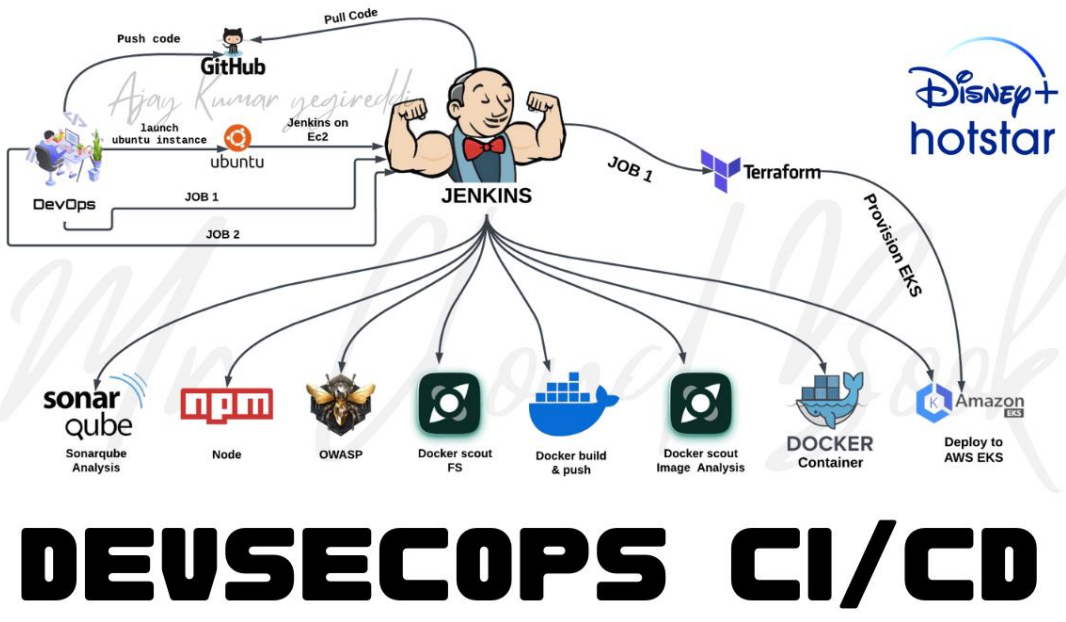
DevSecOps CI/CD: Deploying a Secure Hotstar Clone on EKS

Introduction:

This comprehensive blog guides readers through the implementation of DevSecOps principles in deploying a Hotstar clone on Amazon Web Services (AWS). DevSecOps integrates security throughout the software development lifecycle, promoting a proactive approach to identify and mitigate vulnerabilities. The process involves setting up an AWS EC2 instance with Ubuntu, configuring IAM roles, and automating tool installations using scripts. Jenkins orchestrates the deployment pipeline, creating an Amazon EKS cluster and deploying the Hotstar clone while incorporating security practices.

Key tools and services include Docker, Jenkins, Java, SonarQube, AWS CLI, Kubectl, and Terraform. Security measures, such as static code analysis, OWASP checks, and Docker Scout container scans, are seamlessly integrated. This ensures a secure and robust deployment, fortifying the application against potential threats and vulnerabilities.

Project Architecture:



Prerequisites:

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

What we are going to do?

Step 1: Setting up AWS EC2 Instance with Terraform

- Creating an EC2 instance with Ubuntu AMI, t2. 2xlarge, and 30 GB storage
- Assigning an IAM role with Admin access for learning purposes
- Installation of Required Tools on the Instance
- Writing a terraform HCL to automate the installation of:
    - Docker
    - Jenkins
    - Java
    - SonarQube container
    - AWS CLI
    - Kubectl

Step 2: Jenkins Job Configuration

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

Step 3: Clean-Up Process

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

How to do?

Step 1: Deploy our Jenkins Server (EC2) on AWS with Terraform.

- Install & Configure Terraform and AWS CLI on your local machine to create Jenkins Server on AWS Cloud

- Navigate to the Jenkins-Server-TF
- Do some modifications to the backend.tf file such as changing the bucket name and dynamodb table (make sure you have created both manually on AWS Cloud).
- Initialize the backend by running the below command

terraform init

- Run the below command to check the syntax error

terraform validate

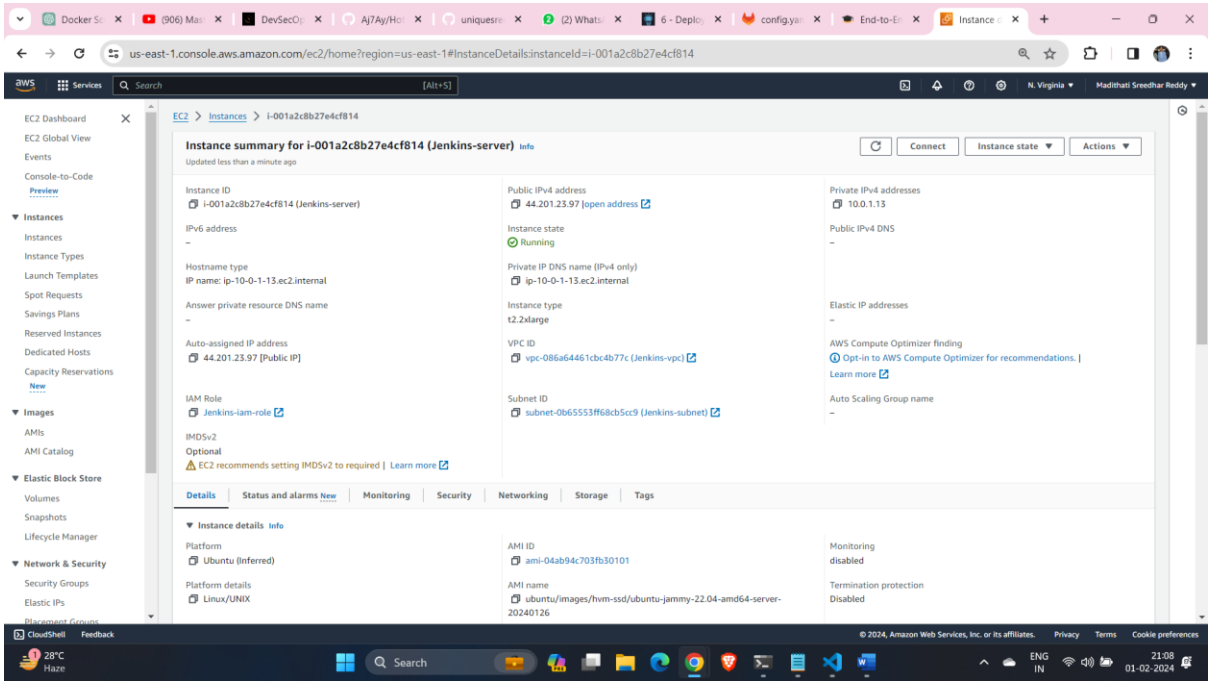
- Run the below command to get the blueprint of what kind of AWS services will be created.

terraform plan -var-file=variables.tfvars

- Now, run the below command to create the infrastructure on AWS Cloud which will take 3 to 4 minutes maximum

terraform apply -var-file=variables.tfvars --auto-approve

- Now, connect to your Jenkins-Server by clicking on Connect.



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

<Ec2-ip:8080> #you will Jenkins login page

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

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

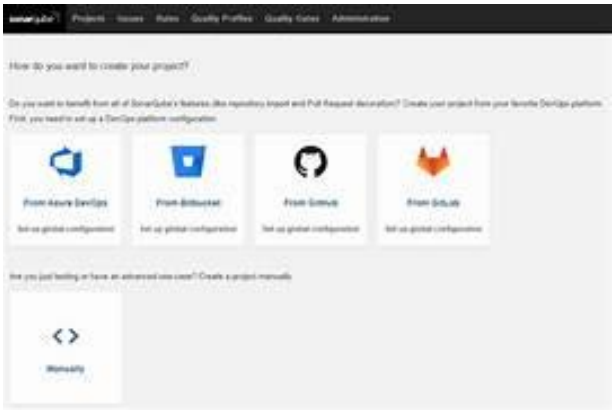
- Now, install the suggested plugins.
- Jenkins will now get installed and install all the libraries.
- Create an admin user
- Click on save and continue.
- Now Copy the public IP again and paste it into a new tab in the browser with 9000

<ec2-ip:9000> #runs sonar container

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

username admin
password admin

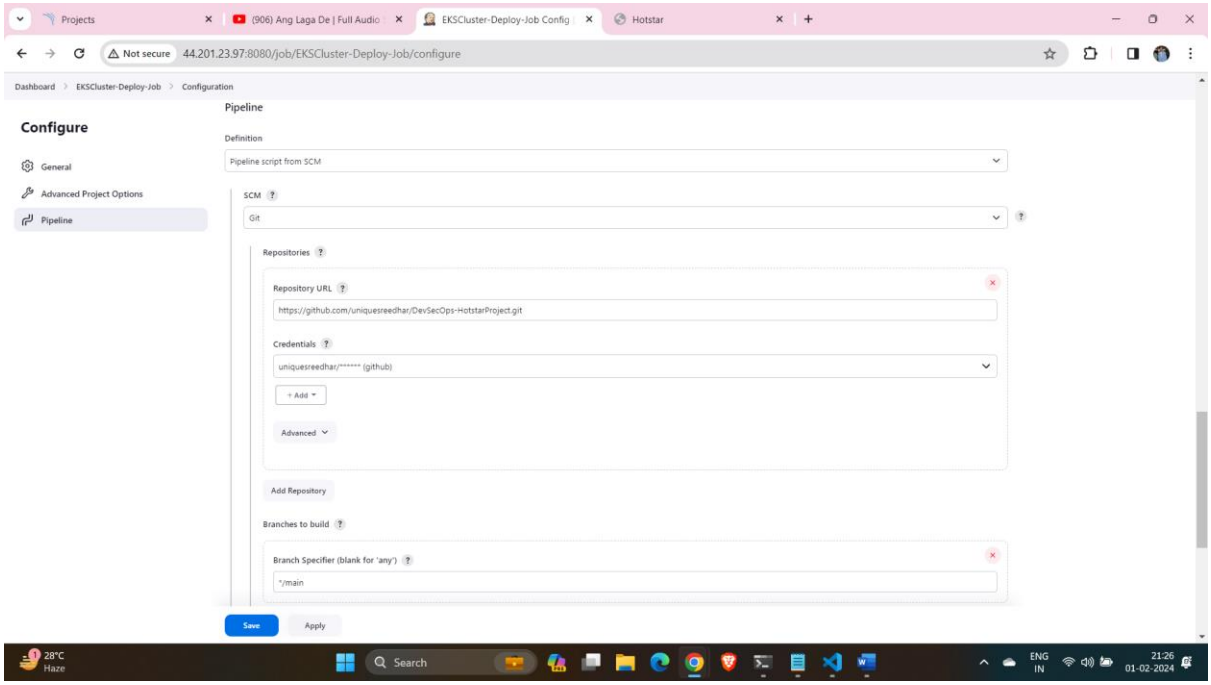
- Update New password, this is Sonar Dashboard.



