Software Verification

#3 정적분석 도구, 단위/시스템테스트 도구

Software Verification Team 4

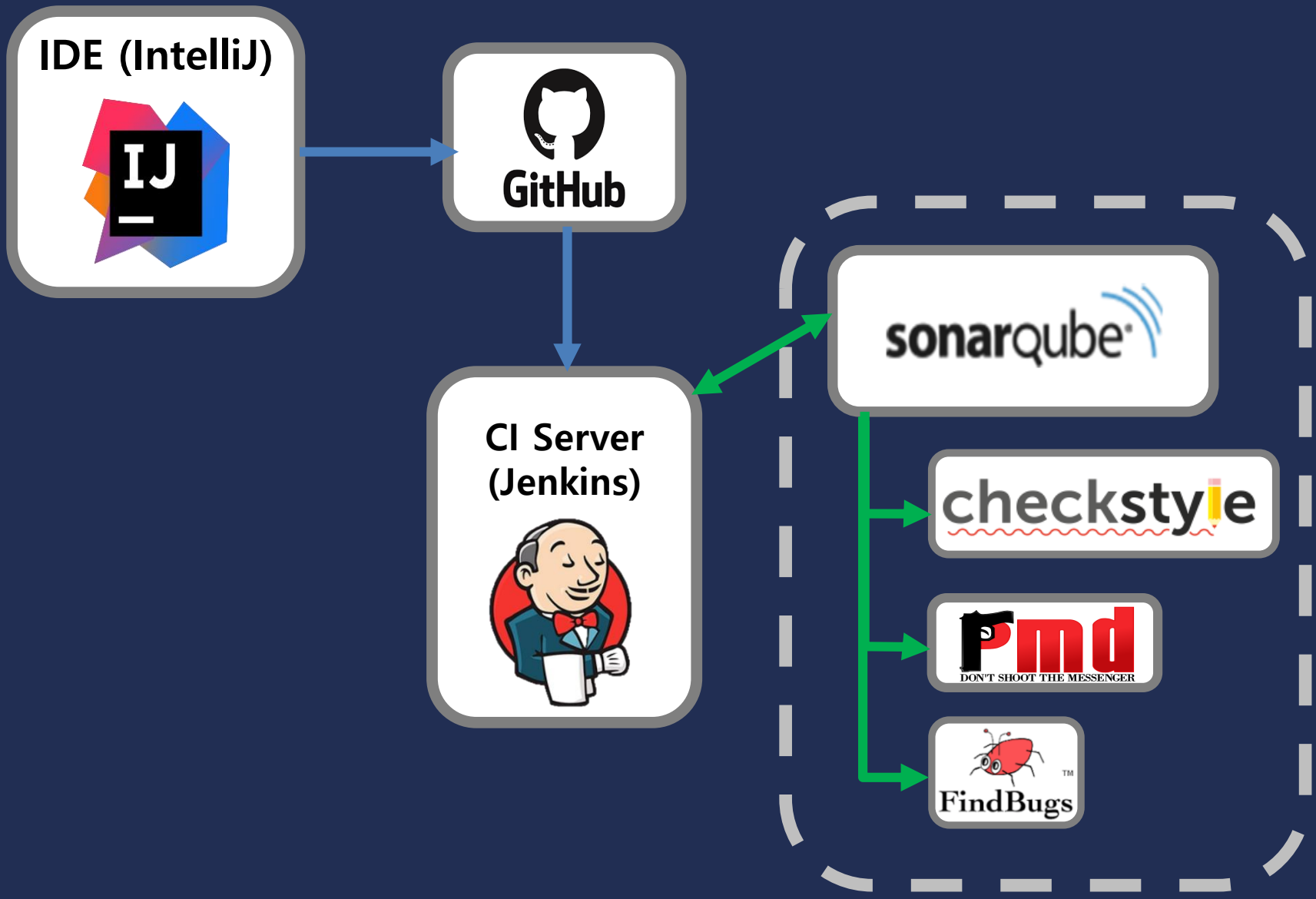
강 정 모
송 상 연
신 승 화

Software Verification

#3 정적분석 도구, 단위/시스템테스트 도구

CONTENTS

01 Overall structure



01 Overall structure

Ⅰ. Jenkins – Redmine 연동 문제

→ plugin 노후화 / gem version 충돌 문제

급한대로 바로가기를 통한 연결. 해결될 가능성은 낮지만 고민해봐야 할 문제.

(Redmine version 4.0.3 ⇒ 3.x.x 로 downgrade ??)

Ⅱ. Jenkins – Mantis 연동 문제

→ 추후 해결이 안될 시 GitHub or TestLink – Mantis 대체 연동을 고려

01 Overall structure

Ⅰ. Jenkins – Redmine 연동 문제

→ zapier webhook으로 해결.

하지만 아직 일감 분류 문제 해결이 필요.

#	유형	상태	우선순위	제목
294	결함	신규	보통	software_verification_project / 126 / SUCCESS
293	결함	신규	보통	software_verification_project / 125 / SUCCESS
292	결함	신규	보통	software_verification_project / 124 / SUCCESS
291	결함	신규	보통	software_verification_project / 123 / FAILURE
290	결함	신규	보통	software_verification_project / 122
289	결함	신규	보통	software_verification_project / 121
288	결함	신규	보통	software_verification_project / 120
287	결함	신규	보통	software_verification_project / 119

Ⅱ. Jenkins – Mantis 연동 문제

→ TestLink – Mantis 대체 연동 시 TestLink – Redmine 연동이 불가

→ 정적 분석 툴 추가 ⇒ 삭제

→ 대체제로 testcase 설치 예정 (현재 DB 설정 충돌 문제 발생)

02 Static Analyzer - SonarQube

- 조직에서 개발된 코드의 지속적인 인스펙션을 통해 품질 목표를 달성할 수 있게 해주는 플랫폼
- 소스코드 품질 현황을 시각화, 리스크 분석 및 소스코드에서 발생하는 문제를 해결

지속적인 인스펙션

지속적인 통합처럼 지속적으로 코드 인스펙션을 통해 품질 문제를 해결합니다.

품질 중앙화

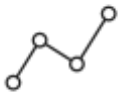
조직의 다양한 언어로 개발된 코드의 품질을 가시화하고 단일 위치에 관리합니다.

DevOps 통합

SonarQube는 다양한 빌드 시스템과 CI 엔진을 지원하여 DevOps에 쉽게 통합됩니다.

품질 요구사항 설정

품질 게이트를 통해 표준화된 코드 품질 요구사항을 설정할 수 있습니다.



다중 언어 분석

SonarQube는 20개의 프로그램 언어에 대한 코드 분석을 지원합니다.

플러그인 라이브러리

다수의 플러그인을 통해 SonarQube의 기능을 확장할 수 있습니다.

02 Static Analyzer - SonarQube

- SonarQube의 경우 기본 DB로 maven의 H2DB를 사용하므로,

  maven을 먼저 설치해준다.

  (필요 시 다른 DB로 대체 가능. 설치 후 환경변수 등록할 것.)

