

DEPLOYED 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 sonarqube url as output**

The screenshot displays a VS Code workspace for a Terraform project. The Explorer on the left shows a directory structure with files like `main.tf`, `output.tf`, `setup.sh`, `variable.tf`, and `terraform.tfstate`. The main editor shows the `main.tf` file with Terraform configuration for an EC2 instance, including `instance_type`, `ami`, `vpc_cidr`, `ec2_count`, `subnet_count`, `aws_availability_zones`, `key_name`, `volume_size`, and `volume_type`. The bottom panel is split into a **DEBUG CONSOLE** and a **TERMINAL**. The terminal shows the execution of `terraform apply`, with logs indicating the successful creation of a VPC, subnets, and EC2 instances. The final output lists the `ec2_public_ips`, `jenkins_url`, `security_group_id`, `sonarqube_url`, `subnets_ids`, and `vpc_id`.

```
modules > terraform.tfvars > # ec2_count
1 instance_type= "t2.large"
2 ami= "ami-0e35ddab05955cf57"
3 vpc_cidr= "10.0.0.0/16"
4 ec2_count= 1
5 subnet_count= 1
6 aws_availability_zones= ["ap-south-1a"]
7 key_name= "netflix-key"
8 volume_size= "30"
9 volume_type= "gp2"
10

PROBLEMS  OUTPUT  TERMINAL  PORTS

DEBAG CONSOLE  TERMINAL

Filter (e.g. text, exclude, \...)

module.vpc.aws_subnet.Public[0]: Creation complete after 11s [id=subnet-07794a39bece52ddd]
module.ec2.aws_instance.Ec2-instance[0]: Creating...
module.vpc.aws_route_table_association.RTA[0]: Creating...
