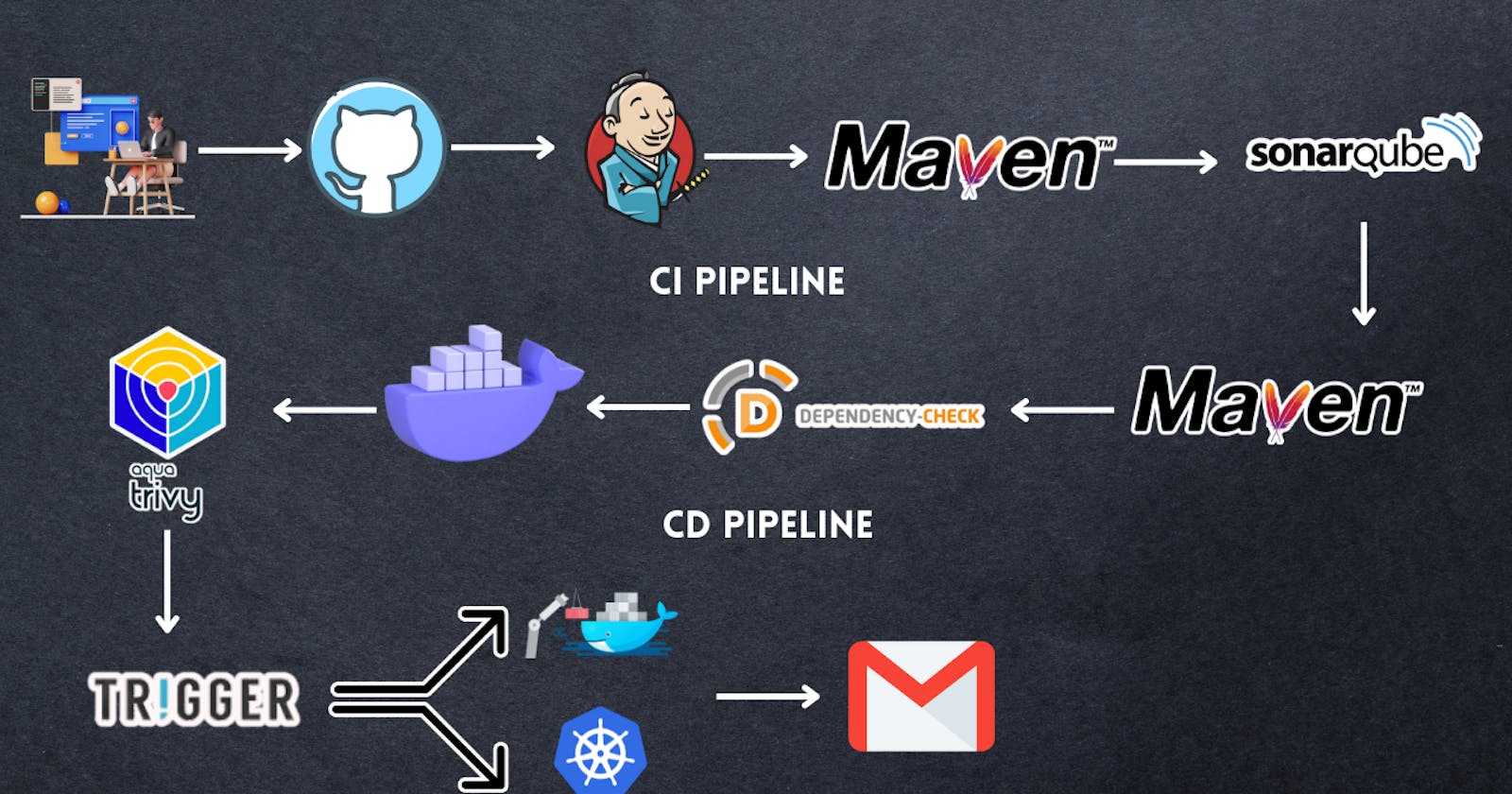
**Deployed a Java-Based JPetstore Application using Jenkins, Docker, and Kubernetes on AWS.**

**Work flow**



**Procedure:**

Step 1: Create an Ubuntu 22.04 T2 Large Instance in AWS.

