**Jenkins Pipeline을 Multiple Agents 방식으로 구성하기**

[참고] <https://www.jenkins.io/doc/book/pipeline/jenkinsfile/>

* **목적**

AKS(Azure Kubernetes Service) Cluster에서 /var/run/docker.sock 사용이 차단되어 있습니다. Cluster에서 동적으로 구현하는 Docker 기능을 사용 할 수 없어 관련 부분은 정적으로 구현하여야 합니다. Pipeline의 Stage 별로 별도의 Agents를 정의하여 구현합니다.

[참고] <https://docs.microsoft.com/ko-kr/azure/aks/cluster-configuration>

텍스트이(가) 표시된 사진

자동 생성된 설명

* **구현**
* 기존 Single Agent 방식의 Jenkins Pipeline Code

pipeline 범위의 agent를 정의하여 사용한다.

pipeline {

agent {

kubernetes {

label 'petclinic-cd'

yamlFile 'jenkins-agent-pod.yaml'

}

}

이하 생략

* Multiple Agents 방식의 Jenkins Pipeline Code

stage 범위의 agent를 각각 정의하여 사용한다.

pipeline {

agent none

생략

stages {

stage('Build') {

agent {

kubernetes {

label 'petclinic-cd'

yamlFile 'jenkins-agent-pod.yaml'

}

}

steps {

container('maven') {

sh 'mvn clean compile'

}

}

}

이하 생략

* [참고] Multiple Agents 방식의 Jenkins Pipeline Code - Full Source Code

pipeline {

agent none

environment {

REGISTRY\_URL = 'https://kktdevops.koreacentral.cloudapp.azure.com'

REGISTRY\_CREDENTIALS = 'credentials\_harbor'

APP\_IMAGE = null

IMAGE\_REPO = 'repo-spring-petclinic-rest'

IMAGE\_NAME = 'spring-petclinic-rest'

IMAGE\_TAG = "test\_${BUILD\_NUMBER}"

APP\_URL='http://azure-helm-petclinic.koreacentral.cloudapp.azure.com/petclinic'

APP\_PORT=80

PerfURL='azure-helm-petclinic.koreacentral.cloudapp.azure.com'

ArgoURL='20.200.241.185'

argocdAppPrefix='azure-helm-petclinic'

appWaitTimeout = 60

}

stages {

stage('Build') {

agent {

kubernetes {

label 'petclinic-cd'

yamlFile 'jenkins-agent-pod.yaml'

}

}

steps {

container('maven') {

sh 'mvn clean compile'

}

}

}

stage('Unit Test') {

agent {

kubernetes {

label 'petclinic-cd'

yamlFile 'jenkins-agent-pod.yaml'

}

}

steps {

container('maven') {

sh 'mvn test'

}

}

post {

always {

junit 'target/surefire-reports/\*.xml'

step([ $class: 'JacocoPublisher' ])

}

}

}

stage('Static Code Analysis') {

agent any

steps {

configFileProvider([configFile(fileId: 'maven-settings', variable: 'MAVEN\_SETTINGS')]) {

sh './mvnw sonar:sonar -s $MAVEN\_SETTINGS'

}

}

}

/\*

stage('Static Code Analysis') {

agent {

kubernetes {

label 'petclinic-cd'

yamlFile 'jenkins-agent-pod.yaml'

}

}

steps {

configFileProvider([configFile(fileId: 'maven-settings', variable: 'MAVEN\_SETTINGS')]) {

container('maven') {

sh 'mvn sonar:sonar -s $MAVEN\_SETTINGS'

}

}

}

}

\*/

stage('Package') {

agent {

kubernetes {

label 'petclinic-cd'

yamlFile 'jenkins-agent-pod.yaml'