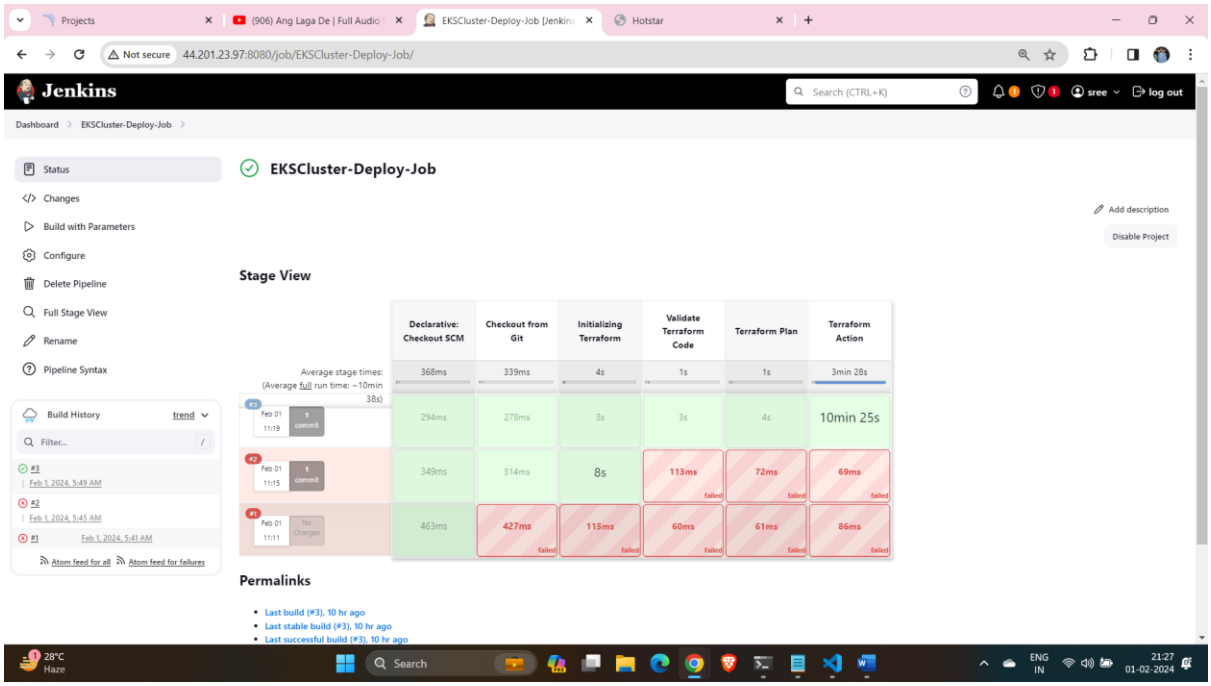
Step 2: Jenkins Job Configuration

Step 2A: EKS Provision job

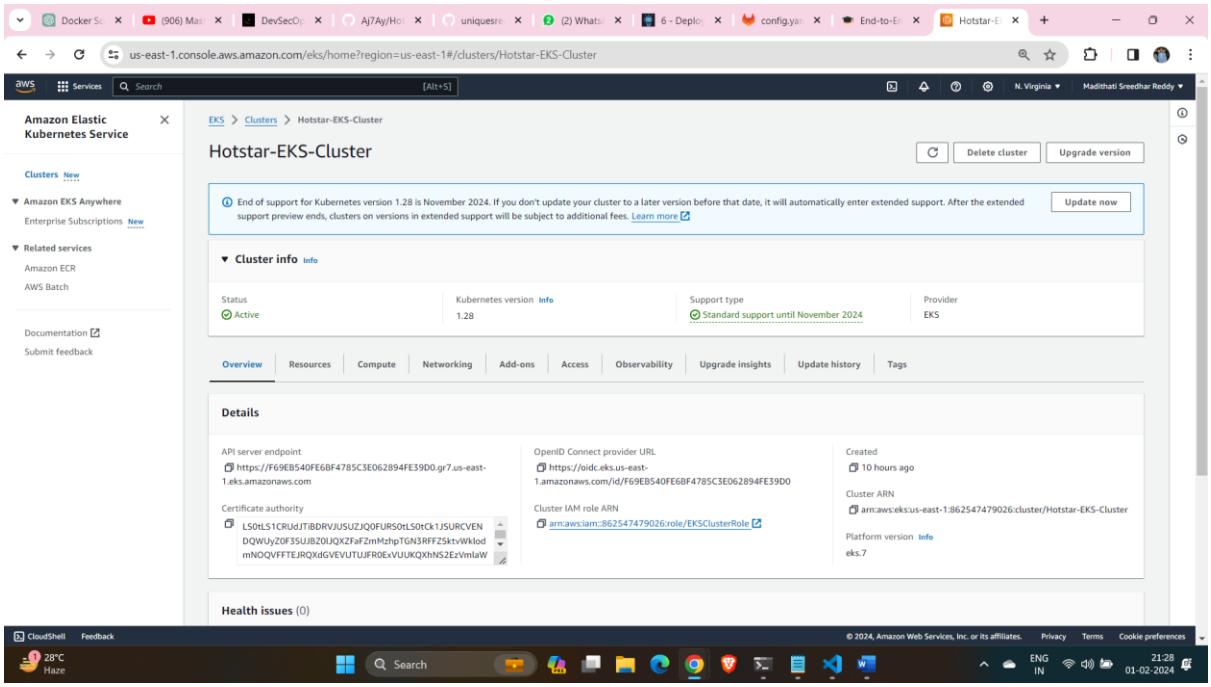
- For this you need to add the Aws credentials with "aws key" as id and github credentials with "github" as id
- Create a new job with a name "EKS-Terraform deploy"
- In the pipeline section provide the configuration as:



- Then Click on save and then build.
- After pipeline got successful it will look like:



- And the EKS cluster will be created along with nodes.