Configure security group to allow HTTP and HTTPS traffic.

Step 2: Install Jenkins, Docker, and Trivy (Docker Image Scan) and also required plugins.

2A: Install Jenkins

* Connect to the instance.
* Create a script jenkins.sh to install Jenkins.
* Update Jenkins to use port 8090 instead of 8080.

2B: Install Docker

* Update the system and install Docker.
* Add the user to the Docker group.
* Start a Sonarqube container.

2C: Install Trivy

* Create a script trivy.sh to install Trivy.
* Add Trivy repository and install Trivy.

Step 3: Install Jenkins Plugins and Configure Tools

* Install Jenkins plugins: Eclipse Temurin Installer, SonarQube Scanner.
* Configure Java and Maven in Global Tool Configuration.
* Create a Jenkins job for the pipeline.

Step 4: Configure Sonar Server in Jenkins

* Configure SonarQube server in Manage Jenkins.
* Generate a token for SonarQube.
* Add SonarQube credentials to Jenkins.

Step 5: Install OWASP Dependency Check Plugin ( **Optional** )

* Install OWASP Dependency-Check plugin.
* Configure the plugin and add it to the pipeline.

Step 6: Docker Image Build and Push

* Install Docker-related plugins.
* Configure DockerHub credentials in Jenkins.
* Add Docker-related stages to the pipeline.

Step 7: Deploy Application using Docker

* Build and push the Docker image.
* Run the Docker container.

Step 8: Kubernetes Setup

* Set up Kubernetes on two Ubuntu 20.04 instances (Master and Worker).
* Install kubectl on Jenkins machine.
* Initialize Kubernetes on the master node and join worker nodes.

Step 9: Access the Application

* Access the application using the public IP of the worker node and port( where U get from (kubectl get all).

**Project Repo( Cloned )**: <https://github.com/Aj7Ay/jpetstore-6.git>

STEP1: Create an Ubuntu(22.04) T2 Large Instance

Launch an AWS T2 Large Instance. Use the image as Ubuntu. You can create a new key pair or use an existing one. Enable HTTP and HTTPS settings in the Security Group and open all ports.

Step 2 — Install Jenkins, Docker and Trivy

* 2A — To Install Jenkins

Connect to your console, and enter these commands to Install Jenkins.

* vi jenkins.sh
* #!/bin/bash
* sudo apt update -y
* #sudo apt upgrade -y
* wget -O - https://packages.adoptium.net/artifactory/api/gpg/key/public | tee /etc/apt/keyrings/adoptium.asc
* echo "deb [signed-by=/etc/apt/keyrings/adoptium.asc] https://packages.adoptium.net/artifactory/deb $(awk -F= '/^VERSION\_CODENAME/{print$2}' /etc/os-release) main" | tee /etc/apt/sources.list.d/adoptium.list
* sudo apt update -y
* sudo apt install temurin-17-jdk -y
* /usr/bin/java --version
* curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee \
* /usr/share/keyrings/jenkins-keyring.asc > /dev/null
* echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
* https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
* /etc/apt/sources.list.d/jenkins.list > /dev/null
* sudo apt-get update -y
* sudo apt-get install jenkins -y
* sudo systemctl start jenkins
* sudo systemctl status Jenkins
* sudo chmod 777 jenkins.sh
* ./jenkins.sh # this will installl Jenkins

Once Jenkins is installed, you will need to go to your AWS EC2 Security Group and open Inbound Port 8080, since Jenkins works on Port 8080.

But for this Application case, we are running Jenkins on another port. so change the port to 8090 using the below commands.

