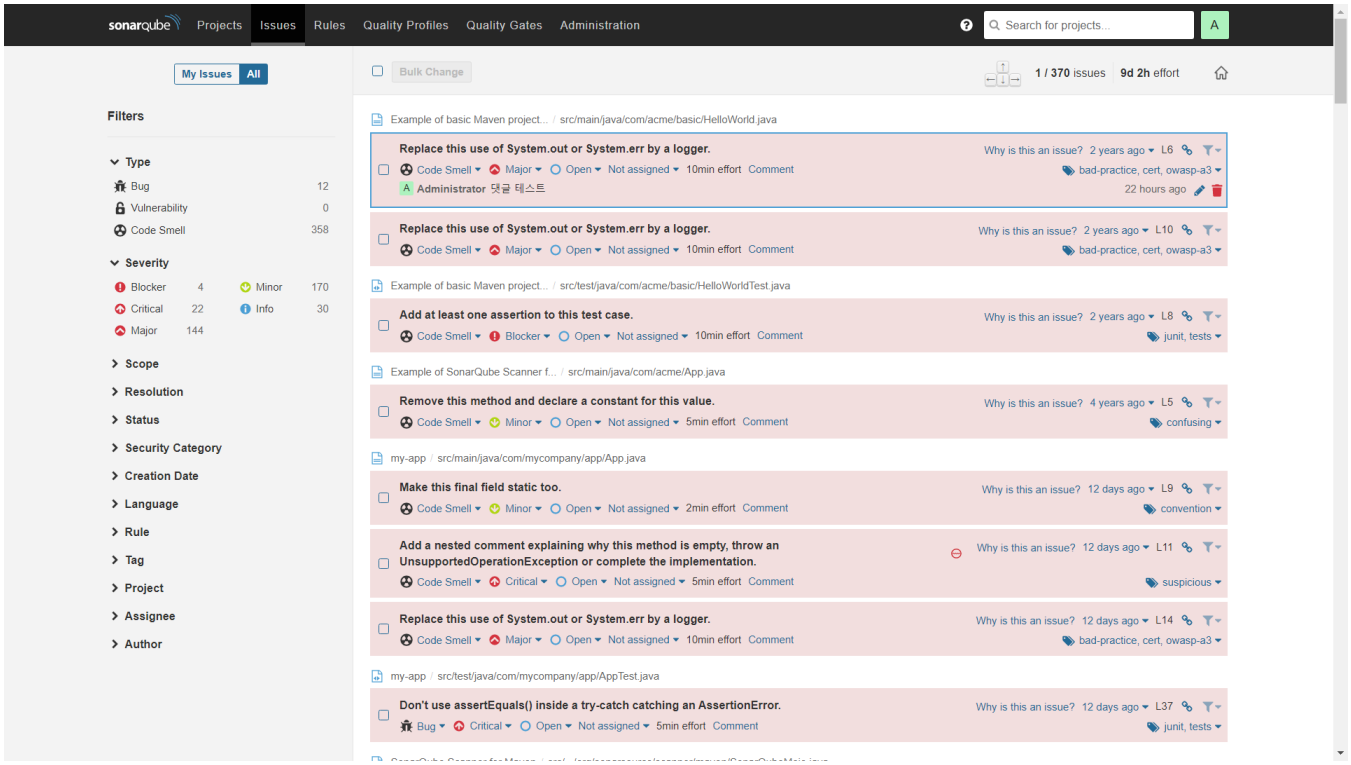
SonarQube Issues

이 문서는 SonarQube Issues 에 대한 정보를 공유하기 위해 작성되었다.



이슈 종류(Issue Types)

이슈에는 다음 세가지 유형이 있다.

1. **Bug** : 런타임에 오류 또는 예기치 않은 동작으로 이어질 수 있는 코딩 실수.
2. **Vulnerability** : 코드에서 공격에 노출된 지점.
3. **Code Smell** : 코드를 혼란스럽고 유지 관리하기 어렵게 만드는 유지 관리 가능성 이슈(Maintainability issue)

이슈 심각도(Issue Severity)

이슈는 다음 다섯가지 심각도 중 하나를 가진다.

