

# SW 개발·테스트 중심 DevOps 환경구축 및 활용

---

## 강사 정보

시네틱스 대표 한동준  
handongjoon@gmail.com  
dongjoon.han@synetics.kr

# 교육 소개

---

## □ 목적

- DevOps 운영에 필수인 개발(Development) 환경의 자동화를 위하여 오픈소스 도구를 활용한 ALM(Application Lifecycle Management) 환경 구축 및 운영 역량을 확보

## □ 내용

- DevOps에서 ALM의 필요성에 구성에 대한 이해
- 주요 오픈소스 ALM 도구 이해
  - 단독으로 사용하는 방법
  - Maven, Jenkins, Eclipse 에서 사용하는 방법

## □ 교육에서 강조하는 부분

- 도구 설치, 도구 간 연동 방법
- Jenkins 중심의 활용 방법

## □ 교육 시간 내에 상대적으로 적게 다루는 부분

- 국내에 이미 책과 교육이 활성화 되어 있는 도구의 기본 사용 방법: Maven, Git

## 강의 순서

---

1. DevOps 와 오픈소스 개발 환경 자동화의 이해
2. 빌드: Maven
3. 버전 관리: Git
4. 지속 통합: Jenkins
5. 단위 테스트: Junit / Cobertura
6. 룰 기반 소스코드 분석: PMD
7. 의존성 분석: Jdepend
8. 소스코드 매트릭: JavaNCSS

## 2 Maven



# 도구 소개

---

## □ 도구 개요

- Java 빌드 관리 도구

## □ 사용 목적

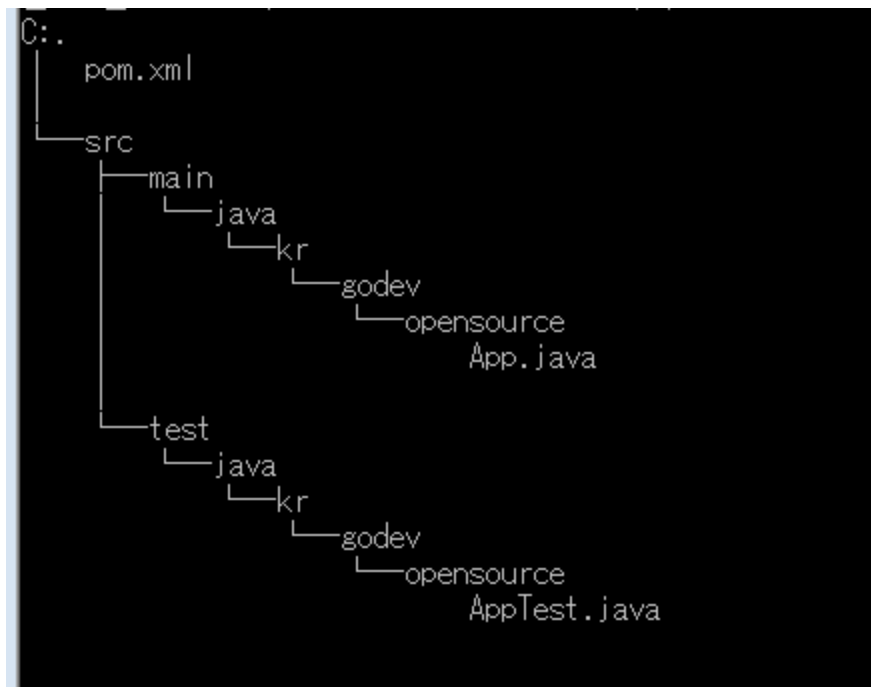
- 모두가 동일한 바이너리를 생성하기 위해서
- 정형화된 소스코드 구조를 사용하기 위해서
- 라이브러리 의존성을 관리하기 위해서
- 다른 도구와 쉽게 연동하기 위해서

## □ 일반적인 사용 방법

- 팀의 AA(Application Architect)가 Maven 구조 생성 및 의존성 관리
- pom.xml 파일에 라이브러리 의존성 및 플러그인 관리
  - QA의 경우, 품질 관련 플러그인 설정
- 개발 팀원은 정해진 패키지에서 소스코드 개발

## 도구 특징

### □ 정형화된 프로젝트 구성



### □ pom.xml

- pom.xml을 살펴보면, 사용하는 라이브러리와 구성 요소를 파악할 수 있음

# 라이프사이클과 단계

## □ 라이프사이클

라이프사이클	설명
clean	이전 빌드에서 생성된 산출물을 삭제하는 기능
default	프로젝트 빌드/배포와 관련된 기능
site	프로젝트의 사이트 문서 생성과 관련된 기능

## □ default의 단계

라이프사이클	설명
validate	정상적인 프로젝트이고 필요한 모든 정보가 준비되었는지 확인
compile	프로젝트의 소스코드를 컴파일
test	컴파일한 소스코드를 단위 테스트 프레임워크를 이용해 테스트
package	JAR와 같이 지정된 포맷으로 패키징
verify	패키지가 품질 수준을 만족하는지 검증
install	패키지를 로컬 저장소에 설치
deploy	최종 패키지를 원격 저장소로 복사

## □ Maven 명령

***mvn clean package site pmd:pmd jdepend:generate***

# 플러그인 Goal과 설정 방법

## □ 플러그인 Goal

### Plugin Documentation

Goals available for this plugin:

Goal	Report?	Description
<a href="#">pmd:check</a>	No	Fail the build if there were any PMD violations in the source code.
<a href="#">pmd:cpd</a>	Yes	Creates a report for PMD's CPD tool. See <a href="http://pmd.sourceforge.net/cpd.html">http://pmd.sourceforge.net/cpd.html</a> for more detail.
<a href="#">pmd:cpd-check</a>	No	Fail the build if there were any CPD violations in the source code.
<a href="#">pmd:help</a>	No	Display help information on maven-pmd-plugin. Call <code>mvn pmd:help -Ddetail=true -Dgoal=&lt;goal-name&gt;</code> to display parameter details.
<a href="#">pmd:pmd</a>	Yes	Creates a PMD report.

## □ 플러그인 설정

### Usage

The PMD plugin generates PMD and CPD reports using the PMD code analysis tool.

To include a report with default rule sets and configuration in your project site, set the following in the `<reporting>` section of your POM:

```
<project>
  ...
  <reporting>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-pmd-plugin</artifactId>
        <version>3.1</version>
      </plugin>
    </plugins>
  </reporting>
  ...
</project>
```



# 다운로드

❑ 대상: 개발자 PC, 빌드 서버

❑ 사전조건: JDK의 설치

❑ 다운로드

- Maven 웹사이트 다운로드: <https://maven.apache.org/download.cgi>
- 최신버전: 3.6. X
- Windows 개발 환경을 위한 'Binary zip archive' 다운로드

## Files

Maven is distributed in several formats for your convenience. Simply pick a ready-made binary distribution archive and follow the [installation instructions](#). Use a source archive if you intend to build Maven yourself.

In order to guard against corrupted downloads/installations, it is highly recommended to [verify the signature](#) of the release bundles against the public [KEYS](#) used by the Apache Maven developers.

	Link	Checksum	Signature
Binary tar.gz archive	<a href="#">apache-maven-3.5.2-bin.tar.gz</a>	<a href="#">apache-maven-3.5.2-bin.tar.gz.md5</a>	<a href="#">apache-maven-3.5.2-bin.tar.gz.asc</a>
Binary zip archive	<a href="#">apache-maven-3.5.2-bin.zip</a>	<a href="#">apache-maven-3.5.2-bin.zip.md5</a>	<a href="#">apache-maven-3.5.2-bin.zip.asc</a>
Source tar.gz archive	<a href="#">apache-maven-3.5.2-src.tar.gz</a>	<a href="#">apache-maven-3.5.2-src.tar.gz.md5</a>	<a href="#">apache-maven-3.5.2-src.tar.gz.asc</a>
Source zip archive	<a href="#">apache-maven-3.5.2-src.zip</a>	<a href="#">apache-maven-3.5.2-src.zip.md5</a>	<a href="#">apache-maven-3.5.2-src.zip.asc</a>

# 설치 및 설정 - 1

---

## □ 설치

- 다운로드 받은 파일 압축 풀기
- 적절한 폴더로 이동
  - 본 강의에서는 "C:\Dev\_tools\Maven" 에 이동
  - Setup.exe 형식의 설치 파일이 아닌, Portable 형식

## □ 설정

- 시스템 환경변수에 'MAVEN\_HOME' 추가
  - 값은 C:\Dev\_tools\Maven
- 시스템 환경변수의 path에 Maven bin 폴더 추가
  - 값은 C:\Dev\_tools\Maven\bin

## ※ 참고: JDK 설치 및 설정

- 설치: 기본 설치
- 설정
  - 시스템 환경변수에 'JAVA\_HOME' 추가: JDK 설치 폴더(JRE 아님!!)
  - 시스템 환경변수의 path에 JDK bin 폴더 추가: JDK 설치 폴더의 bin 폴더

## 설치 및 설정 - 2

### □ 시스템 환경변수가 제대로 적용되도록 재부팅 후 확인

- CMD를 열고, "mvn -version" 실행



```
C:\Windows\system32\cmd.exe

C:\>mvn -version
Apache Maven 3.2.5 (12a6b3acb947671f09b81f49094c53f426d8cea1; 2014-12-15T02:29:23+09:00)
Maven home: C:\maven\bin\..
Java version: 1.7.0_75, vendor: Oracle Corporation
Java home: C:\Program Files\Java\jdk1.7.0_75\jre
Default locale: ko_KR, platform encoding: MS949
OS name: "windows 7", version: "6.1", arch: "amd64", family: "windows"
C:\>
```

# 프로젝트 생성 - 1

## ❑ C:\sourcecode\w\maven1에서 생성

## ❑ 명령

- mvn archetype:generate -DgroupId=kr.godev.opensource -DartifactId=my\_project  
-DinteractiveMode=false

프로젝트 명

```
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building Maven Stub Project (No POM) 1
[INFO] -----
[INFO]
[INFO] ] maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom ]
[INFO]
[INFO] [ maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom [
[INFO]
[INFO] --- maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom ---
[INFO] Generating project in Batch mode
[INFO] No archetype defined. Using maven-archetype-quickstart (org.apache.maven.archetypes:maven-archetype-quickstart:1.0)
[INFO] -----
[INFO] Using following parameters for creating project from Old (1.x) Archetype: maven-archetype-quickstart:1.0
[INFO] -----
[INFO] Parameter: groupId, Value: kr.godev.opensource
[INFO] Parameter: packageName, Value: kr.godev.opensource
[INFO] Parameter: package, Value: kr.godev.opensource
[INFO] Parameter: artifactId, Value: my_project
[INFO] Parameter: basedir, Value: C:\
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir: C:\my_project
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 5.529s
[INFO] Finished at: Tue Feb 04 22:16:07 KST 2014
[INFO] Final Memory: 10M/26M
[INFO] -----
```

## 프로젝트 생성 - 2

### □ 폴더 생성 결과

```
C:.\n├── pom.xml\n└── src\n    ├── main\n    │   ├── java\n    │   │   ├── kr\n    │   │   │   ├── godev\n    │   │   │   │   └── opensource\n    │   │   │   │       App.java\n    ├── test\n    │   ├── java\n    │   │   ├── kr\n    │   │   │   ├── godev\n    │   │   │   │   └── opensource\n    │   │   │   │       AppTest.java
```

# pom.xml

---

- ❑ Maven 프로젝트의 대부분의 설정 정보를 담고 있는 파일
- ❑ 기본 생성된 pom.xml

```
<project xmlns=http://maven.apache.org/POM/4.0.0
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>kr.godev.opensource</groupId>
  <artifactId>my_project</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>my_project</name>
  <url>http://maven.apache.org/</url>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

# 플러그인 설정 추가

## □ PMD 플러그인 추가 예제

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>kr.godev.opensource</groupId>
  <artifactId>my_project</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>my_project</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-pmd-plugin</artifactId>
        <version>3.1</version>
      </plugin>
    </plugins>
  </build>

</project>
```

## 최초 Compile 실습

---

- ❑ mvn package 실행

Build Success 결과 확인



## [실습] Junit 4 빌드

### □ Github에서 Junit 4 소스코드 다운로드

- <https://github.com/junit-team/junit4>
- mvn install 실행

A programmer-oriented testing framework for Java. <http://junit.org/junit4/>

2,195 commits   5 branches   20 releases   138 contributors   EPL-1.0

Branch: master   New pull request   Find file   Clone or download

stefanbirkner Don't build with Oracle JDK 7 on Travis anymore ...