* sudo systemctl stop jenkins
* sudo systemctl status jenkins
* cd /etc/default
* sudo vi jenkins #chnage port HTTP\_PORT=8090 and save and exit
* cd /lib/systemd/system
* sudo vi jenkins.service #change Environments="Jenkins\_port=8090" save and exit
* sudo systemctl daemon-reload
* sudo systemctl restart jenkins
* sudo systemctl status jenkins

Now, grab your Public IP Address:

* <EC2 Public IP Address:8090>
* sudo cat /var/lib/jenkins/secrets/initialAdminPassword

Unlock Jenkins using an administrative password and install the suggested plugins. Jenkins will now get installed and install all the libraries. Create a user click on save and continue. Jenkins gets the Start Screen.

2B- Install Docker

* sudo apt-get update
* sudo apt-get install docker.io -y
* sudo usermod -aG docker $USER #my case is ubuntu
* newgrp docker
* sudo chmod 777 /var/run/docker.sock

After the docker installation, we create a sonarqube container (Remember adding 9000 ports in the security group)

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

Now the installed sonarqube is up and running.

Enter username and password, click on login and change password

**username admin**

**password admin**

* Asks for update password. Please update the password on sonar dasboard.

2c: Install Trivy (DOcker image Scan)

* vi trivy.sh
* sudo apt-get install wget apt-transport-https gnupg lsb-release -y
* wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | gpg --dearmor | sudo tee /usr/share/keyrings/trivy.gpg > /dev/null
* echo "deb [signed-by=/usr/share/keyrings/trivy.gpg] https://aquasecurity.github.io/trivy-repo/deb $(lsb\_release -sc) main" | sudo tee -a /etc/apt/sources.list.d/trivy.list
* sudo apt-get update
* sudo apt-get install trivy –y

Next, we will log in to Jenkins and start to configure our Pipeline in Jenkins.

Step 3 — Install Plugins like JDK, Sonarqube Scanner, Maven, OWASP Dependency Check

3A — Install Plugin

Goto Manage Jenkins →Plugins → Available Plugins →

Install below plugins

1 → Eclipse Temurin Installer (Install without restart)

2 → SonarQube Scanner (Install without restart)

3B — Configure Java and Maven in Global Tool Configuration

Goto Manage Jenkins → Tools → Install JDK(17) and Maven3(3.6.0) → Click on Apply and Save.

3C — Create a Job

Label it as PETSTORE, click on Pipeline and OK.

**Enter this in Pipeline Script as below, where we have plugins installed for jdk and maven. Also we define stages ( cleaning the workspace, checkingout the SCM, maven compilation and maven test)**

pipeline{

agent any

tools {

jdk 'jdk17'

maven 'maven3'

}

stages{

stage ('clean Workspace'){

steps{

cleanWs()

}

}

stage ('checkout scm') {

steps {

git 'https://github.com/Aj7Ay/jpetstore-6.git'

}

}

stage ('maven compile') {

steps {

sh 'mvn clean compile'

}

}

stage ('maven Test') {

steps {

sh 'mvn test'

}

}

}

}

Step 4 — Configure Sonar Server in Manage Jenkins

Grab the Public IP Address of your EC2 Instance, Sonarqube works on Port 9000, so <Public IP>:9000.

* Goto your Sonarqube Server. Click on Administration → Security → Users → Click on Tokens and Update Token → Give it a name → and click on Generate Token.
* Click on update token. Create a token with a name and generate.
* copy Token
* Goto Jenkins Dashboard → Manage Jenkins → Credentials → Add Secret Text. It should look like this
* Now, go to Dashboard → Manage Jenkins → System and name it as sonar-server and give your server URL with the created authentication token.
* Click apply and save.

The Configure System option is used in Jenkins to configure different server. Global Tool Configuration is used to configure different tools that we install using Plugins. We will install a sonar scanner in the tools.

The Configure System option is used in Jenkins to configure different server

Global Tool Configuration is used to configure different tools that we install using Plugins

We will install a sonar scanner in the tools.In the Sonarqube Dashboard add a quality gate also

* Administration--> Configuration-->Webhooks
* Click on create and add the details.
* #in url section of quality gate
* <http://jenkins-public-ip:8090>/sonarqube-webhook/

Let's go to our Pipeline and add Sonarqube Stage in our Pipeline Script.