module.vpc.aws_route_table_association.RTA[0]: Creation complete after 1s [id=rtbassoc-0c4319902a937102b]
module.ec2.aws_instance.Ec2-instance[0]: Still creating... [10s elapsed]
module.ec2.aws_instance.Ec2-instance[0]: Creation complete after 13s [id=i-0f5bfc9fa0e4adcc1]

Apply complete! Resources: 7 added, 0 changed, 0 destroyed.

Outputs:

ec2_public_ips = [
  "13.232.164.107",
]
jenkins_url = "http://13.232.164.107:8080"
security_group_id = "sg-0d29806d23f377c05"
sonarqube_url = "http://13.232.164.107:9000"
subnets_ids = [
  "subnet-07794a39bece52ddd",
]
vpc_id = "vpc-07175873d56820ac4"
[root@rhel-ayush modules]#
```

Checkout infra-code from my github repo

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

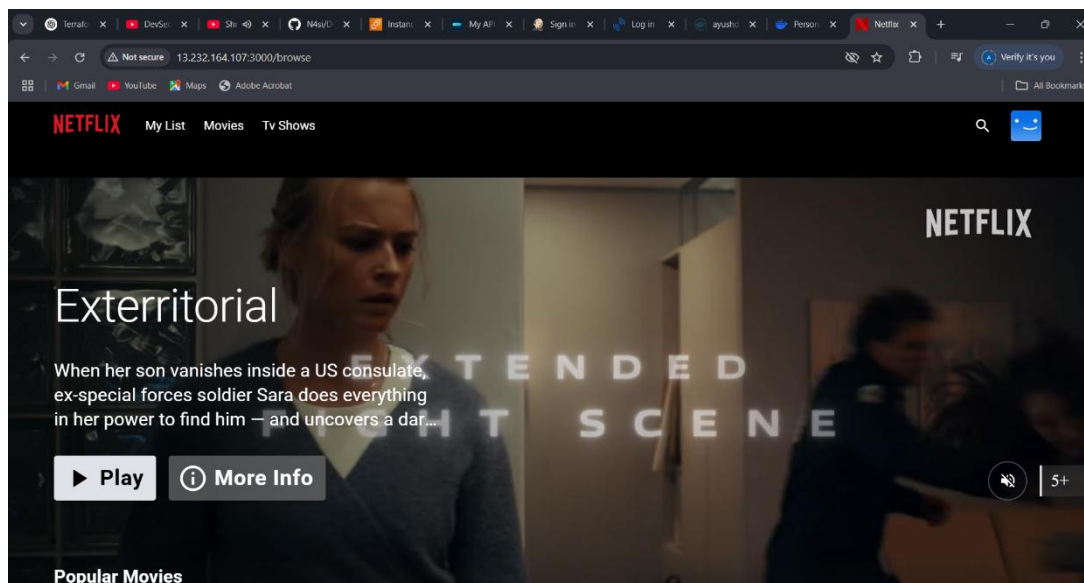
Verified the application functionality locally before deployment

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
```

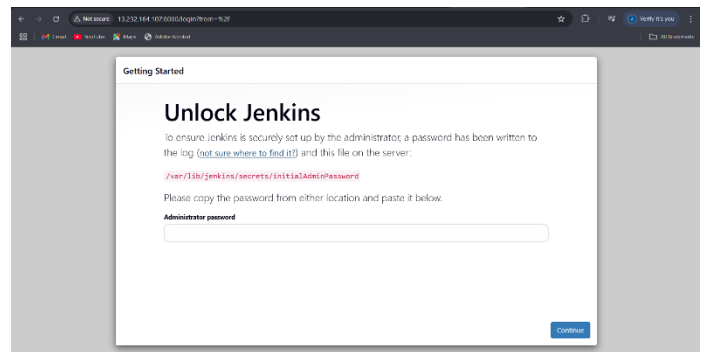
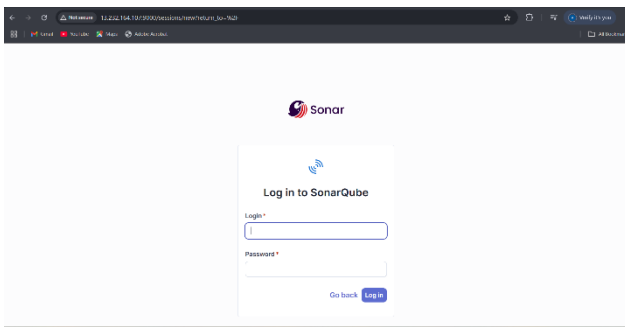
Checked the application is working locally using docker run.



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

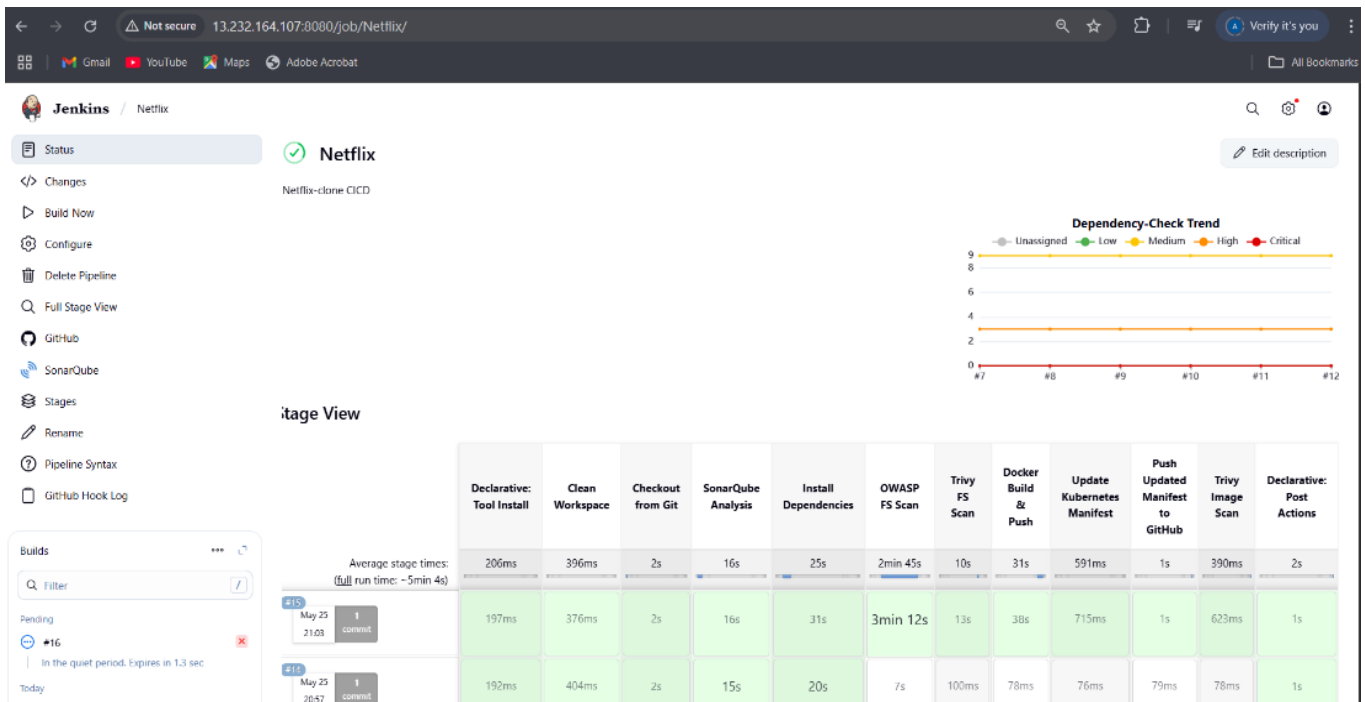
- **Installed essential Jenkins plugins:**
 - **Pipeline**
 - **Docker**
 - **SonarQube Scanner**
 - **OWASP Dependency-Check**
 - **Trivy**
- **Added required credentials in Jenkins:**
 - **Docker Hub credentials (Docker-cred)**
 - **GitHub username and personal access token (github-cred)**