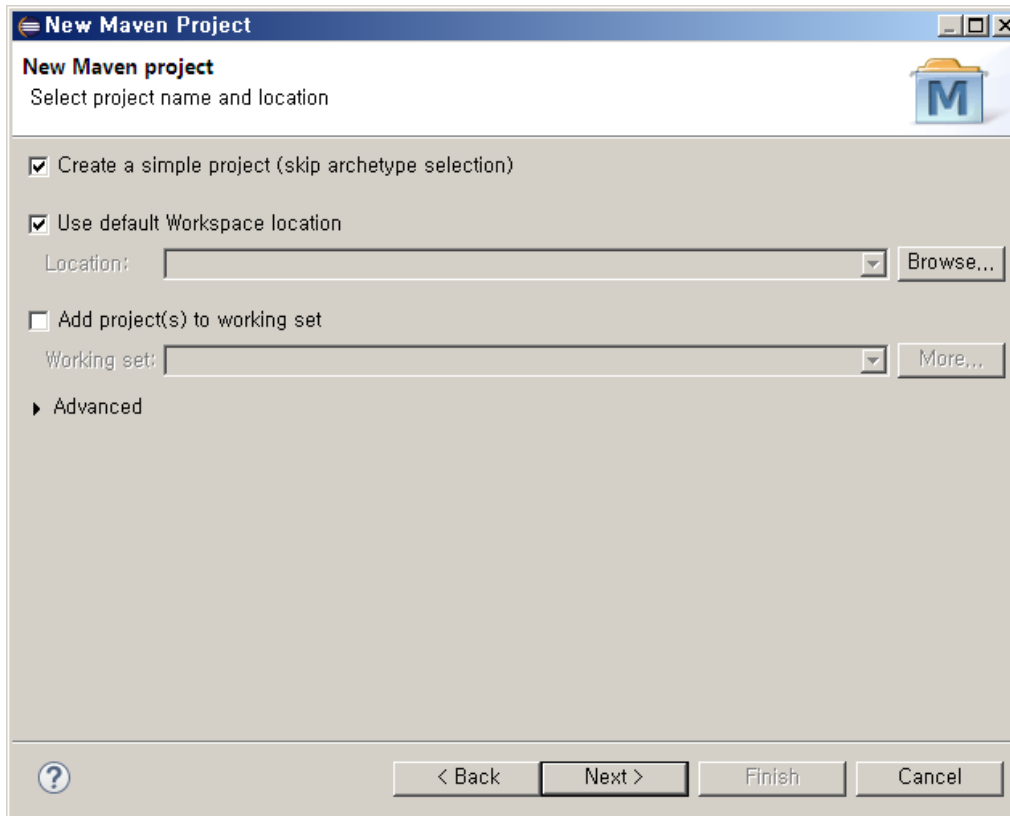
mvn/wrapper	Build with Maven 3.1.1 (using Maven Wrapper)	
.settings	Revert on request of kcooney:	
doc	Build with Maven 3.1.1 (using Maven Wrapper)	
lib	Add Hamcrest source JAR for easy reference	5 years ago
src	Fix dead link to the ant task in FAQ (#1478)	4 months ago
.classpath	Add resource source folders to Eclipse project	3 years ago

Clone with HTTPS ?  
Use Git or checkout with SVN using the web URL.  
<https://github.com/junit-team/junit4.git>

Open in Desktop   **Download ZIP**

## [Eclipse] 사용 개요

- ❑ 최근 버전(Indigo 이후 버전)은 Maven 플러그인이 내장
- ❑ Maven 프로젝트 생성
  - [File] - [New] - [Other] 를 클릭하여 프로젝트 생성 마법사 실행



## [Eclipse] 프로젝트 정보 입력

**New Maven Project**

**New Maven project**

❌ Enter a group id for the artifact.

**Artifact**

Group Id:

Artifact Id:

Version:

Packaging:

Name:

Description:

**Parent Project**

Group Id:

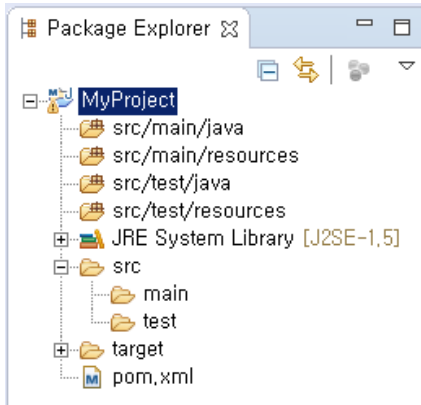
Artifact Id:

Version:

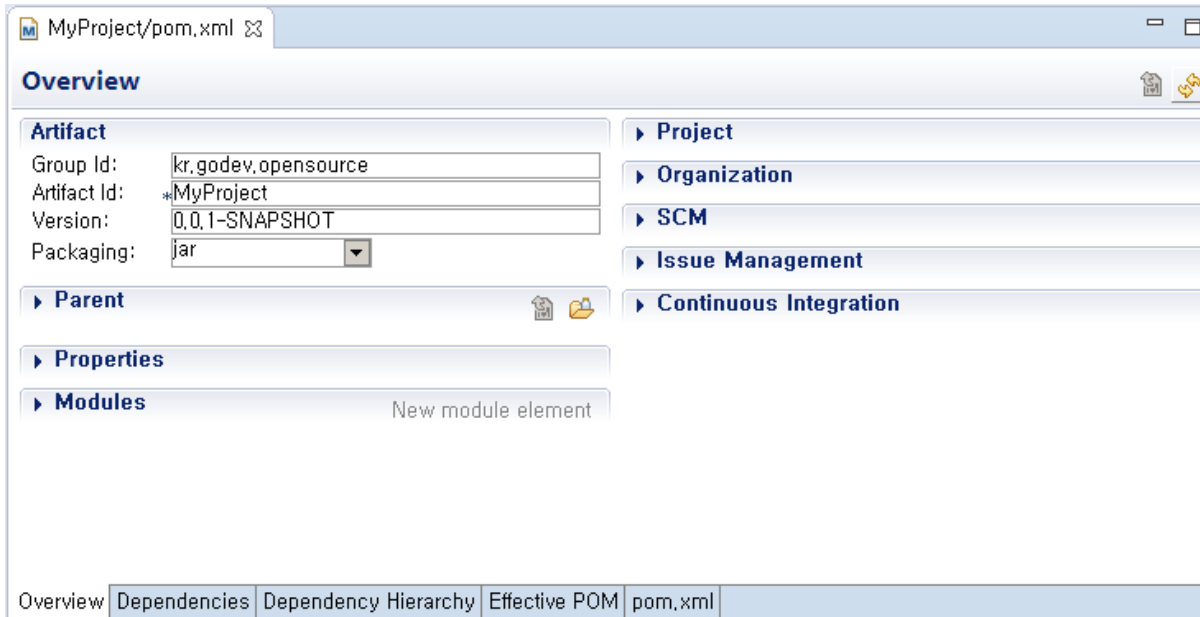
▶ **Advanced**

# [Eclipse] 프로젝트 생성 확인 및 pom 변경

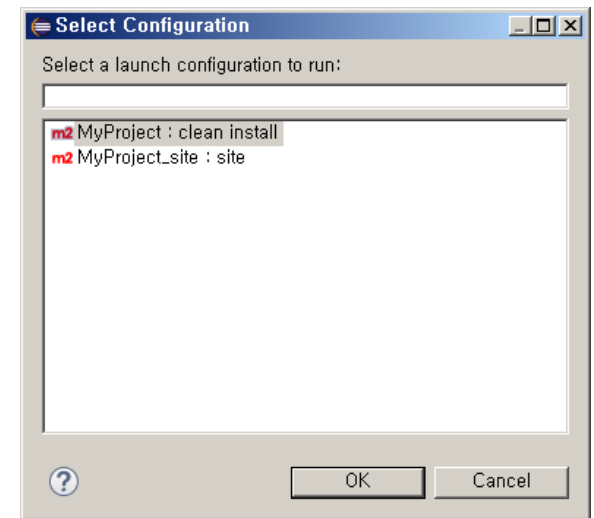
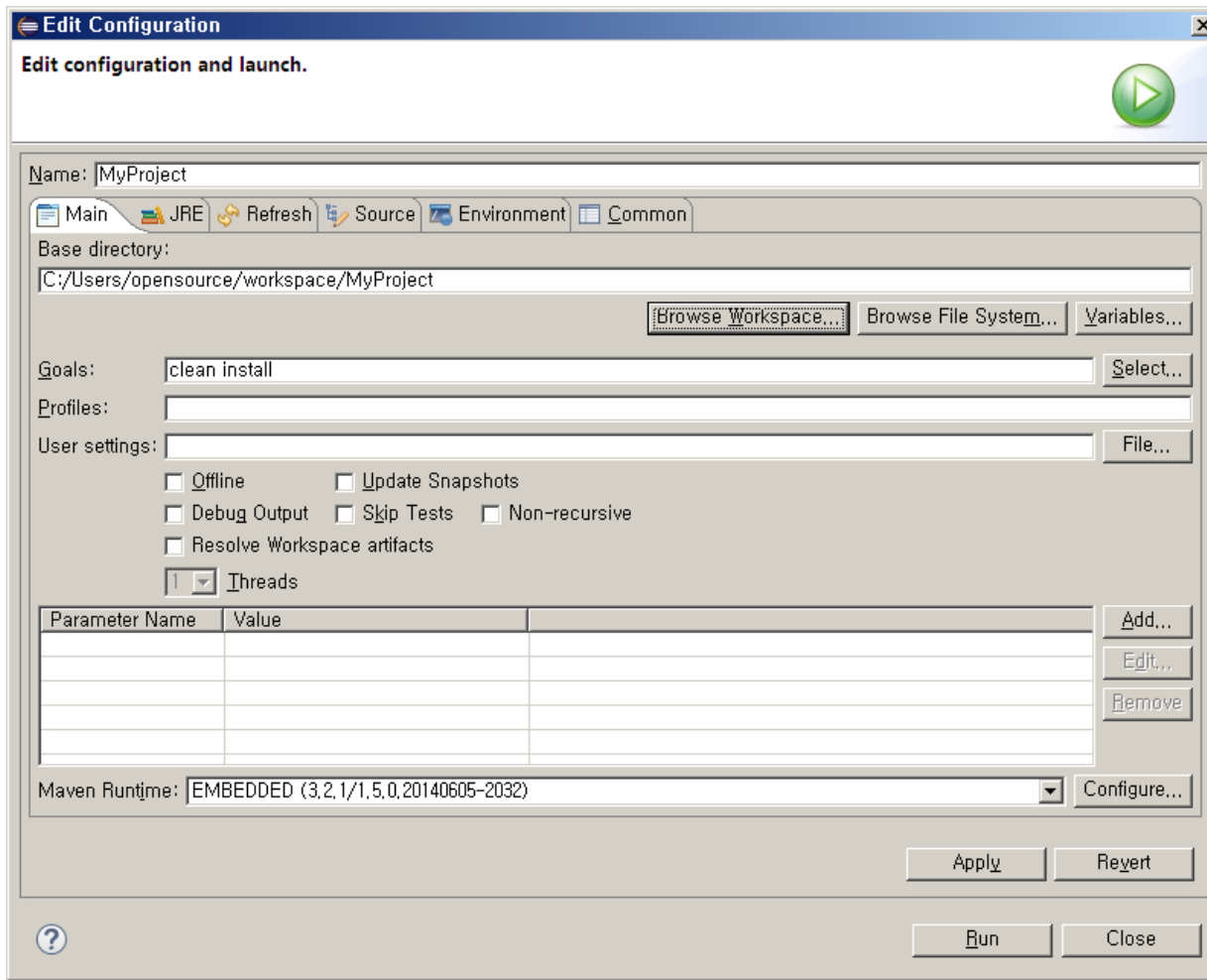
## □ 프로젝트 생성 확인



