https://mrcloudbook.com/youtube-clone-app-with-devsecops-and-jenkins-shared-library/

https://github.com/Sravyatirumala/Youtube-clone-app

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

docker update —restart=always sonar

def COLOR\_MAP = [

'FAILURE' : 'danger',

'SUCCESS' : 'good'

]

pipeline{

agent any

tools{

jdk 'jdk20’

nodejs 'nodeJS’

}

environment {

SCANNER\_HOME=tool 'sonar-scanner'

}

stages {

stage('clean workspace'){

steps{

cleanWs()

}

}

stage('Checkout from Git'){

steps{

git branch: 'main', url: 'https://github.com/Sravyatirumala/Youtube-clone-app.git

}

}

stage("Sonarqube Analysis "){

steps{

withSonarQubeEnv('sonar-server') {

sh ''' $SCANNER\_HOME/bin/sonar-scanner -Dsonar.projectName=youtube \

-Dsonar.projectKey=youtube '''

}

}

}

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', odcInstallation: 'DP-Check'

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-‘creds){

sh "docker build --build-arg REACT\_APP\_RAPID\_API\_KEY=f0ead79813ms -t youtube ."

sh "docker tag youtube sravyatirumala/youtube:latest "

sh "docker push sravyatirumala/youtube:latest "

}

}

}

}

stage("TRIVY"){

steps{

sh "trivy image sevenajay/youtube:latest > trivyimage.txt"

}

}

stage('Deploy to container'){

steps{

sh 'docker run -d --name youtube1 -p 3000:3000 sevenajay/youtube:latest'

}

}

stage('Deploy to kubernets'){

steps{

script{

withKubeConfig(caCertificate: '', clusterName: '', contextName: '', credentialsId: 'k8s', namespace: '', restrictKubeConfigAccess: false, serverUrl: '') {

sh 'kubectl apply -f deployment.yml'

}

}

}

}

}

post {

always {

echo 'Slack Notifications'

slackSend (

channel: ‘#jenkins’,

color: COLOR\_MAP[currentBuild.currentResult],

message: "\*${currentBuild.currentResult}:\* Job ${env.JOB\_NAME} \n build ${env.BUILD\_NUMBER} \n More info at: ${env.BUILD\_URL}"

)

}

}

}

Install docker, kubectl, jenkins, sonarqube etc.

………………………………………………………………………………………………………………………………………………

Install Splunk in 2nd instance:

wget -O splunk-9.1.1-64e843ea36b1-linux-2.6-amd64.deb "https://download.splunk.com/products/splunk/releases/9.1.1/linux/splunk-9.1.1-64e843ea36b1-linux-2.6-amd64.deb"

sudo dpkg -i splunk-9.1.1-64e843ea36b1-linux-2.6-amd64.deb

sudo /opt/splunk/bin/splunk enable boot-start

Create username and password:

Username: admin

Password: sravya@95

sudo ufw allow openSSH

sudo ufw allow 8000

sudo ufw status

sudo ufw enable

………………………………………………………………………………………………………………………………………………

Start splunk:

sudo /opt/splunk/bin/splunk start

You’ll be prompted to log in with the administrator username and password you created during the setup process (typically using the command sudo /opt/splunk/bin/splunk enable boot-start

Now login splunk in UI with username and password:

3.14.253.246 :8000

………………………………………………………………………………………………………………………………………………

**Install the Splunk app for Jenkins**

In Splunk Dashboard

Click on Apps –> Find more apps

Install jenkins for splunk.

We need to give login and password it will download.

On the homepage of Splunk, you will see Jenkins has been added

In the Splunk web interface, go to Settings > Data Inputs.

Click on HTTP Event Collector.

