SonarQube / SonarLint

SonarQube / SonarLint

https://www.sonarsource.com/products/sonarqube

SonarQube

- créé en 2006, maintenue par l'entreprise SonarSource (freemium)
- plateforme open-source pour l'inspection continue de la qualité du code
- serveur Web fournissant des pages d'analyse de la qualité du code
- s'intègre facilement avec les moteurs de production (e.g. Gradle) et les outils de CI/CD comme GitLab

SonarLint

- plugin pour IDE qui permet une partie de l'analyse pendant le codage
- peut être lié à une instance de SonarQube

SONARQUBE FEATURES

the code quality tool for better code

Enable your team to systematically deliver and meet high code quality standards, for every project, at every step of the workflow.

30+ languages, frameworks & IaC platforms

Analyze the code quality of all the languages in your projects. Patch bugs, close vulnerabilities and follow best practices with a single source of truth.

integration with DevOps platforms

Easy project onboarding with integration to GitHub, GitLab, Azure and Bitbucket; in-cloud & on-prem. Plus a Jenkins plugin and easy integration with popular CI tools and build systems.

clear go/no-go Sonar Quality Gate

Fail pipelines when the code quality doesn't meet your defined requirements and prevent problems from being merged or deployed.

high operability

Run your instance your way, as a service, on Docker, or with Kubernetes with vertical and horizontal scaling support, plus multi-threaded, server-side processing.

super-fast analysis

Super-fast analysis gets you actionable Clean Code metrics in minutes instead of hours.

critical security rules for vital languages

Receive actionable, high-precision feedback at the right place and time. Benefit from 5,000+ coding rules and industry-leading taint analysis of Java, C#, PHP, Python, TypeScript & JavaScript.

shared, unified configurations

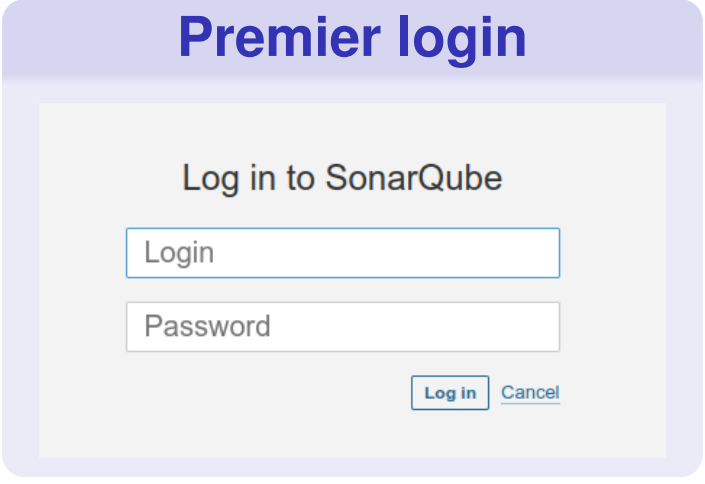
Align your team with a consistent definition of code health. Collaborate efficiently in making your code clean and meeting your team's code quality expectations.

Sonarlint IDE integration

Add the SonarLint extension to your favorite IDE and find code issues on the fly. SonarQube rules and analysis settings synchronize to SonarLint, aligning teams around a single standard of Clean Code.