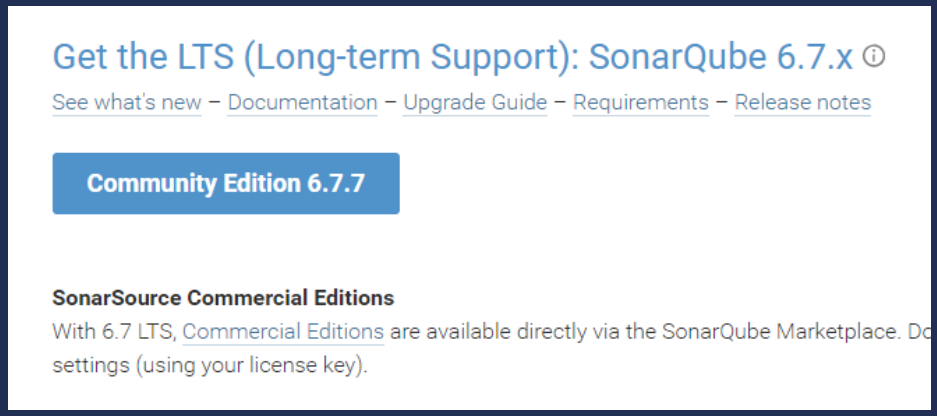
```
wget http://apache.mirror.cdnetworks.com/maven/maven-3/3.6.1/binaries/apache-maven-3.6.1-bin.tar.gz
tar -zxvf apache-maven-3.6.1-bin.tar.gz
```

02 Static Analyzer - SonarQube

- SonarQube 최신 버전을 다운로드 하여 압축을 해제하고,

  bin directory 내부의 linux directory 중 해당 OS에 맞는 directory에 들어가

  sonar.sh 파일을 실행시킨다.

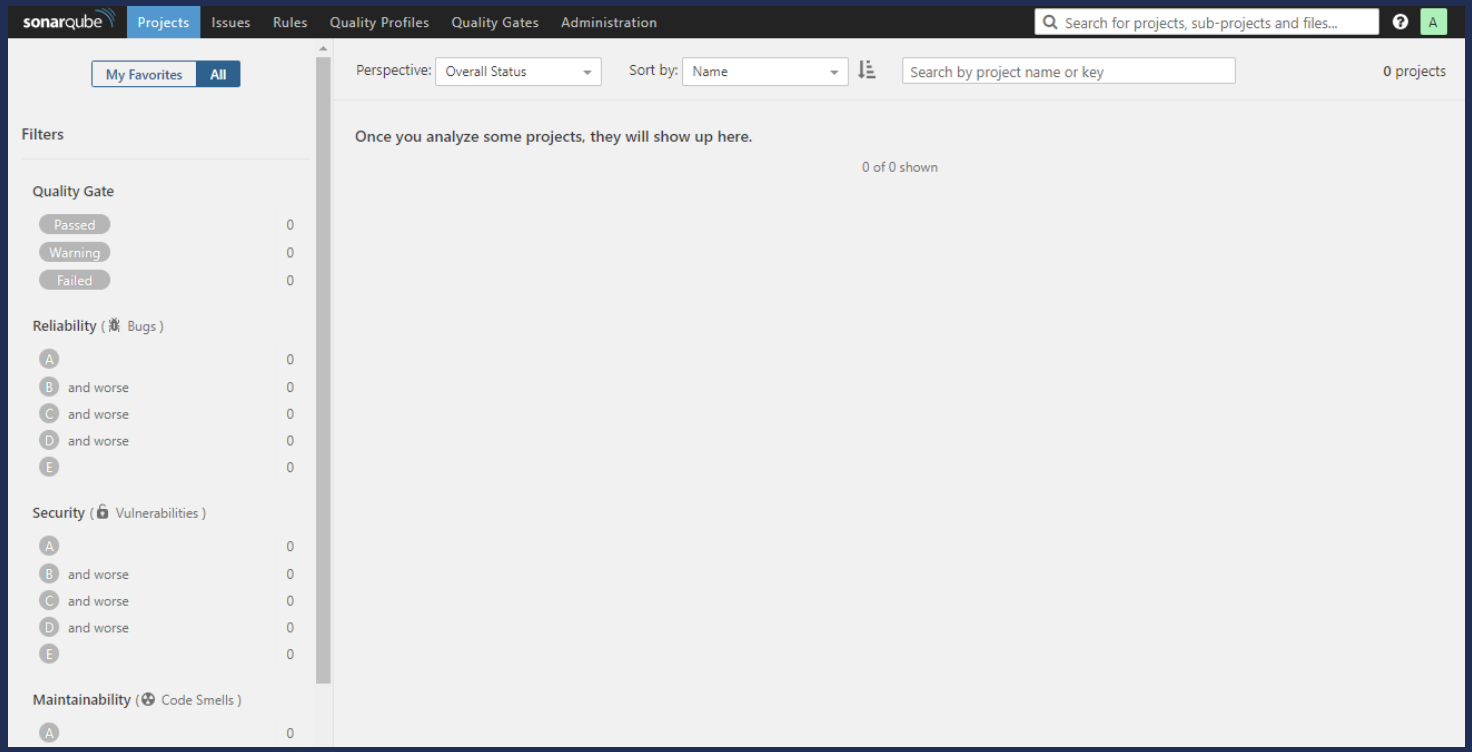



```
wget https://sonarsource.bintray.com/Distribution/sonarqube/sonarqube-6.7.7.zip
unzip sonarqube-6.7.7.zip
cd sonarqube-6.7.7/bin/linux-x86-64
./sonar.sh start
```

02 Static Analyzer - SonarQube

➢ 해당 SonarQube의 default port 번호는 9000이므로,

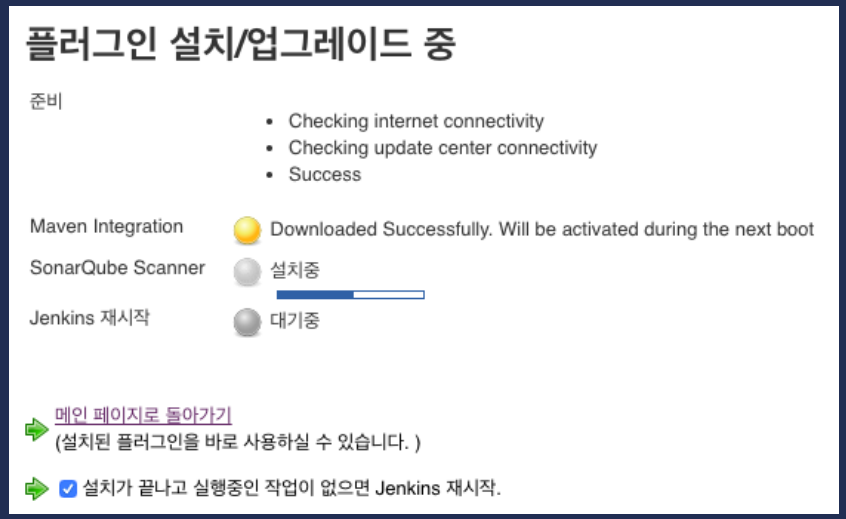
server DNS 뒤에 :9000 port 번호를 입력하여 SonarQube에 접속한다.



02 Static Analyzer - SonarQube

➢ Jenkins 와 연동하기 위해 SonarQube Scanner Plugin과

Sonar scanner를 설치한다.