 - **SonarQube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

Dependency-Check Results

SEVERITY DISTRIBUTION

File Name	Vulnerability	Severity	Weakness
+ nanoid:3.3.4	OSSINDEX CVE-2024-55565	Medium	CWE-835
+ postcss:8.4.18	NVD CVE-2023-44270	Medium	CWE-74
+ rollup:2.79.1	NVD CVE-2024-47068	Medium	
+ vite:3.2.2	OSSINDEX CVE-2024-45811	High	CWE-200
+ vite:3.2.2	NVD CVE-2023-34092	High	CWE-706
+ vite:3.2.2	NVD CVE-2024-23331	High	CWE-284
+ vite:3.2.2	OSSINDEX CVE-2025-32395	Medium	CWE-200
+ vite:3.2.2	OSSINDEX CVE-2025-46565	Medium	CWE-22
+ vite:3.2.2	OSSINDEX CVE-2024-31207	Medium	CWE-200
+ vite:3.2.2	OSSINDEX CVE-2025-24010	Medium	CWE-1385

1 of 2

← → ↻ ⚠ Not secure 13.232.164.107:9000/dashboard?id=Netflix&codeScope=overall ☆ 📄 📄 Verify it's you ⋮

📄 Gmail 📄 YouTube 📄 Maps 📄 Adobe Acrobat 📄 All Bookmarks

SonarQube community Projects Issues Rules Quality Profiles Quality Gates Administration More ▾ 🔍 ? A

⚠ Embedded database should be used for evaluation purposes only. It doesn't support scaling, upgrading to a new SonarQube Server version, or migration to another database engine. [Learn more](#) 📄

☆ Netflix / ⓘ main ✓ ▾ ?

Overview Issues Security Hotspots Measures Code Activity Project Settings ▾ Project Information

main 3.2k Lines of Code · Version **not provided** · 🏠 Set as homepage 📄 Take the Tour

Quality Gate ⓘ **Passed** Last analysis 51 minutes ago

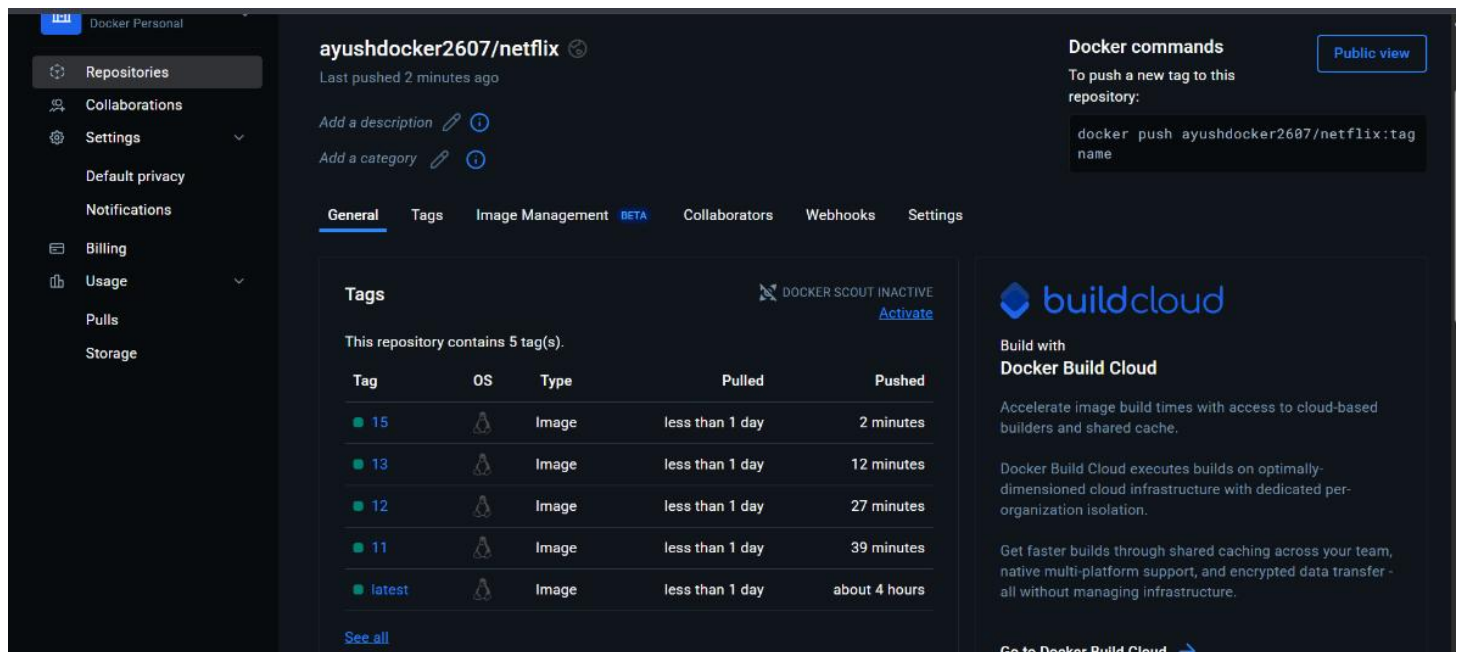
New Code Overall Code