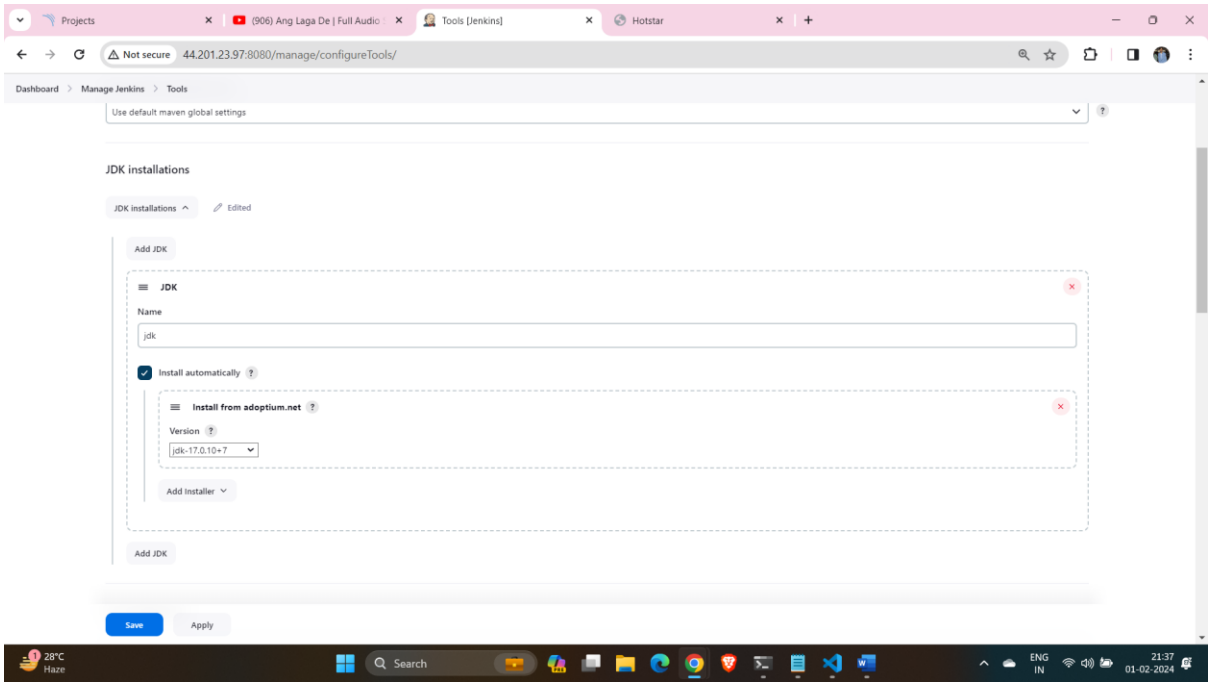
Step 2B: Hotstar job

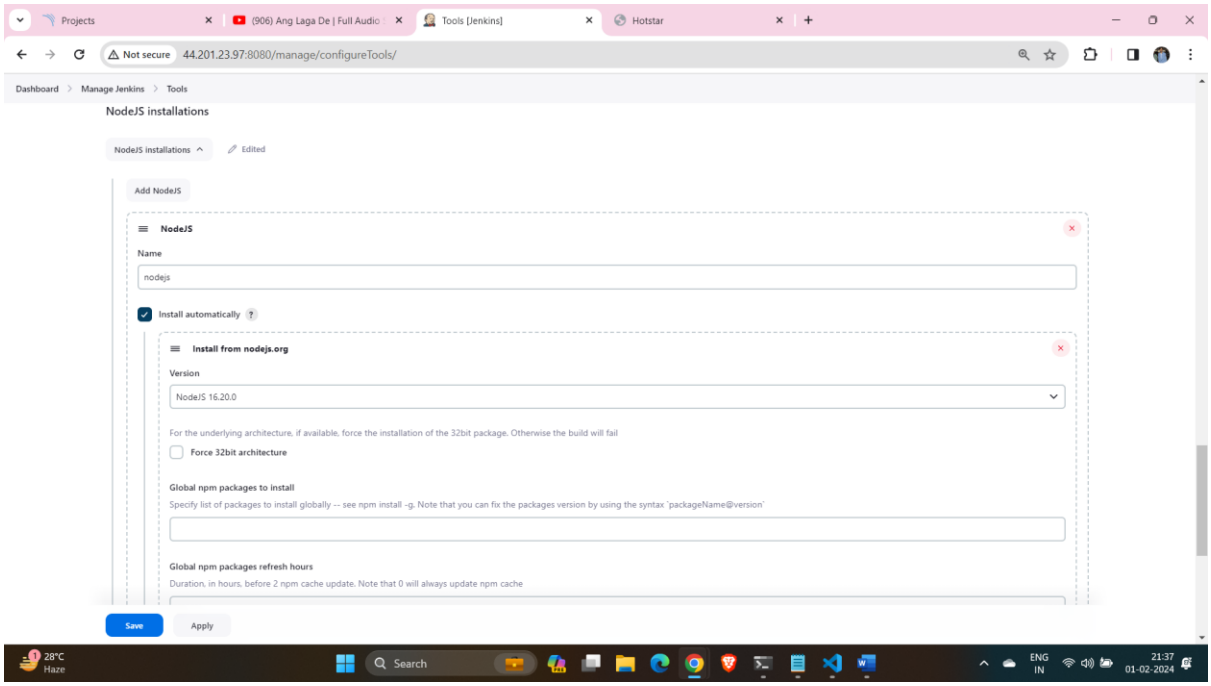
Plugin's installation & setup (Java, Sonar, Nodejs, owasp, Docker)

1. Go to Jenkins dashboard
2. Manage Jenkins –> Plugins –> Available Plugins
3. Search for the Below Plugins
    - Eclipse Temurin installer
    - Sonarqube Scanner
    - NodeJs

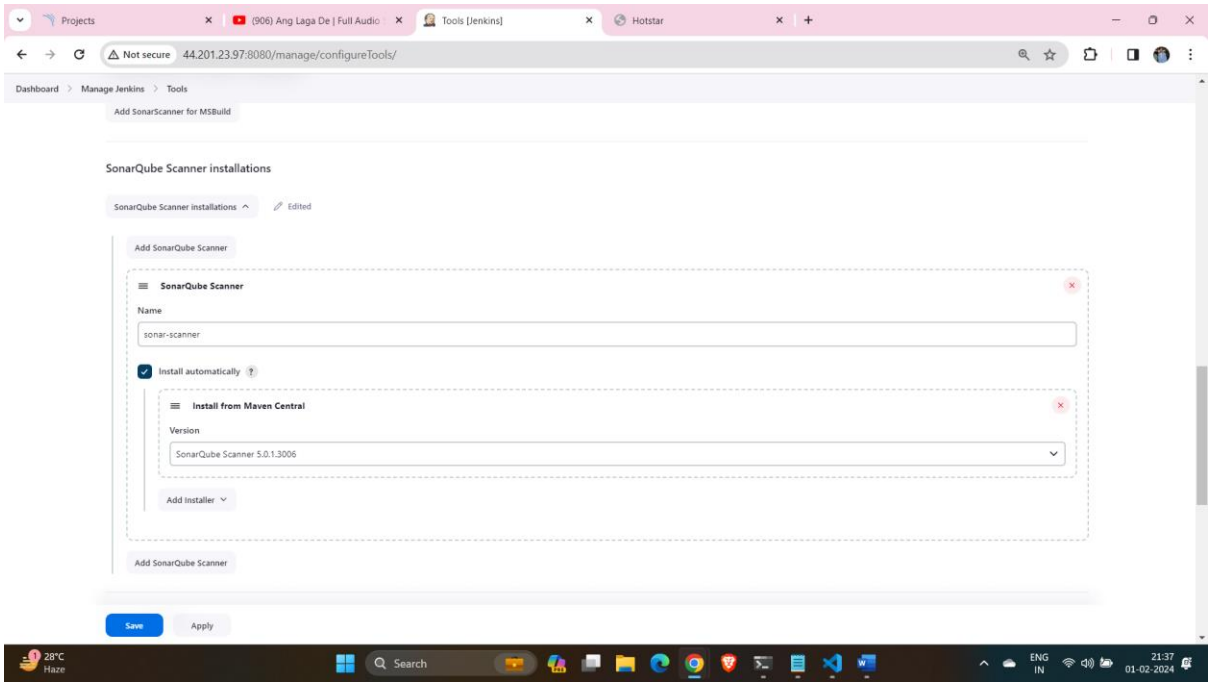
- Owasp Dependency-Check
- Docker
- Docker Commons
- Docker Pipeline
- Docker API
- Docker-build-step

4. Configure in Global Tool Configuration
    - Goto Manage Jenkins → Tools → Install JDK(17) and NodeJs(16)→ Click on Apply and Save

