Software Verification

Junit & IntelliJ 및 빌드 환경

201311276 박형민

201311287 엄현식

201311318 최정헌

201311320 한예훈

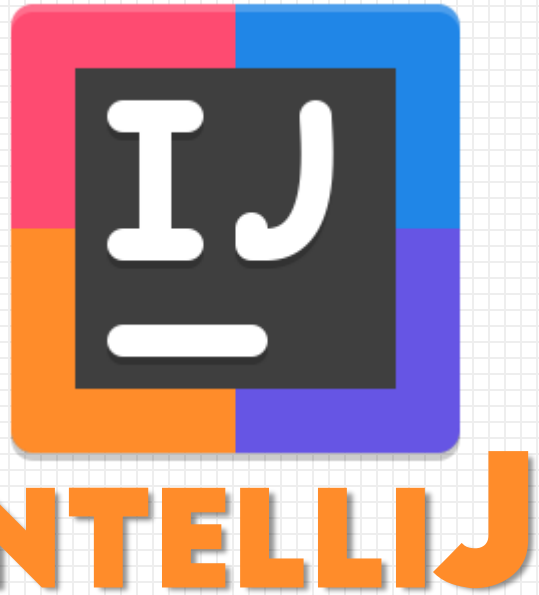
INDEX

- IDE
- Unit Test
- Build
- CI

IDE

IDE란?

효율적으로 소프트웨어를 개발하기 위한 통합개발환경 소프트웨어 어플리케이션 인터페이스



INTELLIJ

ECLIPSE

IDE

왜 INTELLIJ 인가?

IntelliJ 와 Eclipse 비교

VS

장점

안정적인 Plug-in 지원

IDE의 안정성

개발 퍼포먼스가 높음

단점

불안정적인 Plug-in

불안한 IDE - 호환성 문제

개발 퍼포먼스가 낮음

IntelliJ 와 Eclipse 비교

VS

단점

유료

프로젝트 단위로 관리

익숙하지 않음

초기 진입 장벽이 비교적 높음

장점

무료

워크스페이스 단위로 관리

대체적으로 익숙함

진입장벽이 비교적 낮음

이래서 INTELLIJ!

유료 학생용 라이선스 사용하면 무료!

~~프로젝트 단위로 관리~~ Critical 하지 않음!

~~익숙하지 않고,~~

~~초기 진입 장벽이 있는 편.~~

적응만 하면 해결 !

INTELLIJ 설치방법

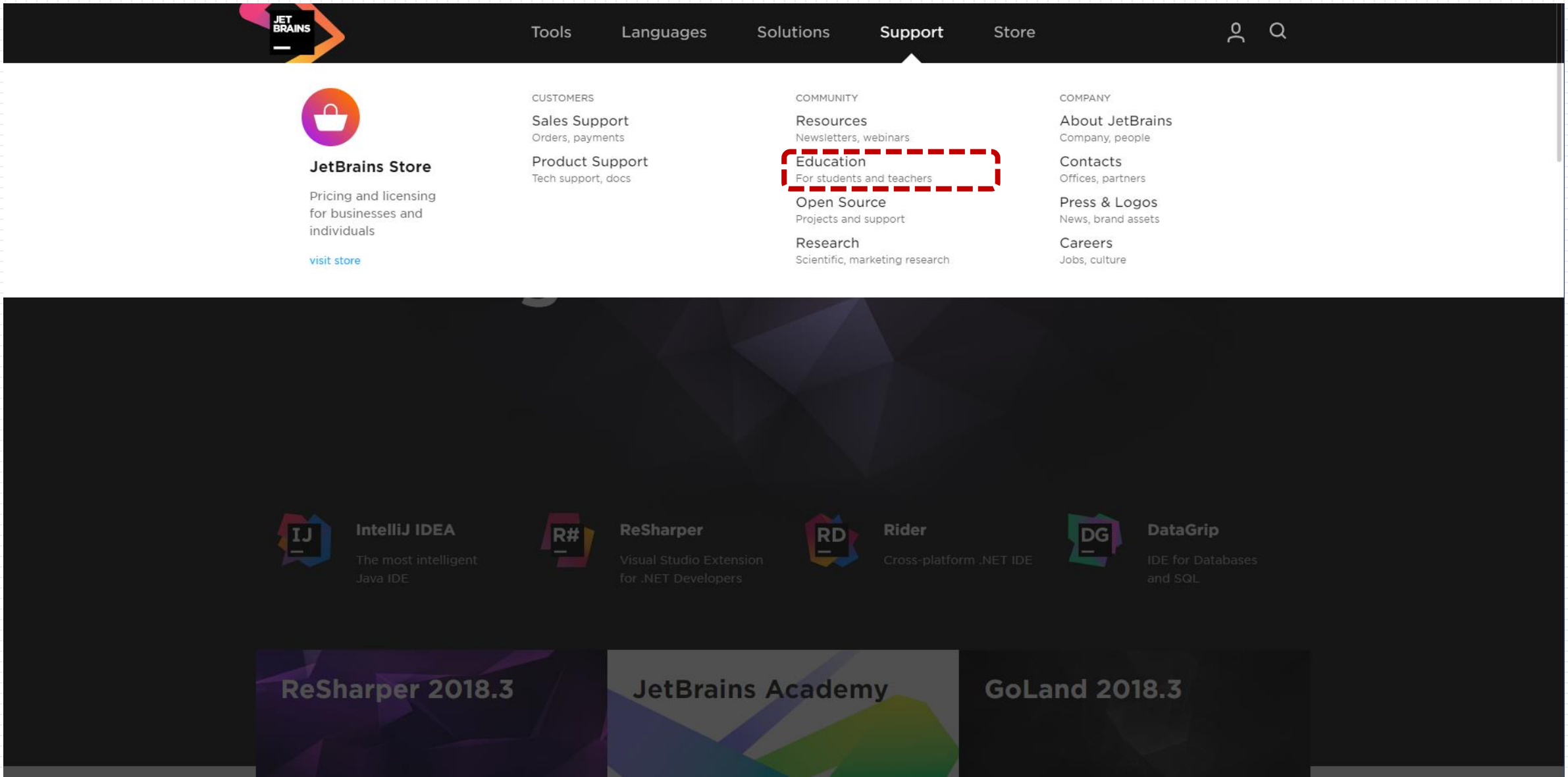
홈페이지 접속

https://www.jetbrains.com/idea/

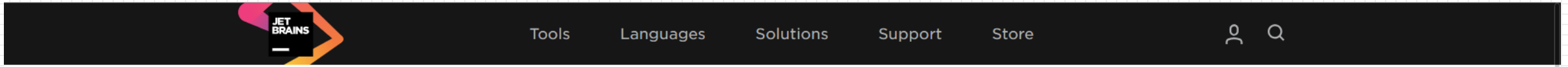
INTELLIJ 설치방법

다운 후, 실행을 하려면 라이선스 필요!

INTELLIJ 학생 라이선스



INTELLIJ 학생 라이선스



Classroom Licenses

Are you a student or an academic staff member?

If you are a student enrolled at an accredited educational institution or a teaching staff member, you can apply individually for all JetBrains products at once!

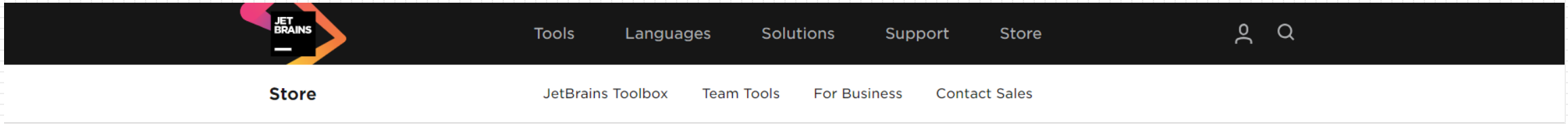
APPLY FOR FREE STUDENT PACK

Classroom Licenses for Accredited Educational Institutions

We provide free licenses to universities, colleges, schools and other non-commercial educational organizations to support classroom instruction in on-site classrooms and computer labs.

- ✓ Licenses are granted to official organization representatives: faculty members & IT support staff.
- ✗ You have to be a staff member to apply for classroom licenses. Applications from students will be rejected. Students should

INTELLIJ 학생 라이선스



Free individual licenses for students and faculty members

English

Students should have access to the highest-quality education available. So if you're learning (or teaching) Java, PHP, Ruby, Python, JavaScript, Kotlin, Objective-C, or .NET technologies, hit the ground running with professional developer tools from JetBrains. The best part is, the educational licenses are completely free!

Who can apply?

Students and faculty staff members from accredited educational institutions: high schools, colleges, universities. Students need to be enrolled in an accredited educational program that takes one or more years of full-time studying to complete.

Not sure if you're eligible? Check out our FAQs.

How do I apply?

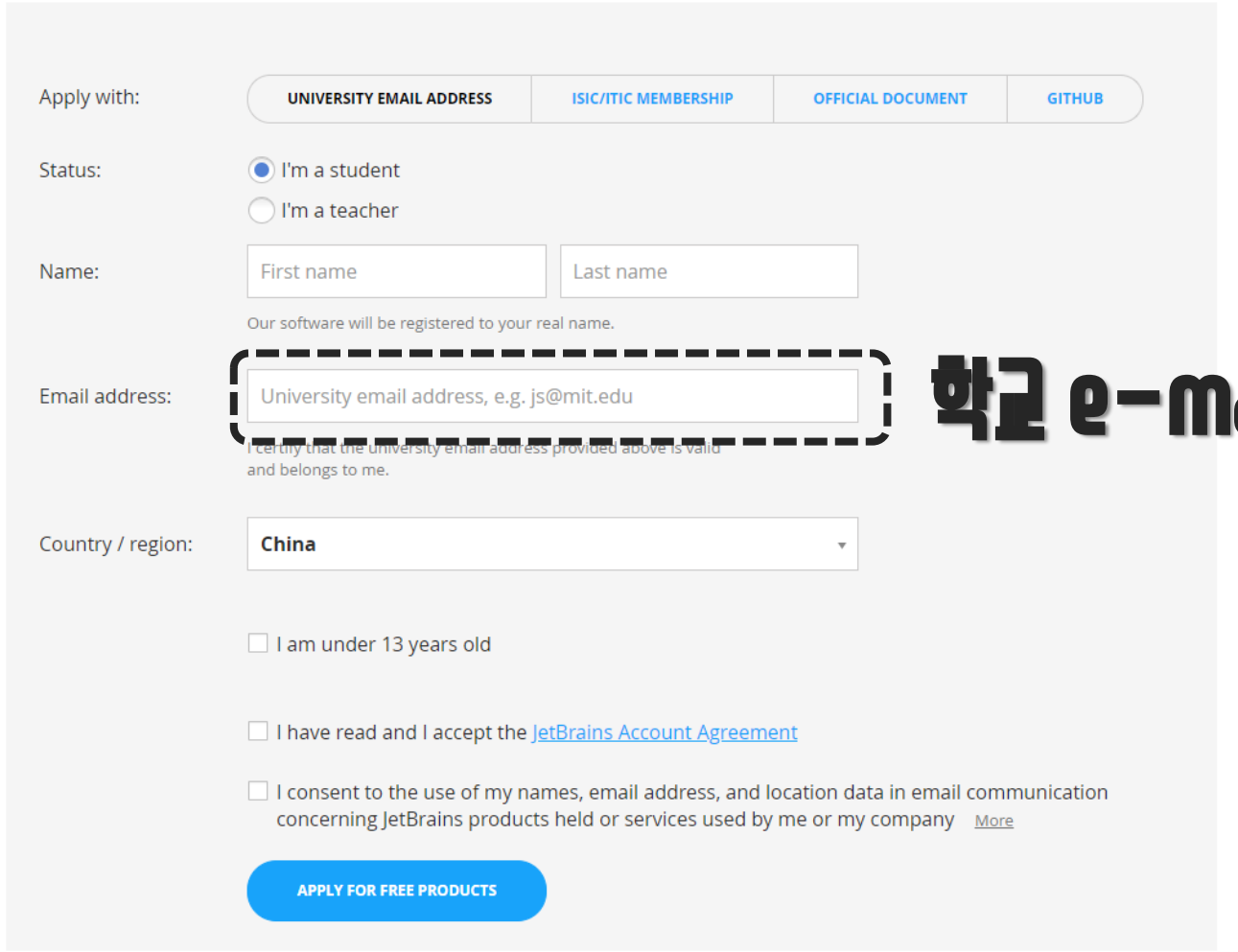
There are a few ways you can apply for a free license. The best way is to use your official institutional email address or ISIC card. In this case, it will only take you a few minutes to get a free educational license pack.

You can also apply with a student/teacher card or any other official document certifying your affiliation with

APPLY NOW

INTELLIJ 학생 라이선스

JetBrains Products for Learning



학교 e-mail을 써야 한다.

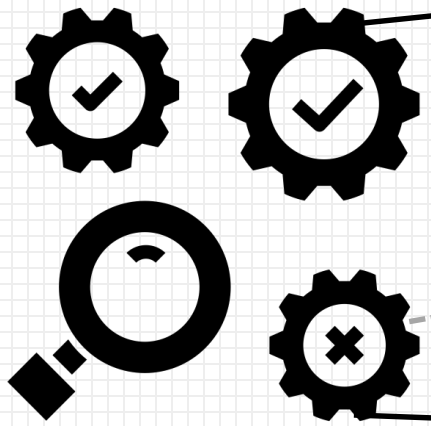
UNIT TEST

- Unit test 설명
- Junit 기능
- 설치
- 사용법

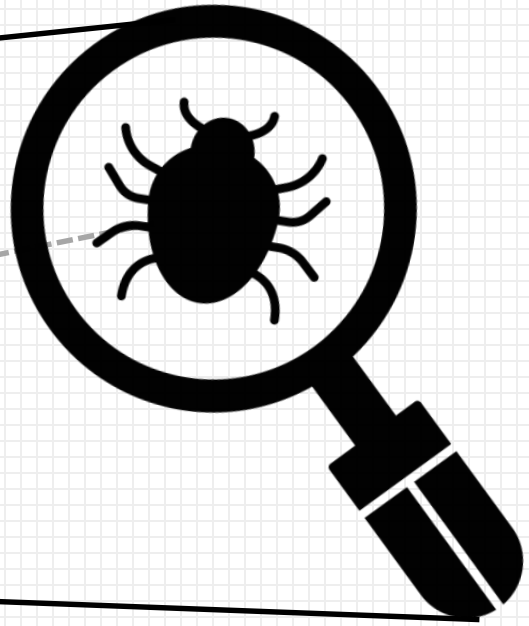
UNIT TEST

UNIT TEST란?