Security 2 Open issues C	Reliability 6 Open issues C	Maintainability 50 Open issues A
Accepted issues	Coverage	Duplications

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



The screenshot shows the Docker Hub interface for the repository **ayushdocker2607/netflix**. The repository was last pushed 2 minutes ago. The left sidebar contains navigation links: Repositories, Collaborations, Settings, Default privacy, Notifications, Billing, Usage, Pulls, and Storage. The main content area has tabs for General, Tags, Image Management (marked BETA), Collaborators, Webhooks, and Settings. The **Tags** tab is active, showing a table of 5 tags. A sidebar on the right contains 'Docker commands' with a 'Public view' button and a command to push a new tag, and a 'buildcloud' advertisement.

ayushdocker2607/netflix Last pushed 2 minutes ago

Add a description Add a category

Tags DOCKER SCOUT INACTIVE Activate

This repository contains 5 tag(s).

Tag	OS	Type	Pulled	Pushed
15	linux	Image	less than 1 day	2 minutes
13	linux	Image	less than 1 day	12 minutes
12	linux	Image	less than 1 day	27 minutes
11	linux	Image	less than 1 day	39 minutes
latest	linux	Image	less than 1 day	about 4 hours

See all

Docker commands Public view

To push a new tag to this repository:

```
docker push ayushdocker2607/netflix:tag name
```

buildcloud

Build with Docker Build Cloud

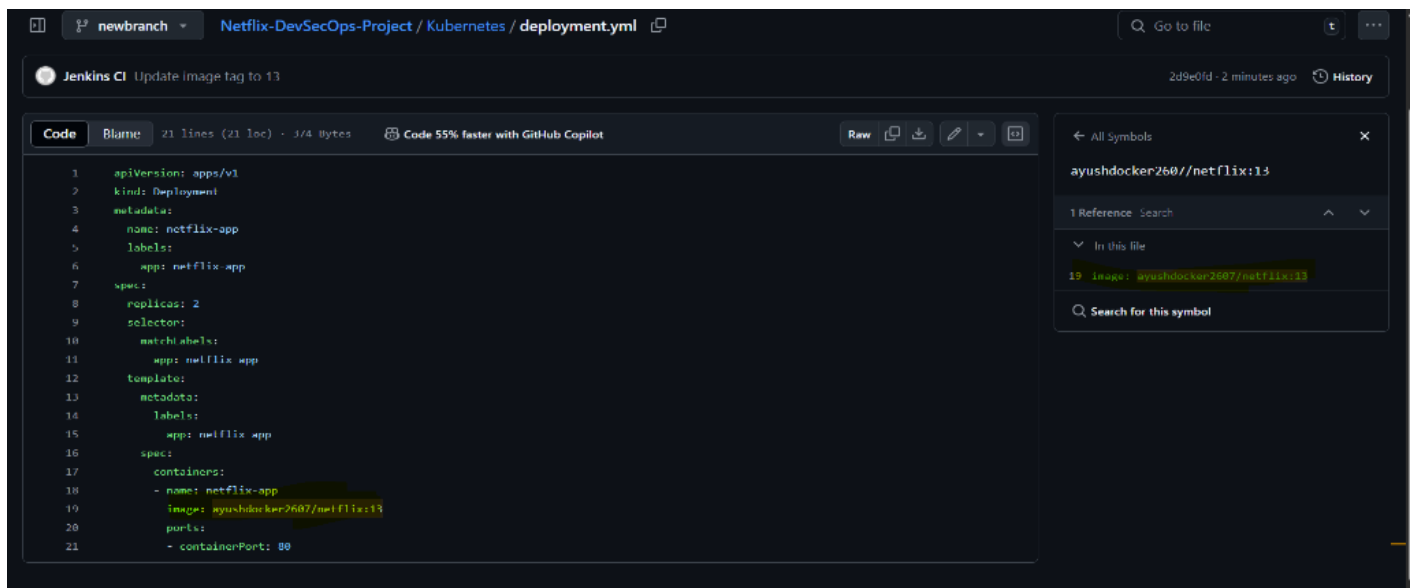
Accelerate image build times with access to cloud-based builders and shared cache.

