prime video **Clone App — End-to-End Secure &**

Scalable Deployment with DevSecOps & Jenkins !

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

Tech Stack & Key Integration

- 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.

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

STEP 1 : Configure Infrastructure In AWS Cloud

- 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: <Instance_Name>
- AMI (Amazon Machine Image): Ubuntu Server 22.04 LTS (HVM), SSD Volume type.
- Instance Type: t3.xlarge (4 vCPUs, 16 GB RAM) o
- Key Pair: Select an existing key pair or create a new one.
- Storage: Default (e.g., 30GB GP3 SSD, adjust as needed

STEP 2 : Configure Security Group

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

Connect EC2 Instance through terminus,mobaxterm

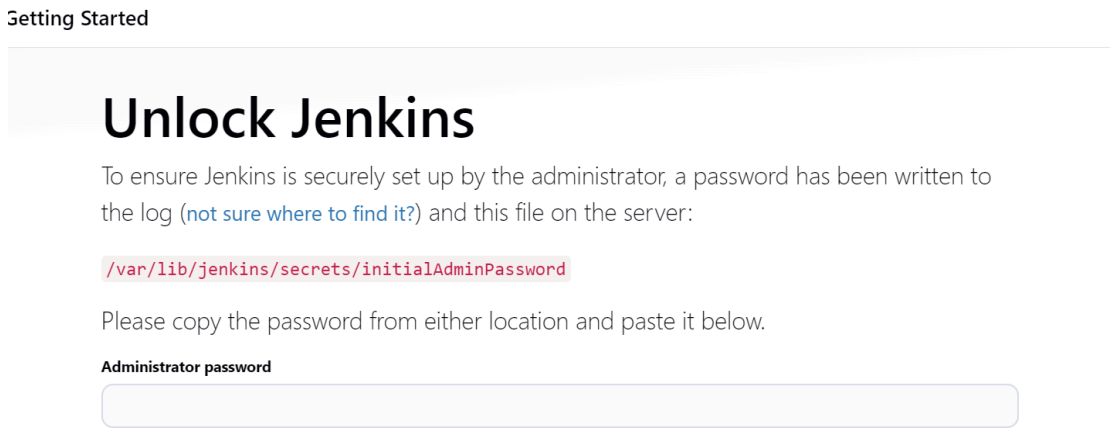
STEP 2 : Install Jenkins, Docker , awscli, terraform, kubectl , eksctl and Trivy

- Clone the GITHUB Project repositories https://github.com/Shwetanarwade/amazon-prime-video-kubernetes.git
- cd hotstar-kubernetes/scripts/
- Install the TOOLS in the VM machine via Scrtipts .
- add executable permission to shell script

```
chmod +x *.sh
```

• Access Jenkins in your browser:

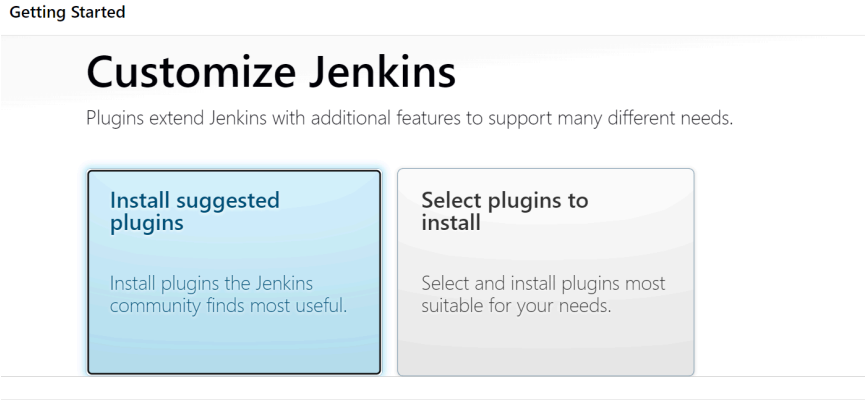
http://<public_TP>:8080



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

  sudo cat /var/lib/jenkins/secrets/initialAdminPassword Unlock Jenkins using an

- administrative password and install the suggested plugins.



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

Install Plugins like JDK, SonarQube Scanner, NodeJs, OWASP Dependency Check Goto Manage Jenkins →Plugins → Available Plugins → Install below plugins

1. Eclipse Temurin Installer (Install without restart)

2. SonarQube Scanner (Install without restart)

3. NodeJs Plugin (Install Without restart) – 16.20.2

4. OWASP Dependency Check Plugins

5. Stage view

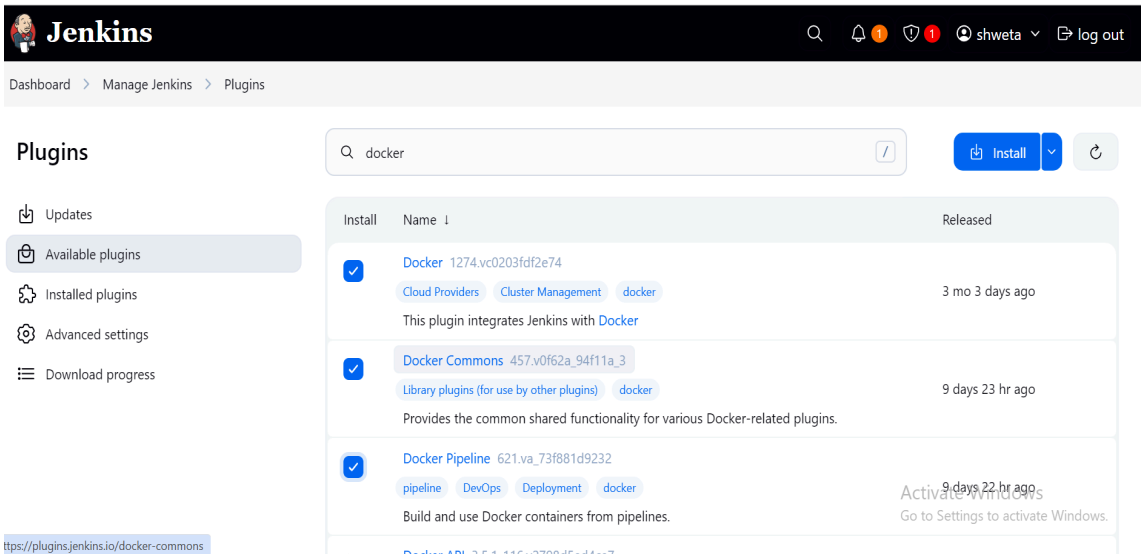
6. jdk Docker plugin

7. Docker 8. Docker Commons

9. Docker Pipeline

10. Docker API

11. Docker-build-step



- Setup SonarQube Server
- we create a sonarqube container

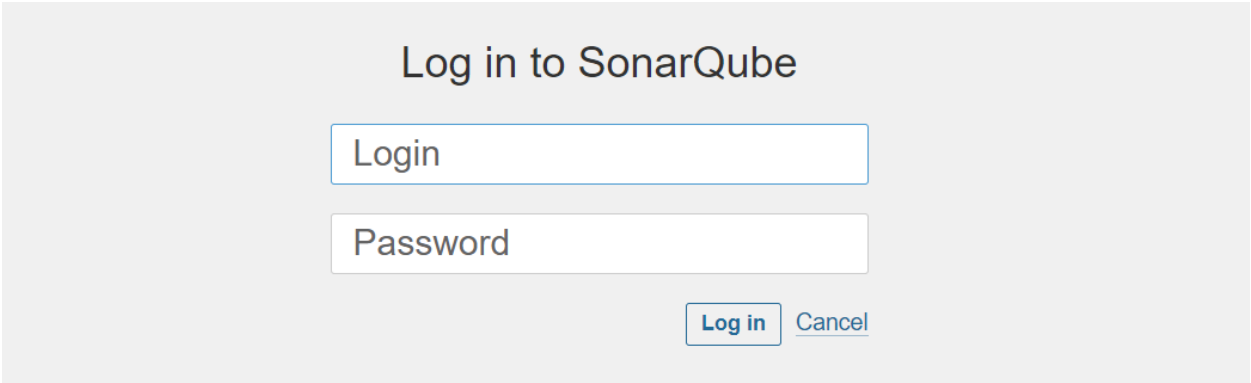
  docker run -d --name sonar -p 9000:9000 sonarqube: lts-community

```
ubuntu@ip-172-31-4-238:~/hotstar-kubernetes/scripts$ docker run -d --name sonar
:9000 sonarqube:lts-community
Unable to find image 'sonarqube:lts-community' locally
lts-community: Pulling from library/sonarqube
89dc6ea4eae2: Pull complete
31436012ac5b: Pull complete
2d16eb76e762: Pull complete
ac81863d97cb: Pull complete
26f6dfeccc10: Pull complete
a7343cdb8fb1: Pull complete
7c8e090ec954: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:1f0acc4139fc0bf09f39b962a63b56b10393cb0d07a4aa28346ea0af5b95f764
Status: Downloaded newer image for sonarqube:lts-community
cf1e37d1fe68d89a6ffe2e352a718c13b922e48aa69e6c887c87f2eae7e93d50
```

