

Jenkins Test

단위 테스트

❖ 개요

- ✓ 유닛 테스트(unit test)는 컴퓨터 프로그래밍에서 소스 코드의 특정 모듈이 의도된 대로 정확히 작동하는지 검증하는 절차
- ✓ 모든 함수와 메소드에 대한 테스트 케이스(Test case)를 작성하는 절차로 이를 통해서 언제라도 코드 변경으로 인해 문제가 발생할 경우 단시간 내에 이를 파악하고 바로 잡을 수 있도록 함
- ✓ 각 테스트 케이스는 서로 분리되어야 하고 이를 위해 가짜 객체(Mock object)를 생성하는 것도 좋은 방법
- ✓ 유닛 테스트는 (일반적인 테스트와 달리) 개발자(developer) 뿐만 아니라 보다 더 심도있는 테스트를 위해 테스터(tester)에 의해 수행되기도 함



단위 테스트

❖ 장점

✓ 문제점 발견

- 유닛 테스트의 목적은 프로그램의 각 부분을 고립시켜서 각각의 부분이 정확하게 동작하는지 확인하는 것으로 프로그램을 작은 단위로 쪼개서 각 단위가 정확하게 동작하는지 검사하고 이를 통해 문제 발생 시 정확하게 어느 부분이 잘못되었는지를 재빨리 확인할 수 있기 때문에 프로그램의 안정성이 높아짐
- 유닛 테스트는 개발 시간을 증가시키는 것처럼 보이지만 개발 기간 중 대부분을 차지하는 디버깅 시간을 단축시킴으로써 여유로운 프로그래밍이 가능

✓ 변경이 쉬움

- 리팩토링 후에도 해당 모듈이 의도대로 작동하고 있음을 유닛 테스트를 통해서 확신할 수 있는데 이를 회귀 테스트(Regression testing)라 하며 어떻게 코드를 고치더라도 문제점을 금방 파악할 수 있고 수정된 코드가 정확하게 동작하는지 쉽게 알 수 있게 되므로 프로그래머들은 더욱 더 의욕적으로 코드를 변경할 수 있게 됨
- 지속적인 유닛 테스트 환경을 구축하면 어떠한 변화가 있더라도 코드와 그 실행이 의도대로 인지를 확인하고 검증할 수 있게 되며 확립된 개발 방법과 유닛 테스트의 범위에 따라서 프로그램의 정확성이 좌우된다.

✓ 통합이 간단

- 유닛 테스트는 유닛 자체의 불확실성을 제거해주므로 상향식(bottom-up) 테스트 방식에서 유용
- 먼저 프로그램의 각 부분을 검증하고 그 부분들은 합쳐서 다시 검증하는 통합 테스트에서 더욱 더 빛을 발함

단위 테스트

❖ Spring Boot 단위 테스트

✓ MySQL 실행

```
docker run --name mysql-container -e MYSQL_ROOT_PASSWORD=<password> -d -p 3306:3306 mysql:latest
```



단위 테스트

❖ Spring Boot 단위 테스트

- ✓ gradle 기반의 Spring Boot Project 생성(Spring Boot DevTools, Lombok, Spring Web, Spring Data JPA, H2 Database, MySQL Driver 의존성 추가)

- ✓ build.gradle에서 의존성 확인

```
dependencies {  
    implementation 'org.springframework.boot:spring-boot-starter-web'  
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'  
  
    // Database  
    runtimeOnly 'com.h2database:h2'  
    runtimeOnly 'com.mysql:mysql-connector-j'  
  
    // Lombok & DevTools  
    compileOnly 'org.projectlombok:lombok'  
    annotationProcessor 'org.projectlombok:lombok'  
    developmentOnly 'org.springframework.boot:spring-boot-devtools'  
  
    // Test (MockMvc, JUnit5, DataJpaTest 등 모든 테스트 도구 포함)  
    testImplementation 'org.springframework.boot:spring-boot-starter-test'  
    testRuntimeOnly 'org.junit.platform:junit-platform-launcher'  
}
```

단위 테스트

❖ Spring Boot 단위 테스트

- ✓ build.gradle에서 spring-framework 버전 수정

```
plugins {  
    id 'java'  
    id 'org.springframework.boot' version '3.4.3'  
    id 'io.spring.dependency-management' version '1.1.7'  
}
```



단위 테스트

❖ Spring Boot 단위 테스트

- ✓ src/main/resources/application.yml 파일 작성

```
server:  
  port: 9090
```

```
spring:  
  config:  
    activate:  
      on-profile: default
```

```
datasource:  
  driver-class-name: com.mysql.cj.jdbc.Driver  
  url:  
    jdbc:mysql://localhost:3306/mysql?allowPublicKeyRetrieval=true&useSSL=false&serverTime  
    zone=Asia/Seoul&characterEncoding=UTF-8  
  username: root  
  password: wnddkd
```



단위 테스트

❖ Spring Boot 단위 테스트

- ✓ src/main/resources/application.yml 파일 작성

jpa:

open-in-view: true

hibernate:

ddl-auto: create

naming:

physical-strategy:

org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl

use-new-id-generator-mappings: false

show-sql: true

properties:

hibernate.format_sql: true

dialect: org.hibernate.dialect.MySQL8InnoDBDialect

logging:

level:

org.hibernate.SQL: debug



단위 테스트

❖ Spring Boot 단위 테스트

- ✓ test/resources/application.yml 파일 작성

spring:

datasource:

H2 인메모리 DB 설정

url: jdbc:h2:mem:testdb;DB_CLOSE_DELAY=-1;DB_CLOSE_ON_EXIT=FALSE

driver-class-name: org.h2.Driver

username: sa

password:

jpa:

Hibernate가 DB 종류를 자동으로 인식하지 못할 때 명시 (방언 설정)

database-platform: org.hibernate.dialect.H2Dialect

hibernate:

ddl-auto: create-drop

show-sql: true

properties:

hibernate:

format_sql: true

DialectFactory 관련 에러가 지속될 경우 아래 설정도 도움될 수 있습니다.

dialect: org.hibernate.dialect.H2Dialect

H2 콘솔 사용 설정 (필요 시)

h2:

console:

enabled: true

단위 테스트

❖ Spring Boot 단위 테스트

- ✓ Member 테이블을 위한 Entity 클래스 추가

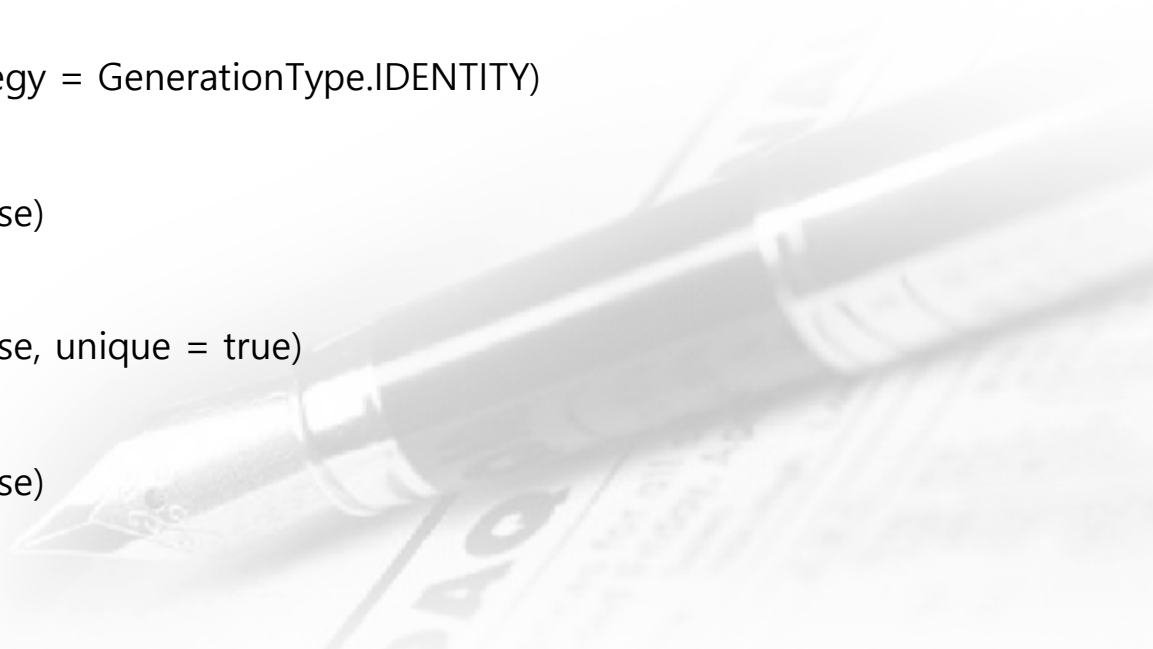
```
import lombok.Builder;
import lombok.Getter;
import lombok.NoArgsConstructor;
import jakarta.persistence.*;

@Getter
@Entity
@NoArgsConstructor
public class Member {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(nullable = false)
    private String name;

    @Column(nullable = false, unique = true)
    private String email;

    @Column(nullable = false)
    private int age;
```



단위 테스트

❖ Spring Boot 단위 테스트

✓ Entity 클래스 추가

```
@Builder
public Member(String name, String email, int age) {
    this.name = name;
    this.email = email;
    this.age = age;
}
}
```



단위 테스트

❖ Spring Boot 단위 테스트

- ✓ 데이터베이스 연동을 위한 MemberRepository 인터페이스 추가

```
import org.springframework.data.jpa.repository.JpaRepository;
```

```
public interface MemberRepository extends JpaRepository<Member, Long> {  
}
```