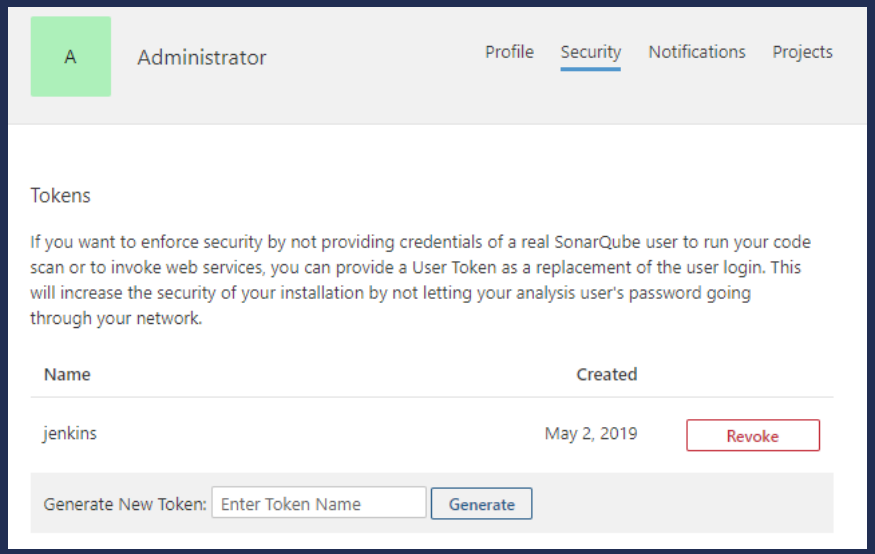

```
wget https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cli-3.3.0.1492-linux.zip
unzip sonar-scanner-cli-3.3.0.1492-linux.zip
mv /home/JungMo/sonar-scanner-cli-3.3.0.1492-linux /opt/sonar-scanner
```

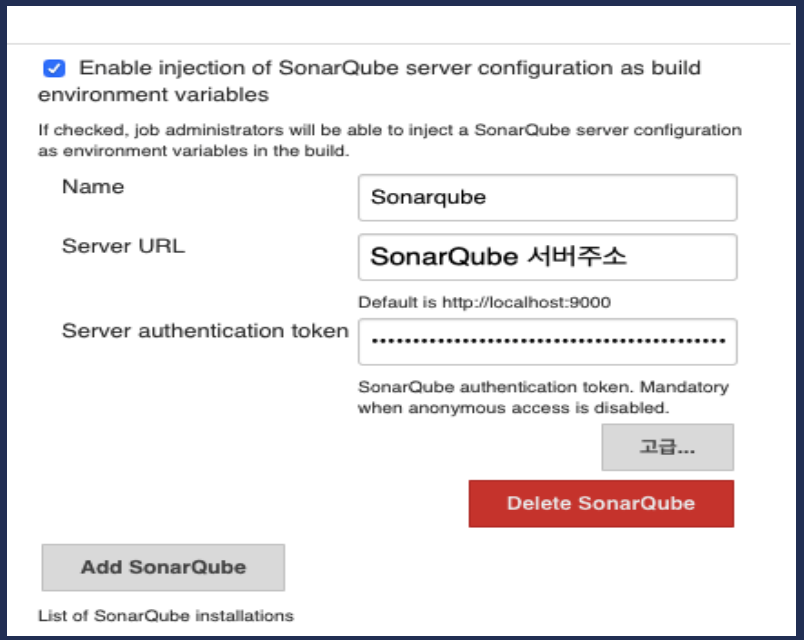
02 Static Analyzer - SonarQube

➢ SonarQube Token 생성 후

해당 Token과 SonarQube server 주소를 이용하여

Jenkins에 SonarQube server 등록한다.

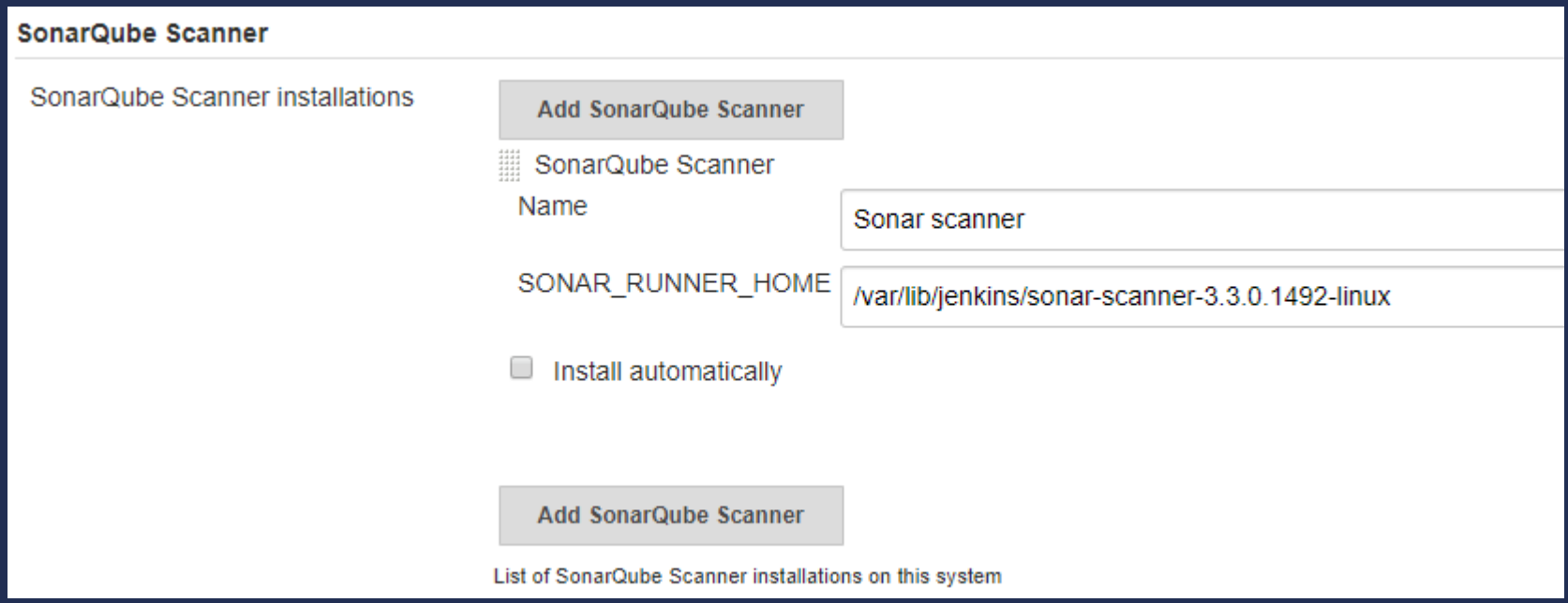




02 Static Analyzer - SonarQube

➢ 설치한 Sonar Scanner를 Jenkins와 연동시켜 준다.