## □ pom 파일 변경



# [Eclipse] Build 설정 및 실행



## 3 Git



# 도구 소개

---

## □ 도구 개요

- 버전 관리의 대표 도구

## □ 목적 : 커뮤니케이션

- 표준: 산출물의 무결성 확보
- 소스코드 버전 관리
  - 마음의 평화: 실수해도 이전 버전으로 빠르게 되돌리기
- 왜 이렇게 수정하였지 확인
- 과거의 내가 얼마나 멋진 개발자였는지 확인

## □ 일반적인 사용 방법

- 별도의 저장소 운영 서버 사용
  - Git: GitHub, Bitbucket, Gitlab, GIBLIT
  - 통합 저장소: SCManager
- 클라이언트는 Core나 별도의 확장 도구 사용
  - Eclipse 플러그인: 기본 포함
  - Tortoise 시리즈
  - Sourcetree

# Git 개요

---

## □ 역사

- 2005년 리눅스 토발즈를 중심으로 개발
- 현재 GitHub에서 운영

## □ 사용 추세

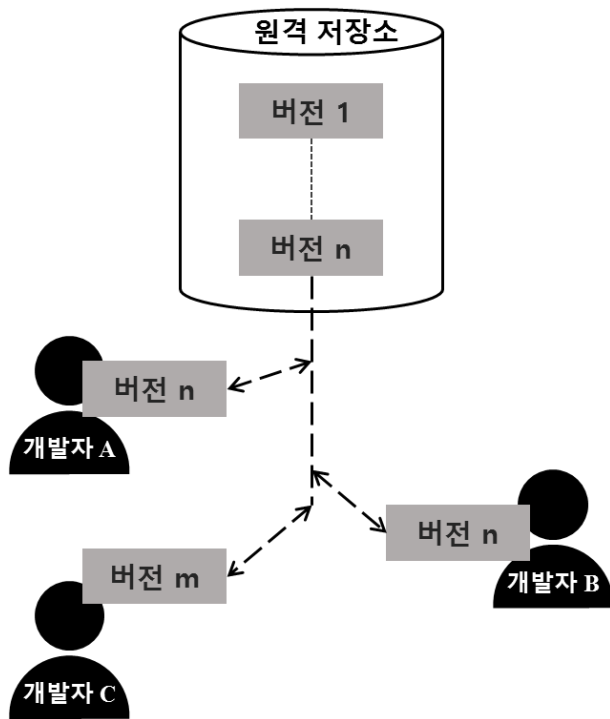
- 리눅스, Git을 포함하여 주요 오픈소스는 Git(GitHub) 사용
- 회사의 신규 프로젝트는 Git을 많이 사용하는 추세



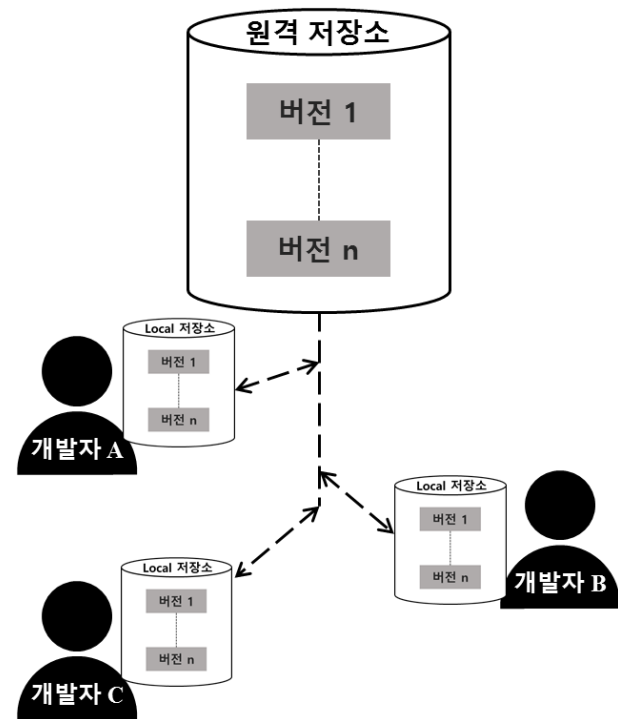
# 버전 관리 방식

## □ 분산 시스템 기반

- 원격 저장소와 동일한 로컬 저장소를 각 개발자가 유지
  - SVN은 최종 리비전만 각 개발자가 유지



Subversion



Git

## 기본 용어

영문	설명
Init	기존 폴더에 저장소를 생성
Clone	원격 저장소를 내 로컬에 복사
Push	내 저장소의 변경사항을 원격 저장소에 반영
Pull	원격 저장소의 변경사항을 내 저장소에 반영
Add	수정된 파일을 Staging Area에 등록하여 Staged 상태 만들
Check-out	작업할 브랜치를 지정
Commit	Staged 상태의 파일을 로컬 저장소에 반영
Remote Repository	공유를 위해 사용하는 저장소, 서버의 역할

# Git의 Branch

---

## ❑ 브랜치(Branch)

- 원본의 복사본
- 대부분의 VCS(Version Control System)에서 지원
- 용도
  - 특정 기능을 원본에 영향을 주지 않고 개발/테스트 하기 위해
  - 베이스라인을 설정하기 위해

## ❑ Subversion 브랜치 사용의 어려움

- 브랜치 생성 시, 원본을 특정 폴더에 복사함 (용량 \* 2)
  - 예) Visual C++의 3기가 소스코드의 단계 별/릴리즈 별 브랜치 10회 = 3기가 \* 10

## ❑ Git 브랜치의 용이함

- Git은 변경 사항을 저장하지 않고, 변경을 스냅샷 포인터로 관리
- 서버 자체를 복제하기 때문에, 포인터를 이용한 브랜치 생성 및 이동 가능
  - 예) Visual C++의 3기가 소스코드의 단계 별/릴리즈 별 브랜치 10회 = 40byte \* 10

## ❑ Git를 보다 잘 사용하고 싶다면, 브랜치를 어떻게 사용하는가에 좌우

## 브랜치 전략

---

### □ 업계에서 3개의 브랜치 전략을 추천

1. Subversion 처럼 사용하는 방식
2. GitFlow를 따르는 방식
3. Pull Request(GitHubFlow)를 따르는 방식

## 1) Subversion 처럼 사용하는 방식

---

### □ 방법

- Master 브랜치(기본 브랜치)를 공유하는 방법

### □ 장점

- Subversion 사용자가 쉽게 Git에 적용 가능

### □ 단점

- 브랜치를 사용하지 않음

### □ 추천하는 조직

- 기존 Subversion 등 중앙 집중식 도구에서 Git으로 넘어오는 조직
- 급격한 변화가 어려운 조직

## 2) GitFlow를 따르는 방식

---

### □ 방법

- GitFlow 도구를 설치하고, GitFlow의 프로세스를 준수
  - Git의 확장으로, Git의 브랜치 사용을 쉽게 적용할 수 있도록 확장 도구로 제공

### □ 장점

- 정형화된 브랜치 전략의 사용
- 참고할 수 있는 자료가 많음

### □ 단점

- 임베디드 SW보다 릴리즈가 잦은 웹 서비스 분야에 보다 적합
  - 임베디드 SW에도 적용이 안되는 것은 아님
  - K사의 사례

### □ 추천하는 조직

- 브랜치를 처음 적용하는 조직
- 중규모(협업 인원이 20명) 이하인 경우

## 2) GitFlow를 따르는 방식 - GitFlow

### □ Vincent Driessen가 개발한 브랜치 전략 및 지원 도구

- Sourcetree와 같은 Git 도구에서 기본 지원
- Master: 운영 환경과 동일한 소스코드가 있는 브랜치



### 3) Pull Request(GitHubFlow)를 따르는 방식

---

#### □ 방법

- 개발자가 저장소를 Fork(복제) 또는 Branch하여 수정하고, 변경 사항의 적용을 요청(Pull Request)
  - GitHub에서 오픈소스에 Contribute 하는 대표적인 방식

#### □ 장점

- 관리자의 검토 후, 개발자의 소스코드 반영 여부를 결정 가능

#### □ 단점

- Pull Request를 처리하기 위한 관리자 필요
- 원활한 사용을 위해서는 Pull Request를 지원하는 서버 도구 사용 필요

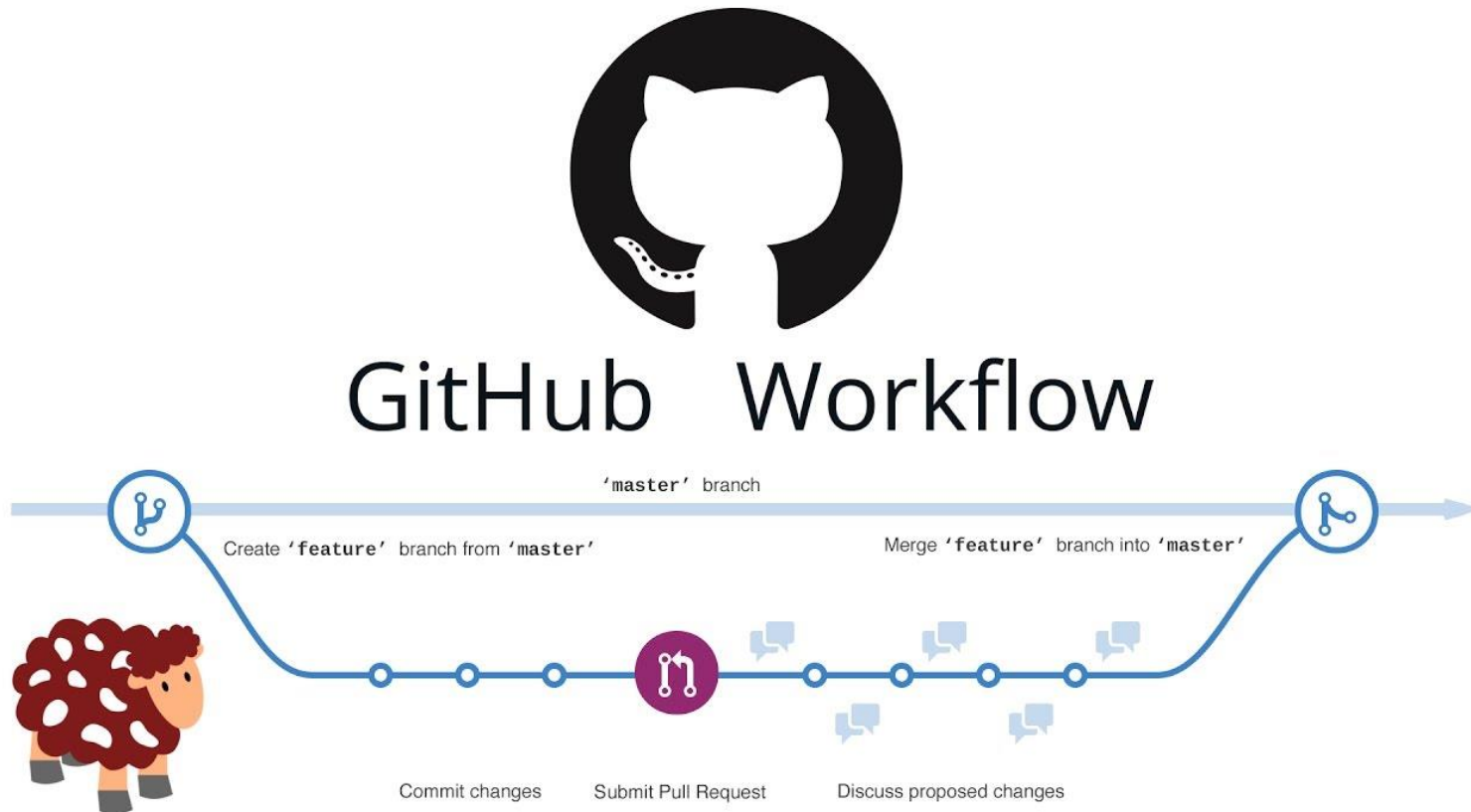
#### □ 추천하는 조직

- 협력 업체와 동일한 저장소로 공유하는 경우
- 대규모 개발인 경우



### 3) Pull Request(GitHubFlow)를 따르는 방식

- GitHub에서 운영되는 대부분의 오픈소스 개발에 적용



## [서버 설정] GitHub을 원격 저장소로 이용하기

### ❑ <https://github.com/>

- 회원 가입하면, public 저장소 무제한 생성 가능 (private 도 3개 가능)
  - 다른 사람에게는 소스코드가 보이지만, 커밋하는 사람을 지정할 수 있음
  - Private 저장소는 최소 월 \$7에서 시작

#### Create a new repository

A repository contains all the files for your project, including the revision history.