Set All tokens to enabled

Uncheck SSL enable

Use 8088 port and click on save

Click new token : splunk-token — next—submit .. Copy token (setting — data inputs — http event collector —token)

Token : 668443ee-4282-4f9a-aff5-5220215e1910

Click on start searching :

………………………………………………………………………………………………………………………………………………

Add splunk plugin in jenkins dashbord:

Now Manage jenkins —System — Splunk — Enable

HTTP input host as SPLUNK PUBLIC IP. : 3.14.253.246

HTTP token that you generated in Splunk : 668443ee-4282-4f9a-aff5-5220215e1910

Unckeck SSL

Jenkins Ip : 3.138.66.92 Test Connection .

Now Go to Instance: sudo ufw allow 8088

sudo ufw status

**Restart Both Splunk and Jenkins:**

**Splunk : Setting —server controls — restart Splunk..**

**Jenkins: http://3.138.66.92:8080/restart**

………………………………………………………………………………………………………………………………………………

**Integrate Slack for Notifications**

Create account and login to slack .

Click on channels — create channel Jenkins

Now go to slack and click on settings — manage apps — Installed apps — Search Jenkins CI — Add to slack — Give # jenkins channel add Jenkins CI Integration.

Copy : Copy the team subdomain and integration token credential ID for later use

*• Team subdomain:*bondevalueworkspace

*• Integration token credential ID: Create a secret text credential using*XlI1PIxvtweYg7wBWjampgf8*as the value*

This will give us some instructios :

………………………………………………………………………………………………………………………………………………

**Install Slack Notification Plugin in Jenkins**

Click on **Manage Jenkins — plugins** — Available — Slack Notification —Install

Manage Jenkins — Credential — Create secret text — Copy token from above where u got credential id — save.

Manage Jenkins —system —slack — bondevalueworkspace (Copy subdomain from CI integration) —Credentials (slack ) — channel name (#jenkins) which u have created.

………………………………………………………………………………………………………………………………………………

**Go to Git-hub : Create repo : Jenkins\_shared-Library**

**Copy code and apply in folder.**

**Create folder in local : Jenkins\_shared-Library —open in VS code — apply init and all commands.**

**Create vars folder .**

Now, Let’s Write a Groovy script for our Pipeline

1. Create a **cleanWorkspace.groovy** file and add the below code.

def call() {

cleanWs()

}

2. Create **checkoutGit.groovy**

**def call(String gitUrl, String gitBranch) {**

**checkout([**

**$class: 'GitSCM',**

**branches: [[name: gitBranch]],**

**userRemoteConfigs: [[url: gitUrl]]**

**])**

**}**

3. npmInstall.groovy

def call() {

sh 'npm install'

}

4. qualityGate.groovy

def call(credentialsId) {

waitForQualityGate abortPipeline: false, credentialsId: credentialsId

}

5. sonarqubeAnalysis.groovy

def call() {

withSonarQubeEnv('sonar-server') {

sh ''' $SCANNER\_HOME/bin/sonar-scanner -Dsonar.projectName=youtube1 \

-Dsonar.projectKey=youtube1 '''

}

}

6. trivyFs.groovy

def call() {

sh 'trivy fs . > trivyfs.txt'

}

7. dockerBuild.groovy

def call(String dockerHubUsername, String imageName) {

// Build the Docker image

sh "docker build --build-arg REACT\_APP\_RAPID\_API\_KEY=b3e6016657m -t ${imageName} ."

// Tag the Docker image

sh "docker tag ${imageName} ${dockerHubUsername}/${imageName}:latest"

// Push the Docker image

withDockerRegistry([url: 'https://index.docker.io/v1/', credentialsId: 'Docker-creds']) {

sh "docker push ${dockerHubUsername}/${imageName}:latest"

}

}

**8. trivyImage.groovy**

def call() {

sh 'trivy image sravyatirumala/youtube:latest > trivyimage.txt'

}

9.  **runContainer.groovy**

def call(){

sh "docker run -d --name youtube1 -p 3000:3000 sravyatirumala/youtube:latest"

}

10. **removeContainer.groovy**

def call(){

sh 'docker stop youtube1'

sh 'docker rm youtube1'

