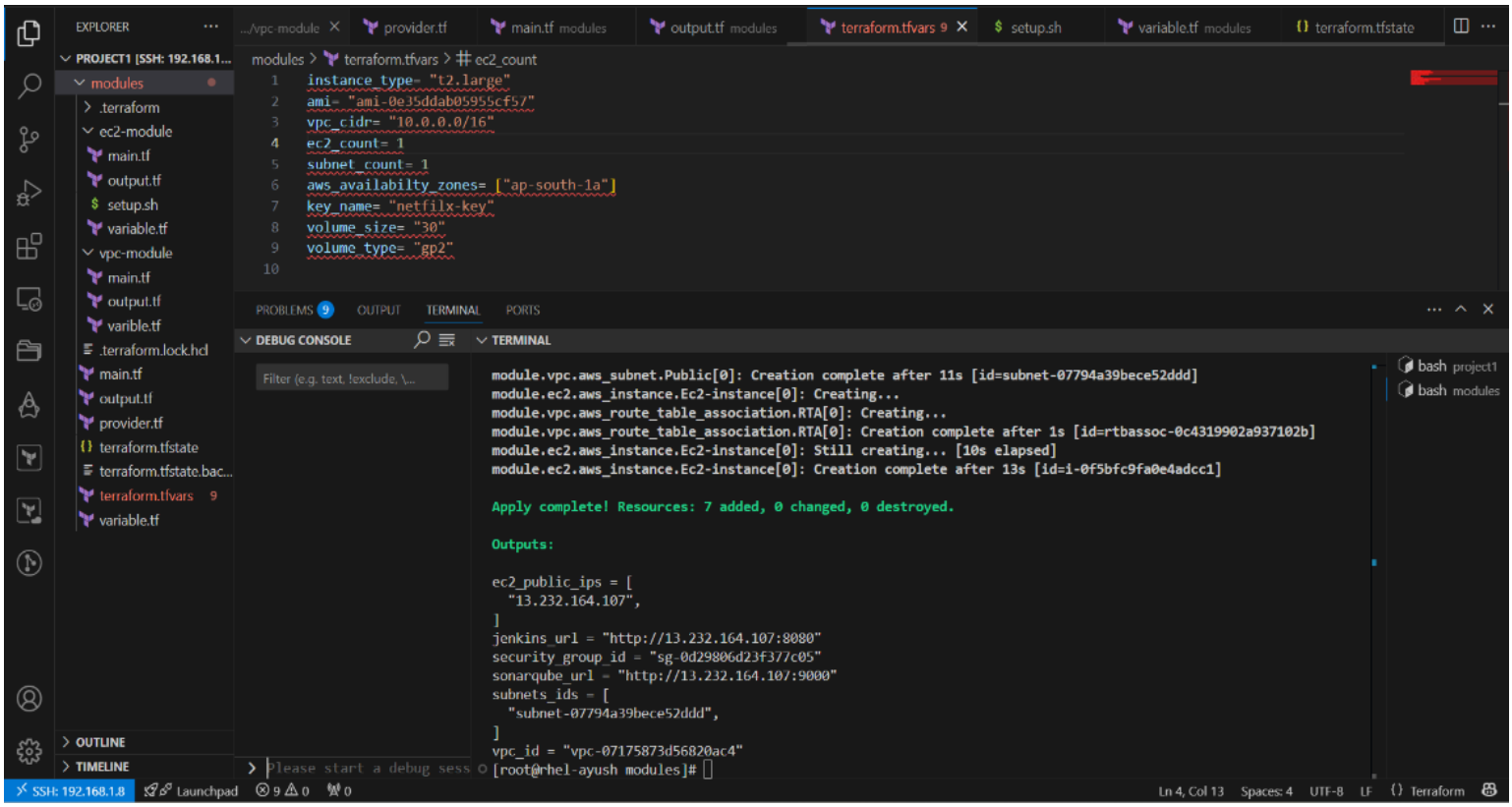
DEPLYOED NETFLIX CLONE APPLICATION ON EKS CLUSTER USING DEVSECOP PRACTISE

Infrastructure Setup using Terraform

- Used **Terraform** to provision complete AWS infrastructure
- Designed a **modular structure**:
  - **VPC Module**: Created VPC, public subnet, route tables, and Internet Gateway
  - **EC2 Module**: Launched EC2 instances with required Security Groups and key pairs
- Used **User Data script** in EC2 module to automate installation of:
  - **Jenkins** (for CI/CD pipeline)
  - **SonarQube** (for code quality analysis)
  - **Trivy** (for vulnerability scanning)
  - **Docker** (to build and push images)
  - **In output we will get the Jenkins url and sonarquebe url as output**



Checkout infra-code from my github repo

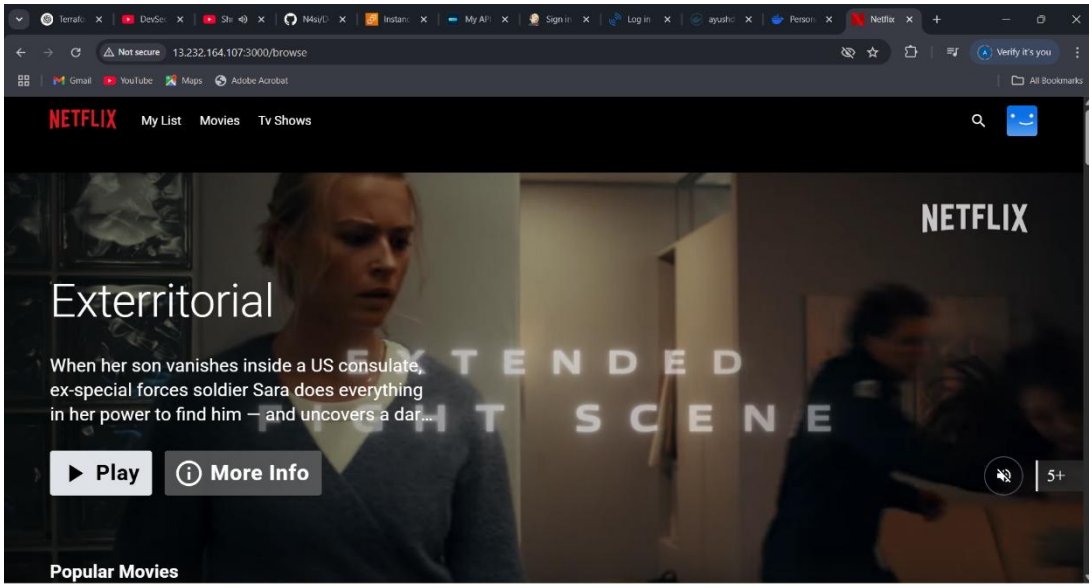
https://github.com/Ayush-bhoyar/Netflix-DevSecOps-Project.git

