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

1. Jenkins?
   1)지속적으로 통합 서비스(CI)를 제공해 주는 툴.
   2)원래 허드슨 프로젝트로 개발.
   3)Continuous Integration(CI) Tool라고도 부른다.
   4)역할은 Github 혹은 Bitbucket에 올라가있는 Source를 Build해주고, 해당 Source를 Compile 해주어서
   5)오류를 감지하고, 테스트 자동화와 맞물려 사용이 가능.
   6)개발 서버나 운영 서버에 배포까지 해주는 절차 가능.
   7)500여가지의 Plugin을 쉽게 install 할 수 있는 환경들을 제공.

2. Jenkins의 장점
   1)Java뿐만이 아니라 다양한 환경에서의 build/test 환경을 맞출 수 있다.
   2)다양한 언어들을 지원하고, 설치해서 해당 Compile 환경들을 만들 수 있다.
   3)정적 코드 분석를 통해서 코딩 규약을 어겼는지를 체크할 수 있다.
   4)기본적으로는 다양한 테스트 환경에 대하여 배포작업을 연결할 수 있다.
   5)Build time을 비롯해서, 실행 시간의 변화등 성능 변화 또한 감지할 수 있다.

3. 설치 방법
   1)Server에 직접 설치(Docker 등 활용)
   2)Local host에 설치(개인 컴퓨터)

---
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

Task1. Localhost에 Jenkins 설치하기
1. Jenkins Local Host에 설치하기
1)https://www.jenkins.io/
2)JDK 설치, Git 설치, Maven 설치
3)Port : 8080
4)services.msc > jenkins service Start 확인
5)http://localhost:8080
6)administrator password --> C:\ProgramData\Jenkins\.jenkins\secrets\initialAdminPassword
7)Customize Jenkins -> Install suggested plugins

2. Jenkins Global Tool Configuration
1)설치 후 좌측 메뉴의 Jenkins 관리
2)System Configuration > Global Tool Configuration 클릭
3)Maven Configuration
-Default settings provider
--Settings file in filesystem 선택
File path : C:\Program Files\apache-maven-3.8.3\conf\settings.xml
-Default global settings provide
----Settings fi,le in filesystem 선택
File path : C:\Program Files\apache-maven-3.8.3\conf\settings.xml
-Apply Click

4)JDK > JDK installations > Add JDK
-Name : jdk11
-Install automatically uncheck
-JAVA_HOME : C:\Program Files\Java\jdk-11.0.12
-Apply click

5)Git
-Name : Default
-Path to Git executable : C:\Program Files\Git\bin\git.exe
-Apply click

6)Maven
-Maven installations > Add Maven
--Name : maven
--Install automatically uncheck
--MAVEN_HOME : C:\Program Files\apache-maven-3.8.3
-Apply click
6)Save

3. 플러그인 관리
1)Jenkins 관리
2)System Configuration > 플러그인 관리
3)설치된 플러그인 목록 탭
-git으로 검색하여 Git plugin과 GitHub plugin이 있는지 확인
4)
-설치 가능 탭으로 이동
-"deploy to container plugin"으로 검색
-[deploy to container plugin] 체크하고 [Install without restart] click

4. Jenkins job이 GitHub의 Push Event 받기
1)Github plugin은 jenkins가 Github의 Webhook을 받을 수 있도록 수동 모드와 자동 모드를 제공한다.
2)수동 모드는 사용자가 Git Repository setting에서 Hook & Services를 직접 등록해 사용하는 것이고,
3)자동 모드는 사용자의 GitHub OAuth 토큰을 이용해 Jenkins가 자동으로 Hooks & Service를 등록한다.
4)자동으로 Webhook이 만들어지지 않는 경우 수동으로 Webhook을 만든다.
5)payload URL : Jenkins 주소/github-webhook/
6)GitHub OAuth 등록
-GitHub에서 Settings 선택
-Settings > Developer settings > Personal access tokens > Generate new token
-Note : jenkins_access_token
-Expiration : 30 days
-Select scopes
--repo
--admin:repo_hook
--delete_repo
--{{GitHub에서 발급받은 personal_access_token}}