#under tools section add this environment

environment {

SCANNER\_HOME=tool 'sonar-scanner'

}

# in stages add this

stage("Sonarqube Analysis "){

steps{

withSonarQubeEnv('sonar-server') {

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

-Dsonar.java.binaries=. \

-Dsonar.projectKey=Petshop '''

}

}

}

stage("quality gate"){

steps {

script {

waitForQualityGate abortPipeline: false, credentialsId: 'Sonar-token'

}

}

}

Click on Build now, you will see the stage in the pipeline..

To see the report, you can go to Sonarqube Server and go to Projects.

You can see the report has been generated and the status shows as passed. To see a detailed report, you can go to issues.

Step 5 — Install OWASP Dependency Check Plugins ( OPTIONAL )

* GotoDashboard → Manage Jenkins → Plugins → OWASP Dependency-Check. Click on it and install it without restart.

First, we configured the Plugin and next, we had to configure the Tool

* Goto Dashboard → Manage Jenkins → Tools →

Click on Apply and Save here.

Now go configure → Pipeline and add this stage to your pipeline and build.

stage ('Build war file'){

steps{

sh 'mvn clean install -DskipTests=true'

}

}

stage("OWASP Dependency Check"){

steps{

dependencyCheck additionalArguments: '--scan ./ --format XML ', odcInstallation: 'DP-Check'

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

}

}

* You will see that in status, a graph will also be generated and Vulnerabilities.

Step 6 — Docker Image Build and Push

We need to install the Docker tool in our system, Goto Dashboard → Manage Plugins → Available plugins → Search for Docker and install these plugins

Docker

Docker Commons

Docker Pipeline

Docker API

docker-build-step and click on install without restart.

* Now, goto Dashboard → Manage Jenkins → Tools →
* Add your DockerHub Username and Password under Global Credentials
* Add the below stage to Pipeline Script with a change in necessary credentials of yours.

stage ('Build and push to docker hub'){

steps{

script{

withDockerRegistry(credentialsId: 'docker', toolName: 'docker') {

sh "docker build -t petshop ."

sh "docker tag petshop sevenajay/petshop:latest"

sh "docker push sevenajay/petshop:latest"

}

}

}

}

stage("TRIVY"){

steps{

sh "trivy image sevenajay/petshop:latest > trivy.txt"

}

}

stage ('Deploy to container'){

steps{

sh 'docker run -d --name pet1 -p 8080:8080 sevenajay/petshop:latest'

}

}

You will see the output below, with a dependency trend and a image scanning done..

* U will have now a new image created in dockerhub .
* Give your <Ec2-public-ip:8080/jpetstore>
* you will get output… Do Check..

Step 8 — Kuberenetes Setup

* Connect your machines in AWS or putty.
* Take-Two Ubuntu 20.04 instances one for k8s master and the other one for worker.
* Install Kubectl on Jenkins machine also.
* Kubectl on Jenkins to be installed. pls follow the below script.
* sudo apt update
* sudo apt install curl
* curl -LO https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl
* sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
* kubectl version --client

we shall create both master node and worker node.

* sudo apt-get update
* sudo apt-get install -y docker.io
* sudo usermod –aG docker Ubuntu
* newgrp docker
* sudo chmod 777 /var/run/docker.sock
* sudo curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
* sudo tee /etc/apt/sources.list.d/kubernetes.list <<EOF
* deb https://apt.kubernetes.io/ kubernetes-xenial main
* EOF
* sudo apt-get update
* sudo apt-get install -y kubelet kubeadm kubectl
* sudo snap install kube-apiserver

Master Node:

* sudo kubeadm init --pod-network-cidr=10.244.0.0/16
* # in case your in root exit from it and run below commands
* mkdir -p $HOME/.kube
* sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
* sudo chown $(id -u):$(id -g) $HOME/.kube/config
* kubectl apply -f <https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml>

