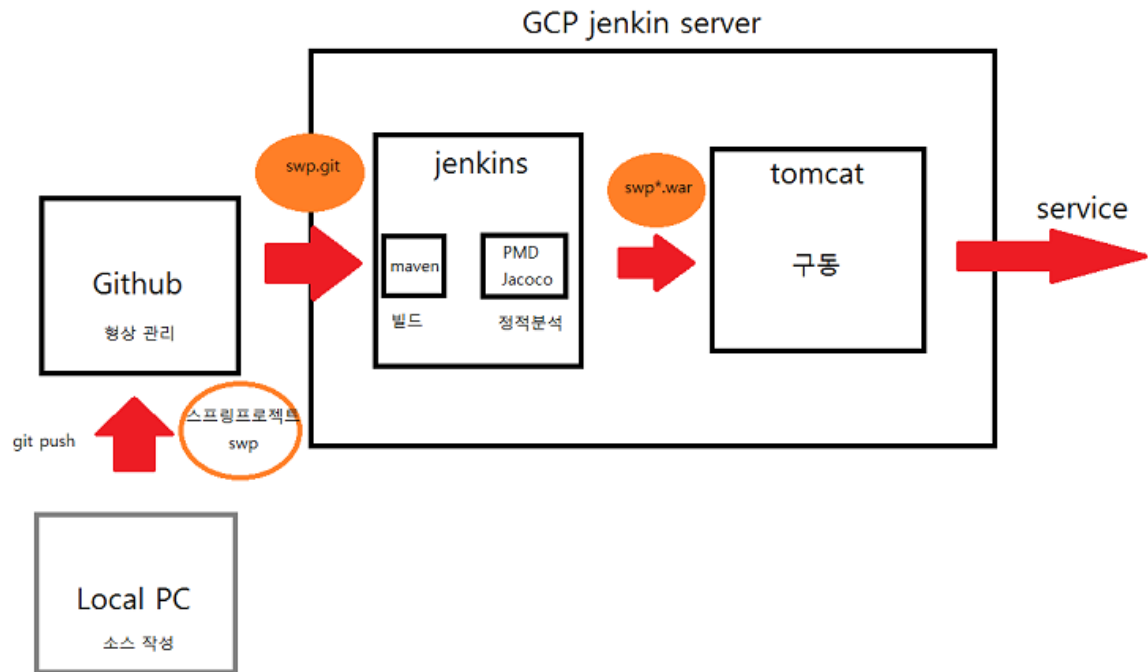


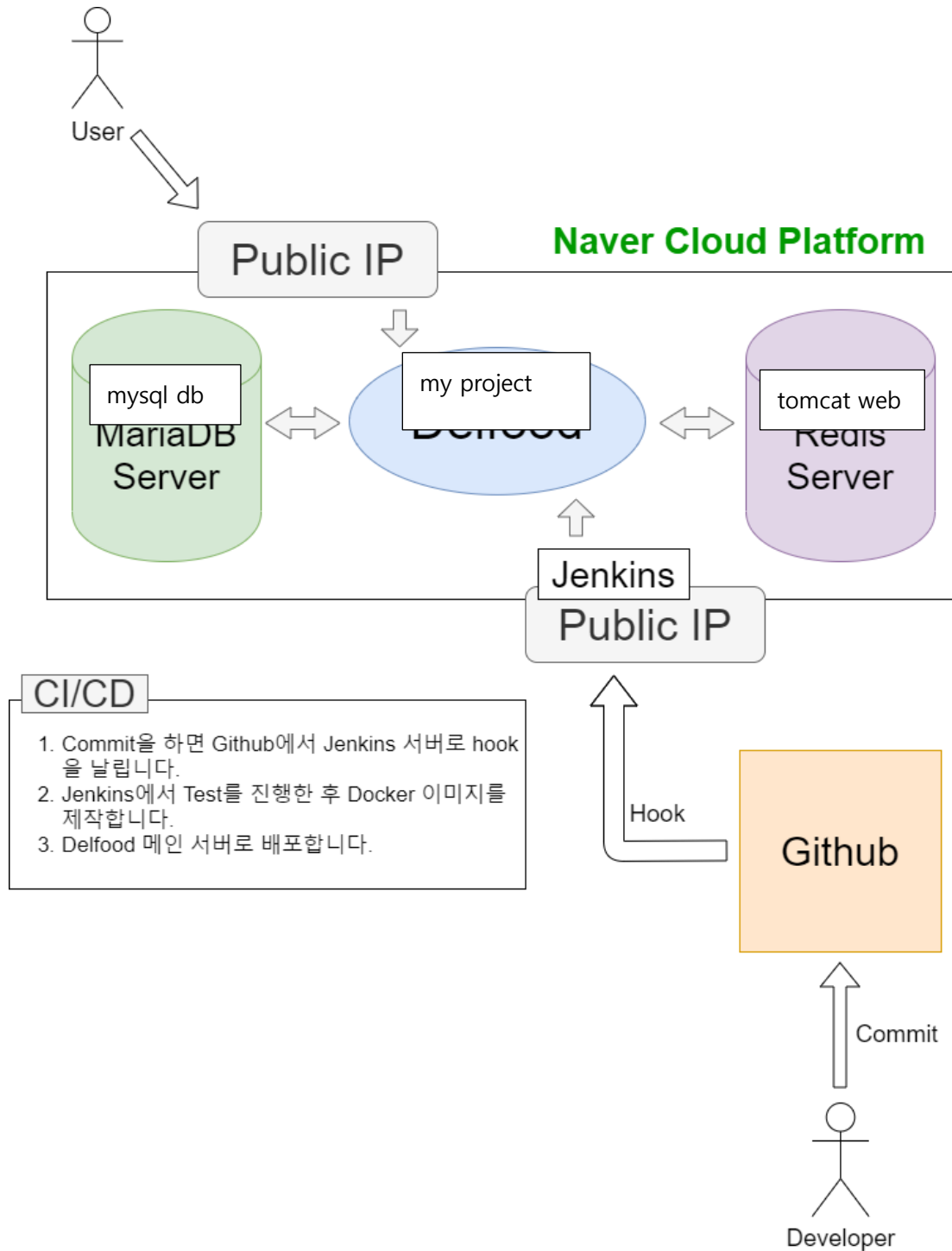
네이버 클라우드 플랫폼 서버 세팅 - 내 프로젝트 올리기



<<프로젝트 서버 구축>>

무료 OS인 Linux와 jdk, 웹서버 Tomcat, 데이터를 저장할 MySQL, 스프링 라이브러리 사용을 위한 maven 설치로 구축 예정
서버 한 대에 이 모든 프로그램 설치

프로젝트 배포를 위한 구성



<<서비스 배포를 위한 서버 구성 순서>>

1. 네이버 클라우드 서버 생성 (비공인ip할당됨)
2. ACG 설정(방화벽 구실로 22번 포트로 원격접속시 또는 웹서비스등의 포트 사용시 필요)
3. 공인 IP 할당
4. port forwarding 설정
5. putty 설치 / 접속
6. jdk 설치
7. maven 설치
