**SESSION - 82**

Static source code analysis

Open source vulnerability scanning

Dynamic application security testing

Static application security testing

Image scanning

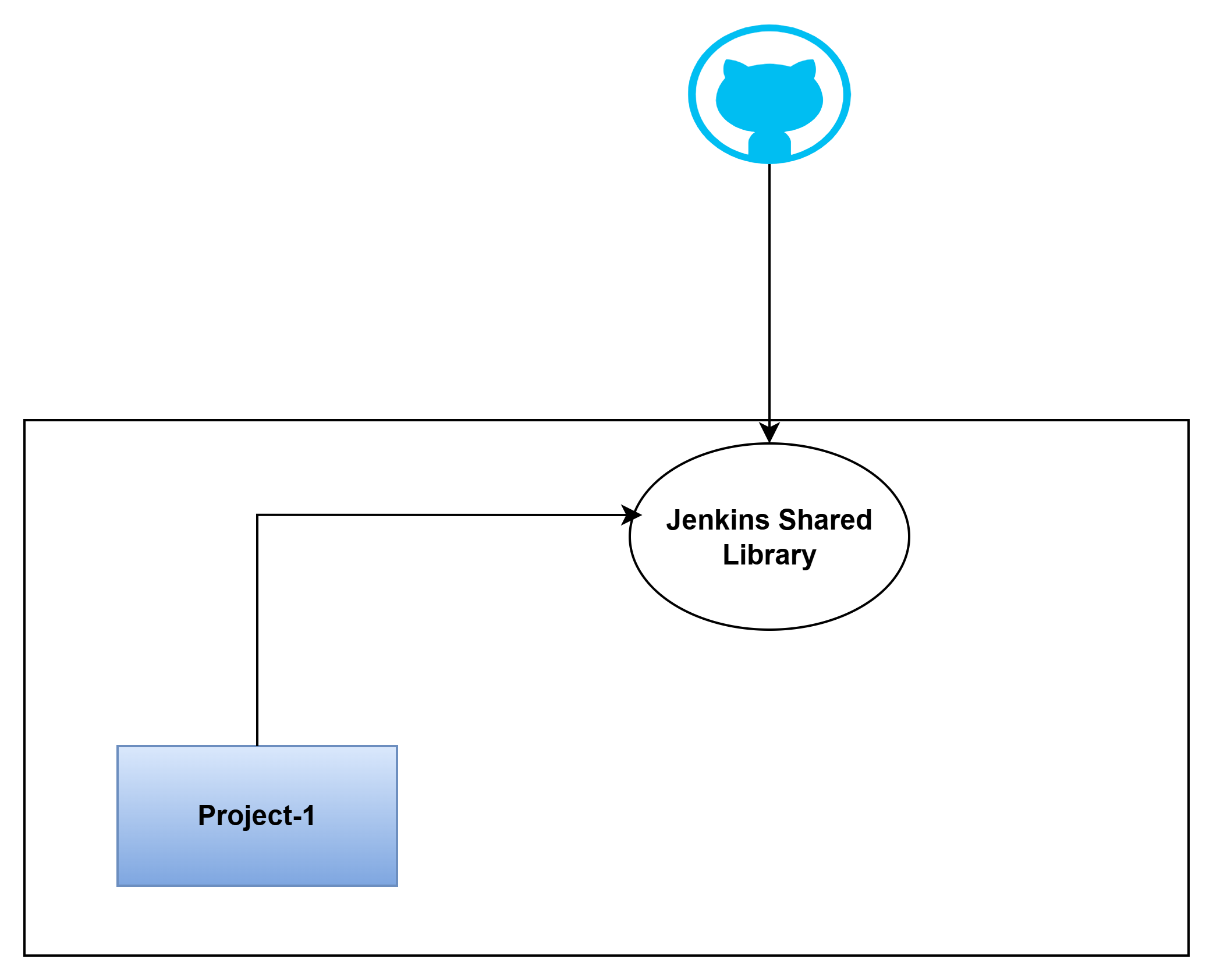
Shift left

Catalogue -> developer

Jenkinsfile -> DevOps Engineer

DRY -> Don't repeat yourself

Ansible Roles, Terraform modules, Shell common functions



**Too many components**

**===================**

each component will have devops engineer

they create their own pipelines

**Programming Language Deployment Platform**

**==================== ===================**

Java EKS

Java VM

Java PCF

--> create a jenkins-shared-library directory

groovy is java syntax, but we can use shell script

**vars/nodejsEKSPipeline.groovy**

def call(Map configMap){

pipeline {

agent {

label 'AGENT-1'

}

environment {

appVersion = ''

REGION = "us-east-1"

ACC\_ID = "160885265516"

PROJECT = configMap.get('project')

COMPONENT = configMap.get('component')

}

options {

timeout(time: 30, unit: 'MINUTES')

disableConcurrentBuilds()

}

parameters {

booleanParam(name: 'deploy', defaultValue: false, description: 'Toggle this value')

}

// Build

stages {

stage('Read package.json') {

steps {

script {

def packageJson = readJSON file: 'package.json'

appVersion = packageJson.version

echo "Package version: ${appVersion}"

}

}

}

stage('Install Dependencies') {

steps {

script {

sh """

npm install

"""

}

}

}

stage('Unit Testing') {

steps {

script {

sh """

echo "unit tests"

"""

}

}

}

/\* stage('Sonar Scan') {

environment {

scannerHome = tool 'sonar-7.2'

}

steps {

script {

// Sonar Server envrionment

withSonarQubeEnv(installationName: 'sonar-7.2') {