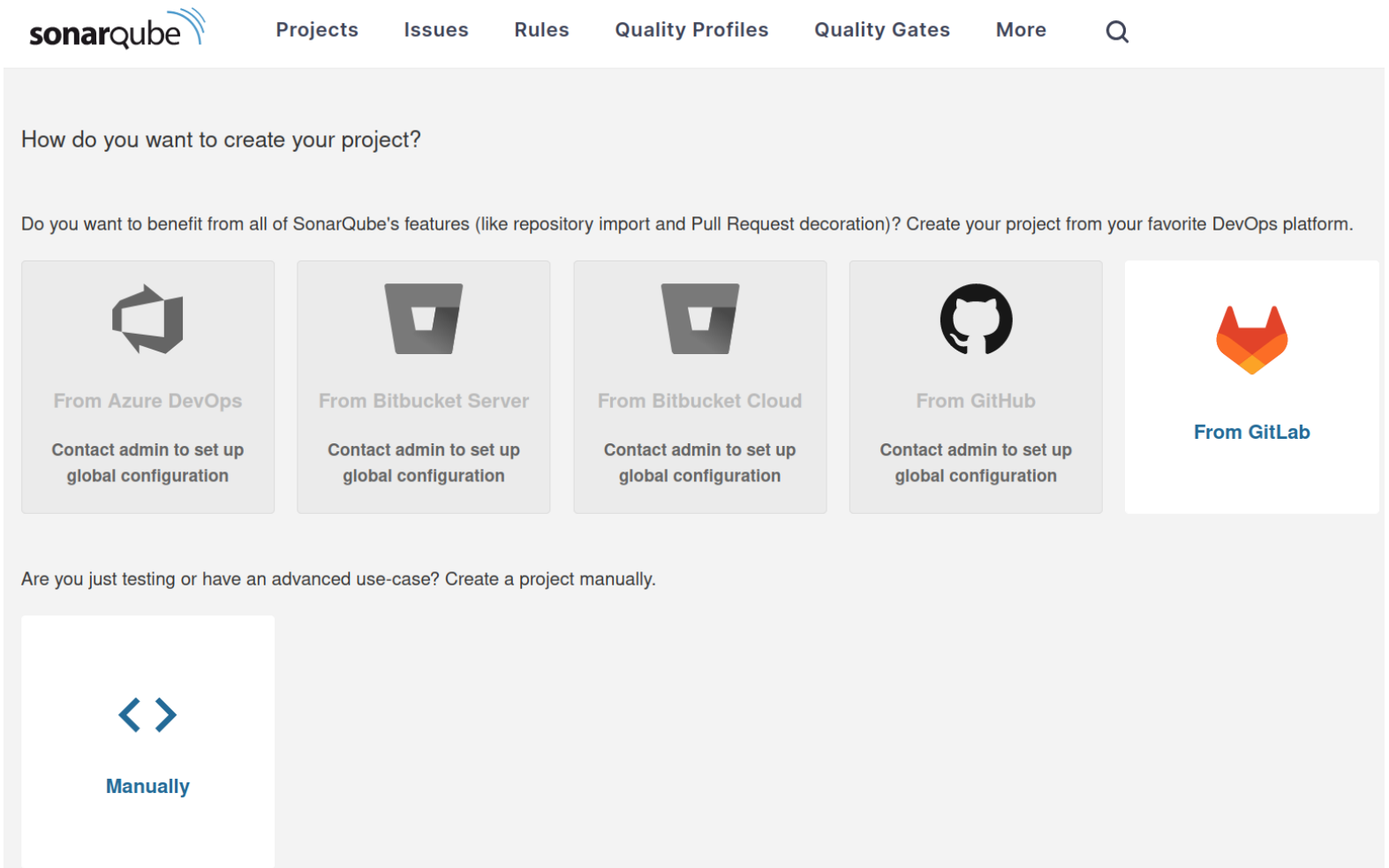
Utilisation du SonarQube de l'IUT

Nous utiliserons l'instance suivante : ▸ SonarQube IUT

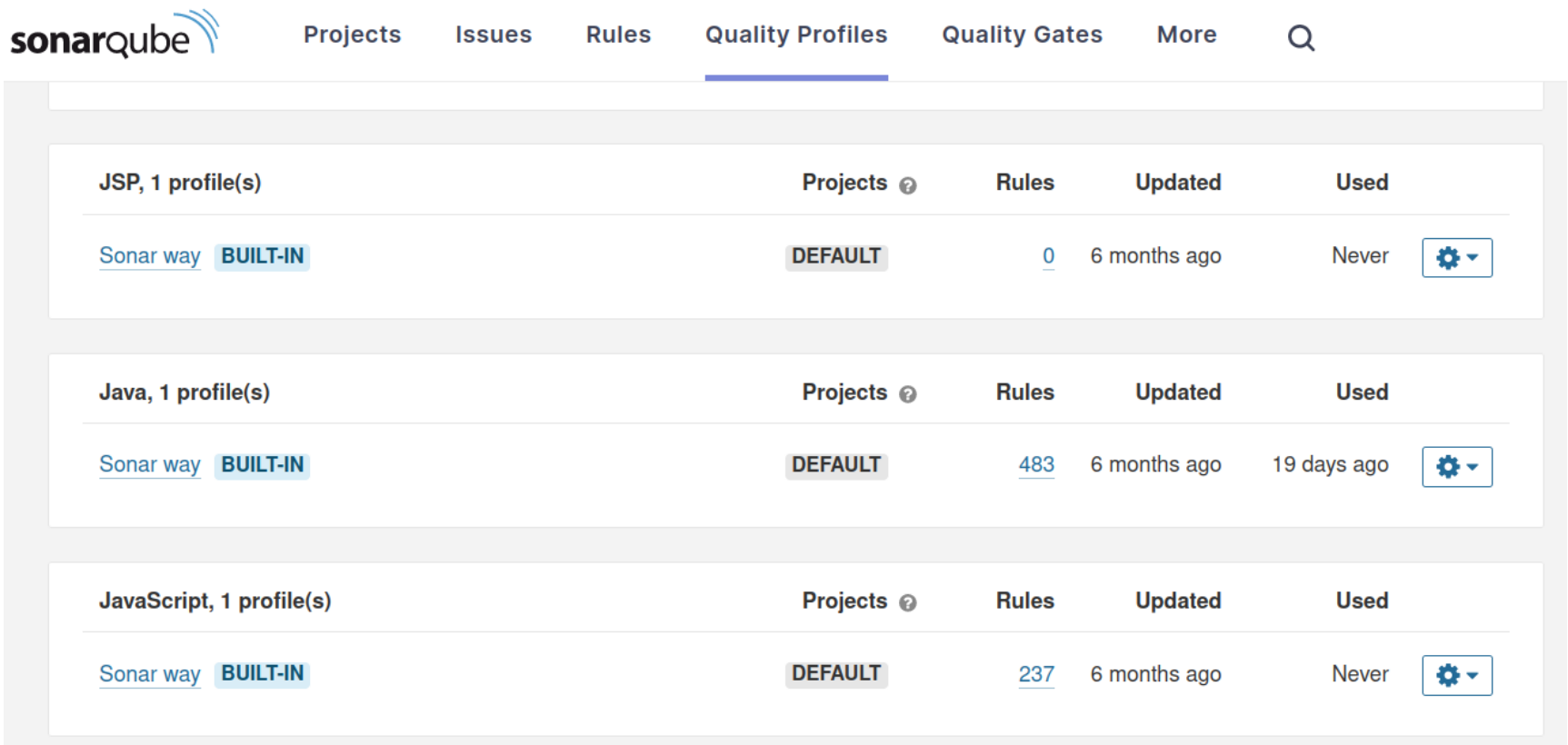


Utilisez vos identifiants habituels.