- Now SonarQube server is running you can access it from

  <publicIP of Instance>:9000

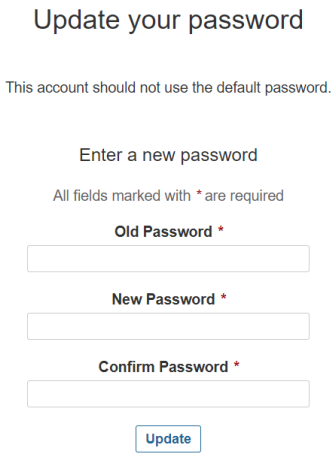
- Login to SonarQube (default username and password is : admin-admin)



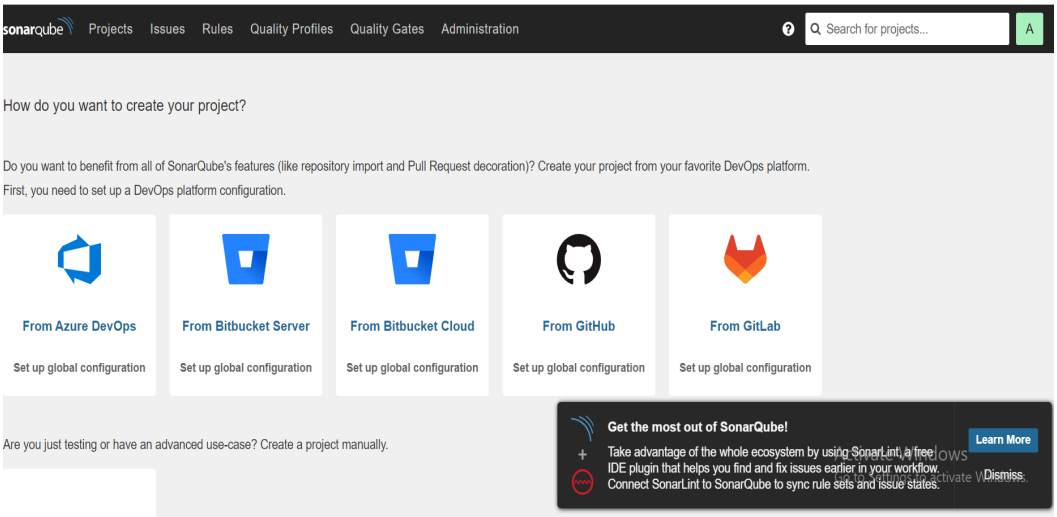
- Enter username and password, click on login and change password username admin password admin

  username admin

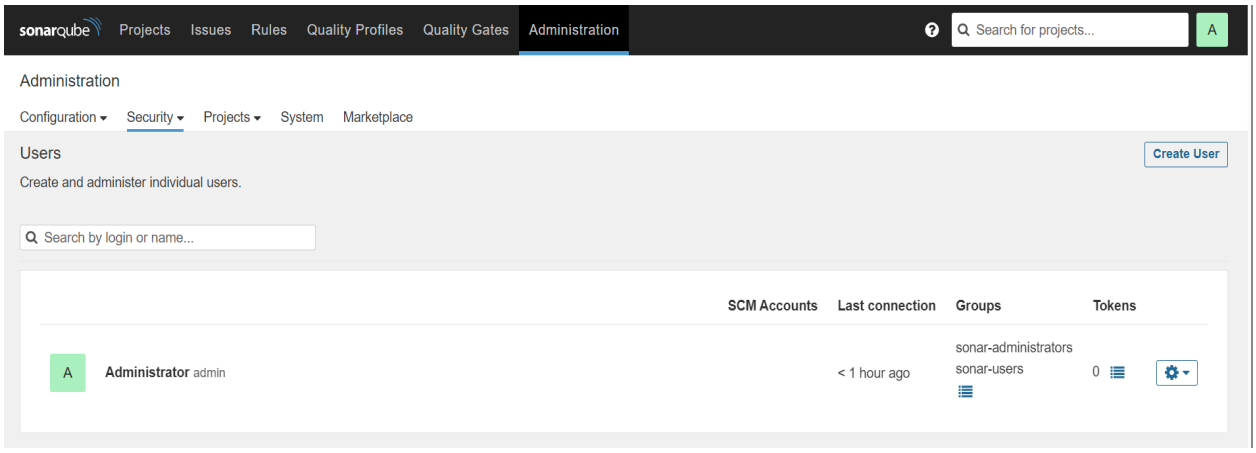
  password admin



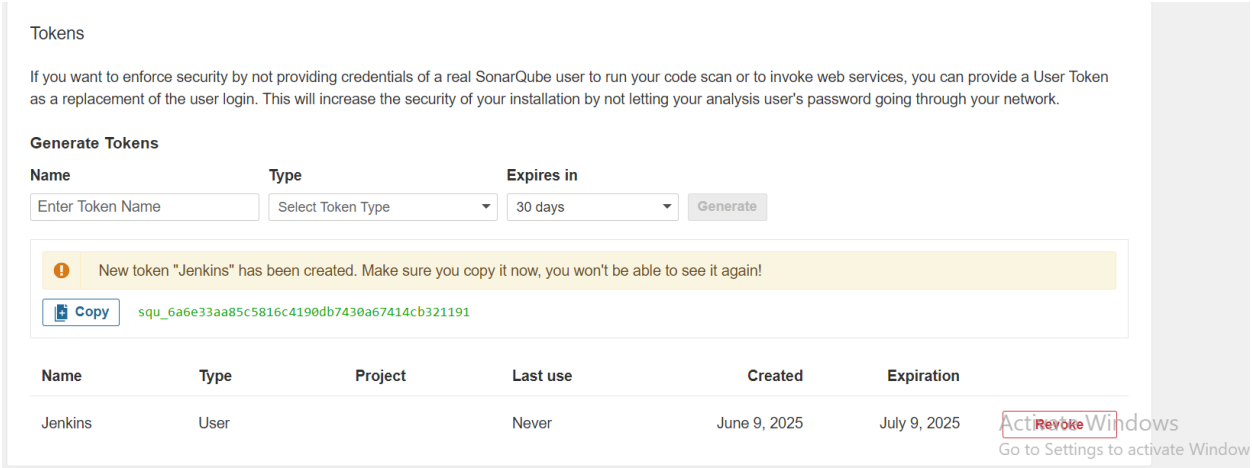
Update New password, This is Sonar Dashboard



B. - Create Sonar token in order to connect with Jenkins Click on Administration → Security → Users → Click on Tokens and Update Token → Give it a name → and click on Generate Token

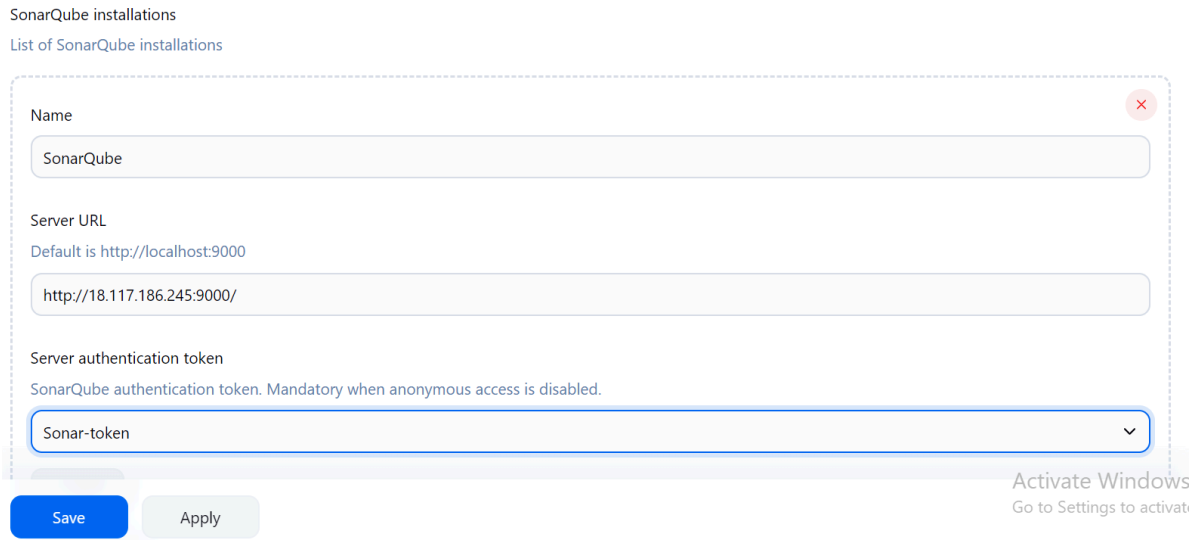


Create a token with a name and generate



Go to manage jenkins & add Credentials → Add Secret Text. It should look like this