7)Jenkins > Jenkins 관리 > 시스템 설정
-GitHub
-GitHub Servers > Add GitHub Server
-GitHub Server
--Name : My GitHub Server
--API URL : 그대로
--Credentials
--Add > Jenkins
---Domain : Global crredentials
---Kind : Secret text 선택
---Scope : Global(Jenkins...)
---Secret : {{GitHub에서 발급받은 personal_access_token}}
---ID : accss-token
---Add Click
--[Test connection] 클릭해서 연결 테스트
---Credentials verified for user gitinstructor, rate limit: 4999
-저장 클릭

5. 새로운 Item 생성하기
1)좌측 메뉴에서 새로운 Item 클릭
2)Enter an item name : deployment_demo
3)Freestyle project 클릭 > OK 클릭
4)GitHub project check
Project url : https://github.com/gitinstructor/springboot-restful-membership/
5)소스 코드 관리 > Git 선택
-Repository URL : https://github.com/gitinstructor/springboot-restful-membership.git
-Credentials > Add > Jenkins
--Domain : Global crredentials

--Kind : SSH Usename with private key
--ID : ssh-key
--Username : ssh-key
--Private Key > Enter directly check
---C:\Users\devex\.ssh\id_rsa의 내용을 에디터로 열어서 내용 복사 후
---[Add] 버튼 클릭하고 그 아래에 붙여넣는다.
--Add click
-방금 생성한 ssh-key에 맞춘다.
6)Branches to build
-Branch Specifier
--*/main
7)저장 버튼 클릭
8)Jenkins 관리 > 시스템 설정
-Jenkins Location
-Jenkins URL : http://192.168.0.14:8080/
-[저장] 클릭

9)GitHub에 Deploy Key 등록하기
-GitHub의 해당 Repository의 Settings > Deploy keys
-Add deploy key click
-Title : jenkins_deploy_key
-Key
--C:\Users\devex\.ssh\id_rsa.pub의 내용을 에디터로 열어서 그 내용을 복사해서 붙여 넣는다.
-Add key click

10)GitHub의 해당 Repository의 Settings
-Webhooks가 자동으로 등록되어 있는지 확인
-만일 자동등록이 안되어 있으면 수동으로 해야 함.
-[Add webhook] click
--Payload URL : http://192.168.0.14:8080/github-webhook/
-Content type : application/json
-[Add webhook] click

11)Jenkins > Dashboard > item 이름 > 구성 > 빌드 유발
-[GitHub hook trigger for GITScm polling] check
-Build section > Add build step 클릭
--Invoke top-level Maven targets 선택
--Maven Version : maven
--Goals : clean package
--[고급] 클릭
---POM : SpringBootRestfulMembership/pom.xml
-[저장] 클릭
12)Build Now click

Task2. Docker Container로 설치하기

1. AWS에 Docker 설치
1)$ sudo docker pull jenkins/jenkins
2)$ sudo docker run -dp 8080:8080 --name jenkins -v /home/jenkins:/var/jenkins_home -v /var/run/docker.sock:/var/run/docker.sock -u root jenkins/jenkins

2. 필수 설치
1)JDK 설치 확인
$ which java
--/opt/java/openjdk/bin/java
2)Git 설치 확인
$ which git
--/usr/bin/git

3)Maven 설치
$ apt update
$ apt install maven
$ mvn -version
Maven home: /usr/share/maven
Java version: 11.0.13, vendor: Eclipse Adoptium, runtime: /opt/java/openjdk

Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "5.11.0-1022-aws", arch: "amd64", family: "unix"
$ which mvn
/usr/bin/mvn

3. http://{{AWS EC2 Jenkins IP}}:8080

4. Jenkins Global Tool Configuration
1)설치 후 좌측 메뉴의 Jenkins 관리
2)System Configuration > Global Tool Configuration 클릭
3)Maven Configuration
-기본값 그대로

