- Ensures system reliability, availability, and performance
- Implements monitoring, alerting, and incident response workflows
- Automates failover, backup, and recovery strategies
- Defines SLOs/SLIs and tracks error budgets for services

DevOps Engineer

- Builds and maintains CI/CD pipelines with Jenkins
- Automates testing, security scans, and artifact deployment
- Works on infrastructure provisioning and environment management
- Bridges development and operations for faster delivery cycles

DevOps Vs SRE Vs Platform Engineering



Application Software Architecture

Monolithic Architecture

In this project, the **Starbucks Clone App** is designed using a **Monolithic Architecture**.

What is Monolithic Architecture?

A **monolithic application** is built as a single, unified codebase that handles multiple functionalities — from UI to business logic to data access — all bundled and deployed together as one unit.

Key Characteristics

-  **Single Codebase** — All features and components are part of one project
-  **Unified Deployment** — Deployed as one artifact (e.g., a JAR, WAR, or Docker container)
-  **Tightly Coupled Components** — Modules interact through direct function calls

- 📦 **Shared Resources** — Uses shared memory/database, often simplifying data management

✓ Advantages

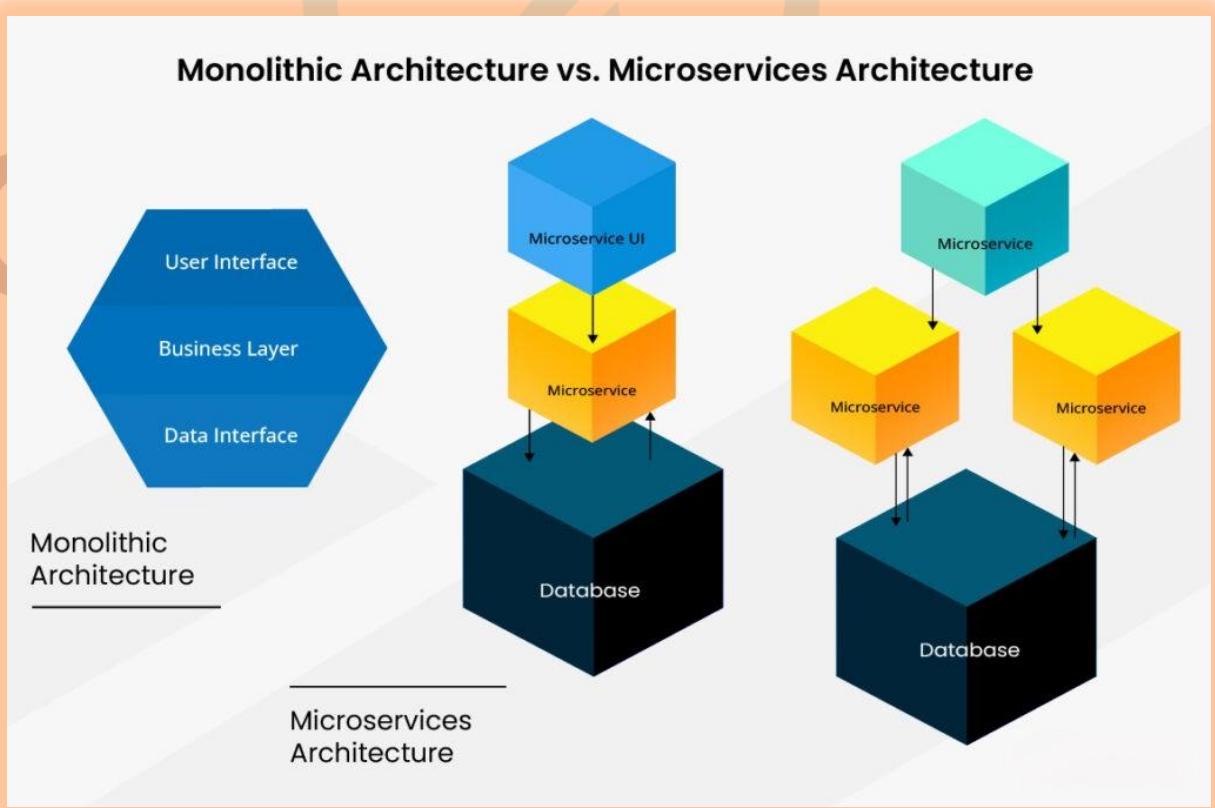
- ⚡ **Faster Development (initially)** — Simple to build and get started
- 🔧 **Easier Testing** — No inter-service communication to mock
- 📦 **Simplified Deployment** — One deployment package to manage

❗ Challenges

- 🚧 **Scalability Limitations** — Can't scale individual components independently
- 🔴 **Single Point of Failure** — A bug in one part can crash the entire system
- ♻️ **Slow Development at Scale** — Harder to maintain as the codebase grows
- 🌐 **Limited Flexibility** — Difficult to adopt new tech stacks for specific modules

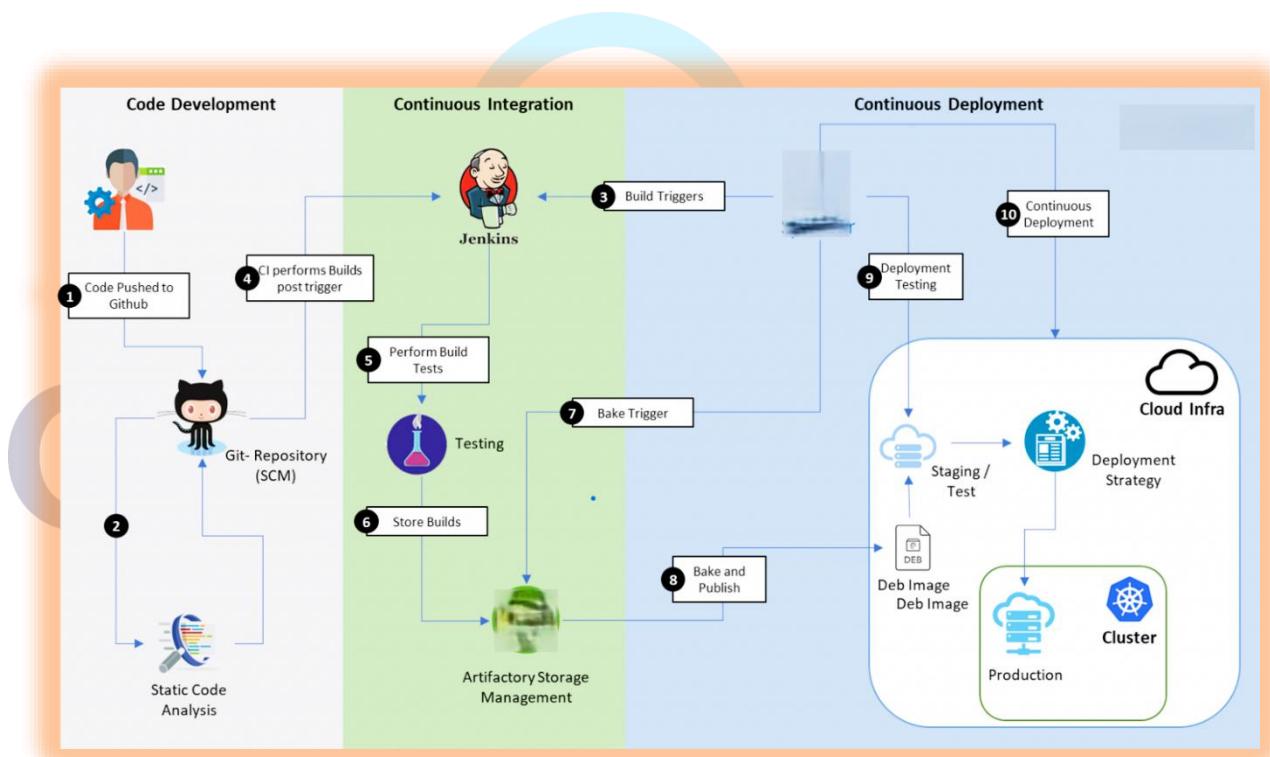
💡 Why Monolithic for This Project?

- 👥 **Suitable for small teams or early-stage MVPs**
- ⌚ **Faster initial setup and deployment using Docker and Kubernetes**
- 🔒 **Easier to integrate security and monitoring tools** across a single service

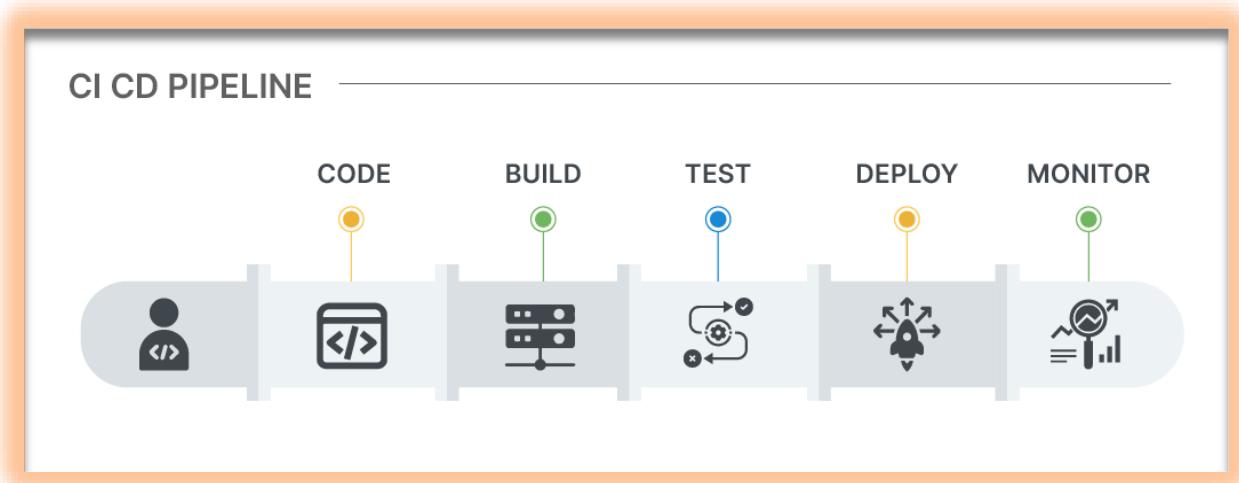


CI/CD pipeline diagram

- 💻 Code Development starts when a developer pushes code to GitHub.
- 📝 Static Code Analysis runs automatically to check code quality and security.
- 🚀 Jenkins CI gets triggered, pulls the latest code, and performs automated build and testing.
- 📦 Successfully built artifacts are stored in Artifactory, and a "bake" process packages them.
- 👉 The baked image is deployed to a Staging/Test environment for deployment testing.
- 🌐 After validation, the app is automatically deployed to production clusters in the cloud infrastructure.



Pipeline :



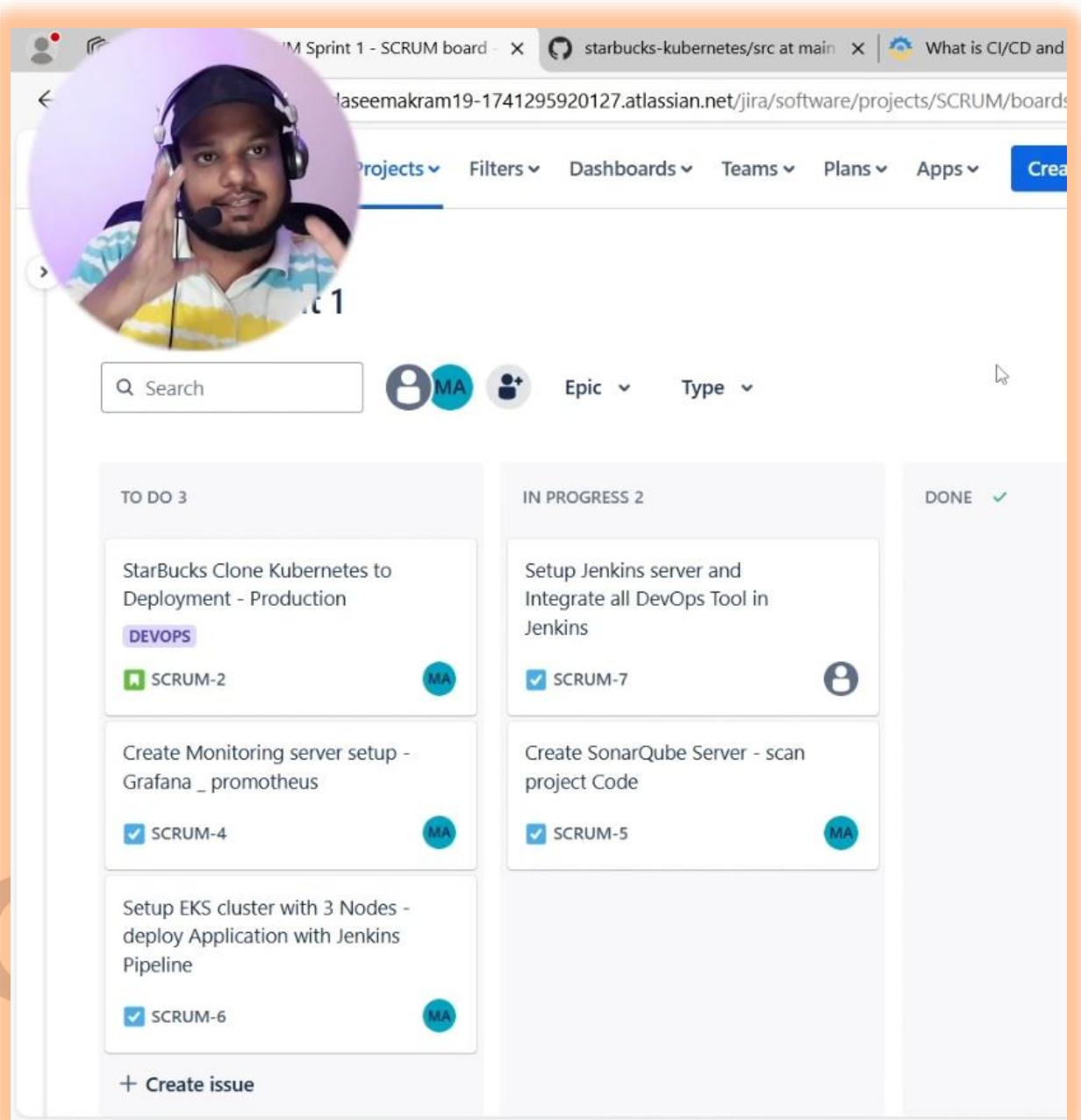
JIRA PROJECT MANAGEMENT TOOL :

What is Jira?

Jira is a powerful **project management** and **issue-tracking** tool developed by Atlassian. It is widely used by **software development teams** to plan, track, release, and maintain software products.

What You Can Do with Jira:

1.  **Plan Work**
 - Create **Epic**s, **User Stories**, **Tasks**, and **Bugs**
 - Prioritize them in a **Backlog**
2.  **Track Progress**
 - Use **Scrum** or **Kanban boards** to visually track work
 - Move tasks through statuses like *To Do* → *In Progress* → *Done*
3.  **Collaborate**
 - Add **comments**, attach files, and tag teammates
 - Link issues to code, builds (CI/CD), and documentation
4.  **Report & Analyze**
 - Use **Dashboards** for real-time updates
 - Generate **burndown charts**, **velocity reports**, and more
5.  **Test & Release**
 - Integrate with tools like **Jenkins**, **GitHub**, **SonarQube**, **Xray**, **Zephyr**
 - Track releases and deployment pipelines



TO DO 3

- StarBucks Clone Kubernetes to Deployment - Production
DEVOPS
SCRUM-2
- Create Monitoring server setup - Grafana _ prometheus
SCRUM-4
- Setup EKS cluster with 3 Nodes - deploy Application with Jenkins Pipeline
SCRUM-6

IN PROGRESS 2

- Setup Jenkins server and Integrate all DevOps Tool in Jenkins
SCRUM-7
- Create SonarQube Server - scan project Code
SCRUM-5

DONE ✓

+ Create issue

Task Description:

Implement the full CI/CD flow as per the shared architecture diagram:

1.  **Code Commit & Analysis**
 - Developers push code to GitHub.
 - Integrate **Static Code Analysis** using tools like SonarQube.
2.  **Continuous Integration (CI)**
 - Configure **Jenkins** to trigger builds on Git push.
 - Run unit and integration tests.
 - Store build artifacts in **Artifactory**.
3.  **Artifact Baking**
 - Bake Docker/DEB images post successful tests.

- Publish them to the artifact repository.
- 4.  **Continuous Deployment (CD)**
 - Deploy artifacts to **Staging** using Jenkins and Kubernetes.
 - Run automated **Deployment Testing**.
 - If successful, proceed to deploy to **Production**.
- 5.  **Monitoring & Notification**
 - Ensure all stages are monitored.
 - Configure **email alerts** and **build/test notifications** to relevant stakeholders.

Awesome! 🙌

🔧 Let's Start the Implementation of the Starbucks Clone App 

  Proceeding to the Lab Setup...