1. **BLOCKER**
   프로덕션 환경에서 애플리케이션의 동작에 영향을 줄 가능성이 높은 버그 → **즉시 수정되어야 하는 코드**
   ex) 메모리 누수(memory leak), JDBC 연결이 닫히지 않음(unclosed JDBC connection) 등
2. **CRITICAL**
   프로덕션 환경에서 애플리케이션의 동작에 영향을 줄 가능성이 낮은 버그 또는 보안 결함을 나타내는 이슈 → **즉시 검토되어야 하는 코드**
   ex) 빈 캐치 블록(empty catch block), SQL 삽입(SQL injection) 등
3. **MAJOR**
   개발자 생산성에 큰 영향을 줄 수 있는 품질 결함
   ex) 코드 조각(uncovered piece of code), 중복 블록(duplicated blocks), 사용하지 않는 매개변수(unused parameters) 등
4. **MINOR**
   개발자 생산성에 약간 영향을 줄 수 있는 품질 결함
   ex) 줄이 너무 길지 않아야 하며 "switch" 문에는 최소 3개의 경우가 있어야 함(lines should not be too long, "switch" statements should have at least 3 cases) 등
5. **INFO**
   버그도 품질 결함도 아닌 단지 발견한 것

이상적으로 팀은 새로운 이슈(새로운 기술적 부채)를 도입(introduce)하지 않을 것이다.
SonarLint는 코드를 SCM으로 다시 Push 하기 전에 로컬 분석을 수행하여 취약 코드를 확인할 수 있고 이는 개발자를 도울 수 있다.
그러나 실제 업무에서 새로운 기술적 부채 없이 코딩 할 수있는 것은 아니며, 때로는 그것에 가치가 없다.

그래서 새로운 이슈가 도입된다(introduced).

이슈 문맥 이해(Understanding issue context)

때때로 이슈는 자명하다. 예를 들어 팀이 init-lower, camelCase 변수 명명 규칙에 동의하고 My_variable에서 이슈가 발생한 경우, 해당 문제를 이해하는 데 많은 문맥이 필요하지 않는다.
그러나 다른 상황에서는 이슈가 제기된 이유를 이해하는 데 문맥이 필수적일 수 있다. 이것이 SonarQube가 이슈 메시지가 표시되는 주요 이슈 위치뿐만 아니라 보조 이슈 위치도 지원하는 이유이다.
예를 들어 보조 이슈 위치는 메서드에 인지 복잡성(Cognitive Complexity)을 추가하는 메서드의 코드 조각을 표시하는 데 사용된다.

그러나 기여하는 위치의 간단한 세탁 목록이 이슈를 이해하기에 충분하지 않은 경우가 있는데, 예를 들어 코드를 통해 일부 경로에서 null 포인터를 역 참조 할 수있는 경우,
실제로 필요한 것은 이슈 흐름(issue flows)이다. 각 흐름은 이슈가 발생할 수 있는 코드를 통과하는 정확한 경로를 표시하도록 정렬된 보조 위치***집합***이다.
예를 들어 리소스가 해제되지 않은 코드를 통해 여러 경로가 있을 수 있기 때문에 SonarQube는 여러 흐름을 지원한다.

여러 위치의 이슈에 대한 자세한 내용은 blocked URL 비디오 참고

이슈 수명 주기(Issues lifecycle)

상태(Status)

검출된 Issue는 다음 5개의 상태 중 하나를 가진다.