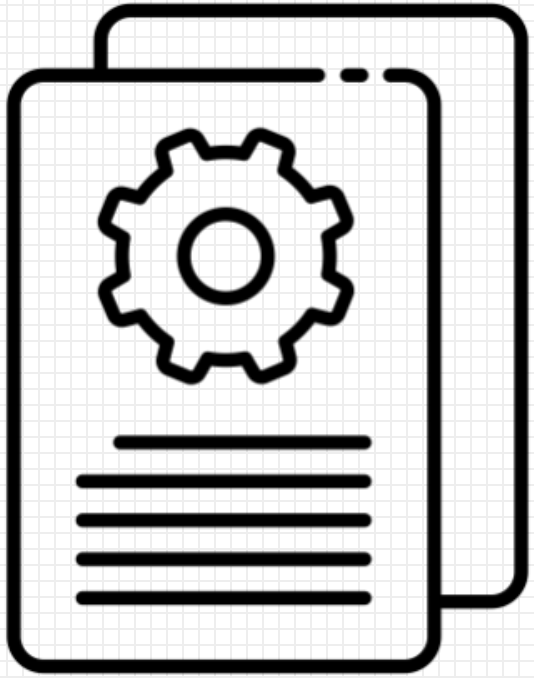
소프트웨어 개발에서 소스 코드로 이루어진 특정 모듈이 요구 사항에 맞게 작동되는지 검증하는 과정
소스 코드로 이루어진 가장 작은 단위의 기능을 테스트한다.



즉, 테스트 케이스는 기능별로 분리 되어야한다.
이상적인 테스트 케이스 작성은 소스 코드의 결합도 낮아지는 효과.

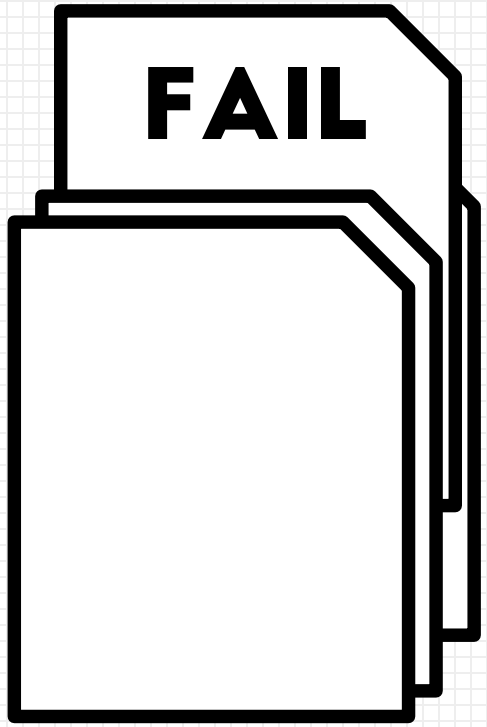


UNIT TEST - PROS



문서화

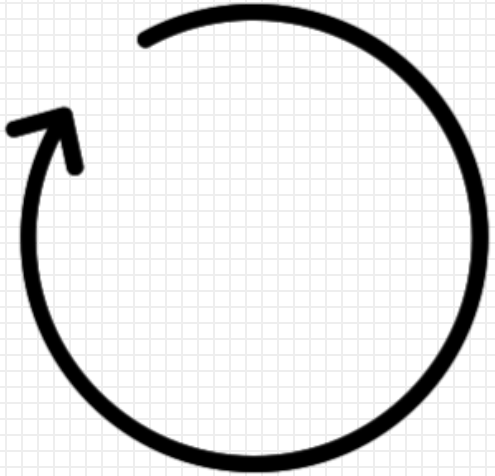
유닛 테스트 코드는 그 자체로 문서처럼 활용할 수 있다.
어떤 모듈을 테스트하는지, 해당 기능이 어떻게 작동 되어야 하는지 명시 되어있다.



문제파악

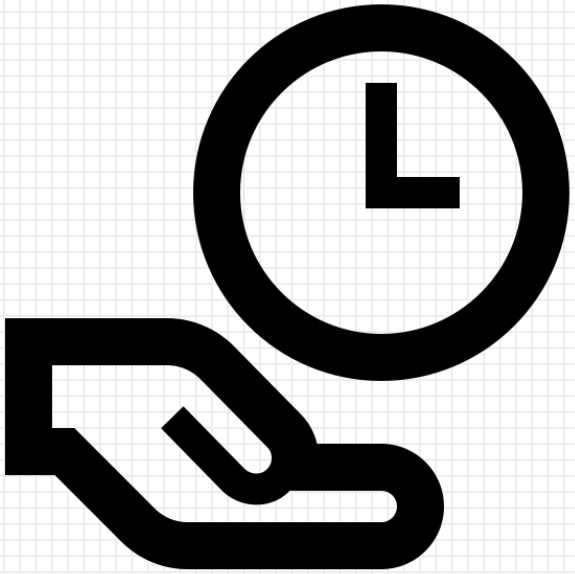
분리되어 있는 테스트 케이스 작성으로, 문제가 발생한 모듈을 쉽게 파악할 수 있다.
수정하고, 다시 테스트하여 빠르게 수정하여 디버깅 시간을 단축한다.

UNIT TEST - PROS



재사용성

유닛 테스트는 테스트 케이스 자체를 잘 구축해놓으면, 재사용에 용이하다.
하나의 테스트 케이스를 다양한 조건에서 수행할 수 있다.



시간 단축

테스트 코드 작성에 시간이 걸리지만,
이후 개발의 많은 시간을 차지하는 디버깅 시간을 최소화할 수 있다.

UNIT TEST - CONS

올바르지 않은 사용

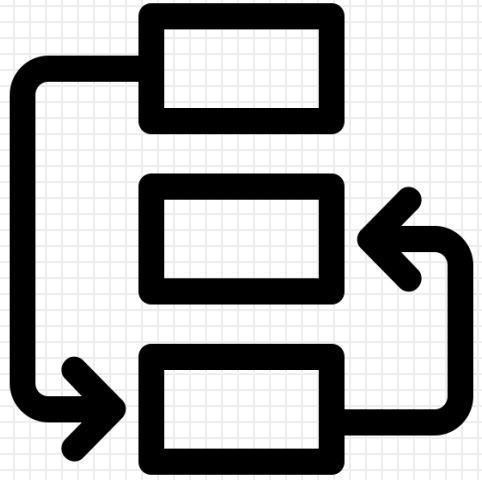
테스트 기능의 단위가 커지면, 오히려 결합도가 올라갈 수 있다.
꼼꼼하게 작성하려다보면(단위의 극소화), 시간이 오래 걸린다.
명확한 테스트 케이스가 없을 경우, 작성하는 시간이 오래 걸린다.

Unit Test

단위 테스트는 단위 테스트일뿐, 그 이후 단계의 버그와 이슈 또한 많다.
결국 모든 시나리오를 실행하기 전까진, 모든 오류와 이슈를 찾을 수 없다.

Black Box Test

Black Box 테스트로 그 안의 알고리즘 효율성이나 내부적인 문제를 해결할 수 없다.



JUNIT5 ?

JUNIT JUPITER

JUNIT VINTAGE

JUNIT PLATFORM

JUNIT 5

Junit5는 Java version 8 이상부터 지원한다.

JUNIT

Junit Test Sample

```java
import static org.junit.jupiter.api.Assertions.assertEquals;

import example.util.Calculator;

import org.junit.jupiter.api.Test;

class MyFirstJUnitJupiterTests {  <-- Test Class

    private final Calculator calculator = new Calculator();

    @Test  <-- Annotation
    void addition() {  <-- Test Method
        assertEquals(2, calculator.add(1, 1));  <-- Assertion
    }

}
```

JUNIT - Class / Method

Test Class 특정 기능을 테스트하기 위한 상위 개념이다.

Test Method Test Class의 특정 기능은 다양한 Test Method를 통해 테스트가 이루어진다.

Lifecycle Method Test Class 안에 있는 Test Method를 수행하는 것에 있어서 항상 수행/테스트해야 하는 것에 대해서 정의하는 것이 Lifecycle Method이다.

JUNIT - Annotations

Annotations

= 주석

어떤 API를 사용하는 것인지 명시

Junit을 실행할 경우, Annotation을 참고해서 테스트를 수행한다.

@Test : 가장 기본적인 테스트 단위(Unit Test)임을 명시

@DisplayName : 테스트 클래스 혹은 메서드의 나타나는 이름을 정의

@BeforeEach / @AfterEach : 각각의 메서드 수행 전,후 실행

@BeforeAll / @AfterAll : 메서드는 소속된 class의 모든 수행 전,후 실행

@Disable : 비활성화

@Tag, @Nested, …

JUNIT – Annotations

2.1. Annotations

JUnit Jupiter supports the following annotations for configuring tests and extending the framework.

Unless otherwise stated, all core annotations are located in the `org.junit.jupiter.api` package in the `junit-jupiter-api` module.

Annotation	Description
`@Test`	Denotes that a method is a test method. Unlike JUnit 4's `@Test` annotation, this annotation does not declare any attributes, since test extensions in JUnit Jupiter operate based on their own dedicated annotations. Such methods are *inherited* unless they are *overridden*.
`@ParameterizedTest`	Denotes that a method is a parameterized test. Such methods are *inherited* unless they are *overridden*.
`@RepeatedTest`	Denotes that a method is a test template for a repeated test. Such methods are *inherited* unless they are *overridden*.
`@TestFactory`	Denotes that a method is a test factory for dynamic tests. Such methods are *inherited* unless they are *overridden*.
`@TestTemplate`	Denotes that a method is a template for test cases designed to be invoked multiple times depending on the number of invocation contexts returned by the registered providers. Such methods are *inherited* unless they are *overridden*.
`@TestMethodOrder`	Used to configure the test method execution order for the annotated test class; similar to JUnit 4's `@FixMethodOrder`. Such annotations are *inherited*.
`@TestInstance`	Used to configure the test instance lifecycle for the annotated test class. Such annotations are *inherited*.
`@DisplayName`	Declares a custom display name for the test class or test method. Such annotations are not *inherited*.

Annotation	Description
`@DisplayNameGeneration`	Declares a custom display name generator for the test class. Such annotations are *inherited*.
`@BeforeEach`	Denotes that the annotated method should be executed *before* **each** `@Test`, `@RepeatedTest`, `@ParameterizedTest`, or `@TestFactory` method in the current class; analogous to JUnit 4's `@Before`. Such methods are *inherited* unless they are *overridden*.
`@AfterEach`	Denotes that the annotated method should be executed *after* **each** `@Test`, `@RepeatedTest`, `@ParameterizedTest`, or `@TestFactory` method in the current class; analogous to JUnit 4's `@After`. Such methods are *inherited* unless they are *overridden*.
`@BeforeAll`	Denotes that the annotated method should be executed *before* **all** `@Test`, `@RepeatedTest`, `@ParameterizedTest`, and `@TestFactory` methods in the current class; analogous to JUnit 4's `@BeforeClass`. Such methods are *inherited* (unless they are *hidden* or *overridden*) and must be `static` (unless the "per-class" test instance lifecycle is used).
`@AfterAll`	Denotes that the annotated method should be executed *after* **all** `@Test`, `@RepeatedTest`, `@ParameterizedTest`, and `@TestFactory` methods in the current class; analogous to JUnit 4's `@AfterClass`. Such methods are *inherited* (unless they are *hidden* or *overridden*) and must be `static` (unless the "per-class" test instance lifecycle is used).
`@Nested`	Denotes that the annotated class is a non-static nested test class. `@BeforeAll` and `@AfterAll` methods cannot be used directly in a `@Nested` test class unless the "per-class" test instance lifecycle is used. Such annotations are not *inherited*.
`@Tag`	Used to declare tags for filtering tests, either at the class or method level; analogous to test groups in TestNG or Categories in JUnit 4. Such annotations are *inherited* at the class level but not at the method level.
`@Disabled`	Used to disable a test class or test method; analogous to JUnit 4's `@Ignore`. Such annotations are not *inherited*.
`@ExtendWith`	Used to register extensions declaratively. Such annotations are *inherited*.
`@RegisterExtension`	Used to register extensions programmatically via fields. Such fields are *inherited* unless they are *shadowed*.
`@TempDir`	Used to supply a temporary directory via field injection or parameter injection in a lifecycle method or test method; located in the `org.junit.jupiter.api.io` package.

JUNIT - Assertions

Assertions

= (사실임을) 주장
테스트 케이스에 대하여 입력 A값 혹은 기능이 실행되었을 때에,
원하는 출력이 나오는지 확인하기 위한 함수

함수	기능	함수	기능
assertEquals(A,B)	A == B 이면 Pass	assertNotEquals(A, B)	A != B 이면 Pass
assertNotNull(A)	A != N 이면 Pass	fail(A)	A 메시지와 함께 테스트 Fail
assertThrows(A, B)	B를 수행했을때 A 예외 발생할 경우 Pass	assertTimeout(A, B)	B를 수행하는 것에 있어서 A가 넘지 않으면 Pass

JUNIT - How to use

Current Versions

JUnit Jupiter	v5.4.1
JUnit Vintage	v5.4.1
JUnit Platform	v1.4.1

sonatype | The Central Repository Quick Stats Who is Sonatype? GitHub

g:org.junit.jupiter

Group ID	Artifact ID	Latest Version		Updated	Download
org.junit.jupiter	junit-jupiter-params	5.4.1	(28)	18-Mar-2019	
org.junit.jupiter	junit-jupiter-migrationsupport	5.4.1	(27)	18-Mar-2019	
org.junit.jupiter	junit-jupiter-engine	5.4.1	(31)	18-Mar-2019	
org.junit.jupiter	junit-jupiter-api	5.4.1	(31)	18-Mar-2019	
org.junit.jupiter		5.4.1	(5)	18-Mar-2019	
org.junit.jupiter	junit-jupiter-migration-support	5.0.0-M4	(2)	02-Apr-2017	