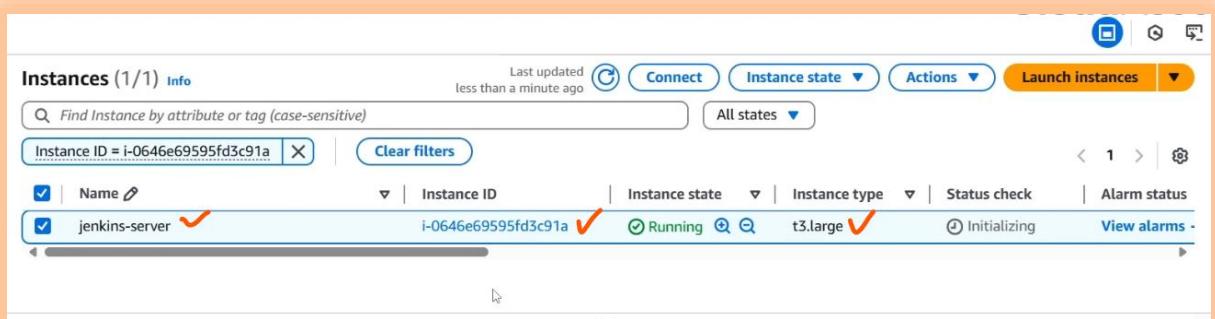
We'll begin with:

1.  **Provisioning Jenkins Server**
2.  **Setting up SonarQube for Code Quality & Security**
3.  **Launching Monitoring Stack (Prometheus & Grafana)**
4.  **Preparing Kubernetes Cluster for Deployment**
5.  **Integrating CI/CD Pipeline with DevSecOps Practices**

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

I. Configure Infrastructure Jenkins Server 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:** <Jenkins server>
 - **AMI (Amazon Machine Image):** Ubuntu Server 22.04 LTS (HVM), SSD Volume Type
 - **Instance Type:** t3.large (2 vCPUs, 8 GB RAM)
 - **Key Pair:** Select an existing key pair or create a new one.
 - **Storage:** Default (e.g., 20GB GP3 SSD, adjust as needed).



The screenshot shows the AWS EC2 Instances page. At the top, there are buttons for 'Connect', 'Instance state', 'Actions', and 'Launch instances'. Below the header, a search bar says 'Find Instance by attribute or tag (case-sensitive)' and a dropdown says 'All states'. There is a 'Clear filters' button. The main table has columns for 'Name', 'Instance ID', 'Instance state', 'Instance type', 'Status check', and 'Alarm status'. One row is selected, showing 'jenkins-server' as the name, 'i-0646e69595fd3c91a' as the Instance ID, 'Running' as the state, 't3.large' as the type, and 'Initializing' as the status check.

To assign an Elastic IP (EIP) to a Jenkins server on AWS, follow these steps: Timestamp : 16:09

1. Allocate an Elastic IP:

- Go to the **EC2 Dashboard** in AWS Management Console.
- Under **Network & Security**, click **Elastic IPs**.
- Click **Allocate new address** and choose your region.

2. Associate Elastic IP with Jenkins Server:

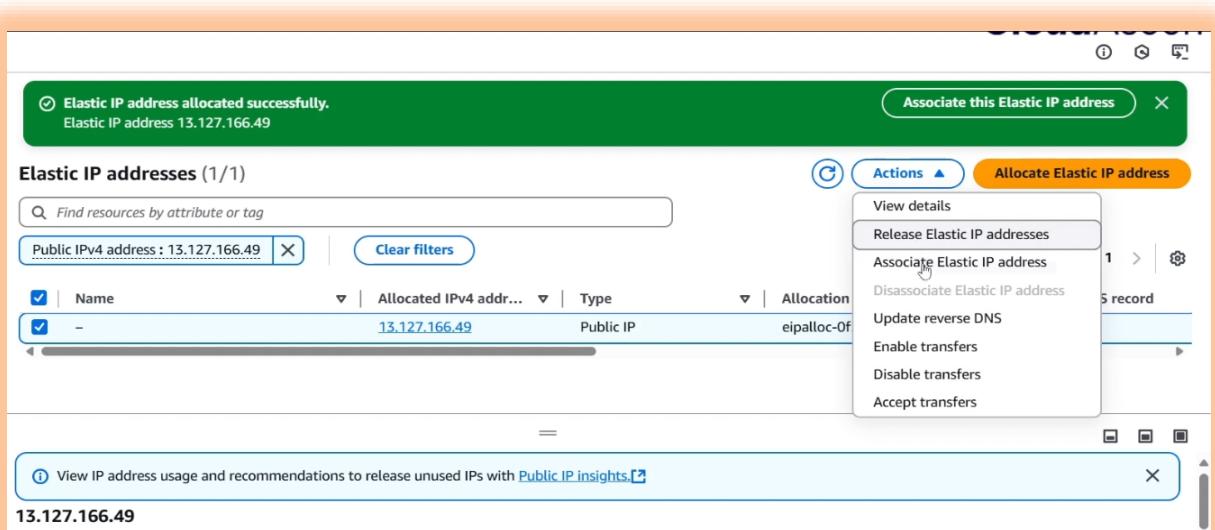
- In the **Elastic IPs** section, select the allocated EIP.
- Click **Actions**, then **Associate Elastic IP address**.
- Select the **Jenkins server instance** (EC2 instance) and click **Associate**.

3. Verify Association:

- Check if the EIP is now associated with the Jenkins EC2 instance.
- Ensure Jenkins is accessible via the new IP.

Benefits:

- **Static IP:** Provides a fixed IP address that won't change even if you stop/start your EC2 instance.
- **Easy Access:** Direct and consistent access to Jenkins from external networks.
- **High Availability:** In case of instance failure, the EIP can be reassigned to a new instance quickly.



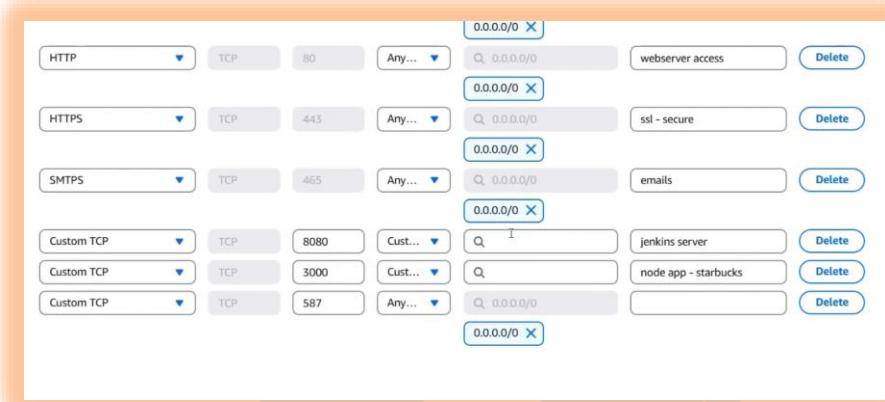
The screenshot shows the AWS Elastic IP Addresses page. A green banner at the top says 'Elastic IP address allocated successfully.' followed by the IP address '13.127.166.49'. Below the banner, there is a 'Associate this Elastic IP address' button. The main table has columns for 'Name', 'Allocated IPv4 addr...', 'Type', and 'Allocation'. One row is selected, showing '-' as the name, '13.127.166.49' as the allocated IP, 'Public IP' as the type, and 'eipalloc-Of...' as the allocation. On the right side, a context menu is open over the selected EIP address, listing options: 'View details', 'Release Elastic IP addresses', 'Associate Elastic IP address' (which is highlighted), 'Disassociate Elastic IP address', 'Update reverse DNS', 'Enable transfers', 'Disable transfers', and 'Accept transfers'. At the bottom left, there is a link to 'View IP address usage and recommendations to release unused IPs with Public IP insights.'

2. Configure Security Group

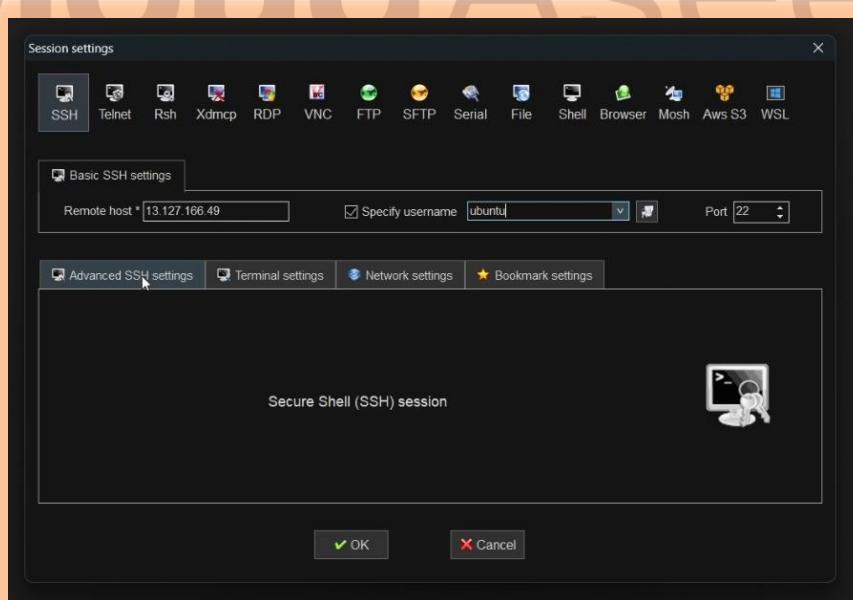
Create or modify a security group to allow the following ports:

| 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 |

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



- Launch & Connect with Mobaxterm App** Click **Launch Instance**. Once the instance is running, connect using:



Create GITHUB Repo and Push code

the step-by-step guide to clone the GitHub repository, make changes, and push it to your own GitHub account, ensuring it's private:

1. Clone the existing GitHub repository:

```
git clone https://github.com/Aseemakram19/starbucks-kubernetes.git
```

2. Navigate to the project directory:

```
cd starbucks-kubernetes/
```

3. Remove the current origin (this is necessary to unlink the original repository):

```
git remote remove origin
```

4. Verify the remote is removed (optional):

```
git remote -v
```

5. Create a new repository on GitHub:

- Go to your [GitHub](#) account.
- Create a new repository with the desired name (e.g., starbucks-kubernetes-project).
- Ensure to make it **Private** when creating the repository.

6. Add your new GitHub repository as the origin:

```
git remote add origin https://github.com/youraccount/starbucks-kubernetes-project.git
```

7. Push the repository to your GitHub account:

```
git push -u origin main
```

8. Enter GitHub credentials:

- When prompted, enter your GitHub **username**.
- Use your **GitHub Personal Access Token** as the password (GitHub now uses tokens instead of passwords for enhanced security).

{ Steps to Create a GitHub Personal Access Token (PAT):

1. Log in to GitHub:

- Open [GitHub](#) and log in to your account.

2. Access Your Settings:

- Click on your **profile icon** in the top right corner.
- Select **Settings** from the dropdown menu.

3. Navigate to Developer Settings:

- On the left sidebar, scroll down and click on **Developer settings**.

4. Go to Personal Access Tokens:

- In the Developer settings section, click on **Personal access tokens**.
- Choose **Tokens (classic)**.

5. Generate New Token:

- Click on **Generate new token** (or **Generate new token (classic)**, depending on the UI).
- Give the token a **descriptive name**, e.g., "Jenkins access token" or "CLI operations".
- Set an **expiration date** (optional), but you can choose **No expiration** if you prefer the token to remain valid indefinitely.

6. Select Scopes:

- Choose the required **scopes** for your token, based on the permissions you need. For example:
 - repo: Full control of private repositories (push, pull, delete, etc.).

- workflow: For managing GitHub Actions workflows.
- admin:org: For managing organization settings.
- read:user: For accessing public user data, etc.

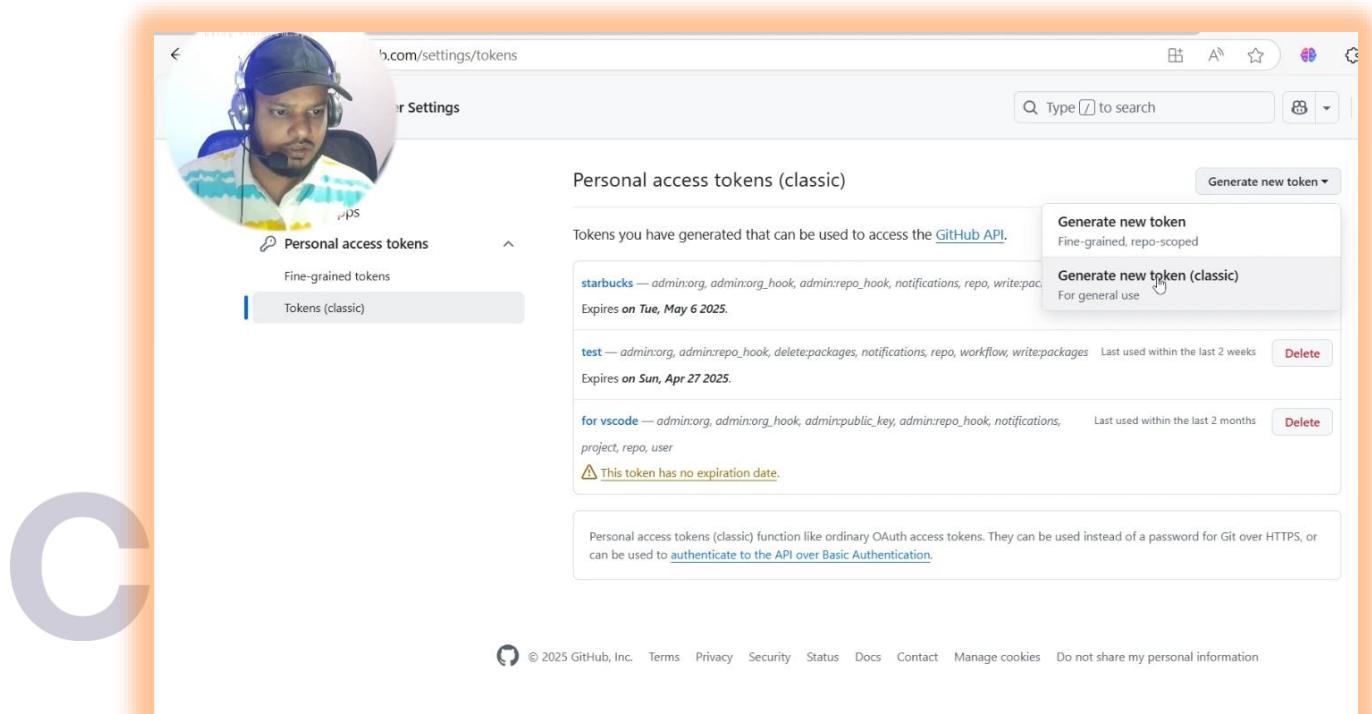
For general Git operations (clone, push, pull, etc.), the repo scope is usually sufficient.

7. Generate Token:

- Once you have selected the necessary scopes, click **Generate token** at the bottom.
- Copy the **token** shown on the screen. **This is the only time it will be visible.**

Important Notes:

- **Save your token securely**, as you won't be able to see it again after this step.
- **Use the token** in place of your GitHub password when performing Git operations, such as cloning or pushing to a repository.
- If you use the token in a CI/CD pipeline or for Jenkins, you can store it securely in the credentials store or environment variables to avoid using it directly in scripts.



The screenshot shows the GitHub settings page for personal access tokens. On the left, there's a profile picture of a person wearing a headset. Below the profile picture, there are two tabs: "Personal access tokens" (which is active) and "Tokens (classic)". The main content area is titled "Personal access tokens (classic)" and contains a table with three rows of tokens. The first token is named "starbucks" and has scopes like "admin:org", "admin:org_hook", etc., and expires on May 6, 2025. The second token is named "test" and has scopes like "admin:org", "admin:repo_hook", etc., and expires on April 27, 2025. The third token is named "vscode" and has scopes like "admin:org", "admin:public_key", etc., and has no expiration date. There are "Delete" buttons next to each token entry. At the top right, there are buttons for "Generate new token" and "Generate new token (classic)". 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](#)".