8. tomcat 설치
9. mysql 설치
11. jenkins 접속 / 플러그인 설치
12. jenkins 업데이트
13. jenkins 와 github 연동
14. jenkins와 jdk, maven, tomcat 연동

- jenkins 설정

네이버 클라우드 플랫폼 사이트 로그인 – 콘솔 이동

0. 구성한 서버의 acg 설정에 18080 포트 추가

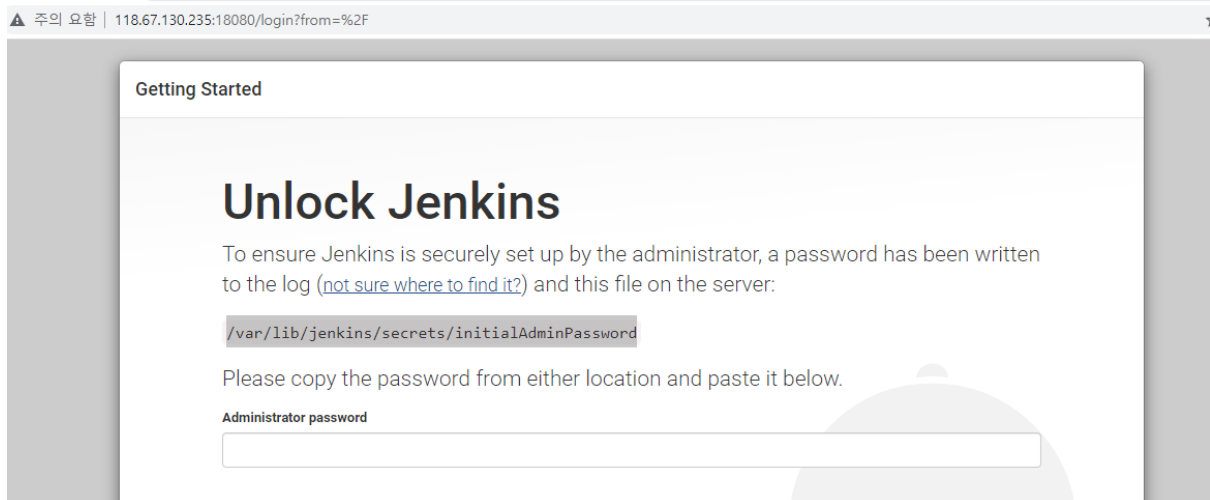
1. <http://공인ip:18080> 접속

==> 안되면

1-1. acg 확인

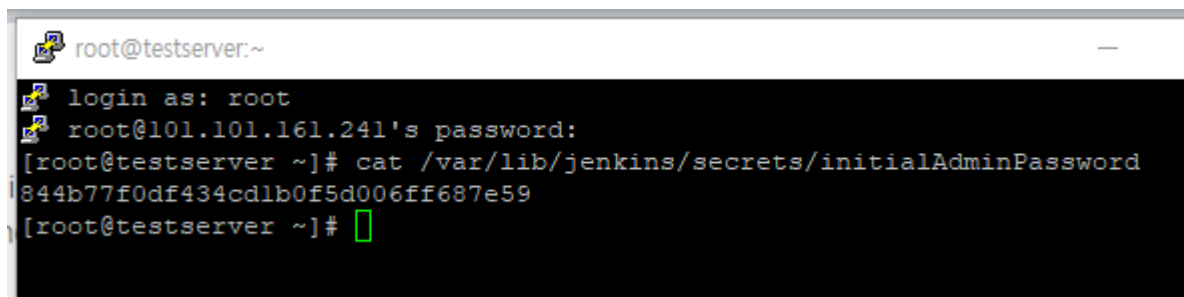
1-2. jenkins 프로세스 정지상태일 수 있으니, putty에서 service jenkins restart 해볼 것

2. 비번 확인하여 jenkins 로 접속.