- **Open** : (자동) 새로운 이슈가 발견된 상태
- **Confirmed** : (수동) 이슈가 유효함을 확인한 상태
- **Resolved** : (수동) 이슈를 해결한 상태 - 다음 분석시 Close 된다.
  - **False Positive** : (수동) 오탐 등록
  - **Won't Fix** : (수동) 어떠한 이유로 고치지 않음.
- **Reopened** : (자동) Resolved된 이슈가 수정되지 않았을때 다시 오픈된다.
- **Closed** : (자동) 이슈가 종료된 상태이다.

해결책 (Resolution)

Close된 Issue는 다음 해결 방법 중 하나를 가진다.

- **Fixed** : (자동) 다음 분석에서 문제가 해결되었다고 표시되면 자동으로 설정된다.
- **Removed** : (자동) 코딩 규칙이 없거나 파일이 삭제되면 자동으로 설정된다.

Resolve된 Issue는 다음 해결 방법 중 하나를 가진다.

- **False Positive** : (수동)
- **Won't Fix** : (수동)

Issue 워크플로우

다음과 같은 경우 Issue가 자동으로 Close된다. → (상태: **Closed**)

- Issue(모든 상태)가 올바르게 수정됨. → (해결책: **Fixed**)
- 관련 코딩 규칙을 비활성화되었거나 더 이상 사용할 수 없기 때문에 더 이상 문제가 발생하지 않음. (예: 플러그인이 제거됨) → (해결책: **Removed**)

다음과 같은 경우 Issue가 자동으로 ReOpen된다. → (상태: **Reopened**)

- 수동으로 Fixed로 Resolve된 Issue(그러나 해결책이 **False Positive** 가 아님)는 후속 분석에서 여전히 존재하는 것으로 표시된다.

어떤 이슈가 "새로운" 이슈인지 이해(Understanding which Issues are "New")

이슈의 생성 날짜를 확인하기 위해 각 분석 중에 알고리즘을 실행하여 이슈가 새로운 이슈인지 또는 이전에 존재했는지 여부를 확인한다.
이 알고리즘은 이슈가 보고된 라인에 대한 콘텐츠 해시(공백 제외)에 의존한다. 여러 라인 이슈의 경우 첫 번째 라인의 해시가 사용된다.
각 파일에 대해(파일 이름 바꾸기 검색 후) 알고리즘은 이전 분석에서 이슈의 기본 목록을 가져와 이러한 이슈를 새 분석에서 보고한 원시 이슈 목록과 일치시키려고 시도한다.
알고리즘은 가장 강력한 증거를 사용하여 먼저 일치를 시도한 다음 약한 휴리스틱으로 되돌아간다.

- 이슈가 동일한 규칙에 있고 동일한 행 번호와 동일한 행 해시를 사용하는 경우(그러나 반드시 동일한 메시지가 있는 것은 아님) > MATCH
- 파일 내에서 블록 이동을 감지한 다음 이슈가 동일한(이동된) 줄과 동일한 규칙에 있는 경우(그러나 반드시 동일한 메시지가 있는 것은 아님) > MATCH
- 동일한 규칙, 동일한 메시지 및 동일한 행 해시 사용(그러나 반드시 동일한 행일 필요는 없음) > MATCH
- 동일한 규칙에 따라 동일한 메시지와 동일한 행 번호(그러나 반드시 동일한 행 해시일 필요는 없음) > MATCH
- 동일한 규칙 및 동일한 행 해시 사용(그러나 동일한 메시지 및 동일한 행이 아님) > MATCH
- 일치하는 CLOSED 이슈가 있습니까 > MATCH 및 Reopen

일치하지 않는 "기본(base)" 이슈는 수정 된(Fixed) 것으로 Close된다.

일치하지 않는 "원시(raw)" 이슈는 새로운(New) 것이다.