}

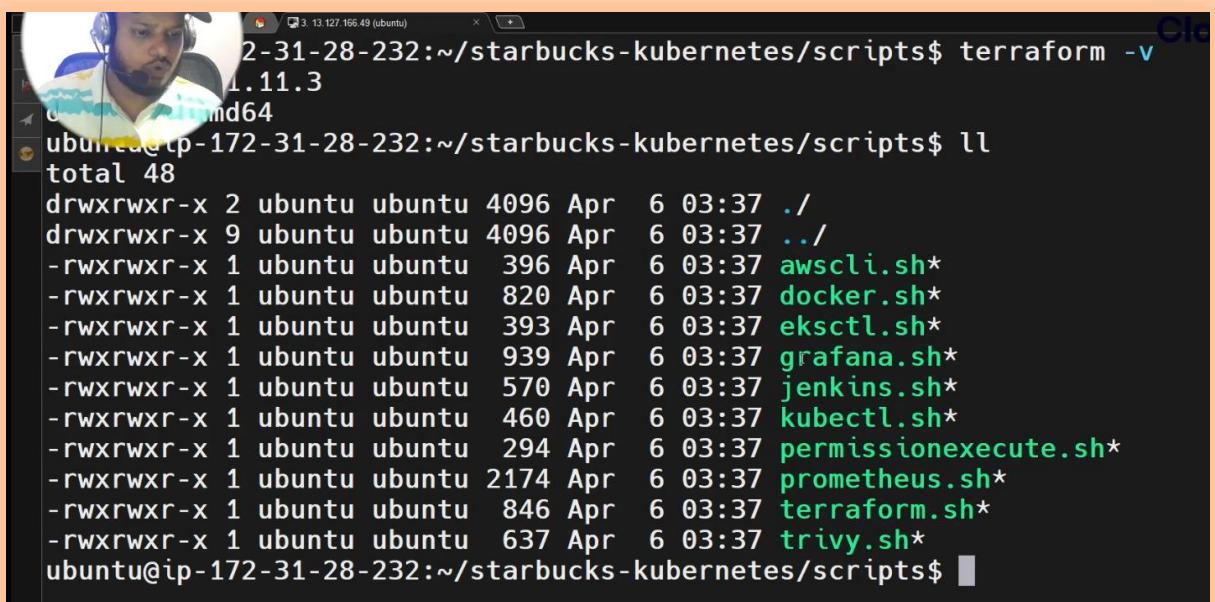
Enhancements:

- Set a default branch (if using main instead of master):
git branch -M main
- Set the repository to private:
 - When creating the repository on GitHub, ensure you select the **Private** option. If the repository is public and you want to change it, you can adjust it in the repository settings on GitHub.

cd [starbucks-kubernetes/scripts/](#)

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

Install Tools



```
2-31-28-232:~/starbucks-kubernetes/scripts$ terraform -v
Terraform v1.11.3
on linux_amd64

ubuntu@ip-172-31-28-232:~/starbucks-kubernetes/scripts$ ll
total 48
drwxrwxr-x 2 ubuntu ubuntu 4096 Apr  6 03:37 .
drwxrwxr-x 9 ubuntu ubuntu 4096 Apr  6 03:37 ..
-rwxrwxr-x 1 ubuntu ubuntu  396 Apr  6 03:37 awscli.sh*
-rwxrwxr-x 1 ubuntu ubuntu  820 Apr  6 03:37 docker.sh*
-rwxrwxr-x 1 ubuntu ubuntu  393 Apr  6 03:37 eksctl.sh*
-rwxrwxr-x 1 ubuntu ubuntu  939 Apr  6 03:37 grafana.sh*
-rwxrwxr-x 1 ubuntu ubuntu  570 Apr  6 03:37 jenkins.sh*
-rwxrwxr-x 1 ubuntu ubuntu  460 Apr  6 03:37 kubectl.sh*
-rwxrwxr-x 1 ubuntu ubuntu  294 Apr  6 03:37 permissionexecute.sh*
-rwxrwxr-x 1 ubuntu ubuntu 2174 Apr  6 03:37 prometheus.sh*
-rwxrwxr-x 1 ubuntu ubuntu  846 Apr  6 03:37 terraform.sh*
-rwxrwxr-x 1 ubuntu ubuntu  637 Apr  6 03:37 trivy.sh*
ubuntu@ip-172-31-28-232:~/starbucks-kubernetes/scripts$
```

CloudAseem

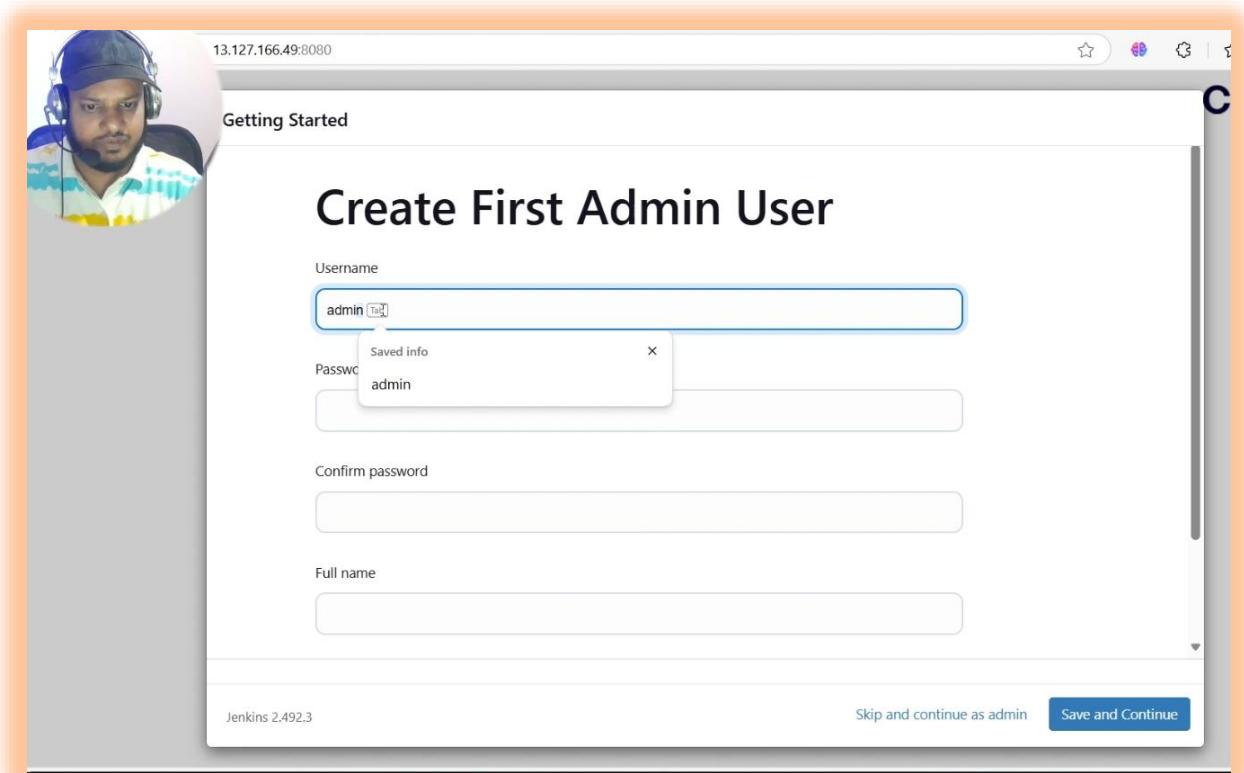
Terraform v1.11.2
on linux_amd64

- 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
```



The screenshot shows the Jenkins 'Create First Admin User' configuration page. At the top, it says 'Getting Started' and 'Create First Admin User'. The 'Username' field contains 'admin'. Below it, a dropdown menu shows 'Saved info' and 'admin'. The 'Confirm password' and 'Full name' fields are empty. At the bottom, it says 'Jenkins 2.492.3' and has 'Skip and continue as admin' and 'Save and Continue' buttons.

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



The screenshot shows the Jenkins 'Customize Jenkins' configuration page. It says 'Plugins extend Jenkins with additional features to support many different needs.' Two options are shown: 'Install suggested plugins' (selected) and 'Select plugins to install'. Both options have descriptive text below them. At the bottom, it says 'Jenkins 2.414.1'.

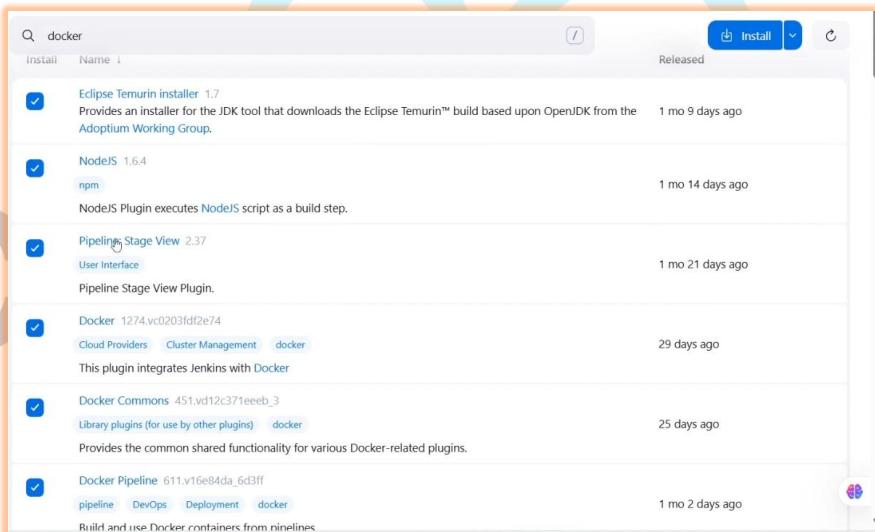
- Create a user click on save and continue.
- Jenkins Getting Started Screen.
- Follow the setup wizard and install recommended plugins.

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

Goto Manage Jenkins → Plugins → Available Plugins →

Install below plugins

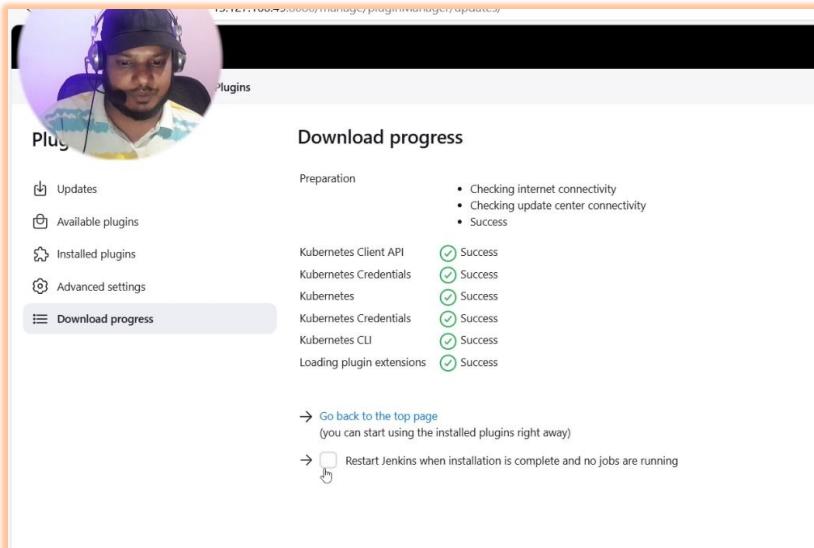
1. Eclipse Temurin Installer (Install without restart)
2. SonarQube Scanner (Install without restart)
3. NodeJs Plugin (Install Without restart) – 16.20.2
5. Stage view
6. jdk
- Docker plugin
7. Docker
8. Docker Commons
9. Docker Pipeline
10. Docker API
11. docker-build-step



The screenshot shows the Jenkins interface for managing plugins. A search bar at the top contains the text 'docker'. Below it, a table lists several available plugins:

| Plugin | Description | Last Published |
|------------------------------------|--|------------------|
| Eclipse Temurin installer 1.7 | Provides an installer for the JDK tool that downloads the Eclipse Temurin™ build based upon OpenJDK from the Adoptium Working Group. | 1 mo 9 days ago |
| NodeJS 1.6.4 | npm | 1 mo 14 days ago |
| Pipeline Stage View 2.37 | User Interface | 1 mo 21 days ago |
| Docker 1274.yc0203fdf2e74 | Cloud Providers Cluster Management docker | 29 days ago |
| Docker Commons 451.yd12c371eeeb_3 | Library plugins (for use by other plugins) docker | 25 days ago |
| Docker Pipeline 611.v16e84da_6d3ff | pipeline DevOps Deployment docker | 1 mo 2 days ago |

Add this too plugins :



The screenshot shows the Jenkins Plugins management interface. On the left, there's a sidebar with options like 'Updates', 'Available plugins', 'Installed plugins', 'Advanced settings', and 'Download progress'. The 'Download progress' option is selected. The main area displays a table of installed and available plugins under the 'Preparation' section:

| Plugin | Status |
|---------------------------|---------|
| Kubernetes Client API | Success |
| Kubernetes Credentials | Success |
| Kubernetes | Success |
| Kubernetes Credentials | Success |
| Kubernetes CLI | Success |
| Loading plugin extensions | Success |

Below the table, there are two links with arrows: 'Go back to the top page (you can start using the installed plugins right away)' and 'Restart Jenkins when installation is complete and no jobs are running'.

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

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.

← 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

starbucks

Create

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



CloudAseem

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 

Add JDK

Setup the Tools form Jenkins Dashboard .

≡ JDK

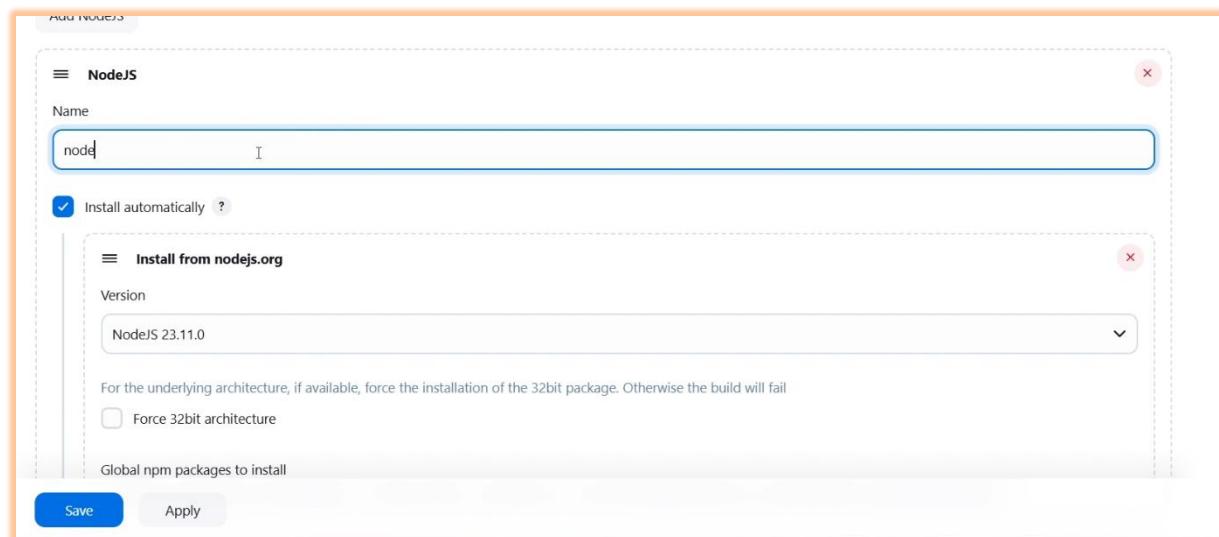
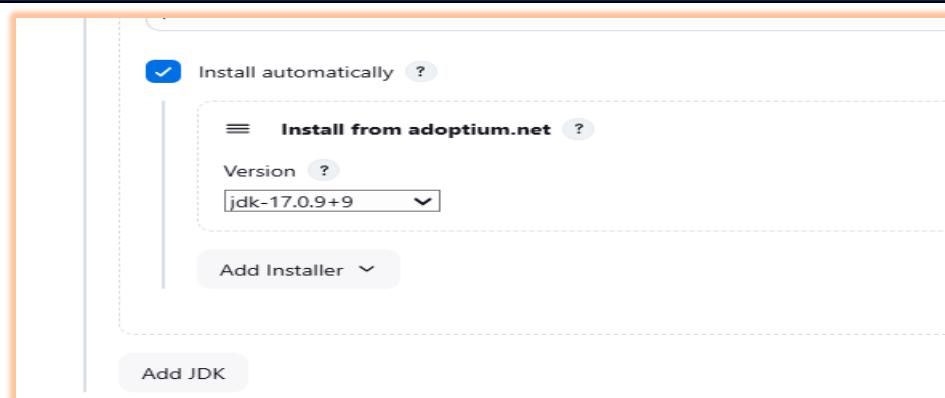
Name

node17

Install automatically 

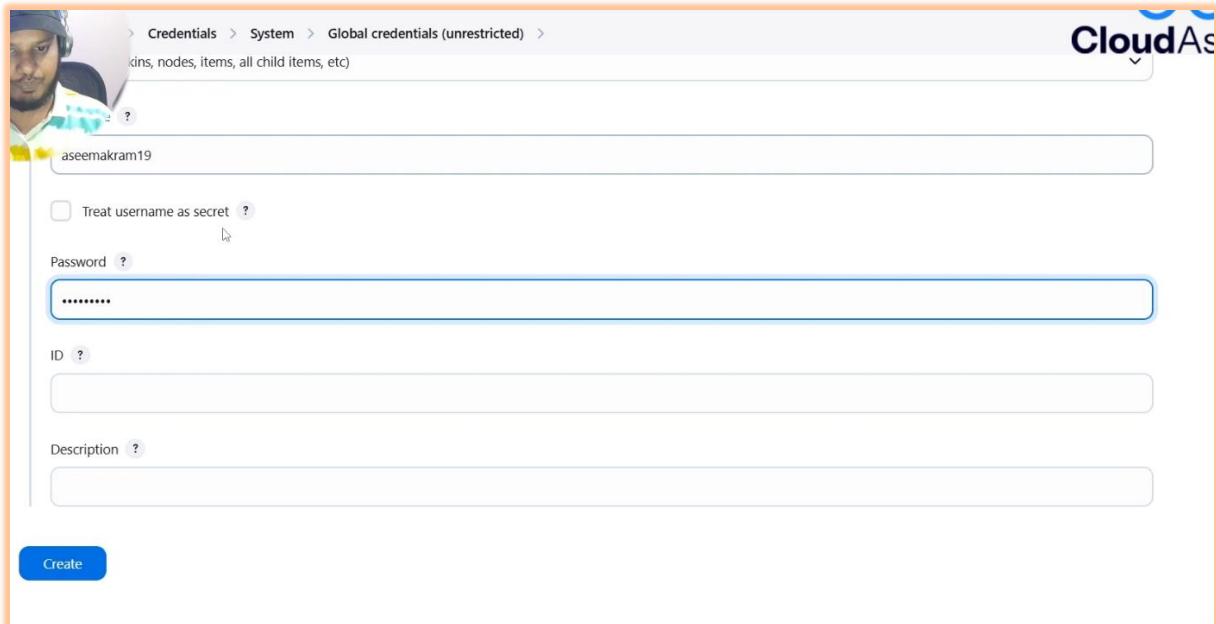
Add Installer ▾

 Add JDK



CloudAseem

Add Docker credential in Jenkins credentials Vault as username as password:



SonarQube server

CloudAseem

Sonarqube Server : Setup a Ec2

Instances (1/1) [Info](#)

Last updated  less than a minute ago [Connect](#) [Instance state](#) [Actions](#) [Launch instances](#)

Find Instance by attribute or tag (case-sensitive) [All states](#)

Instance ID = i-068fdd593ee1d8fc8 [X](#) | [Clear filters](#)

| Name | Instance ID | Instance state | Instance type | Status check | Alarm status | |
|--|---------------------|---|---|--------------|--------------|-----------------------------|
| <input checked="" type="checkbox"/> sonarqube-server | i-068fdd593ee1d8fc8 |  Pending |   | t2.medium | - | View alarms |

i-068fdd593ee1d8fc8 (sonarqube-server)

[Details](#) | [Status and alarms](#) | [Monitoring](#) | [Security](#) | [Networking](#) | [Storage](#) | [Tags](#)

Instance summary [Info](#)

Instance ID: i-068fdd593ee1d8fc8
IPv6 address: -

Public IPv4 address copied: [65.2.171.110](#) | [open address](#)

Private IPv4 addresses: [172.31.42.18](#)
Public IPv4 DNS: [ec2-65-2-171-110.ap-south-1.compute.amazonaws.com](#) | [open address](#)

Access the server and execute this script