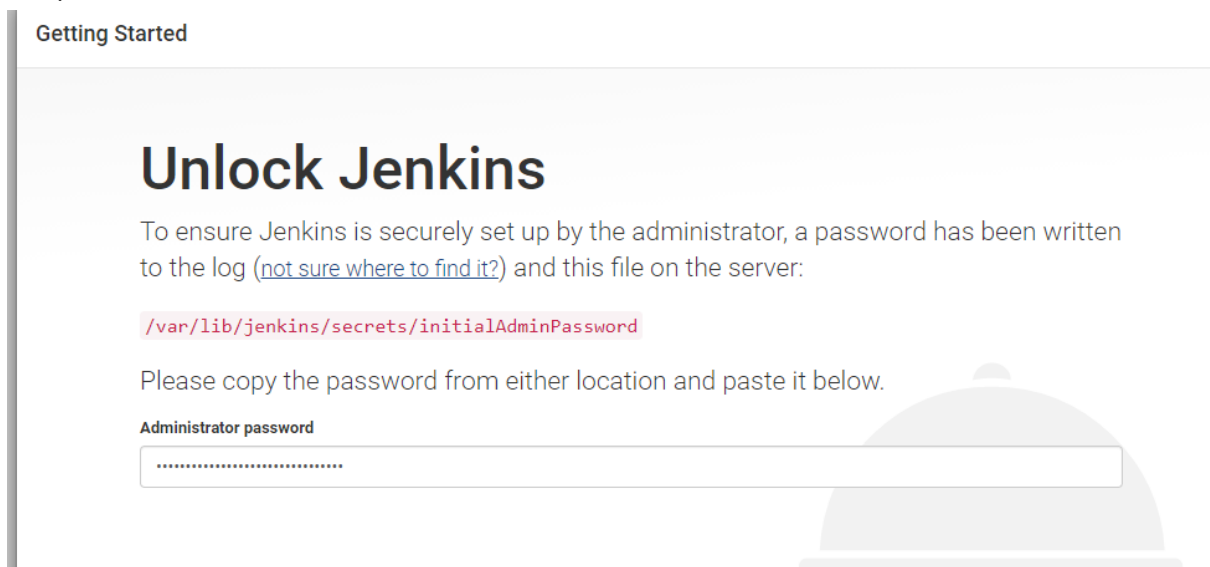
putty 접속

cat `/var/lib/jenkins/secrets/initialAdminPassword`



putty에서 확인된 값을 드래그-엔터하여 복사

입력-continue



install suggested plugin 설치 선택

설치중 화면

Getting Started

✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	✓ Credentials Binding	** SSH server
✓ Timestampers	Workspace Cleanup	Ant	Gradle	Folders
Pipeline	GitHub Branch Source	Pipeline: GitHub Groovy Libraries	Pipeline: Stage View	OWASP Markup Formatter
Git	SSH Build Agents	Matrix Authorization Strategy	PAM Authentication	** Structs
LDAP	Email Extension	Mailer		** Trilead API
				** Pipeline: Step API
				** Token Macro
				Build Timeout
				** JAXB
				** Credentials
				** Plain Credentials
				** SSH Credentials
				Credentials Binding
				** SCM API
				** Pipeline: API
				Timestampers
				** Caffeine API
				** Script Security
				** Plugin Utilities API
				** Font Awesome API
				** Popper.js API
				** JQuery3 API
				** Bootstrap 4 API
				** Snakeyaml API
				** - required dependency

3. 플러그인 설치 후

Getting Started

Jenkins is ready!

Your Jenkins setup is complete.

[Start using Jenkins](#)

6. 젠킨스 실행중인 브라우저에서 새 계정 생성

Create First Admin User

계정명:

암호:

암호 확인:

이름:

이메일 주소:

아래와 같이 입력(암호 jenkins1234)

Create First Admin User

계정명:

암호:

암호 확인:

이름:

이메일 주소:

7. 서비스 새로 시작시 다시 로그인하라고 함. url 확인 - 접속

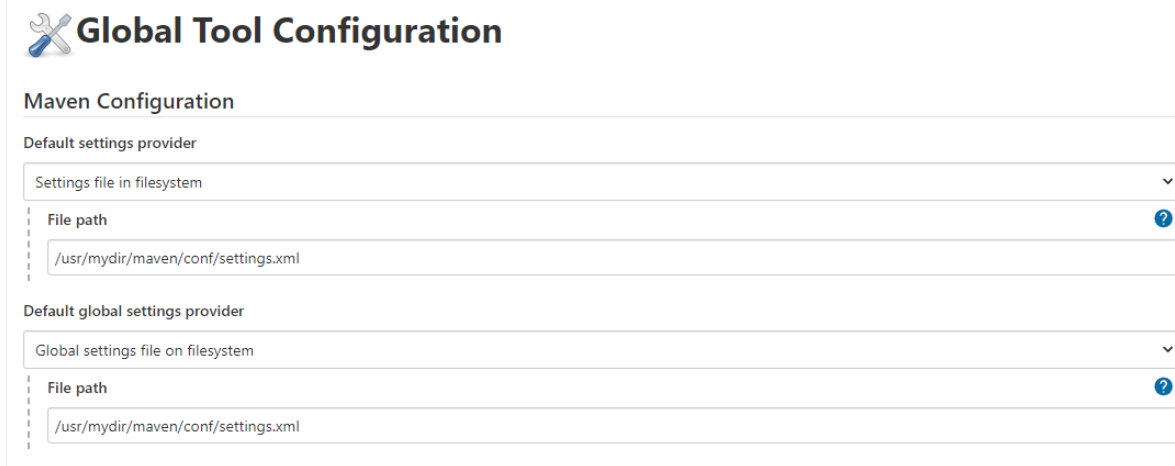


Welcome to Jenkins!

☐ 로그인 상태 유지

8. 젠킨스 메인 화면 - Jenkins 관리 - Global Tool Configuration - Maven Configuration

Default settings provider - Settings file in filesystem 과 Default global settings provider - Global settings file on filesystem 선택 – file path에 maven 경로/conf/settings.xml 등록.