이슈 백데이트 이해(Understanding Issue Backdating)

위에서 설명한 것처럼 이슈가 "새로운" 것으로 결정되면 다음 질문은 이슈를 제공할 날짜다. 예를 들어 오랫동안 코드에 존재했지만 프로필에 새 규칙이 추가되었기 때문에 가장 최근의 분석에서만 발견된다면 어떻게 될까?
이 이슈에 해당 라인의 마지막 변경 날짜를 지정해야 할까, 아니면 처음 제기된 분석 날짜를 지정해야 할까? 즉, 소급되어야 할까?
라인에 대한 마지막 변경 날짜를 사용할 수 있는 경우(SCM 통합 필요) 특정 상황에서 이슈는 소급된다.

- 프로젝트 또는 분기의 첫 번째 분석 시
- 프로필에서 규칙이 새 규칙인 경우(활성화된 새 규칙 또는 비활성화되어 활성화된 규칙)
- SonarQube가 방금 업그레이드되었을 때 (규칙 구현이 더 똑똑해질 수 있기 때문에)
- 규칙이 외부에 있는 경우

결과적으로, 백데이트가 새로 제기 된 이슈를 새 코드에서 제외시킬 가능성이 있다.

자동 문제 할당(Automatic Issue Assignment)

버그, 취약점 및 코드 냄새의 경우(For Bug, Vulnerability and Code Smell)

Committer가 SonarQube 사용자와 상호 연관될 수 있는 경우, 분석 중에 이슈 라인의 마지막 Committer에 새 이슈가 자동으로 할당된다.
참고로 현재 파일 위의 모든 레벨의 이슈(예: Directory/Project)는 자동으로 할당될 수 없다.

사용자 상관 관계(User Correlation)

로그인 및 이메일 상관 관계는 자동으로 이루어진다. 예를 들어 사용자가 이메일 주소로 커밋하고 해당 이메일 주소가 SonarQube 프로필의 일부인 경우,
사용자가 마지막 Committer였던 라인에서 발생한 새로운 이슈가 사용자에게 자동으로 할당된다.

추가 상관 관계는 사용자 프로필에서 수동으로 수행할 수 있다.

알려진 제한 사항(Known Limitation)

이슈와 연결된 SCM 로그인이 이슈 작성자(Issue author)에게 허용되는 255자를 초과하는 경우, 작성자(Author)는 공백으로 처리된다.

문제 편집(Issue edits)

SonarQube의 이슈 워크플로우(Issues workflow)는 이슈를 관리하는 데 도움이 될 수 있다. 이슈에 대해 수행할 수 있는 일곱 가지 작업(코드에서 수정하는 것 외에)은
주석 달기(Comment), 할당(Assign), 확인(Confirm), 심각도 변경(Change Severity), 해결(Resolve), 수정하지 않음(Won't Fix) 및 거짓 긍정(False Positive)이다.

이러한 작업은 세 가지 범주로 나눌 수 있다.

첫 번째는 "기술 검토(Technical review)" 범주다.

기술 검토(Technical Review)

확인(Confirm), 거짓 긍정(False Positive), 수정하지 않음(Won't Fix), 심각도 변경(Severity change) 및 해결(Resolve) 작업은 모두 이 범주에 속하며,
이슈의 유효성을 확인하기 위해 이슈를 초기 검토(Initial review)한다고 가정한다.
마지막 검토 기간(하루, 일주일 또는 전체 스프린트)에 추가 된 기술 부채(Technical debt)를 검토할 시간(Time to review)이라고 가정한다. 각각의 새로운 이슈를 살펴보고 다음을 수행한다.