Owner

Repository name



handongjoon ▾

/

Great repository names are short and memorable. Need inspiration? How about **fictional-barnacle**.

Description (optional)



Public

Anyone can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

☐ Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▾

Add a license: **None** ▾



Create repository

## [저장소 Fork] JUnit4 저장소 Fork

### ❑ Fork: 다른 사람의 저장소를 수정하기 위해, 내 저장소로 복사해 오는 기능

- 향후 Pull Request를 통해 원래 저장소에 병합
- 오픈소스에서 Contribute에 많이 사용하는 방식

The screenshot shows the GitHub repository page for `junit-team / junit4`. The repository has 647 watches, 6,581 stars, and 2,513 forks. It is a Java-based testing framework with 2,195 commits, 5 branches, 20 releases, and 138 contributors. The license is EPL-1.0. The page shows the 'master' branch and a list of recent commits, including one by stefanbirkner about building with Oracle JDK 7 on Travis.

Repository: `junit-team / junit4`

Watch 647 | Star 6,581 | Fork 2,513

Code | Issues 106 | Pull requests 13 | Projects 0 | Wiki | Insights

A programmer-oriented testing framework for Java. <http://junit.org/junit4/>

2,195 commits | 5 branches | 20 releases | 138 contributors | EPL-1.0

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

stefanbirkner Don't build with Oracle JDK 7 on Travis anymore ... Latest commit a832c5a on 16 Oct

File	Commit Message	Time Ago
<code>.mvn/wrapper</code>	Build with Maven 3.1.1 (using Maven Wrapper)	2 months ago
<code>.settings</code>	Revert on request of kcooney:	4 years ago
<code>doc</code>	Build with Maven 3.1.1 (using Maven Wrapper)	2 months ago
<code>lib</code>	Add Hamcrest source JAR for easy reference	6 years ago
<code>src</code>	Fix dead link to the ant task in FAQ (#1478)	4 months ago
<code>.classpath</code>	Add resource source folders to Eclipse project	3 years ago
<code>.gitattributes</code>	Use Unix-style line endings. (#1405)	11 months ago
<code>.gitignore</code>	Use Unix-style line endings. (#1405)	11 months ago

## [저장소 Fork] 내 저장소에 Fork 확인

The screenshot shows the GitHub interface for a repository named 'handongjoon / junit4'. The repository is a fork of 'junit-team/junit4'. The page includes navigation tabs for Code, Pull requests, Projects, Wiki, Insights, and Settings. It also displays repository statistics: 2,196 commits, 5 branches, 20 releases, and 138 contributors. A table lists recent commits, including one by 'lexia q123' and others related to Maven Wrapper and settings.

Repository: **handongjoon / junit4** (forked from junit-team/junit4)

Unwatch 1 Star 0 Fork 2,513

Code Pull requests 0 Projects 0 Wiki Insights Settings

A programmer-oriented testing framework for Java. <http://junit.org/junit4/> [Edit](#)

[Add topics](#)

2,196 commits 5 branches 20 releases 138 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

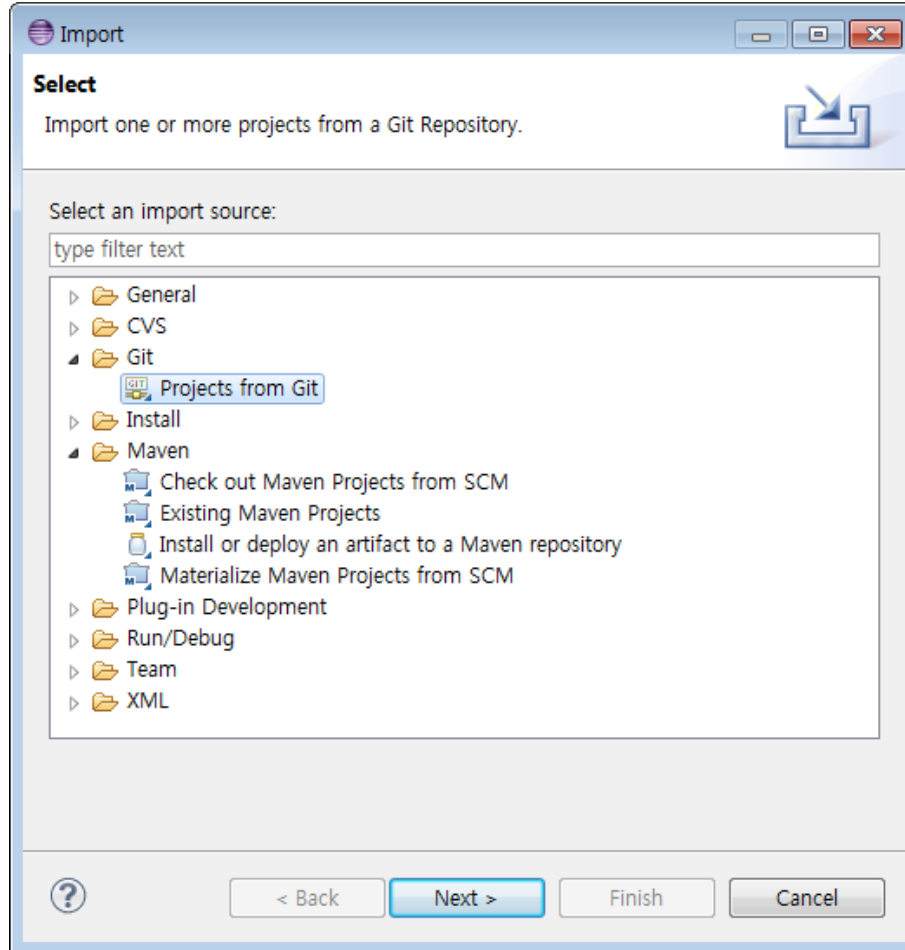
This branch is 1 commit ahead of junit-team:master. [Pull request](#) [Compare](#)

lexia q123	Latest commit db13b1a an hour ago
.mvn/wrapper	Build with Maven 3.1.1 (using Maven Wrapper) 2 months ago
.settings	Revert on request of kcooney: 4 years ago
doc	Build with Maven 3.1.1 (using Maven Wrapper) 2 months ago

## [Eclipse] 내 로컬로 Clone

### ❑ SVN과 동일하게, Import 기능 사용

- Maven 프로젝트 이므로, Maven 하위의 체크아웃 메뉴 이용



# [Eclipse] 내 로컬로 Clone

## ❑ Fork 한 저장소의 Clone을 위한 주소 받아오기

- ssh 설정이 되어있지 않다면, Github 권한으로 사용하기 위해 'Use HTTPS' 클릭

The screenshot shows the GitHub repository page for 'handongjoon / junit4', which is a fork of 'junit-team/junit4'. The repository has 2,196 commits, 5 branches, 20 releases, and 138 contributors. The 'Clone or download' button is highlighted, and a dropdown menu is open. The dropdown menu shows 'Clone with HTTPS' and 'Use SSH' (highlighted with a red box). Below this, it says 'Use Git or checkout with SVN using the web URL.' and provides the URL 'https://github.com/handongjoon/junit4.git' (also highlighted with a red box). At the bottom of the dropdown, there are buttons for 'Open in Desktop' and 'Download ZIP'.

handongjoon / junit4  
forked from junit-team/junit4

Unwatch 1 Star 0 Fork 2,513

Code Pull requests 0 Projects 0 Wiki Insights Settings

A programmer-oriented testing framework for Java. <http://junit.org/junit4/> Edit

Add topics

2,196 commits 5 branches 20 releases 138 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

This branch is 1 commit ahead of junit-team:master.

lexia q123

.mvn/wrapper Build with Maven 3.1.1 (using Maven Wrapper)

.settings Revert on request of kcooney:

doc Build with Maven 3.1.1 (using Maven Wrapper)

Clone with HTTPS ⓘ Use SSH

Use Git or checkout with SVN using the web URL.

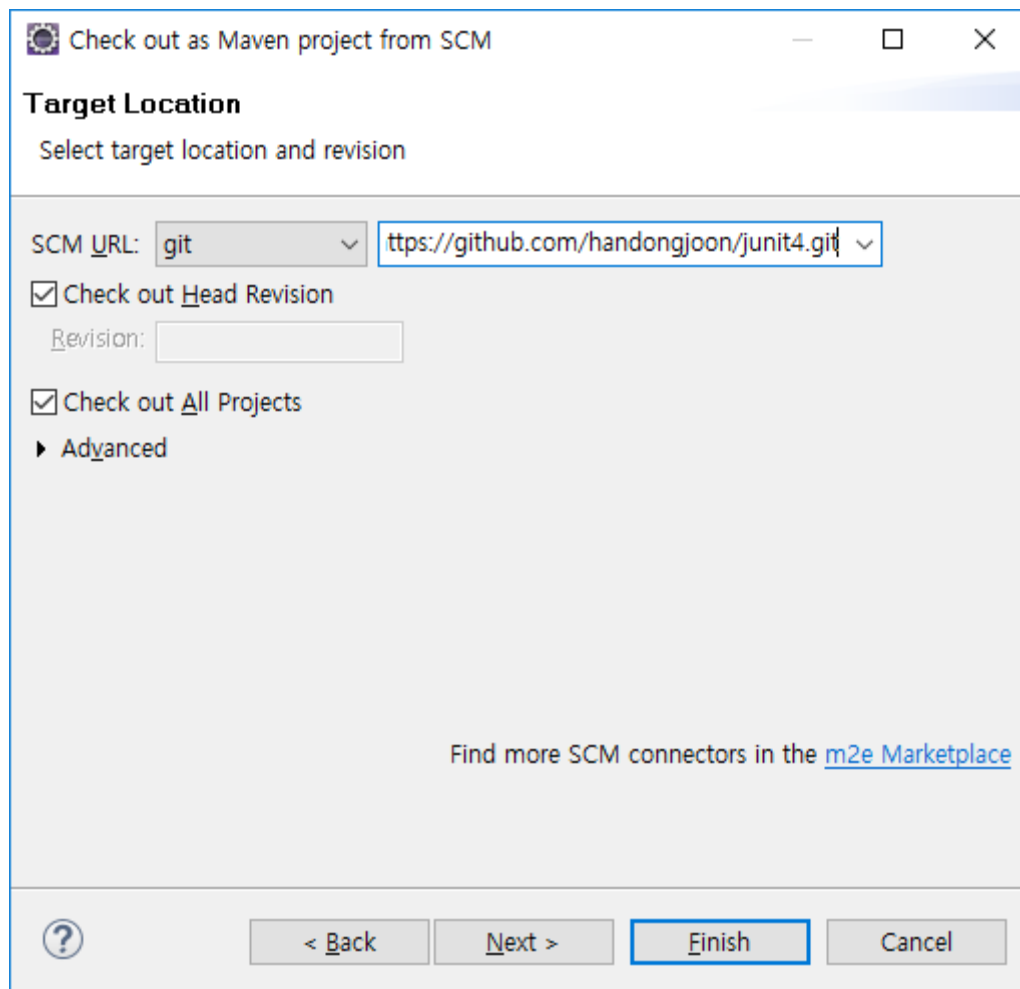
`https://github.com/handongjoon/junit4.git`