```
#!/bin/bash
# Script to install Docker on an EC2 instance and configure permissions

# Update the package list
sudo apt-get update -y

# Install Docker
sudo apt-get install docker.io -y

# Add the 'ubuntu' and 'jenkins' users to the 'docker' group to allow running Docker without sudo
sudo usermod -aG docker ubuntu
sudo usermod -aG docker jenkins

# Apply the new group settings immediately
newgrp docker

# Set correct permissions for the Docker socket to allow 'docker' group members to access it
sudo chmod 660 /var/run/docker.sock
sudo chown root:docker /var/run/docker.sock

# Restart Docker service to apply changes
sudo systemctl restart docker

# Verify installation
docker -version

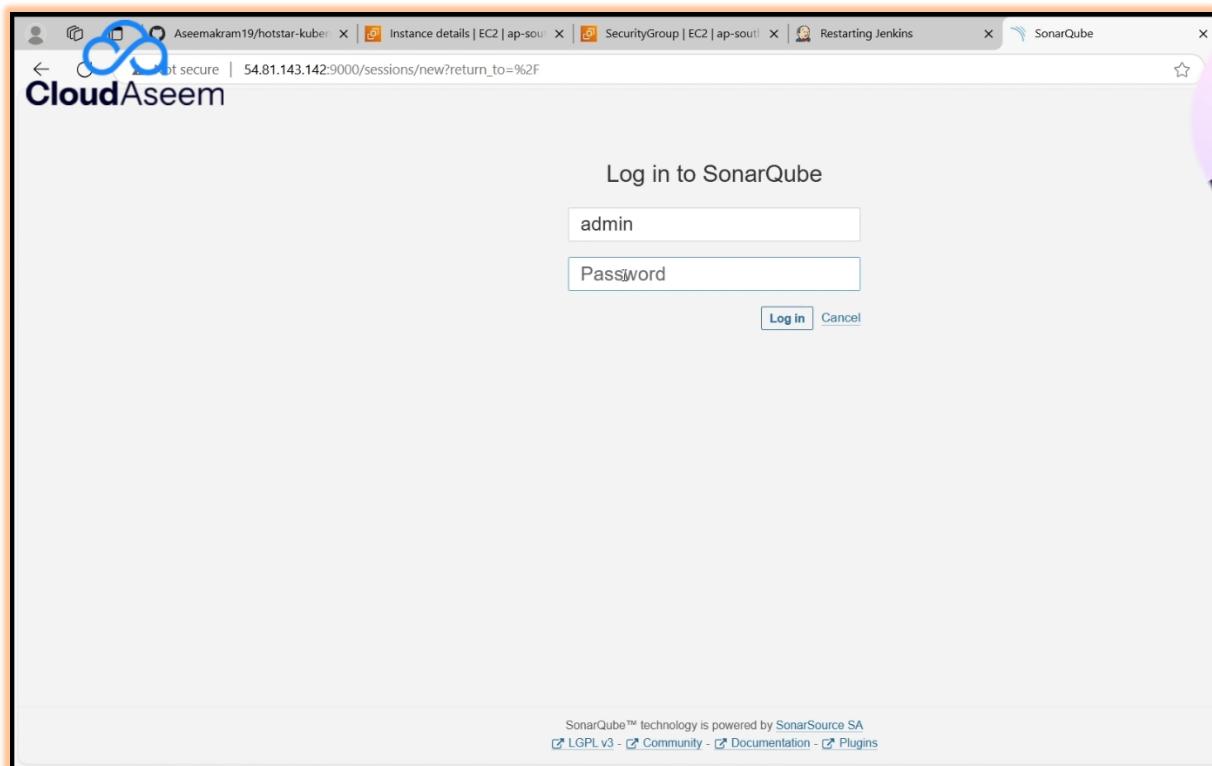
# Run SonarQube container in detached mode with port mapping
docker run -d --name sonar -p 9000:9000 sonarqube:lts-community
```

we create a sonarqube container

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

```
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$ █
```

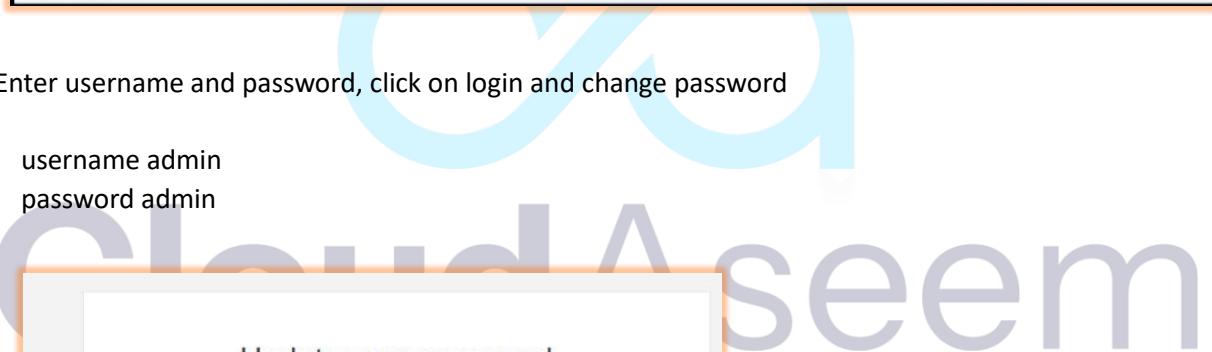
Now our Sonarqube is up and running



Enter username and password, click on login and change password

username admin

password admin



Update 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 New password, This is Sonar Dashboard.

The screenshot shows the SonarQube interface. At the top, there are tabs for Projects, Issues, Rules, Quality Profiles, Quality Gates, and Administration. A banner at the top right encourages upgrading to the latest active version. Below the tabs, a section titled "How do you want to create your project?" lists five options: "From Azure DevOps", "From Bitbucket Server", "From Bitbucket Cloud", "From GitHub", and "From GitLab", each with a "Set up global configuration" link. Below this, a section titled "Are you just testing or have an advanced use-case? Create a project manually." shows a "Manually" button next to a code editor icon. To the right, a sidebar promotes "Get the most out of SonarQube!" with a message about using the Sonar IDE plugin and connecting SonarLint.

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 - Security - Users page. It displays a table of users with columns for SCM Accounts, Last connection, Groups, and Tokens. One row is highlighted for the user "Administrator admin". The "Tokens" column for this user shows "sonar-administrators" and "sonar-users" with a count of 0. A "Generate Token" button is visible in the top right corner of the user card.

Create a token with a name and generate

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



A screenshot of the Jenkins system configuration page. It shows a form for adding a SonarQube tool. The "Name" field is set to "SonarQube". The "Server URL" field contains "http://54.81.143.142:9000". The "Server authentication token" field has a placeholder "SonarQube authentication token". Below these fields are buttons for "none", "+ Add", and "Advanced". At the bottom is a "Save" button labeled "Add SonarQube". On the right side of the page, there is a profile picture of a person wearing a headset.

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

A screenshot of the Jenkins "Add Credential" page. The "Kind" dropdown is set to "Secret file". The "Scope" dropdown shows "Global (Jenkins, nodes, items, all child items, etc)" selected. Below the scope are options for "Global (Jenkins, nodes, items, all child items, etc)", "System (Jenkins and nodes only)", and "Choose File". The "ID" field is empty. The "Description" field is also empty.

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 |
|-------------|-------|-------------|-------------|
| Sonar-token | sonar | Secret text | sonar |

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

☰ SonarQube Scanner

Name

! Required

Install automatically ?

☰ Install from Maven Central

Version

Add Installer ▾

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

Create Webhook

All fields marked with * are required

Name *

URL *

Server endpoint that will receive the webhook payload, for example:
 "http://my_server/foo". If HTTP Basic authentication is used, HTTPS is recommended to avoid man in the middle attacks. Example:
 "https://myLogin:myPassword@my_server/foo"

Secret

If provided, secret will be used as the key to generate the HMAC hex (lowercase) digest value in the 'X-Sonar-Webhook-HMAC-SHA256' header.

Create **Cancel**

- webhook - <http://13.127.166.49:8080/sonarqube-webhook/>

Create Webhook

All fields marked with * are required

Name *
 

URL *
 

Server endpoint that will receive the webhook payload, for example: "http://my_server/foo". If HTTP Basic authentication is used, HTTPS is recommended to avoid man in the middle attacks. Example: "https://myLogin:myPassword@my_server/foo"

Secret

If provided, secret will be used as the key to generate the HMAC hex (lowercase) digest value in the 'X-Sonar-Webhook-HMAC-SHA256' header.

[Create](#) [Cancel](#)

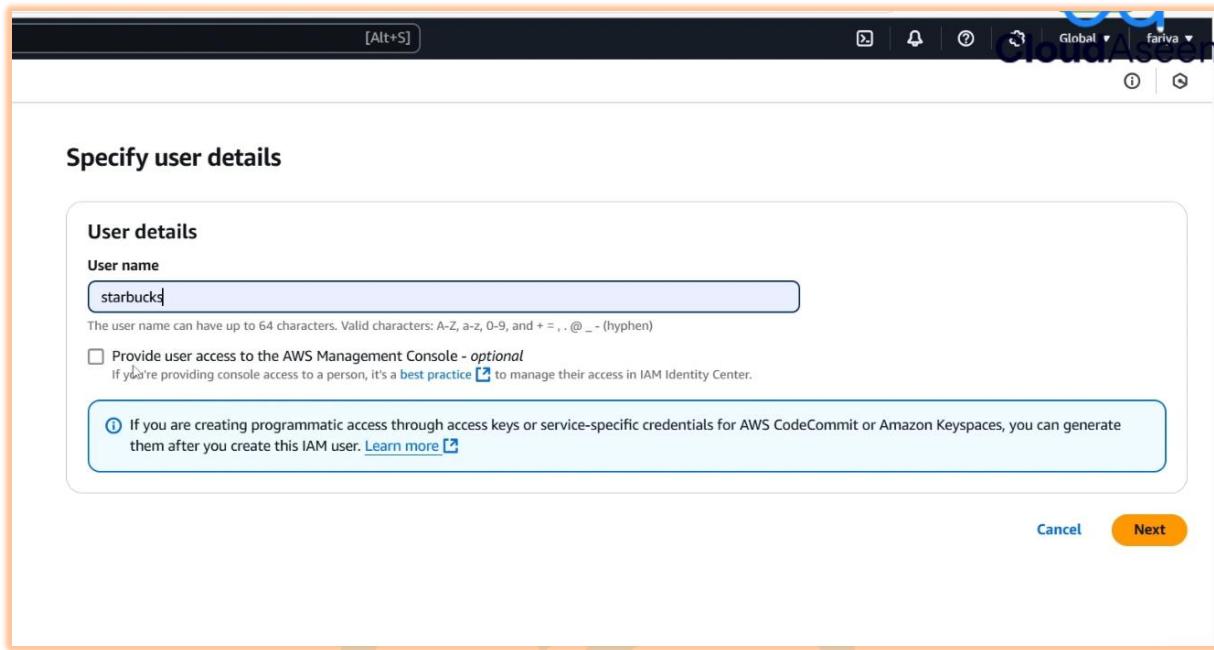
To set up the AWS Access Key and Secret Access Key for configuring AWS EKS from an EC2 instance, follow these steps:

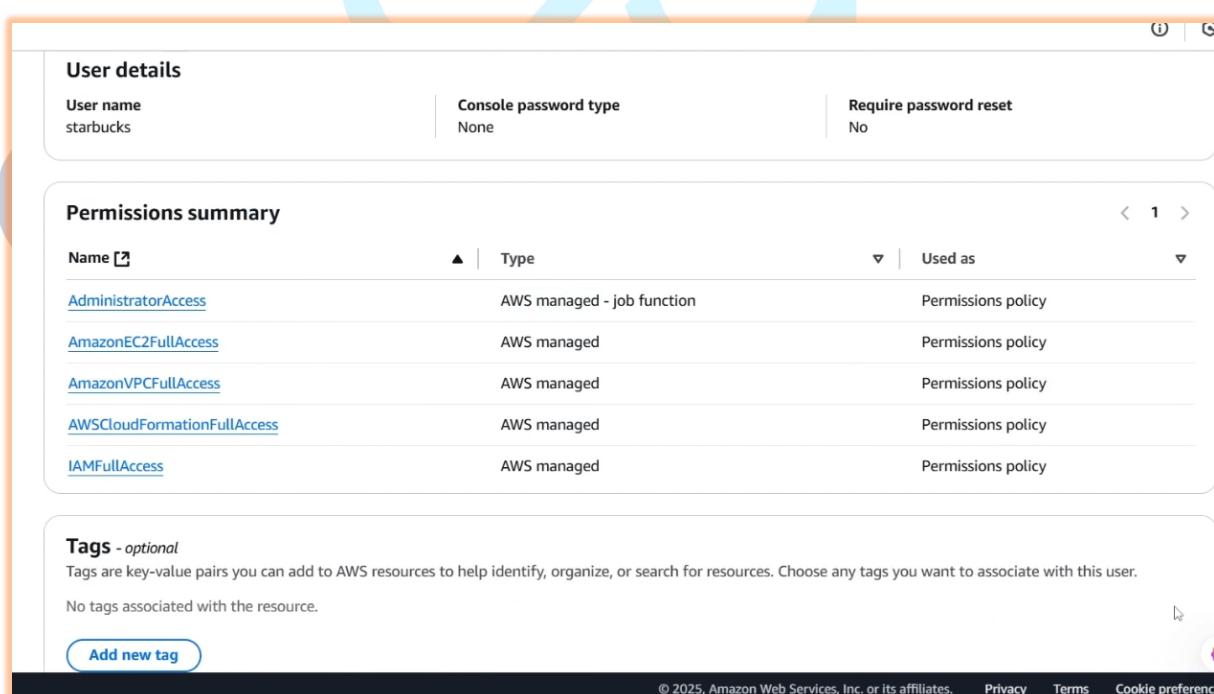
1. Create IAM User with EKS Permissions

1. **Log in to the AWS Management Console.**
2. **Go to IAM (Identity and Access Management).**
3. **Create a new IAM user:**
 - o Click on **Users** from the left sidebar and then click **Add user**.
 - o Provide a username (e.g., `eks-setup-user`).
 - o Choose **Programmatic access** to allow the user to access AWS via CLI or SDK.
4. **Attach permissions:**
 - o Attach the **AmazonEKSClusterPolicy** and **AmazonEKSServicePolicy** for managing EKS clusters.
 - o Optionally, you can attach other permissions such as `AmazonEC2FullAccess` if your user needs access to EC2 instances for EKS worker nodes.
5. **Review and create the user.**
6. **Save the Access Key ID and Secret Access Key:**

- Once the user is created, you'll be provided with an **Access Key ID** and **Secret Access Key**. Save these credentials securely.

Create IAM user for AWS credentials to auth for EKS cluster setup





| Name | Type | Used as |
|---|----------------------------|--------------------|
| AdministratorAccess | AWS managed - job function | Permissions policy |
| AmazonEC2FullAccess | AWS managed | Permissions policy |
| AmazonVPCFullAccess | AWS managed | Permissions policy |
| AWSCloudFormationFullAccess | AWS managed | Permissions policy |
| IAMFullAccess | AWS managed | Permissions policy |

Tags - optional
 Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.
 No tags associated with the resource.

[Add new tag](#)

© 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

> Create access key

the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.

[Retrieve access key](#) [info](#)

Access key

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.

| Access key | Secret access key |
|---|---------------------------------|
| <input type="text"/> AKIAQXPZDEJFEJRVIMXP | <input type="text"/> ***** Show |

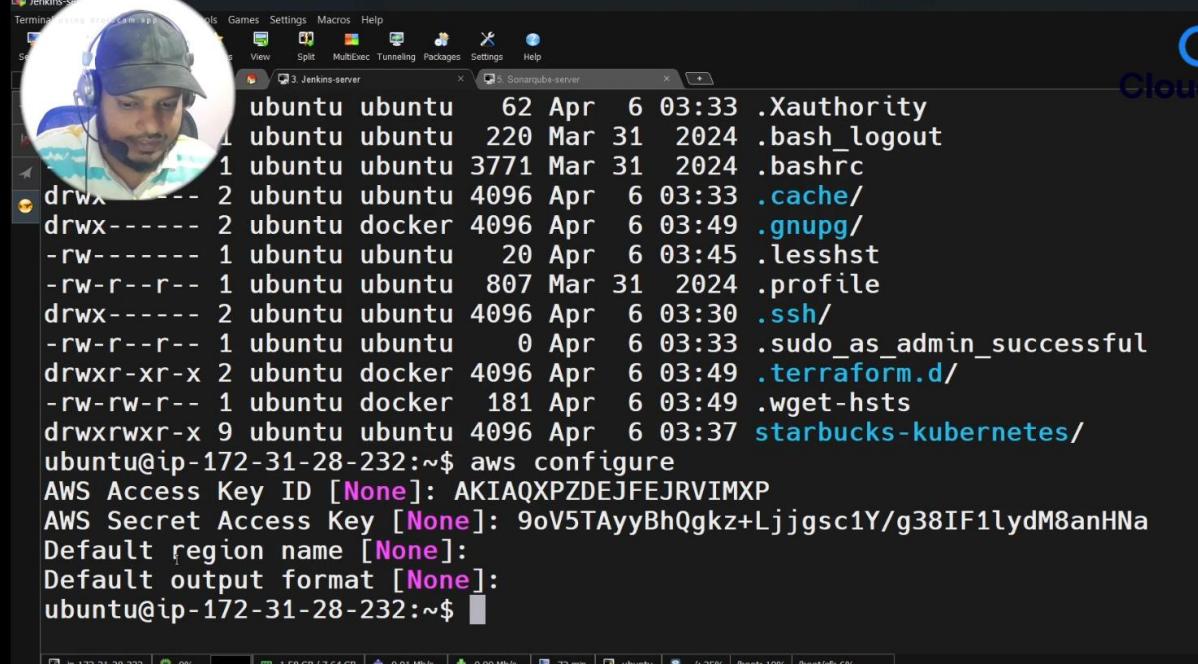
Access key best practices

- 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.
- Rotate access keys regularly.

For more details about managing access keys, see the [best practices for managing AWS access keys](#).

[Download .csv file](#) [Done](#)

Configure on jenkins server