Search for all artifacts with this Artifact ID

sonatype | The Central Repository Quick Stats Who is Sonatype? GitHub

g:org.junit.platform

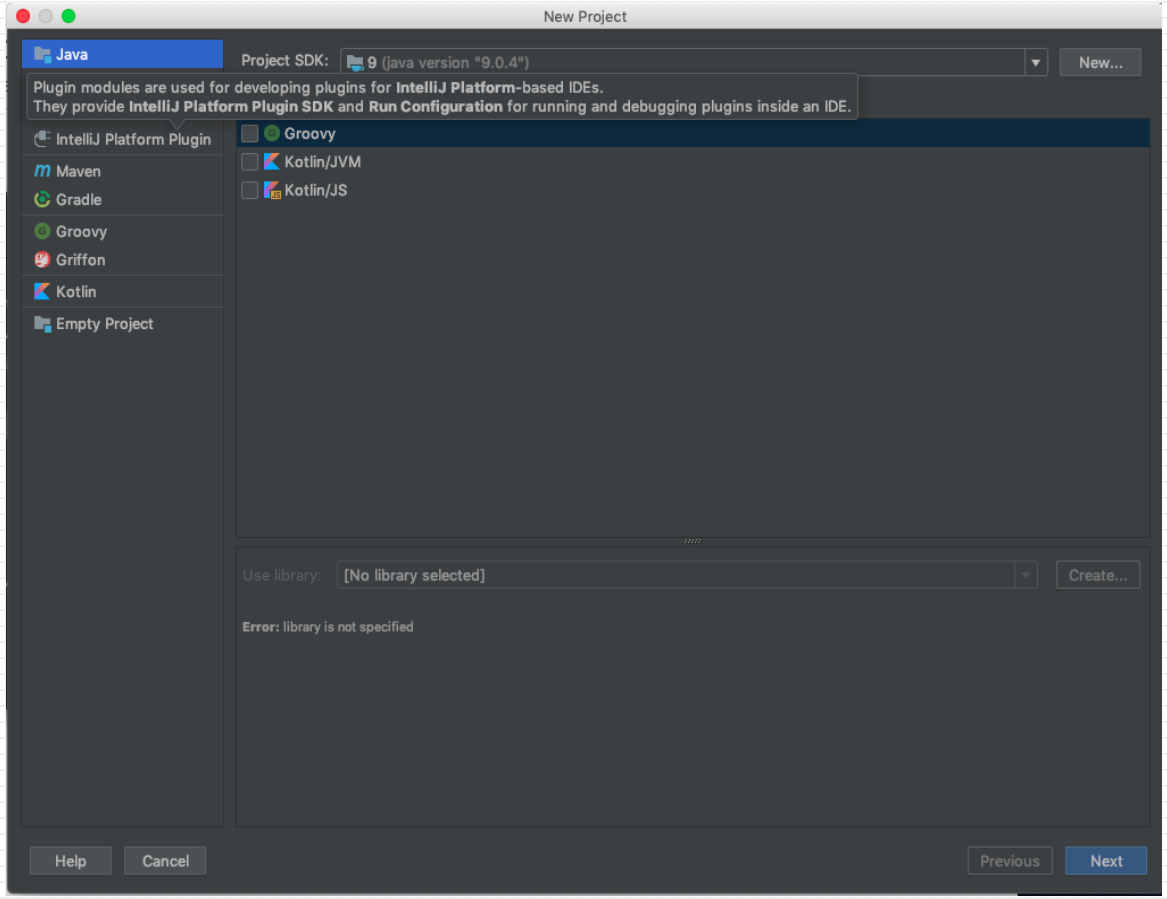
Group ID	Artifact ID	Latest Version		Updated	Download
org.junit.platform	junit-platform-testkit	1.4.1	(5)	18-Mar-2019	
org.junit.platform	junit-platform-suite-api	1.4.1	(28)	18-Mar-2019	
org.junit.platform	junit-platform-runner	1.4.1	(31)	18-Mar-2019	
org.junit.platform	junit-platform-reporting	1.4.1	(5)	18-Mar-2019	
org.junit.platform	junit-platform-launcher	1.4.1	(31)	18-Mar-2019	
org.junit.platform	junit-platform-engine	1.4.1	(31)	18-Mar-2019	
org.junit.platform	junit-platform-console-standalone	1.4.1	(28)	18-Mar-2019	
org.junit.platform	junit-platform-console	1.4.1	(31)	18-Mar-2019	
org.junit.platform	junit-platform-commons	1.4.1	(31)	18-Mar-2019	
org.junit.platform		1.3.2	(26)	26-Nov-2018	
org.junit.platform	junit-platform-gradle-plugin	1.2.0	(21)	30-Apr-2018	

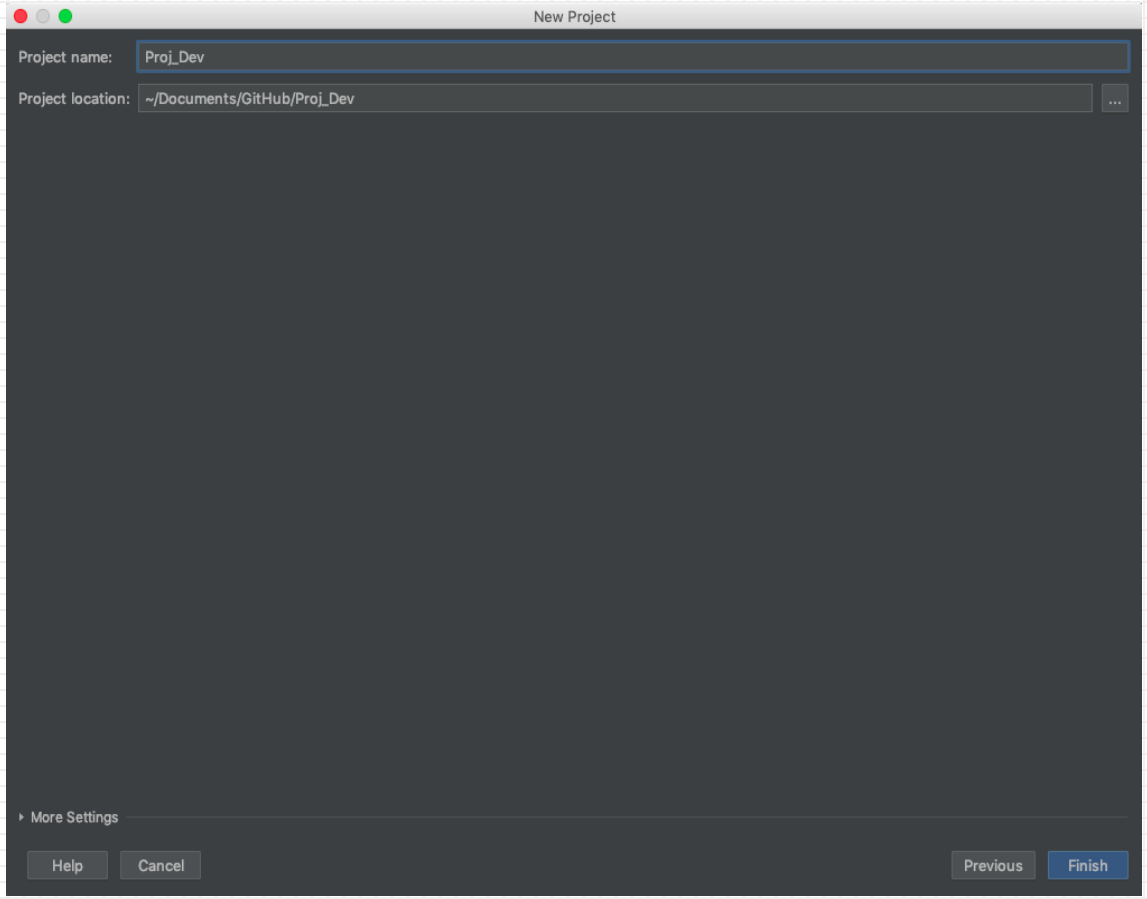
Search for all artifacts with this Artifact ID