sh "${scannerHome}/bin/sonar-scanner"

}

}

}

} \*/

// Enable webhook in sonarqube server and wait for results

/\* stage("Quality Gate") {

steps {

timeout(time: 1, unit: 'HOURS') {

waitForQualityGate abortPipeline: true }

}

}

stage('Check Dependabot Alerts') {

environment {

GITHUB\_TOKEN = credentials('github-token')

}

steps {

script {

// Fetch alerts from GitHub

def response = sh(

script: """

curl -s -H "Accept: application/vnd.github+json" \

-H "Authorization: token ${GITHUB\_TOKEN}" \

https://api.github.com/repos/daws-84s/catalogue/dependabot/alerts

""",

returnStdout: true

).trim()

// Parse JSON

def json = readJSON text: response

// Filter alerts by severity

def criticalOrHigh = json.findAll { alert ->

def severity = alert?.security\_advisory?.severity?.toLowerCase()

def state = alert?.state?.toLowerCase()

return (state == "open" && (severity == "critical" || severity == "high"))

}

if (criticalOrHigh.size() > 0) {

error "❌ Found ${criticalOrHigh.size()} HIGH/CRITICAL Dependabot alerts. Failing pipeline!"

} else {

echo "✅ No HIGH/CRITICAL Dependabot alerts found."

}

}

}

}\*/

stage('Docker Build') {

steps {

script {

withAWS(credentials: 'aws-creds', region: 'us-east-1') {

sh """

aws ecr get-login-password --region ${REGION} | docker login --username AWS --password-stdin ${ACC\_ID}.dkr.ecr.us-east-1.amazonaws.com

docker build -t ${ACC\_ID}.dkr.ecr.us-east-1.amazonaws.com/${PROJECT}/${COMPONENT}:${appVersion} .

docker push ${ACC\_ID}.dkr.ecr.us-east-1.amazonaws.com/${PROJECT}/${COMPONENT}:${appVersion}

#aws ecr wait image-scan-complete --repository-name ${PROJECT}/${COMPONENT} --image-id imageTag=${appVersion} --region ${REGION}

"""