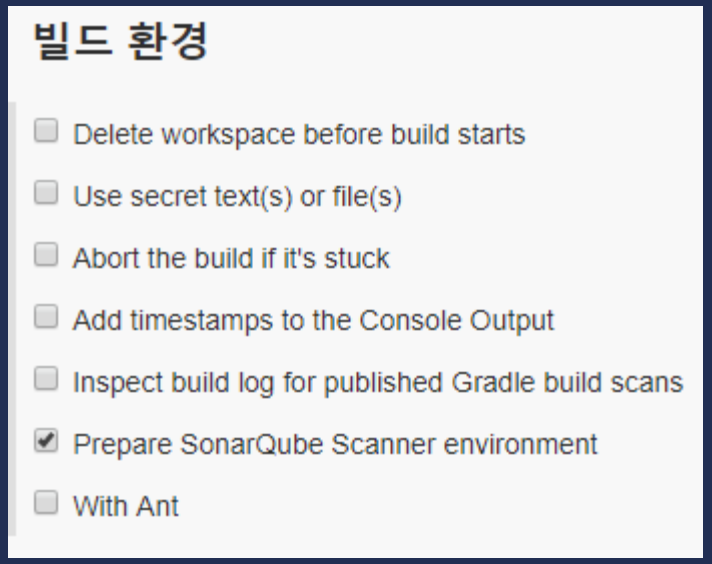
```
mv /opt/sonar-scanner /var/lib/jenkins/sonar-scanner-3.3.0.1492-linux
```

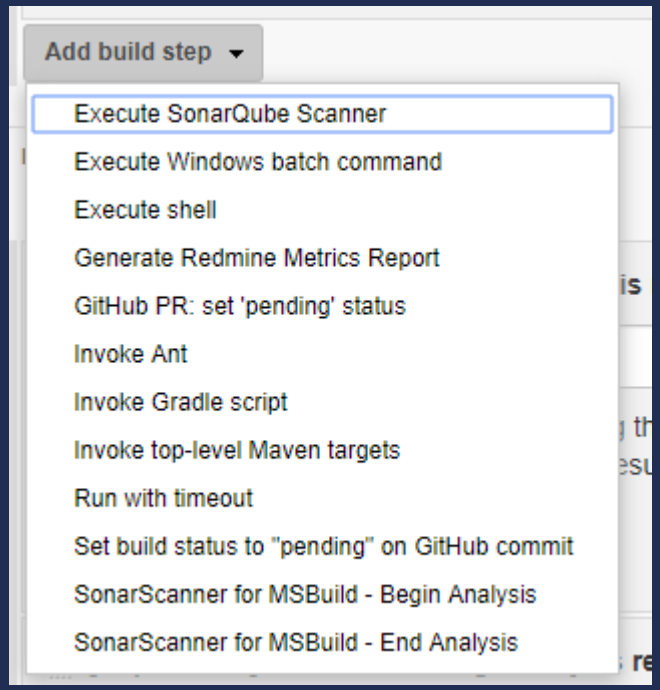


02 Static Analyzer - SonarQube

➢ SonarQube와 연동하기 위한 Jenkins project로 이동하여

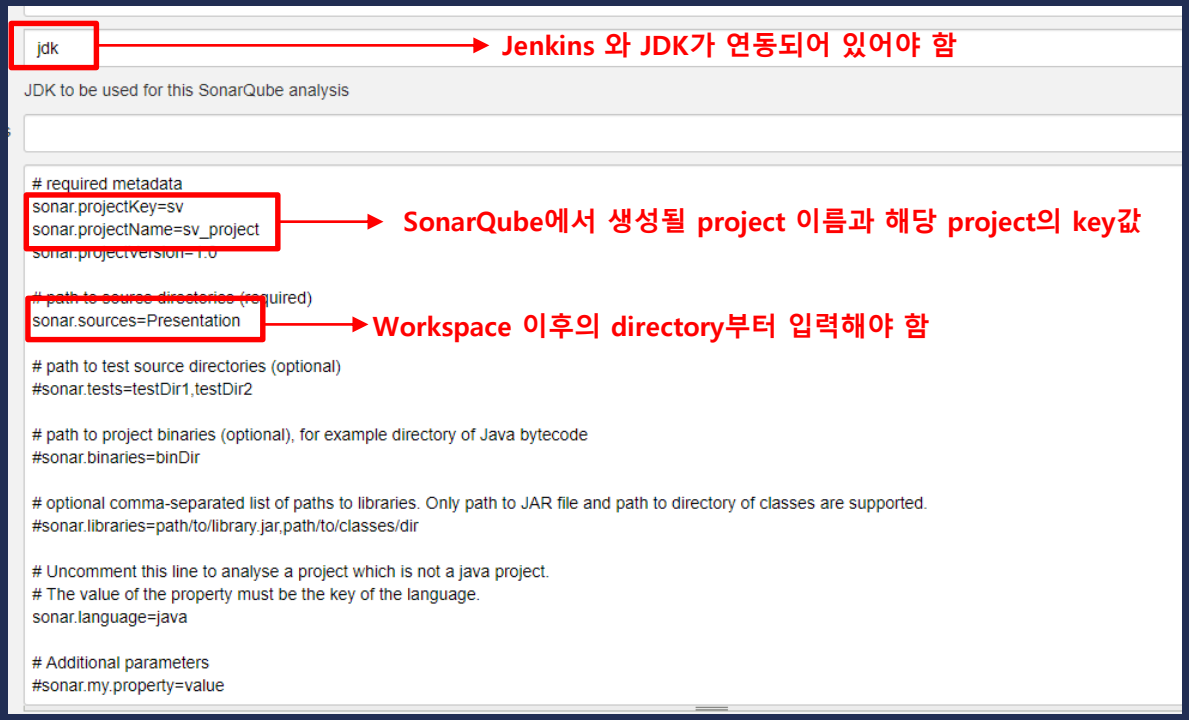
빌드 환경을 설정한다.





