Lab. Using Jenkins with Java

[범례]
※SpringBootRestfulMembership.zip -> 교육 중 학생들에게 배포하고자 하는 Spring Boot Code

1. Git 작업
1)압축 풀기 SpringBootRestfulMembership.zip
2)STS에서 Import
3)Local Mysql에서 CRUD 테스트
4)Github에 Repository 생성하기
-Repository name : springboot-restful-memebership
-Description : Spring Boot의 Restful Membership
-Public
-README.md check
-Choose a license check > MIT

5)STS와 연동하기
-STS의 Git Perspective
-In Git Repositories view
-Clone Git Repository
--URI : https://github.com/gitinstructor/springboot-restful-membership.git
--Authentication
---User : gitinstructor
---Password :
--gitinstructor > Settings > Developer settings > Personal access tokens
--Generate new token
--Next
--main check > Next > Finish
-In Spring Perspective
--In Package Explorer View
--Project > right-click > Team > Share Project
--Configure Git Repository
---Repository : 목록에서 선택 > Finish
--Commit 하기
--Project > right-click > Team > Commit
--All file add to index > Commit Message : "Create Project" > Commit and Push > Close
--Github에서 확인

2. AWS에서
1)EC2 생성
-Name : lab-ec2-db-xx
--Instance type : t2.micro
--보안그룹 : launch-sg-00
---Port : 22, 3306, 8080, 80
-Name : lab-ec2-was-xx
-Instance type : t2.medium
--보안그룹 : launch-sg-00

2)Xshell 연결
3)MySQL 설치하기
4)MySQL 설치후, 작업
-In Command,
>mysql -h {{AWS EC2 MySQL Address}} -u root -p
-test database 생성
mysql> create database test character set utf8;
-scott 계정 생성
mysql> create user 'scott'@'%' identified by 'tiger';
mysql> grant all privileges on test.* to 'scott'@'%';
mysql> flush privileges;
-HeidiSQL에서 연결 테스트

5)STS에서 AWS Mysql 연결테스트
-application.properties 수정
spring.datasource.url=jdbc:log4jdbc:mysql://{{AWS EC2 MySQL Address}}:3306/test
spring.datasource.hikari.driver-class-name=net.sf.log4jdbc.sql.jdbcapi.DriverSpy
spring.datasource.hikari.jdbc-url=jdbc:log4jdbc:mysql://{{AWS EC2 MySQL Address}}:3306/test
spring.datasource.hikari.username=scott
spring.datasource.hikari.password=tiger
-add > commit -m "Modify application.properties file." > Command & Push
-AWS와 연결된 HeidiSQL에서 membership.sql 실행할 것
-STS에서 Web Application 실행 및 테스트
--Tomcat 환경설정 변경
--port : 80
--Web Modules : /
--Maven Update
--Pom.xml > Maven Install

3. EC2 WAS에 Tomcat 설치하기
1)openjdk 설치하기
$ sudo apt update
$ sudo apt install openjdk-11-jdk
$ java -version

openjdk version "11.0.13" 2021-10-19
OpenJDK Runtime Environment (build 11.0.13+8-Ubuntu-0ubuntu1.20.04)
OpenJDK 64-Bit Server VM (build 11.0.13+8-Ubuntu-0ubuntu1.20.04, mixed mode, sharing)

2)Tomcat 설치
$ wget https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.56/bin/apache-tomcat-9.0.56.tar.gz -P /tmp
$ sudo mkdir /opt/tomcat
$ sudo tar -xf /tmp/apache-tomcat-9.0.56.tar.gz -C /opt/tomcat/
$ sudo ln -s /opt/tomcat/apache-tomcat-9.0.56 /opt/tomcat/latest

$ sudo nano /etc/systemd/system/tomcat.service
[Unit]
Description=Tomcat 9 servlet container
After=network.target

[Service]
Type=forking

Environment="JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64"
Environment="JAVA_OPTS=-Djava.security.egd=file:///dev/urandom -Djava.awt.headless=true"

Environment="CATALINA_BASE=/opt/tomcat/latest"
Environment="CATALINA_HOME=/opt/tomcat/latest"
Environment="CATALINA_PID=/opt/tomcat/latest/temp/tomcat.pid"
Environment="CATALINA_OPTS=-Xms512M -Xmx1024M -server -XX:+UseParallelGC"

ExecStart=/opt/tomcat/latest/bin/startup.sh
ExecStop=/opt/tomcat/latest/bin/shutdown.sh

[Install]
WantedBy=multi-user.target

$ sudo systemctl daemon-reload
$ sudo -i
# nano /opt/tomcat/latest/conf/server.xml
Port를 80으로 변경
#exit
$ sudo systemctl enable --now tomcat
$ sudo systemctl status tomcat
-EC2 보안그룹에서 port 80 오픈
-브라우저에서 확인
http://{{ AWS EC2 Tomcat Domain Name}}

4. 필요한 Plugin 설치
1)Jenkins 관리 > System Configuration > 플러그인 관리 > 설치 가능 탭
-"deploy to container plugin"으로 검색
-[deploy to container plugin] 체크하고 [Install without restart] click
2)설치 페이지에서 [성공] 확인
3)다시 Jenkins 관리 > Tools and Actions > Reload Configuration from Disk > 확실합니까? > 확인

5. 빌드 후 조치
1)maven을 통해 빌드 후 패키징됐다면 패키징된 .war 파일을 톰캣이 구동중인 원격 서버에 배포되어야한다. 아래와 같이 [빌드 후 조치] > [Deploy war/ear to a container]를 선택한다.
-해당 프로젝트 > 구성 > 빌드 후 조치 > 빌드 후 조치 추가 > Deploy war/ear to a container 선택