4)JDK > JDK installations > Add JDK
-Name : jdk11
-Install automatically uncheck
-JAVA_HOME : /opt/java/openjdk
-Apply click

5)Git
-Name : Default
-Path to Git executable : /usr/bin/git
-Apply click

6)Maven
-Maven installations > Add Maven
--Name : maven
--Install automatically uncheck
--MAVEN_HOME : /usr/share/maven
-Apply click
7)Save

5. 새로운 Item 생성하기
1)좌측 메뉴에서 [새로운 Item] 클릭
2)[Enter an item name] : demo
3)[Freestyle project] 클릭 > [OK] 클릭

4)소스 코드 관리 > Git 선택
-Repository URL : https://github.com/gitinstructor/springboot-restful-membership.git
-Credentials > Add > Jenkins
--Domain : Global credentials
--Kind : Username with password
--Scope : Global(Jenkins...)
--Username : {{ Github ID}}
--Password : {{GitHub에서 발급받은 Personal access token}}
--ID : accss-token
--Add click
-방금 생성한 gitinstructor/****에 맞춘다.

5)Branches to build
-Branch Specifier
--*/main

6)[Apply] 클릭

7)Build
-Add build step 클릭
--[Invoke top-level Maven targets] 선택
--Goals : clean package
--[고급] 클릭
---POM : SpringBootRestfulMembership/pom.xml
-[저장] 클릭

8)Build Now click

Finished: SUCCESS 확인

9)jenkins container에서 확인
/var/jenkins_home/workspace/demo/SpringBootRestfulMembership/target에서 war 확인

6. 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}}

7. 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

8. 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

Task3. 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

3. Jenkins job이 GitHub의 Push Event 받기
1)Github plugin은 jenkins가 Github의 Webhook을 받을 수 있도록 수동 모드와 자동 모드를 제공한다.
2)수동 모드는 사용자가 Git Repository setting에서 Hook & Services를 직접 등록해 사용하는 것이고,
3)자동 모드는 사용자의 GitHub OAuth 토큰을 이용해 Jenkins가 자동으로 Hooks & Service를 등록한다.
4)자동으로 Webhook이 만들어지지 않는 경우 수동으로 Webhook을 만든다.
5)payload URL : Jenkins 주소/github-webhook/
6)GitHub OAuth 등록
-GitHub에서 Settings 선택
-Settings > Developer settings > Personal access tokens > Generate new token
-Note : jenkins_access_token
-Expiration : 30 days
-Select scopes
--repo
--admin:repo_hook
--delete_repo
--{{GitHub에서 발급받은 personal_access_token}}

7)Jenkins > Jenkins 관리 > 시스템 설정
-GitHub
-GitHub Servers > Add GitHub Server
-GitHub Server
--Name : My GitHub Server
--API URL : 그대로
--Credentials
--Add > Jenkins
---Domain : Global crredentials
---Kind : Secret text 선택
---Scope : Global(Jenkins...)
---Secret : {{GitHub에서 발급받은 personal_access_token}}
---ID : accss-token
---Add Click
--[Test connection] 클릭해서 연결 테스트
---Credentials verified for user gitinstructor, rate limit: 4999

-저장 클릭

8)GitHub의 해당 Repository의 Settings
-Webhooks가 자동으로 등록되어 있는지 확인
-만일 자동등록이 안되어 있으면 수동으로 해야 함.
-[Add webhook] click
--Payload URL : http://192.168.0.14:8080/github-webhook/
-Content type : application/json
-[Add webhook] click

9)GitHub의 해당 Repository의 Settings
-Webhooks가 자동으로 등록되어 있는지 확인
-만일 자동등록이 안되어 있으면 수동으로 해야 함.
-[Add webhook] click
--Payload URL : {{ AWS EC2 Domain Name }}/github-webhook/
-Content type : application/json
-[Add webhook] click