Worker node:

* sudo kubeadm join <master-node-ip>:<master-node-port> --token <token> --discovery-token-ca-cert-hash <hash>

Copy the config file to Jenkins master or the local file manager and save it

* copy it and save it in documents or another folder save it as secret-file.txt
* Install Kubernetes Plugin, Once it's installed successfully
* goto manage Jenkins --> manage credentials --> Click on Jenkins global --> add credentials

Configuring mail server in Jenkins ( Gmail )

* Install Email Extension Plugin in Jenkins
* Go to your Gmail and click on your profile
* Then click on Manage Your Google Account --> click on the security tab on the left side panel you will get this page.
* 2-step verification should be enabled.
* Search for the app in the search bar you will get app passwords like the below image.
* Click on other and provide your name and click on Generate and copy the password
* Once the plugin is installed in Jenkins, click on manage Jenkins --> configure system there under the E-mail Notification section configure the details.
* Click on Apply and save.
* Click on Manage Jenkins--> credentials and add your mail username and generated password
* This is to just verify the mail configuration
* Now under the Extended E-mail Notification section configure the details as required…

Final step to deploy on the Kubernetes cluster and email pipeline.

* stage('K8s'){
* steps{
* script{
* withKubeConfig(caCertificate: '', clusterName: '', contextName: '', credentialsId: 'k8s', namespace: '', restrictKubeConfigAccess: false, serverUrl: '') {
* sh 'kubectl apply -f deployment.yaml'
* }
* }
* }
* }

Give your email id here below..

* #post block after stages
* post {
* always {
* emailext attachLog: true,
* subject: "'${currentBuild.result}'",
* body: "Project: ${env.JOB\_NAME}<br/>" +
* "Build Number: ${env.BUILD\_NUMBER}<br/>" +
* "URL: ${env.BUILD\_URL}<br/>",
* to: 'postbox.aj99@gmail.com',
* attachmentsPattern: 'trivy.txt'
* }
* }

Now trigger the build and you can see all the stages are successfully built and the pipeline goes successful.

In the Kubernetes cluster give this command:

kubectl get all

kubectl get svc

you will have the pods running and through everything gets ready.

With the port, access the web browser.

STEP9:Access from a Web browser with

<public-ip-of-slave:your port>/jpetstore/

You will now land the web page JPetstore with your public IP and the port as kubectl given.

Also check your email, you will receive an email as success..

Step 10 : DONOT forget to terminate all the instances once done..

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Please refer for the complete pipeline here

pipeline{

agent any

tools {

jdk 'jdk17'

maven 'maven3'

}

environment {

SCANNER\_HOME=tool 'sonar-scanner'

}

stages{

stage ('clean Workspace'){

steps{

cleanWs()

}

}

stage ('checkout scm') {

steps {

git 'https://github.com/Aj7Ay/jpetstore-6.git'

}

}

stage ('maven compile') {

steps {

sh 'mvn clean compile'

}

}

stage ('maven Test') {

steps {

sh 'mvn test'

}

}

stage("Sonarqube Analysis "){

steps{

withSonarQubeEnv('sonar-server') {

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

-Dsonar.java.binaries=. \

-Dsonar.projectKey=Petshop '''

}

}

}

stage("quality gate"){

steps {

script {

waitForQualityGate abortPipeline: false, credentialsId: 'Sonar-token'

}

}

}

stage ('Build war file'){

steps{

sh 'mvn clean install -DskipTests=true'

}

}

stage("OWASP Dependency Check"){

steps{

dependencyCheck additionalArguments: '--scan ./ --format XML ', odcInstallation: 'DP-Check'

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

}

}

stage ('Build and push to docker hub'){

steps{

script{

withDockerRegistry(credentialsId: 'docker', toolName: 'docker') {

sh "docker build -t petshop ."