Docker Build Cloud executes builds on optimally-dimensioned cloud infrastructure with dedicated per-organization isolation.

Get faster builds through shared caching across your team, native multi-platform support, and encrypted data transfer - all without managing infrastructure.

Go to Docker Build Cloud

Image tag getting updated in manifest files in GIT repo automatically



The screenshot shows a Jenkins CI job titled 'Update image tag to 13' for the 'Netflix-DevSecOps-Project / Kubernetes / deployment.yml' file. The job was last updated 2d9e0ld - 2 minutes ago. The 'Code' tab is active, showing the contents of the deployment.yml file. The file is a Kubernetes Deployment manifest for 'netflix-app'. The 'image' field in the 'containers' section is highlighted in yellow and matches the tag '13' in the job title. A sidebar on the right shows a search for the symbol 'ayushdocker2607/netflix:13'.

Jenkins CI Update image tag to 13 2d9e0ld - 2 minutes ago History

Code Blame 21 lines (21 loc) - 3/4 Bytes Code 55% faster with GitHub Copilot

```
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   name: netflix-app
5   labels:
6     app: netflix-app
7 spec:
8   replicas: 2
9   selector:
10     matchLabels:
11       app: netflix-app
12   template:
13     metadata:
14       labels:
15         app: netflix-app
16     spec:
17       containers:
18       - name: netflix-app
19         image: ayushdocker2607/netflix:13
20       ports:
21       - containerPort: 80
```

ayushdocker2607/netflix:13

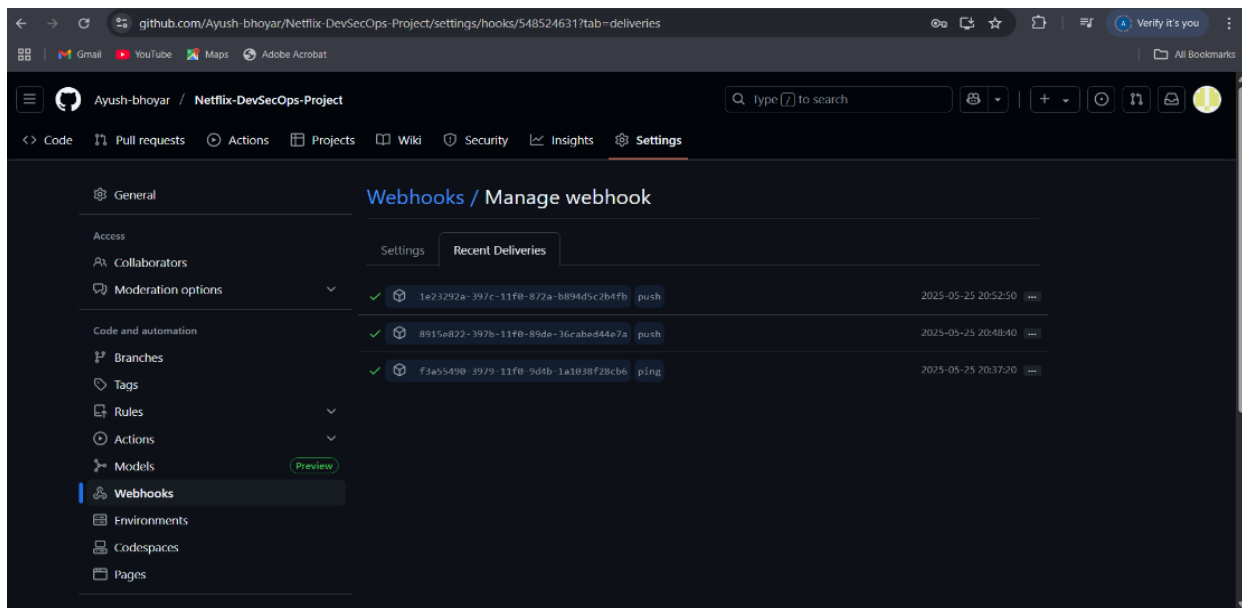
1 Reference Search

In this file

19 image: ayushdocker2607/netflix:13

Search for this symbol

For triggering the CI/CD 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 [!] to cleanup resources, run 'eksctl delete cluster --region=ap-south-1 --name=netflix-cluster'
2025-05-25 12:38:48 [!] Error: failed to cleanup resources: 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 [!] eksctl version 0.208.0
2025-05-25 12:41:55 [!] 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 which 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 nodegroups in Eksctl will be changed to AL2023 in the future.