Items per page: 20 1 - 11 of 11

각 파일을 jar 파일로 설치

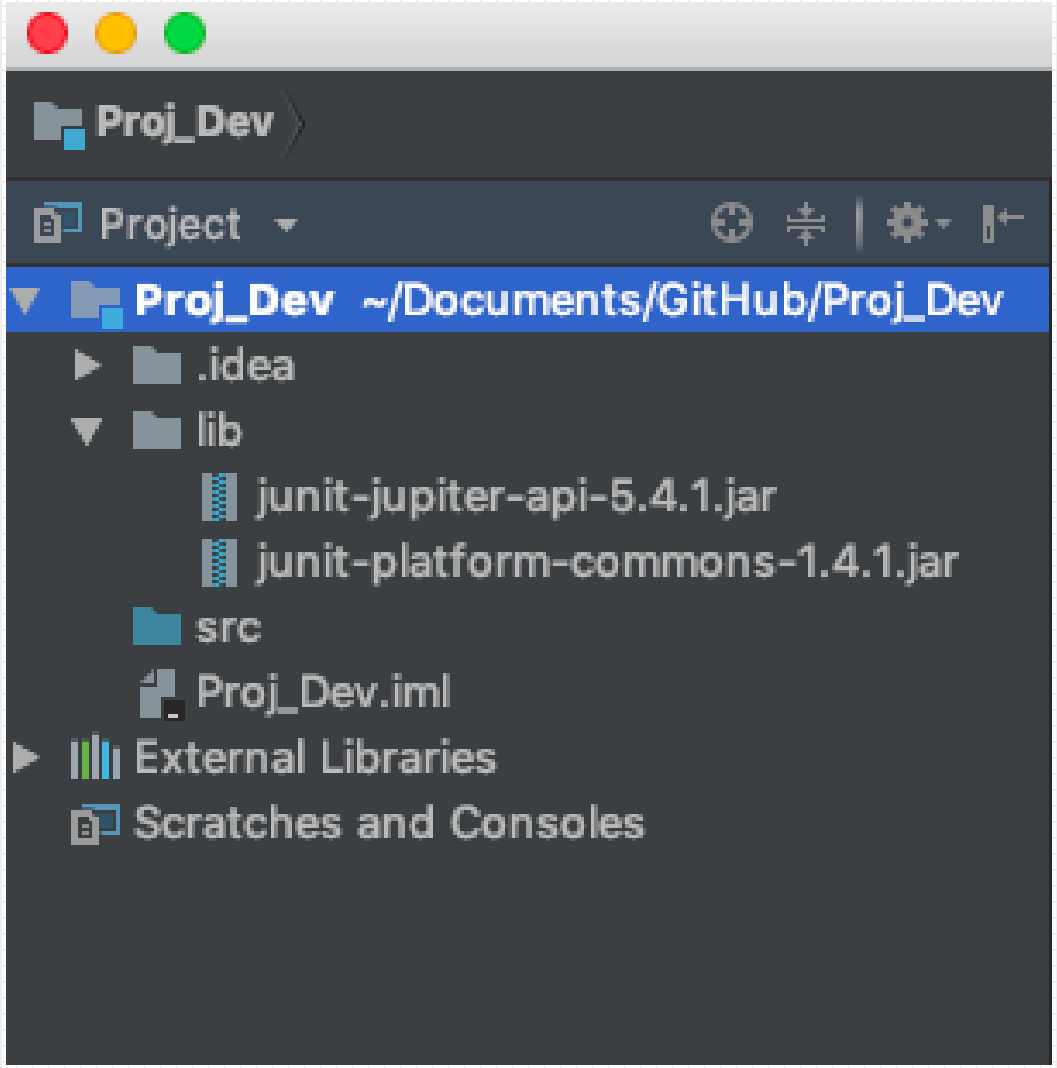
JUNIT - How to use





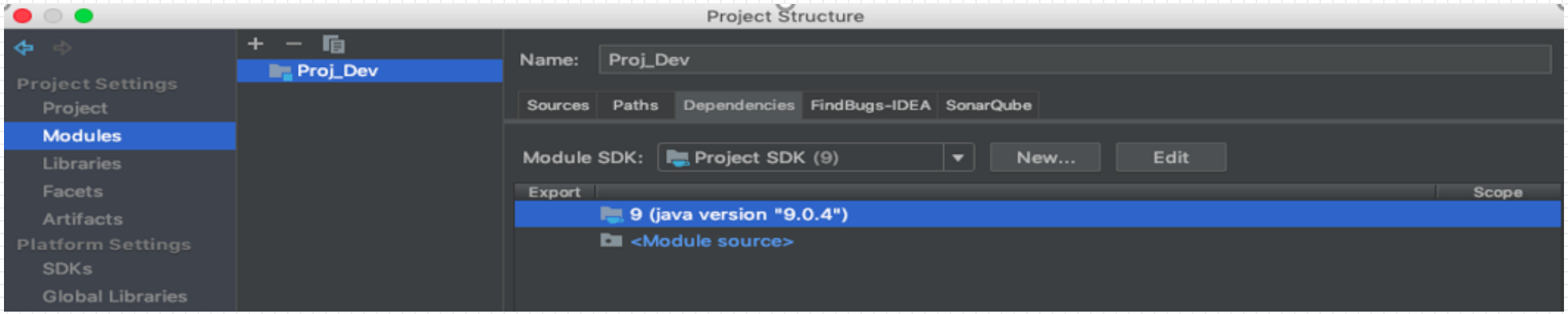
IntelliJ IDE에서 새로운 Java Project 추가

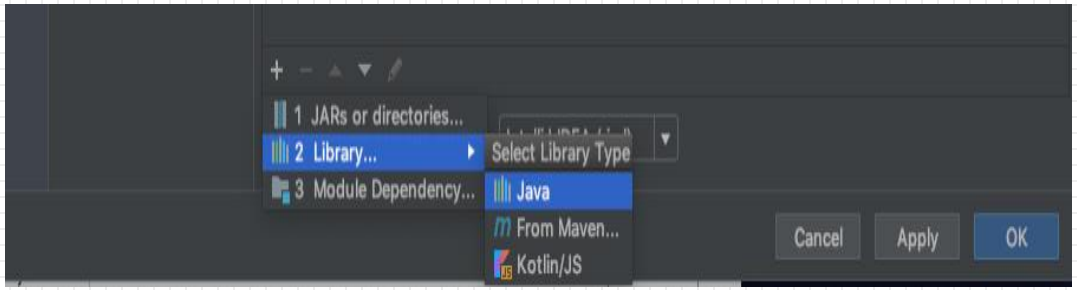
JUNIT - How to use

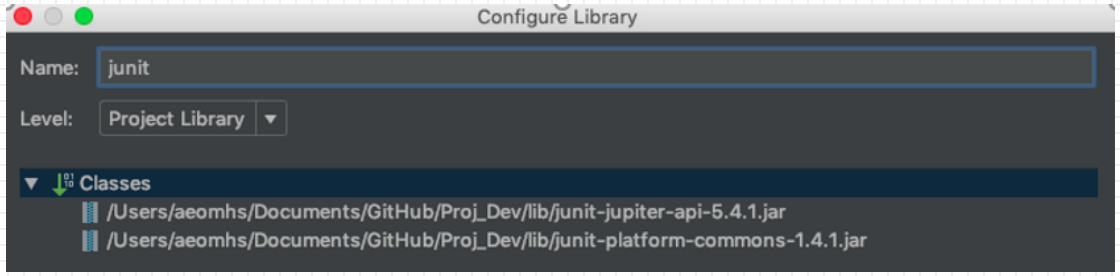


Project에 Jnit library를
보관할 폴더 생성 후
Jar파일 추가

JUNIT - How to use

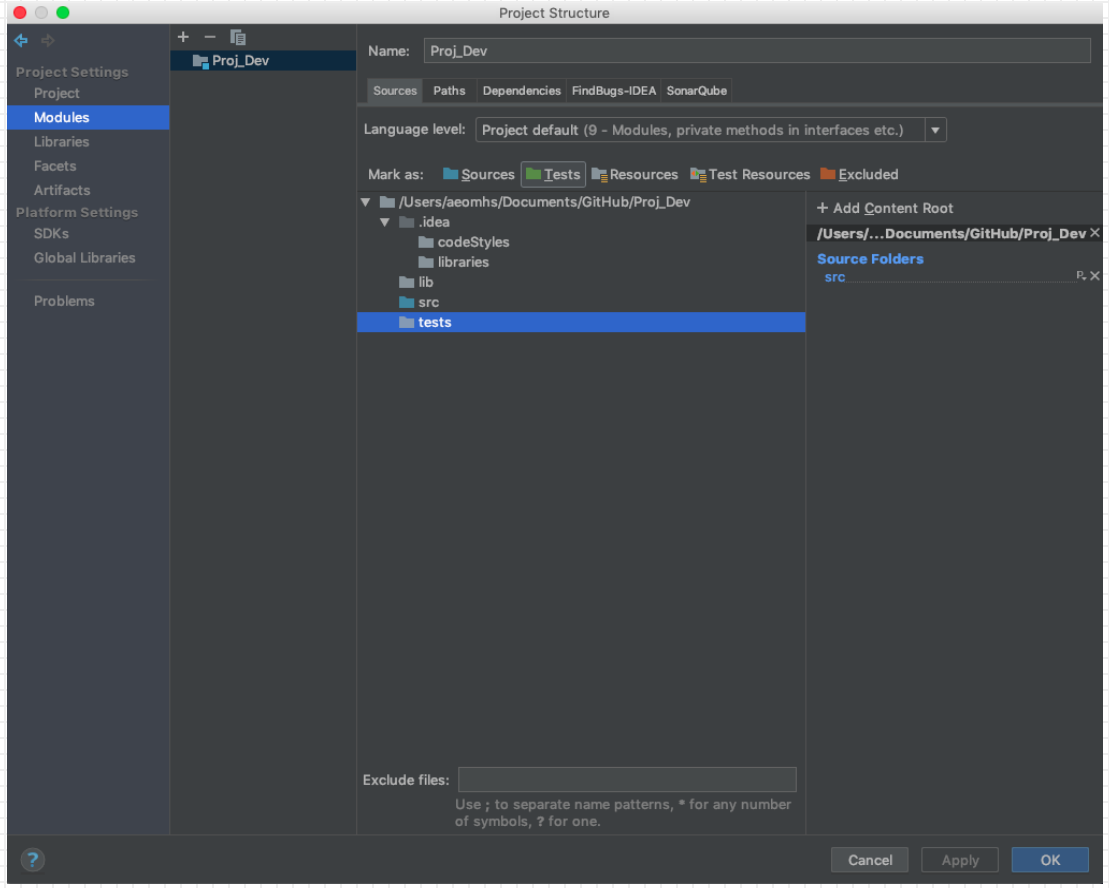


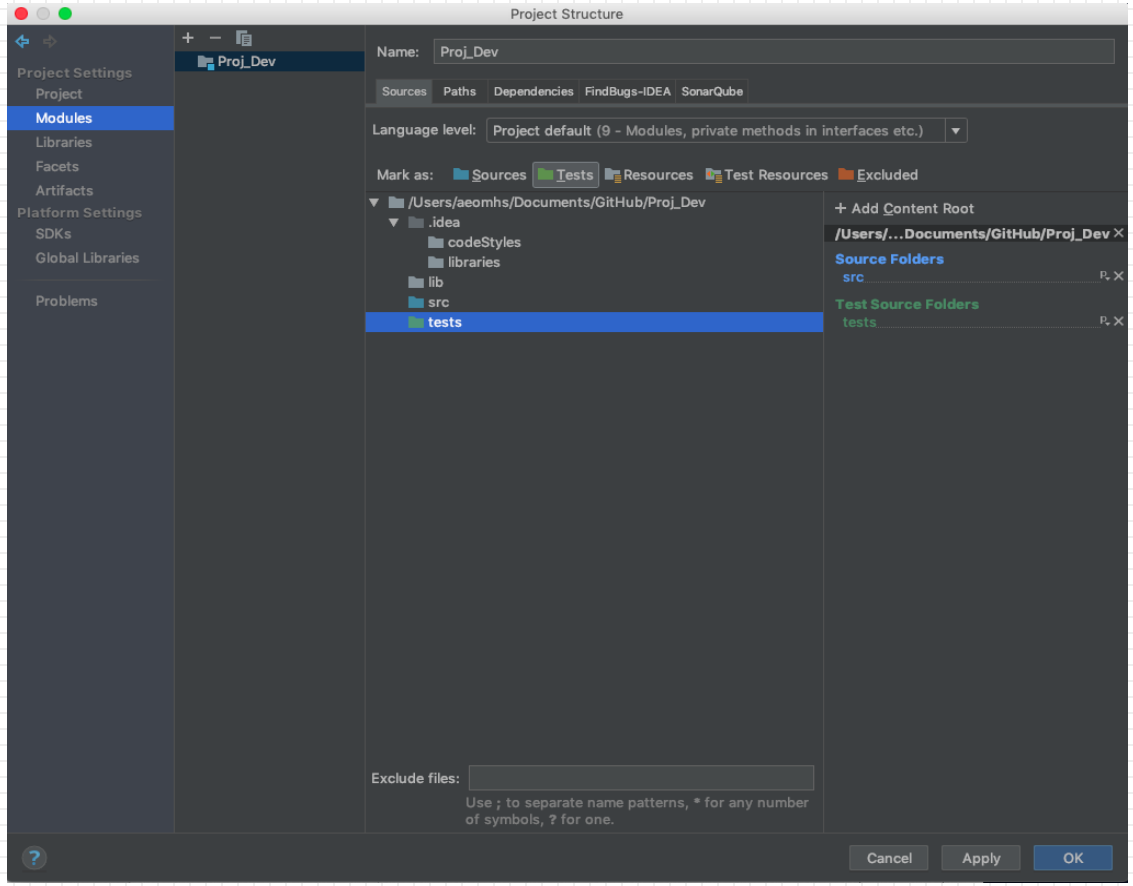




Project에 Jnit library를 보관할 폴더 생성 후
Jar파일 추가

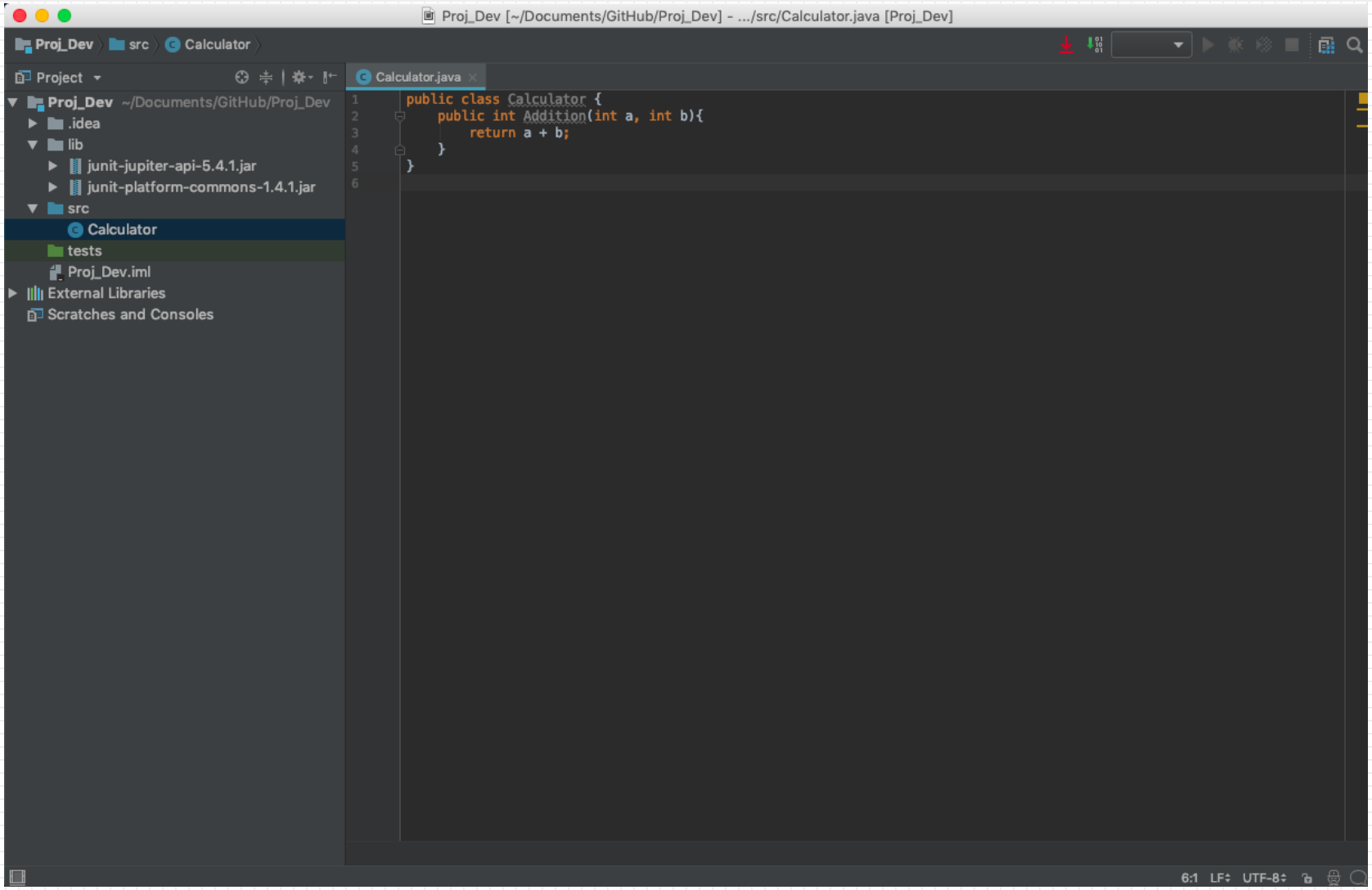
JUNIT - How to use





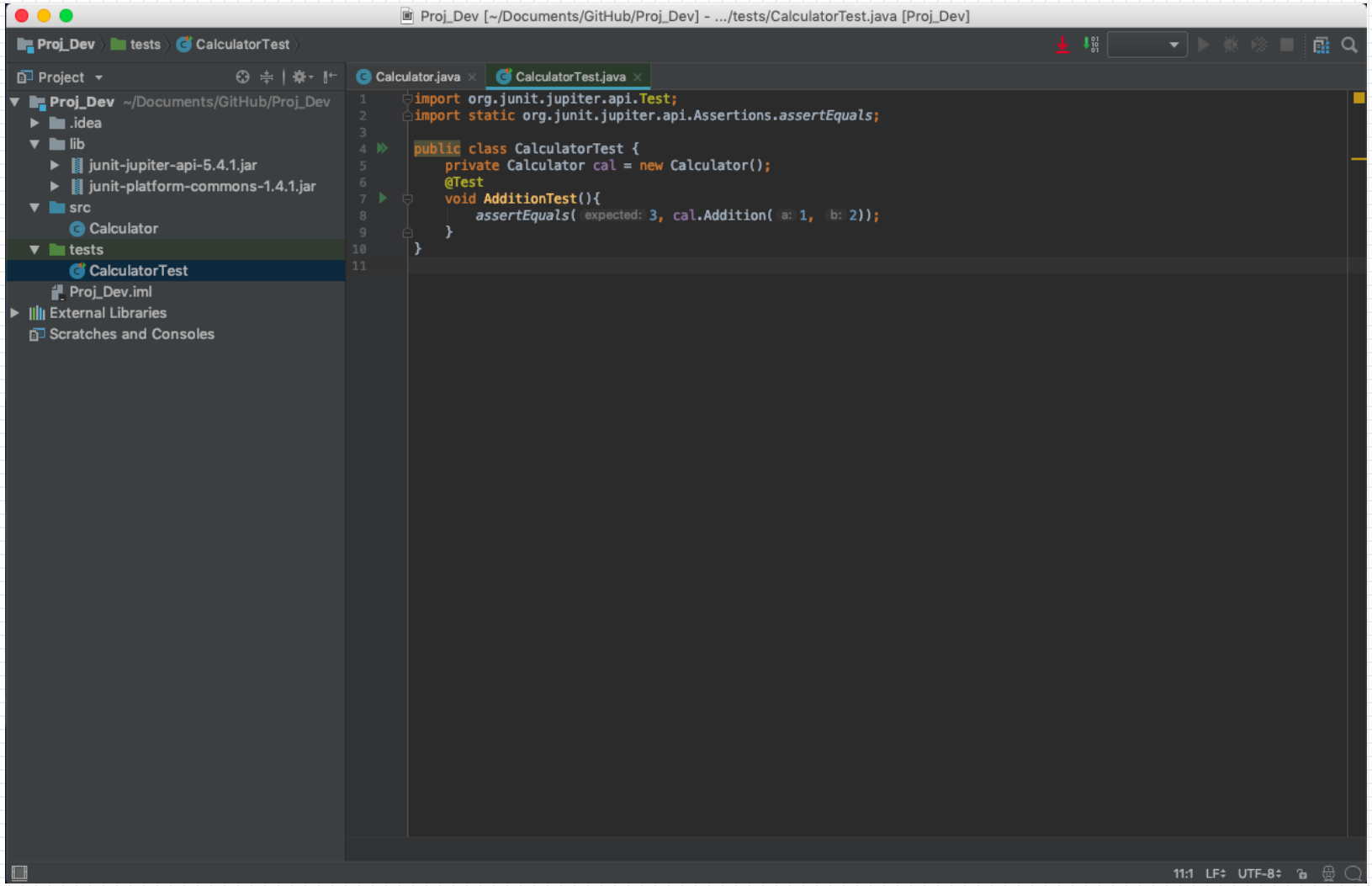
Module Settings -> Project Settings/Modules/..Project../Sources에
"tests"라는 이름의 Test Source Folder 추가