Global Tool Configuration

Maven Configuration

Default settings provider

Settings file in filesystem

File path

/usr/mydir/maven/conf/settings.xml

Default global settings provider

Global settings file on filesystem

File path

/usr/mydir/maven/conf/settings.xml

default settings..... 의 경우 지정하지 않으면 .m2/settings.xml 이다

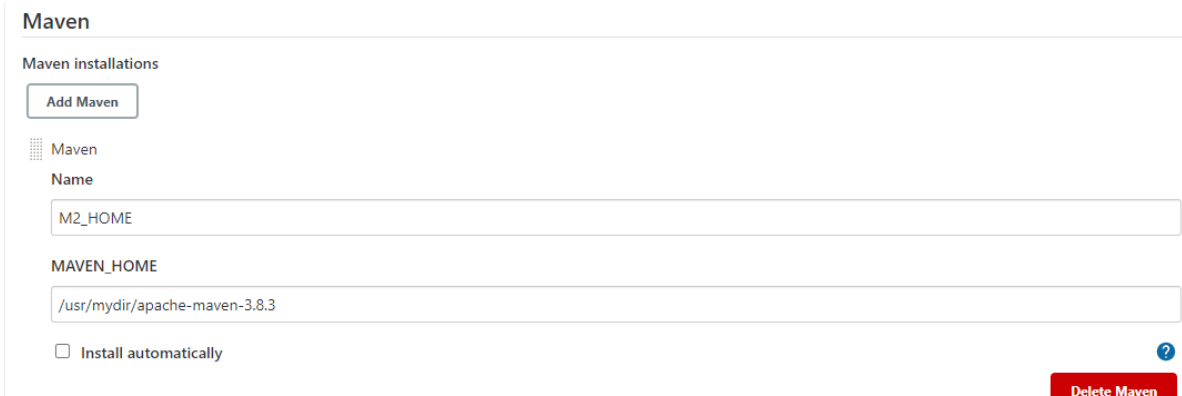
default global settings..... 의 경우 지정하지 않으면 메이븐경로/conf/settings.xml 이다
다른 경로의 설정 파일을 사용한다면 변경한다.

9. 아래로 스크롤 - Maven 메뉴 - Add Maven 버튼 - install automatically 체크 해제 MAVEN_HOME 인풋이 생기면 경로 넣자

Name – M2_HOME

MAVEN-HOME - /usr/mydir/apache-maven-3.8.1/

여러개 다른 버전으로 생성 가능. maven 동작시 필요한 settings.xml 파일은 8번에서 지정한 경로이고 maven 관련 라이브러리 저장은 M2_HOME에서 정한 곳으로 다운로드한다.



Maven

Maven installations

Add Maven

Maven

Name

M2_HOME

MAVEN_HOME

/usr/mydir/apache-maven-3.8.3

☐ Install automatically

Delete Maven

10. 다시 위로 올라가자.

JDK – ADD JDK - install automatically 체크 해제

JAVA_HOME

/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.292.b10-1.el7_9.x86_64/

==> PUTTY에서 echo \$JAVA_HOME 해서 나오는 디렉토리가 우리가 설치한 경로
/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.302.b08-0.el7_9.x86_64

JDK

JDK installations

Add JDK

JDK

Name

JAVA_HOME

JAVA_HOME

/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.302.b08-0.el7_9.x86_64

☐ Install automatically

Delete JDK

11. 저장

Save Apply

- 12. jenkins 메인 - jenkins 관리 – 플러그인 관리 – 설치 가능 - Deploy to container plugin - 검색 설치 – install without restart

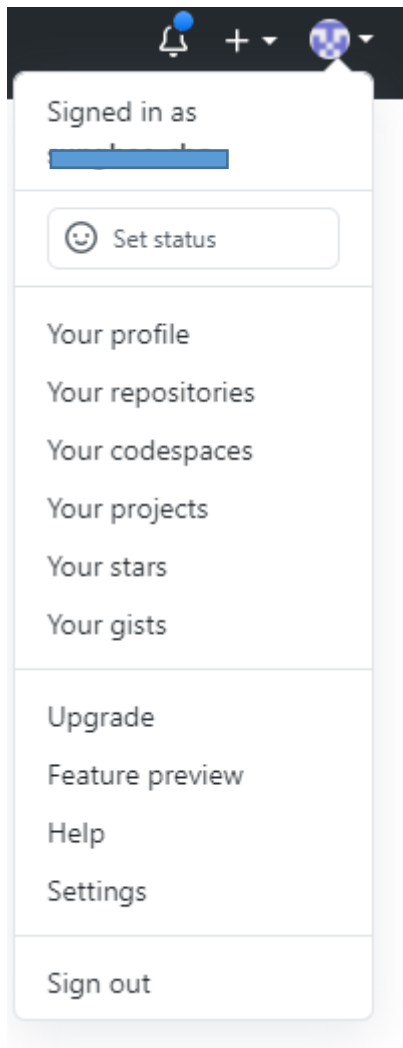
Q deploy

업데이트된 플러그인 목록 설치 가능 설치된 플러그인 목록 고급

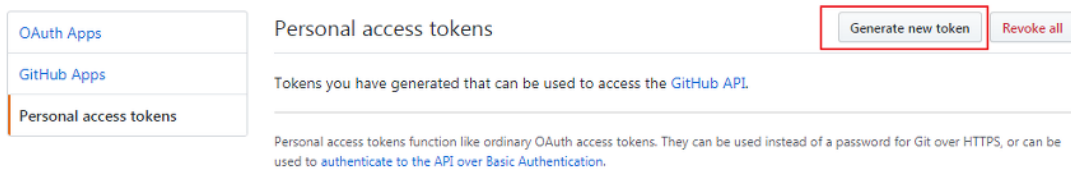
Install ↑	Name	Version	Released
<input checked="" type="checkbox"/>	Deploy to container Artifact Uploaders This plugin allows you to deploy a war to a container after a successful build. Glassfish 3.x remote deployment	1.16	11 mo ago

13. service jenkins retart

14. github 연동을 위해 github 사이트 로그인 – 프로필 아래 – settings - Developer settings



15. 새 토큰을 발급



16. 스코프 선택, 토큰 복사

OAuth Apps

GitHub Apps

Personal access tokens

New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Token description

jenkins-test-token

What's this token for?