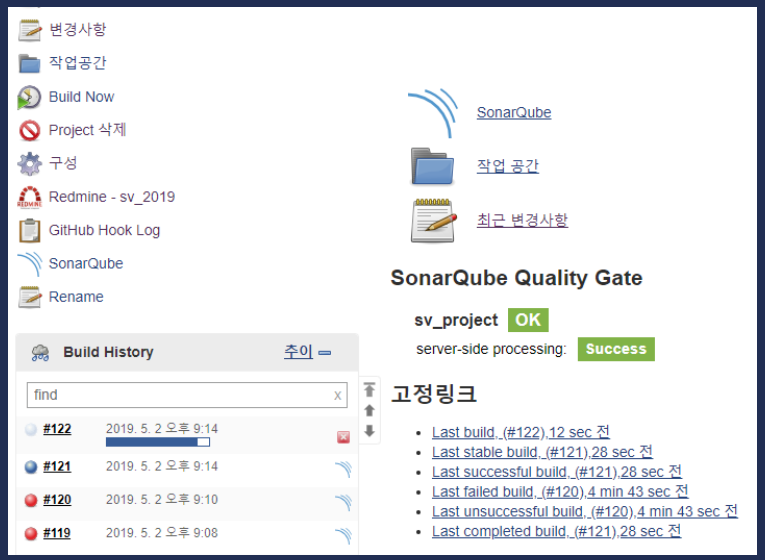
02 Static Analyzer - SonarQube

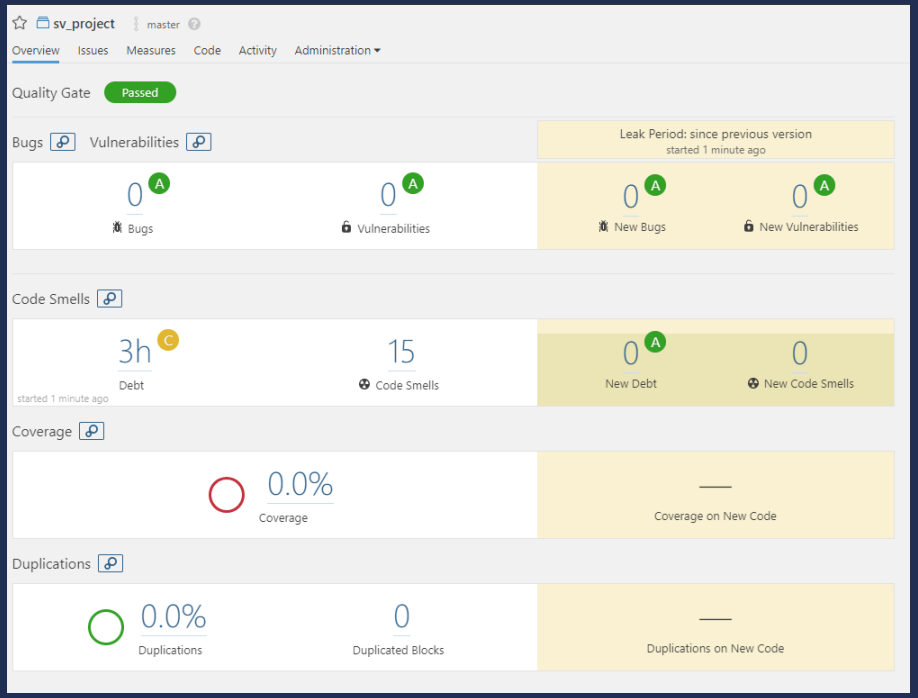
➢ SonarQube와 연동하기 위한 Jenkins project로 이동하여 빌드 환경을 설정한다.



02 Static Analyzer - SonarQube

➢ 실행





03 Static Analyzer - PMD

PMD(Program May Dependable)

- 미사용 변수, 비어있는 코드 block, 불필요한 object 생성과 같이 defect을 유발할 수 있는 코드를 검사한다.
- 미리 정의한 ruleset을 기반으로 구문을 분석한다.
- Java에서 많이 사용하지만, JavaScript, XML 등 다른 언어도 지원한다.

03 Static Analyzer - PMD

➢ 사용할 ruleset을 선택하여 pmd-rule-set.xml 생성

(아래는 생성한 xml파일의 일부)

```
<rule ref="category/java/bestpractices.xml/AvoidStringBufferField" />
<rule ref="category/java/bestpractices.xml/AvoidUsingHardCodedIP" />
<rule ref="category/java/bestpractices.xml/CheckResultSet" />
<rule ref="category/java/bestpractices.xml/ConstantsInInterface" />
<rule ref="category/java/bestpractices.xml/DefaultLabelNotLastInSwitchStmt" />
<rule ref="category/java/bestpractices.xml/ForLoopCanBeForeach" />
<rule ref="category/java/bestpractices.xml/GuardLogStatement" />
<rule ref="category/java/codestyle.xml/FormalParameterNamingConventions" />
<rule ref="category/java/codestyle.xml/ClassNamingConventions" />
<rule ref="category/java/codestyle.xml/LocalVariableNamingConventions" />
<rule ref="category/java/codestyle.xml/MethodNamingConventions" />
<rule ref="category/java/codestyle.xml/PackageCase" />
<rule ref="category/java/design.xml/SimplifyBooleanReturns" />
<rule ref="category/java/design.xml/SimplifyConditional" />
<rule ref="category/java/design.xml/SingularField" />
```

https://github.com/jensgerdes/sonar-pmd/blob/master/docs/RULES.md

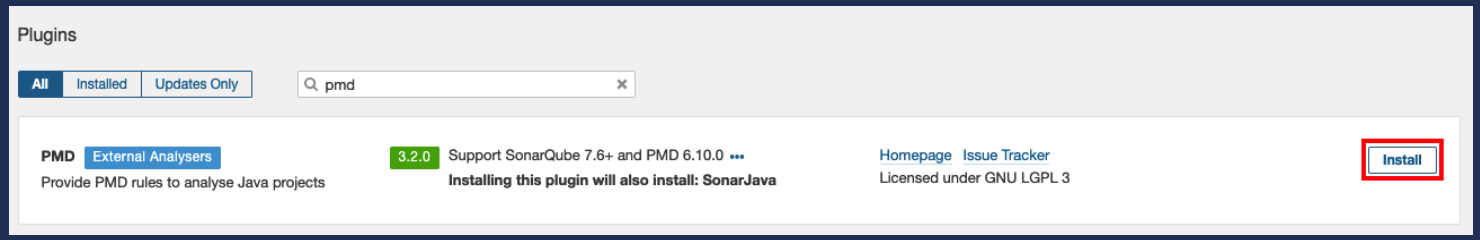
> PMD ruleset reference

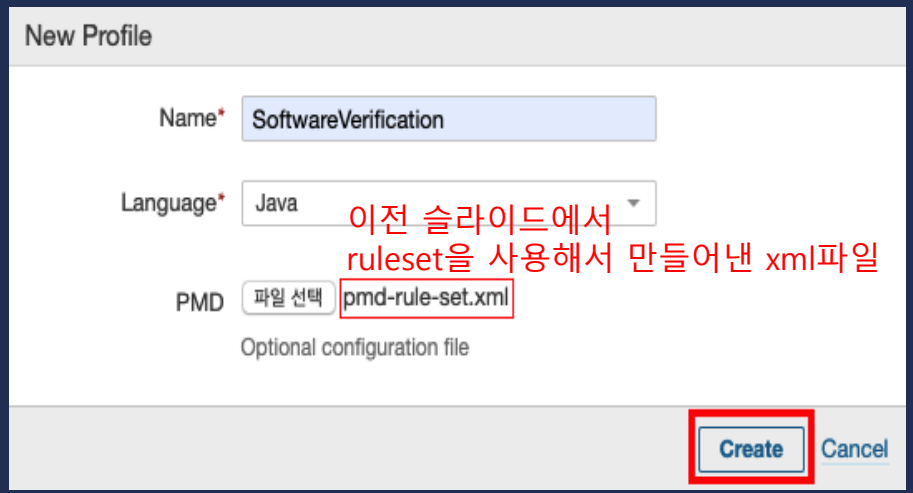
03 Static Analyzer - PMD

- SonarQube의 Market Place에서 PMD plugin을 검색하여 추가하고,

  Quality Profiles tab 에서 Create를 통해 ruleset을 모은

  New Profile을 만든다.