Open in Desktop Download ZIP

2 months ago

## [Eclipse] 내 로컬로 Clone

### ❑ Eclipse에 Clone 대상 저장소 위치 지정

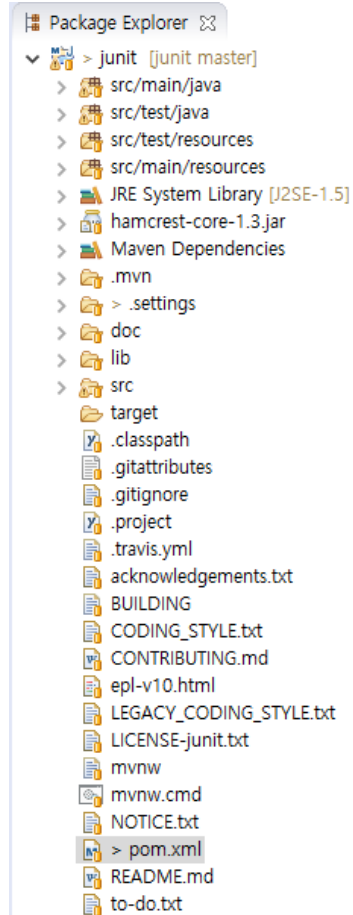


## [Eclipse] 소스코드 수정

### ❑ 에러가 나는 pom.xml의 플러그인 주석처리

- 저장 후, pom.xml 파일과 상위 폴더에 '>' 표시

```
193     </plugin>
194     <!--
195     <plugin>
196
197         <groupId>com.google.code.maven-replacer-plugin<
198         <artifactId>replacer</artifactId>
199         <version>1.5.3</version>
200         <executions>
201
202         </executions>
203         <configuration>
204             <ignoreMissingFile>false</ignoreMissingFile>
205             <file>${project.build.sourceDirectory}/juni
206             <outputFile>${project.build.sourceDirectory
207             <regex>false</regex>
208             <token>@version@</token>
209             <value>${project.version}</value>
210         </configuration>
211     </plugin>-->
212     <plugin><!-- Using jdk 1.5.0_22, package-info.java
213     <!--
214         java compiler plugin forked in extra process
```

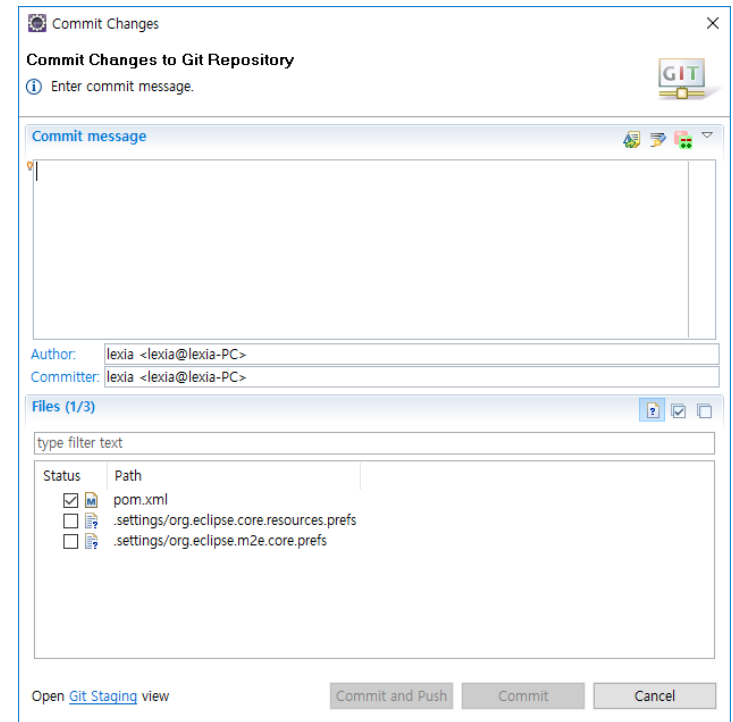
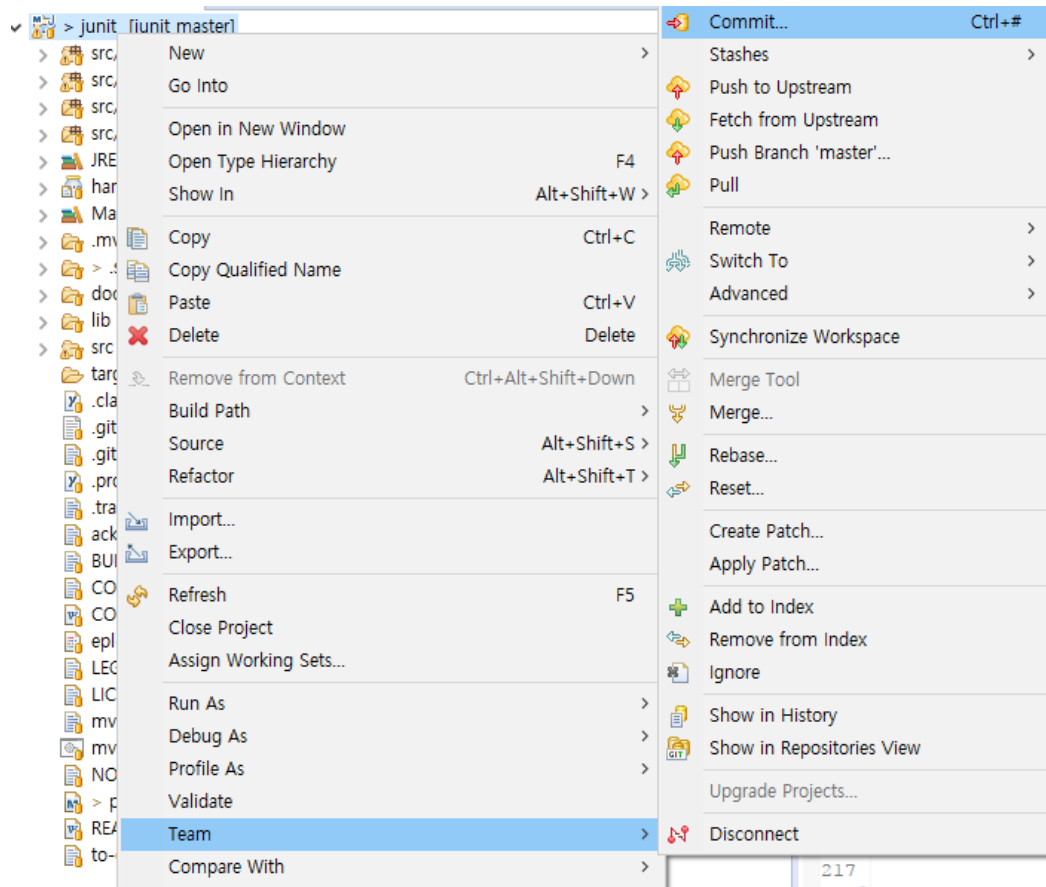




# [Eclipse] 변경내용 Commit

## ❑ Team – Commit 실행

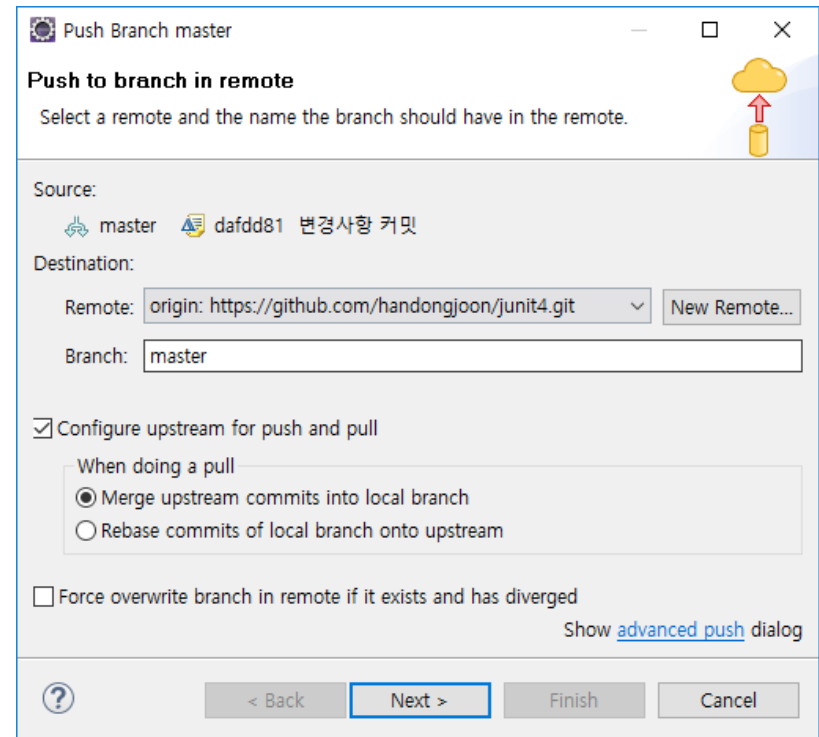
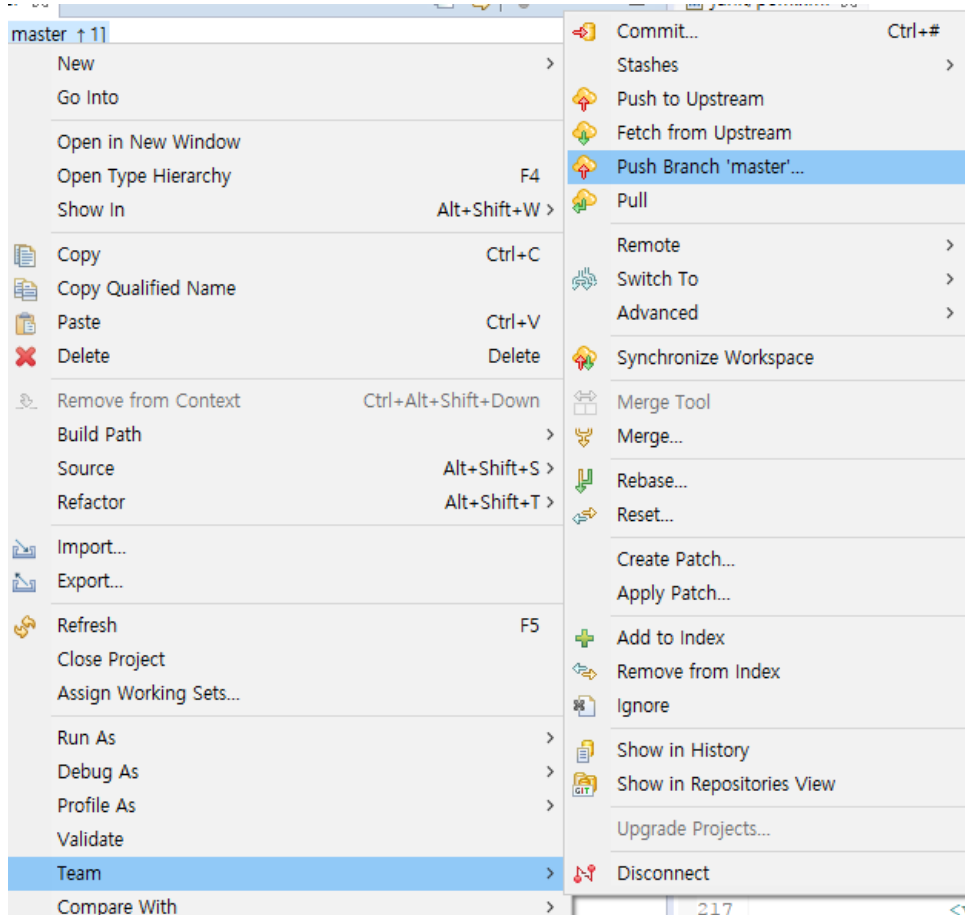
- SVN과 유사하게 커밋 메시지 작성 부분 표시
- 커밋 대상 선정 가능