```

ubuntu@ip-172-31-28-232:~$ aws configure
AWS Access Key ID [None]: AKIAQXPZDEJFEJRVIMXP
AWS Secret Access Key [None]: 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
Default region name [None]:
Default output format [None]:
ubuntu@ip-172-31-28-232:~$ 
```

Create EKS Cluster from Jenkins server

Execute the below commands as separate set

(a)

```
eksctl create cluster --name=Cloudaseem \
--region=ap-south-1 \
--zones=ap-south-1a,ap-south-1b \
--version=1.30 \
--without-nodegroup
```

It will take 5-10 minutes to create the cluster

Goto EKS Console and verify the cluster.

(b)

```
eksctl utils associate-iam-oidc-provider \
--region ap-south-1 \
--cluster Cloudaseem \
--approve
```

The above command is crucial when setting up an EKS cluster because it enables IAM roles for service accounts (IRSA)

Amazon EKS uses OpenID Connect (OIDC) to authenticate Kubernetes service accounts with IAM roles.

Associating the IAM OIDC provider allows Kubernetes workloads (Pods) running in the cluster to assume IAM roles securely.

Without this, Pods in EKS clusters would require node-level IAM roles, which grant permissions to all Pods on a node.

Without this, these services will not be able to access AWS resources securely.

(c)

Before executing the below command, in the 'ssh-public-key' keep the '<PEM FILE NAME>' (dont give .pem. Just give the pem file name) which was used to create Jenkins Server

```
eksctl create nodegroup --cluster=Cloudaseem \
--region=ap-south-1 \
--name=node2 \
--node-type=t3.medium \
--nodes=3 \
--nodes-min=2 \
--nodes-max=4 \
--node-volume-size=20 \
--ssh-access \
--ssh-public-key=starbucks1 \
--managed \
--asg-access \
--external-dns-access \
--full-ecr-access \
--appmesh-access \
```



--alb-ingress-access

It will take 5-10 minutes

A screenshot of the AWS EC2 Instances page. It displays five instances: sonarque-server, Cloudaseem-node2-Node, jenkins-server, and two more Cloudaseem-node2-Node entries. All instances are in a 'Running' state. The interface includes a search bar, filters, and a 'Launch instances' button.

A screenshot of the AWS EKS Cluster Compute tab for the 'Cloudaseem' cluster. It shows cluster info including status (Active), Kubernetes version (1.30), support period (Standard support until July 23, 2025), and provider (EKS). The 'Compute' tab is selected. Below, the 'Nodes (0)' section indicates there are no nodes in the cluster.

Add more policies to Starbucks user

> Add permissions

Review

The following policies will be attached to this user. [Learn more](#)

User details

User name
starbucks

Permissions summary (4)

| Name | Type | Used as |
|--|-------------|--------------------|
| AmazonEKSClusterPolicy | AWS managed | Permissions policy |
| AmazonEKSWorkerNodePolicy | AWS managed | Permissions policy |
| AmazonEC2ContainerRegistryReadOnly | AWS managed | Permissions policy |
| IAMReadOnlyAccess | AWS managed | Permissions policy |

Cancel Previous Add permissions

Stage – 2

CloudAseem

Lets setup our first Jenkins pipeline

Create Job for

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



A screenshot of a web browser window showing the Jenkins "New Item" creation interface. The URL in the address bar is "13.127.166.49:8080/view/all/newJob". The page title is "NEW ITEM". A placeholder text "Enter an item name" is followed by a text input field containing "starbucks-pipeline". Below this, a section titled "Select an item type" lists several options: "Freestyle project", "Maven project", "Pipeline" (which is highlighted with a light gray background), "Multi-configuration project", and "Folder". Each option has a small icon and a brief description. At the bottom of the list is a blue "OK" button.

🔒 1. Add GitHub Credentials to Jenkins

1. Go to Jenkins Dashboard > Manage Jenkins > Credentials.

2. Add a new credential:

- Kind: Username with password
- Username: Your GitHub username
- Password: Your GitHub Personal Access Token (PAT)
- ID: github-creds (you'll use this in the pipeline)
- Description: GitHub Access Token

CloudAseem



CloudAseem

CloudAseem
Mohammed Aseem Akram

Definition

Pipeline script

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/starbu'
8~             }
9~         }
10    }
11 }
```

Hello World

Use Groovy Sandbox ?

Pipeline Syntax

CloudAseem | 54.81.143.142:8080/jenkins/configure
Dashboard > hotstar > Configuration

Configure

General

Triggers

Pipeline

Advanced

Define your Pipeline using Groovy directly or pull it from source control.

Definition

Pipeline script

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 }
```

Hello World

Use Groovy Sandbox ?

Pipeline Syntax



Save

Apply

Apply and Save and click on Build

CloudAseem

CloudAseem
Mohammed Aseem Akram

13.127.166.49:8080/job/starbucks-pipeline/

thsi for Starbucks Project

Stage View

Git-clone

Average stage times:
(full run time: ~6s)

| | | |
|----|-------|------------|
| #1 | 10:23 | No Changes |
| | | 2s |

Permalinks

Builds

No builds

Let's add a pipeline of Project

```
pipeline{
    agent any
    tools{
        jdk 'jdk'
        nodejs 'node17'
    }
    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/starbucks-kubernetes.git'
            }
        }
        stage("Sonarqube Analysis"){
            steps{
                withSonarQubeEnv('SonarQube'){
                    sh """ $SCANNER_HOME/bin/sonar-scanner -Dsonar.projectName=starbucks \
                    -Dsonar.projectKey=starbucks """
                }
            }
        }
        stage("quality gate"){
            steps{
                script{
                    waitForQualityGate abortPipeline: false, credentialsId: 'Sonar-token'
                }
            }
        }
    }
}
```

```
        }
    }
stage('Install Dependencies') {
    steps {
        sh "npm install"
    }
}
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 starbucks ."
                sh "docker tag starbucks aseemakram19/starbucks:latest"
                sh "docker push aseemakram19/starbucks:latest"
            }
        }
    }
}
stage("TRIVY"){
    steps{
        sh "trivy image aseemakram19/starbucks:latest > trivyimage.txt"
    }
}
stage('App Deploy to Docker container'){
    steps{
        sh 'docker run -d --name starbucks -p 3000:3000 aseemakram19/starbucks: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 starbucks 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'
        )
    }
}
}
```

CloudAseem

This Bingo Pipeline

Stage View

| Average stage times: | 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 |
|---------------------------|---------------------------|-----------------|-------------------|--------------------|--------------------------|----------------------|---------------|---------------|---------------------|-------|---------------------|
| | 33s | 416ms | 3s | 26s | 805ms | 47s | 4s | 6s | 2min 22s | 27s | 889ms |
| #3 02:46 No Changes | 33s | 416ms | 3s | 26s | 805ms (paused 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

13.127.166.49:8080/job/starbucks-pipeline/



starbucks-pipeline

for Starbucks Project

Stage View

| Average stage times: | Declarative: Tool Install | clean workspace | Checkout from Git | Sonarqube Analysis | quality gate | Install Dependencies | TRIVY FS SCAN | Docker Build & Push | TRIVY | App Deploy to Docker container | Declarat Post Action |
|---------------------------|---------------------------|-----------------|-------------------|--------------------|--------------|----------------------|---------------|---------------------|-------|--------------------------------|----------------------|
| | 23s | 849ms | 2s | 7s | 219ms | 103ms | 99ms | 103ms | 99ms | 110ms | 10s |
| Builds | 225ms | 402ms | 2s | 14s | | | | | | | |
| #3 10:48 No Changes | | | | | | | | | | | |

Result :

- Emails received

Google.com/mail/u/0/#inbox



Inbox 4,297

Search mail Loading...

Promotions 50 new
Social 50 new
Updates 60 new

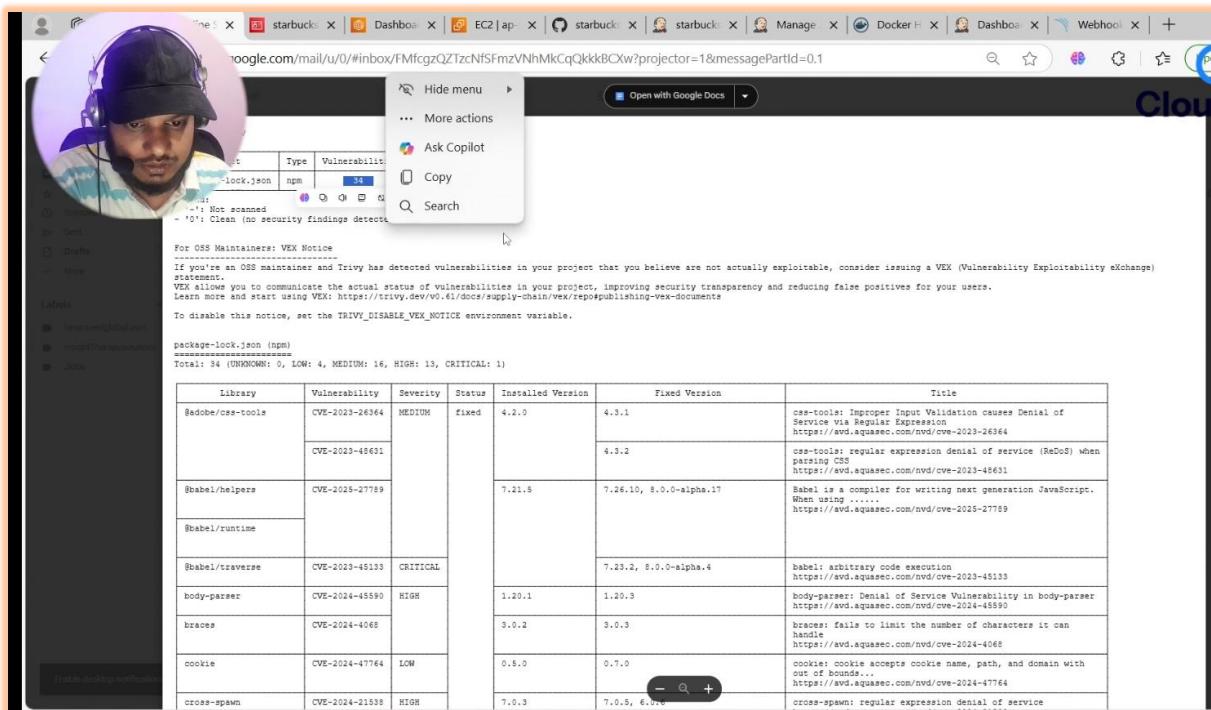
Pipeline SUCCESS: starbucks-pipeline #4 - This is a Jenkins starbucks CI/CD pipeline status. Project: starbucks...

11:18 AM

Test email #2 - This is test email #2 sent from Jenkins

9:42 AM

Report attached in emails



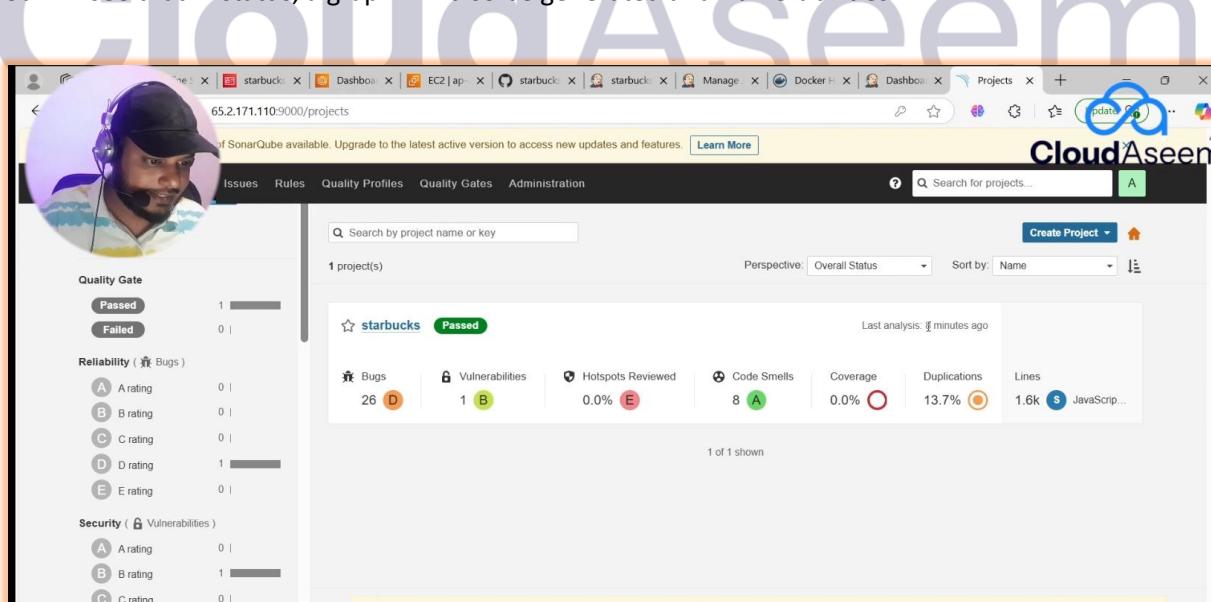
The screenshot shows a Google Mail inbox with a message from "starbuck" containing a Trivy VEX notice report. The report details vulnerabilities found in a package-lock.json file. It includes a table of vulnerabilities with columns for Library, Vulnerability, Severity, Status, Installed Version, Fixed Version, and Title. Key findings include:

| Library | Vulnerability | Severity | Status | Installed Version | Fixed Version | Title |
|------------------|----------------|----------|--------|-------------------------|---|--|
| Badobe/cse-tools | CVE-2023-26364 | MEDIUM | fixed | 4.2.0 | 4.3.1 | cse-tools: Integer Input Validation causes Denial of Service via Regular Expression Denial of Service (ReDoS) when parsing CSS |
| | CVE-2023-45631 | | | 4.3.2 | cse-tools: regular expression denial of service (ReDoS) when parsing CSS | |
| @babel/helpers | CVE-2023-27789 | | 7.21.5 | 7.26.10, 8.0.0-alpha.17 | Babel is a compiler for writing next generation JavaScript. When using | |
| @babel/runtime | | | | 7.23.2, 8.0.0-alpha.4 | babel: arbitrary code execution | |
| @babel/traverse | CVE-2023-45133 | CRITICAL | | | 1.20.1 | babel: Denial of Service Vulnerability in body-parser |
| body-parser | CVE-2024-45590 | HIGH | | 3.0.2 | body-parser: fails to limit the number of characters it can handle | |
| braces | CVE-2024-4068 | | | 0.5.0 | cookies: cookie accepts cookie name, path, and domain with user input without proper validation | |
| cookie | CVE-2024-47764 | LOW | | 7.0.3 | cross-spawn: regular expression denial of service | |
| cross-spawn | CVE-2024-21538 | HIGH | | | | |

3. SonarQube Project scanned :

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.



The screenshot shows the SonarQube interface for the "starbucks" project. The main dashboard displays quality gate status (Passed), reliability metrics (0 bugs), security metrics (0 vulnerabilities), and a summary of 26 bugs, 1 vulnerability, and 8 code smells. The overall status is "Passed".

| Category | Count | Rating |
|-----------------|-------|--------|
| Bugs | 26 | D |
| Vulnerabilities | 1 | B |
| Code Smells | 8 | A |



A screenshot of a SonarQube dashboard for a project named 'main'. The dashboard shows various metrics: 26 Bugs (severity D), 1 Vulnerability (severity B), 3 Security Hotspots (severity E), 21min Debt, 8 Code Smells (severity A), and Maintainability at 13.7%. A progress bar at the bottom indicates 0.0% completion. The URL in the browser is 65.2.171.110:9000/dashboard?id=starbucks.

4. Application as Test Deploy as Docker container
<public-ip:3000>

Our Application is live with this output

A screenshot of a Starbucks website. The top navigation bar includes links for Home, Gift, Order, Pay, and Store. A search bar says 'Looking for something specific?'. Below the navigation is a banner with the text 'Pouring! Sign up now.' and a 'Know More' button. A section titled 'Handcrafted Curations' features six categories: BestSeller (milkshake), Drinks (coffee), Food (sandwich), Merchandise (mug), Coffee At Home (cup), and Ready to Eat (snack). A yellow promotional box at the bottom left says 'Attention Starbucks Fans!' and 'Signature Milkshakes'. It describes the offer and shows a glass of milkshake with the text 'Starting From ₹ 330.00'. A green 'Order Now' button is at the bottom right of the box.

Email alert with Post build

Jenkins Pipeline 2 deploy to EKS cluster

Why add AWS credentials for the `jenkins` user to deploy to EKS?

When you deploy to **Amazon EKS** from a Jenkins pipeline, Jenkins needs **authenticated access** to your AWS account and **authorization** to interact with the EKS cluster — just like any AWS CLI user.

💡 Here's why credentials are needed under the `jenkins` user:

✓ **1. Jenkins Runs Pipelines as the `jenkins` User**

- When Jenkins executes a pipeline, it runs all commands as the `jenkins` user (not `ubuntu`, not `root`).
- So, AWS CLI commands inside the pipeline (`aws eks update-kubeconfig`, `kubectl apply`, etc.) will **fail** unless the `jenkins` user has access to AWS credentials.

✓ **2. AWS CLI Requires `~/.aws/credentials` for Auth**

- AWS CLI looks for credentials in:

`/home/jenkins/.aws/credentials`

- If not present, you'll get errors like:

`Unable to locate credentials`

✓ **3. Required for `aws eks update-kubeconfig`**

This command is used to authenticate to an EKS cluster:

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

It:

- Uses AWS credentials to call EKS APIs

- Writes a kubeconfig file for `kubectl` in `/var/lib/jenkins/.kube/config`

Without AWS credentials, **this step will fail** and `kubectl` won't be able to talk to the cluster.

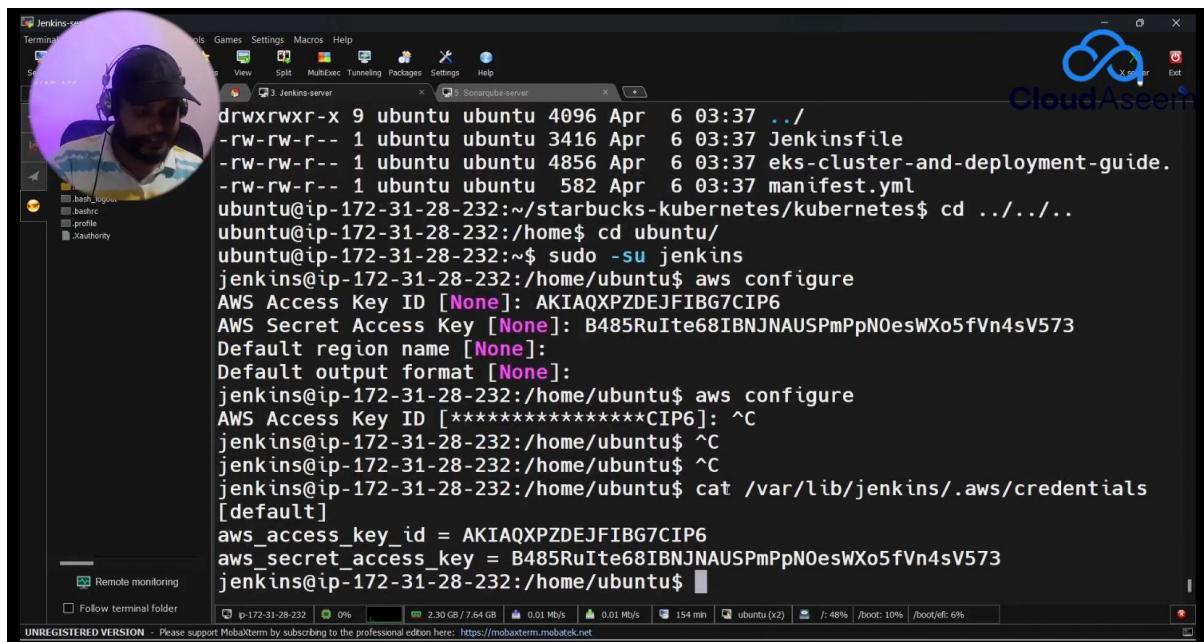
Switch to the jenkins user