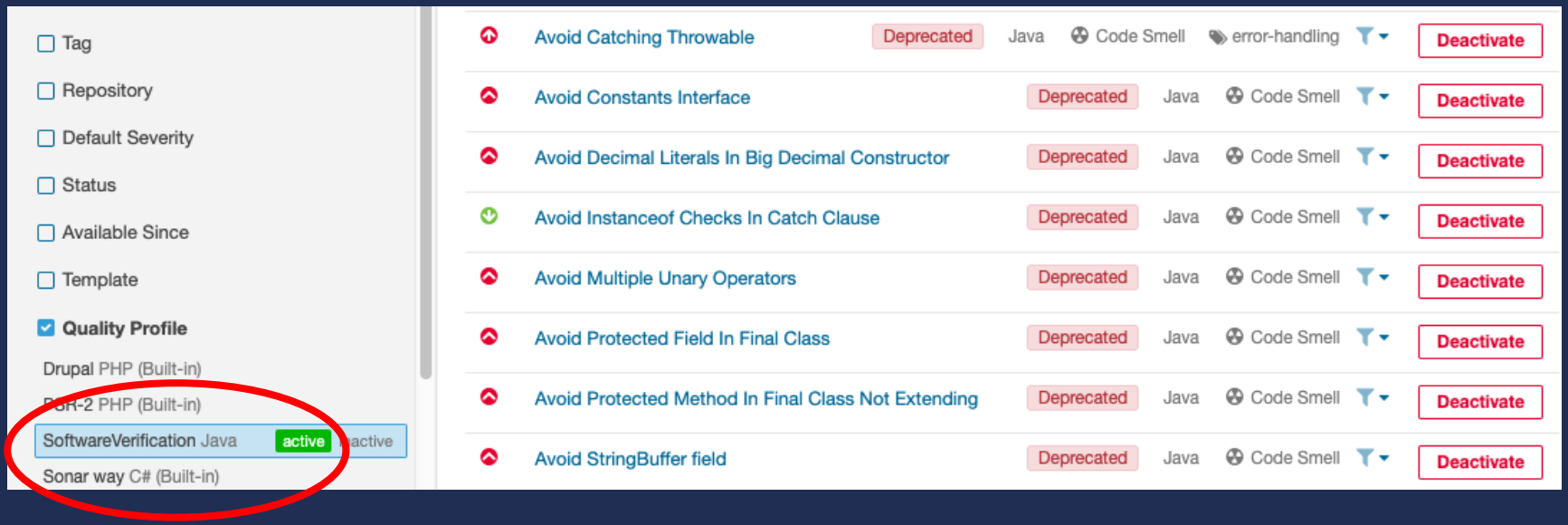




03 Static Analyzer - PMD

➢ 다음과 같이 Quality Profile 항목에

PMD ruleset을 활용한 SoftwareVerification Profile이 import된 것을 볼 수 있다.



04 Static Analyzer – Checkstyle

- 소스코드 내에서 다양한 위반사항에 대해 알 수 있다.
- 개발자들이 체크인 전에 위반사항을 고칠 수 있다.
- 미리 정해 놓은 코딩 규칙을 팀원들이 보다 쉽게 적용할 수 있게 도와주는 도구

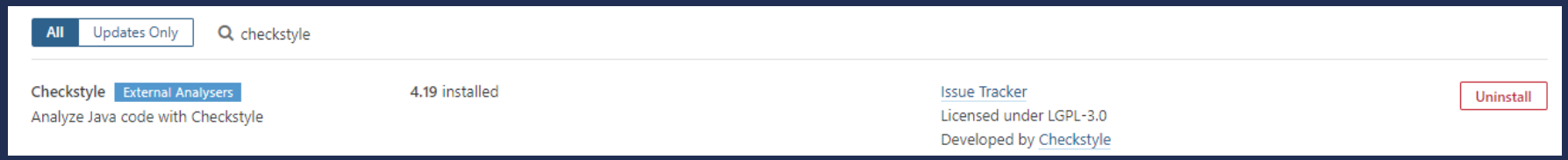
→ error를 조기에 찾을 수 있음.

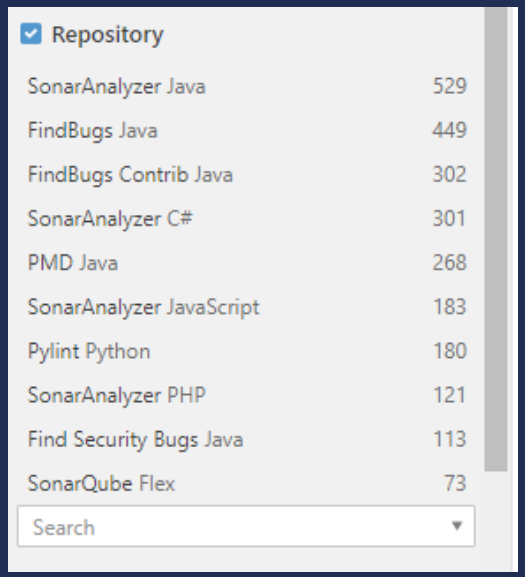
ex) 사용하지 않은 pointer는 Null로 초기화 한다.

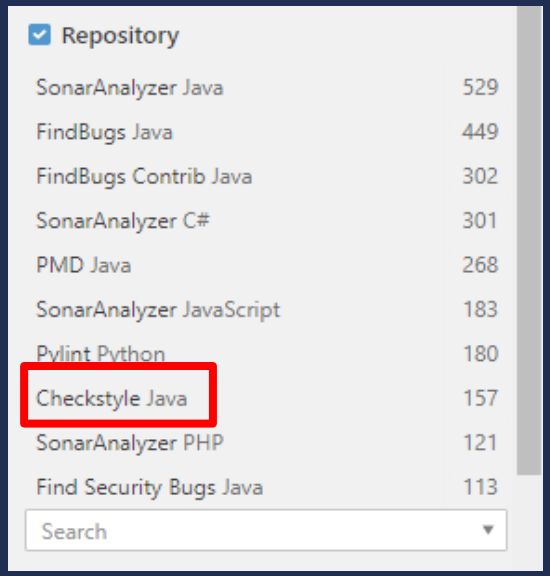
04 Static Analyzer – Checkstyle

➢ PMD와 마찬가지로 Market Place에서 Checkstyle Plugin을 찾아 설치한다.

설치 시 Rules Repository에 Checkstyle Java 항목이 추가된다.







05 Static Analyzer – FindBugs

- Java program에서 잠재적으로 발생 가능한 버그를 찾는 것이 주 목적인 정적 분석 tool.
- Complie된 java byte code를 읽어서 검사

  → 속도가 빠름, 하지만 build 과정이 필수

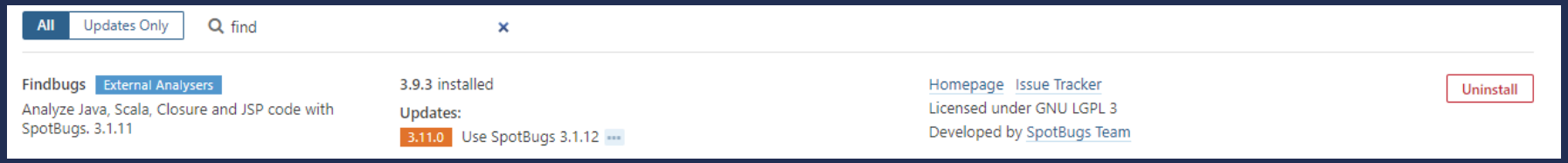
  (SonarQube에서는 해당 option 선택 가능)
- 실제 결함/버그를 잘 찾아주는 편이다.