⍰ Verified the application functionality locally before deployment

⍰ Built the Docker image using docker

```
root@ip-10-0-0-153:/home/ubuntu/DevSecOps-Project# docker ps
CONTAINER ID   IMAGE                 COMMAND                  CREATED          STATUS          PORTS                                          NAMES
70e60084cc4f   2721f5848bb0          "nginx -g 'daemon of…"   4 minutes ago    Up 4 minutes    0.0.0.0:3000->80/tcp, [::]:3000->80/tcp        netflix-con
2e77ab95e69f   sonarqube:community   "/opt/sonarqube/dock…"   25 minutes ago   Up 25 minutes   0.0.0.0:9000->9000/tcp, [::]:9000->9000/tcp    sonarqube
root@ip-10-0-0-153:/home/ubuntu/DevSecOps-Project# docker images
REPOSITORY                    TAG         IMAGE ID       CREATED         SIZE
ayushdocker2607/netflix-clone latest      2721f5848bb0   4 minutes ago   57.5MB
sonarqube                     community   3be54dbc7ac1   2 weeks ago     1.23GB
root@ip-10-0-0-153:/home/ubuntu/DevSecOps-Project#
```

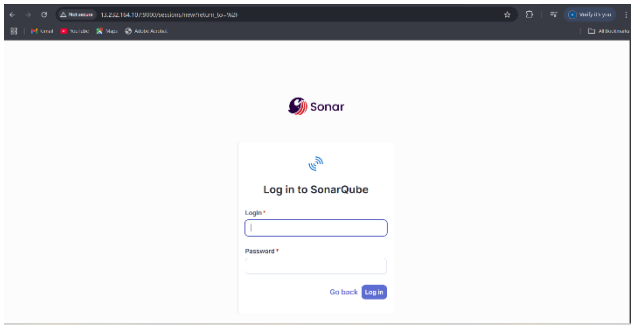
Checked the application is working locally using docker run.

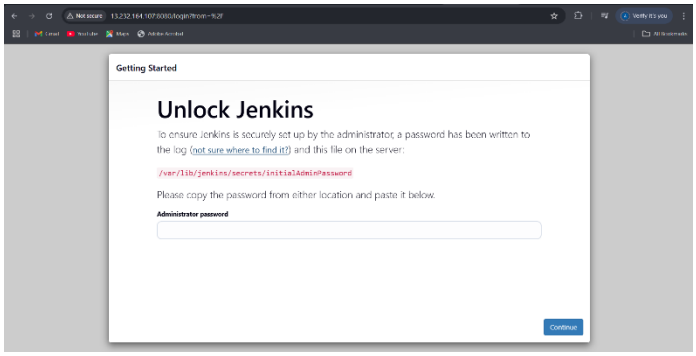


Jenkins Setup & Configuration

- **Configured Jenkins on an EC2 instance using a shell script via Terraform user data.**
- **Installed and configured the following Jenkins tools:**
  - **JDK 17**
  - **Node.js 16**
  - **SonarQube Scanner**
  - **Docker**

- **Installed essential Jenkins plugins:**
  - **Pipeline**
  - **Docker**
  - **SonarQube Scanner**
  - **OWASP Dependency-Check**
  - **Trivy**
- **Added required credentials in Jenkins:**
  - **Docker Hub credentials (Docker-cred)**
  - **GitHub username and personal access token (github-cred)**
  - **SonarQube token (sonar-token)**
- **Configured Jenkins with:**
  - **SonarQube server URL**
  - **SonarQube Scanner tool path**