```
sudo -su jenkins
pwd ---- /home/ubuntu
whoami ---- jenkins
```

Configure AWS credentials:

```
aws configure ---> Configure with access and secret access keys
This will create the AWS credentials file at "/var/lib/jenkins/.aws/credentials"
```

Verify the credentials



The screenshot shows a terminal window titled "Jenkins-server" running on a Linux system. The user has run the command `aws configure`. The output shows the configuration process, including setting the AWS Access Key ID, AWS Secret Access Key, Default region name, and Default output format. The AWS keys are displayed as masked values. The terminal also shows the command `cat /var/lib/jenkins/.aws/credentials` being run to verify the contents of the credentials file.

```
drwxrwxr-x 9 ubuntu ubuntu 4096 Apr  6 03:37 ../
-rw-rw-r-- 1 ubuntu ubuntu 3416 Apr  6 03:37 Jenkinsfile
-rw-rw-r-- 1 ubuntu ubuntu 4856 Apr  6 03:37 eks-cluster-and-deployment-guide.
-rw-rw-r-- 1 ubuntu ubuntu 582 Apr  6 03:37 manifest.yml
ubuntu@ip-172-31-28-232:~/starbucks-kubernetes/kubernetes$ cd ../../..
ubuntu@ip-172-31-28-232:~/starbucks-kubernetes/kubernetes$ sudo -su jenkins
jenkins@ip-172-31-28-232:~/home/ubuntu$ aws configure
AWS Access Key ID [None]: AKIAQXPZDEJFIBG7CIP6
AWS Secret Access Key [None]: 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
Default region name [None]:
Default output format [None]:
jenkins@ip-172-31-28-232:~/home/ubuntu$ aws configure
AWS Access Key ID [*****CIP6]: ^C
jenkins@ip-172-31-28-232:~/home/ubuntu$ ^C
jenkins@ip-172-31-28-232:~/home/ubuntu$ ^C
jenkins@ip-172-31-28-232:~/home/ubuntu$ cat /var/lib/jenkins/.aws/credentials
[default]
aws_access_key_id = AKIAQXPZDEJFIBG7CIP6
aws_secret_access_key = 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
jenkins@ip-172-31-28-232:~/home/ubuntu$
```

`aws sts get-caller-identity`

If the credentials are valid, you should see output like this:

```
{
  "UserId": "EXAMPLEUSERID",
  "Account": "123456789012",
  "Arn": "arn:aws:iam::123456789012:user/example-user"
}
```

Comeout of the Jenkins user to Restart Jenkins

`exit`

`sudo systemctl restart jenkins`

Switch to Jenkins user

```
sudo -su jenkins
```

```
aws eks update-kubeconfig --region ap-south-1 --name Cloudaseem
```

add this New stage in jenkins file

```
stage('Deploy to EKS Cluster') {
    steps {
        dir('kubernetes') {
            script {
                sh ""
                echo "Verifying AWS credentials..."
                aws sts get-caller-identity

                echo "Configuring kubectl for EKS cluster..."
                aws eks update-kubeconfig --region ap-south-1 --name Cloudaseem

                echo "Verifying kubeconfig..."
                kubectl config view

                echo "Deploying application to EKS..."
                kubectl apply -f manifest.yml

                echo "Verifying deployment..."
                kubectl get pods
                kubectl get svc
                ""

            }
        }
    }
}
```

File exist ,

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

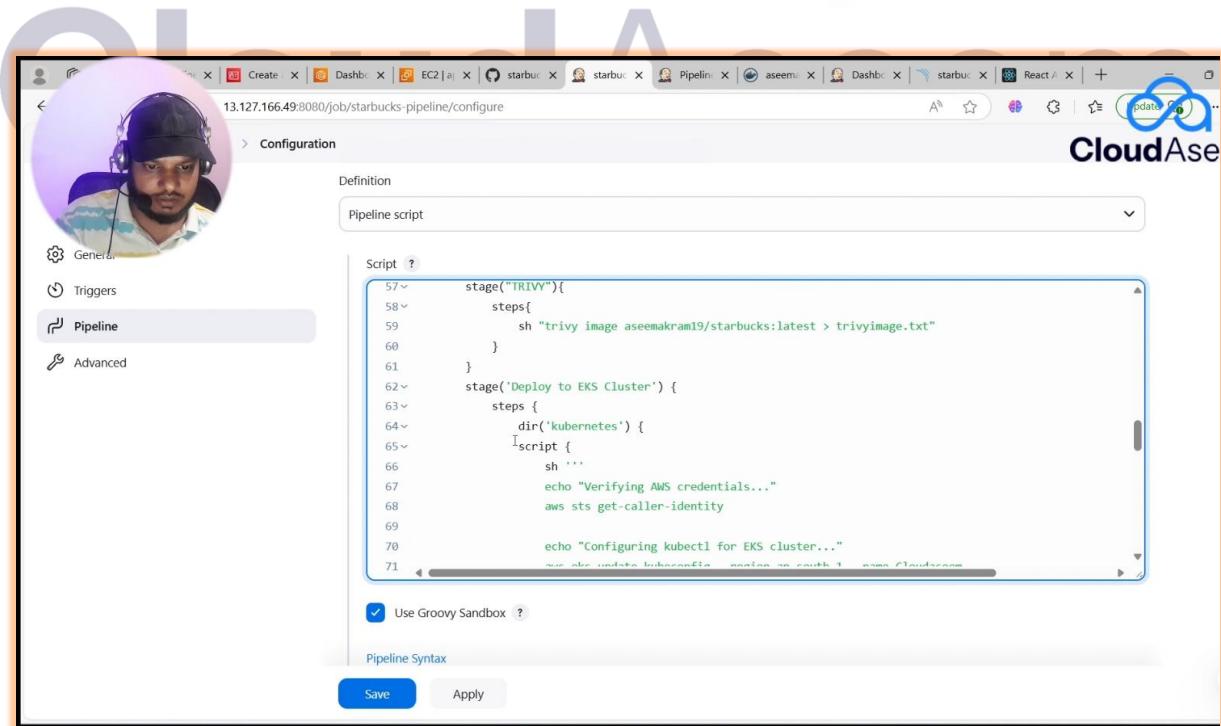
manifest.yml file

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: starbucks-deployment
spec:
  replicas: 2
  strategy:
    type: RollingUpdate
```

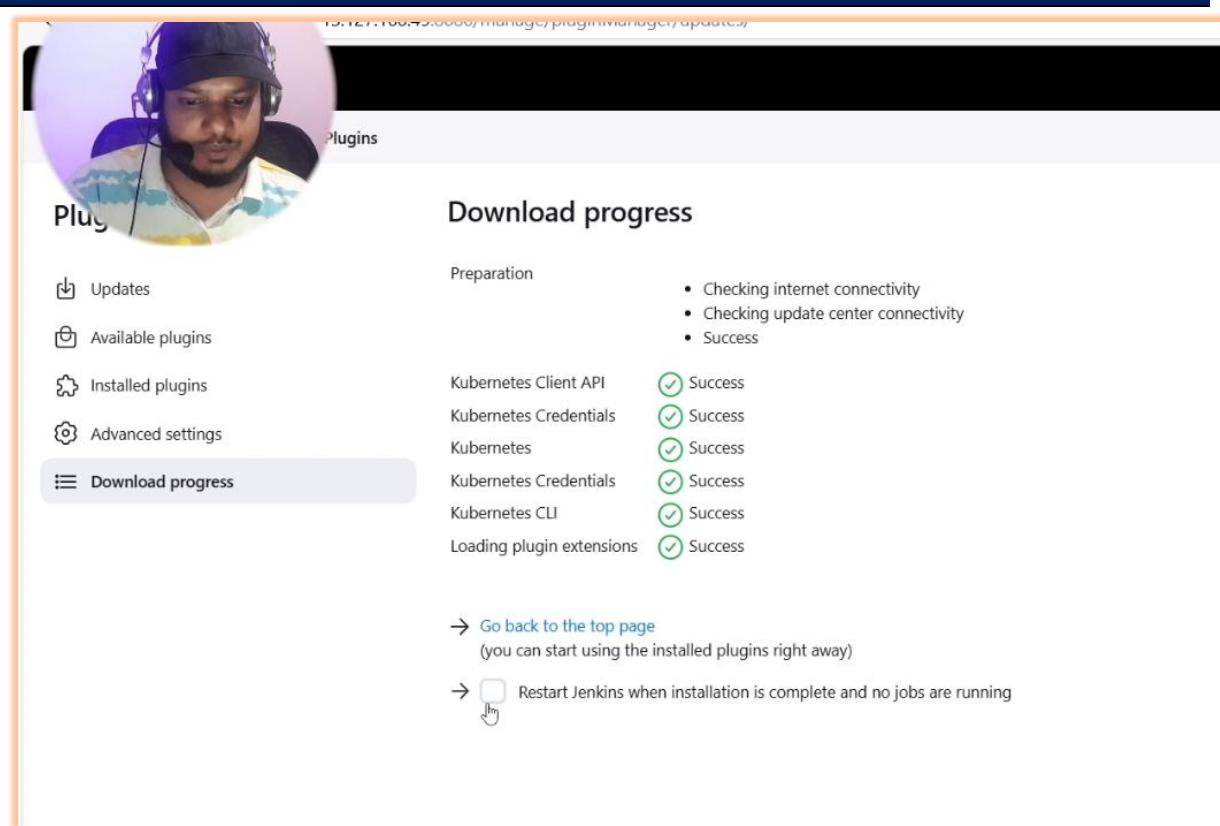
```

selector:
  matchLabels:
    app: starbucks
template:
  metadata:
    labels:
      app: starbucks
spec:
  containers:
    - name: starbucks-container
      image: aseemakram19/starbucks
      ports:
        - containerPort: 3000
---
apiVersion: v1
kind: Service
metadata:
  name: starbucks-service
spec:
  type: LoadBalancer
  selector:
    app: starbucks
  ports:
    - port: 80
      targetPort: 3000

```



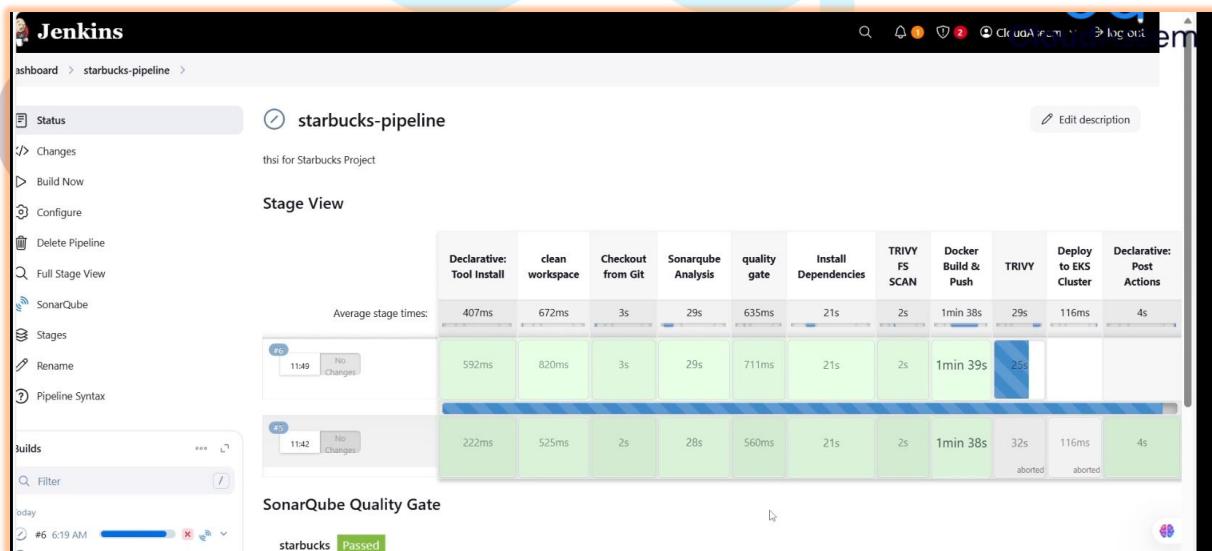
Add this additional Plugins jenkins



→ Go back to the top page
(you can start using the installed plugins right away)

