DevSecOps Pipeline

```
pipeline {
    agent any

    tools {
        maven 'maven'
        dockerTool 'docker'
    }

    environment {
        GITHUB_CREDENTIALS_ID = 'git-token-2'
        DOCKER_CREDENTIALS_ID = 'docker-cred'
        SONARQUBE_URL = 'http://15.207.117.46:9000'
        SONARQUBE_CREDENTIALS_ID = 'sonarqube'
        TRIVY_BIN = '/usr/local/bin/trivy'
        AWS_EKS_CLUSTER_NAME = 'DevSecOps_Cluster_New'
        AWS_DEFAULT_REGION = 'ap-south-1'
        DOCKER_IMAGE_NAME = 'kushagraag/devsecops-image'
        DOCKER_IMAGE_TAG = 'latest'
        FILE_SYSTEM_REPORT = 'file_system_report.json'
        IMAGE_REPORT = 'image_report.json'
        EMAIL_RECIPIENTS = 'akushagra607@gmail.com'
        EMAIL_SUBJECT_SUCCESS = 'Jenkins Pipeline Success'
        EMAIL_SUBJECT_FAILURE = 'Jenkins Pipeline Failure'
    }

    stages {
        stage('Git Checkout') {
            steps {
                git credentialsId: "${GITHUB_CREDENTIALS_ID}", url:
'https://github.com/kushagra023/Sample-maven-webapp.git'
            }
        }

        stage('Trivy Scan - File System') {
            steps {
                script {
                    echo "Checking Trivy installation..."
                    sh "${TRIVY_BIN} --version"

                    echo "Scanning file system..."
                    sh "${TRIVY_BIN} fs --format json --output ${FILE_SYSTEM_REPORT} ."
                }
            }
        }
```

```
        stage('Build WAR File') {
            steps {
                script {
                    echo "Building WAR file..."
                    sh 'mvn clean install'
                }
            }
        }

        stage('Static Code Analysis - SonarQube') {
            steps {
                script {
                    withSonarQubeEnv('sonarqube') {
                        sh 'mvn clean verify sonar:sonar'
                    }
                }
            }
        }

        stage('OWASP Dependency Check') {
            steps {
                script {
                    echo "Running OWASP Dependency Check..."
                    sh 'mvn org.owasp:dependency-check-maven:check'
                }
            }
        }

        stage('Build Docker Image') {
            steps {
                script {
                    echo "Building Docker image
${DOCKER_IMAGE_NAME}:${DOCKER_IMAGE_TAG}..."
                    docker.withRegistry('https://index.docker.io/v1/',
"${DOCKER_CREDENTIALS_ID}") {
                        def customImage =
docker.build("${DOCKER_IMAGE_NAME}:${DOCKER_IMAGE_TAG}")
                    }
                }
            }
        }

        stage('Trivy Scan - Docker Image') {
            steps {
                script {
                    echo "Checking Trivy installation..."
                    sh "${TRIVY_BIN} --version"
```

```
                echo "Scanning Docker image
${DOCKER_IMAGE_NAME}:${DOCKER_IMAGE_TAG}..."
                sh "${TRIVY_BIN} image --format json --output ${IMAGE_REPORT}
${DOCKER_IMAGE_NAME}:${DOCKER_IMAGE_TAG}"
            }
        }
    }

    stage('Push Docker Image to DockerHub') {
        steps {
            script {
                echo "Pushing Docker image
${DOCKER_IMAGE_NAME}:${DOCKER_IMAGE_TAG} to DockerHub..."
                docker.withRegistry('https://index.docker.io/v1/',
"${DOCKER_CREDENTIALS_ID}") {
                    sh "docker push ${DOCKER_IMAGE_NAME}:${DOCKER_IMAGE_TAG}"
                }
            }
        }
    }

    stage('Deploy to Kubernetes') {
        steps {
            withCredentials([[$class: 'AmazonWebServicesCredentialsBinding', credentialsId:
'aws-cred']]){
                withCredentials([file(credentialsId: 'kubeconfig', variable: 'KUBECONFIG')]) {
                    script {
                        sh '''
                        kubectl --kubeconfig="${KUBECONFIG}" apply -f deployment.yaml
                        kubectl --kubeconfig="${KUBECONFIG}" apply -f service.yaml
                        '''
                    }
                }
            }
        }
    }
}

post {
    always {
        script {
            echo "Archiving artifacts..."
            archiveArtifacts artifacts: "${FILE_SYSTEM_REPORT},${IMAGE_REPORT}",
allowEmptyArchive: true
        }
        script {
```

```
            def emailSubject = currentBuild.result == 'SUCCESS' ?
"${EMAIL_SUBJECT_SUCCESS}" : "${EMAIL_SUBJECT_FAILURE}"
            def emailBody = """The pipeline executed ${currentBuild.result == 'SUCCESS' ?
'successfully' : 'with failure'}.

File System Scan Report: ${FILE_SYSTEM_REPORT}
Docker Image Scan Report: ${IMAGE_REPORT}"""

            emailext(
                to: "${EMAIL_RECIPIENTS}",
                subject: "${emailSubject}",
                body: emailBody,
                attachmentsPattern: "${FILE_SYSTEM_REPORT}, ${IMAGE_REPORT}"
            )
        }
        echo "Cleaning up workspace..."
        cleanWs()
    }
  }
}
```