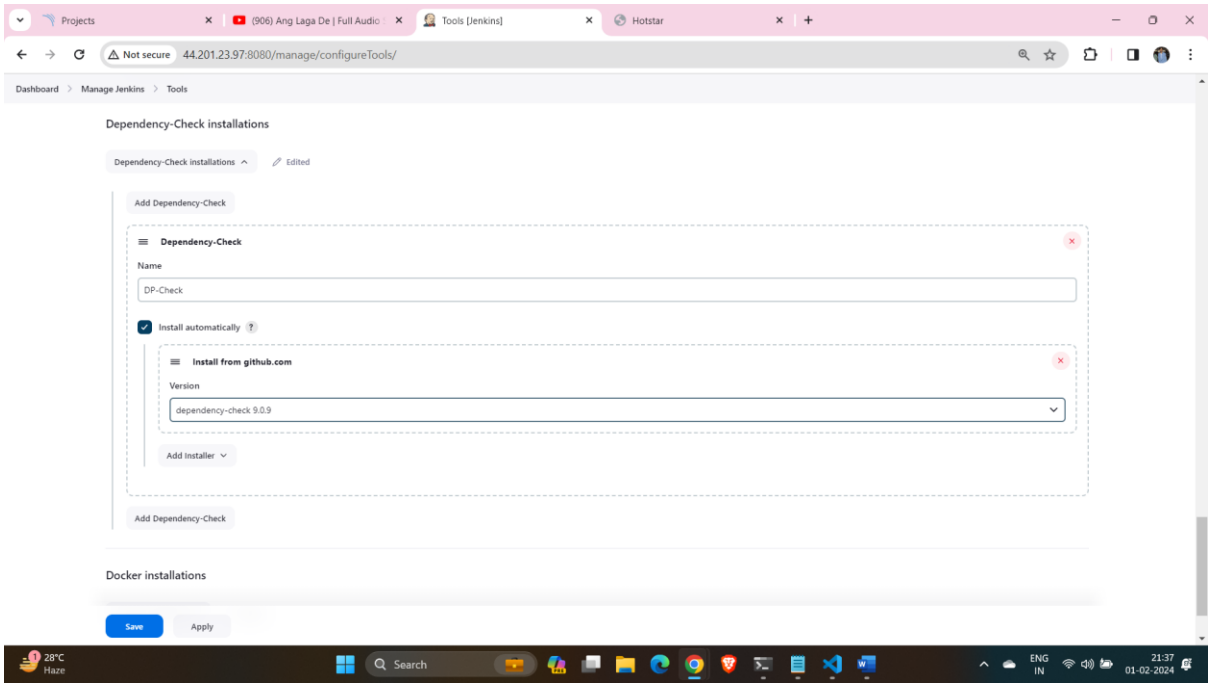




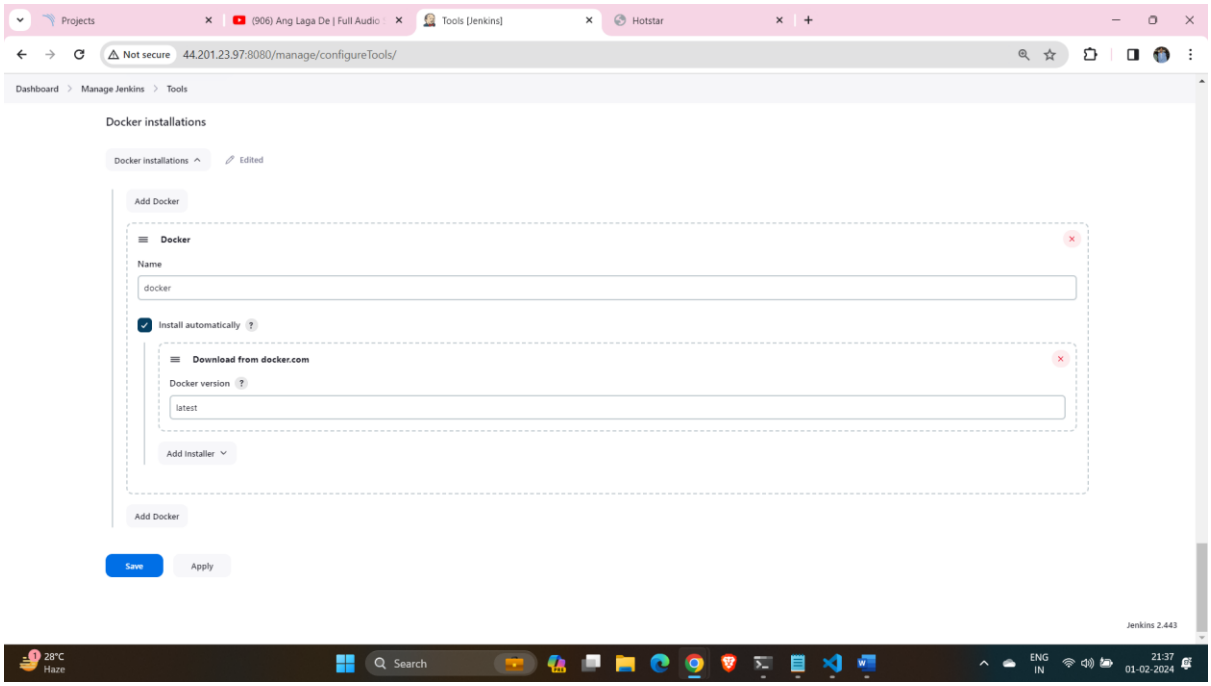
- For Sonarqube use the latest version



- For Owasp use the 9.0.7 version



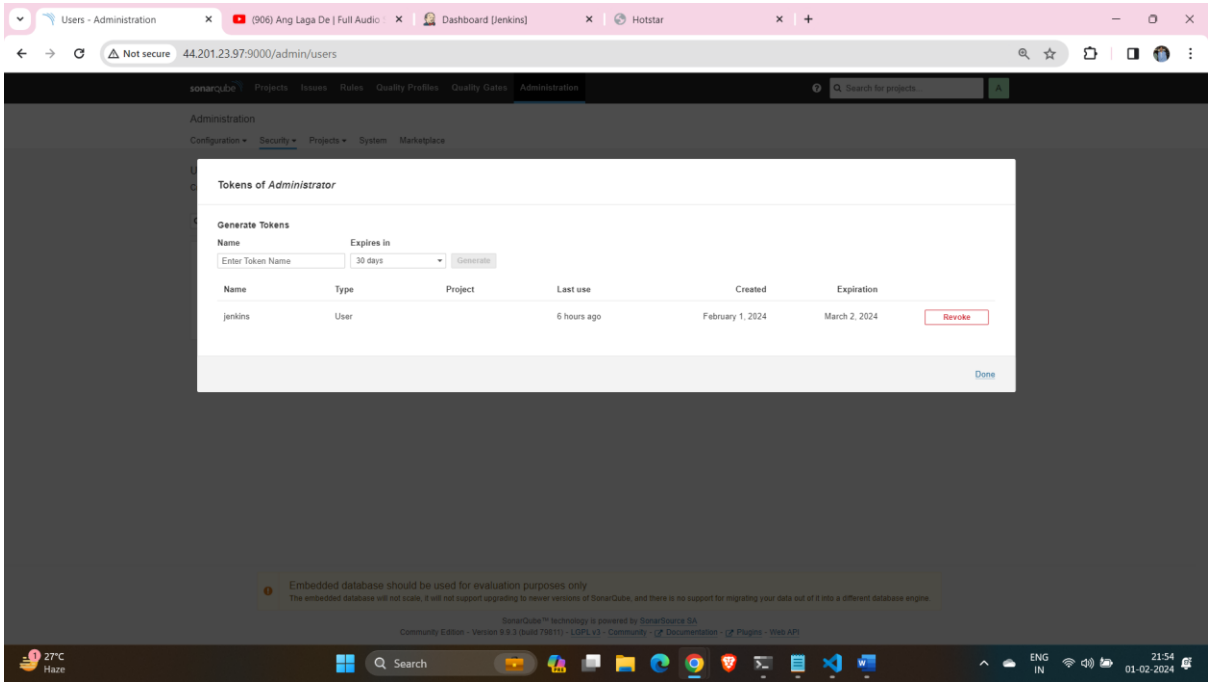
- Use the latest version of Docker



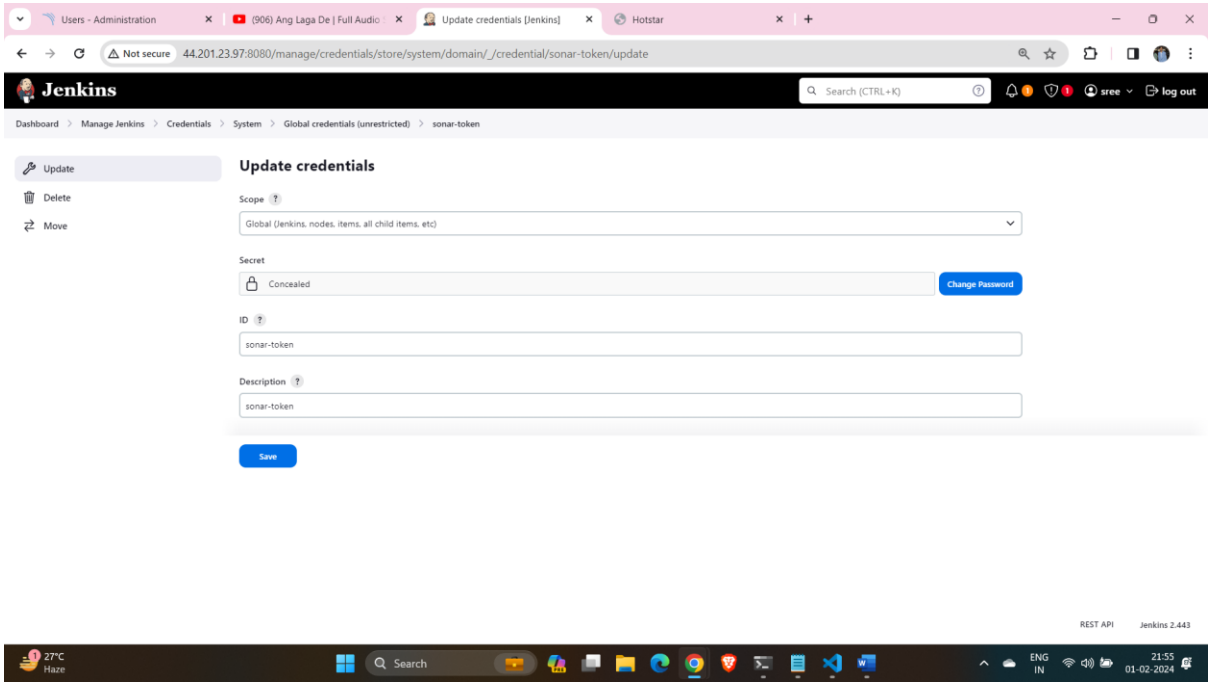
5. Click apply and save.

Configure Sonar Server in Manage Jenkins

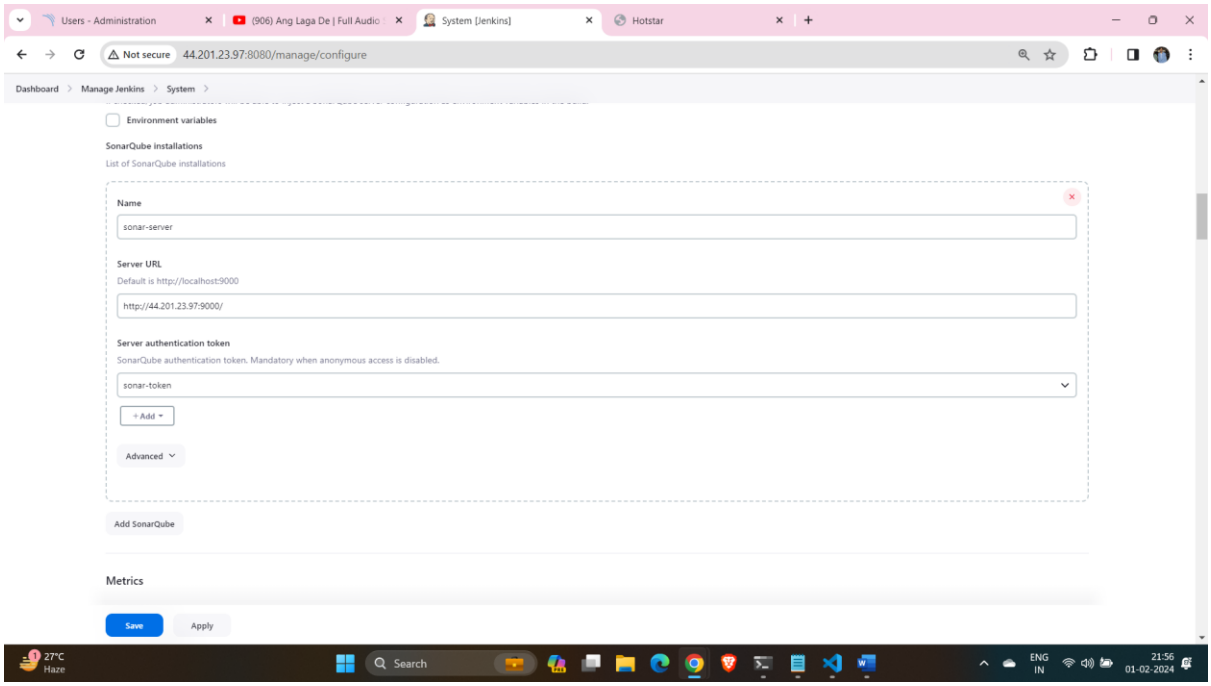
1. Grab the Public IP Address of your EC2 Instance, Sonarqube works on Port 9000, so <Public IP>:9000. Goto your Sonarqube Server.
    - Click on Administration → Security → Users → Click on Tokens and Update Token → Give it a name
    - click on Generate Token
    - Click on update Token
    - Create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