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

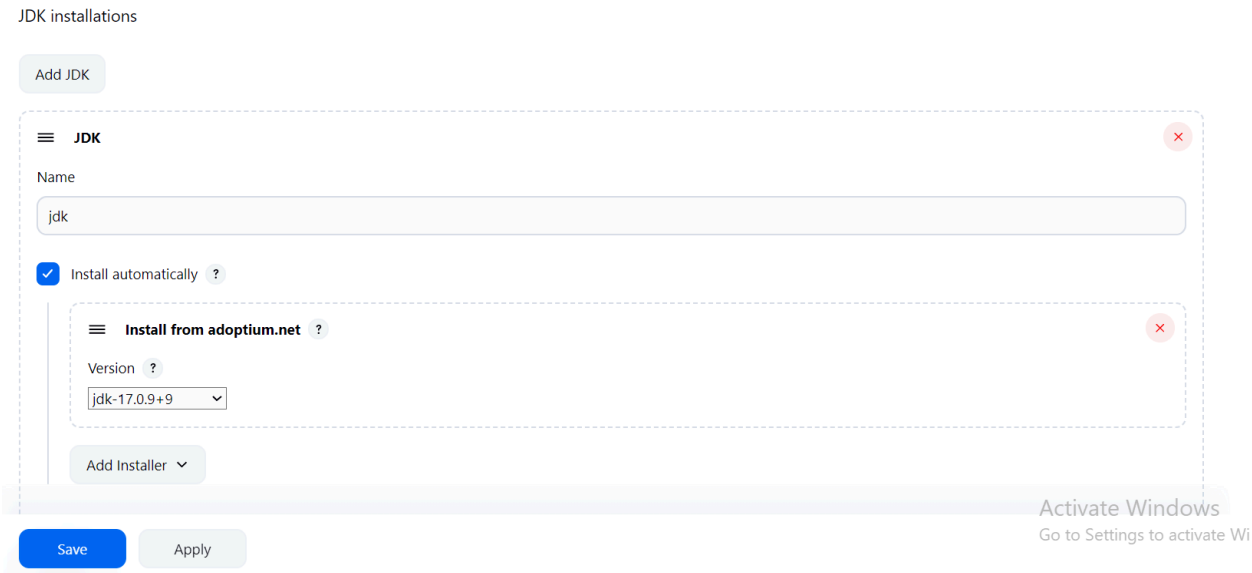


The Configure System option is used in Jenkins to configure different server Global Tool Configuration is used to configure different tools that

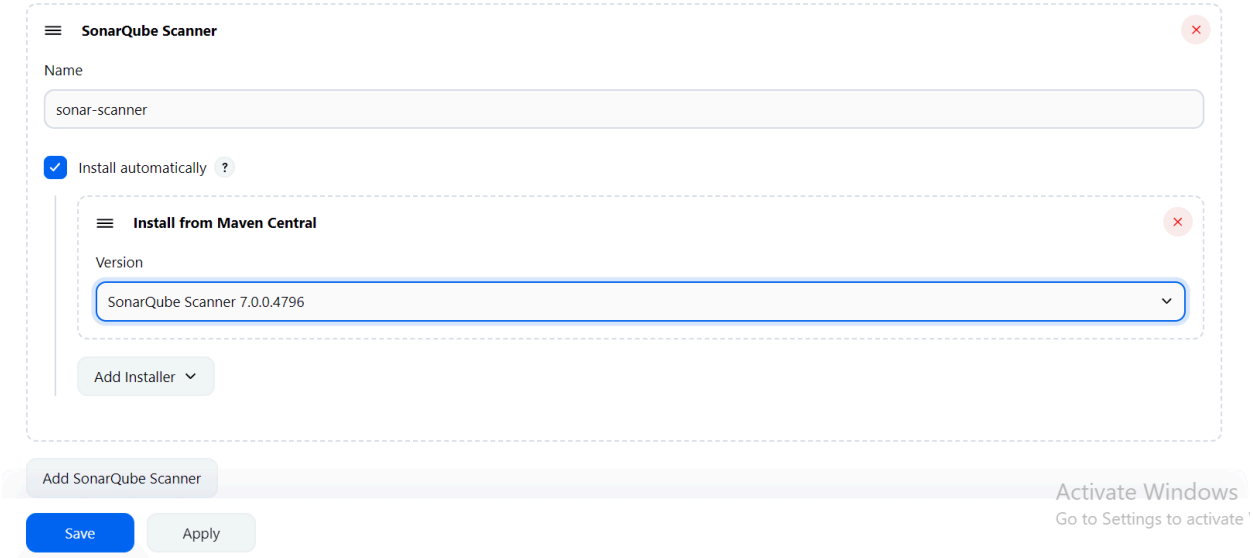
We install using Plugins We will install a sonar scanner in the tools.