2)WAR/EAR files
-실제 빌드시 /var/lib/jenkins/workspace 디렉토리에 .war 파일이 생성되는데 job 실행시 해당 .war를 가져올 수 있도록 경로를 입력한다.
-**/*.war

3)Context path
-배포시 사용할 컨텍스트를 지정한다.
-/

5)Containers
-Tomcat 9.x Remote 선택
-Tomcat URL :
-배포되는 원격 서버 URL을 입력한다. http://{host}:{tomcat port}
http://{{ AWS EC2 Tomcat Domain Name}}

-Credentials : jenkins 서버가 원격서버에 배포할 수 있게 원격서버측에서 jenkins 서버 접근을 허용해줘야 한다.
1)[Jenkins 서버 접속 허용]
-톰캣이 구축되어있는 원격서버에서 아래 경로의 파일 내용을 수정한다.
-/opt/tomcat/latest/webapps/manager/META-INF/context.xml
<Valve className="org.apache.catalina.valves.RemoteAddrValve"
allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1|{Jenkins Container's IP}" />

2)[톰캣 원격 서버 접속 계정 설정]
-톰캣이 구축되어있는 원격서버에서 아래 경로의 파일 내용을 수정하여 jenkins 서버에서 접속시 사용되는 username, password를 설정해준다.
-{tomcat path}/conf/tomcat-users.xml
<role rolename="manager-script" />
<user username="admin" password="javatomcat" roles="manager-script" />

3)다시 jenkins 설정으로 돌아와 Containers > Credentials > Add >Jenkins
-Kind : Username with password
-Username : admin
-Password : javatomcat
-Add click

-Credentials : admin/*****로 선택

6)저장 클릭
7)Build Now 클릭 후 Build Success 확인하고 웹 브라우저에서 확인할 것

Task. Pipeline 만들기
1. 새로운 Item
1)Enter an item name : pipeline_demo
2)Pipeline 선택 후 OK

2. Pipeline tab
1)Definition : Pipeline script 선택
2)[Use Groovy Sandbox] uncheck
3)Script에 다음과 같이 코딩

```
pipeline {
    agent any

    tools {
        maven "maven"
    }

    stages {
        stage("git Pull") {
            steps {

            }
        }
    }
}
```

4)git Pull Stage
-[Pipeline Syntax] click
-Steps
--Sample Step > git: Git 선택
--git
--Repository URL : https://github.com/gitinstructor/demo.git
--Branch : main
--Credentials : gitinstructor/**** 선택
--두개 체크박스 모두 uncheck
--[Generate Pipeline Script] 클릭
--아래 코드를 steps 사이에 넣고 저장

```
git branch: 'main', changelog: false, credentialsId: 'access-token1', poll: false, url:
'https://github.com/gitinstructor/demo.git'
```

-Build Now로 테스트

5)Build Stage

```
stage('Build') {
    steps {
        sh "mvn -Dmaven.test.failure.ignore=true -N -f SpringBootRestfulMembership/pom.xml clean package"
    }
}
```

-저장 후 Build Now로 테스트

6)Deploy Stage
-[Pipeline Syntax] click
-Steps
--Sample Step > deploy: Deploy war/ear to a container
--deploy
---WAR/EAR files : **/*.war
---Context path : /
---Containers
Tomcat 9.x Remote
Tomcat URL : {{ AWS EC2 Domain Name}}
Credentials : admin/*****로 선택
-[Deploy on failure] uncheck
-[Generate Pipeline Script] 클릭

```
deploy adapters: [tomcat9(credentialsId: '844eae6f-190e-4d14-a7cb-f34e9dcf35d1', path: '', url: '{{ AWS EC2 Domain
Name}}')], contextPath: '/', onFailure: false, war: '**/*.war'
```

7)Restart Stage
-[Pipeline Syntax] click
-Steps
--Sample Step > sh: Shell Script 선택
--Shell Script
다음 2줄을 넣는다.
여기서 admin은 아이디, javatomcat은 패스워드

```
curl -u admin:javatomcat {{ AWS EC2 Domain Name}}/host-manager/text/stop
```

```
            curl -u admin:javatomcat h{{ AWS EC2 Domain Name}}/host-manager/text/start
        --[Generate Pipeline Script] 클릭
            sh '''curl -u admin:javatomcat {{ AWS EC2 Domain Name}}/host-manager/text/stop
            curl -u admin:javatomcat {{ AWS EC2 Domain Name}}/host-manager/text/start'''

8)전체 코드
    pipeline {
        agent any

        tools {
            maven "maven"
        }

        stages {
            stage("git Pull") {
                steps {
                    git branch: 'main', changelog: false, credentialsId: 'access-token1', poll: false, url:
                    'https://github.com/gitinstructor/demo.git'
                }
            }

            stage('Build') {
                steps {
                    sh "mvn -Dmaven.test.failure.ignore=true -N -f SpringBootRestfulMembership/pom.xml clean package"
                }
            }

            stage("Deploy") {
                steps {
                    deploy adapters: [tomcat9(credentialsId: '844eae6f-190e-4d14-a7cb-f34e9dcf35d1', path: '', url: '{{ AWS EC2
                    Domain Name}}')], contextPath: '/', onFailure: false, war: '**/*.war'
                }
            }

            stage("RESTART") {
                steps {
                    sh '''curl -u admin:javatomcat {{ AWS EC2 Domain Name}}/host-manager/text/stop
    curl -u admin:javatomcat {{ AWS EC2 Domain Name}}/host-manager/text/start'''
                }
            }
        }
    }

9)저장 후 Build Now 클릭
10)테스트
    -src/main/webapp/view.html을 /static/으로 이동 후 commit
    -index.html에 image 추가
```