}

}

}

}

/\* stage('Check Scan Results') {

steps {

script {

withAWS(credentials: 'aws-creds', region: 'us-east-1') {

// Fetch scan findings

def findings = sh(

script: """

aws ecr describe-image-scan-findings \

--repository-name ${PROJECT}/${COMPONENT} \

--image-id imageTag=${appVersion} \

--region ${REGION} \

--output json

""",

returnStdout: true

).trim()

// Parse JSON

def json = readJSON text: findings

def highCritical = json.imageScanFindings.findings.findAll {

it.severity == "HIGH" || it.severity == "CRITICAL"

}

if (highCritical.size() > 0) {

echo "❌ Found ${highCritical.size()} HIGH/CRITICAL vulnerabilities!"

currentBuild.result = 'FAILURE'

error("Build failed due to vulnerabilities")

} else {

echo "✅ No HIGH/CRITICAL vulnerabilities found."

}

}

}

}

} \*/

stage('Trigger Deploy') {

when{

expression { params.deploy }

}

steps {

script {

//build job: 'catalogue-cd',

build job: "../${COMPONENT}-cd",

parameters: [

string(name: 'appVersion', value: "${appVersion}"),

string(name: 'deploy\_to', value: 'dev')

],

propagate: false, // even SG fails VPC will not be effected

wait: false // VPC will not wait for SG pipeline completion

}

}

}

}

post {

always {

echo 'I will always say Hello again!'

deleteDir()

}

success {

echo 'Hello Success'

}

failure {

echo 'Hello Failure'

}

}

}

}

**vars/samplePipeline.groovy**

def call(Map configMap){

pipeline {

agent {

label 'AGENT-1'

}

environment {

COURSE = 'jenkins'

greeting = configMap.get('greeting')

}

options {

timeout(time: 30, unit: 'MINUTES')

disableConcurrentBuilds()

}

parameters {

string(name: 'PERSON', defaultValue: 'Mr Jenkins', description: 'Who should I say hello to?')

text(name: 'BIOGRAPHY', defaultValue: '', description: 'Enter some information about the person')

booleanParam(name: 'TOGGLE', defaultValue: true, description: 'Toggle this value')

}

// Build

stages {

stage('Build') {

steps {

script{

sh """

echo "Hello Build"

sleep 10

env

echo "Hello ${params.PERSON}"

"""

}

}

}

stage('Test') {

steps {

script{

echo 'Testing..'

}

}

}

}

post {

always {

echo 'I will always say Hello again!'

deleteDir()

}

success {

echo 'Hello Success'

}

failure {

echo 'Hello Failure'

}

}

}

}

--> push the code

--> connect jenkins

**catalogue/Jenkinsfile**

@Library('jenkins-shared-library') \_

def configMap = [

project : "roboshop",

component: "catalogue"

]

if( ! env.BRANCH\_NAME.equalsIgnoreCase('main') ){ // if not equals to main

nodejsEKSPipeline(configMap) // by default it will call, call function inside this pipeline

}

else{

echo "Please proceed with PROD process"

}

--> Push the code

--> git checkout -b feature-1

**catalogue-cd/values-dev.yaml**

deployment:

imageURL: 160885265516.dkr.ecr.us-east-1.amazonaws.com/roboshop/catalogue

imageVersion: 1.0.0

hpa:

maxReplicas: 10

utilisation: 75

configMap:

MONGO\_URL: mongodb://mongodb:27017/catalogue

**catalogue-cd/templates/configmap.yaml**

apiVersion: v1

kind: ConfigMap

metadata:

name: catalogue

namespace: roboshop

labels:

component: catalogue

project: roboshop

tier: app

data:

MONGO: "true"

#MONGO\_URL: "mongodb://mongodb:27017/catalogue"

MONGO\_URL: {{ .Values.configMap.MONGO\_URL }}

**catalogue-cd/values-qa.yaml**

deployment:

imageURL: 315069654700.dkr.ecr.us-east-1.amazonaws.com/roboshop/catalogue

imageVersion: IMAGE\_VERSION

hpa:

maxReplicas: 10

utilisation: 75

configMap:

MONGO\_URL: mongodb://mongodb:27017/catalogue

**catalogue-cd/values-prod.yaml**

deployment:

imageURL: 315069654700.dkr.ecr.us-east-1.amazonaws.com/roboshop/catalogue

imageVersion: IMAGE\_VERSION

hpa:

maxReplicas: 10

utilisation: 75

configMap:

MONGO\_URL: mongodb://mongodb:27017/catalogue

--> catalogue is a folder

--> push the code

hello-world-nodejs

developers --> feature branch --> main

**Catalogue-cd/Jenkinsfile**

pipeline {

agent {

label 'AGENT-1'

}

environment {

appVersion = ''

REGION = "us-east-1"

ACC\_ID = "160885265516"

PROJECT = "roboshop"

COMPONENT = "catalogue"

}

options {

timeout(time: 30, unit: 'MINUTES')

disableConcurrentBuilds()

}

parameters {

string(name: 'appVersion', description: 'Image version of the application')

choice(name: 'deploy\_to', choices: ['dev', 'qa', 'prod'], description: 'Pick the Environment')

}

// Build

stages {

stage('Check Status'){

steps{

script{

withAWS(credentials: 'aws-creds', region: 'us-east-1') {

def deploymentStatus = sh(returnStdout: true, script: "kubectl rollout status deployment/catalogue --timeout=30s -n $PROJECT || echo FAILED").trim()

if (deploymentStatus.contains("successfully rolled out")) {

echo "Deployment is success"

} else {

sh """

helm rollback $COMPONENT -n $PROJECT

sleep 20

"""

def rollbackStatus = sh(returnStdout: true, script: "kubectl rollout status deployment/catalogue --timeout=30s -n $PROJECT || echo FAILED").trim()

if (rollbackStatus.contains("successfully rolled out")) {

error "Deployment is Failure, Rollback Success"

}

else{

error "Deployment is Failure, Rollback Failure. Application is not running"

}

}

}

}

}

}

stage('Deploy') {

steps {

script {

withAWS(credentials: 'aws-creds', region: 'us-east-1') {

sh """

aws eks update-kubeconfig --region $REGION --name "$PROJECT-${params.deploy\_to}"

kubectl get nodes

kubectl apply -f 01-namespace.yaml

sed -i "s/IMAGE\_VERSION/${params.appVersion}/g" values-${params.deploy\_to}.yaml

#helm upgrade --install $COMPONENT -f values-${params.deploy\_to}.yaml -n $PROJECT .

kubectl apply -f application.yaml

"""

}

}

}

}

// API Testing

stage('Functional Testing'){

when{

expression { params.deploy\_to = "dev" }

}

steps{

script{

echo "Run functional test cases"

}

}

}

// All components testing

stage('Integration Testing'){

when{

expression { params.deploy\_to = "qa" }

}

steps{

script{

echo "Run Integration test cases"

}

}

}

stage('PROD Deploy') {

when{

expression { params.deploy\_to = "prod" }

}

steps {

script {

withAWS(credentials: 'aws-creds', region: 'us-east-1') {

sh """

echo "get cr number"

echo "check with in the deployment window"

echo "is CR approved"

echo "trigger PROD deploy"

"""

}

}

}

}

}

post {

always {

echo 'I will always say Hello again!'

deleteDir()

}

success {

echo 'Hello Success'

}

failure {

echo 'Hello Failure'

}

}

}

multi branch pipelines

QA, SIT, UAT, PROD

build once and run anywhere --> build in DEV environment, promoting that build to all other environments with diff configuration

https://catalogue-dev.daws84s.site/api/catalogue/robo-123/id

**CR process**

**============**

change release process

DEV, QA/SIT/UAT/PERF/PRE-PROD, PROD

jira, change management tool

**ticket cr-1234**

**================**

project

component

date and time

rollback

change description

team lead - 1st level

team manager - 2nd level

testing manager - 3rd level

client - 4th level

dependencies of this application