Manage Jenkins –> Tools –> SonarQube Scanner

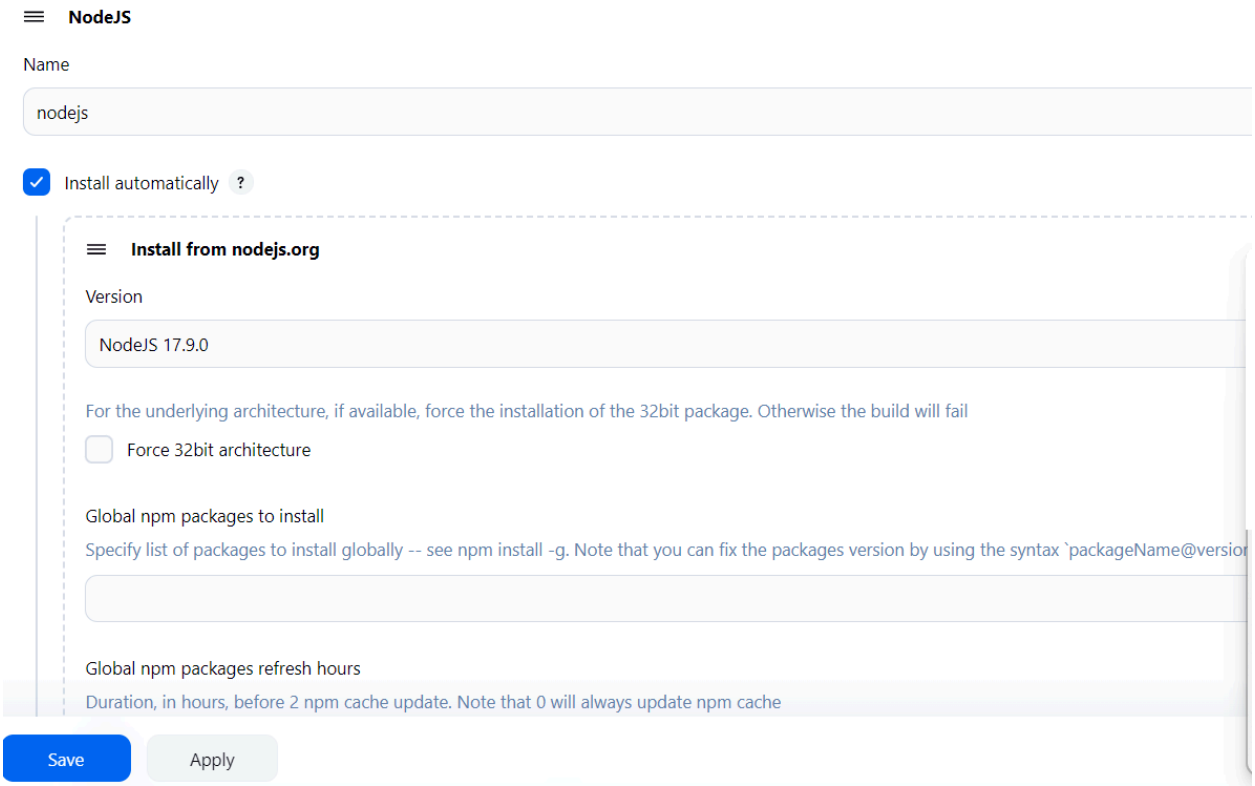
1.JDK Configuration



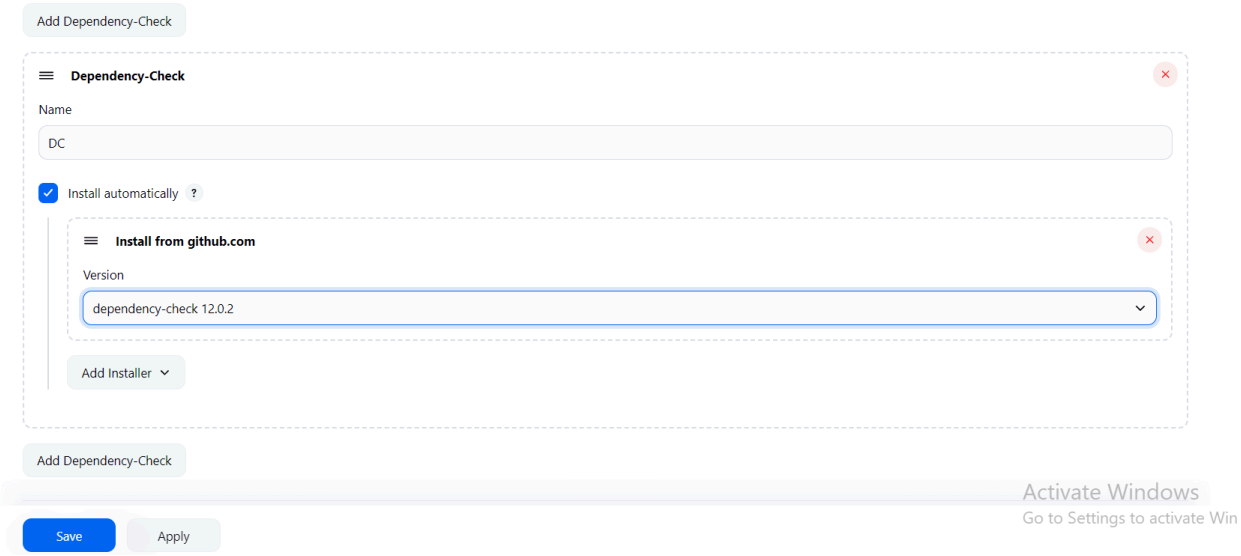
2. Configuration of sonar-scanner



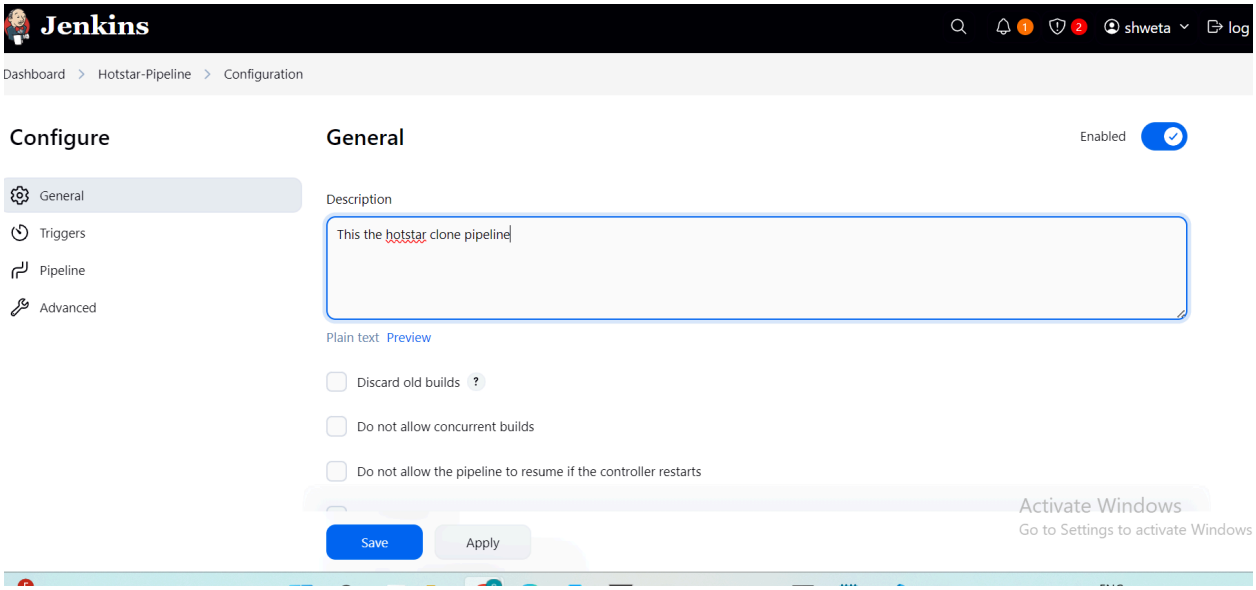
3.Configuration of nodejs



4.Configure Dependency Check



Create Job for Hotstar Let's add a pipeline :



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).

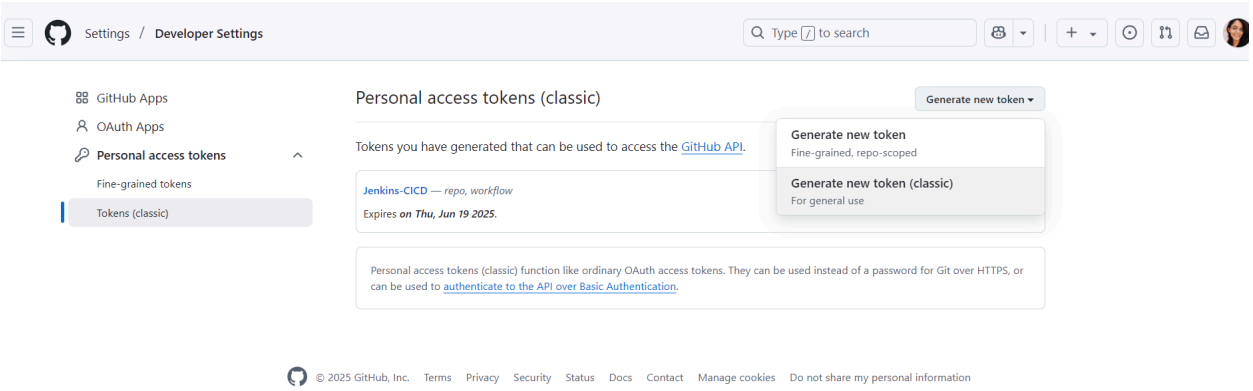
Set Scopes:

o repo → Full control of private repositories.

o admin:repo_hook → Manage repository webhooks.

o workflow → Required for GitHub Actions (optional).

Click Generate token and copy the token



Configure Jenkins with GitHub Token Add Credentials in Jenkins

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

2. Click Global Credentials (Unrestricted) → Add Credentials.

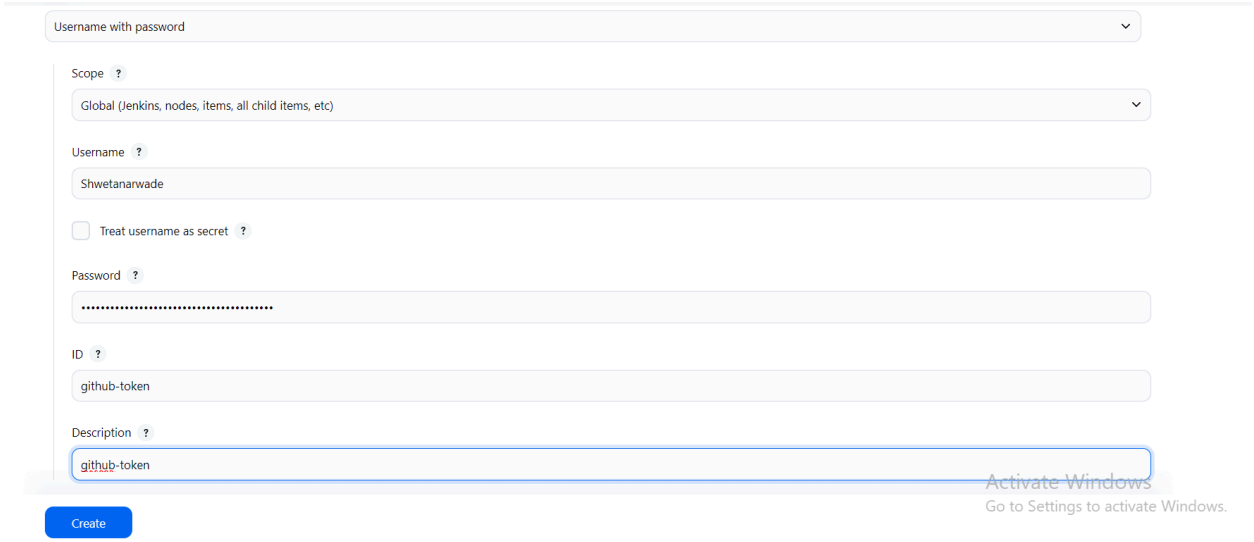
3. Select: o Kind: Username and password