# [Eclipse] 원격 저장소로 Push

## ❑ Commit 내용 원격 저장소로 Push 하기

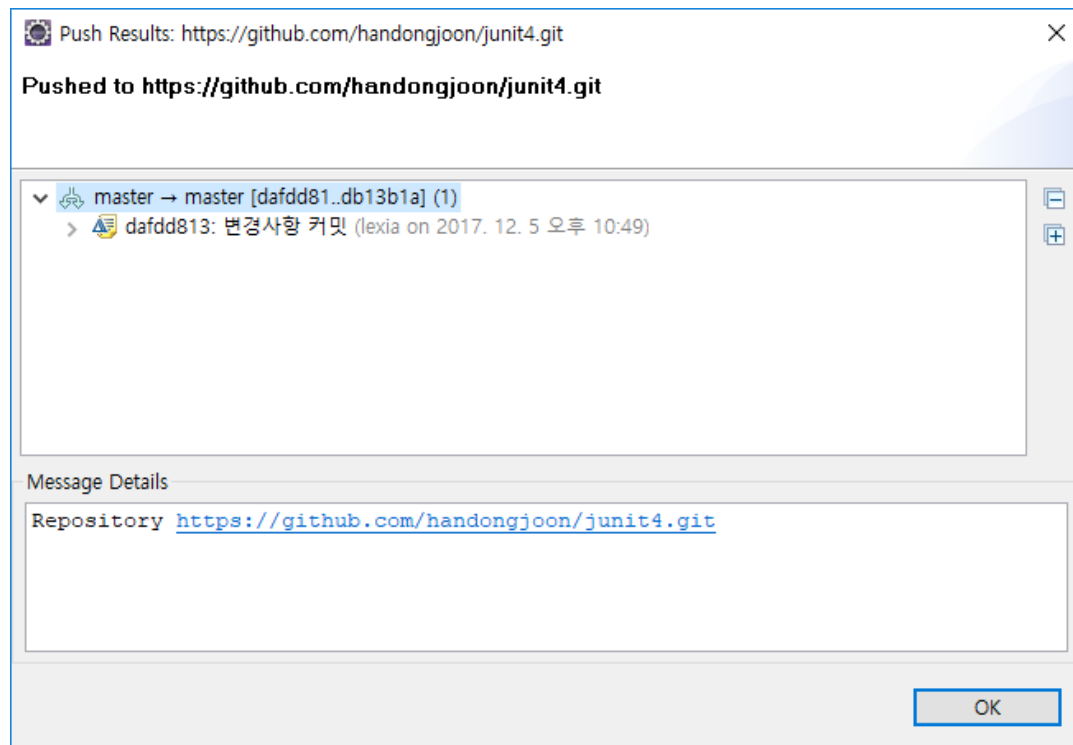
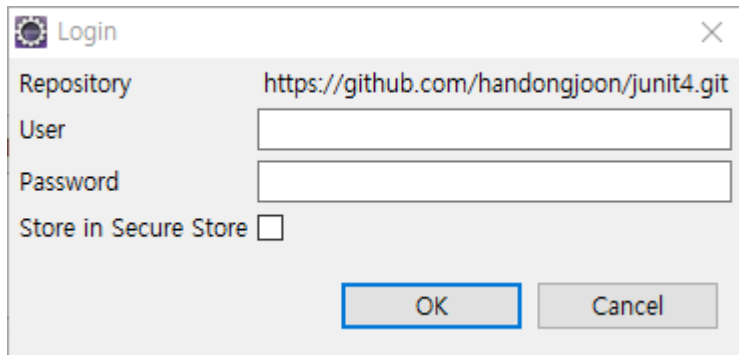
- 현재 작업 중인 master 브랜치 Push



## [Eclipse] 원격 저장소로 Push

### ❑ Github 계정으로 권한 확인

- Github의 내 저장소에서 변경내용 적용 확인



## 4 Jenkins



## 지속 통합(CI: Continuous Integration)

- ✓ 팀 구성원들이 자신이 한 일을 자주 통합하는 소프트웨어 개발 실천 방법
- ✓ 각 통합은 자동화된 빌드를 검증하여 최대한 빨리 통합 오류를 탐지
- ✓ 하루에 여러 번 통합 빌드를 수행하는 것

- 마틴 파울러

소프트웨어 통합 오류를 개발 초기부터 예방하는 것  
소프트웨어 통합을 위해 현존하는 가장 훌륭한 전략

---

# 통합 지옥

# Integration HELL



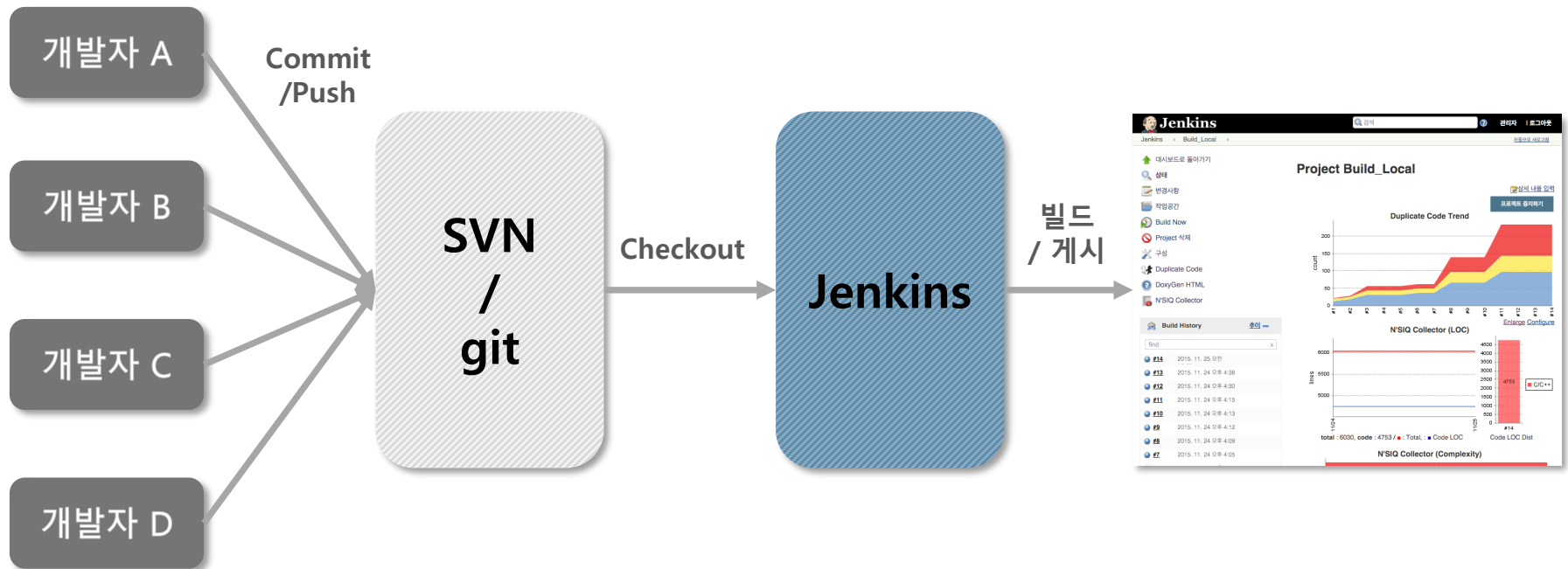
# Jenkins

<http://www.jenkins-ci.org>

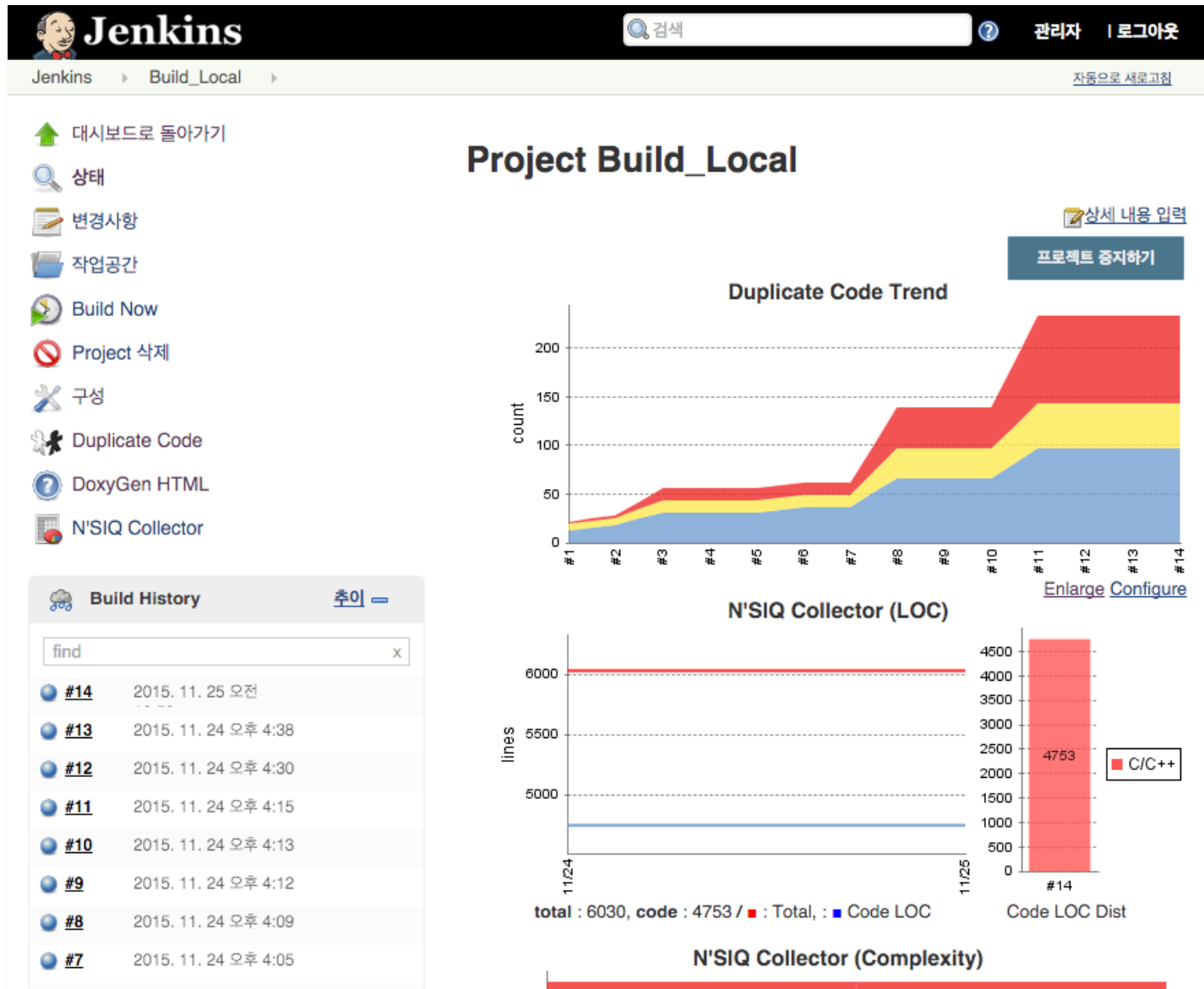
- ✓ 웹 기반 오픈소스 CI 도구
- ✓ 점유율 약 70%
- ✓ 1,300 개의 플러그인

## 지속적 통합 - 동작 방식

- ① 버전관리 도구에서 **최신 리비전을 체크아웃** 받아
- ② 주어진 **명령대로 빌드**하여
- ③ 결과를 **게시/전달**함







## [설치]

### ❑ Tomcat(WAS) 상에 Jenkins WAR 파일을 실행하는 방식

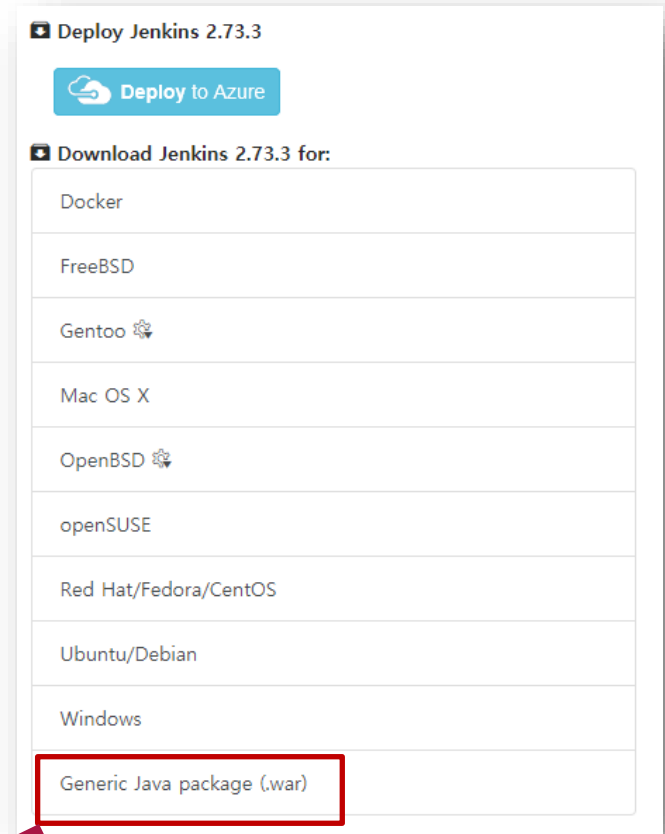
- 단독 실행(standalone)이 가능한 패키지도 제공
- Docker 이미지 및 Azure 배포 제공