}

**11. kubeDeploy.groovy**

**def call() {**

**withKubeConfig(caCertificate: '', clusterName: '', contextName: '', credentialsId: 'k8\'scertificate', namespace: '', restrictKubeConfigAccess: false, serverUrl: '')**

**sh "kubectl apply -f deployment.yml"**

**}**

**12. kubeDelete.groovy**

**def call() {**

**withKubeConfig(caCertificate: '', clusterName: '', contextName: '', credentialsId: 'k8\'scertificate', namespace: '', restrictKubeConfigAccess: false, serverUrl: '')**

**sh "kubectl delete -f deployment.yml"**

**}**

**Push files to git repo : Jenkins\_Shared\_library: git add .**

**git commit -m “first checkout”**

**git push origin main.**

………………………………………………………………………………………………………………………………………………

**Go to Jenkins pipeline — create pipeline syntax — click git checkout — give https://github.com/Sravyatirumala/Jenkins\_shared-Library.git This will create script:**

**branches: [[name: '\*/main']], extensions: [], userRemoteConfigs: [[url: 'https://github.com/Sravyatirumala/Youtube-clone-app.git']])**

**Add Jenkins shared library to Jenkins system:**

**Manage jenkins — system —** Global Trusted Pipeline Libraries — Name : jenkins\_shared\_library

Git repo: https://github.com/Sravyatirumala/Jenkins\_shared-Library.git

**Library Path : ./**

………………………………………………………………………………………………………………………………………………

**Run Pipeline:**

**@Library('jenkins\_shared\_library') \_**

**def COLOR\_MAP = [**

**'FAILURE': 'danger',**

**'SUCCESS': 'good'**

**]**

**@Library('jenkins\_shared\_library') \_**

**def COLOR\_MAP = [**

**'FAILURE': 'danger',**

**'SUCCESS': 'good'**

**]**

**pipeline {**

**agent any**

**parameters {**

**choice(name: 'action', choices: ['create', 'delete'], description: 'Select create or destroy.')**

**string(name: 'DOCKER\_HUB\_USERNAME', defaultValue: 'sravyatirumala', description: 'Docker Hub Username')**

**string(name: 'IMAGE\_NAME', defaultValue: 'youtube', description: 'Docker Image Name')**

**}**

**tools {**

**jdk 'jdk20'**

**nodejs 'nodeJS'**

**}**

**environment {**

**SCANNER\_HOME = tool 'sonar-scanner'**

**}**

**stages {**

**stage('Clean Workspace') {**

**steps {**

**script {**

**cleanWorkspace()**

**}**

**}**

**}**

**stage('Checkout from Git') {**

**steps {**

**script {**

**checkoutGit('https://github.com/Sravyatirumala/Youtube-clone-app.git', 'main')**

**}**

**}**

**}**

**stage('SonarQube Analysis') {**

**when {**

**expression { params.action == 'create' }**

**}**

**steps {**

**sonarqubeAnalysis()**

**}**

**}**

**stage('SonarQube Quality Gate') {**

**when {**

**expression { params.action == 'create' }**

**}**

**steps {**

**script {**

**def credentialsId = 'Sonar-token'**

**qualityGate(credentialsId)**

**}**

**}**

**}**

**stage('Npm Install') {**

**when {**

**expression { params.action == 'create' }**

**}**

**steps {**

**npmInstall()**

**}**

**}**

**stage('Trivy File Scan') {**

**when {**

**expression { params.action == 'create' }**

**}**

**steps {**

**trivyFs()**

**}**

**}**

**// stage('OWASP FS SCAN') {**

**// when {**

**// expression { params.action == 'create' }**

**// }**

**// steps {**

**// dependencyCheck additionalArguments: '--scan ./ --disableYarnAudit --disableNodeAudit', odcInstallation: 'DP-Check'**