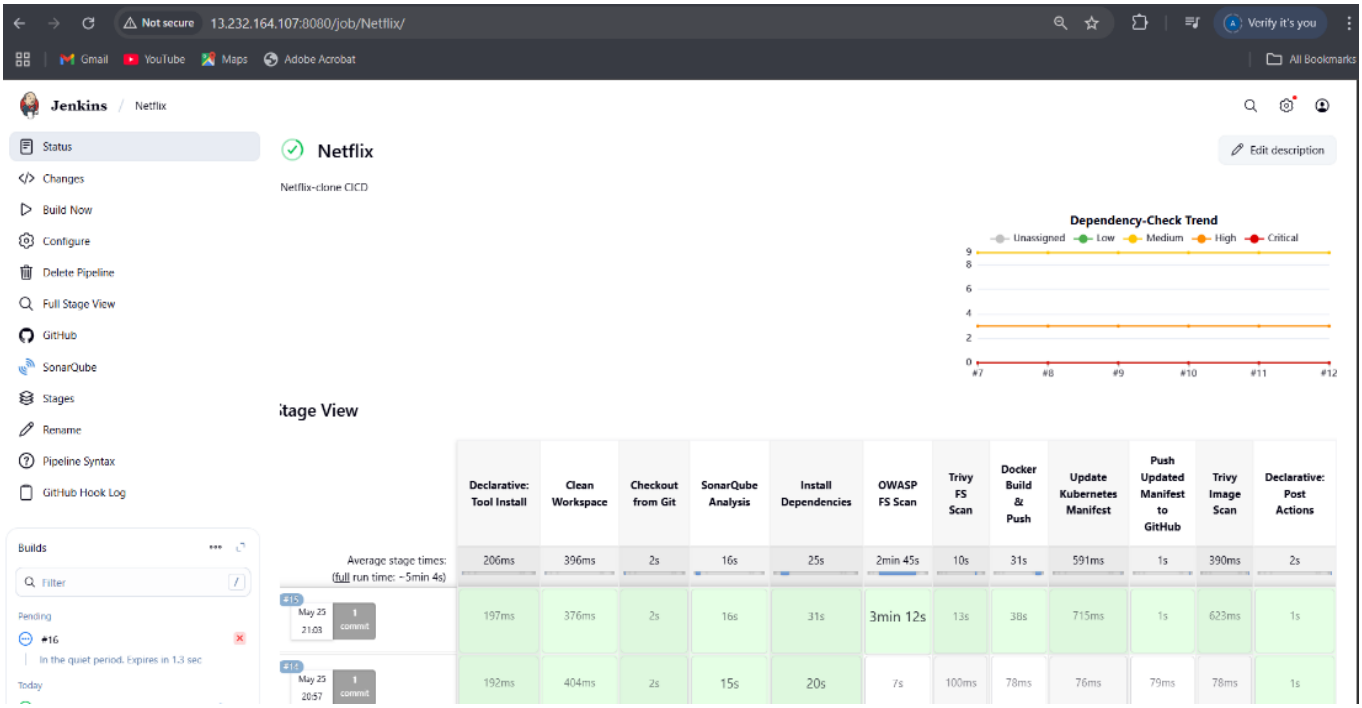




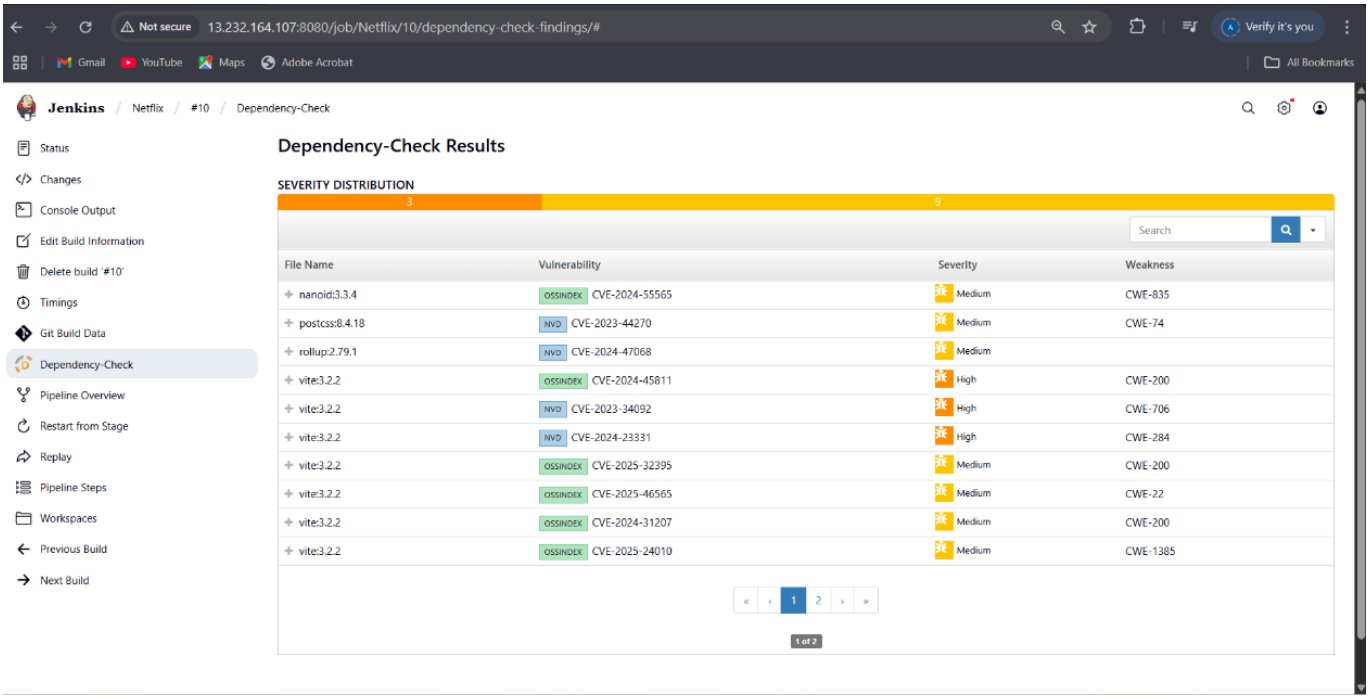
CI/CD Pipeline with Jenkins

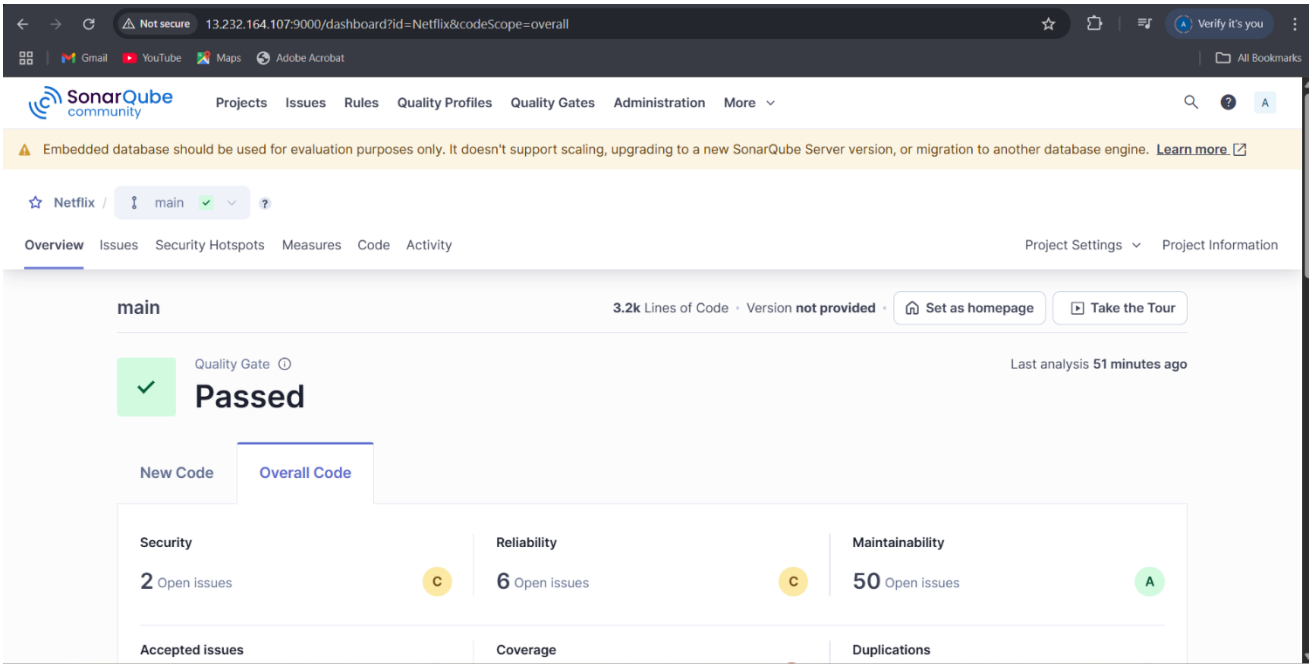
- **Created a Jenkins Pipeline (Jenkinsfile) to automate the full CI/CD process.**
- **The pipeline includes the following stages:**
  1. **Checkout Code – Pulls the latest code from GitHub.**
  2. **Code Quality Check – Runs SonarQube analysis.**
  3. **Build & Push Docker Image – Builds the Docker image and pushes it to Docker Hub.**
  4. **Update Kubernetes Manifest – Automatically updates the image tag in the deployment YAML.**
  5. **Push to GitHub – Commits the updated manifest back to the GitHub repo.**
  6. **Trivy Image Scan – Scans the Docker image for vulnerabilities using Trivy.**