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

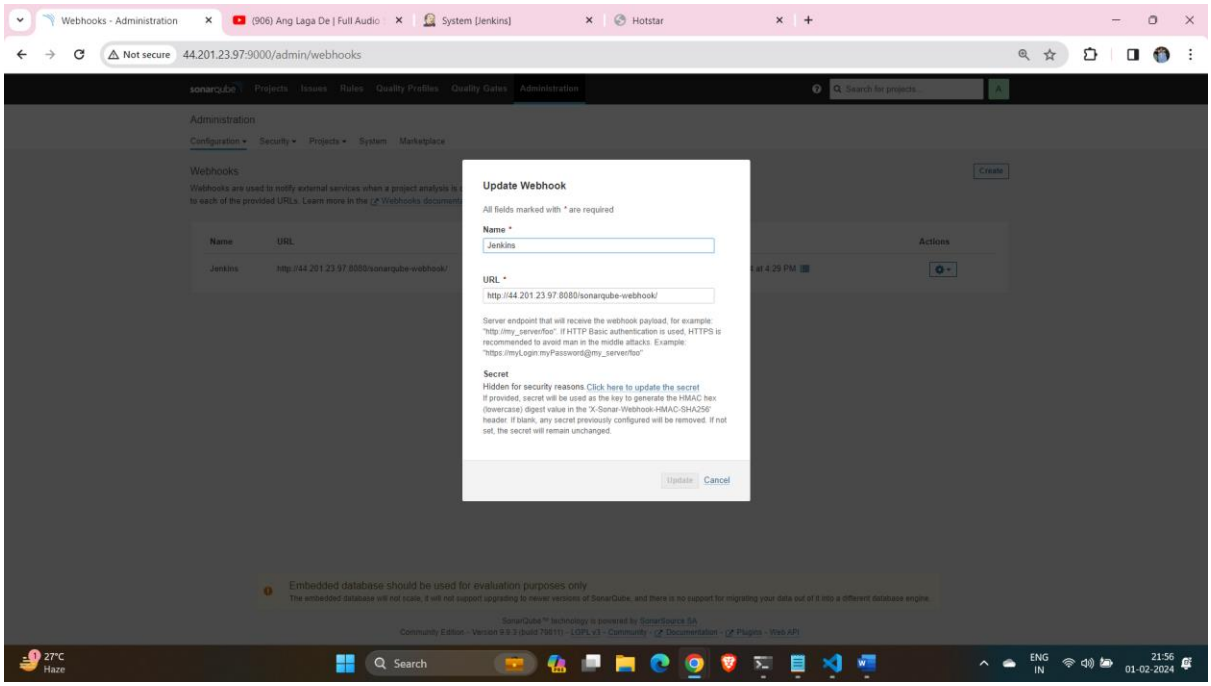


3. Now, go to Dashboard → Manage Jenkins → System and Add like the below image.

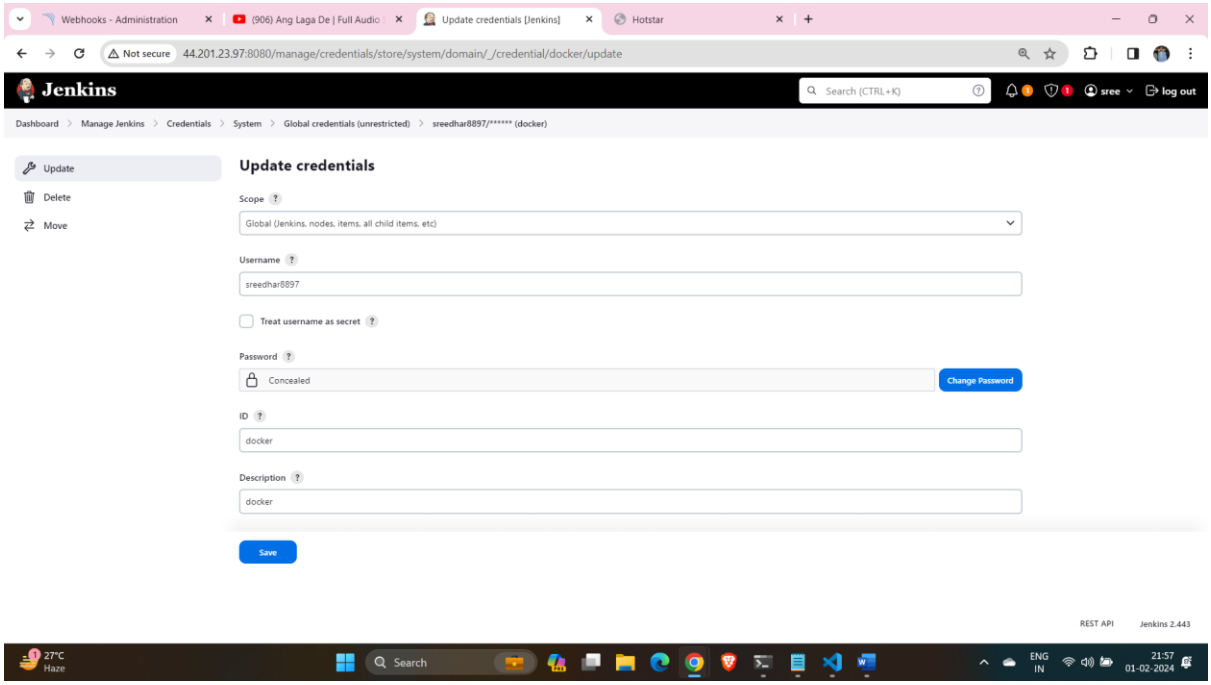


4. Click on Apply and Save
5. In the Sonarqube Dashboard add a quality gate also
    - Administration–> Configuration–>Webhooks
    - Click on Create
    - Add details
      #in url section of quality gate

<http://jenkins-public-ip:8080>/sonarqube-webhook/>

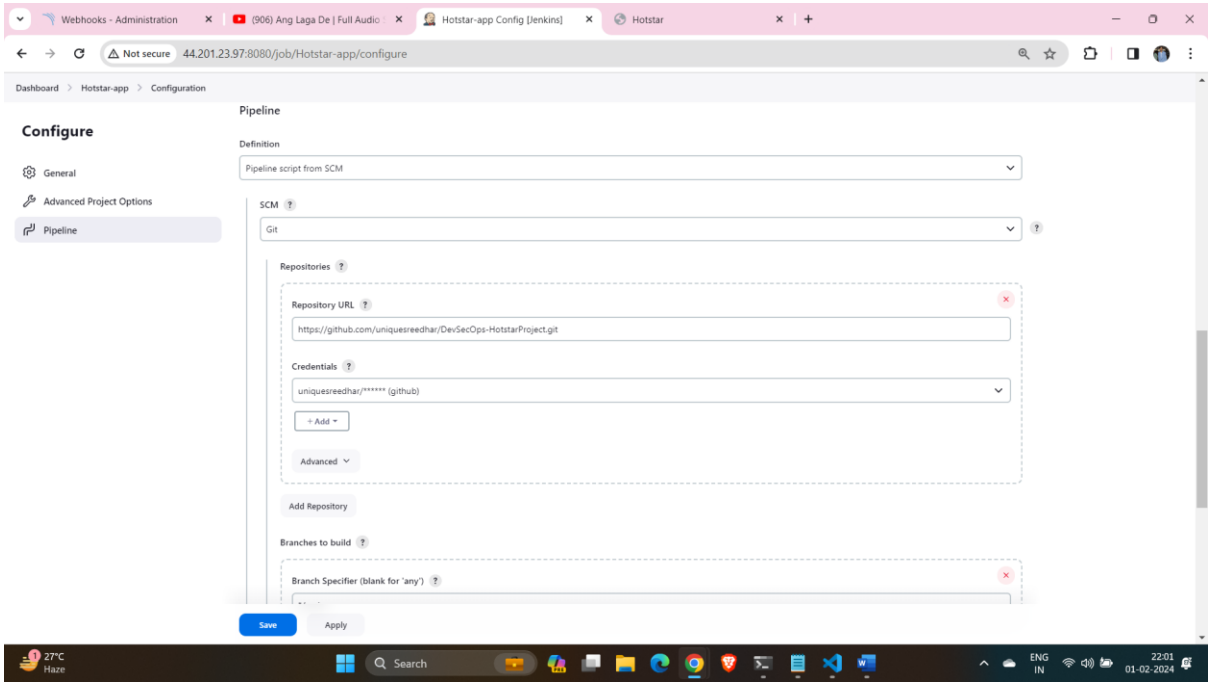


6. Now add Docker credentials to the Jenkins to log in and push the image
    - Manage Jenkins –> Credentials –> global –> add credentials
    - Add DockerHub Username and Password under Global Credentials
    - Create.