### ❑ 다운로드

- <https://jenkins.io/download/>

### ❑ WAR 복사

- 이름을 jenkins.war 로 변경
- jenkins.war 를 Tomcat의 webapp 폴더로 이동
- Tomcat 재시작
- 최초 시작을 위한 특별 ID 입력 필요



## [설정] Initial Password 입력

### ❑ Docker의 영구 데이터(Volume)에서 확인

Getting Started

# Unlock Jenkins

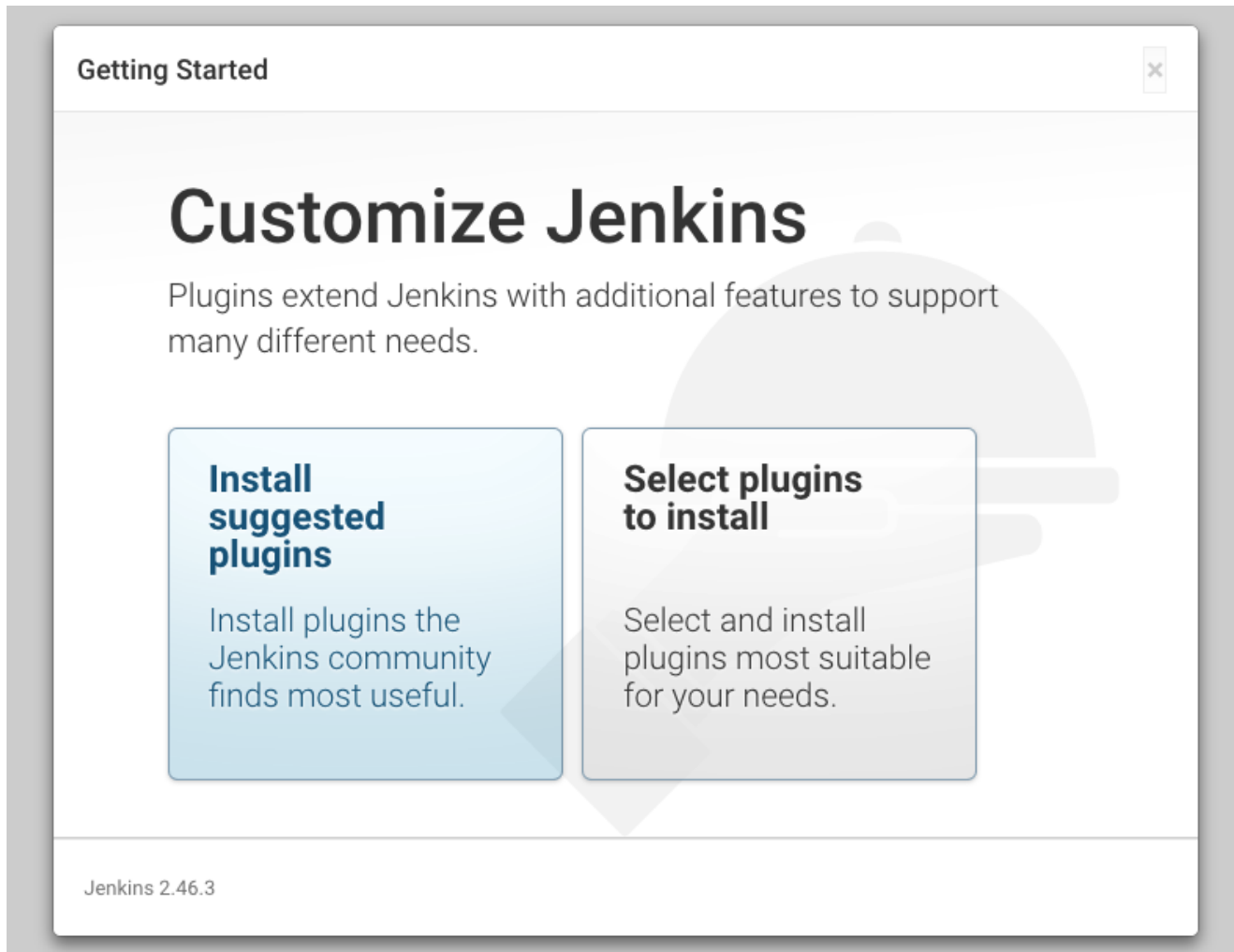
To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

```
/root/.jenkins/secrets/initialAdminPassword
```


Please copy the password from either location and paste it below.

Administrator password

Continue




## [설정] 초기화면


 **Jenkins**


검색


자동으로 새로고침


Jenkins >


 새로운 Item

 사람

 빌드 기록

 Jenkins 관리

 Credentials

 상세 내용 입력

### Jenkins에 오신 것을 환영합니다.


시작하려면 **새 작업**를 만들어 주시기 바랍니다.

빌드 대기 목록

빌드 대기 항목이 없습니다.

빌드 실행 상태

1 대기 중  
2 대기 중

 이 페이지 한글화 도와주기

페이지 생성일시: 2016. 2. 11 오후 5시 46분 07초

[REST API](#)

Jenkins ver. 1.647

## [설정] Jenkins 관리

#	메뉴	설명
1	시스템 설정	Jenkins 실행에 대한 전반적인 환경을 설정한다. 초기 화면에 보여질 시스템 메시지, 전역 변수, Maven, JDK 설정 등이 포함된다.
2	Configure Global Security	Jenkins 전반적인 보안 항목을 설정한다. 기본 값은 보안을 사용하지 않는 것이다. LDAP 사용 여부, ID 사용 여부 및 권한이 포함된다.
3	플러그인 관리	약 840여개의 플러그인을 설치하고 업데이트 할 수 있다.
4	노드 관리	빌드 하기 위한 Master(Jenkins가 설치된 서버)와 Slave(빌드만 전용으로 하는 머신)를 설정한다.
5	사용자 관리	Jenkins 사용자를 설정하는 기능으로, 'Configure Global Security'의 설정 결과에 따라 이 메뉴가 표시된다.  최초 설치 후에는 표시되지 않는다.

## [설정] 시스템 기본 설정

---

### □ 홈 디렉토리

- 빌드 Job을 설정하고 빌드할 때 소스 코드 저장소에서 체크아웃한 파일이 저장되고, 빌드되는 로컬 디렉토리의 위치
- 특정 위치로(예를 들면 D:\w 등) 설정하고 싶다면, Jenkins 설치 전에 설정 필요
  - web.xml의 HUDSON\_HOME 변경

### □ # of executors

- Master에서 동시에 빌드하는 Job의 개수

### □ SCM checkout retry count

- 소스 코드 저장소에서 체크아웃을 실패했을 때 재시도 하는 횟수
- 기본값인 0은 체크아웃이 실패해도 빌드가 실패한 것으로 간주

### □ Help make Jenkins better by sending anonymous usage statistics and crash reports to the Jenkins project.

- Jenkins 개선을 위한 정보 보내기
- 오픈소스 프로젝트에 기여하는 하나의 방법

## [설정] Jenkins Location 설정

---

### ❑ Jenkins URL

- Jenkins 접속 주소. 반드시 IP나 URL로 변경 필요

### ❑ System Admin e-mail address

- Jenkins에서 사용자에게 알림 메일을 보낼 때 사용할 관리자의 e-mail 주소

#### Jenkins Location

---

Jenkins URL

http://localhost:8080/jenkins/



 Please set a valid host name, instead of localhost

System Admin e-mail address

address not configured yet <nobody@nowhere>

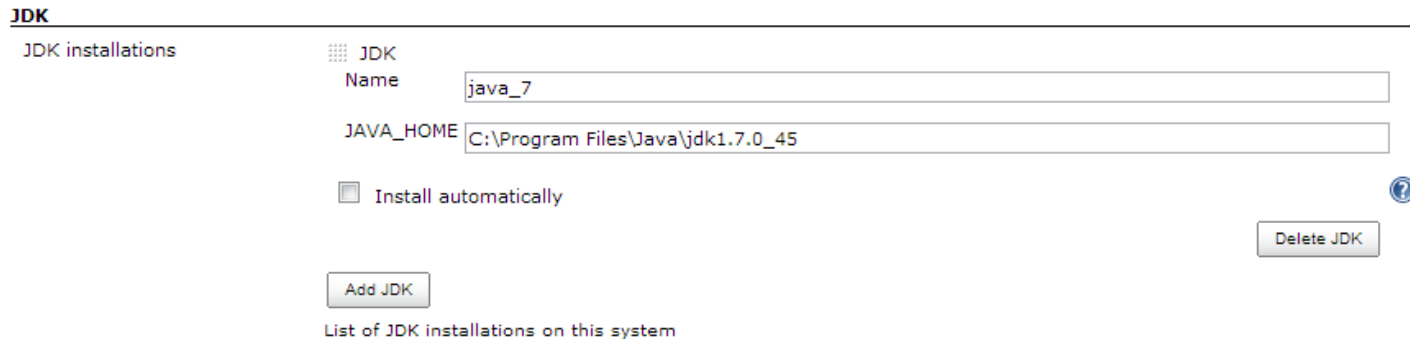




## [설정] 빌드 도구 설정

### □ JDK 설정

- 'Add JDK'를 클릭해 JDK 추가
- 자동으로 특정 버전을 추가하거나 JDK 설치 후 JAVA\_HOME의 위치를 지정



### □ Maven 설정

- JDK와 동일한 방식으로 추가

## [설정] Configure Global Security (권한) 설정

- ❑ 권한 없어도 접근 가능한 수준부터, 프로젝트 별 권한까지 설정 가능
- ❑ 사용 권한을 설정하기 위해서는 'Enable security'를 선택

☒ Enable security [?](#)

TCP port for JNLP slave agents ☐ Fixed :  ☒ Random ☐ Disable [?](#)

Disable remember me ☐ [?](#)

Access Control

**Security Realm**

☐ Delegate to servlet container [?](#)

☐ Jenkins' own user database [?](#)

☐ LDAP

**Authorization**

☒ Anyone can do anything [?](#)

☐ Legacy mode [?](#)

☐ Logged-in users can do anything [?](#)

☐ Matrix-based security [?](#)

☐ Project-based Matrix Authorization Strategy [?](#)

## [설정] Configure Global Security (권한) 설정

---

### ❑ Security Realm

- 권한이 어디에 위치하는지 지정하는 것으로 Jenkins 내부의 DB에 저장할 수도 있고, 다른 곳의 권한을 사용할 수도 있음
  - Delegate to servlet container: 컨테이너, 즉 Tomcat과 같은 WAS의 권한을 위임받아 사용
  - Jenkins' own user database: Jenkins의 내부 데이터 베이스에 아이디 정보 저장
  - LDAP: 사용자 계정 서버와 연결하여 사용
- 사용자 계정 서버가 없는 경우는 두 번째 선택값인 Jenkins 내부의 DB를 사용


## [설정] Configure Global Security (권한) 설정

### ❑ Authorization

- Anyone can do anything: 로그인 여부와 상관없이, Jenkins에 접속하는 누구나 Jenkins의 설정을 포함한 모든 기능을 사용할 수 있게 한다. 기본값이지만, 기본 설정 후 보안을 위해 반드시 다른 항목으로 변경해야 한다.
- Legacy mode: Tomcat과 같은 WAS의 admin 권한을 이용하여, admin만 모든 기능을 사용할 수 있고 그 외에는 읽기 전용 권한을 부여한다.
- Logged-in users can do anything: 1번 항목과 비슷하나, 로그인한 사용자가 모든 기능을 사용할 수 있게 한다.
- Matrix-based security: 역할이나 사용자에게 따라 권한을 설정한다. 기본값으로 'Anonymous'가 포함되나, 다음 그림과 같이 'Add' 버튼을 이용해서 사용자나 그룹을 추가할 수 있다.