o Username: Your GitHub username

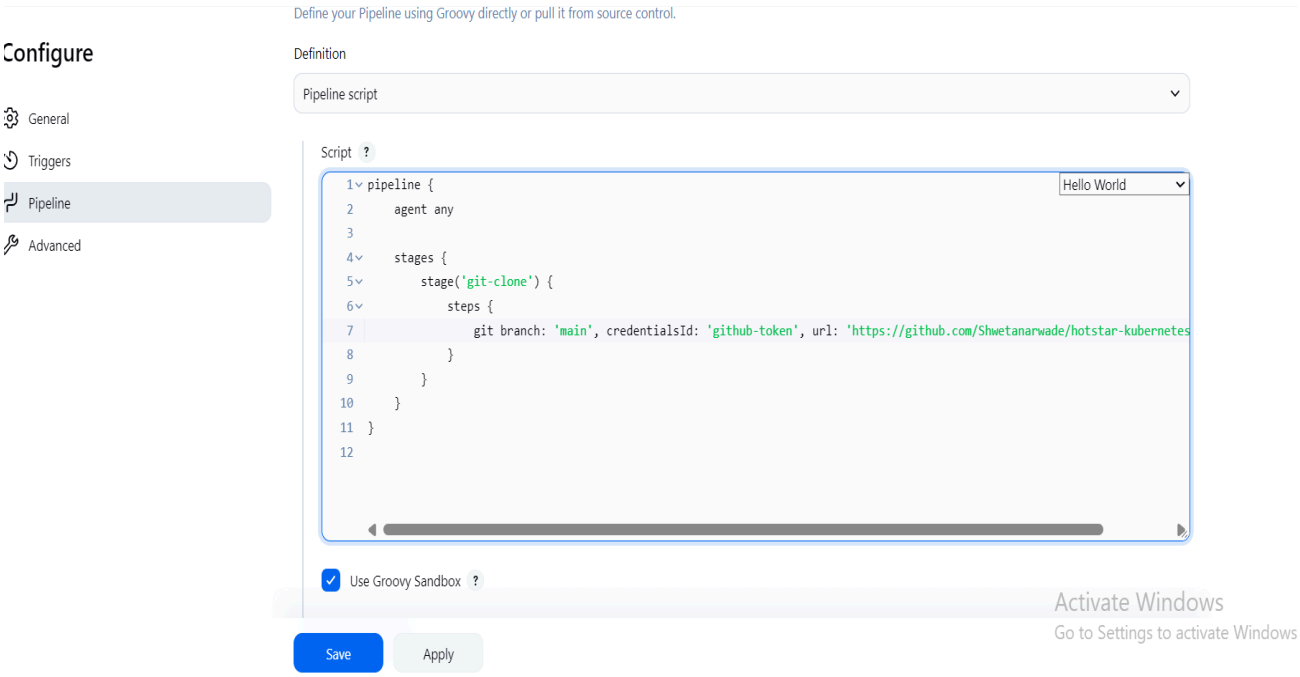
o Password: Paste your GitHub Token

o ID: github-token

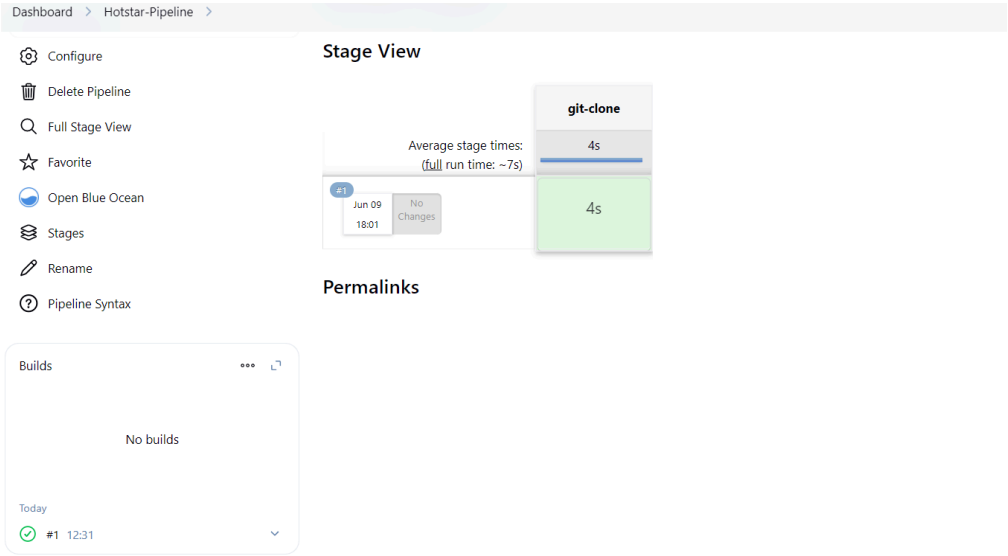
o Description: GitHub Classic Token



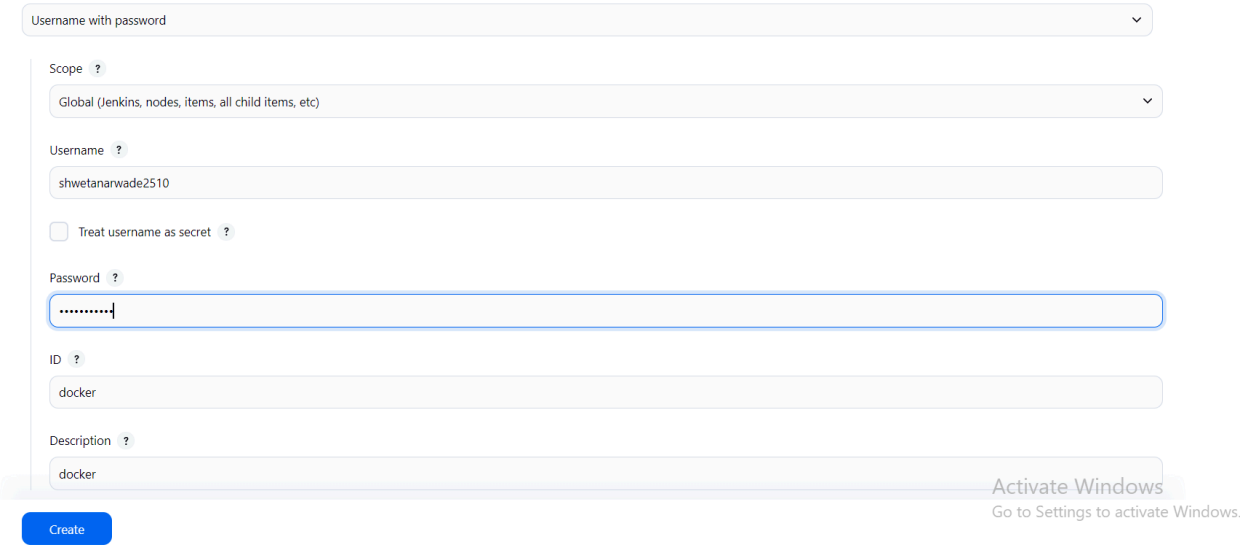
Create Job for Hotstar Let's add a pipeline , to test the Github Clone stage of Private Registry



Apply and Save and click on Build



Add docker credentials

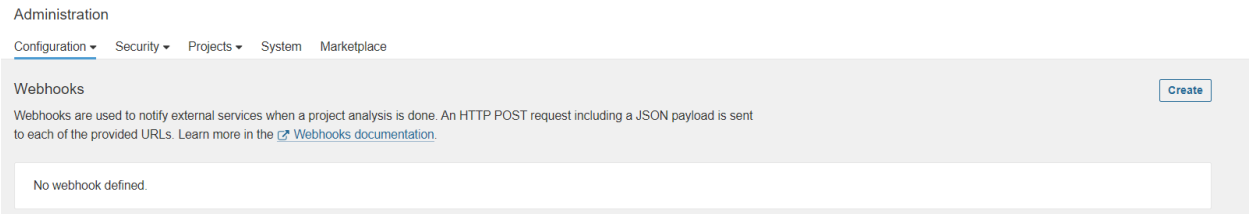


Install Dockers scout in the app server Install Docker Scout: docker login Give Dockerhub credentials here curl - sSfL https://raw.githubusercontent.com/docker/scout-cli/main/install.sh | sh -s -- -b /usr/local/bin