→  Restart Jenkins when installation is complete and no jobs are running

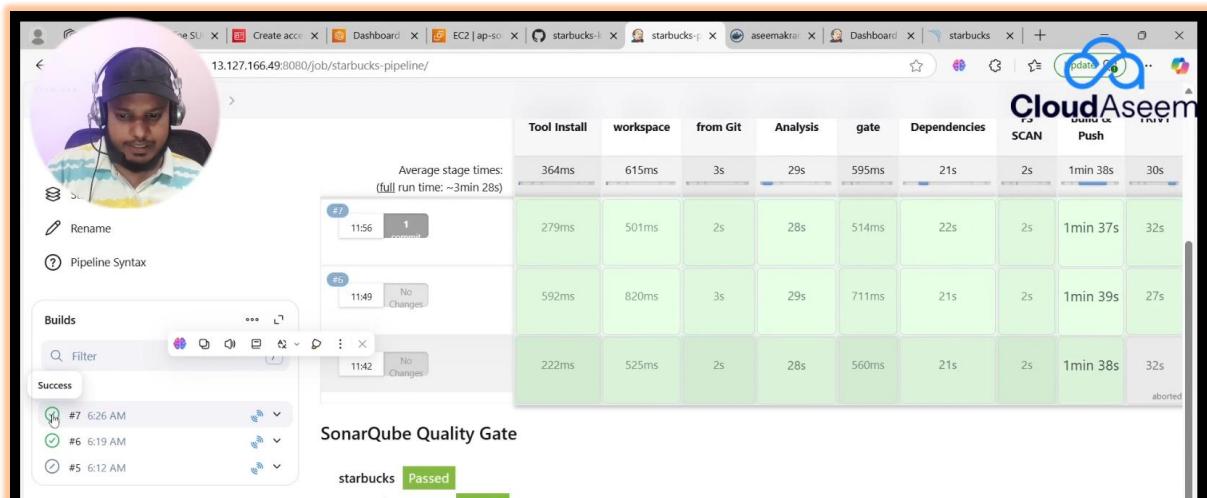
and execute the pipeline now



| | Declarative: Tool Install | clean workspace | Checkout from Git | Sonarqube Analysis | quality gate | Install Dependencies | TRIVY FS SCAN | Docker Build & Push | TRIVY | Deploy to EKS Cluster | Declarative: Post Actions |
|---------------------|---------------------------|-----------------|-------------------|--------------------|--------------|----------------------|---------------|---------------------|-------|-----------------------|---------------------------|
| #6 11:49 No Changes | 407ms | 672ms | 3s | 29s | 635ms | 21s | 2s | 1min 38s | 29s | 116ms | 4s |
| #5 11:42 No Changes | 592ms | 820ms | 3s | 29s | 711ms | 21s | 2s | 1min 39s | 25s | | |
| | 222ms | 525ms | 2s | 28s | 560ms | 21s | 2s | 1min 38s | 32s | 116ms | 4s |

SonarQube Quality Gate

starbucks Passed



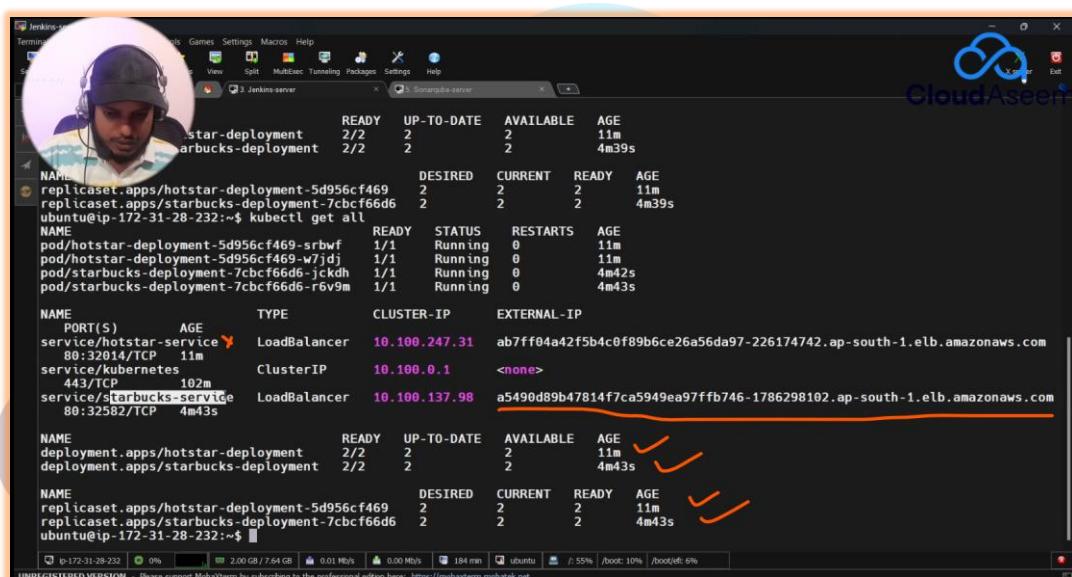
Average stage times:
(full run time: ~3min 28s)

| | Tool Install | workspace | from Git | Analysis | gate | Dependencies | | | |
|----|--------------|-----------|----------|----------|-------|--------------|----|----------|-----|
| #7 | 364ms | 615ms | 3s | 29s | 595ms | 21s | 2s | 1min 38s | 30s |
| #6 | 279ms | 501ms | 2s | 28s | 514ms | 22s | 2s | 1min 37s | 32s |
| #5 | 592ms | 820ms | 3s | 29s | 711ms | 21s | 2s | 1min 39s | 27s |
| | 222ms | 525ms | 2s | 28s | 560ms | 21s | 2s | 1min 38s | 32s |

SonarQube Quality Gate

starbucks Passed

kubectl get all



```

READY UP-TO-DATE AVAILABLE AGE
star-deployment 2/2 2 2 11m
starbucks-deployment 2/2 2 2 4m39s

NAME           DESIRED CURRENT READY AGE
replicaset.apps/hotstar-deployment-5d956cf469 2 2 2 11m
replicaset.apps/starbucks-deployment-7cbc66d6 2 2 2 4m39s
ubuntu@ip-172-31-28-232:~$ kubectl get all
NAME          READY STATUS RESTARTS AGE
pod/hotstar-deployment-5d956cf469-srbwf 1/1 Running 0 11m
pod/hotstar-deployment-5d956cf469-w7jdf 1/1 Running 0 11m
pod/starbucks-deployment-7cbc66d6-jckdh 1/1 Running 0 4m42s
pod/starbucks-deployment-7cbc66d6-r6v9m 1/1 Running 0 4m43s

NAME          TYPE CLUSTER-IP EXTERNAL-IP
service/hotstar-service  LoadBalancer 10.100.247.31 ab7ff04a42f5b4c0f89b6ce26a56da97-226174742.ap-south-1.elb.amazonaws.com
service/kubernetes  ClusterIP 10.100.0.1 <none>
service/starbucks-service  LoadBalancer 10.100.137.98 a5490d89b47814f7ca5949ea97ffb746-1786298102.ap-south-1.elb.amazonaws.com
ubuntu@ip-172-31-28-232:~$ 

NAME          READY UP-TO-DATE AVAILABLE AGE
deployment.apps/hotstar-deployment 2/2 2 2 11m ✓
deployment.apps/starbucks-deployment 2/2 2 2 4m43s ✓

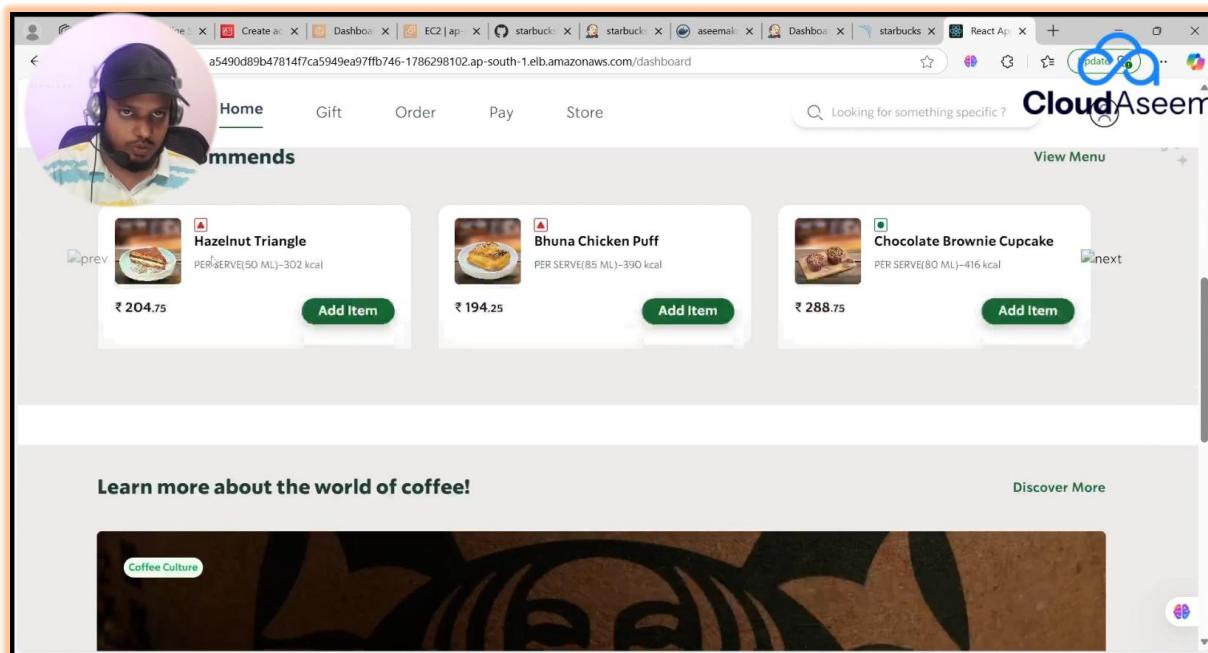
NAME           DESIRED CURRENT READY AGE
replicaset.apps/hotstar-deployment-5d956cf469 2 2 2 11m ✓
replicaset.apps/starbucks-deployment-7cbc66d6 2 2 2 4m43s ✓
ubuntu@ip-172-31-28-232:~$ 

```

UNREGISTERED VERSION - Please support MobaTerm by subscribing to the professional edition here: <https://mobaterm.mobiak.net>



Access app with Loadbalancer

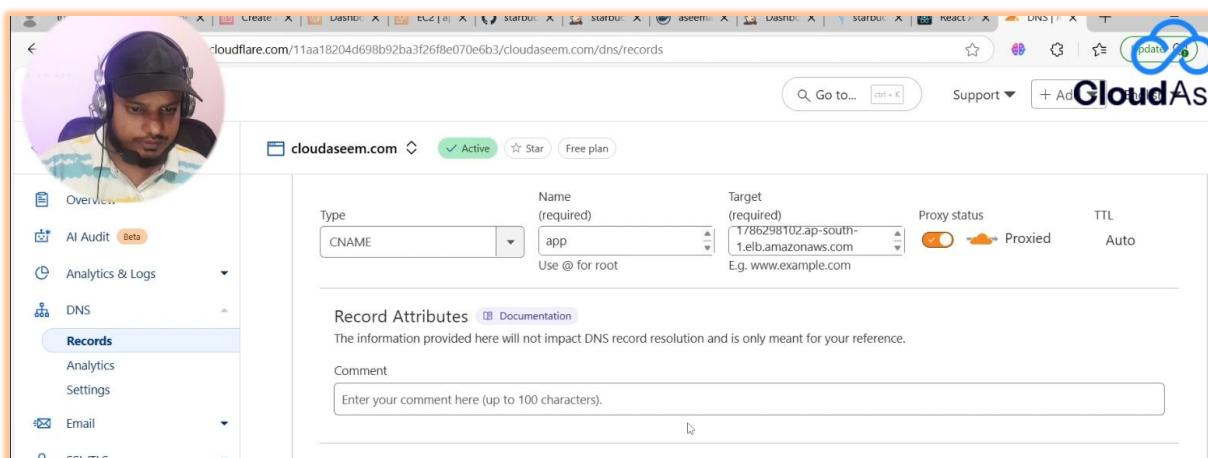


Add Doamin to Our application

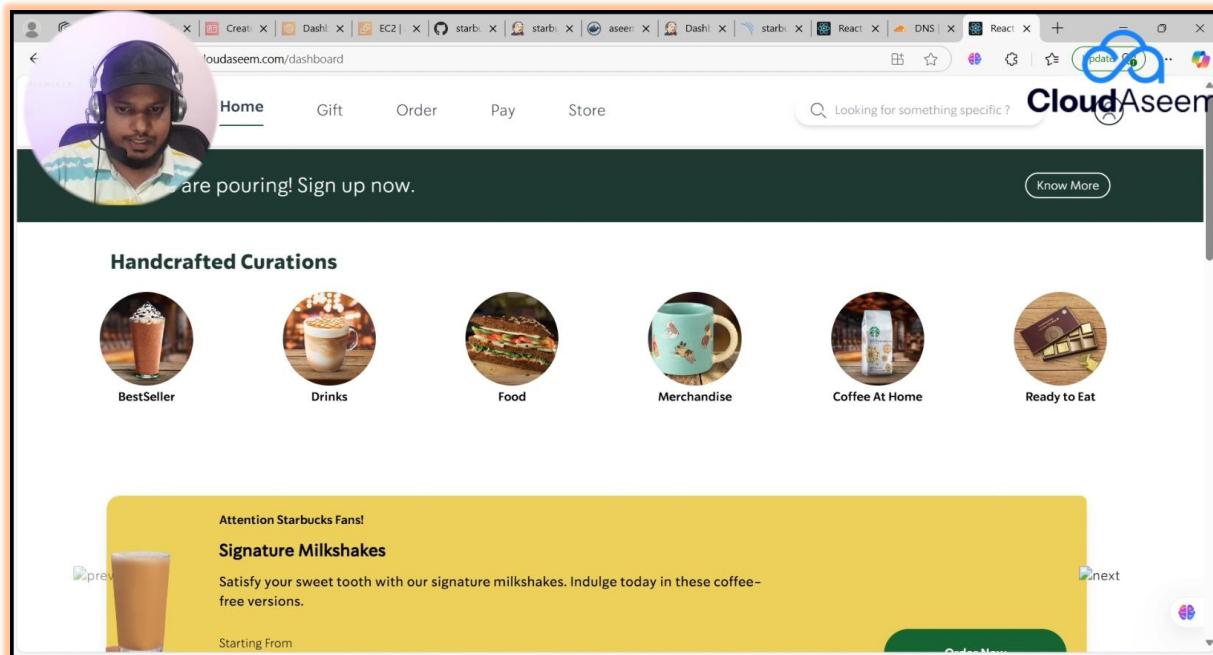
Add Loadbalance in Cloudflare to apply domain and ssl to access by client with cname entry

Configure CNAME for Client Access

1. Go to Cloudflare DNS Settings.
2. Add a CNAME Record:
 - 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 Starbucks Kubernetes with Loadbalancer Enabled and SSL certificated



Part – 03

monitoring server with JENKINS + TERRAFORM = MONITORING SERVER SETUP

Create a new pipeline as below :

Monitoring Server setup with Jenkins + Terraform

Jenkins

Dashboard > All > New Item

New Item

Enter an item name

Saved info

Select... monitoringserver

Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments,

OK

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 ?
action

Choices ?
apply
destroy

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:

action
 apply

Build **Cancel**

```

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/starbucks-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.'
  }
}
```

CloudAseem

CloudAseem
Mohammed Aseem Akram

Configuration

Pipeline

Definition

Pipeline script

```
1v pipeline {  
2 agent any  
3  
4v environment {  
5     AWS_ACCESS_KEY_ID      = credentials('AWS_ACCESS_KEY_ID')  
6     AWS_SECRET_ACCESS_KEY = credentials('AWS_SECRET_ACCESS_KEY')  
7 }  
8  
9v parameters {  
10    string(name: 'action', defaultValue: 'apply', description: 'Terraform action: apply or destroy')  
11 }  
12  
13v stages {  
14    stage('Checkout from Git') {  
15 }
```

Use Groovy Sandbox ?