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input type="checkbox"/> admin:org	Full control of orgs and teams
<input type="checkbox"/> write:org	Read and write org and team membership
<input type="checkbox"/> read:org	Read org and team membership
<input type="checkbox"/> admin:public_key	Full control of user public keys
<input type="checkbox"/> write:public_key	Write user public keys
<input type="checkbox"/> read:public_key	Read user public keys
<input checked="" type="checkbox"/> admin:repo_hook	Full control of repository hooks
<input checked="" type="checkbox"/> write:repo_hook	Write repository hooks
<input checked="" type="checkbox"/> read:repo_hook	Read repository hooks

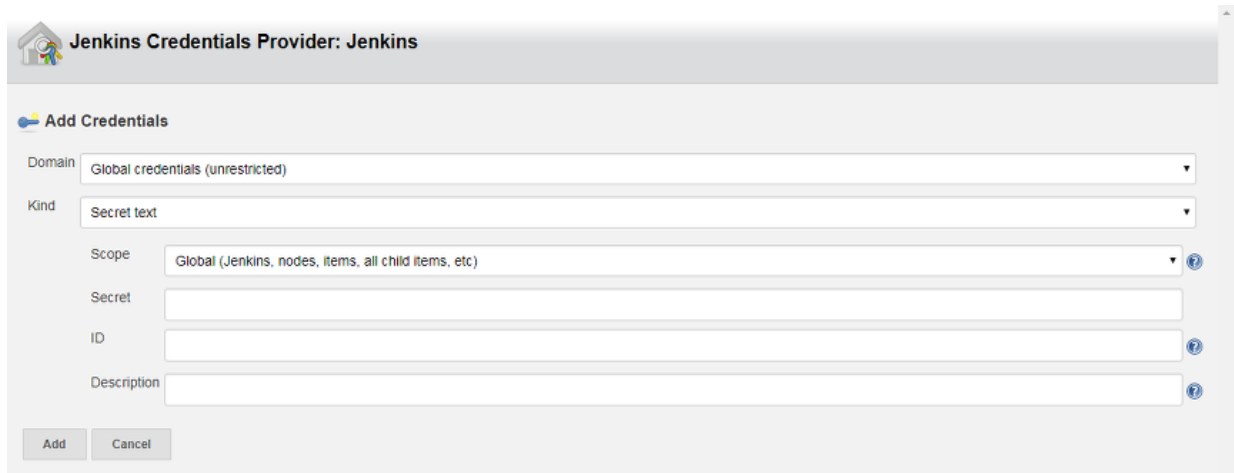
토큰 복사

(이 값 잊어버렸다면 다시 해당 토큰 들어가 regenerate 할 것)

17. 젠킨스로 돌아가서, jenkins 관리 - 시스템 설정

18. 아래로 스크롤 - Github - Add GitHub Server 클릭 - mygithub 이름 추가, API URL 그대로, Add 버튼 클릭

19. 도메인, 종류 선택 - Secret에 발급받은 access token 붙여넣기. ID는 구분용이니 아무거나.



Jenkins Credentials Provider: Jenkins

Add Credentials

Domain: Global credentials (unrestricted)

Kind: Secret text


Scope: Global (Jenkins, nodes, items, all child items, etc)

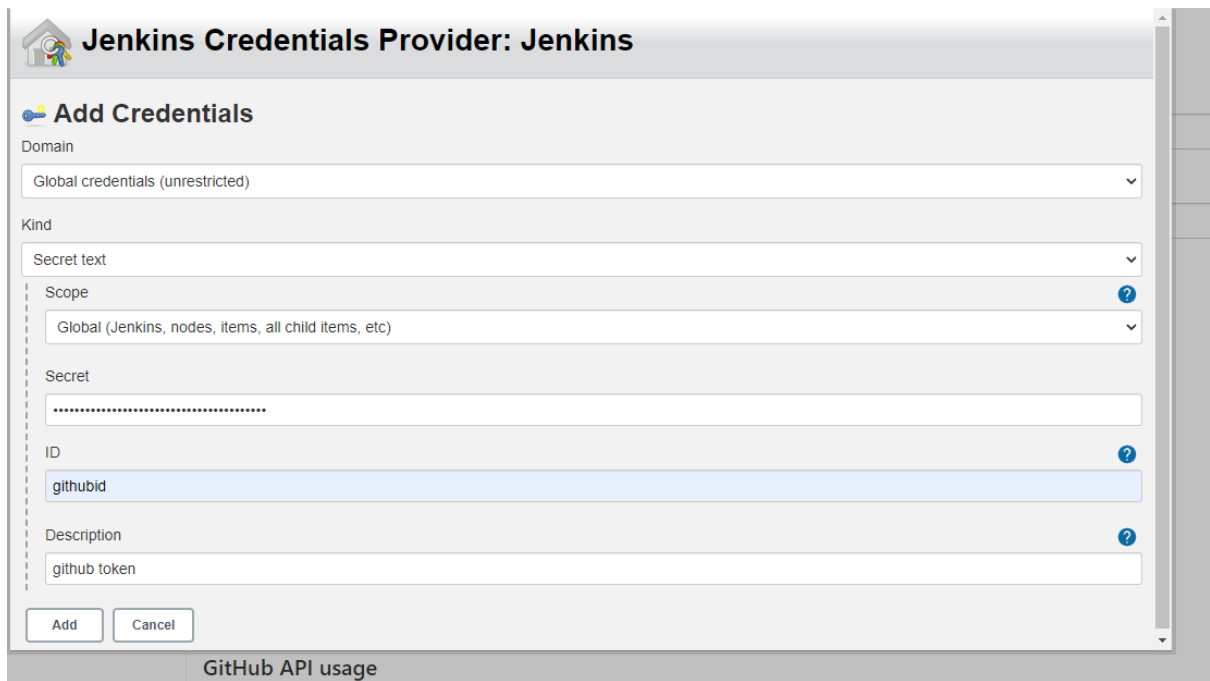
Secret:

ID:

Description:

Add Cancel

github token 발급 - 



Jenkins Credentials Provider: Jenkins

Add Credentials

Domain: Global credentials (unrestricted)

Kind: Secret text

Scope: Global (Jenkins, nodes, items, all child items, etc)

Secret:

ID: githubid

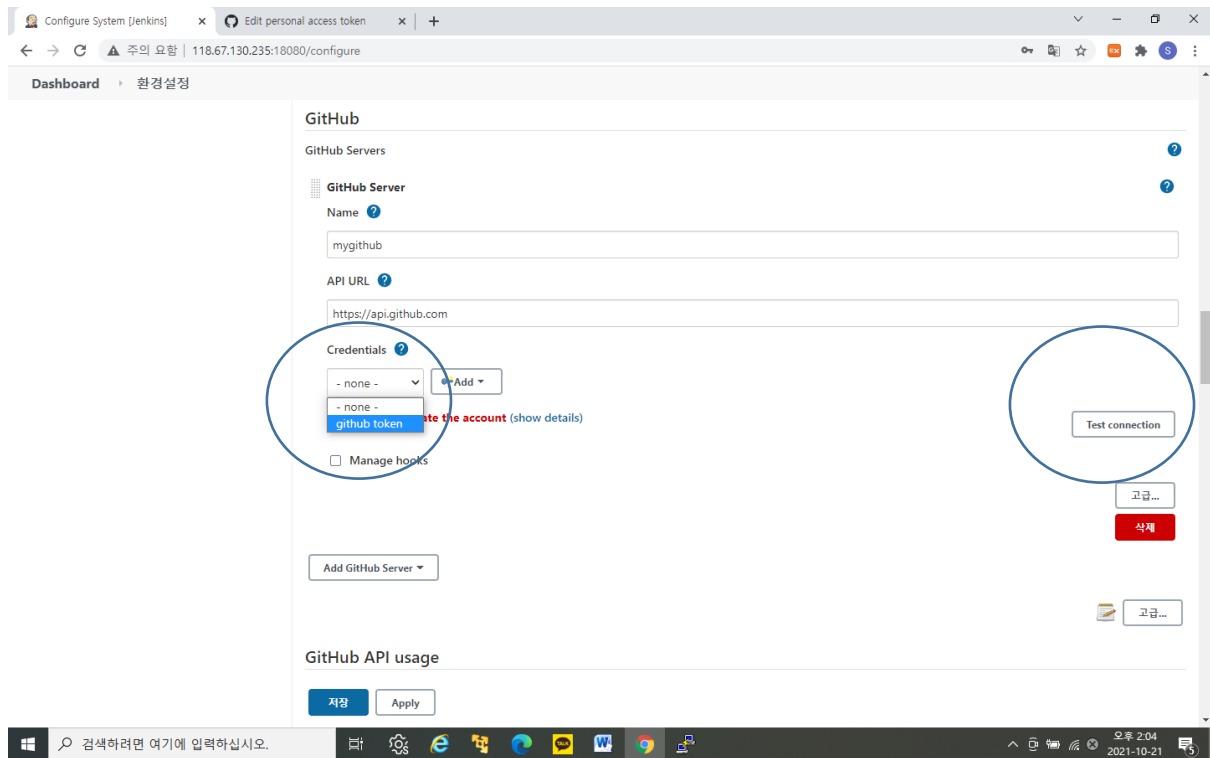
Description: github token

Add Cancel

GitHub API usage

add 클릭

20. Test connection



결과 확인



github와 연동 완료

===== 내 컴퓨터의 sts 시작 =====

21. github에 push할 스프링 부트 프로젝트 생성

첫 화면에서 **packaging : WAR** 압축 선택!!!!(그러면 ServletInitializer.java 자동 생성되는데 다른 서버에서 war로 배포시 web.xml 역할을 하게 된다)

==> 수동으로 추가했더니 네이머서버에서 jsp를 서블릿 소스로 변경하여 컴파일하는 과정이 실행되지 않는 점이 발견되었다!!! 주의하자.)

New Spring Starter Project

Service URL:

Name:

☒ Use default location

Location:

Type: Packaging:

Java Version: Language:

Group:

Artifact:

Version:

Description:

Package:

Working sets

☐ Add project to working sets

test용 spring boot 프로젝트 jenkinsproject 생성

pom.xml – jsp, jstl, server restart 관련 dependency 추가

application.properties – 서버 포트, jsp 자동 재시작, jsp view 설정, mybatis 관련 설정

webapp\WEB-INF\views*.jsp 추가

Controller 추가

브라우저에서 테스트

테스트 후에는 스프링 부트 서버 종료(그래야 git에 올린다)

==> pom.xml에 jar 압축으로 되어있다면 war 압축으로 변경한다!!!!