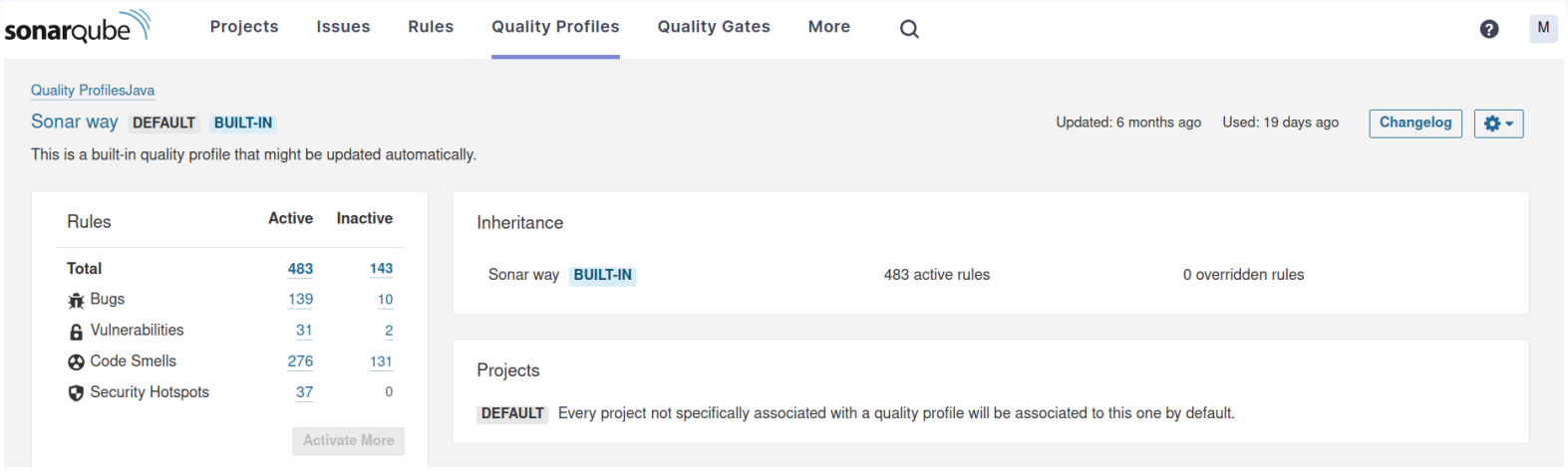
Première page d'accueil



Quality Profiles



Java Quality Profile

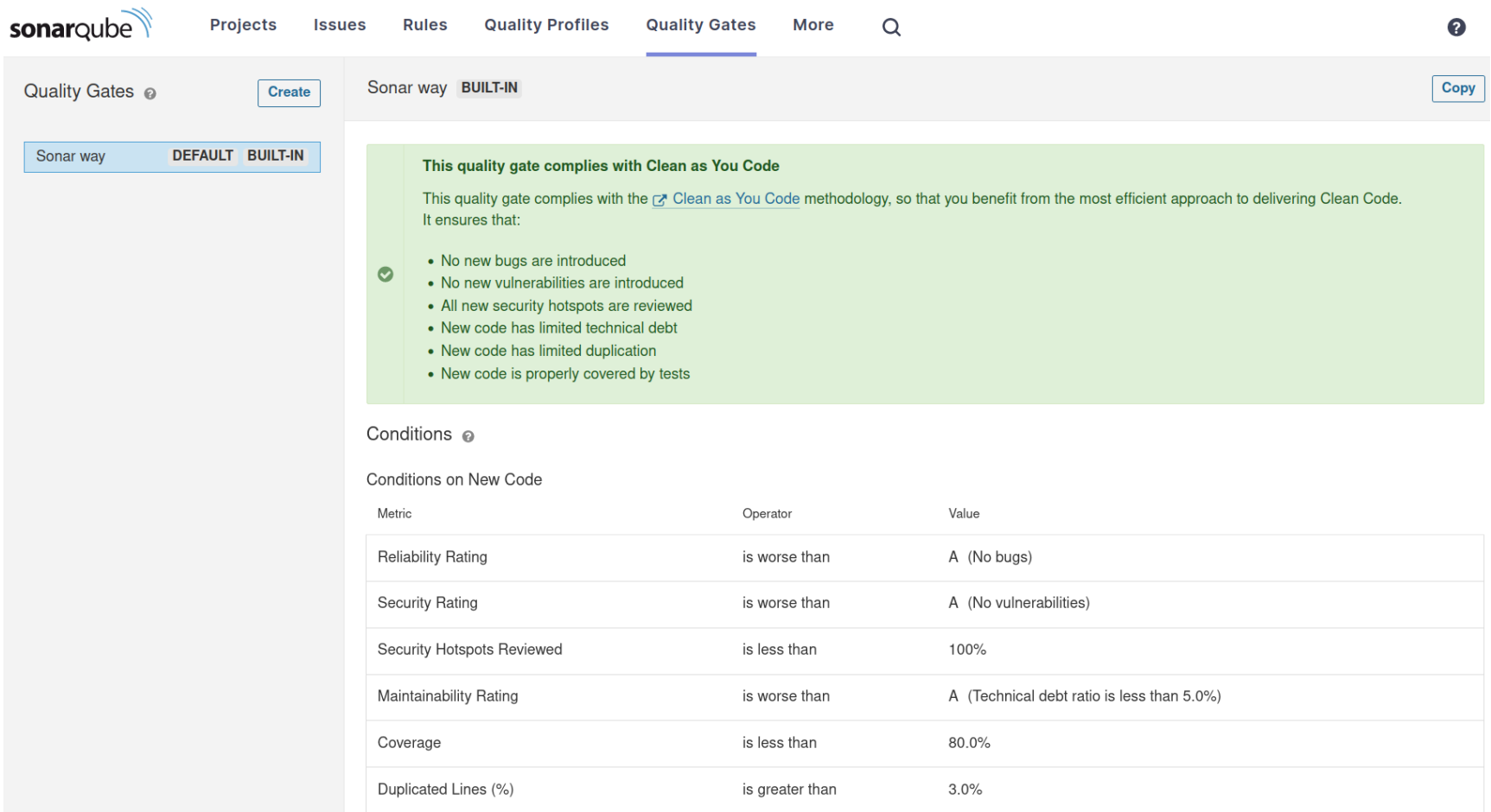


Le profile Java par défaut : 483 règles actives, i.e. une norme ou une pratique de codage qui doit être suivie

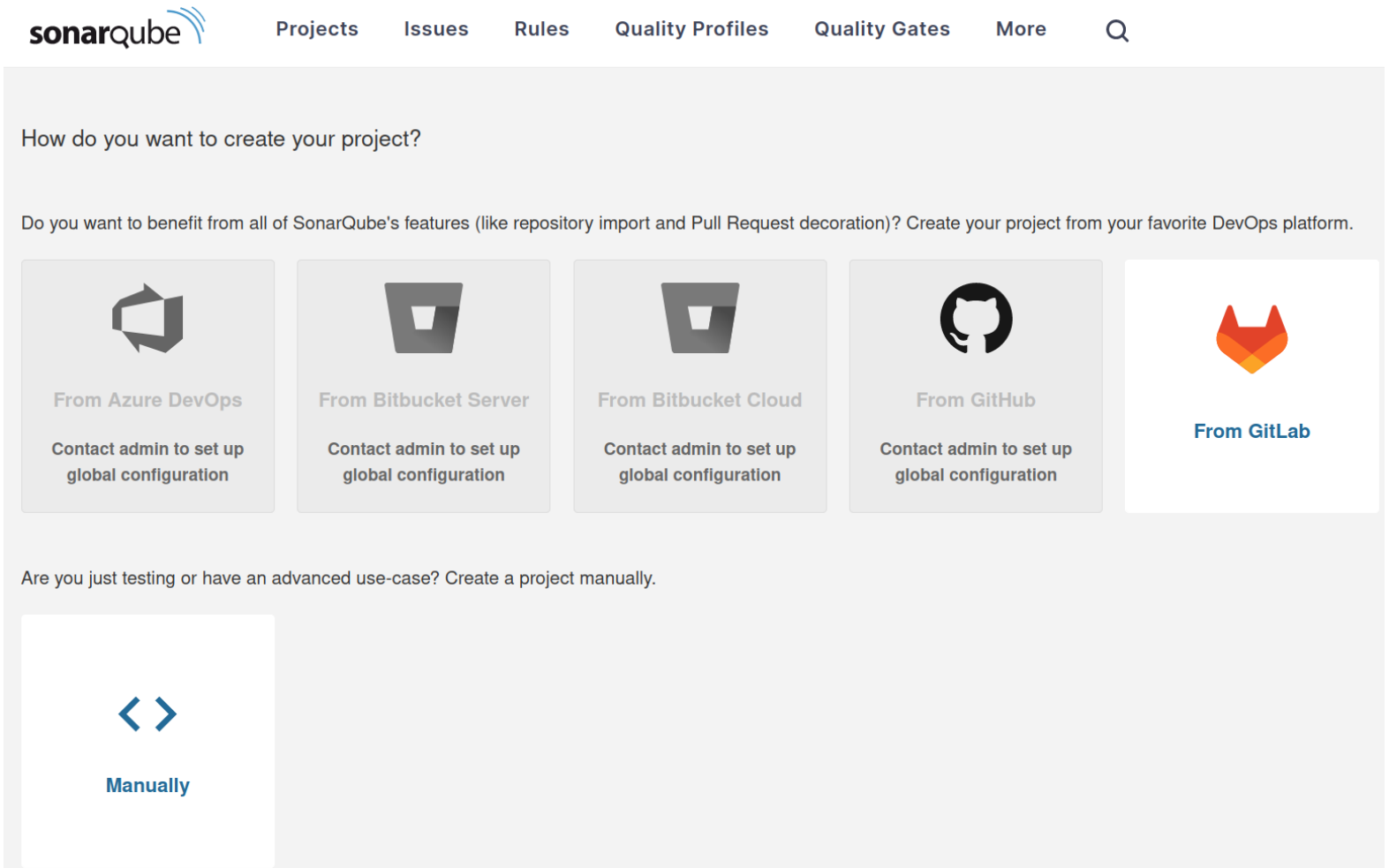
- *Bugs* : anomalies évidentes du code, qui impactent la fiabilité.
- *Vulnerabilities* : faiblesses du code pouvant nuire au système.
- *codes smells* : code mal écrit qui peut ralentir le développement et augmenter les coûts de maintenance.
- *Security hotspot* : code sensible à la sécurité et qui doit être examiné manuellement.