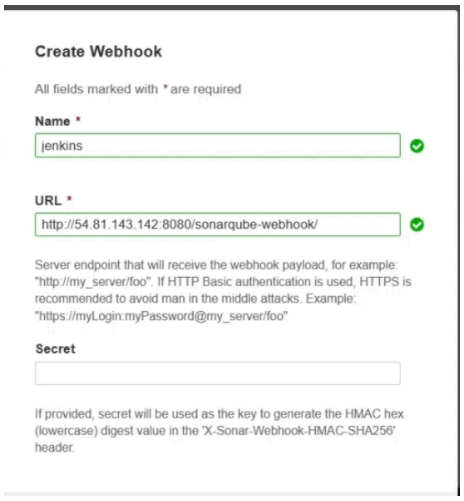
```
Username: aseemakram19
Password:
WARNING! Your password will be stored unencrypted in /home/ubuntu/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
ubuntu@ip-10-0-1-103:/var/lib/jenkins/workspace/amazon-prime-video$ curl -sSfL https://raw.g
ithubusercontent.com/docker/scout-cli/main/install.sh | sh -s -- -b /usr/local/bin
[info] fetching release script for tag='v1.17.1'
[info] using release tag='v1.17.1' version='1.17.1' os='linux' arch='amd64'
install: cannot create regular file '/usr/local/bin/docker-scout': Permission denied
[error] failed to install docker-scout
ubuntu@ip-10-0-1-103:/var/lib/jenkins/workspace/amazon-prime-video$ sudo su
root@ip-10-0-1-103:/var/lib/jenkins/workspace/amazon-prime-video# curl -sSfL https://raw.git
hubusercontent.com/docker/scout-cli/main/install.sh | sh -s -- -b /usr/local/bin
[info] fetching release script for tag='v1.17.1'
[info] using release tag='v1.17.1' version='1.17.1' os='linux' arch='amd64'
[info] installed /usr/local/bin/docker-scout
root@ip-10-0-1-103:/var/lib/jenkins/workspace/amazon-prime-video#
```

In the Sonarqube Dashboard add a quality gate also

Administration–> Configuration–>Webhooks



Create a webhook



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

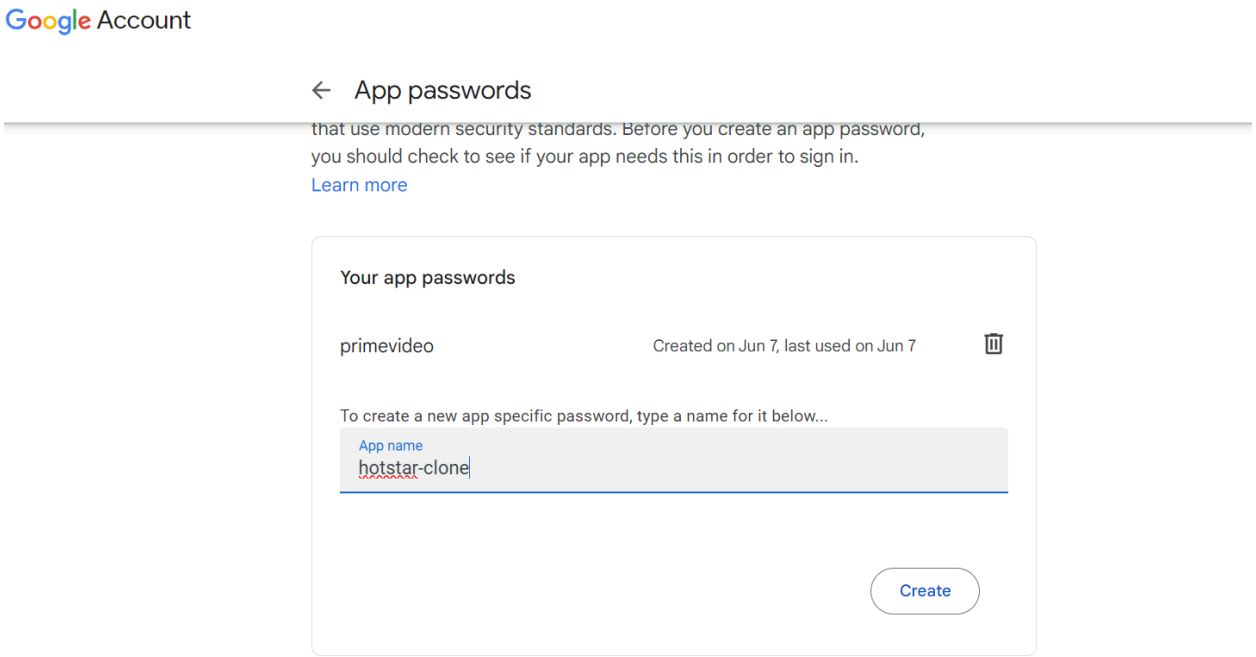
2. Sign in with your Google Account.

3. Under "Select app", choose "Mail".

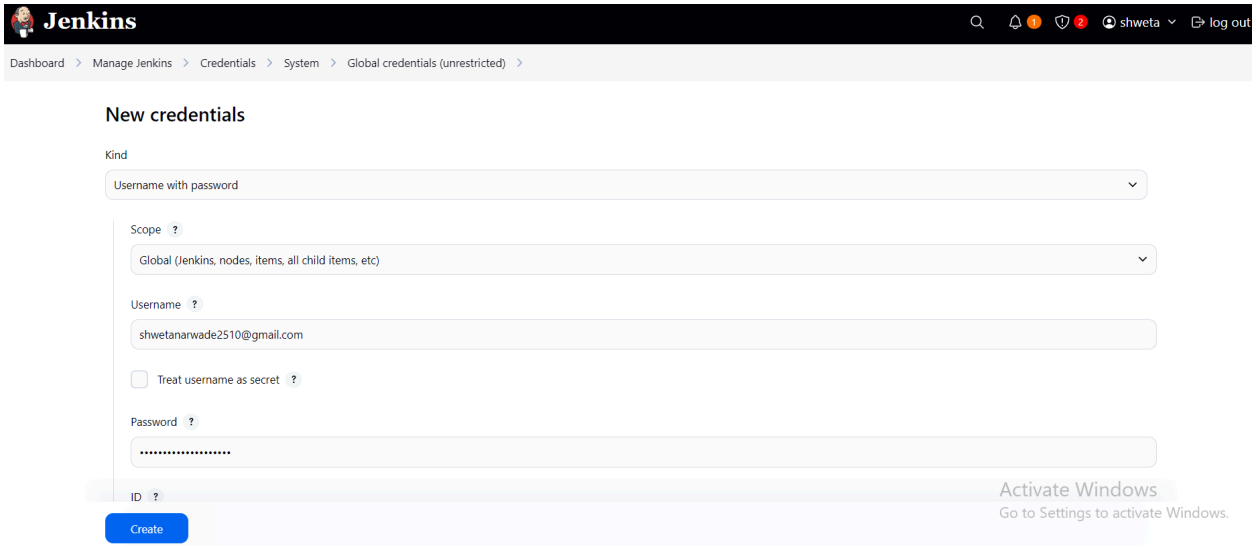
4. Under "Select device", choose "Other (Custom Name)" and enter "Jenkins SMTP".

5. Click "Generate".

6. Copy the 16-character App Password (e.g., abcd efgh ijkl mnop).



Add credentials as Username and password in jenkins



Step 3: Configure Gmail SMTP in Jenkins

1. Go to Jenkins Dashboard → Manage Jenkins → Configure System.

2. Scroll to "E-mail Notification".

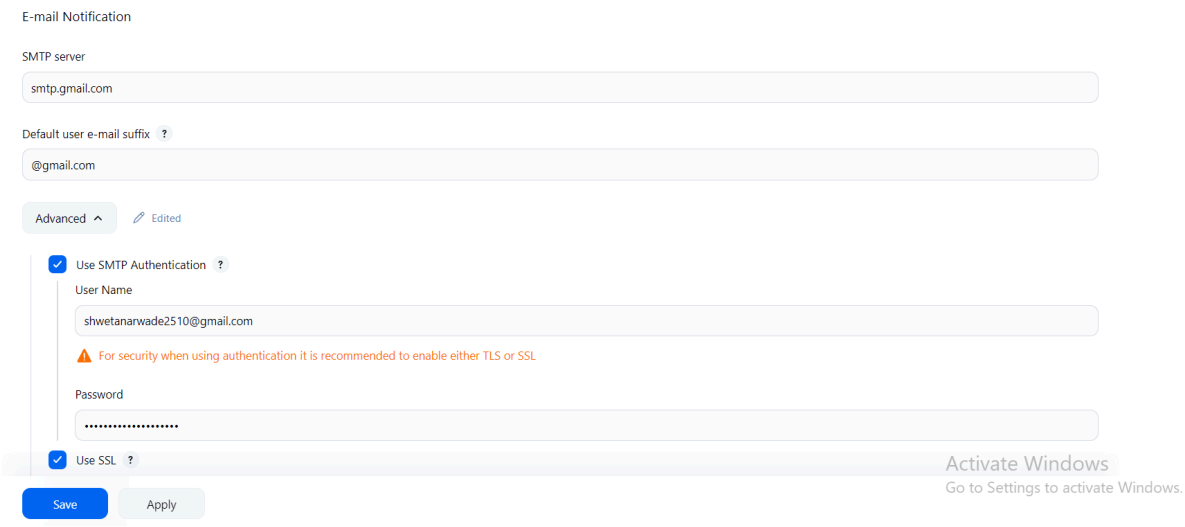
3. Set the following: o SMTP Server: smtp.gmail.com