==> 2가지 확인한다. pom.xml, ServletInitializer.java

```
<groupId>com.example</groupId>
```

```
<artifactId>naverserver</artifactId>
```

```
<version>0.0.1-SNAPSHOT</version>
```

```

<packaging>war</packaging>
<name>naverserver</name>
<description>Demo project for Spring
Boot</description>
<properties>
    <java.version>1.8</java.version>
</properties>

```

그러면 ServletInitializer.java 도 같이 생성된다.

```

public class
ServletInitializer extends
SpringBootServletInitializer
{
    @Override
    protected SpringApplicationBuilder
configure(SpringApplicationBuilder application) {
        return
application.sources(JenkinsmysqldbprojectApplication.class);
    }
}

```

22. github에 push 하자.

github 로그인

github에 repository 생성 - 이름 입력 - code - uri 복사

내자리 - sts - clone git repository - 아이디는 깃헙 이메일, 암호는 sts 용으로 생성했던 토큰값을 넣는다

참고사항

- sts에서 깃허브로 push 할 때 암호로 인해 문제 발생시 window-preferences-general-security-secure storage - contents - default secure... - git 선택 후 delete
- 암호 초기화되었으므로 새로운 토큰값으로 새로 인증하면 됨

내 로컬 저장소 확인

프로젝트 선택 - team - share project - repository 선택(여러개라면) -

프로젝트 선택 - team - add to index - commit...- commit and push

==> 진행안되면 토큰값 안맞을 확률 큼. 확인하여 토큰 다시 생성 붙여넣는다

===== 내 컴퓨터의 sts 종료 =====

23. 젠킨스 메인화면 - **New Item** - **Freestyle project** 를 선택.

Enter an item name

jenkins-test

» Required field

Freestyle project

이것은 Jenkins의 주요 기능입니다. Jenkins은 어느 빌드 시스템과 어떤 SCM(형상관리)으로 묶인 당신의 프로젝트웨어 빌드보다 다른 어떤 것에 자주 사용될 수 있습니다.

Pipeline

24. GitHub project 체크 - 프로젝트 url 입력.

<https://github.com/유저네임/레포지토리명> 입니다.

The image shows the Jenkins configuration interface for a new project. The 'General' tab is selected. Under the 'Description' section, there is a text area and a '미리보기' (Preview) button. Below this, there are three checkboxes: 'Commit agent's Docker container' (unchecked), 'Define a Docker template' (unchecked), and 'GitHub project' (checked). Under the 'GitHub project' section, the 'Project url' is set to 'https://github.com/sunghee-cho/jenkinstest/'.

github 프로젝트명 - <https://github.com/사용자명/레포지토리명>

25. 소스코드 관리 -Git

<https://github.com/사용자명/레포지토리명.git>

Credentials- Add. - Username with password – username - github 아이디(프로필 사진 아래 볼드처리), password- github 패스워드를 입력. ID 는 구분용이니 아무거나.

26. build

Build 탭 – add build step - Invoke top-level Maven targets

Maven Version Global Tool Configuration - Maven 이름 선택.

Branches to build – 공백으로~

아래와 같이 채워 주세요.



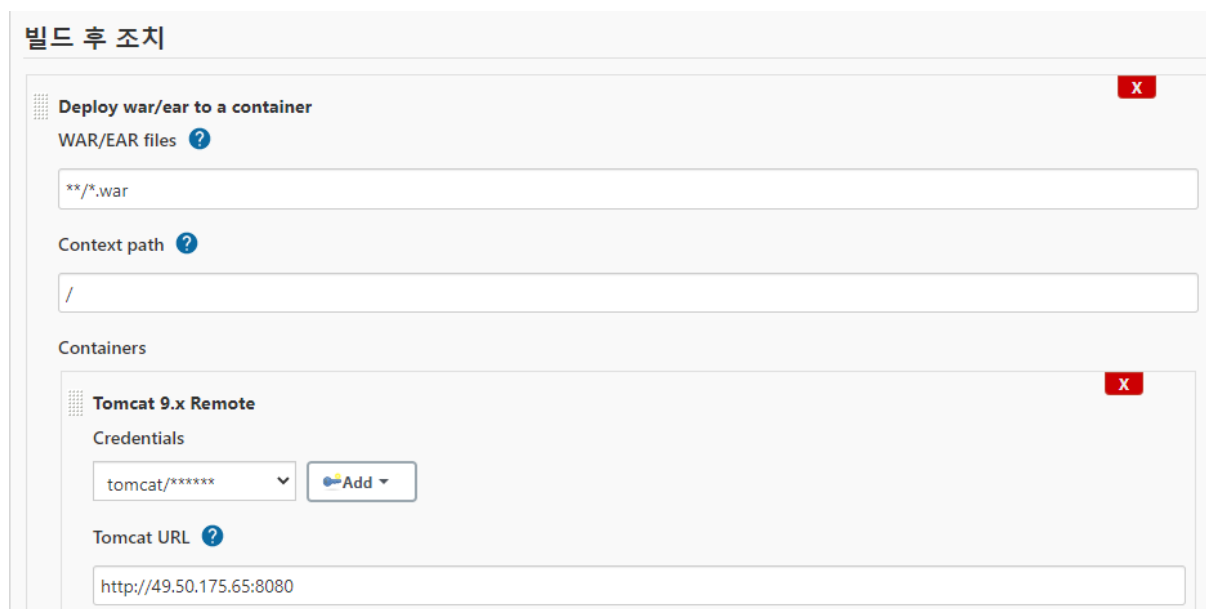
The screenshot shows the 'Build' configuration section in Jenkins. It is titled 'Invoke top-level Maven targets'. There are three main fields: 'Maven Version' set to 'M2 HOME', 'Goals' set to 'clean package', and 'POM' set to 'jenkinsproject/pom.xml'. Each field has a dropdown arrow on the right. There are also red 'X' and blue '?' icons in the top right corner of the section.