Quality Gates

Permet de définir des seuils de qualité pour le code ajouté après une analyse : par exemple pour stopper un pipeline de CI



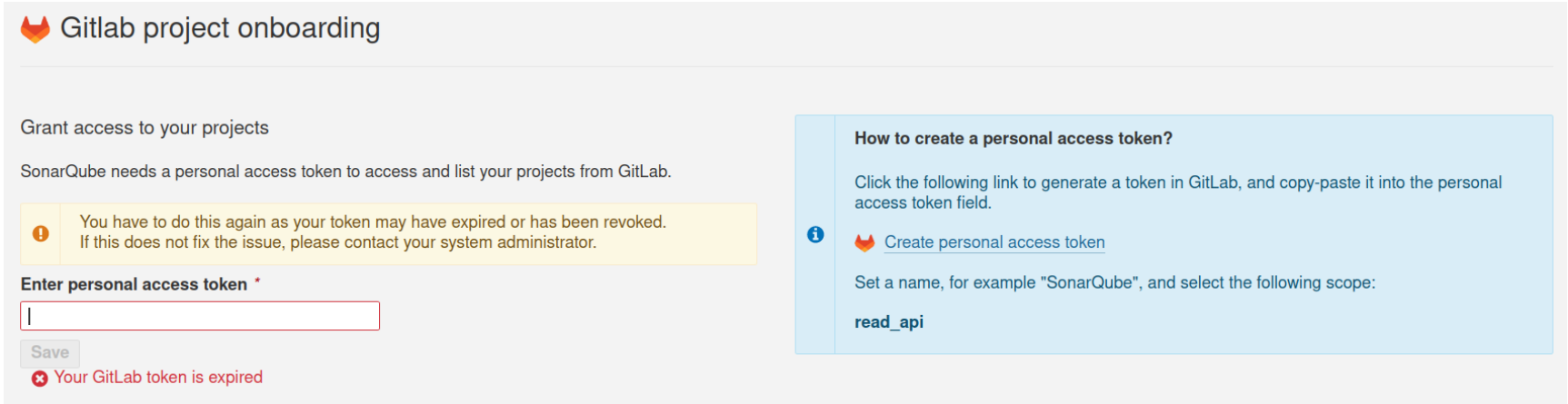
Import d'un projet depuis GitLab



Création d'un token pour l' interaction SonarQube / GitLab

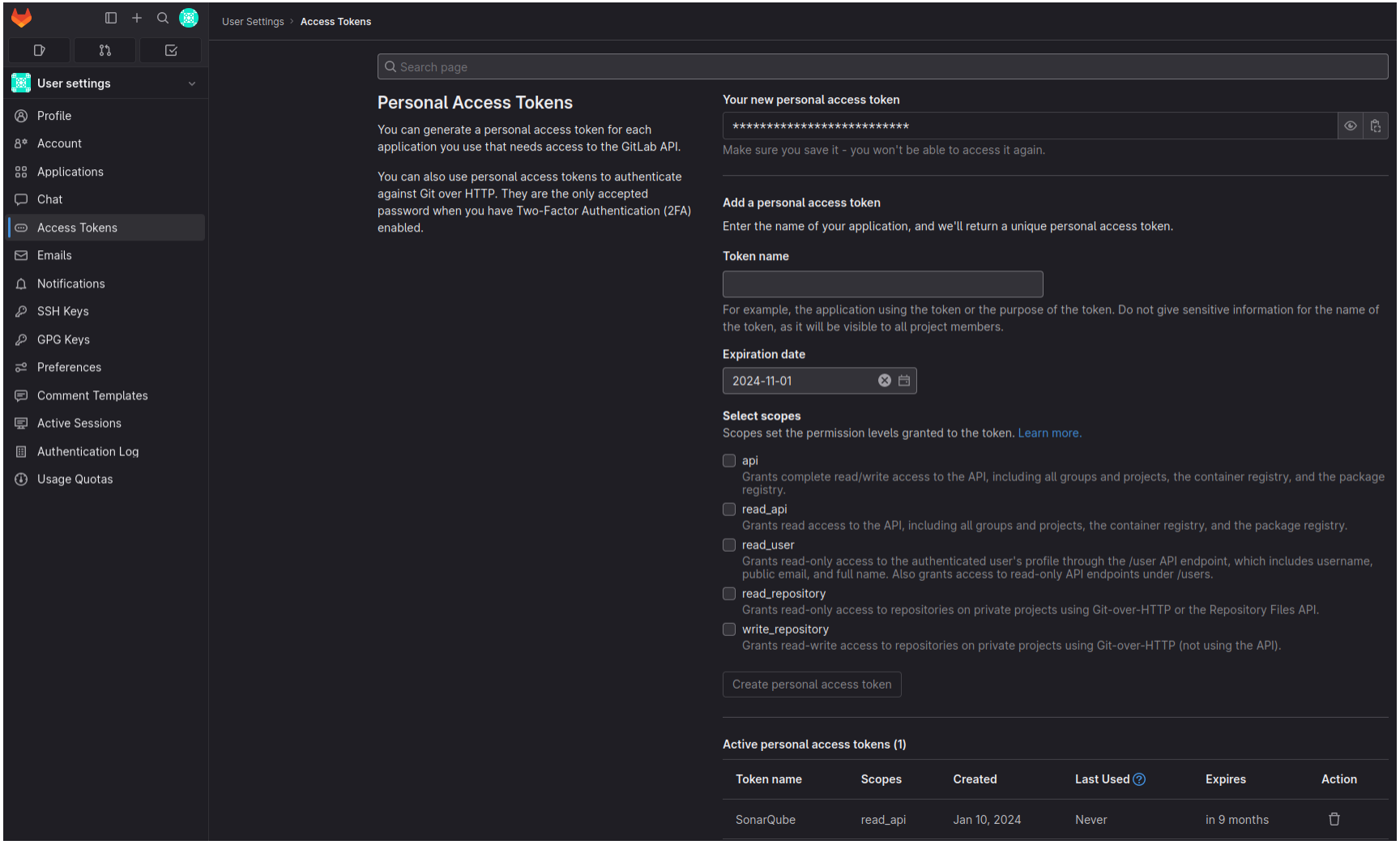
Pour pouvoir importer vos projets depuis GitLab, il faut que SonarQube puisse communiquer avec notre instance de GitLab.