**// dependencyCheckPublisher pattern: '\*\*/dependency-check-report.xml'**

**// }**

**// }**

**stage('Docker Build') {**

**when {**

**expression { params.action == 'create' }**

**}**

**steps {**

**script {**

**def dockerHubUsername = params.DOCKER\_HUB\_USERNAME**

**def imageName = params.IMAGE\_NAME**

**dockerBuild(dockerHubUsername, imageName)**

**}**

**}**

**}**

**stage('Trivy Image Scan') {**

**when {**

**expression { params.action == 'create' }**

**}**

**steps {**

**trivyImage()**

**}**

**}**

**stage('Run Container') {**

**when {**

**expression { params.action == 'create' }**

**}**

**steps {**

**runContainer()**

**}**

**}**

**// stage('Remove Container') {**

**// when {**

**// expression { params.action == 'delete' }**

**// }**

**// steps {**

**// removeContainer()**

**// }**

**// }**

**stage('Kube Deploy') {**

**when {**

**expression { params.action == 'create' }**

**}**

**steps {**

**kubeDeploy()**

**}**

**}**

**// stage('Kube Delete') {**

**// when {**

**// expression { params.action == 'delete' }**

**// }**

**// steps {**

**// kubeDelete()**

**// }**

**// }**

**}**

**post {**

**always {**

**echo 'Slack Notifications'**

**slackSend(**

**channel: '#jenkins',**

**color: COLOR\_MAP.get(currentBuild.currentResult, 'warning'),**

**message: "\*${currentBuild.currentResult}:\* Job `${env.JOB\_NAME}`\nBuild #${env.BUILD\_NUMBER}\n<${env.BUILD\_URL}|View Details>"**

**)**

**}**

**}**

**}**

NOTE:

Stage with the Dependency Check steps cannot be directly used inside a shared library.

The main reason is that pipelines loaded from shared libraries have more restrictive script security by default. So the dependencyCheck and dependencyCheckPublisher steps would fail with rejected signature errors.

………………………………………………………………………………………………………………………………………………

**Create an API key from Rapid API**

**Login and search for YoutubeV3 : Copy API key**

Copy API and use it in the groovy file

docker build –build-arg REACT\_APP\_RAPID\_API\_KEY=<API-KEY> -t ${imageName}

Search for Youtubev3 — create API key —copy and paste in dockerBuild.groovy

**http:// ip : 3000**

**Once it is done delete container by changing script and it will remove container.**

………………………………………………………………………………………………………………………………………………

Now Save Kube credentials in Global credentials :

Manage jenkins — system —global —credentials — K8’s cert : Upload cert. **k8scert**

To get cert :

cd .kube

cat config : copy cert and paste in local and upload.

………………………………………………………………………………………………………………………………………………

**Install Helm & Monitoring K8S using Prometheus and Grafana**

curl -fsSL -o get\_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3

chmod 700 get\_helm.sh

./get\_helm.sh

helm repo add stable https://charts.helm.sh/stable

Add Prometheus:

helm repo add prometheus-community https://prometheus-community.github.io/helm-charts

Create Prometheus namespace:

kubectl create namespace prometheus

**Install kube-Prometheus-stack:**

**helm install stable prometheus-community/kube-prometheus-stack -n prometheus**

**kubectl get pods -n prometheus**

**kubectl get svc -n prometheus**

**Edit Prometheus Service:**

**kubectl edit svc stable-kube-prometheus-sta-prometheus -n prometheus**

**Type: LoadBalancer.**

**kubectl edit svc stable-grafana -n prometheus**

**Type: LoadBalancer.**

**kubectl get svc -n prometheus. ( We can get External IP)**

**Login to Grafana**

**admin**

**prom-operator**

**Dashboard: For Kubernetes dashboard. —Import — 15661 — add.**

**For kube cluster monitor via prometheus : 3119**

**For Grafana : 6417**

**Login to Prometheus:**

………………………………………………………………………………………………………………………………………………

After giving Kubedeploy run pipeline.

Make sure you have Kubenetes