단위 테스트

❖ Spring Boot 단위 테스트

✓ Calculator 클래스 추가

```
import org.springframework.stereotype.Service;
```

```
@Service
```

```
public class Calculator {
```

```
    public int sum(int a, int b) {
```

```
        return a + b;
```

```
    }
```

```
}
```



단위 테스트

❖ Spring Boot 단위 테스트

✓ MemberService 클래스 추가

```
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Service;

@Service
@RequiredArgsConstructor
public class MemberService {
    private final MemberRepository memberRepository;

    void save() {
        memberRepository.save(
            Member.builder()
                .age(1)
                .email("email@email.com")
                .name("name")
                .build()
        );
    }
}
```



단위 테스트

❖ Spring Boot 단위 테스트

✓ MemberController 클래스 생성

```
import lombok.RequiredArgsConstructor;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
```

```
@RestController
```

```
@RequiredArgsConstructor
```

```
class CalculatorController {
```

```
    private final Calculator calculator;
    private final MemberService memberService;
```

```
    @RequestMapping("/")
    String save() {
        memberService.save();
        return "Success";
    }
```

```
    @RequestMapping("/health")
    String health() {
        return "OK";
    }
```

단위 테스트

❖ Spring Boot 단위 테스트

✓ MemberController 클래스 생성

```
@RequestMapping("/sum")
String sum(@RequestParam("a") Integer a,
           @RequestParam("b") Integer b) {
    return String.valueOf(calculator.sum(a, b));
}
```



단위 테스트

❖ Spring Boot 단위 테스트

- ✓ test 디렉토리에 Repository 테스트용 클래스(MemberRepositoryTest) 추가

```
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.autoconfigure.jdbc.AutoConfigureTestDatabase;
import org.springframework.boot.test.autoconfigure.orm.jpa.DataJpaTest;
import org.springframework.test.context.ActiveProfiles;
import org.springframework.test.context.TestPropertySource;

@ActiveProfiles("test")
@DataJpaTest
@AutoConfigureTestDatabase(replace = AutoConfigureTestDatabase.Replace.ANY)
class MemberRepositoryTest {
    @Autowired
    MemberRepository memberRepository;

    @Test
    void save() {
        memberRepository.save(
            Member.builder()
                .age(1)
                .email("email@email.com")
                .name("name")
                .build()
        );
    }
}
```

단위 테스트

❖ Spring Boot 단위 테스트

- ✓ test 디렉토리에 Calculator 테스트용 클래스(CalculatorTest) 추가

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

```
public class CalculatorTest {  
    private Calculator calculator = new Calculator();
```

```
    @Test  
    public void testSum() {  
        assertEquals(5, calculator.sum(2, 3));  
    }  
}
```



단위 테스트

❖ Spring Boot 단위 테스트

- ✓ test 디렉토리에 Controller 테스트용 클래스(ControllerTest) 추가

```
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.http.MediaType;
import org.springframework.test.web.servlet.MockMvc;
import org.springframework.test.web.servlet.result.MockMvcResultHandlers;
import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.*;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.*;
```

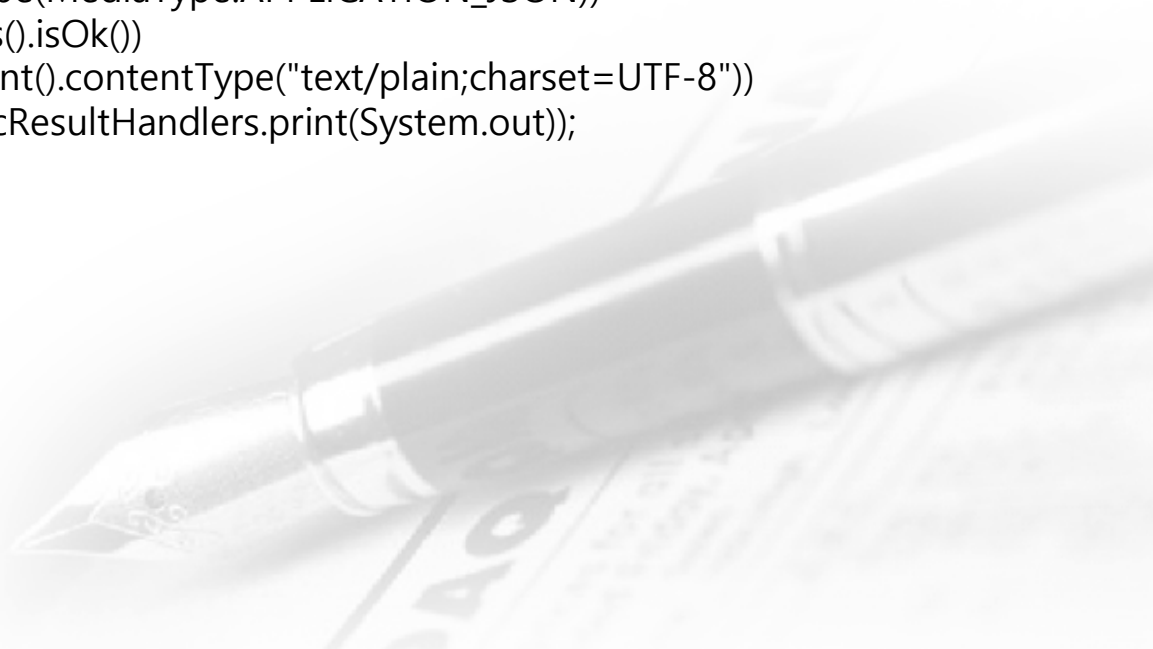


단위 테스트

❖ Spring Boot 단위 테스트

- ✓ test 디렉토리에 Controller 테스트용 클래스 추가

```
@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.MOCK)
@AutoConfigureMockMvc
public class ControllerTest {
    @Autowired
    private MockMvc mockMvc;
    @Test
    public void testGetHotelById() throws Exception {
        mockMvc.perform(post("/health")
            .contentType(MediaType.APPLICATION_JSON))
            .andExpect(status().isOk())
            .andExpect(content().contentType("text/plain;charset=UTF-8"))
            .andDo(MockMvcResultHandlers.print(System.out));
    }
}
```



단위 테스트

- ❖ Spring Boot 단위 테스트
 - ✓ 테스트 명령 수행
 - ./gradlew compileJava
 - ./gradlew test



단위 테스트

- ❖ 현재 프로젝트를 github에 push



단위 테스트

❖ Jenkins Pipeline 프로젝트에서 단위 테스트

✓ 스크립트 추가

```
pipeline {  
  agent any  
  stages {  
    stage('github clone') {  
      steps {  
        git branch: 'main', url: 'https://github.com/itstudy001/jenkinstest.git'  
      }  
    }  
  }  
}
```



단위 테스트

❖ Jenkins Pipeline 프로젝트에서 단위 테스트

✓ 스크립트 추가

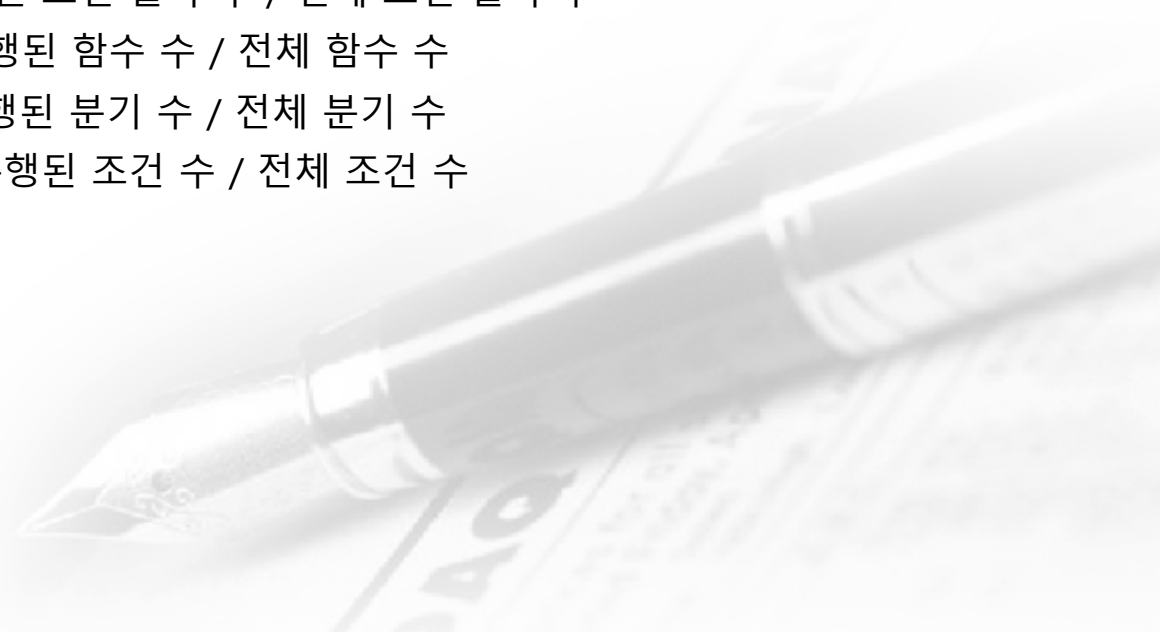
```
stage("permission"){
    steps{
        sh "chmod +x ./gradlew"
    }
}
stage("compile"){
    steps{
        sh "./gradlew compileJava"
    }
}
stage("test"){
    steps{
        sh "./gradlew test"
    }
}
}
```