SonarQube demande de créer un token depuis l'IU de GitLab :



Dans GitLab

Dans le paramétrage de votre profil :

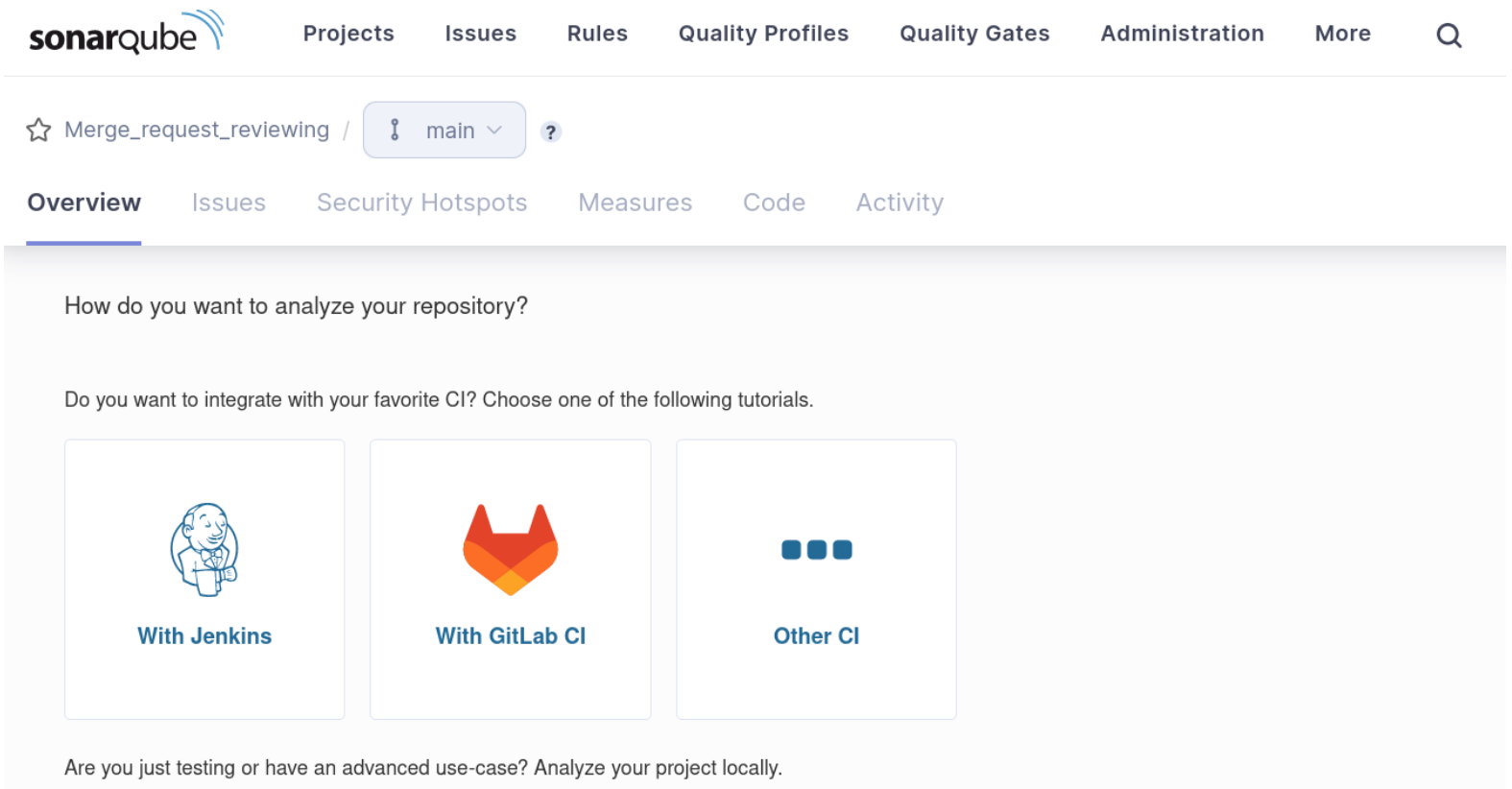


Dans SonarQube

Une fois le token ajouté, la liste des projets GitLab appraît dans SonarQube :