JUNIT - How to use



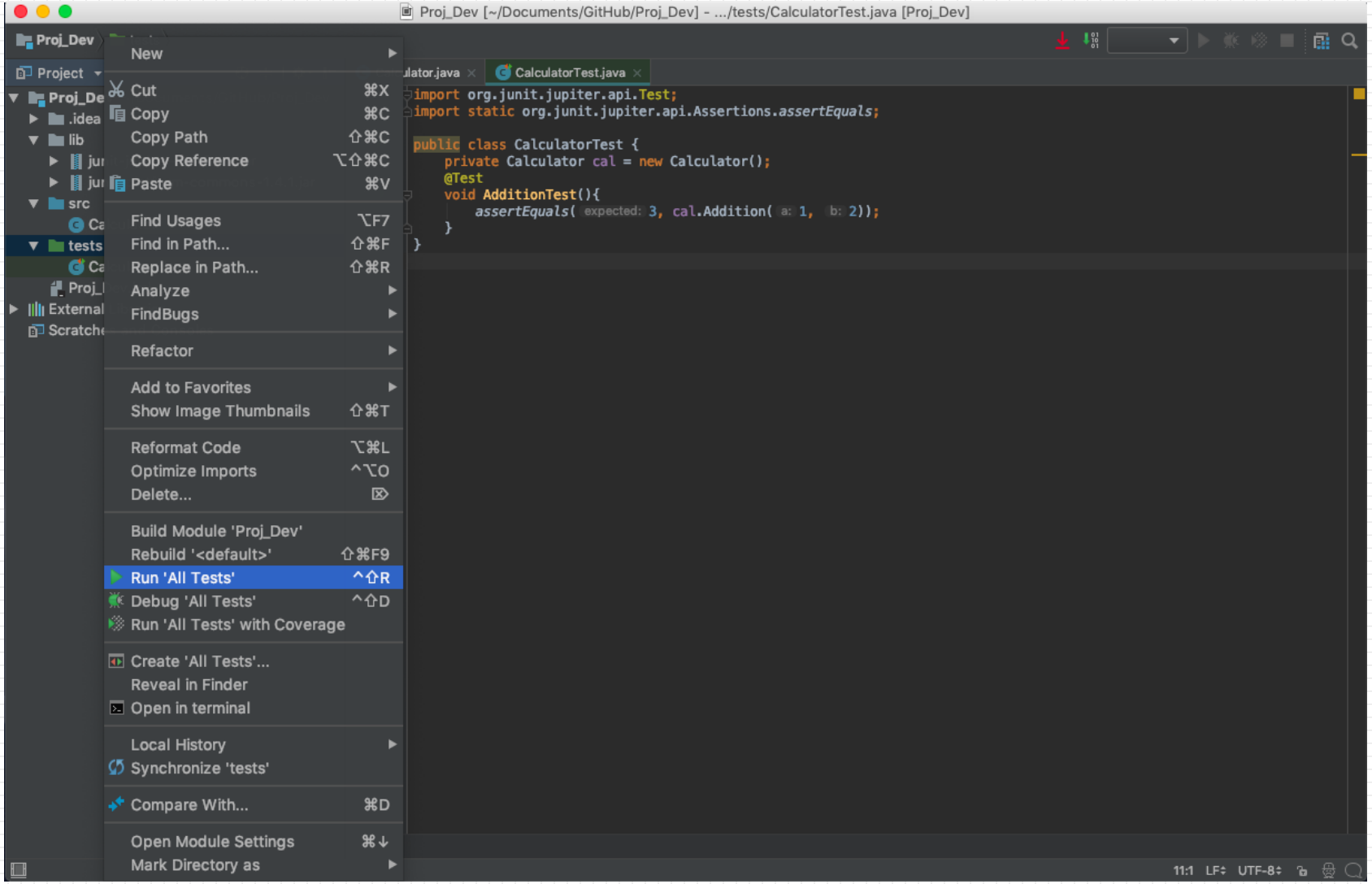
Project/src/ 경로에
코드 구현

JUNIT – How to use



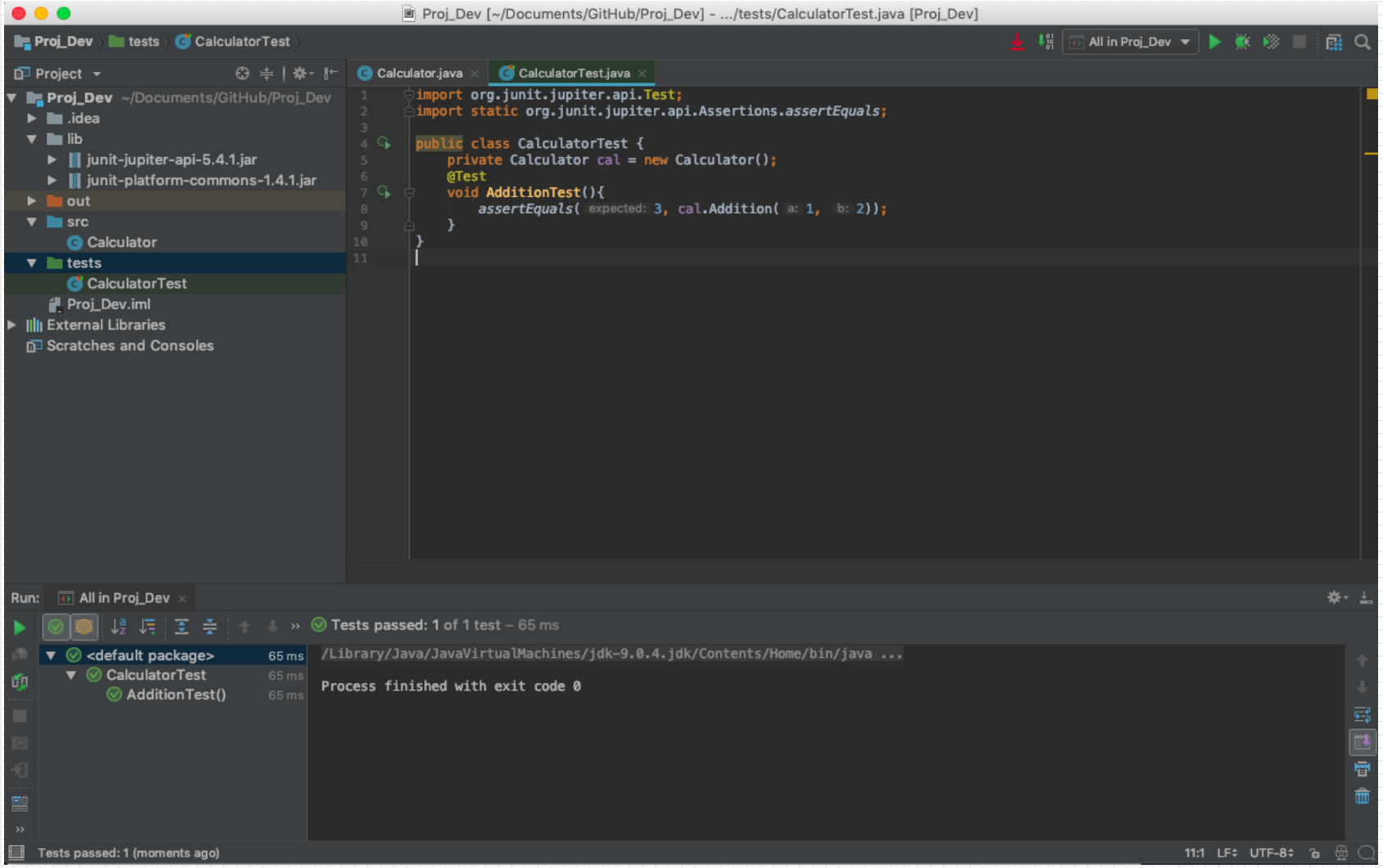
Project/tests/ 경로에
Test Code 구현

JUNIT - How to use



Test 진행

JUNIT - How to use



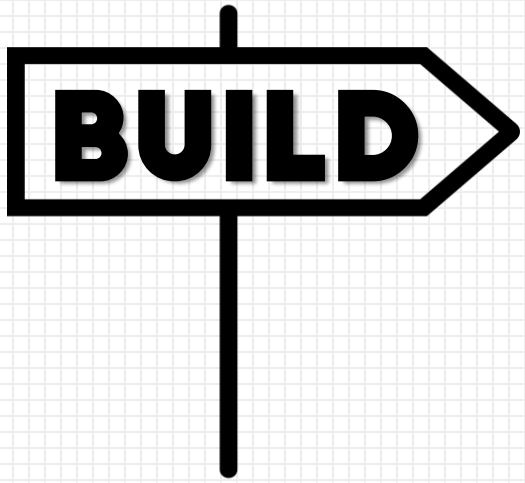
Test 진행

BUILD - Build tools

BUILD란?

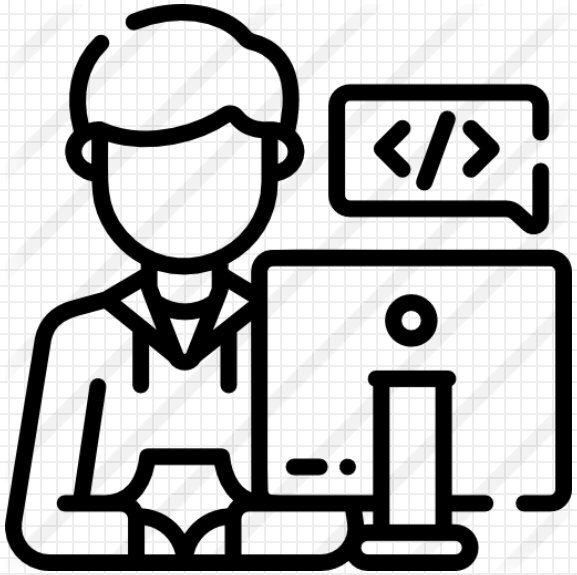
소스코드 파일을 컴퓨터에서 실행 할 수 있는 독립 소프트웨어 가공물로 변환하는 과정을 말하거나 그에 대한 결과물을 일컫는 것.

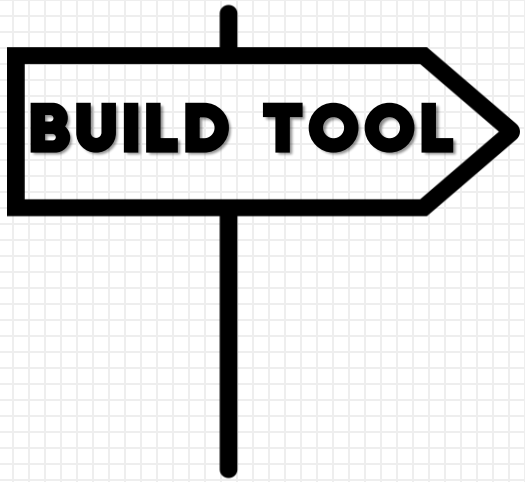
```java
import java.util.*;

class hackermeter {
  public static void run(Scanner scanner) {
    // Code here!
  }

  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    int cases =
Integer.parseInt(scanner.nextLine());
    for(int i = 0; i < cases; i++) {
      run(scanner);
    }
  }
}
```

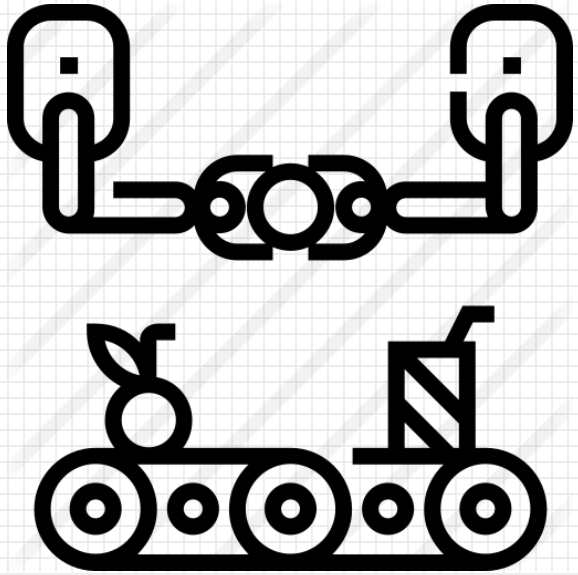


BUILD - Build tools

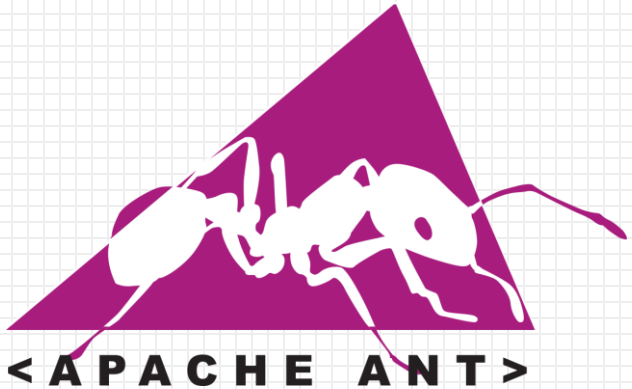
Build Automation(빌드 자동화)란?







BUILD – Build tools



2000년 출시

출시 이유: 유닉스의 make 명령어의 대안으로 개발

유연함

2004년 출시

출시 이유 : Ant를 개발할 때
직면한 일부 문제를 개선

편리한 의존성 관리

2012년 출시

출시 이유 : Ant와 Maven의 문제점을 개선, 장점만을 취함.

유연함 + 의존성

비교

병렬 빌드 및 병렬 의존성 해결

XML 사용 -> 가독성 떨어짐

의존관계가 복잡한 프로젝트를 설정하기에 부적절

속도

유연성

의존성 관리

MAVEN의 약 10~100배

Groovy을 기반으로 유연하고 확장성이 뛰어남

Apache Ivy에 기반한 강력한 의존성 관리

상속 구조

-> 특정 설정을 소수 모듈에게 공유하기 위해 부모 프로젝트 생성 해야함

멀티프로젝트 구조

설정 주입 방식

-> 프로젝트의 조건을 체크하여 프로젝트별 설정을 다르게 할 수 있음

그 외에도!!