o Use SMTP Authentication: Checked o User Name: Your Gmail ID (your-email@gmail.com)

o Password: Paste the App Password

o SMTP Port: 465

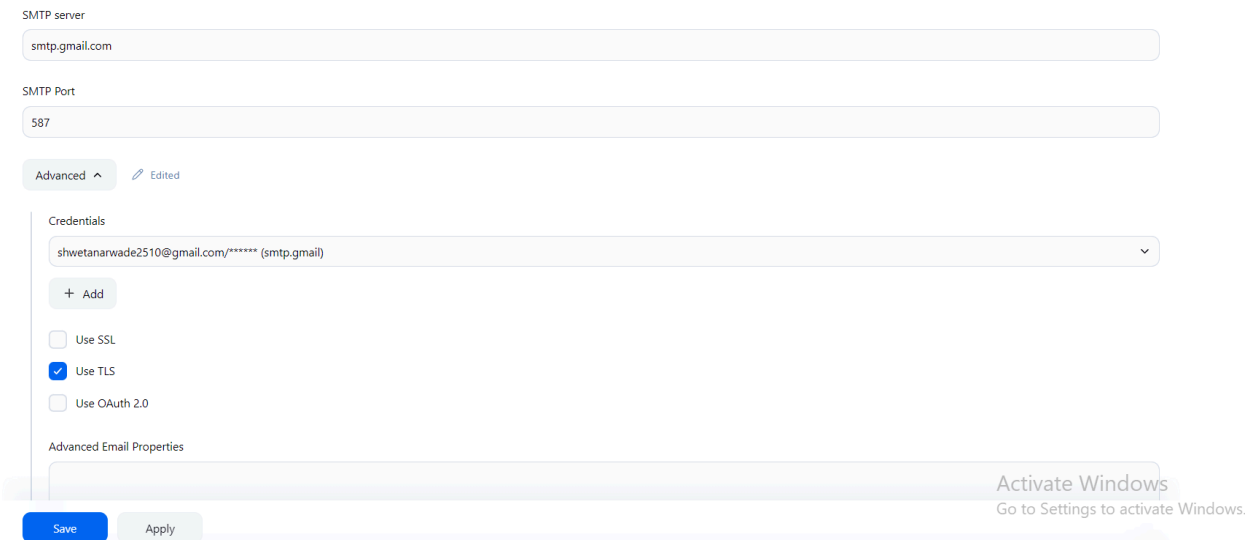
o Use TLS: Checked



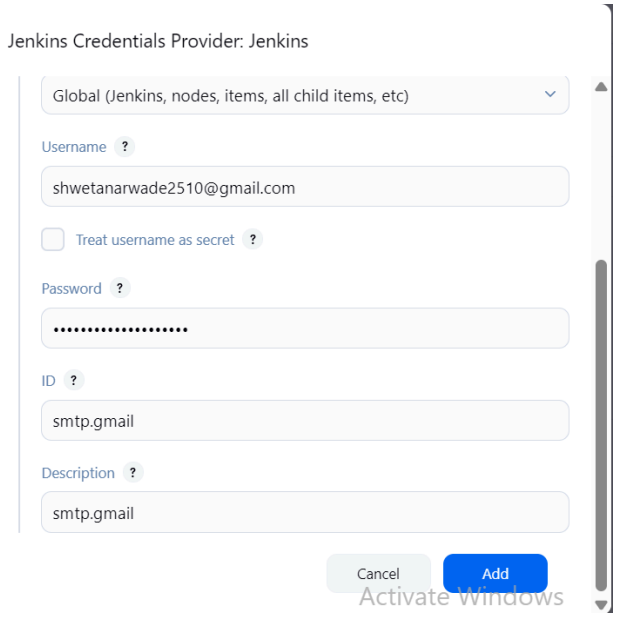
4. Click Save. Email Extension Plugin xsuc kxeb xcvk xqkf

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



Add your gmail credentials in credentials section of Jenkins



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).

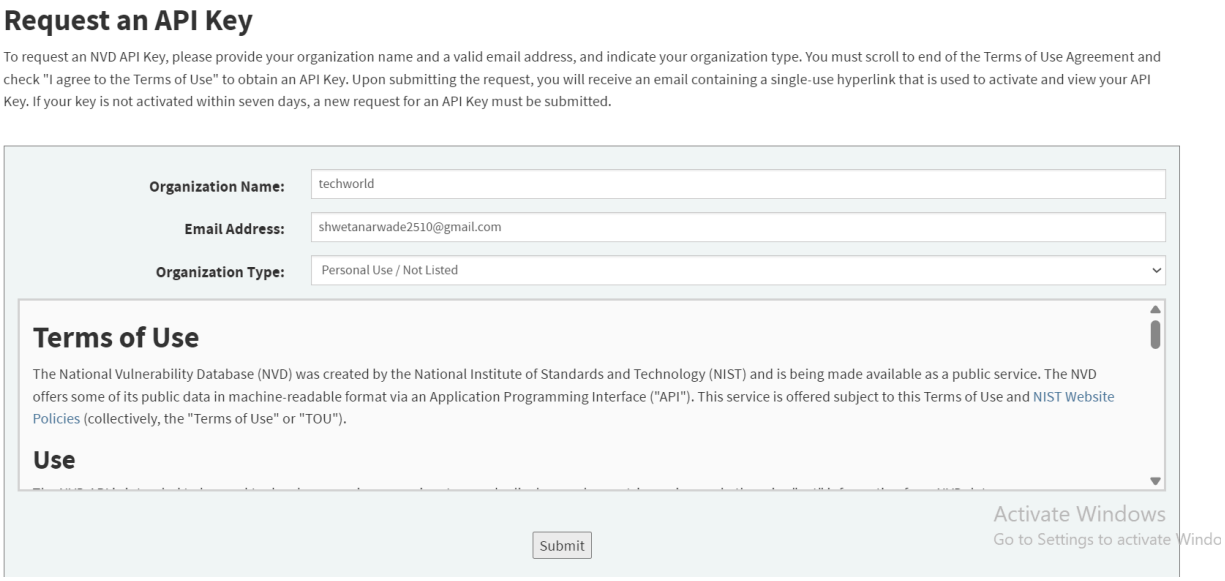
3. Request an API Key:

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.



Then Configure the pipeline

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

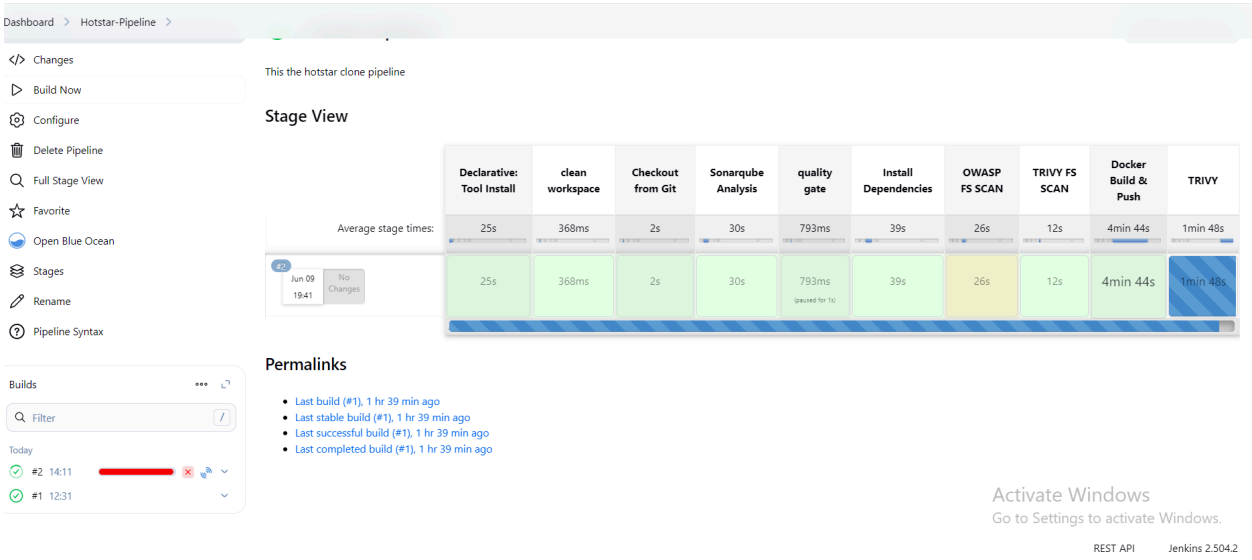
```
    stage('Checkout from Git'){
      steps{
        git branch: 'main', credentialsId: 'github-token', url:
'https://github.com/Aseemakram19/hotstar-kubernetes.git'
      }
    }
    stage("Sonarqube Analysis "){
      steps{
        withSonarQubeEnv('SonarQube') {
          sh "' $SCANNER_HOME/bin/sonar-scanner -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 shwetanarwade2510/hotstar:latest "
                sh "docker push  shwetanarwade2510/hotstar:latest "
              }
            }
        }
    }
    stage("TRIVY"){
        steps{
            sh "trivy image shwetanarwade2510/hotstar:latest > trivyimage.txt"
        }
    }
```

```
        stage('Deploy to container'){
            steps{
                sh 'docker run -d --name hotstar -p 3000:3000 shwetanarwade2510/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: 'shwetanarwade2510@gmail.com',
                from: 'shwetanarwade2510@gmail.com',
                replyTo: 'shwetanarwade2510@gmail.com',
```

```
            mimeType: 'text/html',
            attachmentsPattern: 'trivyfs.txt,trivyimage.txt'
        )
    }
  }
 }
}
```