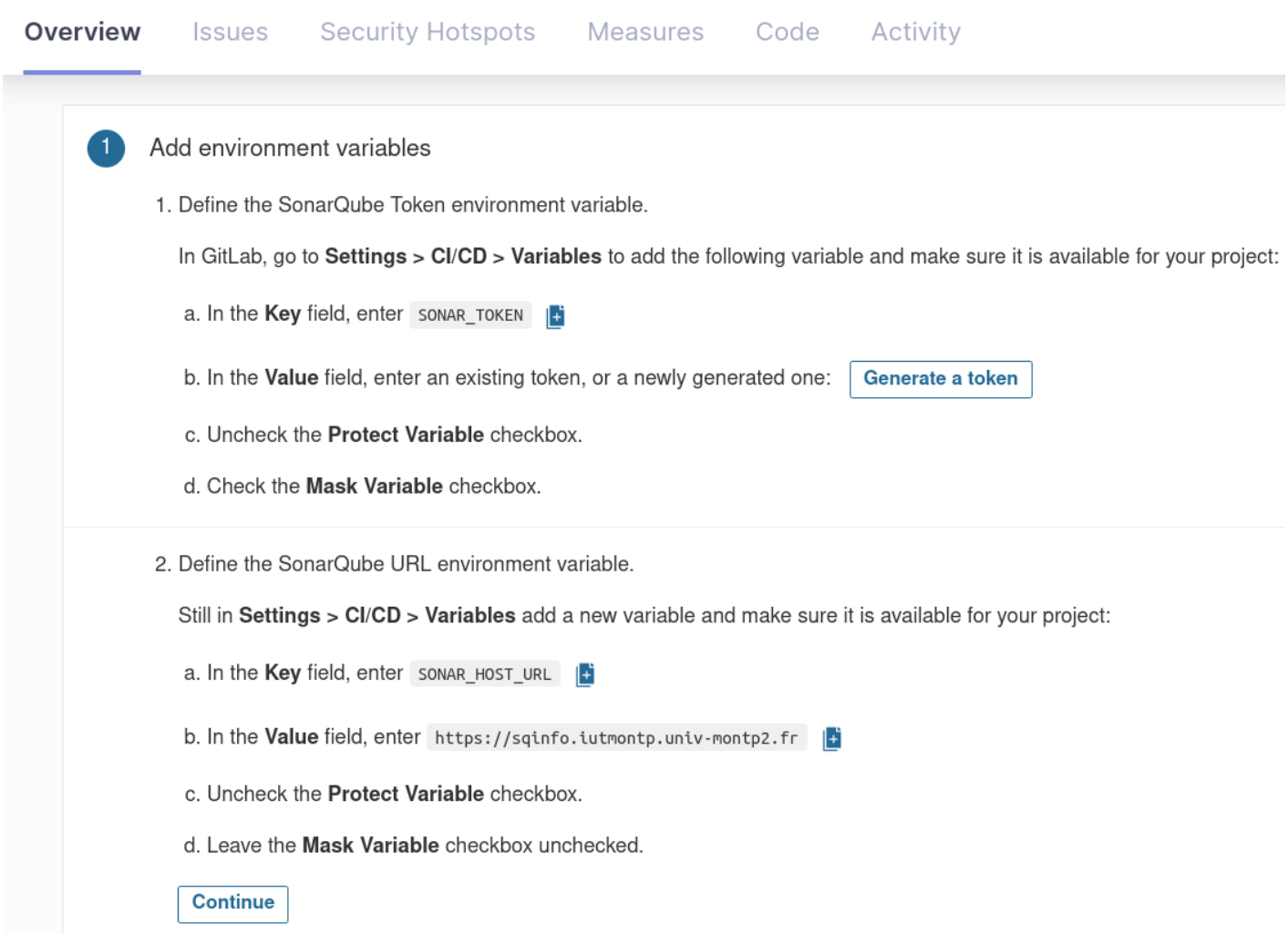
Choix du type de CI

Choisir permet d'obtenir des squelettes pour les scripts de Gradle et de CI



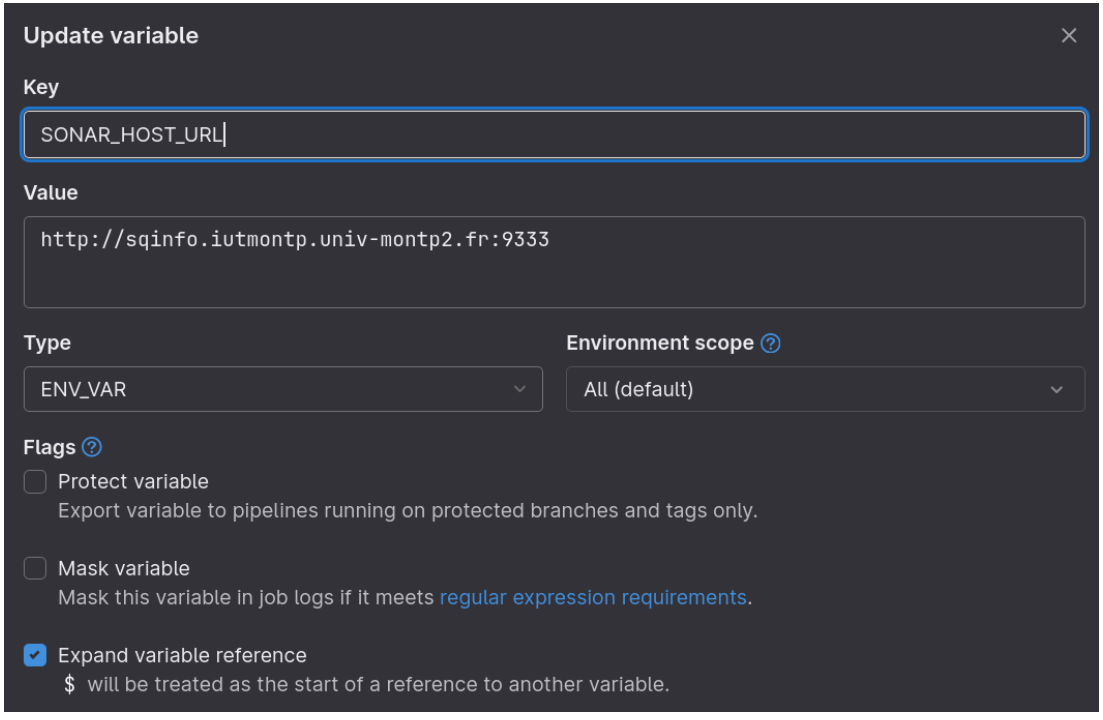
Info à utiliser dans GitLab

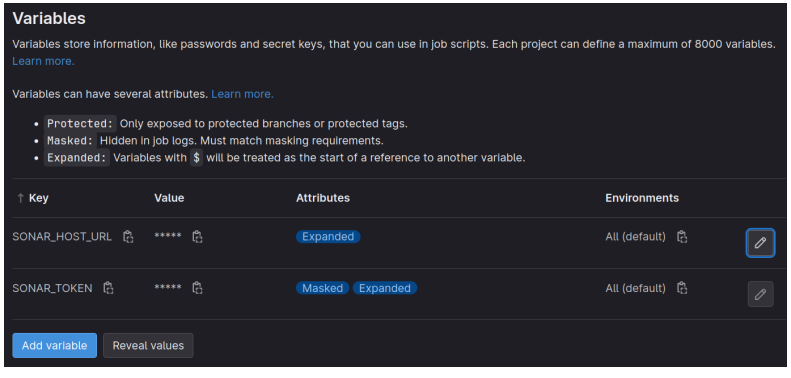
Création des variables qui seront utilisées dans le script de CI



Info à utiliser dans GitLab

Attention à bien mettre l'URL suivante ; en http et avec le bon port





Mise en place dans le projet GitLab

2 Create or update the configuration file

1. What option best describes your build?

Maven | Gradle | .NET | Other (for JS, TS, Go, Python, PHP, ...)

2. Add the following to your `build.gradle` or `build.gradle.kts` file:

build.gradle | build.gradle.kts

```
plugins {
  id "org.sonarqube" version "4.2.1.3168"