2025-05-25 12:41:55 [!] skipping ap-south-1c from selection because it doesn't support the following instance type(s): t2.medium
2025-05-25 12:41:55 [!] setting availability zones to [ap-south-1a ap-south-1b]
2025-05-25 12:41:55 [!] subnets for ap-south-1a - public:192.168.0.0/19 private:192.168.64.0/19
2025-05-25 12:41:55 [!] subnets for ap-south-1b - public:192.168.32.0/19 private:192.168.96.0/19
2025-05-25 12:41:55 [!] nodegroup "netflix-nodes" will use "" [AmazonLinux2/1.32]
2025-05-25 12:41:55 [!] using Kubernetes version 1.32
2025-05-25 12:41:55 [!] creating EKS cluster "netflix-cluster" in "ap-south-1" region with managed nodes
2025-05-25 12:41:55 [!] will create 2 separate CloudFormation stacks for cluster itself and the initial managed nodegroup
2025-05-25 12:41:55 [!] 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 [!] 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 [!] CloudWatch logging will not be enabled for cluster "netflix-cluster" in "ap-south-1"
2025-05-25 12:41:55 [!] 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 [!] default addons coredns, metrics-server, vpc-cni, kube-proxy were not specified, will install them as EKS addons
2025-05-25 12:41:55 [!]
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 [!] building cluster stack "eksctl-netflix-cluster-cluster"
2025-05-25 12:41:55 [!] deploying stack "eksctl-netflix-cluster-cluster"
2025-05-25 12:42:25 [!] waiting for CloudFormation stack "eksctl-netflix-cluster-cluster"
2025-05-25 12:42:55 [!] waiting for CloudFormation stack "eksctl-netflix-cluster-cluster"
2025-05-25 12:43:55 [!] waiting for CloudFormation stack "eksctl-netflix-cluster-cluster"
2025-05-25 12:44:55 [!] waiting for CloudFormation stack "eksctl-netflix-cluster-cluster"
2025-05-25 12:45:55 [!] waiting for CloudFormation stack "eksctl-netflix-cluster-cluster"
2025-05-25 12:46:55 [!] 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.

The screenshot shows the ArgoCD web interface for an application named 'netflix'. The interface is divided into several sections:

- Header:** Shows the ArgoCD logo and version 'v3.0.3+ta14b012'.
- Left Sidebar:** Contains navigation links for 'Applications', 'Settings', 'User Info', and 'Documentation'. Below these are 'Resource filters' for 'NAME' and 'KINDS'.
- Top Navigation:** Includes links for 'APPLICATION DETAILS TREE', 'DETAILS', 'DIFF', 'SYNC', 'SYNC STATUS', 'HISTORY AND ROLLBACK', 'DELETE', and 'REFRESH'.
- Main Content Area:**
 - APP HEALTH:** Displays a green heart icon and the status 'Healthy'.
 - SYNC STATUS:** Shows a green checkmark and the status 'Synced to newbranch (f52f515)'. It also indicates 'Auto sync is enabled' and provides author and comment information.
 - LAST SYNC:** Shows a green checkmark and the status 'Sync OK to f52f515'. It indicates 'Succeeded 4 minutes ago (Sun May 25 2025 19:55:10 GMT+0530)' and provides author and comment information.
 - Resource View:** A diagram showing the application's resources, including pods and services, with a progress bar indicating 70% completion.