Code Coverage

❖ 개요

- ✓ 코드 커버리지는 소프트웨어 테스트에서 사용되는 지표 중 하나로 코드의 실행 여부를 기반으로 코드의 품질과 완성도를 측정하는 방법
- ✓ 테스트 케이스를 실행할 때 코드 내의 각 요소가 실행되는 빈도를 측정하여 테스트 케이스가 커버하는 코드 비율을 나타내는 지표
- ✓ 코드 커버리지는 테스트 케이스의 실행 여부를 기준으로 계산하는데 이를 통해 테스트 케이스가 커버하지 않은 코드 블록을 식별하여 개발자가 이를 보완할 수 있도록 도와줌
- ✓ 코드 커버리지는 일반적인 측정 요소
 - Statement Coverage: 실행된 문장 수 / 전체 문장 수
 - Branch Coverage: 실행된 조건 블록 수 / 전체 조건 블록 수
 - Function Coverage: 실행된 함수 수 / 전체 함수 수
 - Decision Coverage: 수행된 분기 수 / 전체 분기 수
 - Condition Coverage: 수행된 조건 수 / 전체 조건 수



Code Coverage

❖ JaCoCo Code Coverage

✓ 설정 및 수행

- build.gradle 파일에 플러그인 추가

```
plugins {
```

```
    id 'org.springframework.boot' version '2.6.1'
```

```
    id 'io.spring.dependency-management' version '1.0.11.RELEASE'
```

```
    id 'java'
```

```
    id 'jacoco'
```

```
}
```



Code Coverage

❖ JaCoCo Code Coverage

✓ 설정 및 수행

- build.gradle 파일에 설정 추가
- ```
jacocoTestCoverageVerification{
```

```
 violationRules{
```

```
 rule{
```

```
 limit{
```

minimum = 0.2 //전체 테스트 커버리지가

20% 이상이어야 함

```
 }
```

```
 }
```

```
}
```

```
}
```



# Code Coverage

## ❖ JaCoCo Code Coverage

### ✓ 설정 및 수행

#### ● 실행

`./gradlew test jacocoTestCoverageVerification`

`./gradlew test jacocoTestReport`



# Code Coverage

## ❖ JaCoCo Code Coverage

### ✓ 설정 및 수행

#### ● 결과 확인

프로젝트의 `build/reports/jacoco/test/html/index.html`

## calculator

| Element                                                                                                                 | Missed Instructions                                                               | Cov. | Missed Branches | Cov. | Missed | Cxty | Missed | Lines | Missed | Methods | Missed | Classes |
|-------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|------|-----------------|------|--------|------|--------|-------|--------|---------|--------|---------|
|  <a href="#">com.leszko.calculator</a> |  | 57%  |                 | n/a  | 9      | 23   | 16     | 29    | 9      | 23      | 0      | 6       |
| Total                                                                                                                   | 53 of 126                                                                         | 57%  | 0 of 0          | n/a  | 9      | 23   | 16     | 29    | 9      | 23      | 0      | 6       |

# Code Coverage

## ❖ JaCoCo Code Coverage

### ✓ 결과 해석

#### ● Class Coverage

- ◆ 클래스가 1번 이상 사용되었는지를 기준으로 커버리지를 계산
- ◆ 클래스 사용 기준은 클래스 내 메소드 중 1개 이상 사용, 생성자, Static initializer 호출 시 클래스가 사용된 것으로 판단

#### ● Method Coverage

- ◆ 메소드가 최소 1번 이상 호출 되었는지를 기준으로 커버리지를 계산
- ◆ 메소드 내부의 모든 코드가 실행 되었는지는 판단 기준에서 제외
- ◆  $\text{Method Coverage(\%)} = (\text{실행된 메소드의 수} / \text{전체 메소드의 수}) \times 100$

#### ● Line Coverage

- ◆ 테스트 중에 실행된 소스 코드 라인 수와 전체 소스 코드 라인 수 중 몇 개의 라인이 실행 되었는지를 측정하며 이를 통해 테스트에서 놓친 코드 라인이 얼마나 있는지 확인할 수 있음
- ◆  $\text{Line Coverage(\%)} = (\text{실행된 코드 라인 수} / \text{전체 소스 코드 라인 수}) \times 100$

# Code Coverage

## ❖ JaCoCo Code Coverage

### ✓ 결과 해석

#### ● Branch Coverage

- ◆ 소스 코드 내 분기문(if, Switch 등)에 대해 테스트의 실행 흐름이 어떻게 되는지 측정하는 것
- ◆ 특정한 조건에 따라 코드 실행 흐름이 달라질 때 모든 분기문이 정상적으로 실행 되는지 확인할 수 있음
- ◆ 브랜치 커버리지(%) = (실행된 분기 수 / 전체 분기 수) x 100

#### ● 라인 커버리지 vs 브랜치 커버리지

- ◆ 브랜치 커버리지가 100%인 경우 모든 코드 라인이 수행되므로 라인 커버리지도 100%
- ◆ 라인 커버리지가 100%라고 해서 브랜치 커버리지가 100%가 되지는 않음

#### ● Instructions Coverage

- ◆ JaCoCo에서 제공하는 가장 작은 단위의 커버리지 측정 방법
- ◆ 자바 바이트 코드 단위에서 실행되었거나 실행되지 않은 코드의 양에 대한 정보를 측정
- ◆ 작성한 자바 코드의 바이트 코드는 여러가지 방법으로 확인이 가능하며 IntelliJ IDE에서는 jclasslib 이라는 plugin을 통해 쉽게 확인이 가능

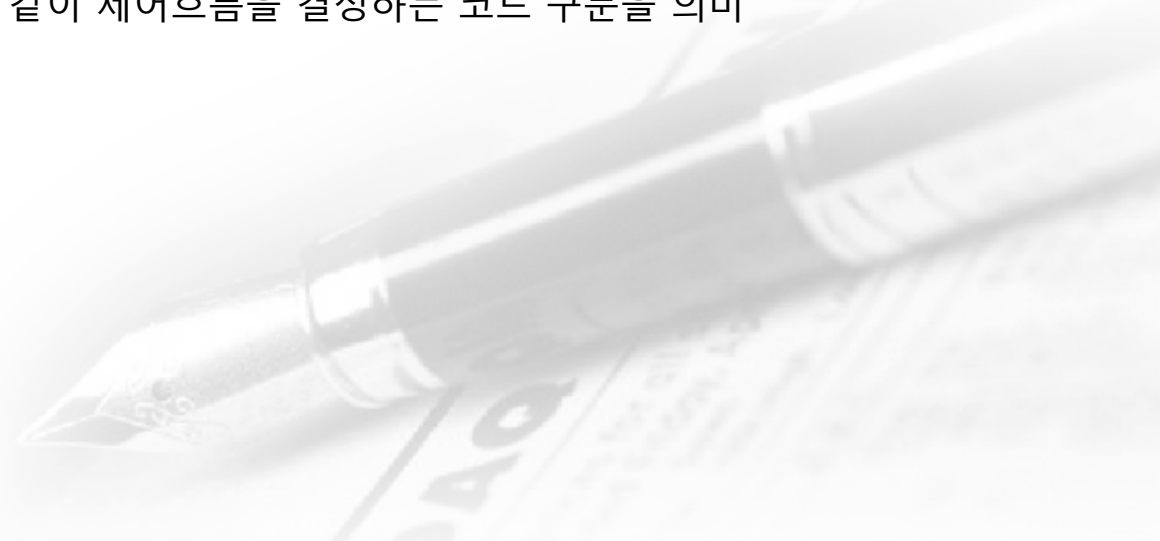
# Code Coverage

## ❖ JaCoCo Code Coverage

### ✓ 결과 해석

#### ● Complexity by Type

- ◆ JaCoCo 리포트에 Cxty(Complexity by Type) 필드가 있는데 Cxty는 코드의 복잡도를 나타내는 매트릭
- ◆ 클래스나 인터페이스와 같은 코드 유형의 복잡도를 측정하는 데 사용하며 순환 복잡도(Cyclomatic Complexity)를 아래 공식으로 계산
- ◆  $v(G) = B(\text{the number of branches}) - D(\text{the number of decision points}) + 1$ 
  - B(the number of branches): the number of possible paths through the code.
  - D(the number of decision points): 조건문(if, switch) 나 루프문 (for, while, do-while)과 같이 제어흐름을 결정하는 코드 구문을 의미





