End-to-End Netflix Clone Deployment with Kubernetes, SonarQube, Trivy, Jenkins, Prometheus, Grafana, Argo CD

1. Initial Setup

1.1 Prerequisites

Before starting, ensure the following are installed and configured:

- ***Docker** for containerization.*
- ***Kubernetes (K8s)**: Either set up a local Kubernetes environment (using Minikube or kind) or use a cloud provider's Kubernetes service (e.g., EKS, GKE, AKS).*
- ***Helm** for managing Kubernetes applications.*
- ***Git** for version control.*
- ***Jenkins** for Continuous Integration/Continuous Delivery (CI/CD).*
- ***Argo CD** for GitOps-based deployment.*
- ***SonarQube** for code quality analysis.*
- ***Trivy** for container scanning.*
- ***Prometheus and Grafana** for monitoring and visualization.*

1.2 Create a Netflix Clone Application

Create or use an existing simple microservices-based application mimicking Netflix. The architecture could include:

- ***Frontend**: React or Angular app to simulate a Netflix-like UI.*
- ***Backend**: REST APIs for handling user data, content, recommendations, etc.*
- ***Database**: Use a database like PostgreSQL or MongoDB.*
- ***Authentication**: JWT or OAuth-based authentication.*

2. Containerization with Docker

2.1 Dockerizing the Application

Each part of the application (Frontend, Backend, Database) needs to be dockerized. Create a `Dockerfile` for each service:

Frontend Dockerfile
Dockerfile
CopyEdit

```
# Frontend Dockerfile
FROM node:14 AS build
WORKDIR /app
COPY . .
RUN npm install
RUN npm run build

FROM nginx:alpine
```

```
COPY --from=build /app/build /usr/share/nginx/html
```

1.

Backend Dockerfile

```
Dockerfile
CopyEdit
# Backend Dockerfile
FROM openjdk:11-jre-slim
WORKDIR /app
COPY . /app
RUN ./gradlew build
CMD ["java", "-jar", "build/libs/your-app.jar"]
```

2.

Database Dockerfile *(if using custom database)*

```
Dockerfile
CopyEdit
FROM postgres:latest
ENV POSTGRES_USER=admin
ENV POSTGRES_PASSWORD=password
ENV POSTGRES_DB=netflix_clone
```

3.

2.2 Build and Push Images

Use Docker commands to build and push images to a Docker registry (like DockerHub or a private registry).

```
bash
CopyEdit
docker build -t username/netflix-frontend .
```

```
docker push username/netflix-frontend
```

3. Kubernetes Setup

3.1 Kubernetes Cluster Setup

Minikube Setup (for local development):

bash

CopyEdit

```
minikube start
```

1.
2. **Cloud Kubernetes (EKS/GKE/AKS):** Follow the respective documentation to set up a Kubernetes cluster.

3.2 Deploying the Application on Kubernetes

Create Kubernetes manifests (YAML files) for each microservice (frontend, backend, database).

Frontend Deployment

yaml

CopyEdit

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
spec:
  replicas: 2
  selector:
```

```
    matchLabels:
      app: frontend
  template:
    metadata:
      labels:
        app: frontend
    spec:
      containers:
      - name: frontend
        image: username/netflix-frontend:latest
        ports:
        - containerPort: 80
```

1.

Backend Deployment

yaml

CopyEdit

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend
spec:
  replicas: 2
  selector:
    matchLabels:
      app: backend
  template:
    metadata:
      labels:
        app: backend
```

```
    spec:
      containers:
      - name: backend
        image: username/netflix-backend:latest
        ports:
        - containerPort: 8080
```

2.