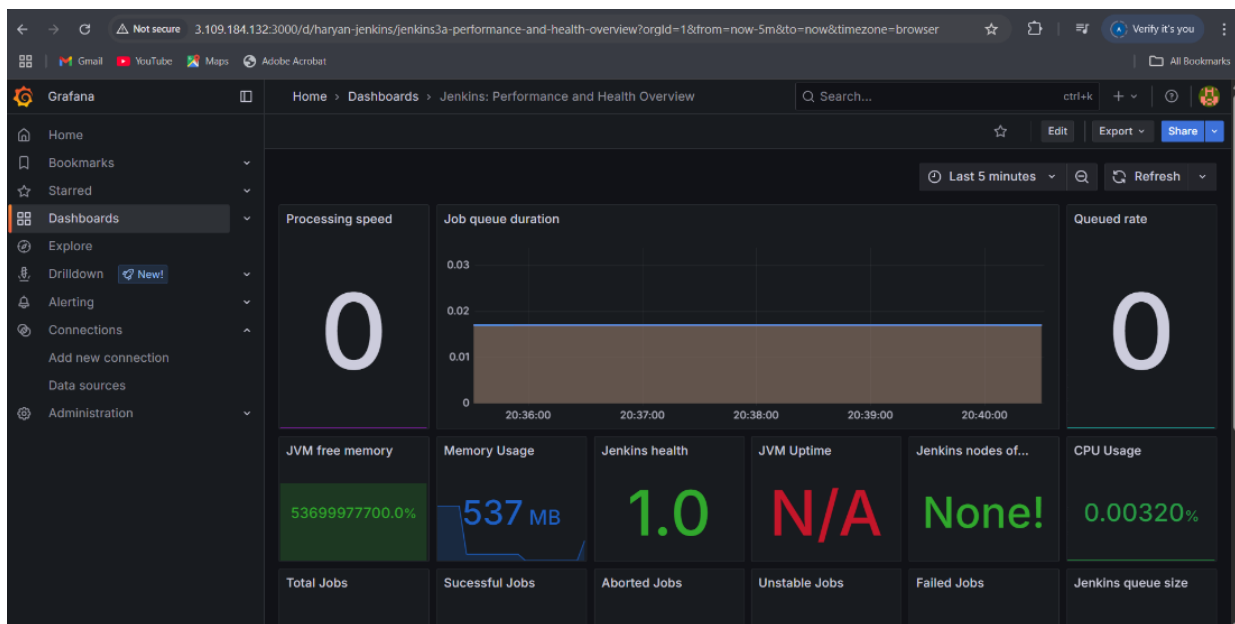
EKS cluster all pods and services also you can see the EKS cluster name

```
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
YX5zeVftbD0i7q8o
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#
```

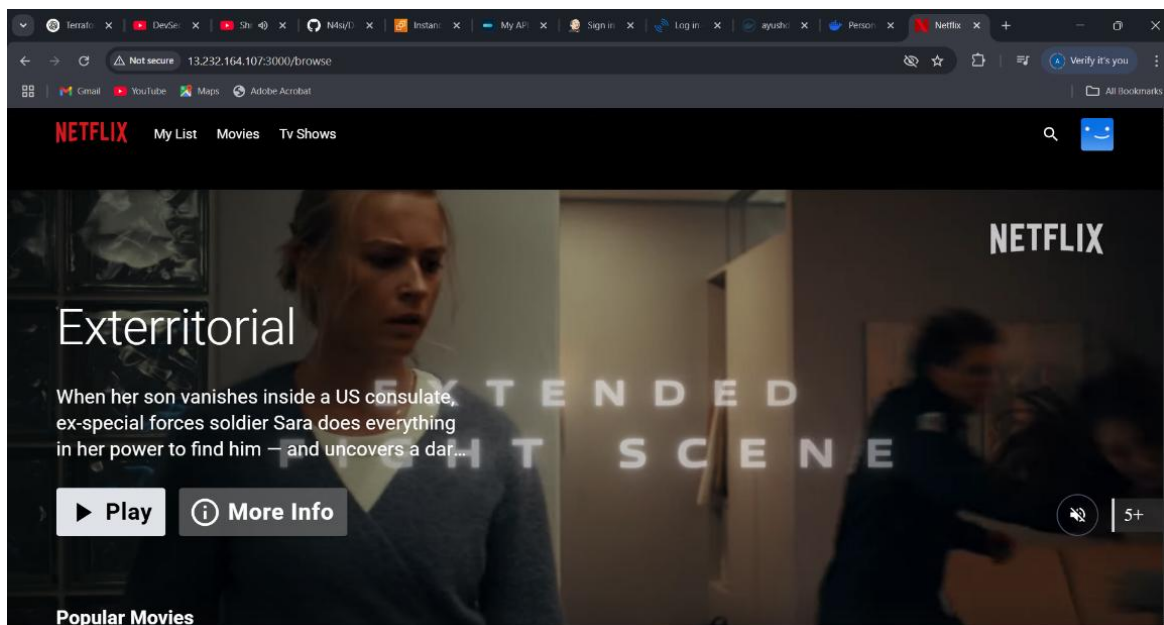
Setup the Monitoring using Prometheus & Grafana

The screenshot shows the Prometheus web interface. At the top, there's a navigation bar with 'Prometheus', 'Alerts', 'Graph', 'Status', and 'Help'. Below this is the 'Targets' section. It includes a filter bar with 'All scrape pools', 'All', 'Unhealthy', and 'Collapse All' buttons, along with a search box labeled 'Filter by endpoint or labels'. On the right, there are status indicators: 'Unknown' (checked), 'Unhealthy' (checked), and 'Healthy' (checked). The main content area lists three targets:

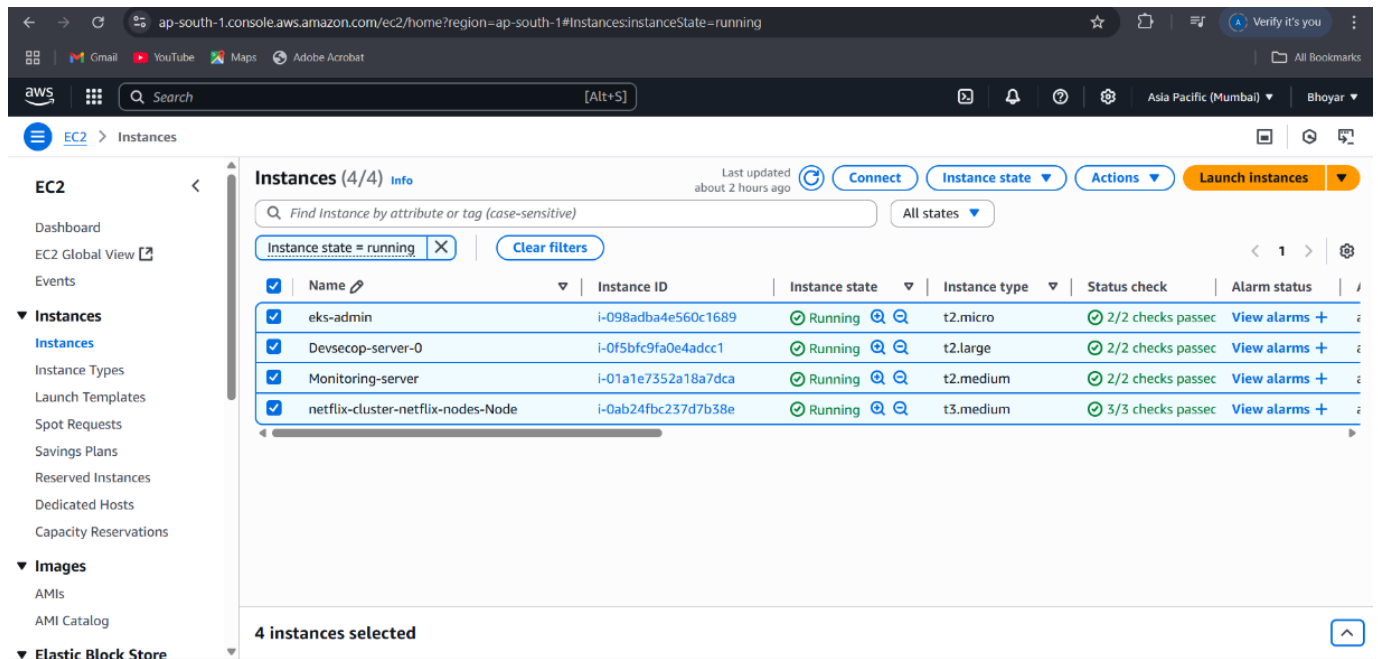
- jenkins (1/1 up)** (show less)
 - Endpoint: <http://13.232.164.107:8080/prometheus>
 - State: UP
 - Labels: Instance="13.232.164.107:8080", job="jenkins"
 - Last Scrape: 1.828s ago
 - Scrape Duration: 10.086ms
 - Error:
- nodeexporter (1/1 up)** (show less)
 - Endpoint: <http://3.109.184.132:9100/metrics>
 - State: UP
 - Labels: Instance="3.109.184.132:9100", job="nodeexporter"
 - Last Scrape: 4.79s ago
 - Scrape Duration: 14.582ms
 - Error:
- prometheus (1/1 up)** (show less)