# Code Coverage

## ❖ Code Coverage의 이점

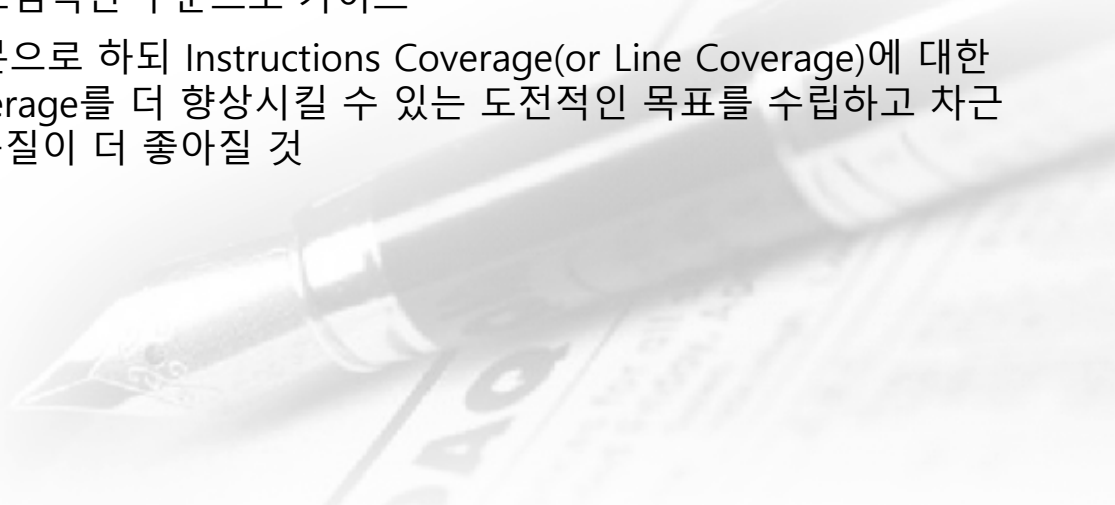
- ✓ 버그 감소: 테스트 코드가 더 많은 부분을 실행하므로 버그를 사전에 검출하는데 도움이 되는데 코드 커버리지가 낮으면 테스트 되지 않은 코드에서 버그가 발생할 수 있음
- ✓ 코드 품질 향상: 더 많은 코드가 테스트되고 버그가 감소하게 되므로 코드의 안정성과 신뢰성이 높아짐
- ✓ 유지보수 용이성: 코드 변경 사항에 대한 테스트 코드를 작성하는 것이 더 쉬워짐
- ✓ 개발 생산성 향상: 개발자가 버그를 수정하고 새로운 기능을 추가하는데 소요되는 시간이 줄어듦



# Code Coverage

## ❖ Code Coverage의 목표

- ✓ 코드 커버리지 100% 가 무결점 소프트웨어를 의미하진 않는다는 것을 기억해야 하는데 코드 커버리지가 100%가 되었다고 하더라도 아래의 경우에서 문제점이 발생할 수 있기 때문
  - 요구사항에 있는 기능 누락된 경우
  - 구현에 오류가 있는 경우
  - 테스트 케이스가 누락된 경우
- ✓ 단순히 코드 커버리지 숫자를 높이기 위해 테스트 케이스를 생성하는 일은 지양해야 하며 요구 사항 및 구현 코드에 대한 이해를 바탕으로 올바른 테스트 케이스를 작성해야 함
- ✓ Google Testing Blog에 작성된 Code Coverage Best Practices 글을 참고 해보면 코드의 커버리지는 코드의 비즈니스 영향/중요도, 코드를 얼마나 자주 건드려야 하는지, 코드의 복잡성, 유지 기간 및 도메인 변수에 따라 도메인 오너가 수준을 정하는 것이 좋다고 가이드 하고 있으며 또한 구글에서 제시하고 있는 코드 커버리지 수준은 60%를 적절한 수준, 75%를 칭찬받을 만한 수준, 90%를 모범적인 수준으로 가이드
- ✓ 메소드 커버리지 100%는 기본으로 하되 Instructions Coverage(or Line Coverage)에 대한 지표를 도입하고 Branch Coverage를 더 향상시킬 수 있는 도전적인 목표를 수립하고 차근 차근 진행해 나가면 코드의 품질이 더 좋아질 것



# SonarQube

## ❖ 정적 코드 분석

### ✓ 개요

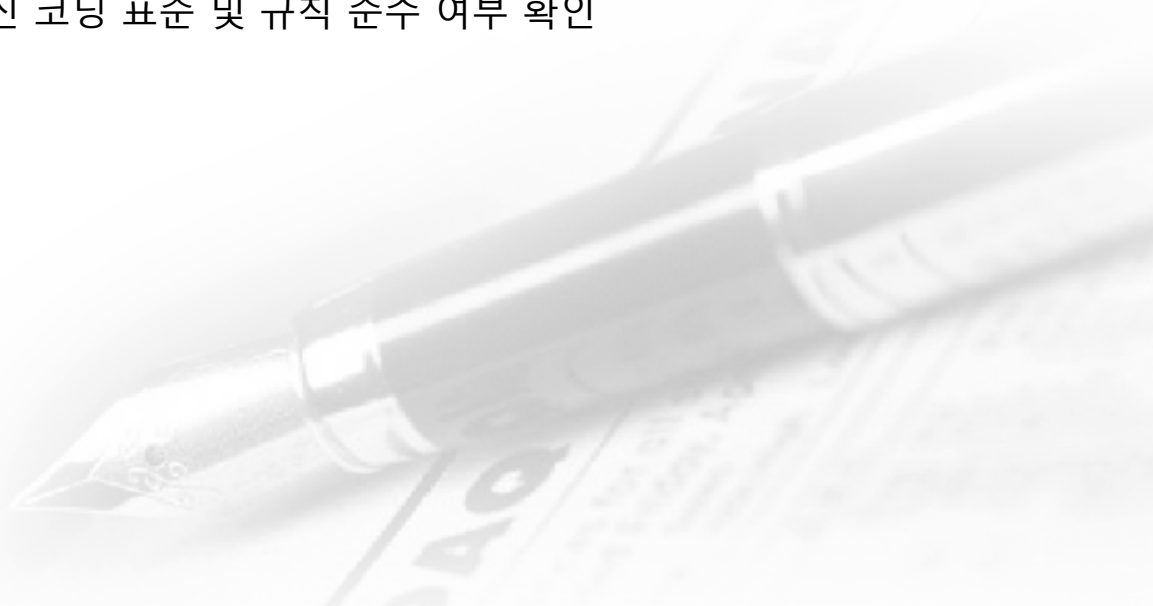
- 소스 코드를 실행하지 않고 코드의 품질을 검사하는 과정
- 코드를 읽기 쉽고 유지보수하기 좋도록 만드는 것을 보증
- checkstyle 이나 findbug, PMD 등을 이용

### ✓ 목적

- 코드 품질 개선: 복잡한 코드, 중복 코드, 나쁜 코드 스타일을 검출
- 버그 사전 탐지: 런타임 오류를 초래할 수 있는 잠재적 문제 탐지
- 보안 강화: SQL 인젝션, XSS, 취약한 암호화 알고리즘과 같은 보안 취약점 검출
- 코드 일관성 유지: 정해진 코딩 표준 및 규칙 준수 여부 확인

### ✓ 정적 코드 분석 도구

- SonarQube
- Checkstyle
- SpotBugs
- PMD
- FindSecBugs



# SonarQube

## ❖ SonarQube

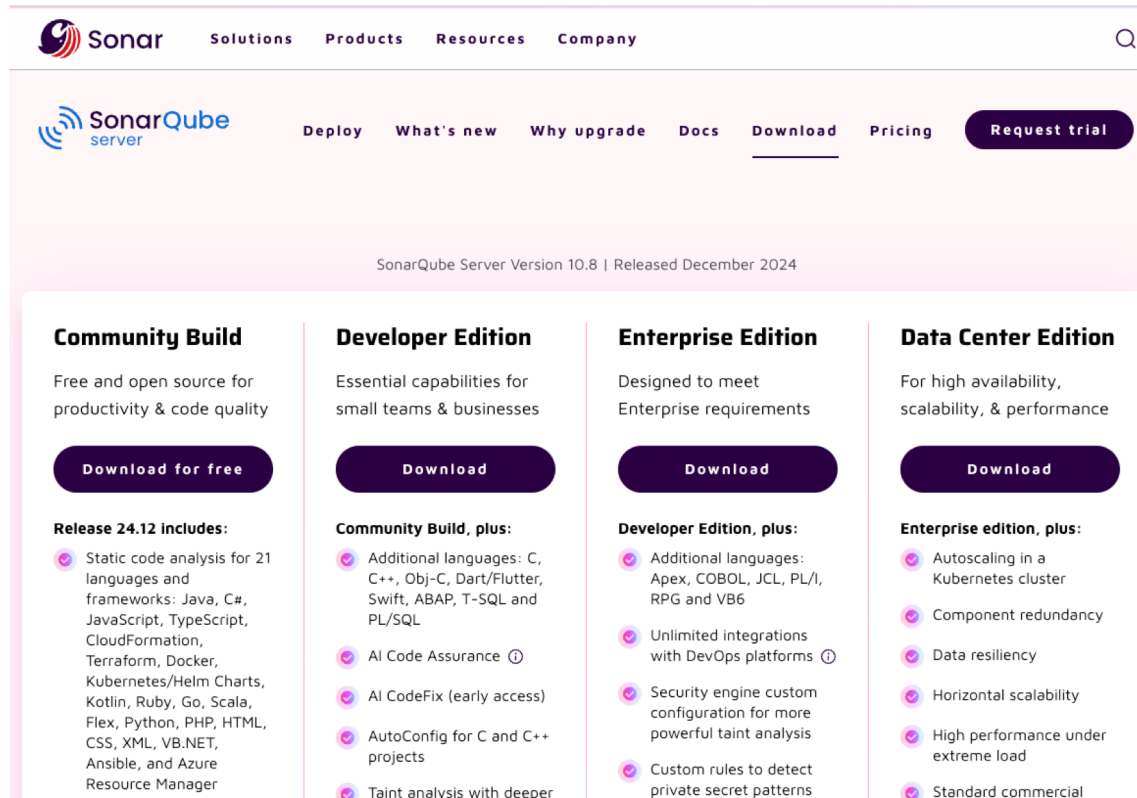
- ✓ <https://www.sonarqube.org/>
- ✓ Continuous Integration + Analysis
  - Code Quality Assurance tool -> Issues + Defect + Code Complexity
  - Detect Bugs & Vulnerabilities
  - Track Code Smells
  - Code Quality Metrics & History
- ✓ 17 languages 지원: Java, C#, JavaScript, TypeScript, CloudFormation, Terraform, Kotlin, Ruby, Go, Scala, Flex, Python, PHP, HTML, CSS, XML, VB.NET 등
- ✓ CI/CD integration
- ✓ Extensible, with 50+ community plugins