7. **Configured GitHub Webhook to trigger the pipeline automatically whenever a new**
8. **push is made to the repository (eliminating the need to click "Build Now" manually).**



- **Used Jenkins credentials and environment variables to securely handle secrets and dynamic values**





Docker images with Jenkins build tag every time new build the image will get that tag

```
REPOSITORY                TAG         IMAGE ID       CREATED        SIZE
netflix                   latest      c3f626f49c5f   5 hours ago    57.5MB
ayushdocker2607/netflix   11          c3f626f49c5f   5 hours ago    57.5MB
ayushdocker2607/netflix   12          c3f626f49c5f   5 hours ago    57.5MB
ayushdocker2607/netflix   13          c3f626f49c5f   5 hours ago    57.5MB
ayushdocker2607/netflix   latest      c3f626f49c5f   5 hours ago    57.5MB
sonarqube                 community   3be54dbc7ac1   2 weeks ago    1.23GB
root@ip-10-0-0-153:/home/ubuntu/DevSecOps-Project#
```

pushing to docker hub

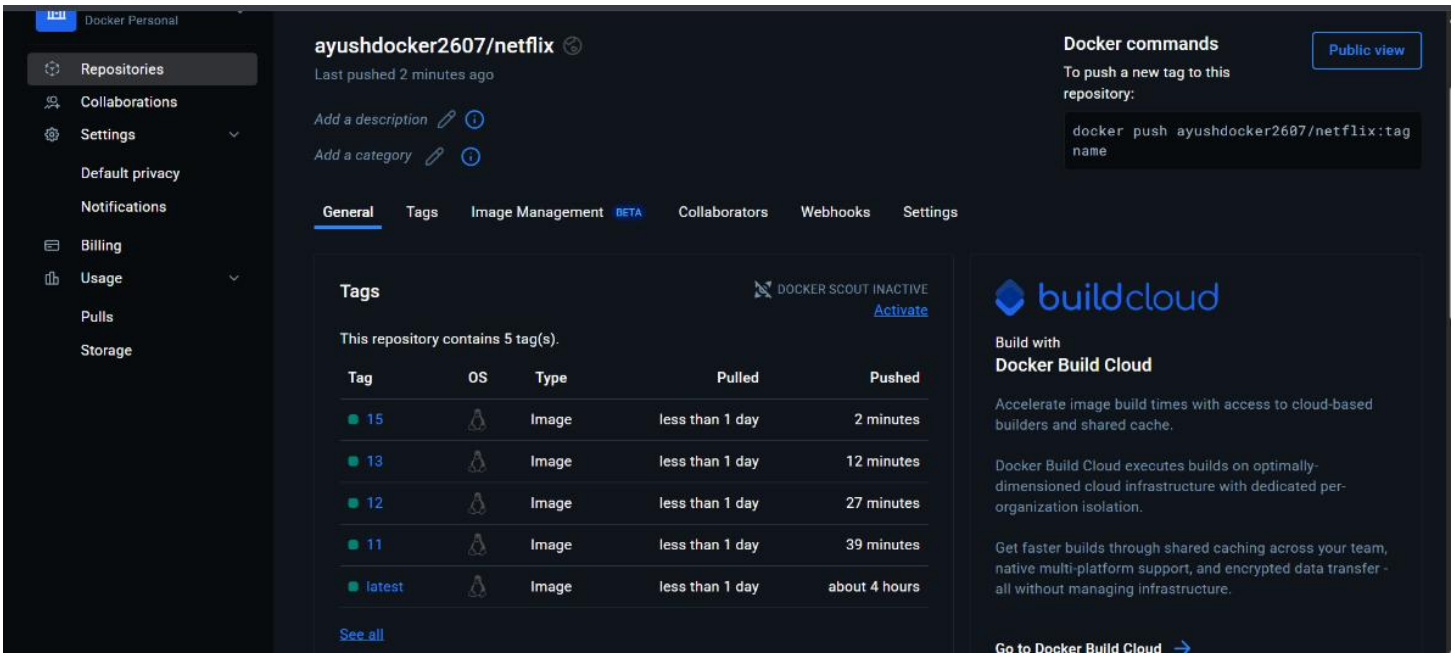
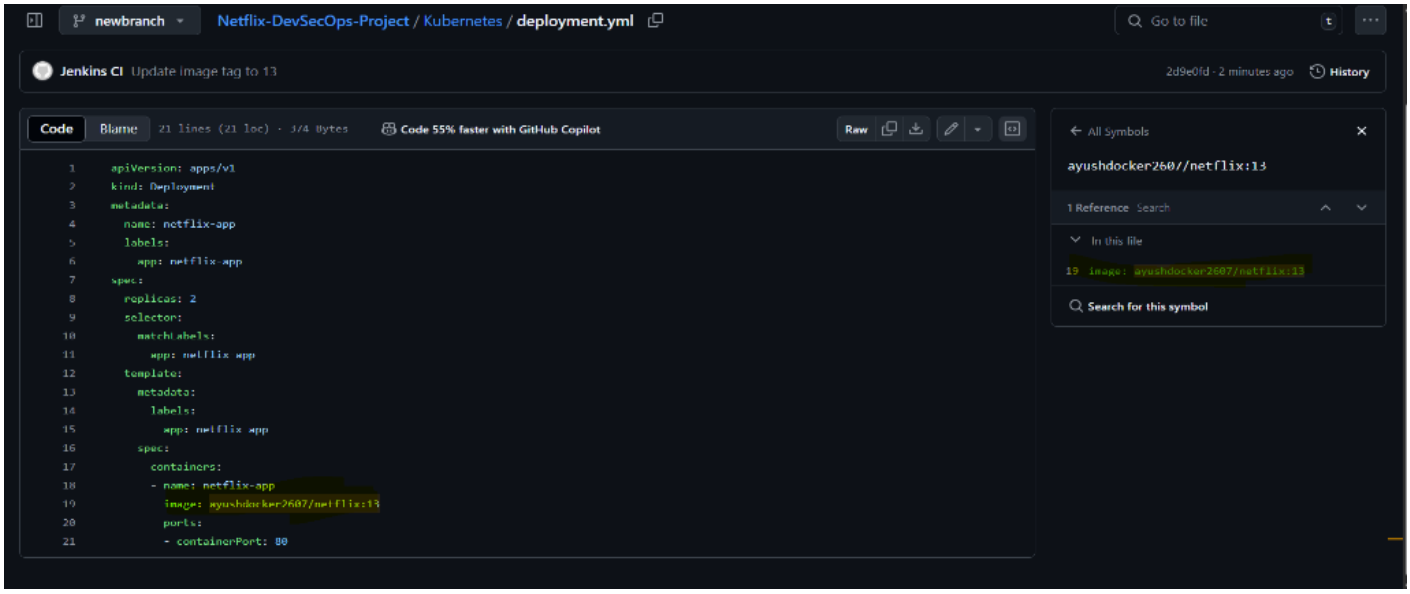
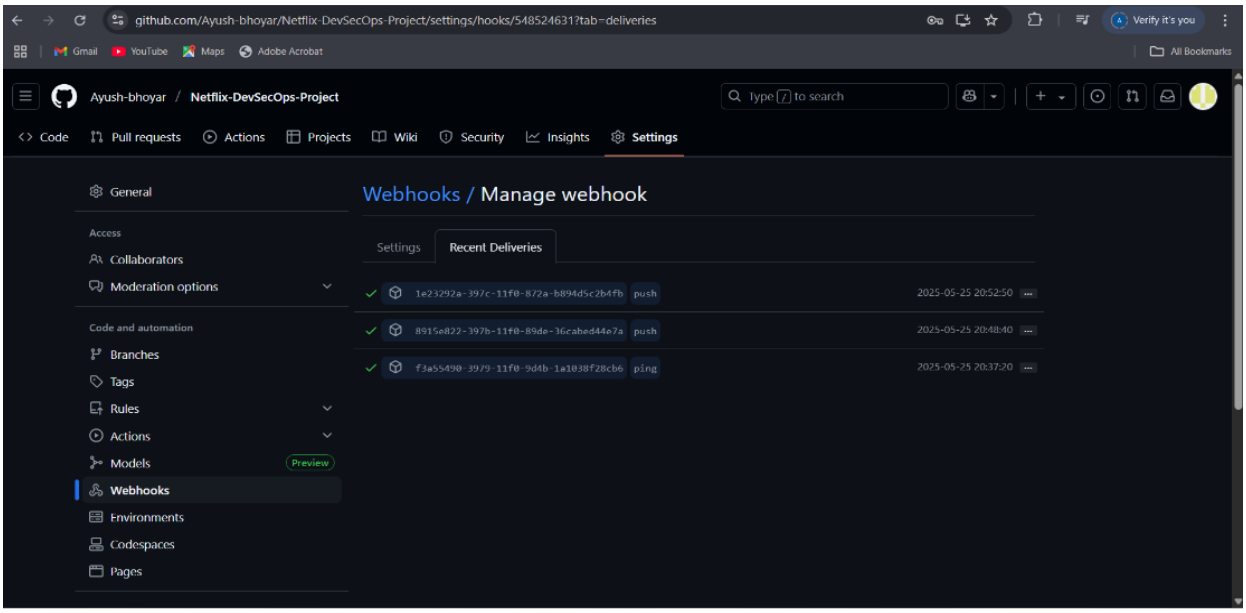