Service Definitions Define services for exposing the deployments:

yaml

CopyEdit

```
apiVersion: v1
kind: Service
metadata:
  name: frontend-service
spec:
  selector:
    app: frontend
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

3. ***Service Definitions Define services for exposing the deployments:***

```
apiVersion: v1

kind: Service
```

```
metadata:
  name: frontend-service
spec:
  selector:
    app: frontend
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

4. CI/CD Pipeline wi

4. CI/CD Pipeline with Jenkins

4.1 Jenkins Setup

1. ***Install Jenkins** on a VM or cloud instance.*
2. ***Install Plugins**: Kubernetes CLI plugin, GitHub plugin, Docker plugin, etc.*

***Create Jenkinsfile** for the pipeline. A simple Jenkinsfile might look like this:*

```
groovy
CopyEdit
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                script {

docker.build('username/netflix-backend')

docker.build('username/netflix-frontend')
                }
            }
        }
        stage('Test') {
            steps {
                sh 'docker run username/netflix-backend
test'
            }
        }
        stage('Push to Registry') {
            steps {
                withDockerRegistry([credentialsId:
'dockerhub-credentials', url:
'https://index.docker.io/v1/']) {
                    sh 'docker push
username/netflix-backend'
                    sh 'docker push
username/netflix-frontend'
```

```
                }
            }
        }
        stage('Deploy') {
            steps {
                sh 'kubectl apply -f
k8s/frontend-deployment.yaml'
                sh 'kubectl apply -f
k8s/backend-deployment.yaml'
            }
        }
    }
}
```

4.2 Triggering the Pipeline

Set up webhooks in GitHub/GitLab to trigger Jenkins jobs upon code pushes.

5. Code Quality with SonarQube

5.1 Set up SonarQube

1. ***Install SonarQube** on a VM or cloud instance.*
2. *Create a **SonarQube Project** for each service (Frontend, Backend) and configure the SonarQube scanner in Jenkins.*

SonarQube Scanner Configuration: *Add this to your Jenkins pipeline to run SonarQube analysis:*

groovy

CopyEdit

```
stage('SonarQube Analysis') {
    steps {
        script {
            withSonarQubeEnv('SonarQube') {
                sh 'mvn clean verify sonar:sonar
-Dsonar.projectKey=netflix-backend'
            }
        }
    }
}
```

3. ***SonarQube Scanner Configuration:*** *Add this to your Jenkins pipeline to run SonarQube analysis:*

```
stage('SonarQube Analysis') {

    steps {

        script {

            withSonarQubeEnv('SonarQube') {

                sh 'mvn clean verify sonar:sonar
-Dsonar.projectKey=netflix-backend'

            }

        }

    }
```

```
}
```

6. Security with Trivy

6.1 Install Trivy

Trivy can scan your Docker images for vulnerabilities. Add Trivy scanning in your Jenkins pipeline:

groovy
CopyEdit

```
stage('Scan with Trivy') {
    steps {
        script {
            sh 'trivy image --exit-code 1 --no-progress username/netflix-backend'
        }
    }
}
```

7. Monitoring with Prometheus and Grafana

7.1 Set up Prometheus

Install Prometheus on Kubernetes using Helm:

bash

CopyEdit

```
helm install prometheus
prometheus-community/kube-prometheus-stack
```

1.
2. *Configure **Prometheus scraping** of your services by adding Prometheus annotations to your service YAML files.*

7.2 Set up Grafana

Grafana is used for visualizing metrics from Prometheus:

Install Grafana:

bash

CopyEdit

```
helm install grafana grafana/grafana
```

1.
2. *Connect Grafana to Prometheus as a data source and create dashboards for monitoring the health and performance of your services.*

8. GitOps with Argo CD

8.1 Set up Argo CD

***Install Argo CD** on Kubernetes:*

bash

CopyEdit

```
kubectl create namespace argocd
```

kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml

1.
2. *Connect Argo CD to your Git repository where your Kubernetes manifests are stored.*
3. *Create an **Application** in Argo CD to sync your deployment files with Kubernetes.*

End-to-End Netflix Clone Deployment with Kubernetes, SonarQube, Trivy, Jenkins, Prometheus, and Grafana