sh "docker tag petshop sevenajay/petshop:latest"

sh "docker push sevenajay/petshop:latest"

}

}

}

}

stage("TRIVY"){

steps{

sh "trivy image sevenajay/petshop:latest > trivy.txt"

}

}

stage ('Deploy to container'){

steps{

sh 'docker run -d --name pet1 -p 8080:8080 sevenajay/petshop:latest'

}

}

stage('K8s'){

steps{

script{

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

sh 'kubectl apply -f deployment.yaml'

}

}

}

}

}

post {

always {

emailext attachLog: true,

subject: "'${currentBuild.result}'",

body: "Project: ${env.JOB\_NAME}<br/>" +

"Build Number: ${env.BUILD\_NUMBER}<br/>" +

"URL: ${env.BUILD\_URL}<br/>",

to: 'postbox.aj99@gmail.com',

attachmentsPattern: 'trivy.txt'

}

}

}

Trigger code

CI-Petstore-pipeline

pipeline{

agent any

tools {

jdk 'jdk17'

maven 'maven3'

}

environment {

SCANNER\_HOME=tool 'sonar-scanner'

}

stages{

stage ('clean Workspace'){

steps{

cleanWs()

}

}

stage ('checkout scm') {

steps {

git 'https://github.com/Aj7Ay/jpetstore-6.git'

}

}

stage ('maven compile') {

steps {

sh 'mvn clean compile'

}

}

stage ('maven Test') {

steps {

sh 'mvn test'

}

}

stage("Sonarqube Analysis "){

steps{

withSonarQubeEnv('sonar-server') {

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

-Dsonar.java.binaries=. \

-Dsonar.projectKey=Petshop '''

}

}

}

stage("quality gate"){

steps {

script {

waitForQualityGate abortPipeline: false, credentialsId: 'Sonar-token'

}

}

}

stage ('Build war file'){

steps{

sh 'mvn clean install -DskipTests=true'

}

}

stage("OWASP Dependency Check"){

steps{

dependencyCheck additionalArguments: '--scan ./ --format XML ', odcInstallation: 'DP-Check'

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

}

}

stage ('Build and push to docker hub'){

steps{

script{

withDockerRegistry(credentialsId: 'docker', toolName: 'docker') {

sh "docker build -t petshop ."

sh "docker tag petshop sevenajay/petshop:latest"

sh "docker push sevenajay/petshop:latest"

}

}

}

}

stage("TRIVY"){

steps{

sh "trivy image sevenajay/petshop:latest > trivy.txt"

}

}

stage("Trigger deployment"){

steps{

// Trigger the deployment pipeline and wait for it to complete

build job: 'CD-petshop', wait: true

}

}

}

post {

always {

emailext attachLog: true,

subject: "'${currentBuild.result}'",

body: "Project: ${env.JOB\_NAME}<br/>" +

"Build Number: ${env.BUILD\_NUMBER}<br/>" +

"URL: ${env.BUILD\_URL}<br/>",

to: 'postbox.aj99@gmail.com',

attachmentsPattern: 'trivy.txt'

}

}

}

CD-petstore-pipeline

pipeline{

agent any

stages{

stage ('clean Workspace'){

steps{

cleanWs()

}

}

stage ('checkout scm') {

steps {

git 'https://github.com/Aj7Ay/jpetstore-6.git'

}

}

stage ('Deploy to container'){

steps{

sh 'docker run -d --name pet1 -p 8080:8080 sevenajay/petshop:latest'

}

}

stage('K8s'){

steps{

script{

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

sh 'kubectl apply -f deployment.yaml'

}

}

}

}

}

post {

always {

emailext attachLog: true,

subject: "'${currentBuild.result}'",

body: "Project: ${env.JOB\_NAME}<br/>" +

"Build Number: ${env.BUILD\_NUMBER}<br/>" +

"URL: ${env.BUILD\_URL}<br/>",

to: 'postbox.aj99@gmail.com',

attachmentsPattern: 'trivy.txt'

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*