Image tag getting updated in manifest files in GIT repo automatically



For triggering the CICD pipeline I used webhook trigger



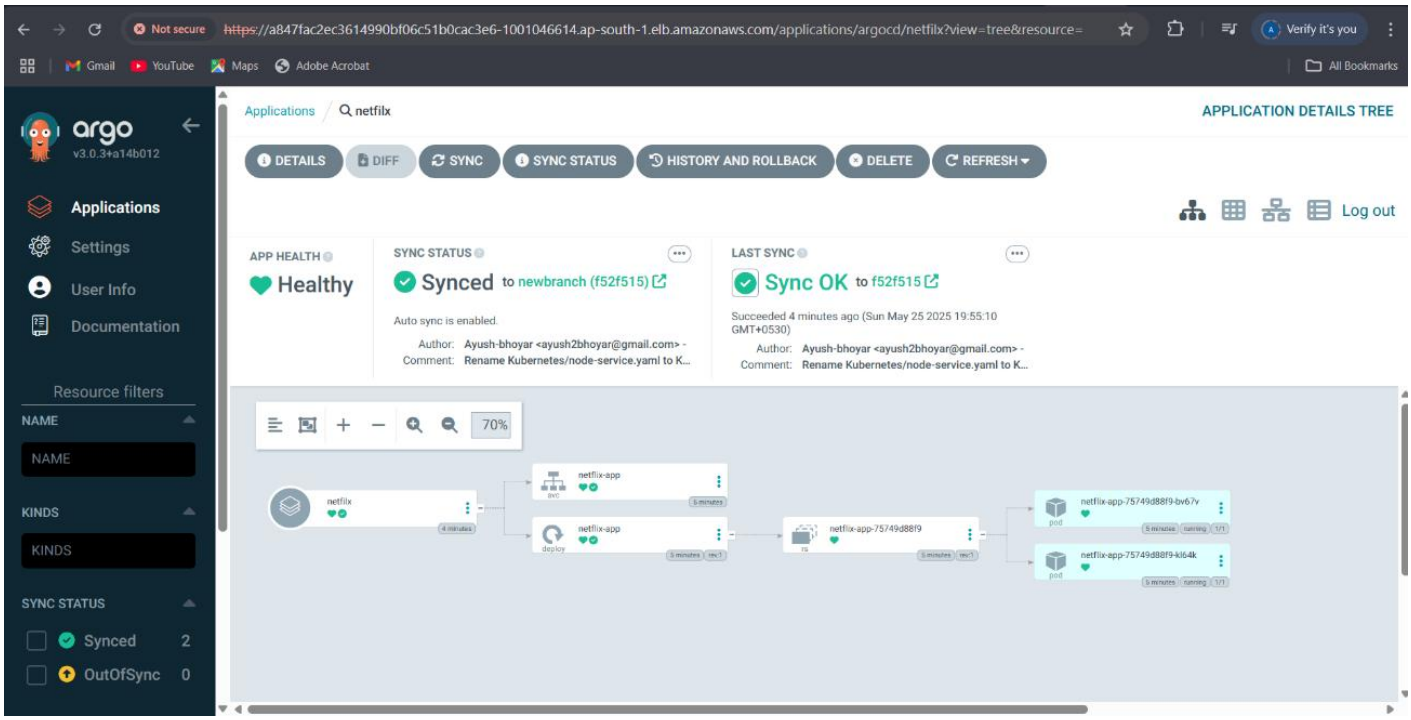
EKS Cluster Setup using Terraform (Manually)