<사용자 지원>

빠른 IDE 추가 지원, 최신 CLI 제공

<문서화>

공식 홈페이지에서 방대하고
평이한 자료 및 문서 제공

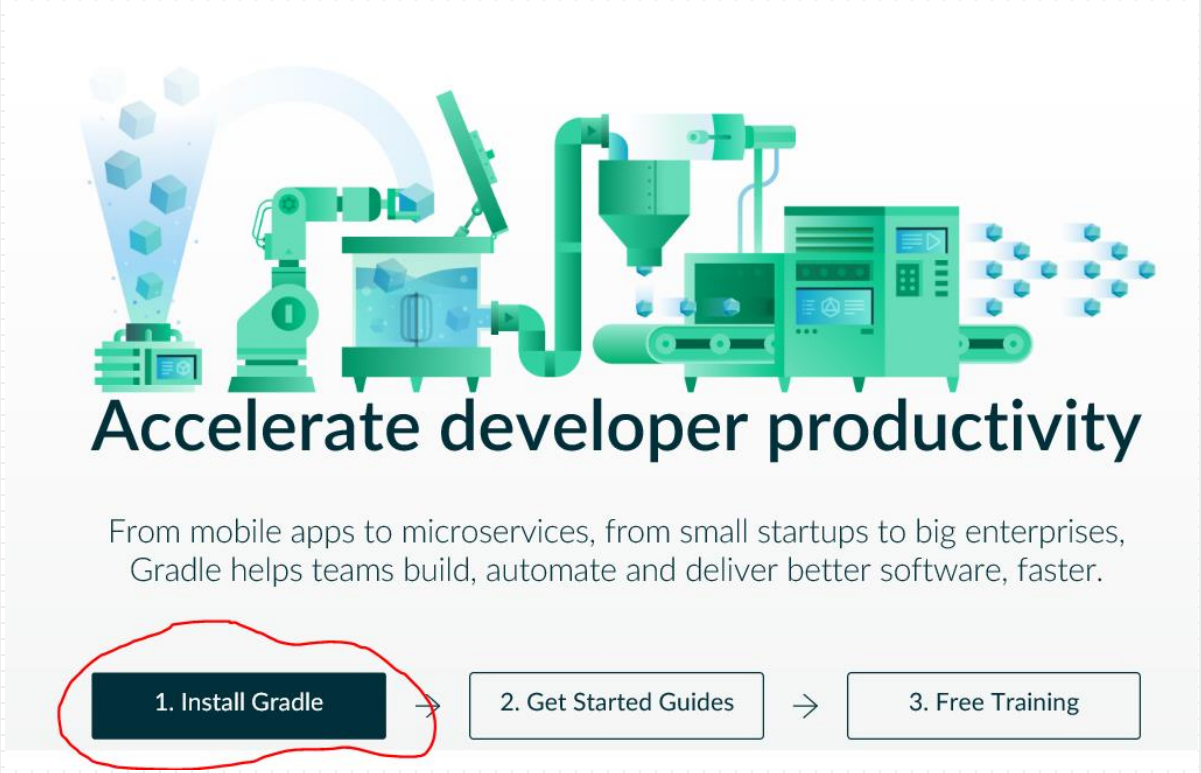
<플러그인>

다양한 종류의 플러그인 사용 가능

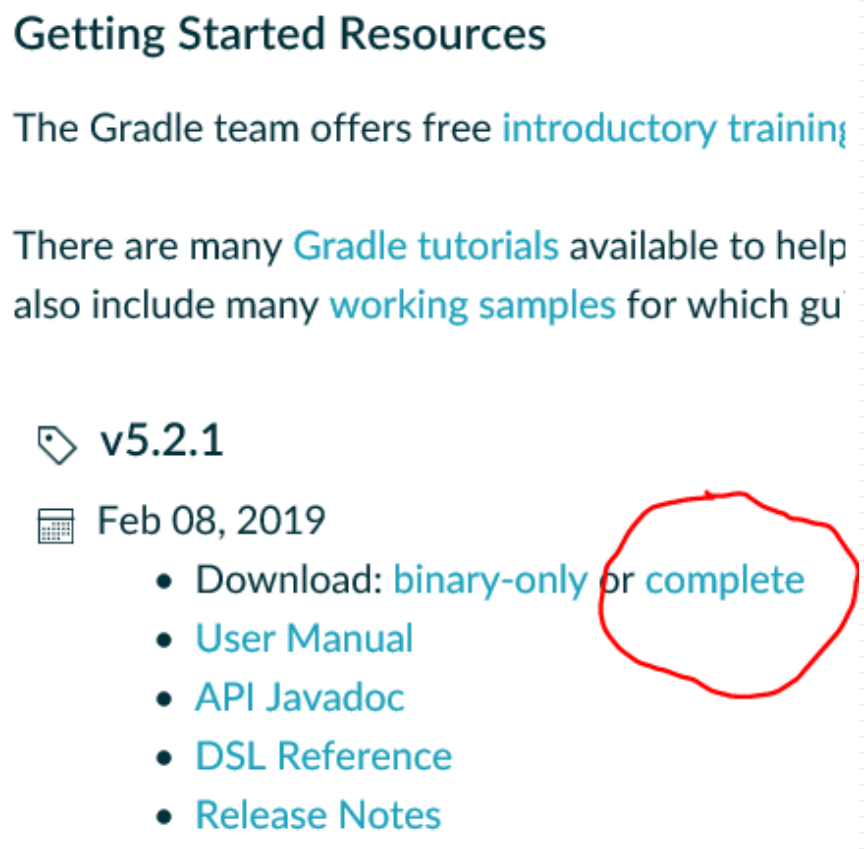
<유연성>

여러가지 언어들에 대한
빌드 환경을 제공

GRADLE 설치방법

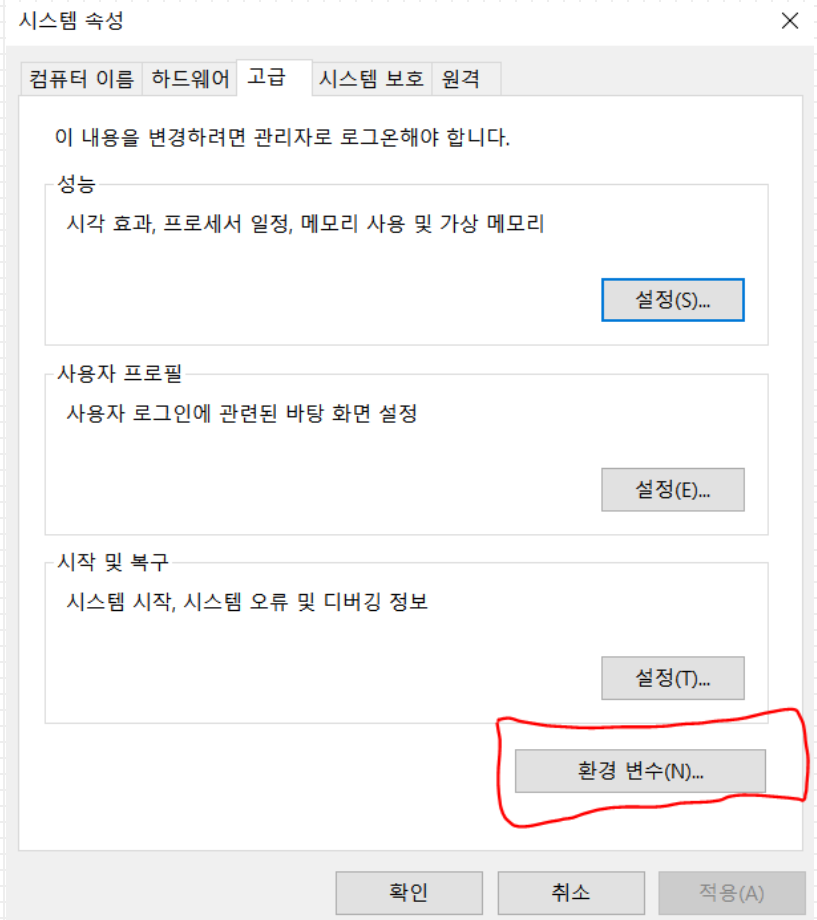


다운로드 - https://gradle.org/install/



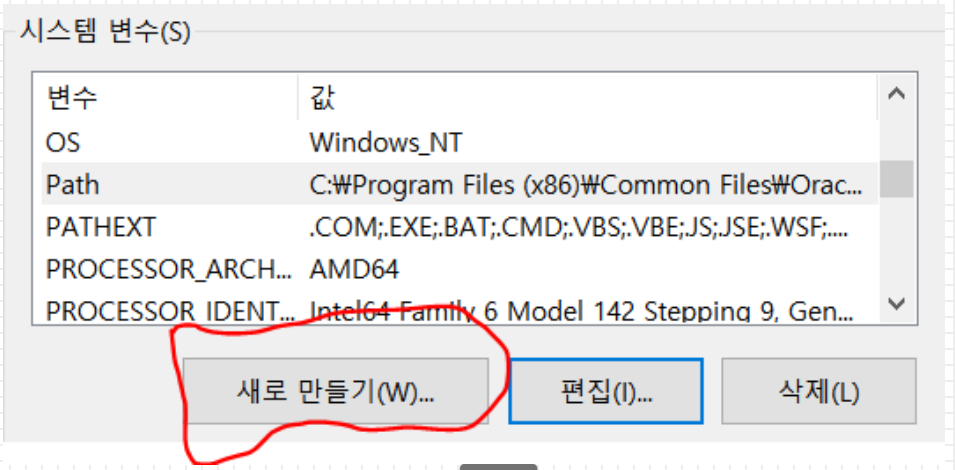
환경에 맞는 버전 설치

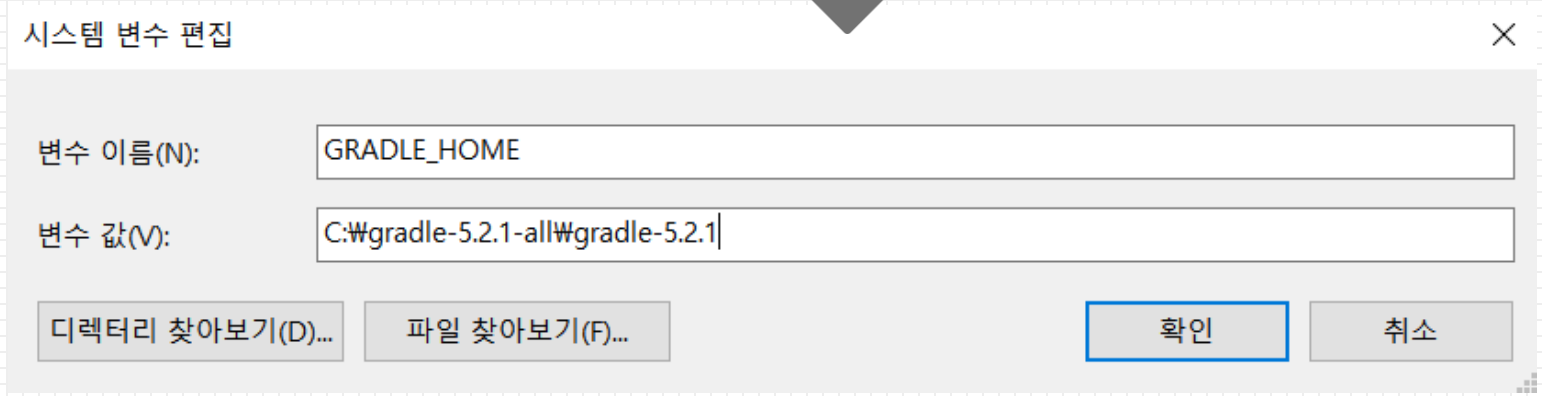
GRADLE 설치방법



시스템 속성 - 환경 변수 클릭

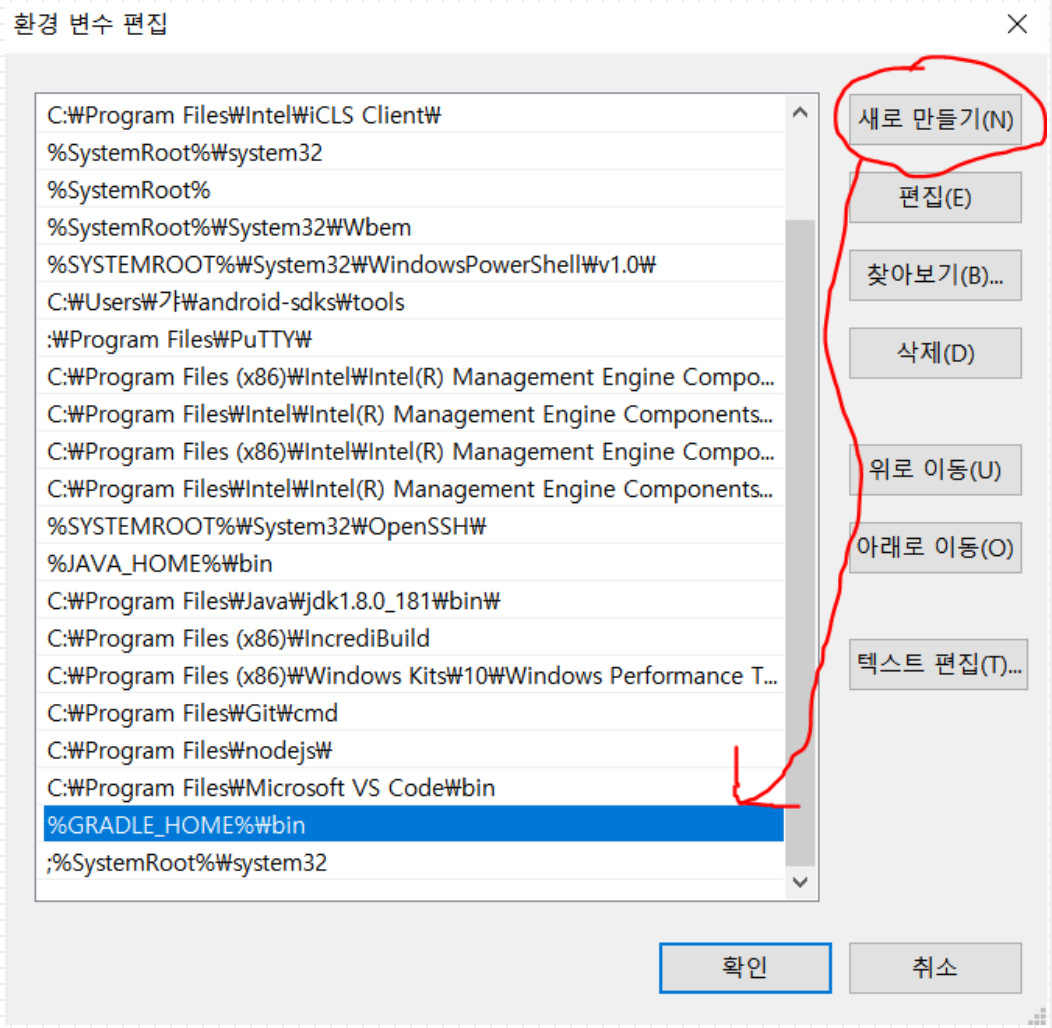
시스템 변수 - 새로 만들기





Gradle 압축을 푼 경로로 지정

GRADLE 설치방법



시스템 변수 - 편집 - 새로 만들기 - 파랑색 추가

```
C:₩Users₩가>gradle -v

Welcome to Gradle 5.2.1!

Here are the highlights of this release:
 - Define sets of dependencies that work together with Java Platform plugin
 - New C++ plugins with dependency management built-in
 - New C++ project types for gradle init
 - Service injection into plugins and project extensions

For more details see https://docs.gradle.org/5.2.1/release-notes.html


------------------------------------------------------------
Gradle 5.2.1
------------------------------------------------------------

Build time:   2019-02-08 19:00:10 UTC
Revision:     f02764e074c32ee8851a4e1877dd1fea8ffb7183

Kotlin DSL:   1.1.3
Kotlin:       1.3.20
Groovy:       2.5.4
Ant:          Apache Ant(TM) version 1.9.13 compiled on July 10 2018
JVM:          1.8.0_181 (Oracle Corporation 25.181-b13)
OS:           Windows 10 10.0 amd64
```

Cmd창에서 gradle -v로 최종 확인

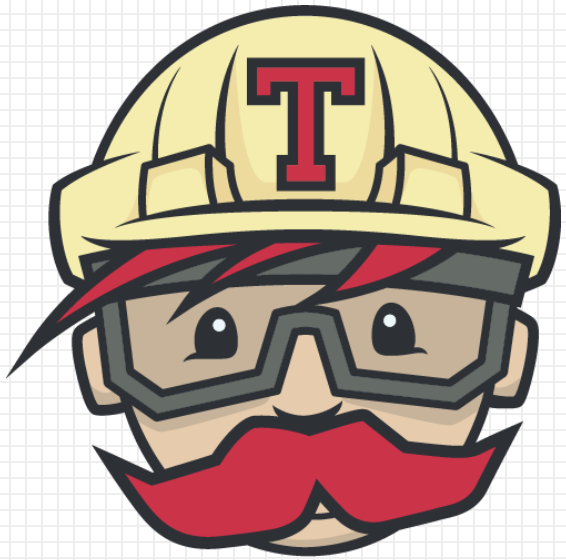
CI/CD INDEX

- 지속적인 자동화 빌드
- 지속적인 자동화 테스트
- 지속적인 자동화 배포 관리
- 강력한 Reporting 기능

CI/CD VISION

“ 지속성 & 자동화 ”