7. Before Adding pipeline install Docker Scout
    - docker login #use credentials to login
    - curl -sSfL https://raw.githubusercontent.com/docker/scout-cli/main/install.sh | sh -s -- -b /usr/local/bin

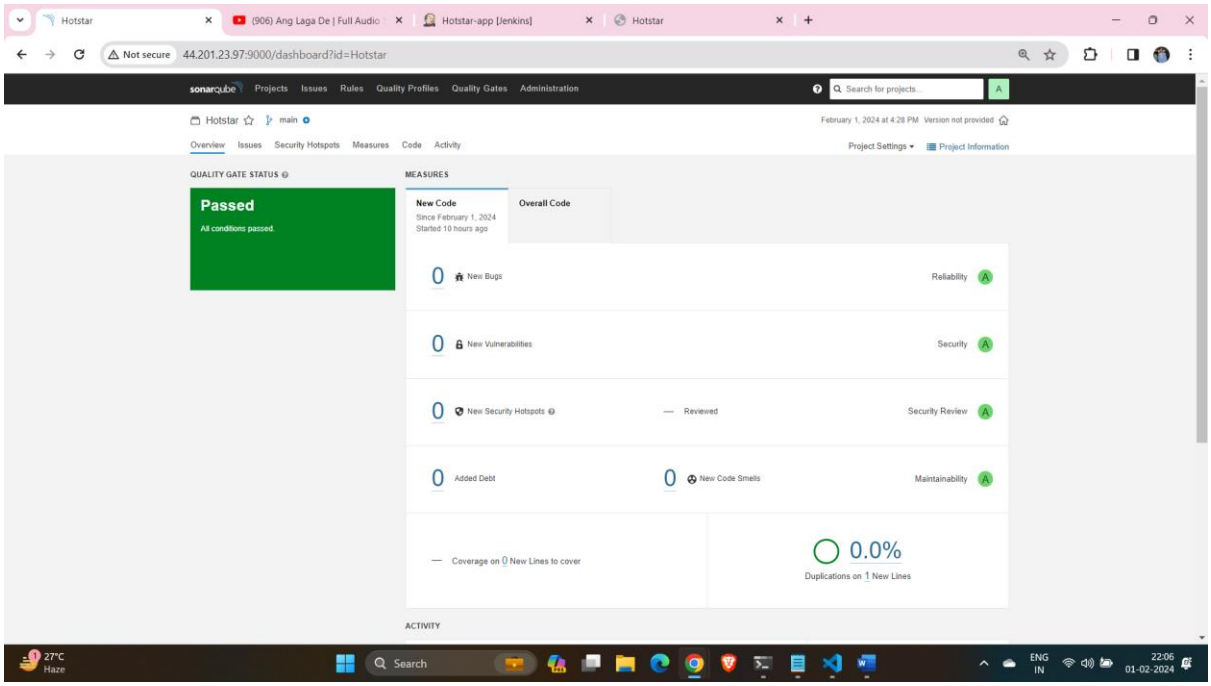
8. Create a Pipeline with a name and in the pipeline, section give the github url and the Jenkinsfile location with the github credentials.



9. Click on Apply and save and then Build now
10. Stage view

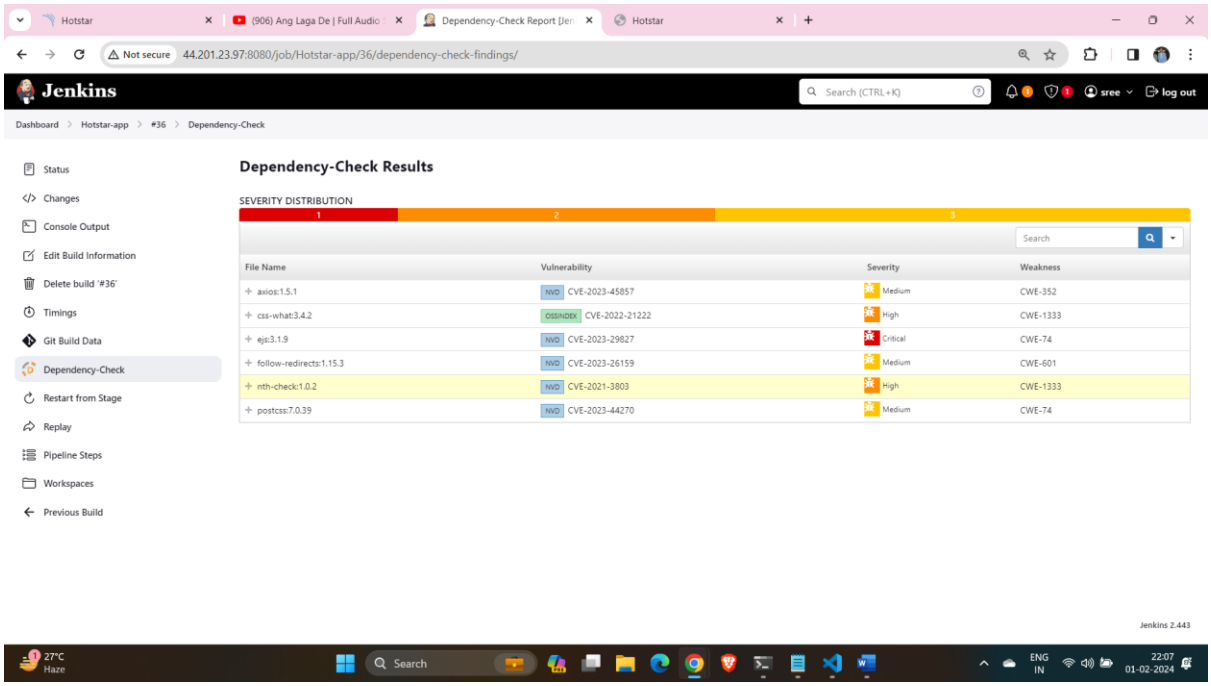
	Declarative: Checkout SCM	Declarative: Tool Install	clean workspace	Checkout from Git	Sonarqube Analysis	quality gate	Install Dependencies	OWASP FS SCAN	Docker Scout FS	Docker Build & Push	Docker Scout Image	deploy_docker
Average stage times: (Average full run time: ~4min 54s)	284ms	176ms	346ms	806ms	19s	405ms	12s	56s	5s	32s	31s	425ms
#36 Feb 01 16:28 4 commits	321ms	191ms	310ms	807ms	19s	377ms	15s	1min 53s	10s	1min 5s	1min 3s	777ms

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

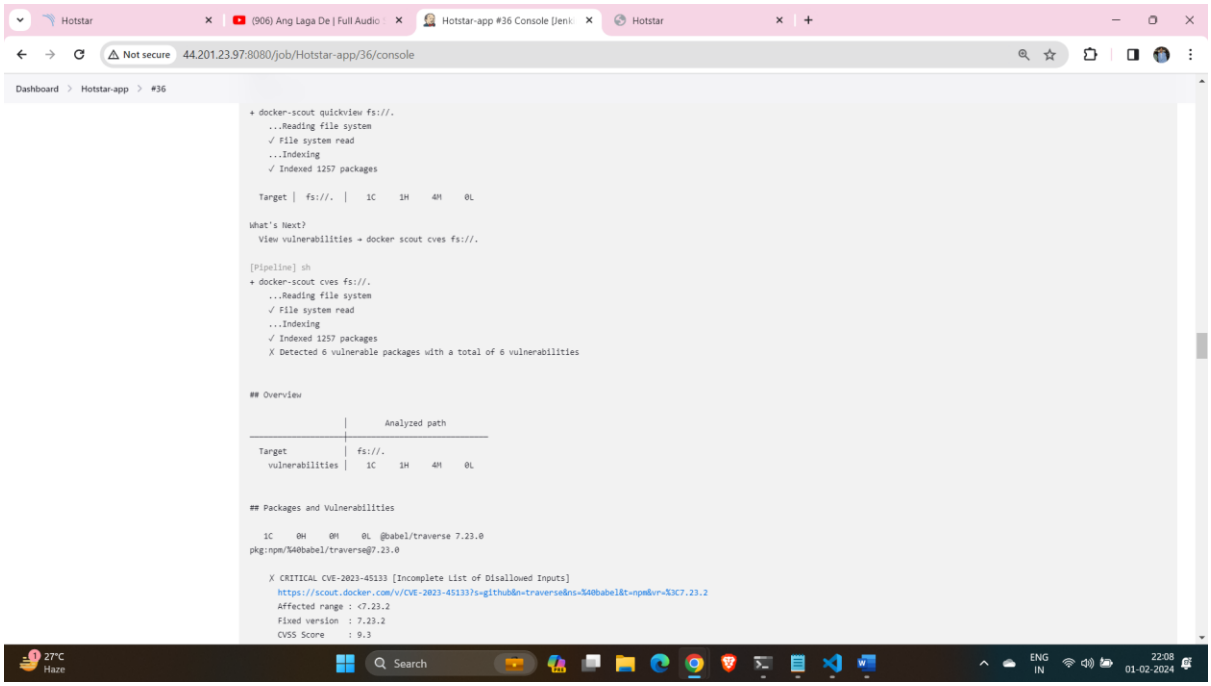


You can see the report has been generated and the status shows as passed. You can see that there are 854 lines it scanned. To see a detailed report, you can go to issues.