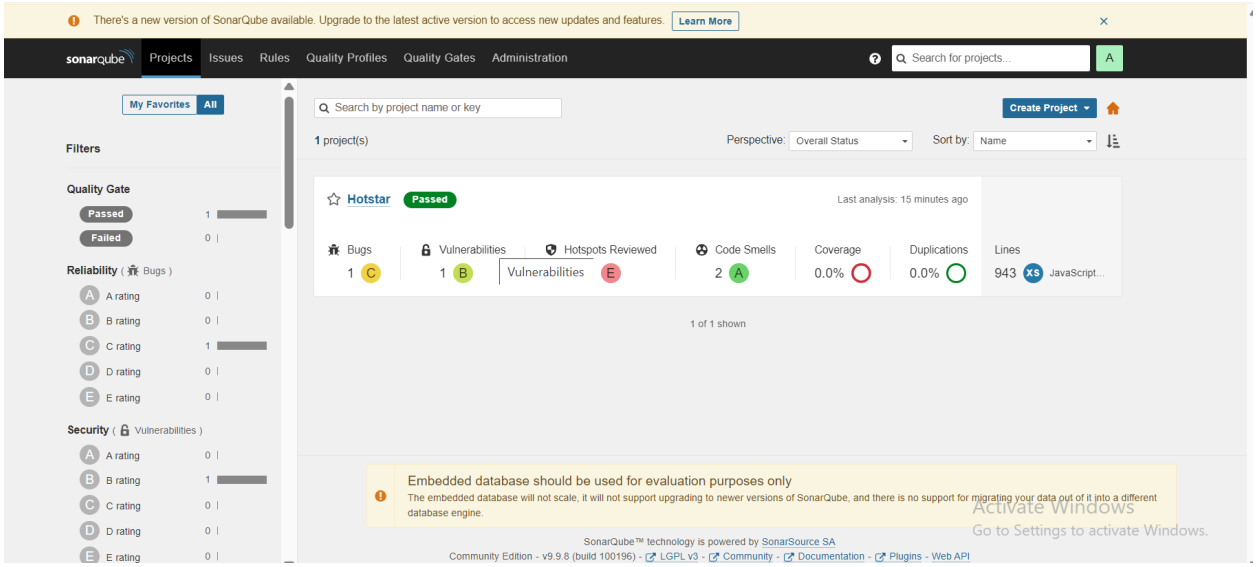
Stage View after building pipeline



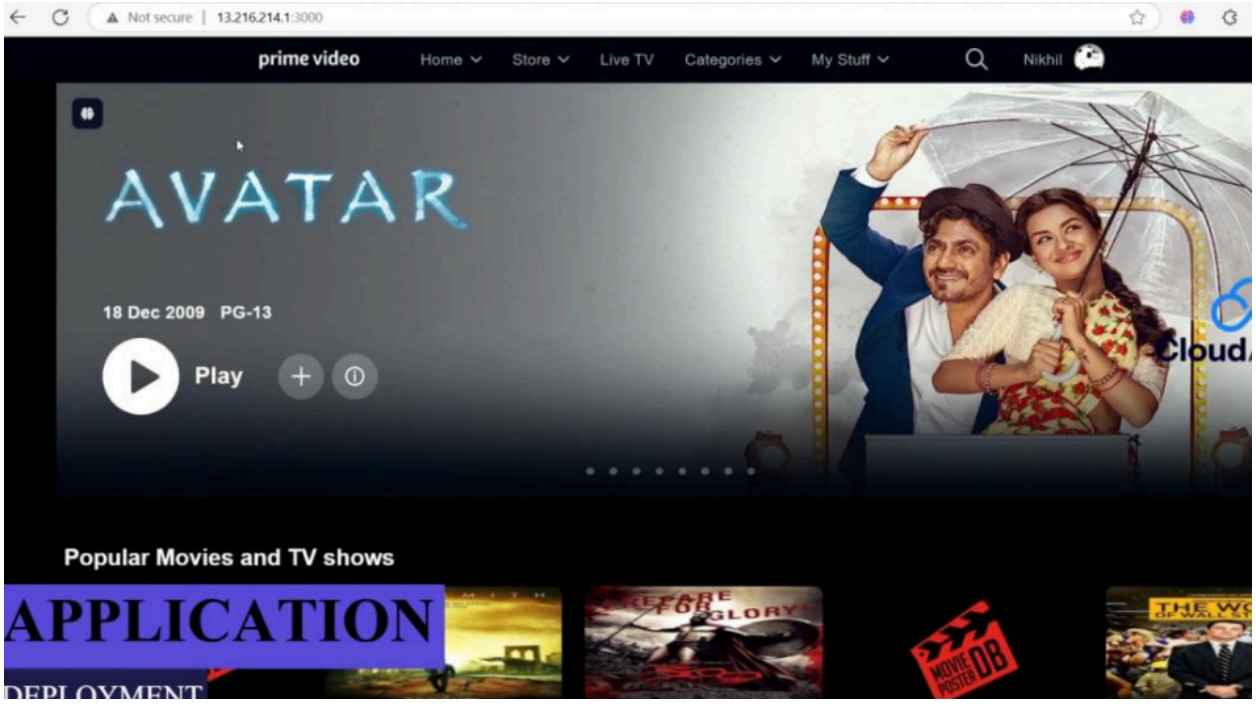
Our Application is live with this output

<publicip_of_instance>:3000

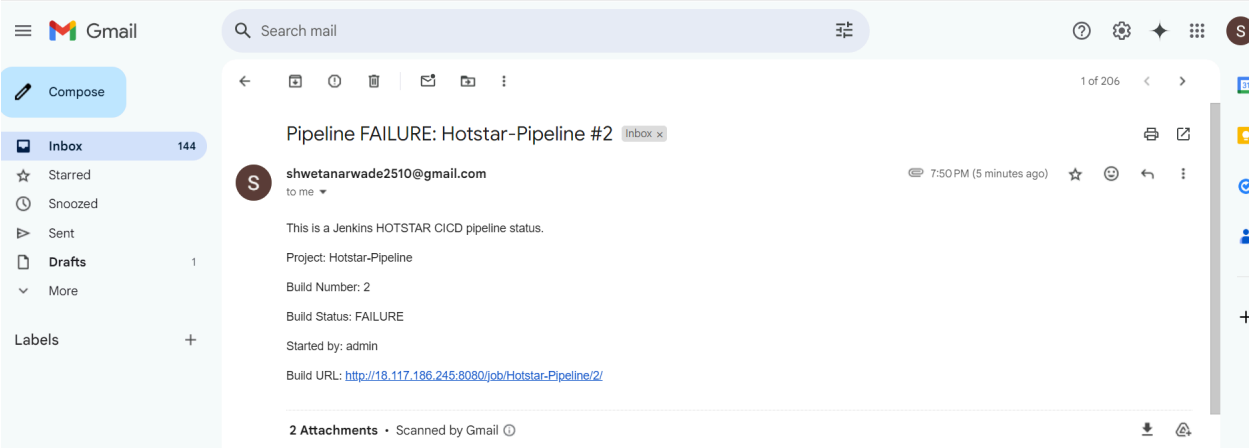
You can see the report has been generated and the status shows as passed. You can see that there are 943 lines it scanned. To see a detailed report, you can go to issues. You will see that in status, a graph will also be generated and Vulnerabilities.



Our Application is live with this output



Email alert with Post build



2. Verify Docker Images in Docker Hub