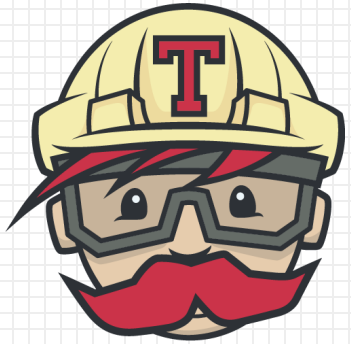
CI/CD TOOLS



VS

stackoverflow

JENKINS VS TRAVIS

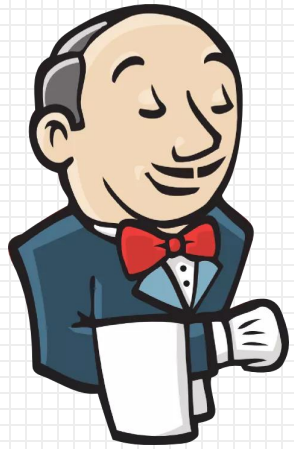


작업준비 알아서 해준다

호스팅 환경 제공

규모별 비용 부과, 소규모 무료

Easy to Start



직접 환경을 구성해야 한다

호스팅 환경 제공 안됨

오픈소스, 다양한 플러그인을 지원한다

Easy to Customize

JENKINS

JENKINS - INSTALL

BREW : Package Manager

```
~ brew install jenkins
```

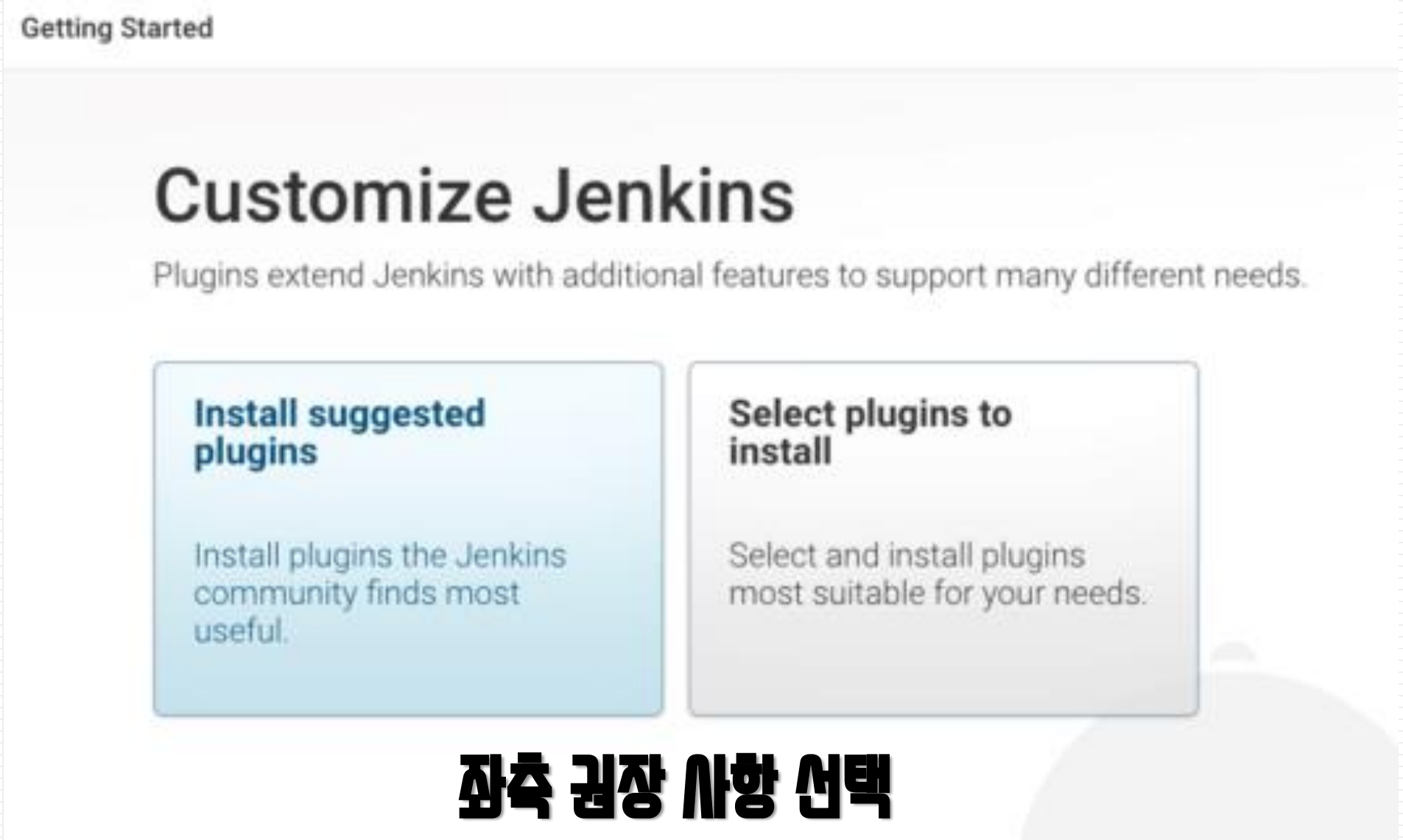
JENKINS - START

```
~ brew services start jenkins
```

JENKINS - INIT PWD

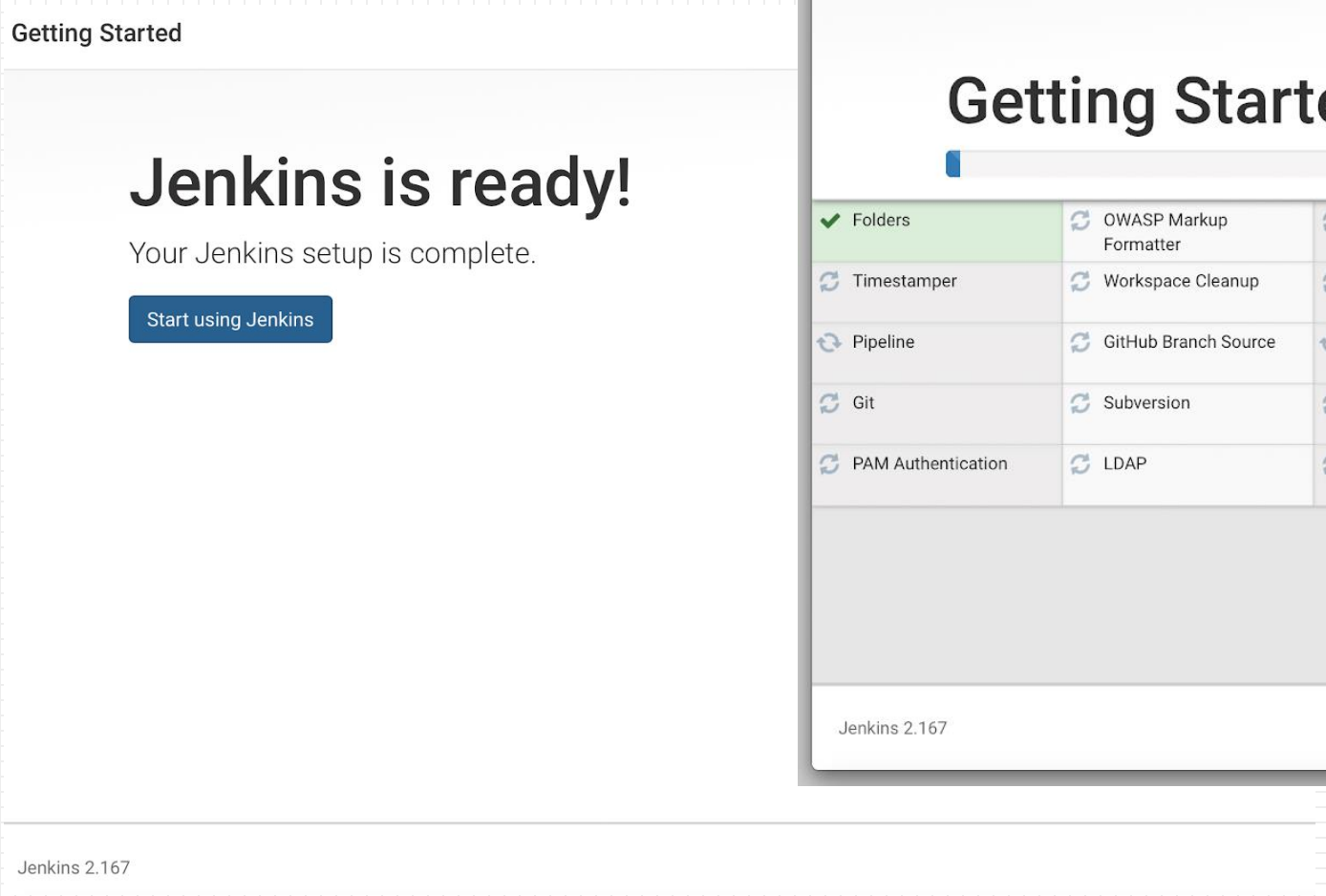
기본 포트인 localhost:8080로 접속 후
/Users/(username)/.Jenkins/secrets/intialAdminPassword에서
비밀번호 복사 후 입력