# SonarQube

## ❖ 설치

✓ <https://www.sonarqube.org/downloads/>



The screenshot shows the SonarQube website's download page. The top navigation bar includes 'Solutions', 'Products', 'Resources', and 'Company'. Below this, the 'SonarQube server' logo is on the left, and a secondary navigation bar contains 'Deploy', 'What's new', 'Why upgrade', 'Docs', 'Download' (which is underlined), 'Pricing', and a 'Request trial' button. A banner below the navigation bar states 'SonarQube Server Version 10.8 | Released December 2024'. The main content area is divided into four columns, each representing a different edition of SonarQube: Community Build, Developer Edition, Enterprise Edition, and Data Center Edition. Each column includes a description, a 'Download' button, and a list of features or capabilities.

| Community Build                                                                                                                                                                                                                                                                                                                             | Developer Edition                                                                                                                                                                                                                                                                                                   | Enterprise Edition                                                                                                                                                                                                                                                                                                                            | Data Center Edition                                                                                                                                                                                                                                                                          |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Free and open source for productivity & code quality                                                                                                                                                                                                                                                                                        | Essential capabilities for small teams & businesses                                                                                                                                                                                                                                                                 | Designed to meet Enterprise requirements                                                                                                                                                                                                                                                                                                      | For high availability, scalability, & performance                                                                                                                                                                                                                                            |
| <a href="#">Download for free</a>                                                                                                                                                                                                                                                                                                           | <a href="#">Download</a>                                                                                                                                                                                                                                                                                            | <a href="#">Download</a>                                                                                                                                                                                                                                                                                                                      | <a href="#">Download</a>                                                                                                                                                                                                                                                                     |
| <b>Release 24.12 includes:</b> <ul style="list-style-type: none"><li>✓ Static code analysis for 21 languages and frameworks: Java, C#, JavaScript, TypeScript, CloudFormation, Terraform, Docker, Kubernetes/Helm Charts, Kotlin, Ruby, Go, Scala, Flex, Python, PHP, HTML, CSS, XML, VB.NET, Ansible, and Azure Resource Manager</li></ul> | <b>Community Build, plus:</b> <ul style="list-style-type: none"><li>✓ Additional languages: C, C++, Obj-C, Dart/Flutter, Swift, ABAP, T-SQL and PL/SQL</li><li>✓ AI Code Assurance ⓘ</li><li>✓ AI CodeFix (early access)</li><li>✓ AutoConfig for C and C++ projects</li><li>✓ Taint analysis with deeper</li></ul> | <b>Developer Edition, plus:</b> <ul style="list-style-type: none"><li>✓ Additional languages: Apex, COBOL, JCL, PL/I, RPG and VB6</li><li>✓ Unlimited integrations with DevOps platforms ⓘ</li><li>✓ Security engine custom configuration for more powerful taint analysis</li><li>✓ Custom rules to detect private secret patterns</li></ul> | <b>Enterprise edition, plus:</b> <ul style="list-style-type: none"><li>✓ Autoscaling in a Kubernetes cluster</li><li>✓ Component redundancy</li><li>✓ Data resiliency</li><li>✓ Horizontal scalability</li><li>✓ High performance under extreme load</li><li>✓ Standard commercial</li></ul> |

# SonarQube

## ❖ 설치

### ✓ 도커를 이용한 설치

- Apple Silicon(M1) Chip이 아닌 경우

```
docker run --rm -p 9000:9000 --name sonarqube sonarqube
```

- Apple Silicon(M1) Chip의 경우

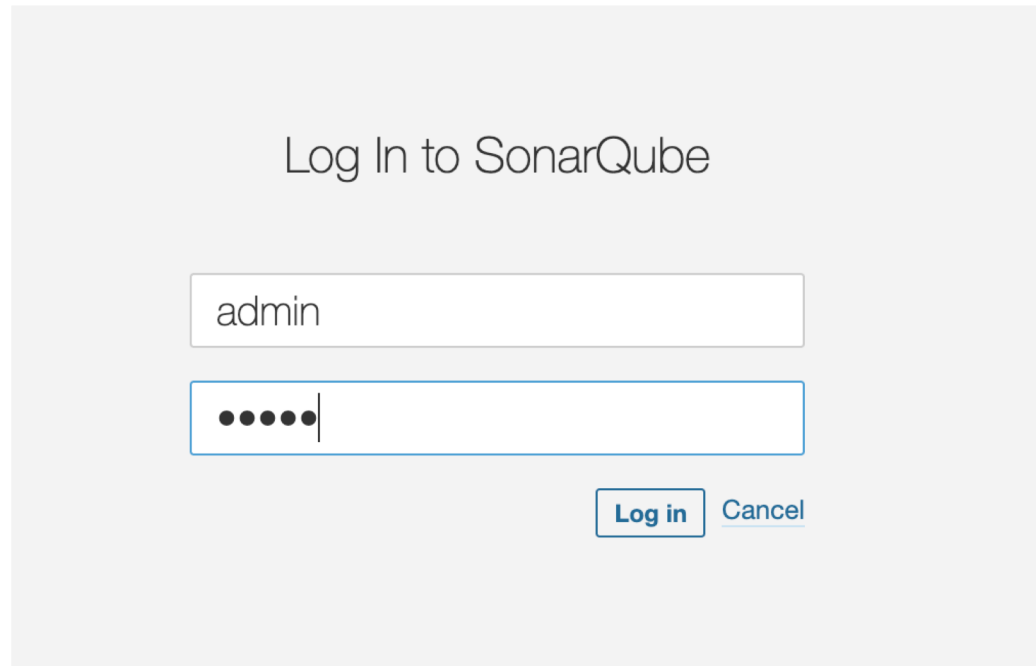
```
docker run --rm -p 9000:9000 --name sonarqube sonarqube:its-community
```



# SonarQube

## ❖ 설치

- ✓ localhost:9000 으로 접속
  - 계정은 admin 이고 비밀번호도 admin

A screenshot of the SonarQube login interface. It features a light gray background with the text "Log In to SonarQube" centered at the top. Below the text are two input fields: the first contains the username "admin", and the second contains five dots representing a password. At the bottom right of the dialog are two buttons: "Log in" and "Cancel".

Log In to SonarQube

admin

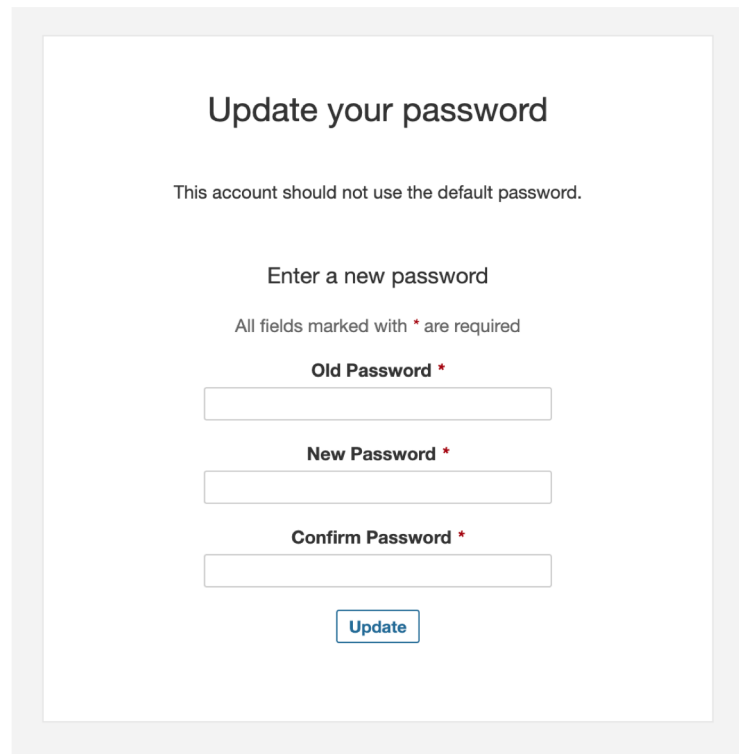
.....

Log in Cancel

# SonarQube

## ❖ 설치

- ✓ localhost:9000 으로 접속



The image shows a SonarQube web interface for updating a password. The form is titled "Update your password" and includes a message: "This account should not use the default password." Below this, it says "Enter a new password" and "All fields marked with \* are required". There are three input fields: "Old Password \*", "New Password \*", and "Confirm Password \*". Each field has a corresponding text label above it. At the bottom of the form is an "Update" button.

Update your password

This account should not use the default password.