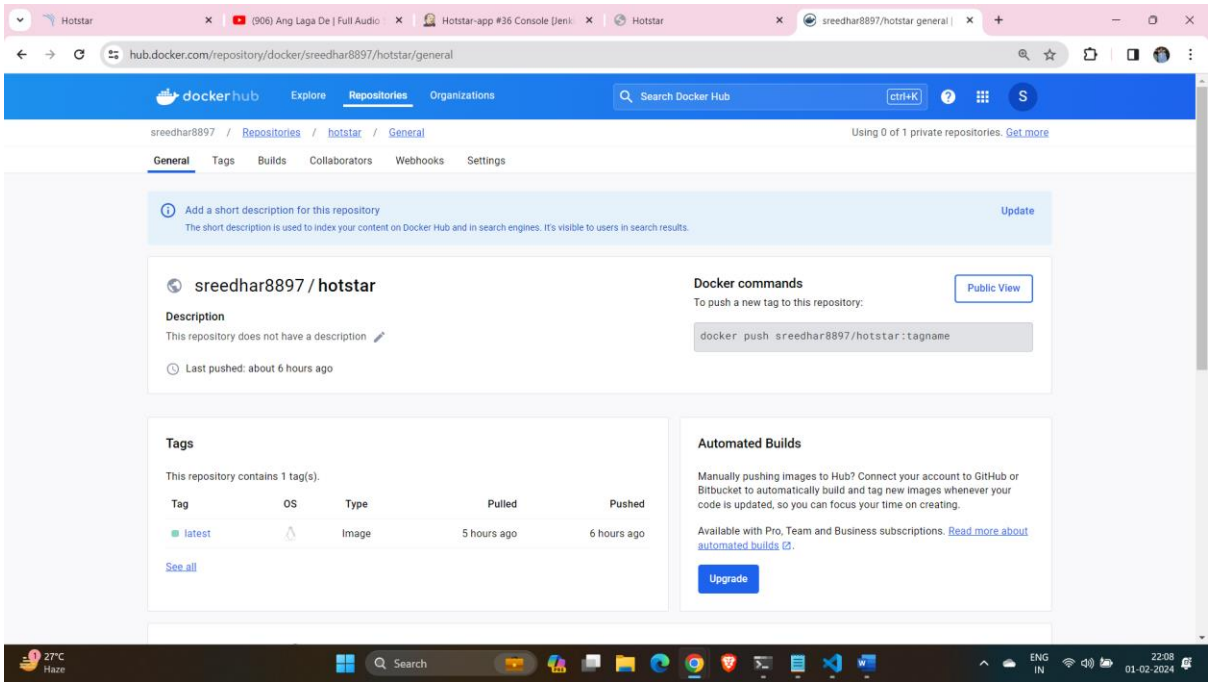
12. OWASP, you will see that in status, a graph will also be generated and Vulnerabilities.



13. Let’s See Docker Scout File scan report

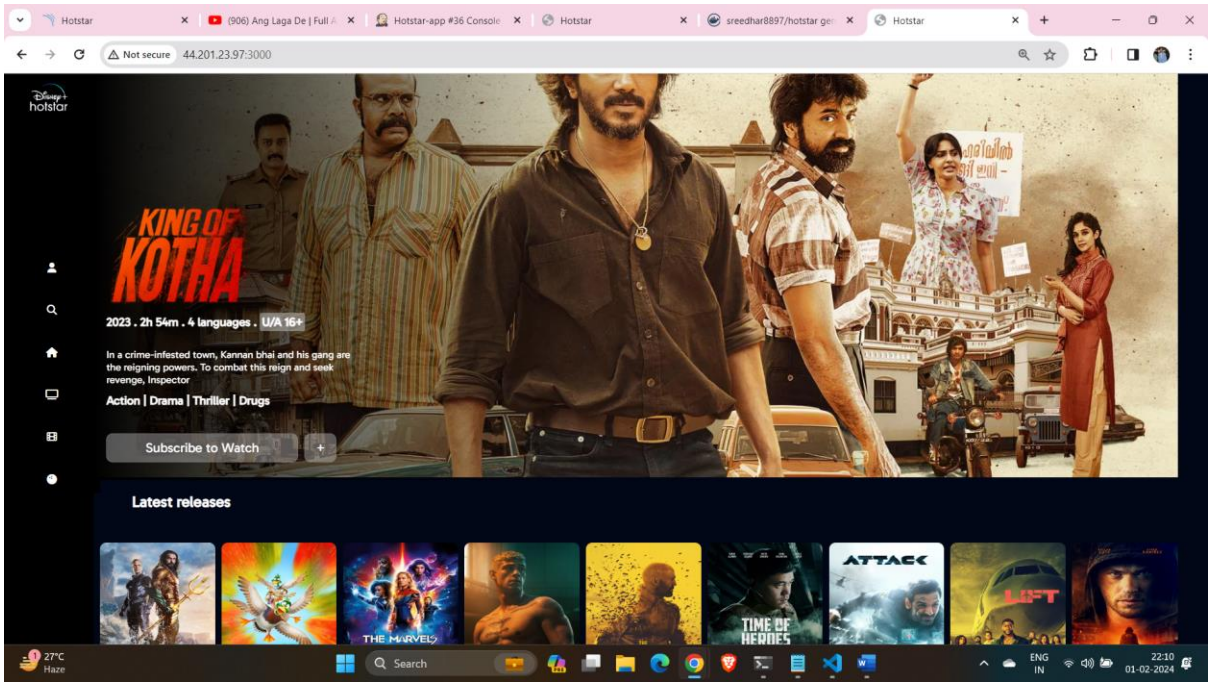


14. When you log in to Dockerhub, you will see a new image is created.

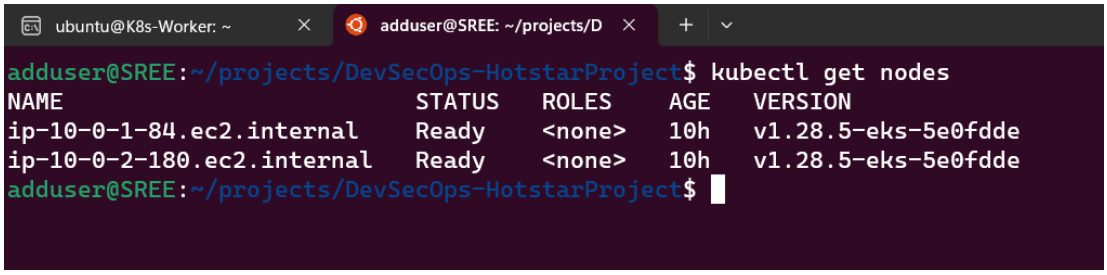


15. Deploy to Container
<ec2-ip:3000>

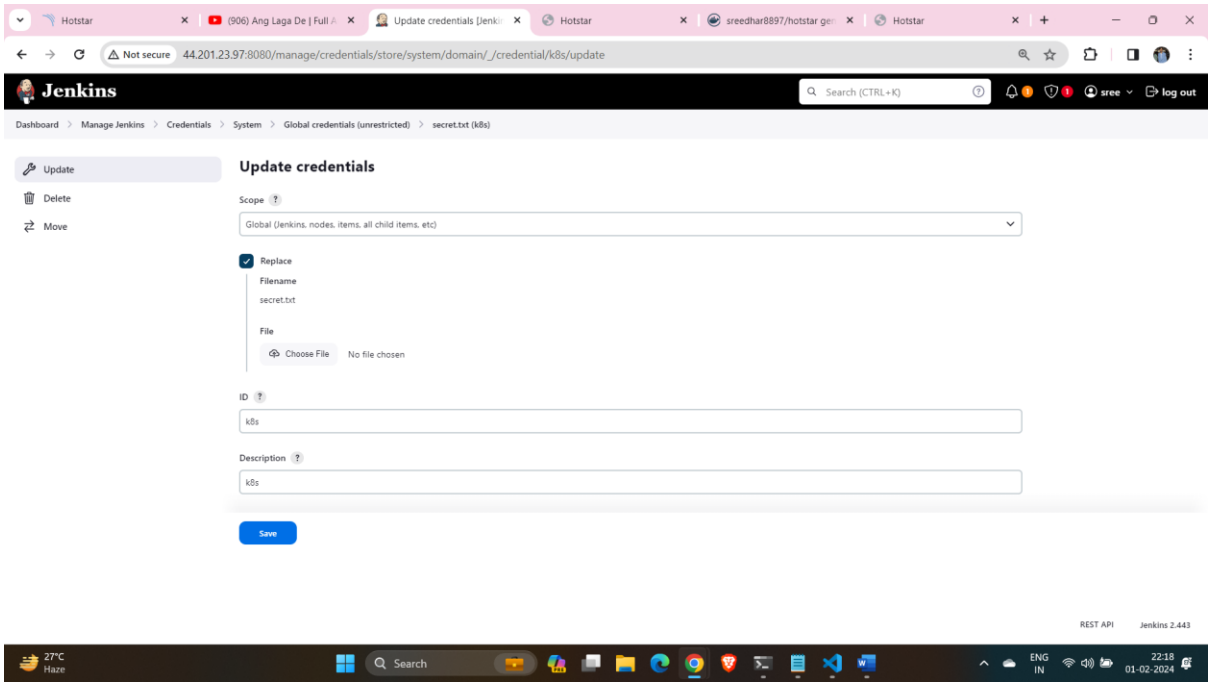
Then the page looks like:



16. Go to Putty of your Jenkins instance SSH and enter the below command
    aws eks update-kubeconfig --name <CLUSTER NAME> --region <CLUSTER REGION>
17. Let's see the nodes
    kubectl get nodes