- **Provisioned an EKS (Elastic Kubernetes Service) Cluster on AWS using Terraform.**
- **Created a dedicated IAM role with appropriate permissions for EKS nodes.**
- **Defined required resources:**
  - **EKS Cluster**
  - **Node Group (using t3.medium instances)**
  - **IAM Roles and Policies**
  - **Security Groups and Networking**
- **I used the Ec2 instance (t2.micro) to setup the eks cluster and installed the awscli, kubectl and eksctl .**
- **configured kubeconfig to connect to the EKS cluster from the Bastion**

```
2025-05-25 12:38:48 [!]  1 error(s) occurred and cluster hasn't been created properly, you may wish to check CloudFormation console
2025-05-25 12:38:48 [i]  to cleanup resources, run 'eksctl delete cluster --region=ap-south-1 --name=netflix-cluster'
2025-05-25 12:38:48 [✖]  ResourceNotReady: failed waiting for successful resource state
Error: failed to create cluster "netflix-cluster"
root@ip-172-31-14-66:/home/ubuntu# eksctl create cluster   --name netflix-cluster   --region ap-south-1   --nodegroup-name netflix-nodes   --node-type t2.medium   --nodes 2   --managed
2025-05-25 12:41:55 [i]  eksctl version 0.208.0
2025-05-25 12:41:55 [i]  using region ap-south-1
2025-05-25 12:41:55 [!]  Amazon EKS will no longer publish EKS-optimized Amazon Linux 2 (AL2) AMIs after November 26th, 2025. Additionally, Kubernetes version 1.32 is the last version for w
hich Amazon EKS will release AL2 AMIs. From version 1.33 onwards, Amazon EKS will continue to release AL2023 and Bottlerocket based AMIs. The default AMI family when creating clusters and n
odegroups in Eksctl will be changed to AL2023 in the future.
2025-05-25 12:41:55 [i]  skipping ap-south-1c from selection because it doesn't support the following instance type(s): t2.medium
2025-05-25 12:41:55 [i]  setting availability zones to [ap-south-1a ap-south-1b]
2025-05-25 12:41:55 [i]  subnets for ap-south-1a - public:192.168.0.0/19 private:192.168.64.0/19
2025-05-25 12:41:55 [i]  subnets for ap-south-1b - public:192.168.32.0/19 private:192.168.96.0/19
2025-05-25 12:41:55 [i]  nodegroup "netflix-nodes" will use "" [AmazonLinux2/1.32]
2025-05-25 12:41:55 [i]  using Kubernetes version 1.32
2025-05-25 12:41:55 [i]  creating EKS cluster "netflix-cluster" in "ap-south-1" region with managed nodes
2025-05-25 12:41:55 [i]  will create 2 separate CloudFormation stacks for cluster itself and the initial managed nodegroup
2025-05-25 12:41:55 [i]  if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=ap-south-1 --cluster=netflix-cluster'
2025-05-25 12:41:55 [i]  Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "netflix-cluster" in "ap-south-1"
2025-05-25 12:41:55 [i]  CloudWatch logging will not be enabled for cluster "netflix-cluster" in "ap-south-1"
2025-05-25 12:41:55 [i]  you can enable it with 'eksctl utils update-cluster-logging --enable-types={SPECIFY-YOUR-LOG-TYPES-HERE (e.g. all)} --region=ap-south-1 --cluster=netflix-cluster'
2025-05-25 12:41:55 [i]  default addons coredns, metrics-server, vpc-cni, kube-proxy were not specified, will install them as EKS addons
2025-05-25 12:41:55 [i]
2 sequential tasks: { create cluster control plane "netflix-cluster",
    2 sequential sub-tasks: {
        2 sequential sub-tasks: {
            1 task: { create addons },
            wait for control plane to become ready,
        },
        create managed nodegroup "netflix-nodes",
    }
}
2025-05-25 12:41:55 [i]  building cluster stack "eksctl-netflix-cluster-cluster"
2025-05-25 12:41:55 [i]  deploying stack "eksctl-netflix-cluster-cluster"
2025-05-25 12:42:25 [i]  waiting for CloudFormation stack "eksctl-netflix-cluster-cluster"
2025-05-25 12:42:55 [i]  waiting for CloudFormation stack "eksctl-netflix-cluster-cluster"
2025-05-25 12:43:55 [i]  waiting for CloudFormation stack "eksctl-netflix-cluster-cluster"
2025-05-25 12:44:55 [i]  waiting for CloudFormation stack "eksctl-netflix-cluster-cluster"
2025-05-25 12:45:55 [i]  waiting for CloudFormation stack "eksctl-netflix-cluster-cluster"
2025-05-25 12:46:55 [i]  waiting for CloudFormation stack "eksctl-netflix-cluster-cluster"
```

ArgoCD Installation & GitOps Deployment

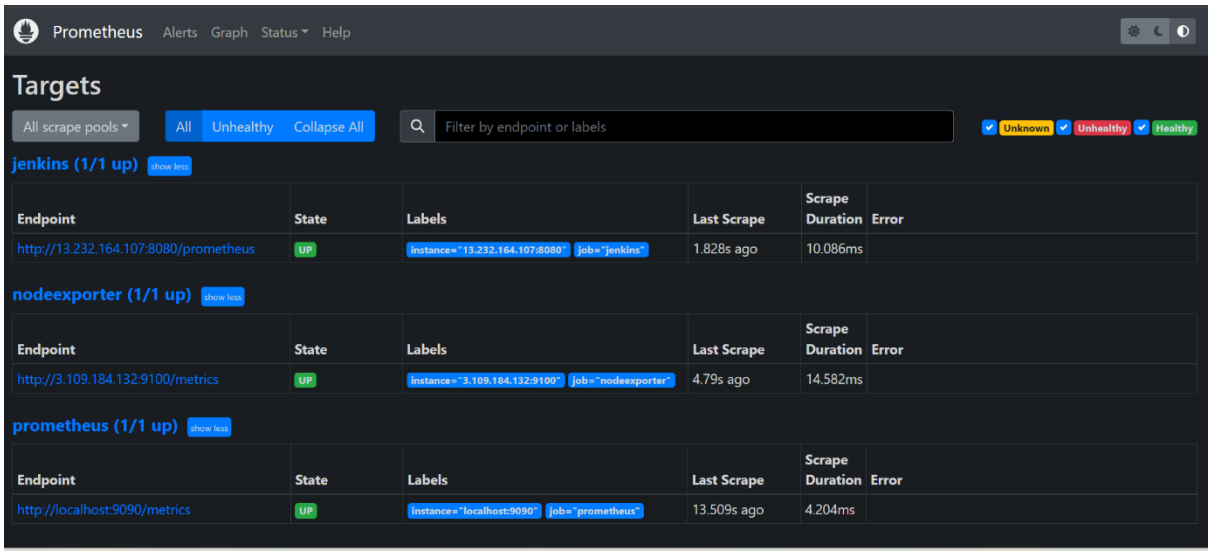
- **Installed ArgoCD on the Kubernetes cluster using kubectl apply for manifests.**
- **Exposed the ArgoCD UI using a LoadBalancer**
- **Configured ArgoCD to watch the GitHub repository containing Kubernetes manifests.**
- **Created an Application in ArgoCD pointing to the Git repo and target namespace.**
- **Enabled automatic sync in ArgoCD:**
    - **On every commit (e.g., updated image tag), ArgoCD synced the latest state from Git and deployed the app automatically.**