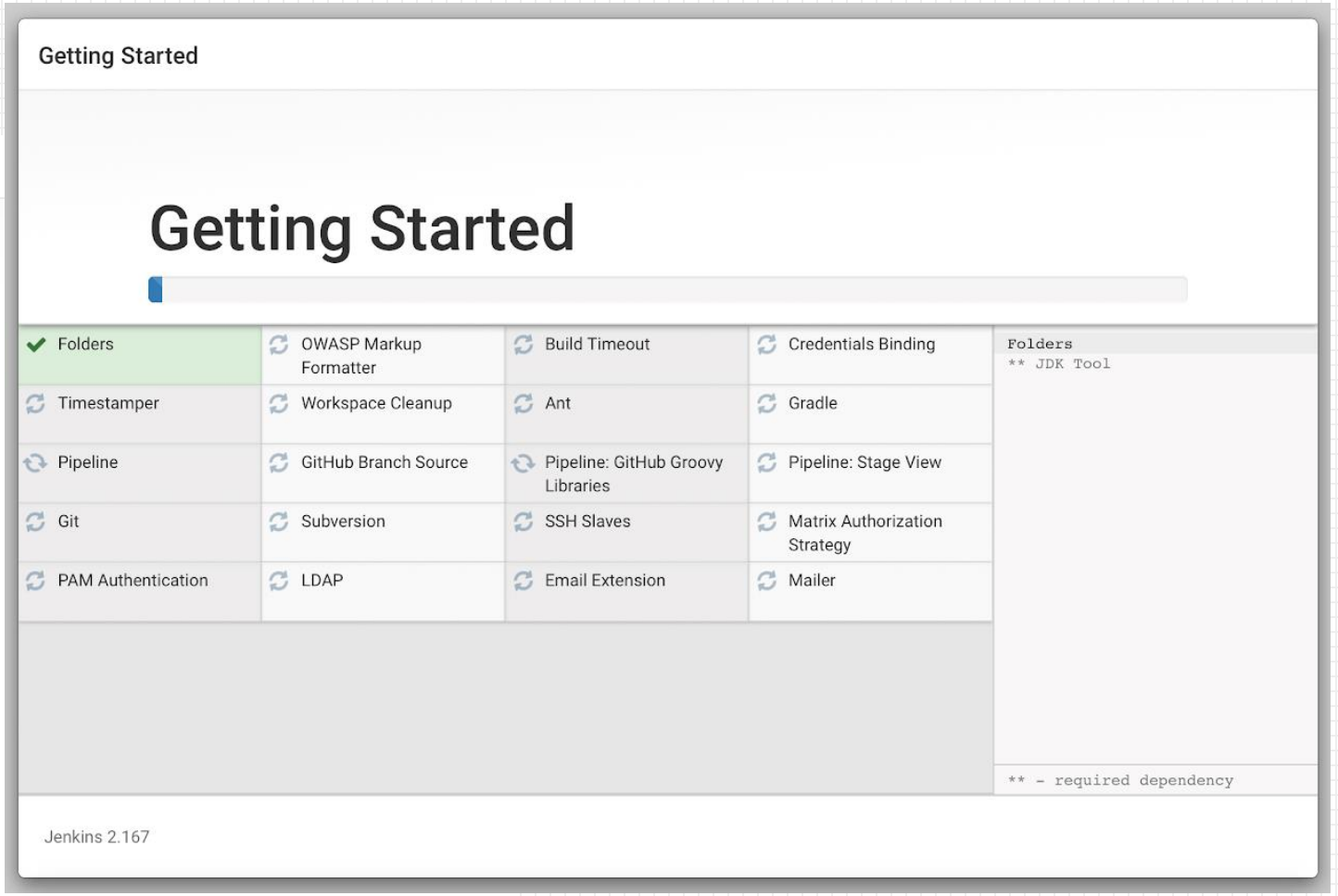
JENKINS - CUSTOMIZE



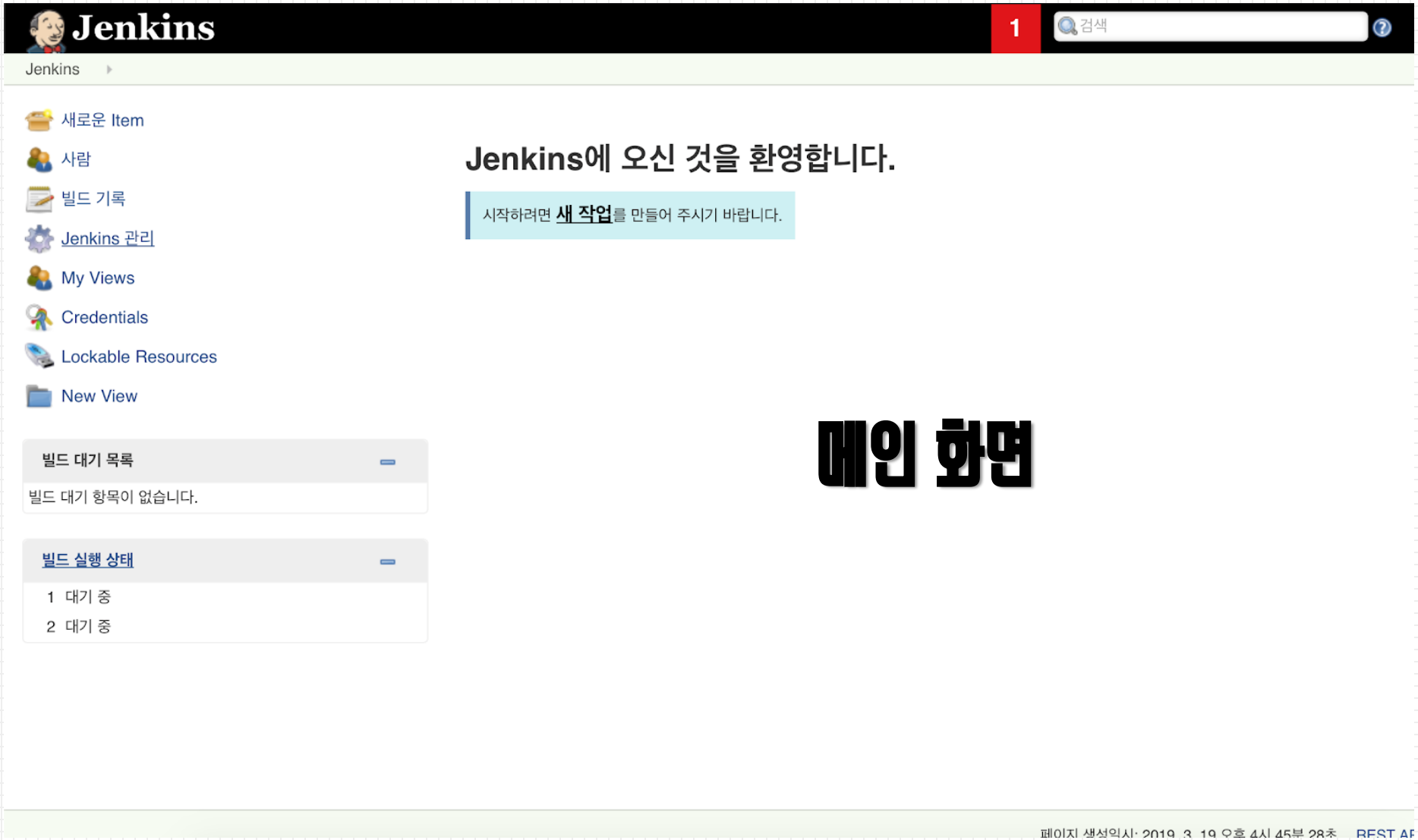
좌측 권장 사항 선택

JENKINS - INSTALL PLUGIN

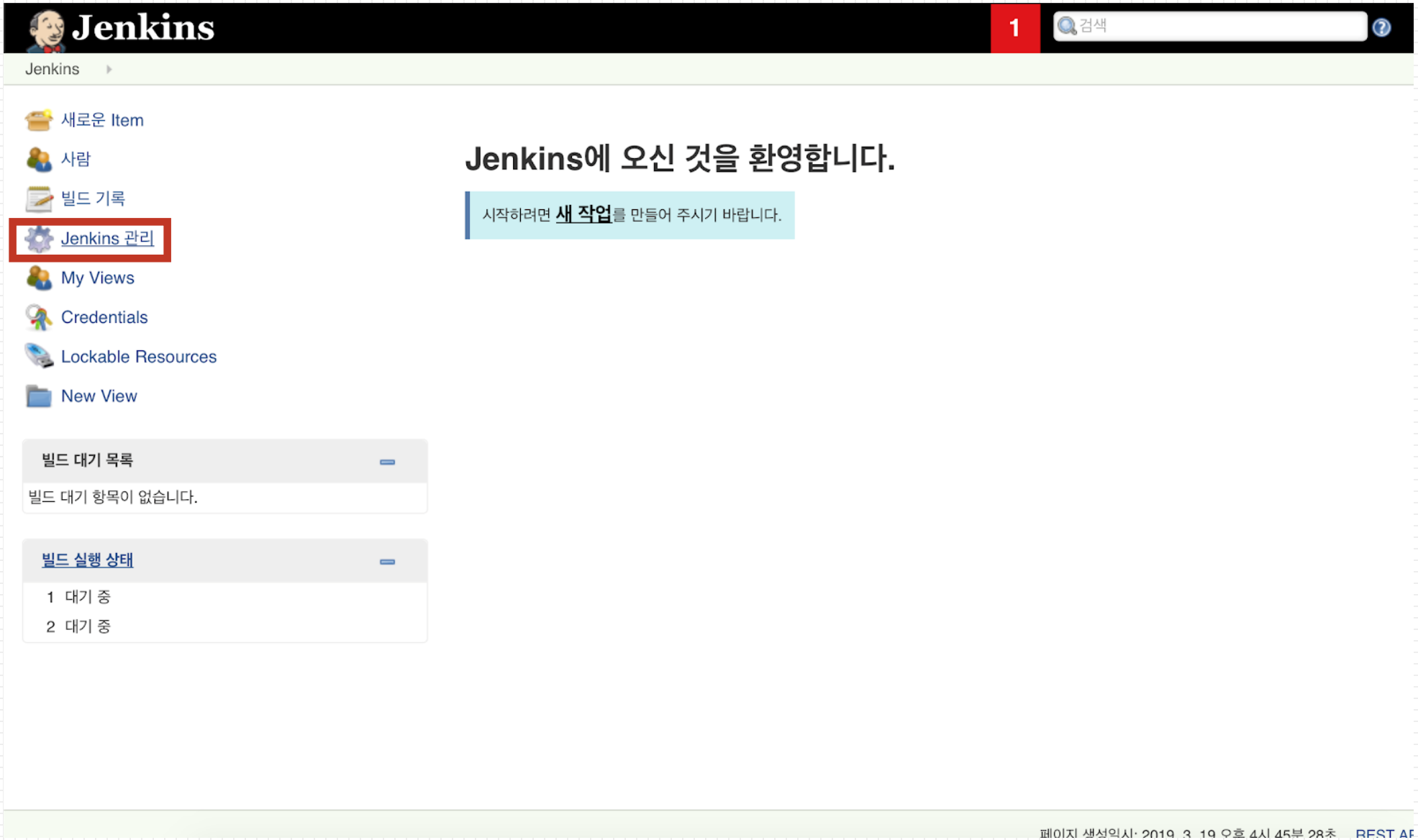




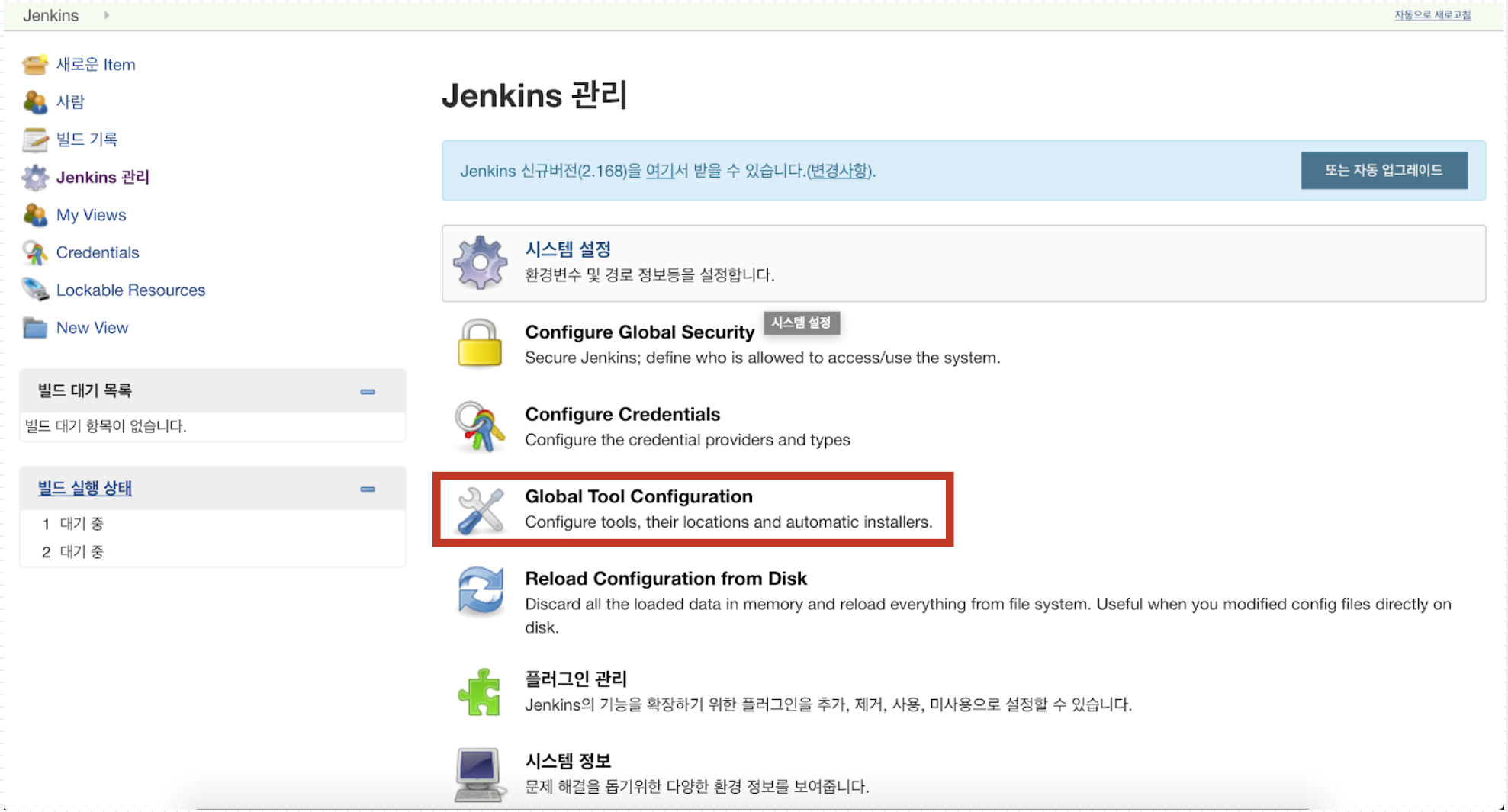
JENKINS - STARTED



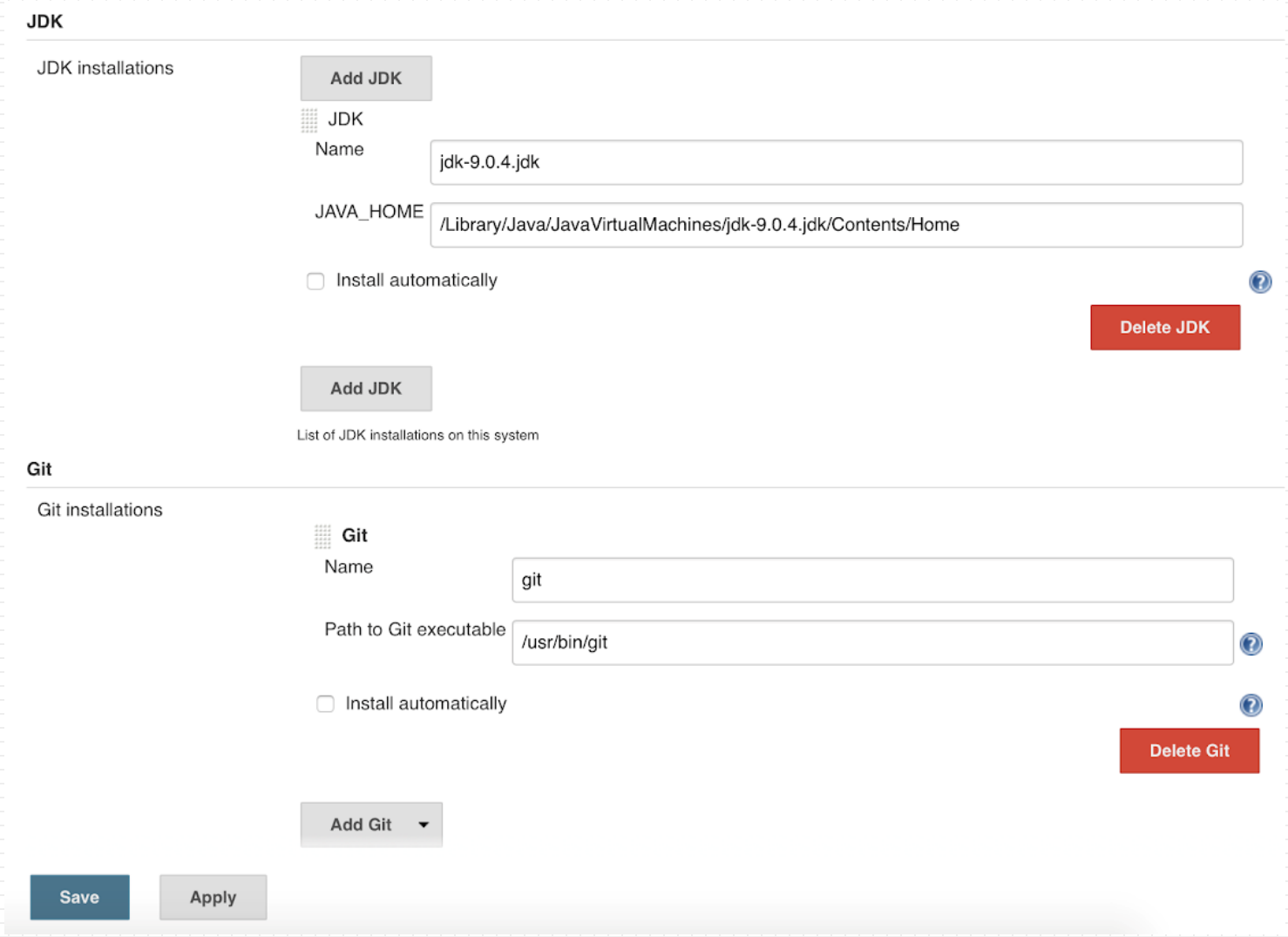
JENKINS - SETTING



JENKINS - SETTING

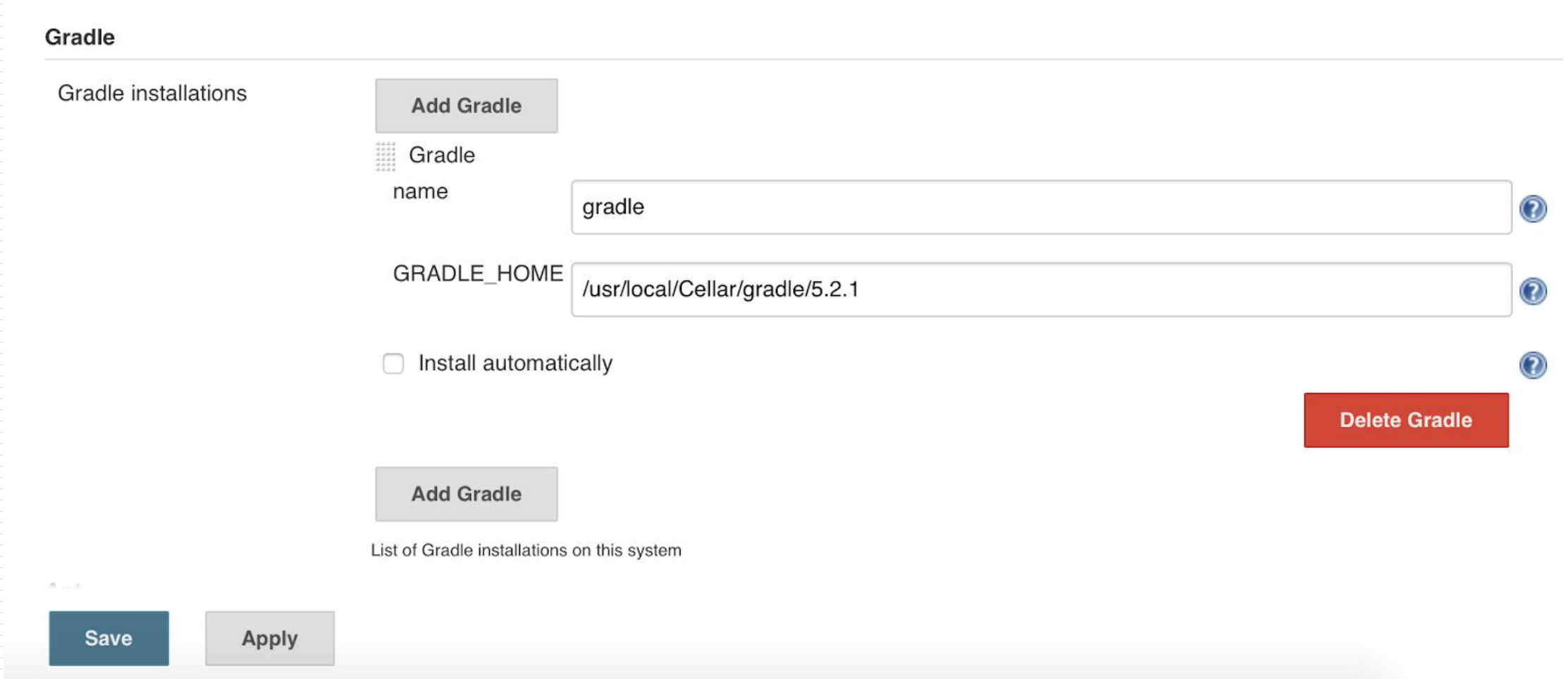


JENKINS - JDK, GIT



JDK가 위치하는
디렉토리 입력
추가적으로
Git도 설정 가능

JENKINS - GRADLE



Gradle도 마찬가지로 설치된 위치 지정

QnA