 - Endpoint: <http://localhost:9090/metrics>
 - State: UP
 - Labels: Instance="localhost:9090", job="prometheus"
 - Last Scrape: 13.509s ago
 - Scrape Duration: 4.204ms
 - Error:



Application running on EKS cluster



EC2 servers that I used.



The screenshot shows the AWS Management Console for the 'ap-south-1' region. The 'Instances' page is active, displaying a list of 4 running EC2 instances. The left sidebar shows the navigation menu with 'Instances' selected. The top navigation bar includes the AWS logo, a search bar, and the region 'Asia Pacific (Mumbai)'. The main content area shows the 'Instances (4/4)' header with a search bar and filters. The filter 'Instance state = running' is applied. The table below lists the instances:

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status
<input checked="" type="checkbox"/>	eks-admin	i-098adba4e560c1689	Running	t2.micro	2/2 checks passed	View alarms
<input checked="" type="checkbox"/>	Devsecop-server-0	i-0f5bfc9fa0e4adcc1	Running	t2.large	2/2 checks passed	View alarms
<input checked="" type="checkbox"/>	Monitoring-server	i-01a1e7352a18a7dca	Running	t2.medium	2/2 checks passed	View alarms
<input checked="" type="checkbox"/>	netflix-cluster-netflix-nodes-Node	i-0ab24fbc237d7b38e	Running	t3.medium	3/3 checks passed	View alarms

4 instances selected

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.