}

}

steps {

container('maven') {

sh 'mvn clean package -DskipTests'

archiveArtifacts artifacts: 'target/\*.jar', fingerprint: true

}

}

}

stage('Build Docker image') {

agent any

steps {

script {

APP\_IMAGE = docker.build("${IMAGE\_REPO}/${IMAGE\_NAME}:${IMAGE\_TAG}")

}

}

}

stage('Push Docker image') {

agent any

steps {

script {

docker.withRegistry(REGISTRY\_URL, REGISTRY\_CREDENTIALS) {

APP\_IMAGE.push()

APP\_IMAGE.push('latest')

}

}

}

}

stage('Update manifest') {

agent any

steps {

sh """

git config --global user.name 'skccdevops03'

git config --global user.email 'skcc.devops03@sk.com'

git config --global credential.helper cache

git config --global push.default simple

"""

git url: 'https://github.com/skccdevops03/azure-helm-petclinic.git', credentialsId: 'credentials\_git', branch: 'main'

sh """

sed -i 's/tag:.\*/tag: "${IMAGE\_TAG}"/g' values.yaml

git add values.yaml

git commit -m 'Update Docker image tag: ${IMAGE\_TAG}'

git push origin main

"""

}

}

stage('Argo'){

agent {

kubernetes {

label 'petclinic-cd'

yamlFile 'jenkins-agent-pod.yaml'

}

}

steps {

withCredentials([usernamePassword(credentialsId: 'credentials\_argo', usernameVariable: 'ARGOCD\_USER', passwordVariable: 'ARGOCD\_PWD')]) {

container('argocd') {

sh """

yes | argocd login --insecure ${ArgoURL} --username ${ARGOCD\_USER} --password ${ARGOCD\_PWD}

argocd app sync ${argocdAppPrefix}

argocd app wait ${argocdAppPrefix} --timeout ${appWaitTimeout}

argocd logout ${ArgoURL}

sleep 10

"""

}

}

}

}

stage('PerforfmanceTest') {

agent {

kubernetes {

label 'petclinic-cd'

yamlFile 'jenkins-agent-pod.yaml'

}

}

steps {

withCredentials([usernamePassword(credentialsId: 'credentials\_git', usernameVariable: 'GIT\_USER', passwordVariable: 'GIT\_PWD')]) {

container('jmeter') {

sh """

git checkout origin/master

git config remote.origin.url https://${GIT\_USER}:${GIT\_PWD}@github.com/skccdevops03/spring-petclinic-rest.git

git config --global user.email 'skcc.devops03@sk.com'

git config --global user.name '${GIT\_USER}'

"""

sh """

JVM\_ARGS="-Xms1G -Xmx1G" && export JVM\_ARGS && /usr/local/jmeter/apache-jmeter-5.4.1/bin/jmeter.sh \

-n -f -t PerformanceTest/TS01\_TC01\_AWSPipeline.jmx -Jurl=${PerfURL} \

-l PerformanceTest/TestResult/Result\_${BUILD\_NUMBER}.jtl \

-e -o PerformanceTest/TestResult/Result\_html\_${BUILD\_NUMBER}

mv jmeter.log PerformanceTest/TestResult/Result\_html\_${BUILD\_NUMBER}/jmeter.log

"""

sh """

git add . && git commit -am 'Publish Jmeter result' && git push origin HEAD:master

"""

}

}

}

post {

always {

perfReport 'PerformanceTest/TestResult/Result\_${BUILD\_NUMBER}.jtl'

}

}

}

stage('API Test') {

agent {

kubernetes {

label 'petclinic-cd'

yamlFile 'jenkins-agent-pod.yaml'

}

}

steps {

container('newman') {

sh """

newman run api\_test.json \

--env-var 'baseUrl=${APP\_URL}' --env-var 'petTypeId=""'\

--reporters cli,junit \

--reporter-junit-export 'petclinic-report.xml'

"""

}

}

post {

always {

junit 'petclinic-report.xml'

}

}

}

} /\* stages \*/

} /\* pipeline \*/