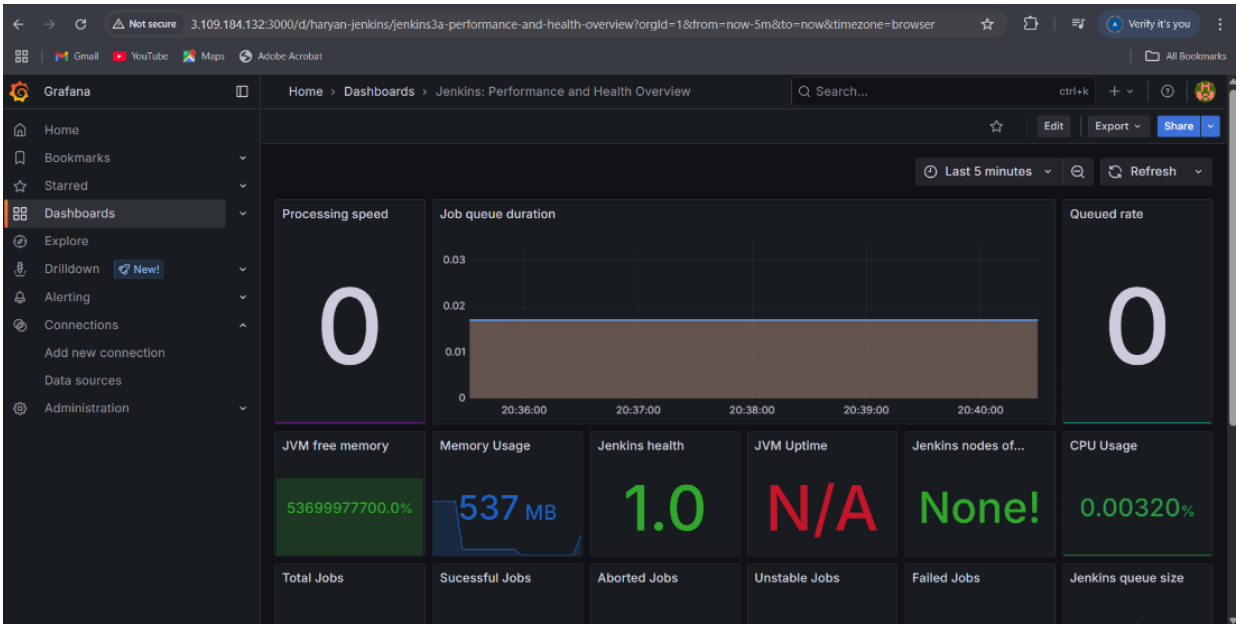


EKS cluster all pods and services also you can see the EKS cluster name

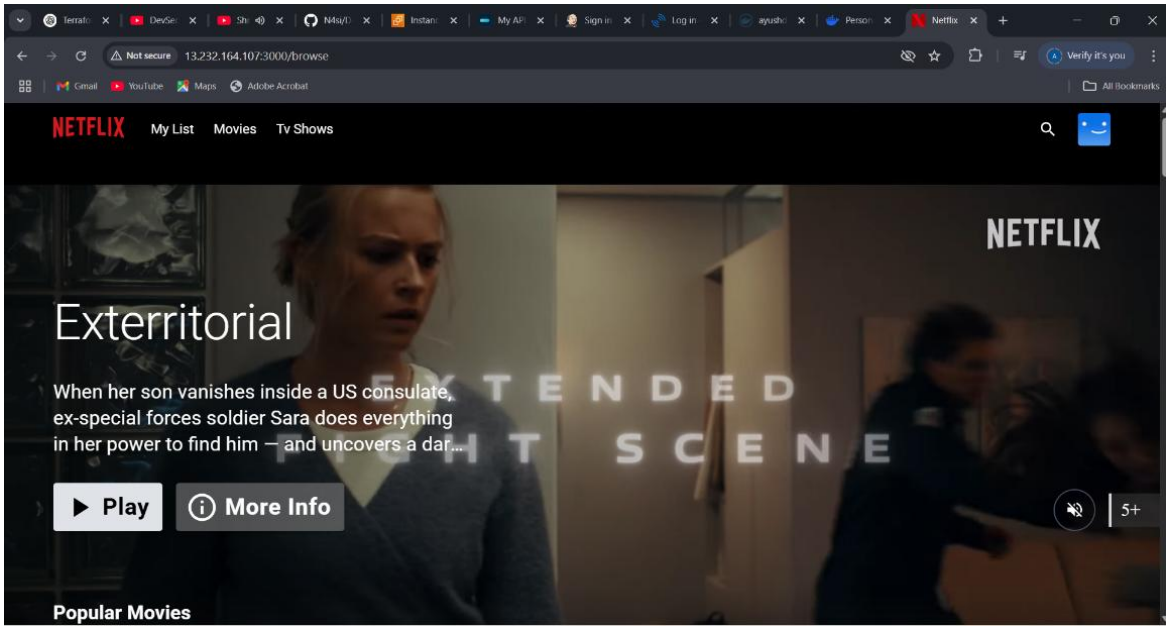
```
argocd-redis                              ClusterIP      10.100.105.214   <none>                                                                    6379/TCP                    12m
argocd-repo-server                        ClusterIP      10.100.187.156   <none>                                                                    8081/TCP,8084/TCP           12m
argocd-server                             LoadBalancer   10.100.4.28      a847fac2ec3614990bf06c51b0cac3e6-1001046614.ap-south-1.elb.amazonaws.com   80:30958/TCP,443:32183/TCP  12m
argocd-server-metrics                     ClusterIP      10.100.117.161   <none>                                                                    8083/TCP                    12m
root@ip-172-31-14-66:/home/ubuntu# ^C
root@ip-172-31-14-66:/home/ubuntu# kubectl get secret argocd-initial-admin-secret -n argocd \
  -o jsonpath="{.data.password}" | base64 -d && echo
YX5zevfbtoDi7q8o
root@ip-172-31-14-66:/home/ubuntu# kubectl get ns
NAME              STATUS   AGE
argocd            Active   33m
default           Active   48m
kube-node-lease   Active   48m
kube-public       Active   48m
kube-system       Active   48m
root@ip-172-31-14-66:/home/ubuntu# kubectl get svc -n argocd
NAME                                      TYPE           CLUSTER-IP       EXTERNAL-IP                                                               PORT(S)                     AGE
argocd-applicationset-controller          ClusterIP      10.100.95.111    <none>                                                                    7000/TCP,8080/TCP           37m
argocd-dex-server                         ClusterIP      10.100.218.27    <none>                                                                    5556/TCP,5557/TCP,5558/TCP  37m
argocd-metrics                            ClusterIP      10.100.26.181    <none>                                                                    8082/TCP                    37m
argocd-notifications-controller-metrics   ClusterIP      10.100.236.88    <none>                                                                    9001/TCP                    37m
argocd-redis                              ClusterIP      10.100.105.214   <none>                                                                    6379/TCP                    37m
argocd-repo-server                        ClusterIP      10.100.187.156   <none>                                                                    8081/TCP,8084/TCP           37m
argocd-server                             LoadBalancer   10.100.4.28      a847fac2ec3614990bf06c51b0cac3e6-1001046614.ap-south-1.elb.amazonaws.com   80:30958/TCP,443:32183/TCP  37m
argocd-server-metrics                     ClusterIP      10.100.117.161   <none>                                                                    8083/TCP                    37m
root@ip-172-31-14-66:/home/ubuntu# kubectl get deployments -n default
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
netflix-app   2/2     2            2           49s
root@ip-172-31-14-66:/home/ubuntu# kubectl get pods -n default
NAME                           READY   STATUS    RESTARTS   AGE
netflix-app-84b456dcf5-2rdwg   1/1     Running   0          68s
netflix-app-84b456dcf5-k7nzf   1/1     Running   0          68s
root@ip-172-31-14-66:/home/ubuntu# kubectl get svc -n default
NAME          TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes    ClusterIP   10.100.0.1      <none>        443/TCP        63m
netflix-app   NodePort    10.100.49.202   <none>        80:30007/TCP   80s
root@ip-172-31-14-66:/home/ubuntu# kubectl get svc -n default -o wide
NAME          TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE   SELECTOR
kubernetes    ClusterIP   10.100.0.1      <none>        443/TCP        63m   <none>
netflix-app   NodePort    10.100.49.202   <none>        80:30007/TCP   93s   app=netflix-app
root@ip-172-31-14-66:/home/ubuntu# kubectl get pods -n default
NAME                           READY   STATUS    RESTARTS   AGE
netflix-app-84b456dcf5-2rdwg   1/1     Running   0          7m14s
netflix-app-84b456dcf5-k7nzf   1/1     Running   0          7m14s
root@ip-172-31-14-66:/home/ubuntu# kubectl get svc -n default -o wide
NAME          TYPE           CLUSTER-IP      EXTERNAL-IP                                                                PORT(S)        AGE    SELECTOR
kubernetes    ClusterIP      10.100.0.1      <none>                                                                     443/TCP        76m    <none>
netflix-app   LoadBalancer   10.100.36.51    a96392b04a8f74274b391962892561fb-1804913201.ap-south-1.elb.amazonaws.com   80:30007/TCP   110s   app=netflix-app
root@ip-172-31-14-66:/home/ubuntu#
```