Enter a new password

All fields marked with \* are required

Old Password \*

New Password \*

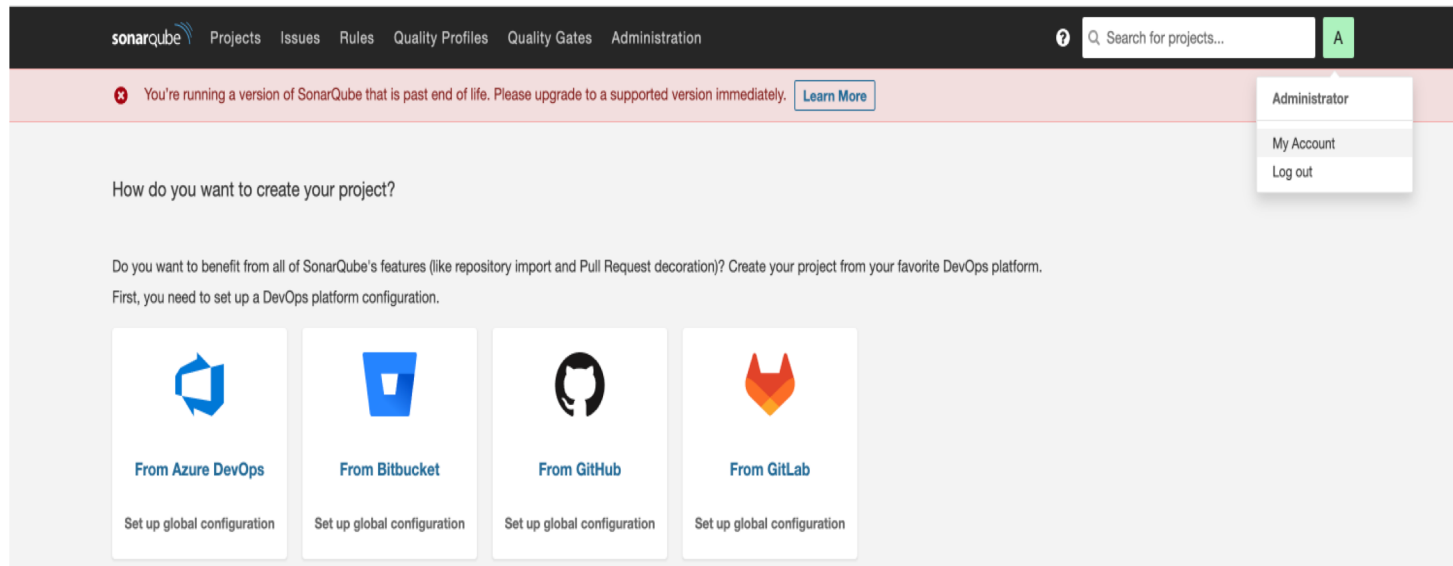
Confirm Password \*

Update



# SonarQube

## ❖ SonarQube Token 발행



sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

Q Search for projects... A

❗ You're running a version of SonarQube that is past end of life. Please upgrade to a supported version immediately. [Learn More](#)

Administrator  
My Account  
Log out

How do you want to create your project?

Do you want to benefit from all of SonarQube's features (like repository import and Pull Request decoration)? Create your project from your favorite DevOps platform.  
First, you need to set up a DevOps platform configuration.



From Azure DevOps

[Set up global configuration](#)



From Bitbucket

[Set up global configuration](#)



From GitHub

[Set up global configuration](#)



From GitLab

[Set up global configuration](#)

# SonarQube

## ❖ SonarQube Token 발행

squ\_3e12a0eb662e0bf3b8c5ad9e8cd236a38c7c3f67

A

Administrator

Profile

Security

Notifications

Projects

Tokens

If you want to enforce security by not providing credentials of a real SonarQube user to run your code scan or to invoke web services, you can provide a User Token as a replacement of the user login. This will increase the security of your installation by not letting your analysis user's password going through your network.

sonar-token

User Token

Generate

| Name      | Type | Project | Last use | Created |
|-----------|------|---------|----------|---------|
| No tokens |      |         |          |         |

# SonarQube

## ❖ Gradle 프로젝트에 적용

- ✓ build.gradle 파일에 플러그인 설치

```
plugins {
 id 'org.springframework.boot' version '2.6.1'
 id 'io.spring.dependency-management' version '1.0.11.RELEASE'
 id 'java'
 id("org.sonarqube") version "7.2.3.7755"
}
```



# SonarQube

- ❖ Gradle 프로젝트에 적용
    - ✓ 검사 수행
- `./gradlew clean build`

`./gradlew sonar -Dsonar.host.url=http://localhost:9000 -  
Dsonar.login=squ_b5ab8f5f235afedf6896b48280ecaea9aa075e6c`



# SonarQube

- ❖ Gradle 프로젝트에 적용
  - ✓ 결과 확인

Search by project name or key

Create Project

1 projects

Perspective: Overall Status

Sort by: Name

☆ calculator Passed

Last analysis: 4 minutes ago

| Bugs | Vulnerabilities | Hotspots Reviewed | Code Smells | Coverage                                | Duplications                              | Lines      |
|------|-----------------|-------------------|-------------|-----------------------------------------|-------------------------------------------|------------|
| 0 A  | 0 A             | 0.0% E            | 1 A         | 0.0% <span style="color: red;">○</span> | 0.0% <span style="color: green;">○</span> | 32 XS Java |

1 of 1 shown

# SonarQube

## ❖ Bad Code

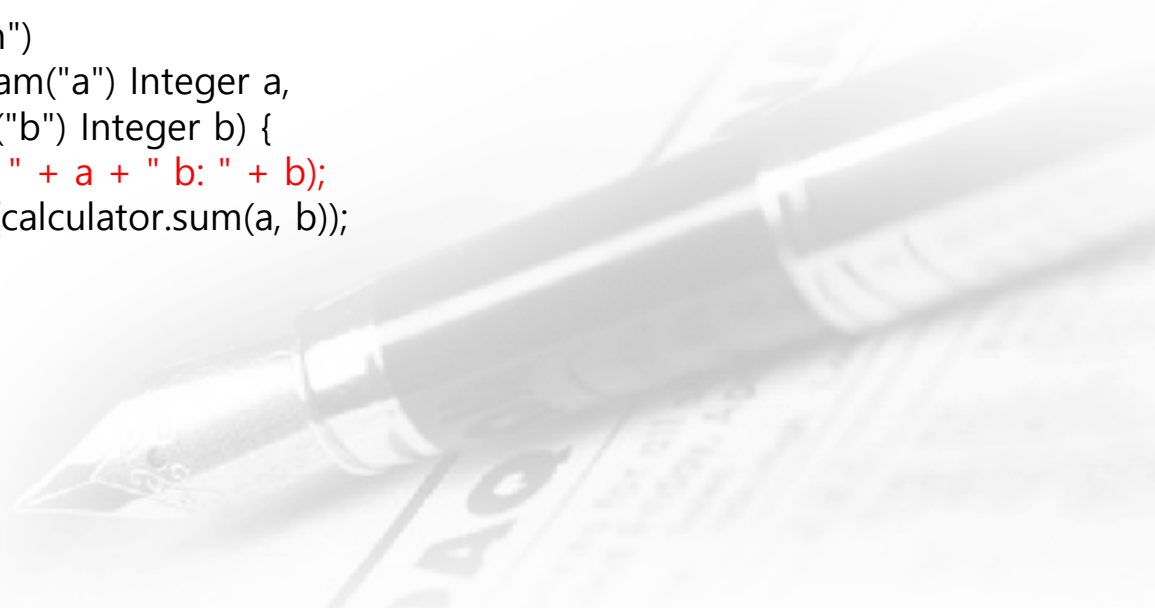
### ✓ Controller Code 수정

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

```
@RestController
class MemberController {
 @Autowired
 private Calculator calculator;
```

```
 private final Logger logger = LoggerFactory.getLogger(MemberController.class);
```

```
 @RequestMapping("/sum")
 String sum(@RequestParam("a") Integer a,
 @RequestParam("b") Integer b) {
 System.out.println("a: " + a + " b: " + b);
 return String.valueOf(calculator.sum(a, b));
 }
}
```



# SonarQube

## ❖ Bad Code

### ✓ 검사 수행

`./gradlew clean build`

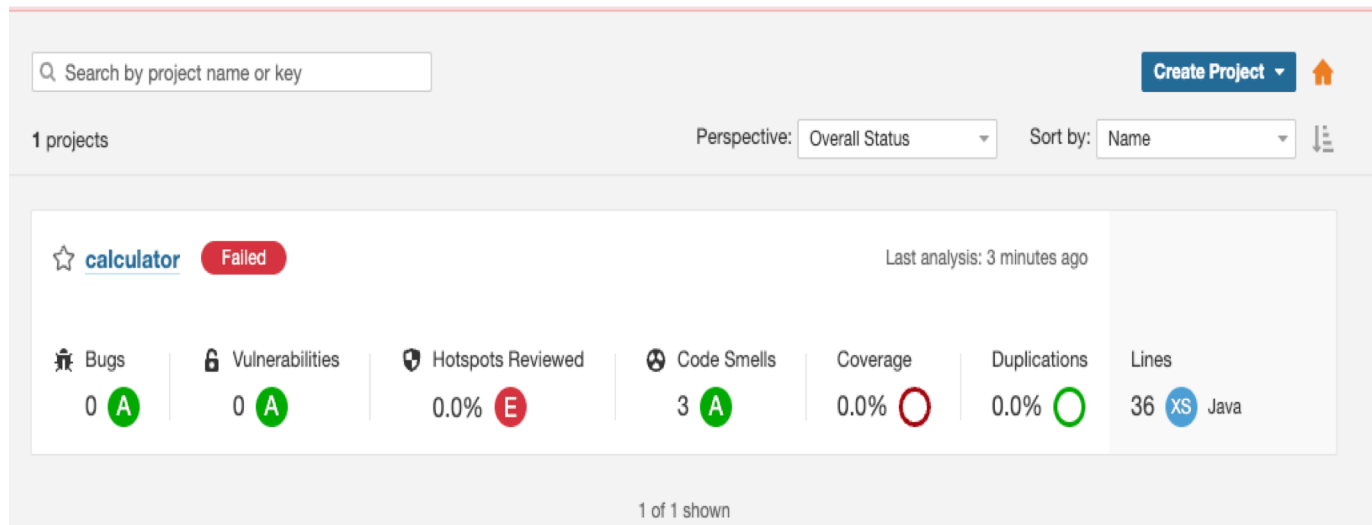
`./gradlew sonar -Dsonar.host.url=http://localhost:9000 -  
Dsonar.login=squ_b5ab8f5f235afedf6896b48280ecaea9aa075e6c`