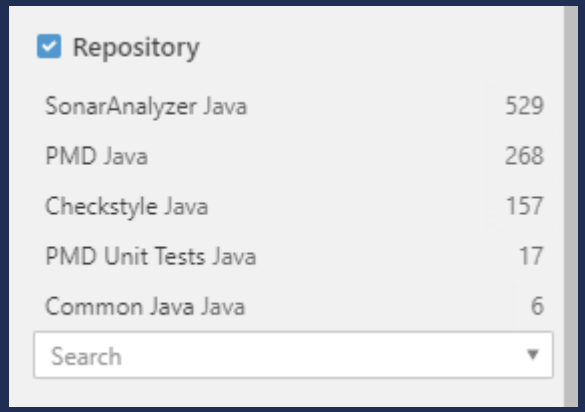
  (static analysis에서 발생가능한 false alarm이 존재할 수 있음)

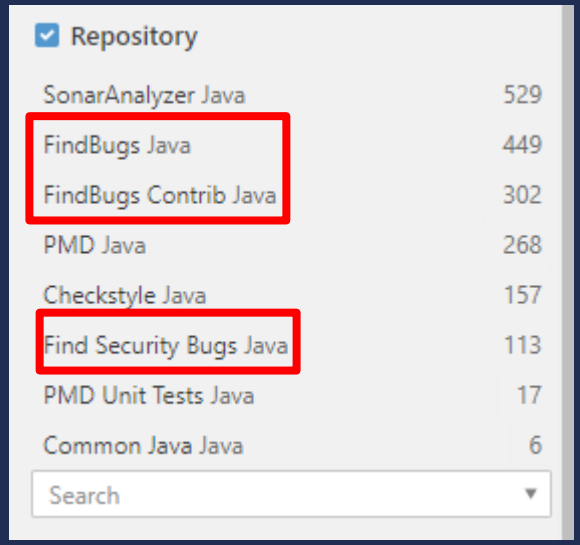
05 Static Analyzer – FindBugs

➢ 역시 FindBugs도 Market Place에서 FindBugs Plugin을 찾아 설치한다.

설치 시 Rules Repository에 각종 FindBugs Java 항목이 추가된다.

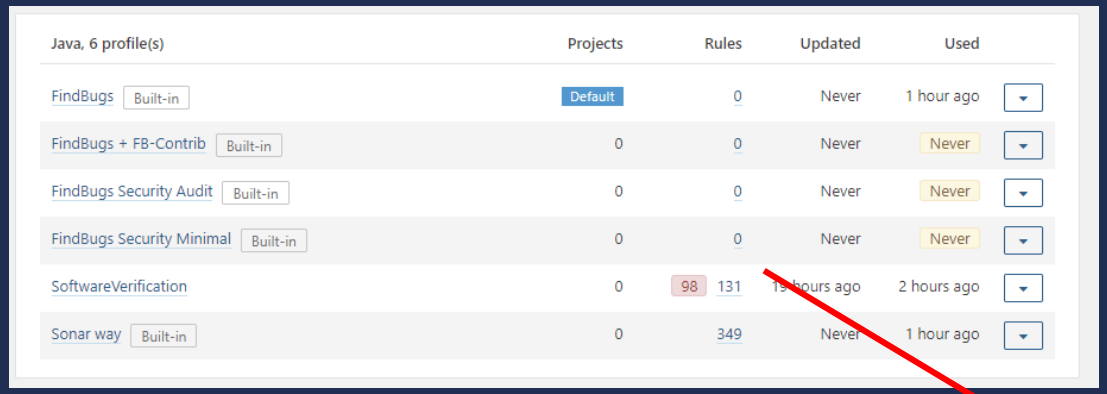


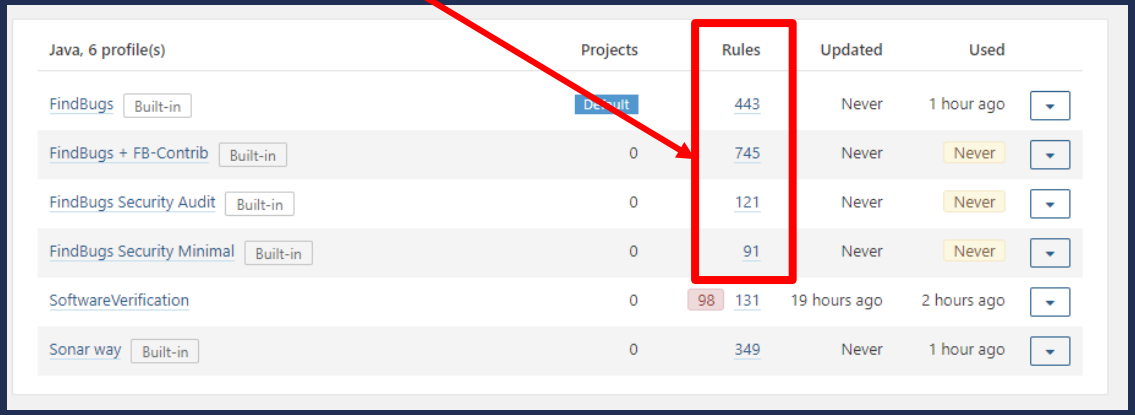




05 Static Analyzer – FindBugs

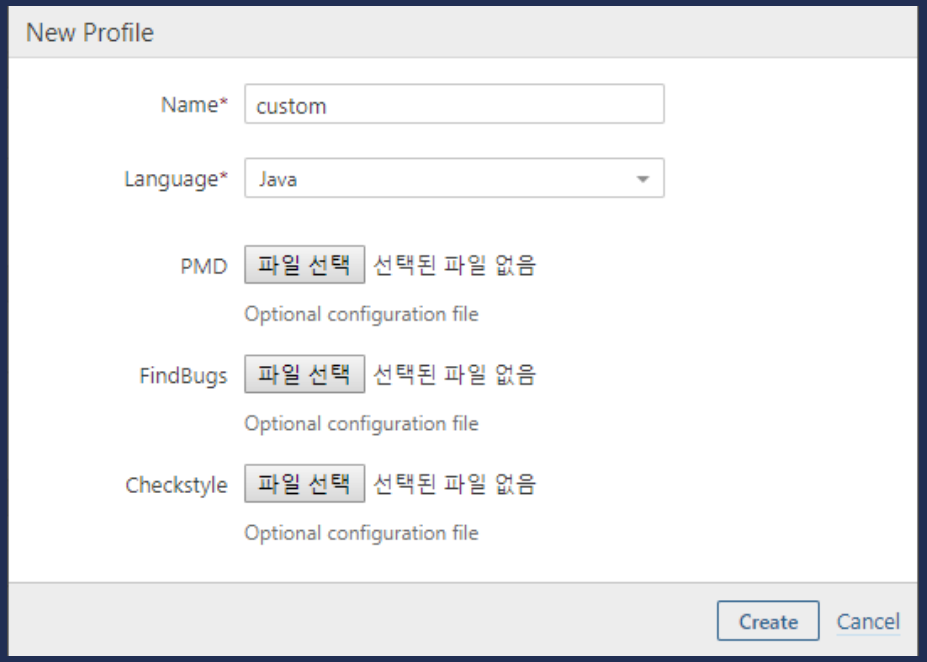
➢ Quality Profile 쪽에도 Rules들이 추가된다.