- **Confirm** : 이슈를 확인하면 기본적으로 "네, 그건 문제다."라고 말한다. 이렇게 하면 "Open" 상태에서 "Confirmed"으로 이동한다.
- **False Positive** : 문맥에서 이슈를 살펴보면 그 이슈가 어떤 이유로든 실제로 문제가 되지 않는다는 것을 알게 된다.
  그래서 당신은 그것을 **False Positive**으로 표시하고 계속 진행한다. 이 때 프로젝트에 대한 문제 관리 권한(Administer Issues permission)이 필요하다.
- **Won't Fix** : 문맥에서 이슈를 살펴보면 그 이슈가 유효한 문제이지만 실제로 해결해야 할 문제는 아니라는 것을 알 수 있다. 즉, 그것은 받아 들여지는 기술적 부채(Technical debt)를 나타낸다.
  그래서 당신은 그것을 **Won't Fix**로 표시하고 계속 진행한다. 이 때 프로젝트에 대한 문제 관리 권한(Administer Issues permission)이 필요하다.
- **Severity change** : 처음 두 옵션 사이의 중간 지점을 뜻한다. 그 이슈는 문제가 맞지만, 규칙의 기본 심각도(Default severity) 만큼 심각한 문제는 아니거나 실제로 더 심각한 문제 일 수 있다.
  어느 쪽이든, 이슈의 심각도를 조정이 필요하다고 생각한다. 이 때 프로젝트에 대한 문제 관리 권한(Administer Issues permission)이 필요하다.
- **Resolve** : Open된 이슈를 수정했다고 생각되면 해결할 수 있다. 옳다면, 다음 분석에 그 이슈는 Close된 상태로 변경될 것이고, 틀렸다면, 그 이슈의 상태는 Re-open될 것이다.

많은 이슈를 **거짓 긍정(False Positive)** 또는 **수정하지 않음(Won't Fix)**으로 표시하는 경향이 있다면, 일부 코딩 규칙이 문맥에 적합하지 않다는 것을 의미한다.
따라서 품질 프로필(Quality Profile)에서 완전히 비활성화하거나 이슈 제외를 사용하여 애플리케이션의 특정 부분(또는 개체 유형)에 사용되지 않도록 규칙의 초점을 좁힐 수 있다.
마찬가지로 심각도를 자주 변경할 경우, 프로필에서 규칙 심각도를 업데이트하는 것을 고려하라는 메시지가 표시된다.

이슈를 편집하면 관련 메트릭(예: 신규 버그)이 자동으로 업데이트되며, 관련된 경우 품질 게이트(Quality Gate) 상태도 자동으로 업데이트된다.

위치 변경(Dispositioning)

일단 이슈가 기술적 검토(Technical review)를 거치면 누가 이슈를 처리 할 것인지 결정할 시간이 온다. 기본적으로 이슈가 발생할 때 이슈 라인의 마지막 Committer에 할당되지만,
자신이나 다른 사람에게 다시 할당할 수 있다. 알림을 등록한 경우 할당된 사용자는 할당 받은 이슈에 대한 메일 알림을 받게 되며,
내 계정 공간의 내 이슈 목록을 포함하여 이슈가 표시되는 모든 곳에 할당된 사용자가 표시된다.

일반(General)

이슈의 수명 주기 동안 언제든지 이슈에 대한 설명을 기록할 수 있다. 댓글(Comment)은 실행 중인 로그의 문제 세부 정보에 표시된다. 작성한 댓글(Comment)을 편집하거나 삭제할 수 있다.

이슈의 태그를 편집할 수 있다. 이슈는 해당 태그를 만든 규칙의 태그를 상속하지만, 문제에 설정된 태그는 완전히 편집할 수 있다.
태그는 프로젝트에 대한 찾아보기 권한(Browse permissio)이 있는 사용자에 대해 마음대로 생성, 추가 및 제거할 수 있다.
처음에는 관련 규칙에서 상속되지만 이슈에 대한 태그는 규칙과 동기화되지 않으므로 이슈에 태그를 추가해도 해당 태그가 규칙의 이슈에 추가되지는 않습니다.

대량 변경(Bulk Change)

이러한 모든 변경 사항은 이슈 검색 결과 창의 대량 변경(Bulk Change) 옵션을 사용하여 한 번에 여러 이슈에 대해 수행할 수 있다.

닫힌 문제 제거(Purging Closed Issues)

기본적으로 Close된 이슈는 30일 동안 유지된다. 자세한 내용은 하우스 키핑을 참조.

참조 링크

- Issues | SonarQube Docs