#### Authorization

- ☐ Anyone can do anything
- ☐ Legacy mode
- ☐ Logged-in users can do anything
- ☒ Matrix-based security

User/group	Overall						Credentials				Slave											
	Administer	Configure	Update	Center	Read	Run	Scripts	Upload	Plugins	Create	Delete	Manage	Domains	Update	View	Build	Configure	Connect	Create	Delete	Disc	connect
Anonymous	<input type="checkbox"/>		<input type="checkbox"/>		<input type="checkbox"/>		<input type="checkbox"/>		<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>
 admin	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>

User/group to add:

Add

- ☐ Project-based Matrix Authorization Strategy

## [설정] Configure Global Security (권한) 설정

### ❑ Authorization (계속)

- Project-based Matrix Authorization Strategy: Matrix-based security와 유사하나, 각 Job 설정에서 다시 한 번 Job에 대한 권한을 설정할 수 있다. 예를 들어, A 프로젝트에서 빌드하는 Job과 B 프로젝트에서 빌드하는 Job이 다르고, A와 B 프로젝트 팀원은 다른 팀의 Job은 볼 수 없도록 설정할 때 사용한다. 첫 번째 그림과 같이 사용자나 그룹 별 권한을 설정하고, 두 번째 그림과 같이 Job 설정 화면에서 권한을 다시 한 번 설정할 수 있다.

#### Authorization

- ☐ Anyone can do anything
- ☐ Legacy mode
- ☐ Logged-in users can do anything
- ☐ Matrix-based security
- ☒ Project-based Matrix Authorization Strategy

User/group	Overall					Credentials				
	Administer	Configure	Update	Center	Read	Run	Scripts	Upload	Plugins	Create
admin	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Anonymous	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

User/group to add:

Add

☒ Enable project-based security

☐ Block inheritance of global authorization matrix

User/group	Credentials					Job					Run	SCM
	Create	Delete	Manage	Domains	Update	View	Build	Cancel	Configure	Delete	Discover	Read
Anonymous	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
admin	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

User/group to add: admin

Add

## [설정] Configure Global Security (권한) 설정

---

### □ 주의할 점

- Jenkins를 설치한 후 'Jenkins's own user database'을 이용하여 사용 권한을 처음 설정할 때에는, 다음의 순서로 진행하자. 설치 후에는 Jenkins 사용자 DB에 따로 사용자가 없기 때문에, 만약 'Anyone can do anything' 외에 다른 항목으로 설정한다면 아이디가 없기 때문에 로그인을 못해서 아무 작업도 못할 수 있다.
  1. 'Security Realm'에서 'Jenkins' own user database'를 선택한다
  2. 'Authorization'에서 'Anyone can do anything'를 선택하고 적용한다.
  3. 'Jenkins 관리'의 'Manage Users'에서 'admin' 또는 원하는 관리자 계정을 생성한다.
  4. 다시 'Configure Global Security' 메뉴로 돌아와서, 'Authorization' 항목의 다른 권한 설정을 선택한다.
  5. 단, Matrix-based security 등을 선택 시에는, 3번에서 생성한 관리자 계정을 'Add'로 추가하고 필요한 관리자 권한을 부여한다.

## [설정] 플러그인 관리

- ❑ 플러그인의 추가/삭제/업데이트 설정
- ❑ 플러그인 추가 후, Jenkins 재시작이 필요할 수 있음

업데이트된 플러그인 목록

설치 가능

설치된 플러그인 목록

고급

설치	이름 ↓	버전	설치됨
<input type="checkbox"/>	<a href="#">SCM API Plugin</a> This plugin provides a new enhanced API for interacting with SCM systems.	1.1	1.0

지금 다운로드하고 재시작 후 설치하기

Update information obtained: 5 sec ago


지금 확인

## [Job 설정] Job 추가


### □ 메인화면의 '새로운 Item' 클릭

- 최신 버전은 Maven 프로젝트가 Freestyle 프로젝트에 포함


**Enter an item name**  
  
» This field cannot be empty, please enter a valid name




**Freestyle project**  
이것은 Jenkins의 주요 기능입니다. Jenkins은 어느 빌드 시스템과 어떤 SCM(형상관리)으로 묶인 당신의 프로젝트를 빌드할 것이고, 소프트웨어 빌드보다 다른 어떤 것에 자주 사용될 수 있습니다.




**Maven project**  
Maven 프로젝트를 빌드합니다. Jenkins은 POM 파일의 이점을 가지고 있고 급격히 설정을 줄입니다.



**Pipeline**  
Orchestrates long-running activities that can span multiple build slaves. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.




**External Job**  
이 유형의 작업은 원격 장치처럼 Jenkins 외부에서 동작하는 프로세스의 실행을 기록하는 것을 허용합니다. 그렇게 설계되어서, 기존의 자동 시스템의 대시보드로서 Jenkins을 사용할 수 있습니다.



**Multi-configuration project**  
다양한 환경에서의 테스트, 플래폼 특성 빌드, 기타 등등 처럼 다수의 서로다른 환경설정이 필요한 프로젝트에 적합함.

if you want to create a new item from other existing, you can use this option:



Copy from

OK



## [Job 설정] 설명 설정

---

### □ Job에 대해 알리고 싶은 내용을 작성





이름

Maven Project

설명

[Plain text] [미리보기](#)

## [Job 설정] 기본 설정

- ☐ 오래된 빌드 삭제 
- ☐ 이 빌드는 매개변수가 있습니다 
- ☐ 빌드 안함 (프로젝트가 다시 빌드를 할 때까지 새로운 빌드가 실행되지 않습니다.) 
- ☐ 필요한 경우 concurrent 빌드 실행 

설정 항목	설명
오래된 빌드 삭제	이 항목을 설정하지 않으면 Jenkins는 빌드 결과를 계속 유지한다. 그러나 하드디스크 용량 등을 고려했을 때, 지난 빌드 결과는 삭제하는 것이 필요할 수도 있다. 이 항목을 선택하면 특정 기간이나 빌드 수만큼 빌드 결과를 보관할 수 있다.
이 빌드는 매개변수가 있습니다.	빌드 시 매개변수를 정의하고 사용할 수 있다. 매개변수는 단순히 Text도 가능하지만, Subversion의 tag 목록도 가능하다.
빌드 안함	이 항목을 선택하면 빌드가 수행되지 않는다. 모든 빌드 설정은 유지하면서, 잠시 빌드를 사용하지 않을 때 활용한다.
필요한 경우 concurrent 빌드 실행	Jenkins는 하나의 Job을 빌드 중 또 다시 해당 빌드를 시작하면 Queue에 쌓이게 된다. 그러나 Queue에는 1개까지만 유지되고, 나머지 빌드는 무시된다. 이 항목을 선택하면 빌드 머신이 가능한 경우 동시 빌드를 수행한다.

## [Job 설정] 소스코드 관리

### □ 빌드를 위해 소스코드 저장소에서 소스코드를 가져오는 설정

소스 코드 관리

☐ None

☐ CVS

☐ CVS Projectset

☐ Git

☒ Subversion

Modules

Repository URL

?

Repository URL is required.

Local module directory (optional)

?

Repository depth

infinity

▼

?

Ignore externals

☐

?

Add more locations...

Check-out Strategy

Use 'svn update' as much as possible

▼

Use 'svn update' whenever possible, making the build faster. But this causes the artifacts from the previous build to remain when a new build starts.

Repository browser

(자동)

▼

?

고급...

## [Job 설정] 소스코드 관리

### □ URL에 저장소 주소 입력

- 권한이 필요한 저장소라면, 계정 정보 추가 입력

### □ 체크아웃 전략 설정(SVN의 경우)

전략	설명
Use 'svn update' as much as possible	최초 빌드 또는 최초 빌드가 아니더라도 체크아웃한 기록이 없을 경우 체크아웃을 하고, 그 이후 빌드에는 업데이트 명령만 수행한다. 빌드를 빠르게 진행할 수 있지만, 과거 빌드의 결과가 남아있을 수 있다. 이 메뉴의 기본 값이다.
Always check out a fresh copy	빌드 시작 시 기존에 체크아웃 받은 내용과 빌드 결과를 삭제하고, 새롭게 체크아웃한다. 매번 새롭게 체크아웃하므로 빌드 시간은 오래 걸리나, 클린 빌드가 필요한 경우 사용한다.
Emulate clean checkout by first deleting unversioned/ignored files, then 'svn update'	Always check out a fresh copy과 유사한 결과를 빠르게 만들기 위한 방법으로, 먼저 Subversion 관리를 받지 않는 파일을 삭제하고, 업데이트를 수행한다.
Use 'svn update' as much as possible, with 'svn revert' before update	Use 'svn update' as much as possible과 유사하지만, 업데이트 전 Subversion의 revert 명령을 수행한다. revert 명령은 체크아웃 후 수정이 있을 경우, 모든 수정을 체크아웃 받은 상태로 '되돌아 가도록' 만든다.

## [Job 설정] 빌드 유발

### □ 빌드를 언제 수행할 것인지 지정

#### 빌드 유발

- ☐ Build after other projects are built
- ☐ Build periodically
- ☐ Poll SCM



항목	설명
Build after other projects are built	다른 Job의 빌드가 완료되면 이 빌드를 시작한다. 예를 들어 Library란 Job의 빌드가 완료된 후 이 빌드를 시작하려면, 'Projects to watch' 항목에 Job 이름인 Library를 입력하면 된다.
Build periodically	정해진 일정에 따라 빌드를 진행한다. 일정 지정 형식은 cron 표기법과 유사하다.
Poll SCM	정해진 일정에 따라 소스 코드 저장소에서 변경 사항이 있는지 확인한다. 변경이 있을 경우 빌드를 진행한다. 일정 지정 형식은 cron 표기법과 유사하다.

## [Job 설정] 빌드 유발

---

### □ Build periodically과 Poll SCM의 일정 지정은 cron 표기법을 사용

- TAB이나 공백으로 구분된 5자리
  - 분 시 일 월 요일순
- 예) 매일 새벽 5시에 빌드를 진행
  - 0 5 \* \* \*
- 예) 15분 단위로 소스 코드 저장소의 변경 사항을 확인
  - H/15 \* \* \* \*
- cron 표기법 참고
  - <https://en.wikipedia.org/wiki/Cron>

## [Job 설정] Maven 프로젝트

### ❑ Build 에서 'Invoke top-level Maven Target' 선택

**Build**

Invoke top-level Maven targets

Maven Version

(Default)

▼

Goals

▼

고급...

×

?

Add build step ▼

## [Job 실행] 빌드 결과 확인

- ❑ Jenkins 메인화면에서 각 Job을 클릭하면 Job 화면으로 이동
- ❑ Job이 정상적으로 빌드되었는지 결과만 확인하려면, 왼쪽 하단의 Build History를 확인
  - #1, #2 순으로 빌드가 표시
  - 파란색 공일 경우 빌드 성공
  - 빨간색 공일 경우 빌드 실패

대시보드로 돌아가기

상태

변경사항

작업공간

Build Now

Maven project 삭제

구성

Modules

### Maven project MVN\_Test

상세 내용 입력


프로젝트 중지하기


Workspace

Recent Changes



### 고정링크

- [Last build. \(#1\). 19 sec 전](#)
- [Last stable build. \(#1\). 19 sec 전](#)
- [Last successful build. \(#1\). 19 sec 전](#)
- [Last completed build. \(#1\). 19 sec 전](#)

 Build History 추진

 #1


2016. 3. 29 오전 1:16

 RSS (전체)  RSS (실패)





## [Job 실행] 빌드 실패 원인 확인

### □ 각 빌드 #를 클릭하고, Console Output 확인

 프로젝트로 돌아가기


 상태

 바뀐점

 Console Output

 빌드 정보 수정


 이 빌드를 삭제

 Tag this build

 Redeploy Artifacts

 See Fingerprints

 빌드 #1 (2016. 3. 29 오전 1:16:57)

 Revision: 3  
No changes.

 사용자 [관리자](#)에 의해 시작됨

#### Module Builds

 [my\\_project](#) 1.6 sec

# End of Document

---

**Any Question?**