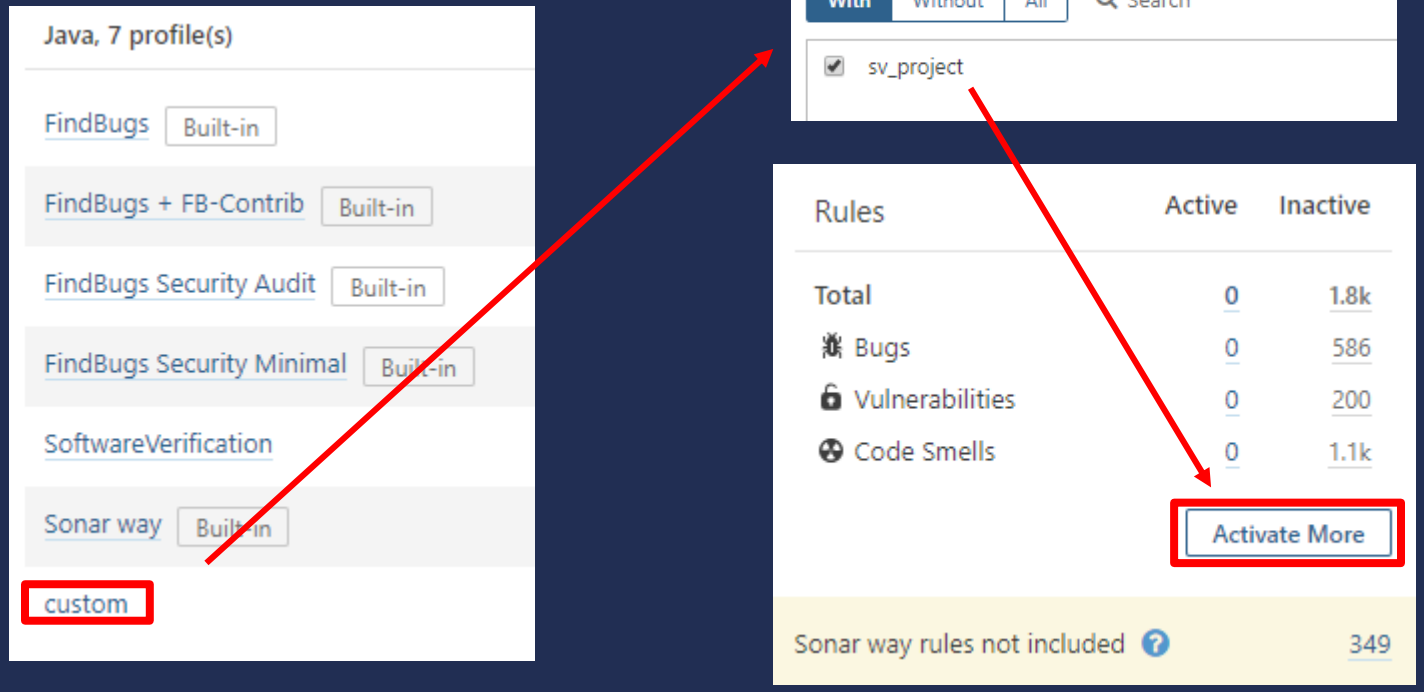
06 Summary

➢ Quality Profile 탭 우측 상단의 Create를 눌러 Profile을 만든다.

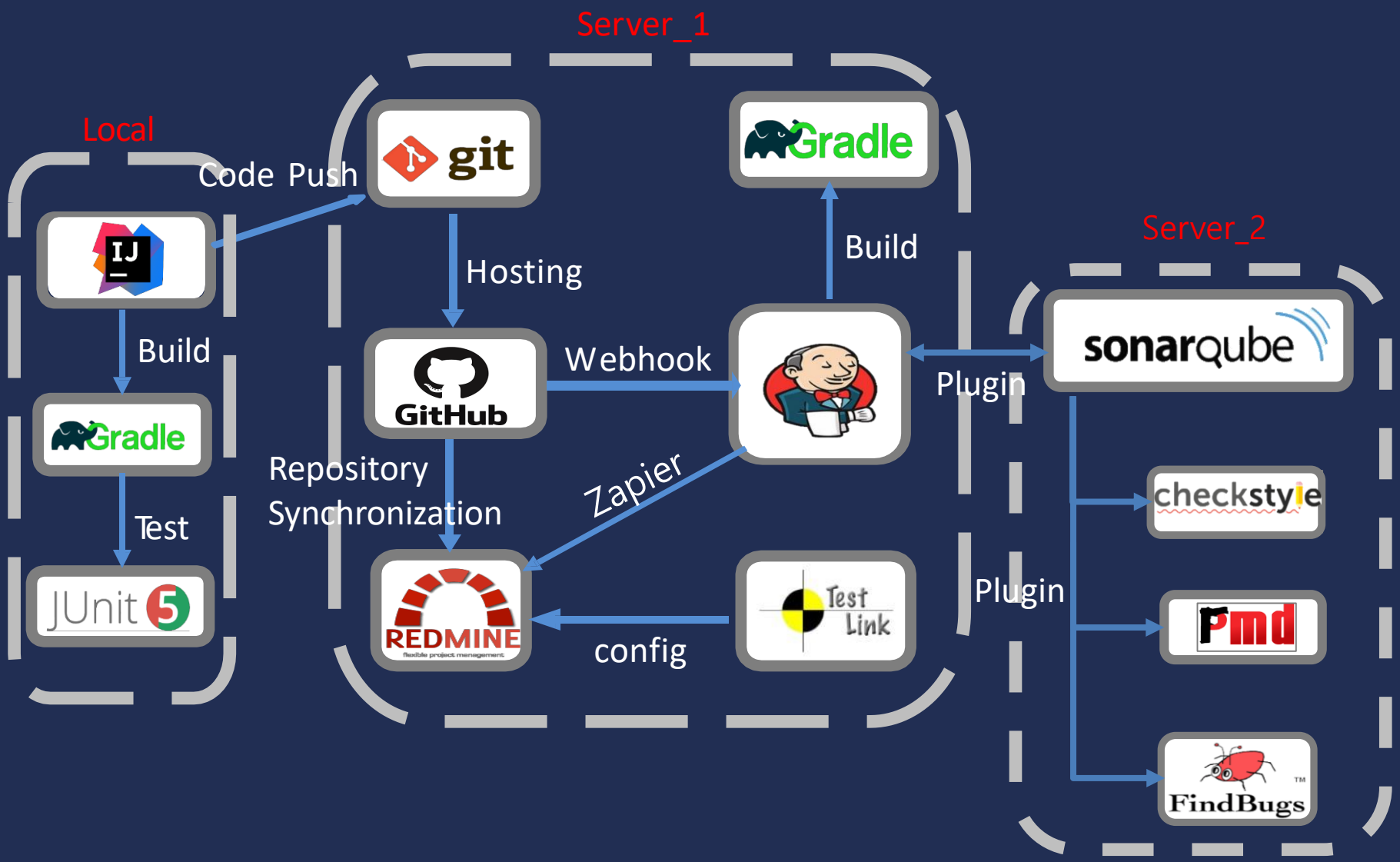


06 Summary

➢ 생성한 Profile을 선택한 후 Change Project로 SonarQube project를 할당할 수 있고, 할당 이후에는 Activate More 을 클릭하여 각종 rule을 마음대로 추가할 수 있다.



06 Summary


Server_1
Local
Code Push
git
Gradle
IJ
Hosting
Build
Build
Gradle
GitHub
Webhook
Server_2
sonarqube
Plugin
Test
Repository
Synchronization
Zapier
checkstyle
JUnit 5
REDMINE
Test Link
config
Plugin
Pmd
FindBugs

감사합니다

THANK YOU

Software Verification Team 4

강 정 모
송 상 연
신 승 화