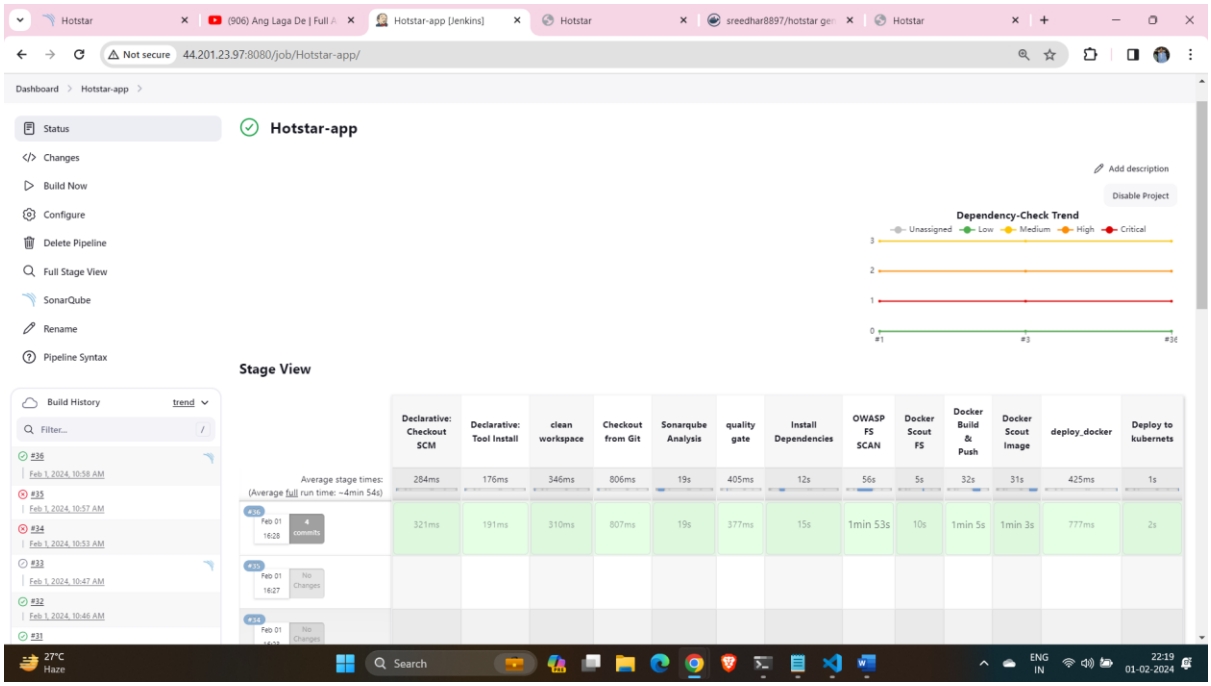


18. Now Give this command in CLI
    cat /root/. kube/config
19. Copy the config file to Jenkins's master or the local file manager and save it
20. Copy it and save it in documents or another folder save it as secret-file.txt
    Note: create a secret-file.txt in your file explorer save the config in it and use this at the kubernetes credential section.
21. Install Kubernetes Plugin, once it's installed successfully
    goto manage Jenkins –> manage credentials –> Click on Jenkins global –> add credentials



22. Final step to deploy on the Kubernetes cluster

- Add the Deploy stage and run the pipeline again

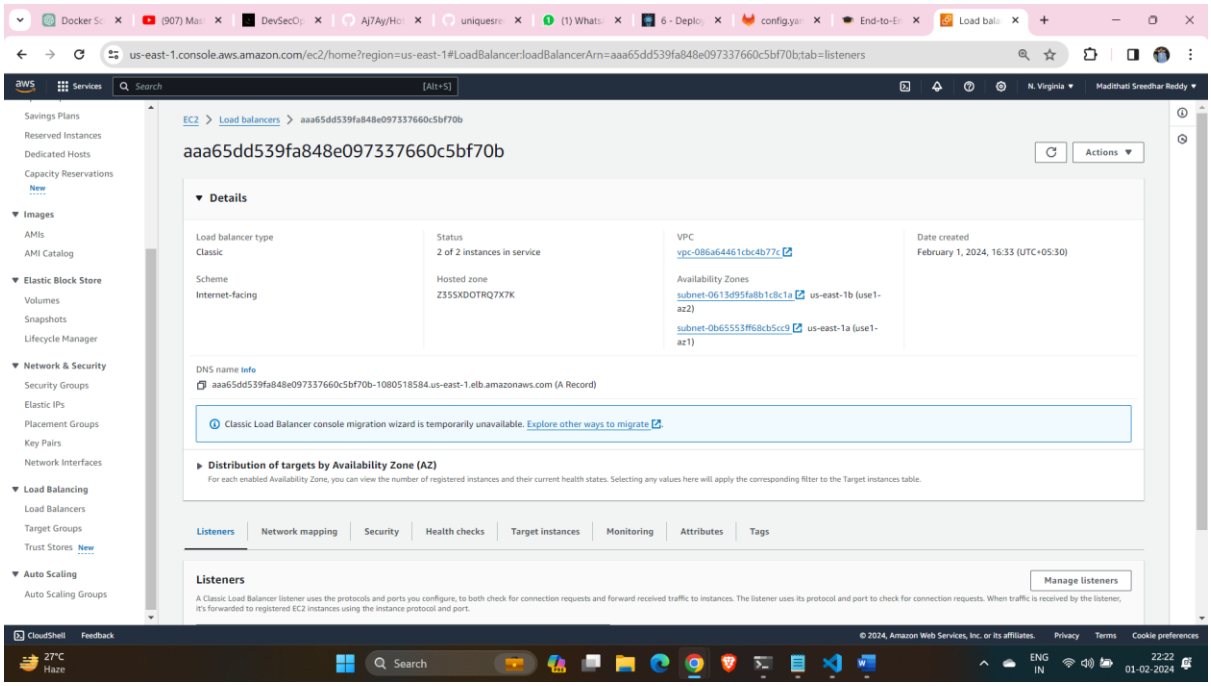


- Give the command after pipeline success

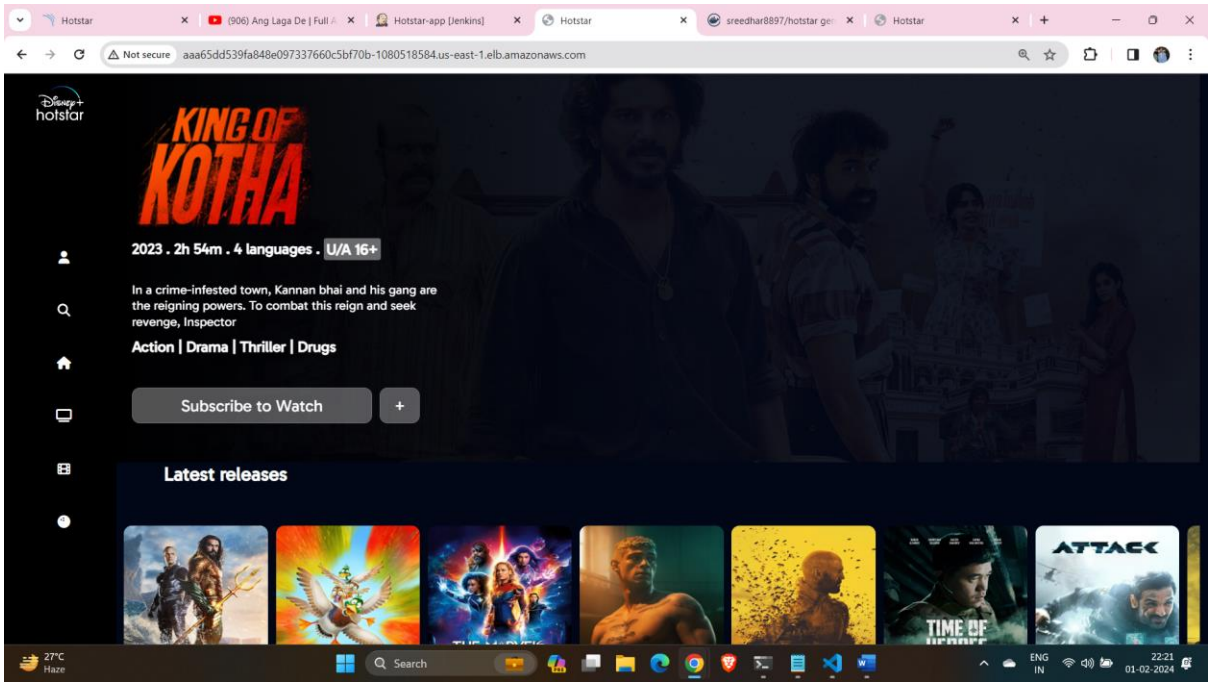
  kubectl get all

```
adduser@SREE:~/projects/DevSecOps-HotstarProject$ kubectl get all
NAME                                    READY   STATUS    RESTARTS   AGE
pod/hotstar-deployment-74f8455478-5wdsp   1/1     Running   0          5h46m

NAME                      TYPE           CLUSTER-IP      EXTERNAL-IP                                                              PORT(S)
  AGE
service/hotstar-service   LoadBalancer   172.20.84.150   aaa65dd539fa848e097337660c5bf70b-1080518584.us-east-1.elb.amazonaws.com   80:30760/TCP
  5h46m
service/kubernetes        ClusterIP      172.20.0.1      <none>                                                                   443/TCP
  10h

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/hotstar-deployment   1/1     1            1           5h46m

NAME                                          DESIRED   CURRENT   READY   AGE
replicaset.apps/hotstar-deployment-74f8455478   1         1         1       5h46m
adduser@SREE:~/projects/DevSecOps-HotstarProject$
```

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



You will see output like this.



If you get the above page:

Congratulations You have successfully completed the Project. 😊

Step 3: Destruction

- Now Go to Jenkins Dashboard and click on EKSCluster-Terraform- job
  And build with parameters and destroy action
  It will delete the EKS cluster that provisioned
- After 10 minutes cluster will delete and wait for it. Don't remove ec2 instance till that time.
- Delete the Ec2 instance & IAM role.
- Check the load balancer also if it is deleted or not.

Happy Learning!.. 😊