Save Apply

Note: Store aws secret and secret access key in jenkins credentials below

montirong server

Stage View

| Checkout from Git | Terraform version | Terraform init | Terraform validate | Success plan | Terraform apply/destroy |
|-------------------|-------------------|----------------|--------------------|--------------|-------------------------|
| 2s | 1s | 10s | 6s | 6s | 1s |

Average stage times:

#1 12:12 No Changes

Permalinks

Last build (#1), 0.16 sec ago

Note: Use same Key pair to access monitoring server

Verify the Monitoring server



A screenshot of the AWS Management Console EC2 Instances page. The page title is "Instances (6) Info". It shows a table of six EC2 instances. The columns include Name, Instance ID, Instance state, Instance type, Status check, and Alarm status. One instance, "Monitoring_server", has a red checkmark next to its name. The "Instance state" column shows all instances as "Running". The "Instance type" column shows various types like t2.medium, t3.medium, t3.large, and t3.small. The "Status check" column shows mostly green circles with "2/2 checks passed" or "3/3 checks passed". The "Alarm status" column shows "View alarms" for each instance. The browser address bar at the top shows "https://ec2.console.aws.amazon.com/ec2/home?region=ap-south-1#Instances:instanceState=running".

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

| Security group rule ID | Type | Protocol | Port range | Source | Description - optional |
|------------------------|------------|----------|------------|--|--|
| sgr-0fda058f97f1313dc | Custom TCP | TCP | 3000 | Cust... <input type="button" value="Q"/> | garafan <input type="button" value="Delete"/> |
| sgr-08babc2dab3c7db0b | SSH | TCP | 22 | Cust... <input type="button" value="Q"/> | ssh access <input type="button" value="Delete"/> |
| - | Custom TCP | TCP | 9090 | Any... <input type="button" value="Q"/> | promoflie <input type="button" value="Delete"/> |

Add rule

⚠ Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

After installation, you can access Grafana at:

<http://your-server-ip:3000> (default user: admin, password: admin)

2. Install Prometheus

```
#!/bin/bash
#this script belong to "CLOUDASEEM" YOUTUBE CHANNEL

# Define Prometheus version
PROMETHEUS_VERSION="2.51.2"
```

```
# Update system and install necessary packages
echo "Updating system and installing dependencies..."
sudo apt update -y
sudo apt install -y wget tar

# Create Prometheus user
echo "Creating Prometheus user..."
sudo useradd --no-create-home --shell /bin/false prometheus

# Create Prometheus directory
echo "Creating /etc/prometheus directory..."
sudo mkdir -p /etc/prometheus

# Download and extract Prometheus
echo "Downloading and extracting Prometheus..."
cd /tmp
wget
https://github.com/prometheus/prometheus/releases/download/v${PROMETHEUS_VERSION}/prometheus-${PROMETHEUS_VERSION}.linux-amd64.tar.gz
tar -xvzf prometheus-${PROMETHEUS_VERSION}.linux-amd64.tar.gz

# Move all extracted files to /etc/prometheus
echo "Moving Prometheus files to /etc/prometheus..."
sudo mv prometheus-${PROMETHEUS_VERSION}.linux-amd64/* /etc/prometheus/

# Set ownership and permissions
echo "Setting permissions..."
sudo chown -R prometheus:prometheus /etc/prometheus

# Create symbolic links for binaries
echo "Creating symlinks for Prometheus binaries..."
sudo ln -s /etc/prometheus/prometheus /usr/local/bin/prometheus
sudo ln -s /etc/prometheus/promtool /usr/local/bin/promtool

# Create Prometheus systemd service
echo "Creating systemd service..."
cat <<EOF | sudo tee /etc/systemd/system/prometheus.service
[Unit]
Description=Prometheus Monitoring System
Wants=network-online.target
After=network-online.target

[Service]
User=prometheus
Group=prometheus
Type=simple
```

```
ExecStart=/usr/local/bin/prometheus \\
--config.file=/etc/prometheus/prometheus.yml \\
--storage.tsdb.path=/etc/prometheus/data \\
--web.console.templates=/etc/prometheus/consoles \\
--web.console.libraries=/etc/prometheus/console_libraries
```

Restart=always

[Install]
WantedBy=multi-user.target
EOF

```
# Reload systemd, enable and start Prometheus
echo "Starting Prometheus..." 
sudo systemctl daemon-reload
sudo systemctl enable prometheus
sudo systemctl start prometheus

# Check Prometheus status
echo "Prometheus installation completed!"
sudo systemctl status prometheus --no-pager
```

Install Blackbox exporter

```
l-v-20-0-0.1.1nux-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
b2X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=releaseassetprod%2F20250320%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20
250320T012924Z&X-Amz-Expires=300&X-Amz-Signature=2d54029d913c128ab7b481bfaba38b65fbef005b05aa20b986af70d939579869&X-Amz-SignedHeade
rs=host&response-content-disposition=attachment%3B%20filename%3Dblackbox_exporter-0.26.0.linux-amd64.tar.gz&response-content-type=a
pplication%2Foctet-stream [following]
--2025-03-20 01:29:24-- https://objects.githubusercontent.com/github-production-release-asset-2e65be/41964498/37e20444-c65c-45e1-9
674-99ed1137b43b2X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=releaseassetprod%2F20250320%2Fus-east-1%2Fs3%2Faws4_reques
t&X-Amz-Date=20250320T012924Z&X-Amz-Expires=300&X-Amz-Signature=2d54029d913c128ab7b481bfaba38b65fbef005b05aa20b986af70d939579869&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-amd64.tar.gz 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: IP: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=INFO 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=2025-03-20T01:35:15.401Z level=INFO source=notifier.go:702 msg="Stopping notification manager..." component=notifier
time=2025-03-20T01:35:15.401Z level=INFO source=notifier.go:409 msg="Draining any remaining notifications..." component=notifier
time=2025-03-20T01:35:15.401Z level=INFO source=notifier.go:415 msg="Remaining notifications drained" component=notifier
time=2025-03-20T01:35:15.401Z level=INFO source=notifier.go:345 msg="Notification manager stopped" component=notifier
time=2025-03-20T01:35:15.401Z level=INFO source=main.go:1361 msg="Notifier manager stopped"
time=2025-03-20T01:35:15.401Z 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#

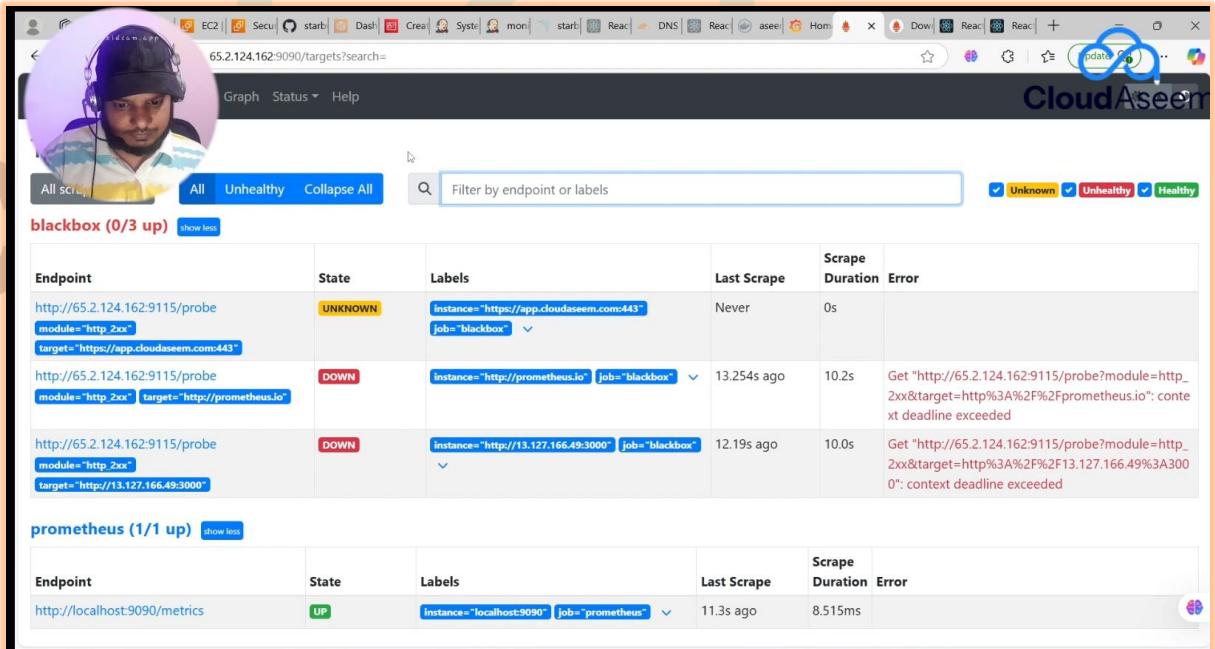
```

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 following details:

- Targets:** A table titled "blackbox (0/3 up)" showing three entries. The first two are labeled "DOWN" and the third is "UP". Each entry includes an "Endpoint" (URL), "State" (UNKNOWN or DOWN), "Labels" (instance, job), "Last Scrape" (timestamp), "Scrape Duration" (duration), and an "Error" message (context deadline exceeded for the DOWN entries).
- Data Sources:** A table titled "prometheus (1/1 up)" showing one entry. The entry is labeled "UP", has "Labels" (instance, job), and was last scraped 11.3s ago with a duration of 8.515ms.

Add

Go to Dashboards → Import.

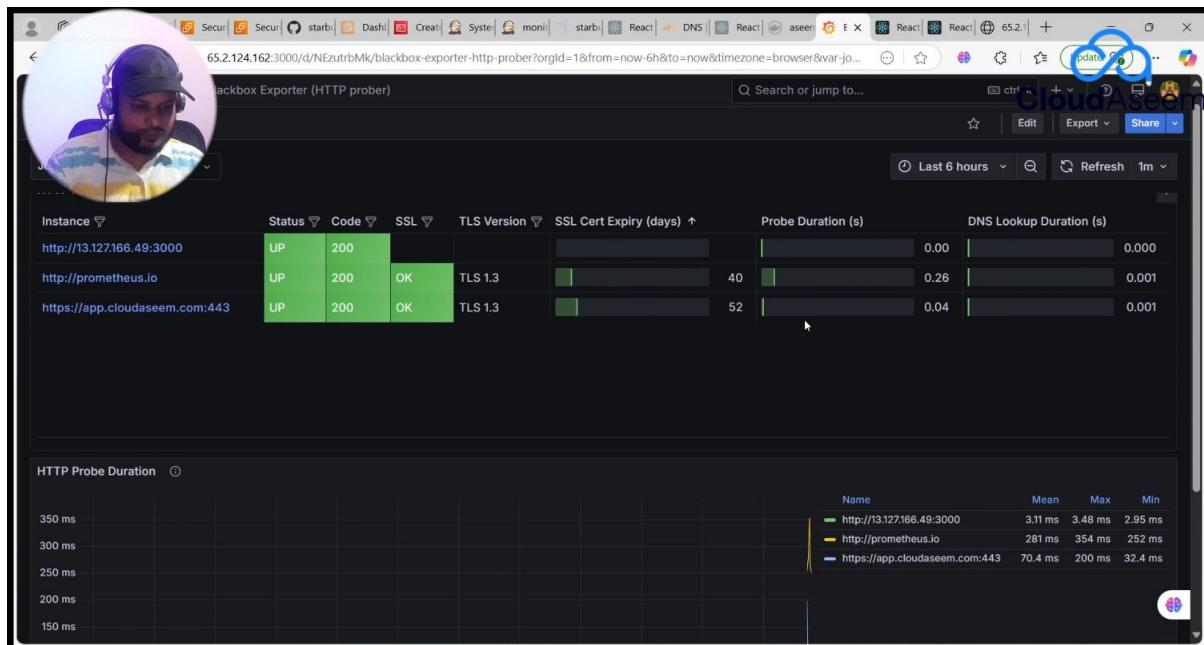
Enter Dashboard ID: 13659 (Prometheus Blackbox Exporter).

Click Load.

Select Prometheus as the Data Source.



Click Import.



Step - 4 :

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

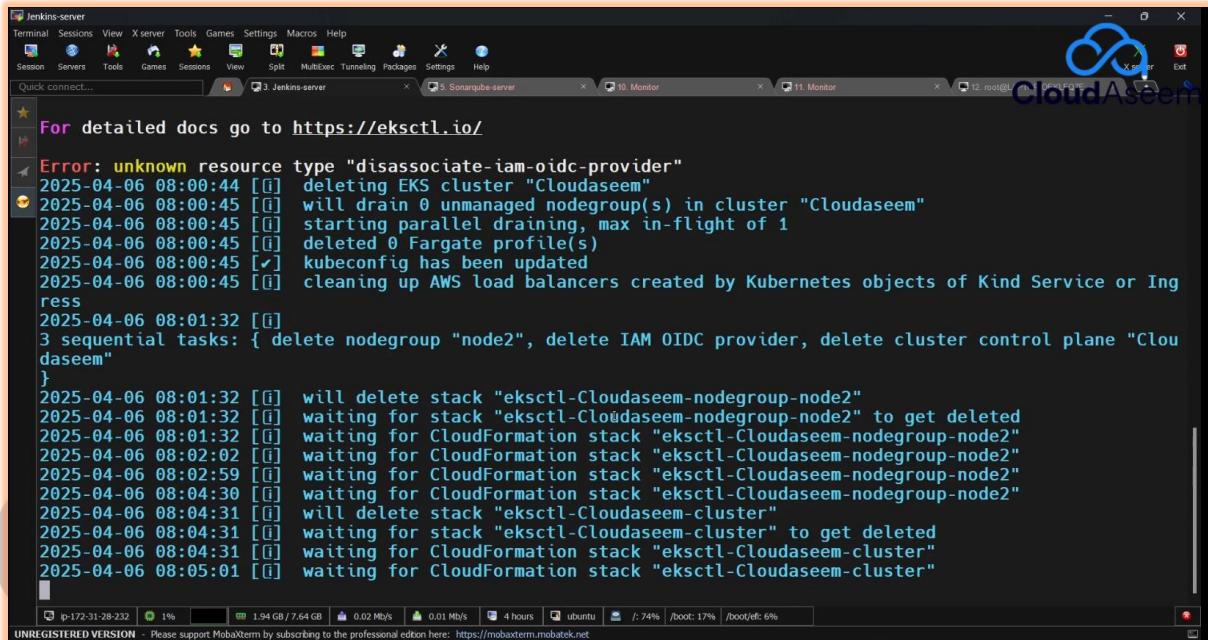
A screenshot of a Jenkins pipeline configuration page. The pipeline is named 'monitoringserver'. On the left, there's a sidebar with options like Status, Changes, Build with Parameters, Configure, Delete Pipeline, Full Stage View, Favorite, Open Blue Ocean, Stages, Rename, and Pipeline Syntax. The main area shows the pipeline stages. The first stage is titled 'Pipeline monitoringserver' with a description: 'This build requires parameters: action'. A dropdown menu is open, showing the option 'destroy'. Below the stage are 'Build' and 'Cancel' buttons. At the bottom, there's a 'Builds' section with a table for filtering builds.

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

```
eksctl delete nodegroup --cluster=Cloudaseem --name=node2 --region=ap-south-1 --wait
```

```
eksctl delete cluster --name=Cloudaseem --region=ap-south-1 --wait
```

```
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
```



```
For detailed docs go to https://eksctl.io/

Error: unknown resource type "disassociate-iam-oidc-provider"
2025-04-06 08:00:44 [i] deleting EKS cluster "Cloudaseem"
2025-04-06 08:00:45 [i] will drain 0 unmanaged nodegroup(s) in cluster "Cloudaseem"
2025-04-06 08:00:45 [i] starting parallel draining, max in-flight of 1
2025-04-06 08:00:45 [i] deleted 0 Fargate profile(s)
2025-04-06 08:00:45 [✓] kubeconfig has been updated
2025-04-06 08:00:45 [i] cleaning up AWS load balancers created by Kubernetes objects of Kind Service or Ingress
2025-04-06 08:01:32 [i]
3 sequential tasks: { delete nodegroup "node2", delete IAM OIDC provider, delete cluster control plane "Cloudaseem"
}
2025-04-06 08:01:32 [i] will delete stack "eksctl-Cloudaseem-nodegroup-node2"
2025-04-06 08:01:32 [i] waiting for stack "eksctl-Cloudaseem-nodegroup-node2" to get deleted
2025-04-06 08:01:32 [i] waiting for CloudFormation stack "eksctl-Cloudaseem-nodegroup-node2"
2025-04-06 08:02:02 [i] waiting for CloudFormation stack "eksctl-Cloudaseem-nodegroup-node2"
2025-04-06 08:02:59 [i] waiting for CloudFormation stack "eksctl-Cloudaseem-nodegroup-node2"
2025-04-06 08:04:30 [i] waiting for CloudFormation stack "eksctl-Cloudaseem-nodegroup-node2"
2025-04-06 08:04:31 [i] will delete stack "eksctl-Cloudaseem-cluster"
2025-04-06 08:04:31 [i] waiting for stack "eksctl-Cloudaseem-cluster" to get deleted
2025-04-06 08:04:31 [i] waiting for CloudFormation stack "eksctl-Cloudaseem-cluster"
2025-04-06 08:05:01 [i] waiting for CloudFormation stack "eksctl-Cloudaseem-cluster"
```

✓ Congratulations! You have successfully deployed **Starbucks 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)



CloudAseem

[Mohammed Aseem Akram](#)

🌟 Stay tuned for more DevOps insights! 🚀

#Cloud #DevOps #Technology #Kubernetes



CloudAseem