# SonarQube

## ❖ Bad Code

✓ 결과 확인

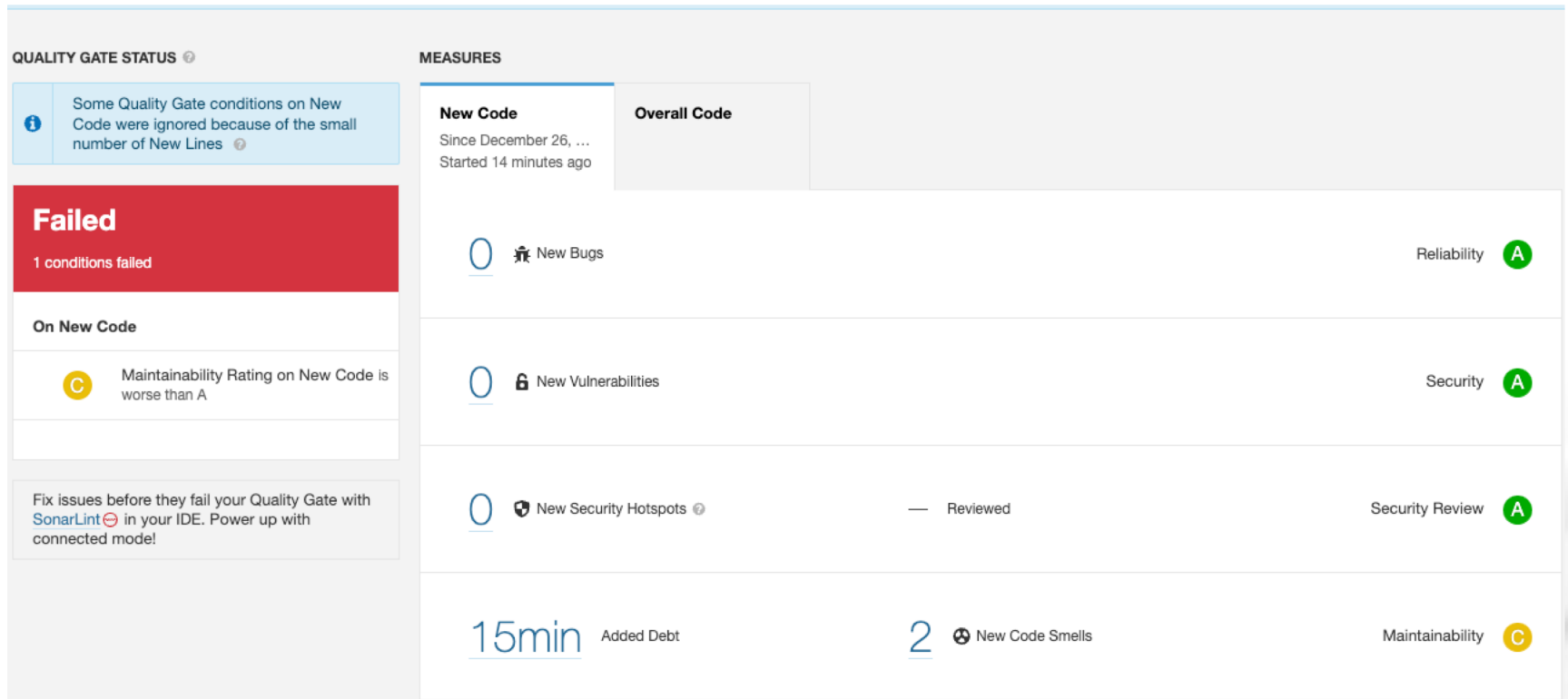




# SonarQube

## ❖ Bad Code

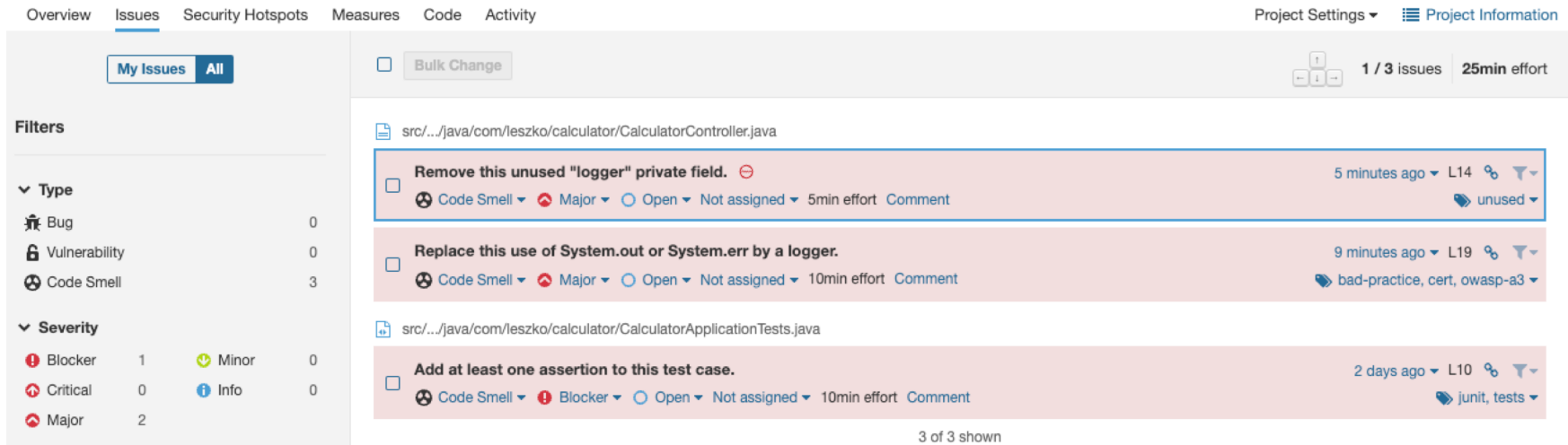
✓ 결과 확인



# SonarQube

## ❖ Bad Code

✓ 결과 확인



The screenshot displays the SonarQube interface for the 'Issues' tab. The top navigation bar includes 'Overview', 'Issues' (selected), 'Security Hotspots', 'Measures', 'Code', and 'Activity'. On the right, there are links for 'Project Settings' and 'Project Information'. Below the navigation bar, a summary bar shows '1 / 3 issues' and '25min effort'. The left sidebar contains filters for 'Type' (Bug, Vulnerability, Code Smell) and 'Severity' (Blocker, Critical, Major, Minor, Info). The main content area lists three issues:

- Remove this unused "logger" private field.** (Code Smell, Major, Open, Not assigned, 5min effort, Comment). Location: `src/.../java/com/ieszko/calculator/CalculatorController.java`. Status: 5 minutes ago, L14, unused.
- Replace this use of System.out or System.err by a logger.** (Code Smell, Major, Open, Not assigned, 10min effort, Comment). Location: `src/.../java/com/ieszko/calculator/CalculatorController.java`. Status: 9 minutes ago, L19, bad-practice, cert, owasp-a3.
- Add at least one assertion to this test case.** (Code Smell, Blocker, Open, Not assigned, 10min effort, Comment). Location: `src/.../java/com/ieszko/calculator/CalculatorApplicationTests.java`. Status: 2 days ago, L10, junit, tests.