Goals – clean package – 이전 빌드 내용 삭제

이 때 pom.xml 은 프로젝트명/pom.xml

27. 빌드 후 조치

tomcat 으로 자동배포하자.



The screenshot shows the '빌드 후 조치' (Post-build Actions) configuration section in Jenkins. It is titled 'Deploy war/ear to a container'. There are three main fields: 'WAR/EAR files' set to '**/*.war', 'Context path' set to '/', and 'Containers' set to 'Tomcat 9.x Remote'. The 'Containers' section has a red 'X' icon in the top right corner. Below 'Containers', there are two sub-sections: 'Credentials' and 'Tomcat URL'. The 'Credentials' section has a dropdown menu set to 'tomcat/*****' and an 'Add' button. The 'Tomcat URL' section has a text field set to 'http://49.50.175.65:8080'. There are also red 'X' and blue '?' icons in the top right corner of the section.

위 내용 중 add 버튼 클릭하여 id 와 password 입력.-->

root / 1234

이 화면 상태로 저장해야 함 credentials 에 none 뜨지 않도록!!!

배포 서버의 톰캣 경로/**conf/tomcat-users.xml** 에 아래 role 과 user 추가.

위에서 **Credentials Add -** 추가한 user 의 name, password 입력.

```
<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<role rolename="manager-status"/>
<user name="아이디" password="패스워드" roles="manager-gui,manager-script,manager-status" />
```

putty

vi /usr/mydir/tomcat/conf/tomcat-users.xml

```
<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<role rolename="manager-status"/>
<user username="tomcat" password="1234" roles="manager-gui,manager-script,manager-status" />
```

vi /usr/mydir/tomcat/webapps/manager/META-INF/context.xml

<Valve 태그 주석처리

tomcat 재시작

/usr/mydir/tomcat/bin/shutdown.sh

/usr/mydir/tomcat/bin/startup.sh

<http://118.67.130.235:8080/manager/html>

tomcat / 1234 입력하여 테스트

28. build now

console output 보면

/var/lib/jenkins/jobs/젠킨스 item 명//workspace/깃허브프로젝트명/target/깃허브프로젝트명-0.0.1-SNAPSHOT.war

이 빌드된 결과이다.

tomcat/webapps/ROOT.war로 디플로이된다.(컨텍스트패스 '/')

이제 tomcat에서 실행하자.

브라우저 열고

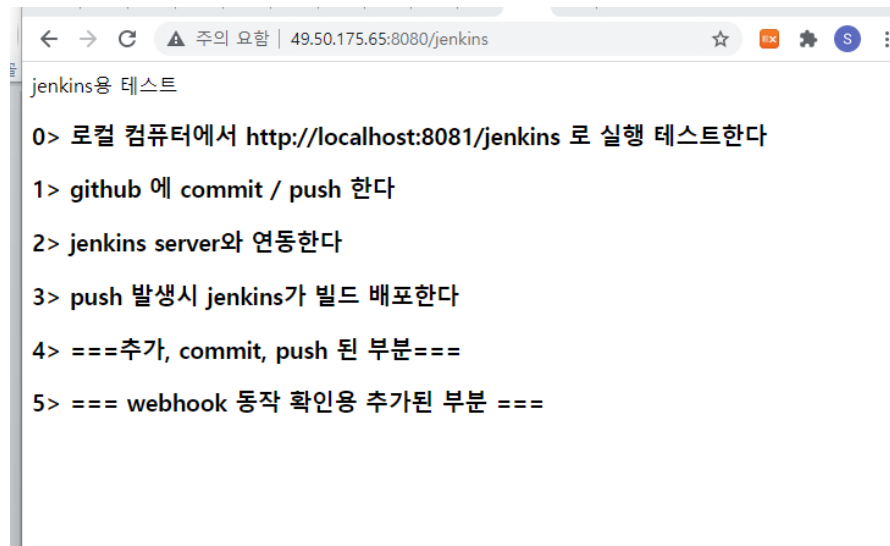
<http://공인ip:8080/jenkins> 실행하며 결과 확인

1> 만약 톰캣에 빌드시 컨텍스트 패스를 /test 형식으로 했다면

tomcat/webapps/test.war로 디플로이된다.(컨텍스트패스 '/test')

그러면 브라우저에서 호출시

<http://49.50.175.65:8080/test/jenkins> 형식으로 호출한다.



-리눅스 사용시 참고사항들

1 리눅스 서버 파일 업로드 경로 만들기

1-1. putty 접속

```
mkdir /usr/mydir/upload
```

1-2. WebConfig.java 열어서

```
package com.multi.myboot01;
```

```
import org.springframework.context.annotation.Configuration;
```

```
import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;
```

```
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
```

```
@Configuration //현재클래스 설정 모든 결과 xml 파일 <bean
```

```
//@ComponentScan-<context:component-scan.대신)
```

```
public class WebConfig implements WebMvcConfigurer {
```

```
    @Override
```

```
    public void addResourceHandlers(ResourceHandlerRegistry registry) {
```

```
        registry.addResourceHandler("/upload/**")//url 설정
```

```
        .addResourceLocations("file:///c:/upload/")//windows 실제경로
```

```
        .addResourceLocations("file:/usr/mydir/upload/")// linux
```

```
//addResourceLocations("file:/DATA/video/"); //리눅스 root 에서 시작하는 폴더 경로
```

```
        registry.addResourceHandler("/faceimages/**")
```

```
        .addResourceLocations("file:///C:/Users/student/Desktop/images/");  
        registry.addHandler("/faceimages/**")  
        .addResourceLocations("file:///C:/Users/user/Desktop/images/");  
    }  
}
```

1-3. UploadController 열어서

```
String savePath = "/usr/mydir/upload/";
```