Setup the Monitoring using Prometheus & Grafana

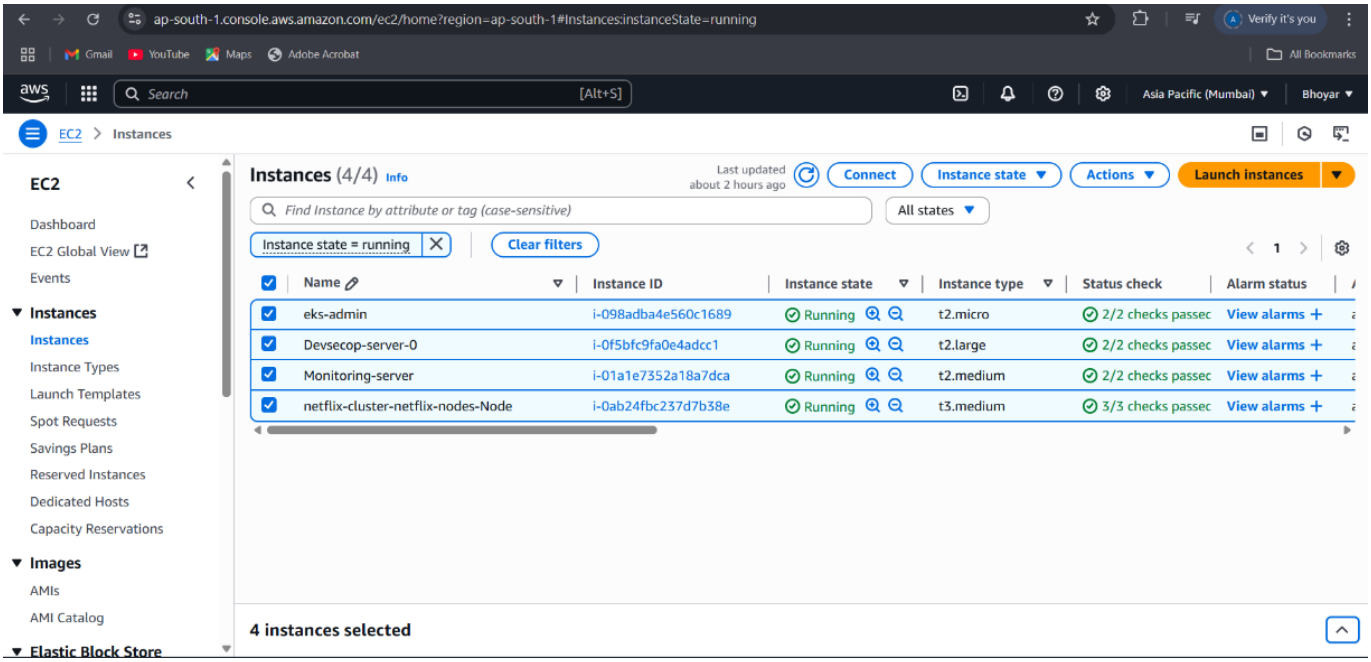




Application running on EKS cluster



EC2 servers that I used.



Challenges Faced

- **Email notifications from Jenkins didn't work** as expected, even though the test email succeeded. Likely causes included incorrect SMTP settings, post-build triggers misfiring, or email configuration being skipped during pipeline execution.
- Despite adding emailext in the pipeline, failure/success emails were not consistently delivered.
- This will be a focus area for improvement in future CI/CD setups.