3 of 3 shown

# SonarQube

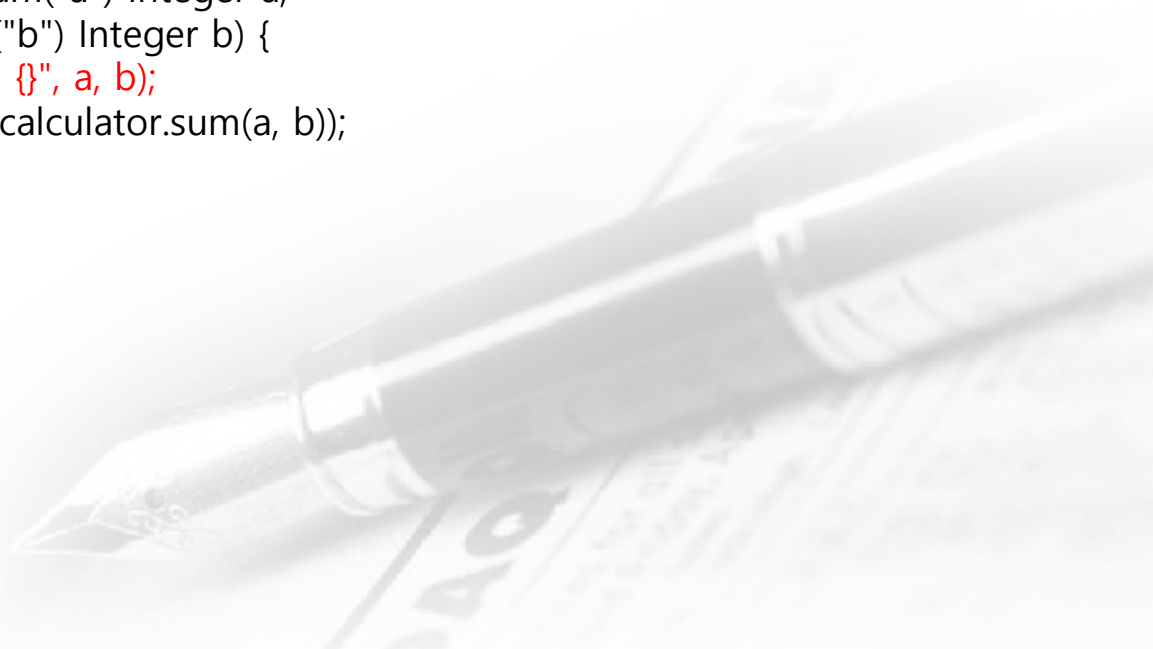
## ❖ Bad Code

### ✓ Controller Code 수정

```
@RestController
class CalculatorController {
 @Autowired
 private Calculator calculator;

 private final Logger logger = LoggerFactory.getLogger(CalculatorController.class);

 @RequestMapping("/sum")
 String sum(@RequestParam("a") Integer a,
 @RequestParam("b") Integer b) {
 logger.debug("a: {}, b: {}", a, b);
 return String.valueOf(calculator.sum(a, b));
 }
}
```



# SonarQube

## ❖ Bad Code

### ✓ 검사 수행

`./gradlew clean build`

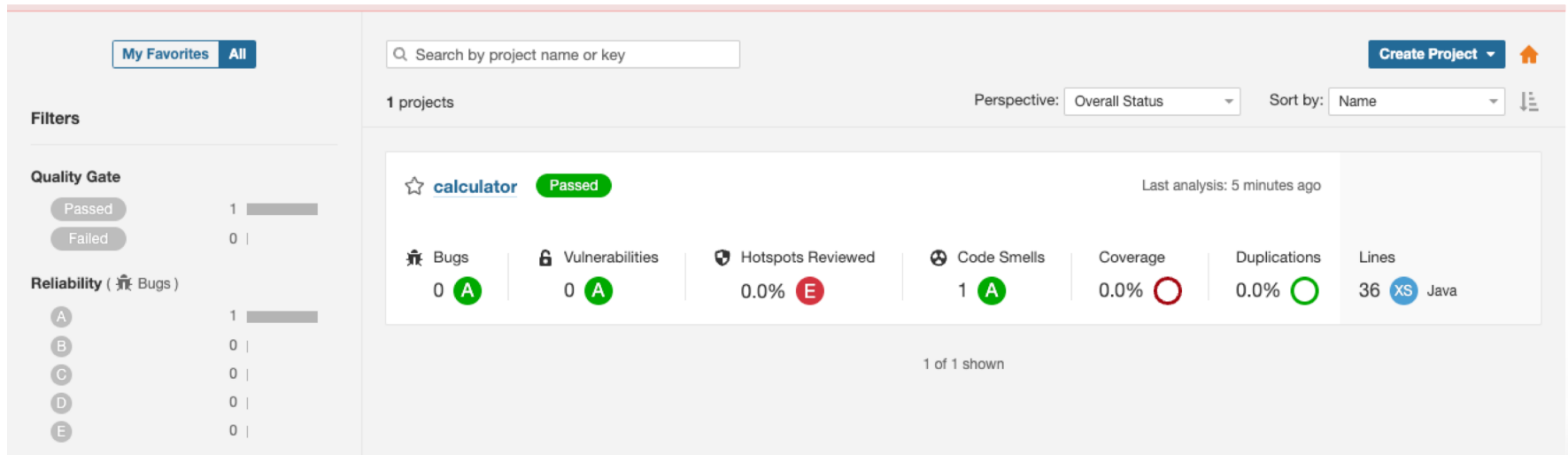
`./gradlew sonarqube -Dsonar.host.url=http://localhost:9000 -  
Dsonar.login=squ_6a9c9f99fbe267f395a7b3533b32d82a04536210`



# SonarQube

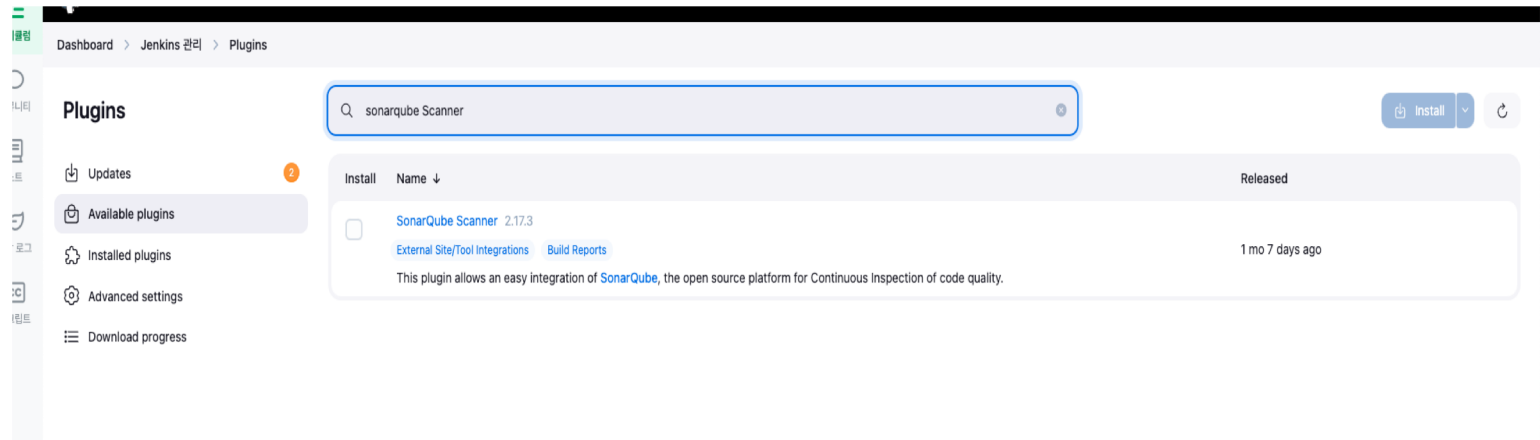
## ❖ Bad Code

✓ 결과 확인



# SonarQube

- ❖ Jenkins 와 연결
  - ✓ 플러그인 설치



# SonarQube

## ❖ Jenkins 와 연결

- ✓ Sonarqube Credential 생성

### New credentials

Kind

Secret text

Scope ?

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

Secret

.....

ID ?

sonarqube-token

Description ?

Create



# SonarQube

## ❖ Jenkins 와 연결

### ✓ Sonarqube Server 설정

- 도커 네트워크 정보 확인: `docker network inspect bridge`

```
"Containers": {
 "10de896a537b80c551466d6fd11c084c29c70a251bee46064a576d8ac6b96416":
 {
 "Name": "sonarqube",
 "EndpointID":
 "4d230fa351494f51d77d88569d39c96c1179848ecba764c9f91be803d6045376",
 "MacAddress": "02:42:ac:11:00:03",
 "IPv4Address": "172.17.0.3/16",
 "IPv6Address": ""
 },
 "89b2614399fd6b017a8c6d2190487b1298e032f2ceed52c55a6d1cdb9db1887d": {
 "Name": "jenkins-server",
 "EndpointID":
 "dfbf57d5392a4c3237c289c67facbba89c353f1ffc791f5ba52fb5d0f2e62dd3",
 "MacAddress": "02:42:ac:11:00:02",
 "IPv4Address": "172.17.0.2/16",
 "IPv6Address": ""
 }
},
```



# SonarQube

## ❖ Jenkins 와 연결

### ✓ Sonarqube Server 설정

#### ● 설정

#### SonarQube servers

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

☒ Environment variables

#### SonarQube installations

[List of SonarQube installations](#)

Name

SonarQube-server

Server URL

Default is <http://localhost:9000>

<http://172.17.0.3:9000>

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled.

sonarqube-token

+ Add

고급 ▾

# SonarQube

## ❖ Jenkins 와 연결

### ✓ 스크립트 수정

```
pipeline {
 agent any
 environment {
 SONAR_HOST_URL = 'http://172.16.0.3:9000'
 SONAR_LOGIN = credentials('Sonarqube')
 }

 stages {
 stage('github clone') {
 steps {
 git branch: 'main', url: 'https://github.com/itstudy001/jenkinstest.git'
 }
 }
 stage("permission"){
 steps{
 sh "chmod +x ./gradlew"
 }
 }
 }
}
```



# SonarQube

## ❖ Jenkins 와 연결

### ✓ 스크립트 수정

```
stage("compile"){
 steps{
 sh "./gradlew compileJava"
 }
}
stage("test"){
 steps{
 sh "./gradlew test"
 }
}

stage("Sonarqube"){
 steps{
 withSonarQubeEnv('Sonarqube-Server') {
 sh " ./gradlew sonar -Dsonar.host.url=$SONAR_HOST_URL -
Dsonar.login=$SONAR_LOGIN"
 }
 }
}
}
```