}

sonar {
  properties {
    property "sonar.projectKey", "etumichel_merge_request_reviewing_AYzy-Hx6gDYCt8iirpi
    property "sonar.projectName", "Merge_request_reviewing"
    property "sonar.qualitygate.wait", true
  }
}
```

Copy

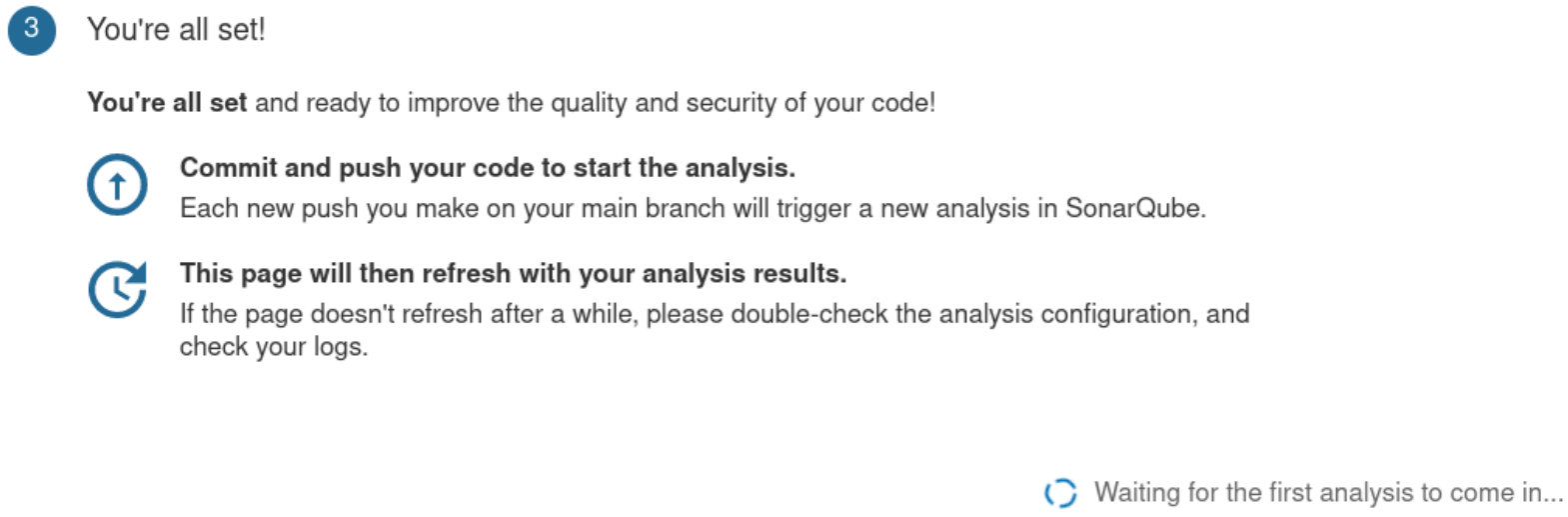
3. Create or update your `.gitlab-ci.yml` file with the following content.

```
sonarqube-check:
  image: gradle:jre11-slim
  variables:
    SONAR_USER_HOME: "${CI_PROJECT_DIR}/.sonar"  # Defines the location of the analysis
    GIT_DEPTH: "0"  # Tells git to fetch all the branches of the project, required by
  cache:
    key: "${CI_JOB_NAME}"
    paths:
      - .sonar/cache
  script: gradle sonar
  allow_failure: true
  rules:
    - if: $CI_COMMIT_BRANCH == 'main'
```

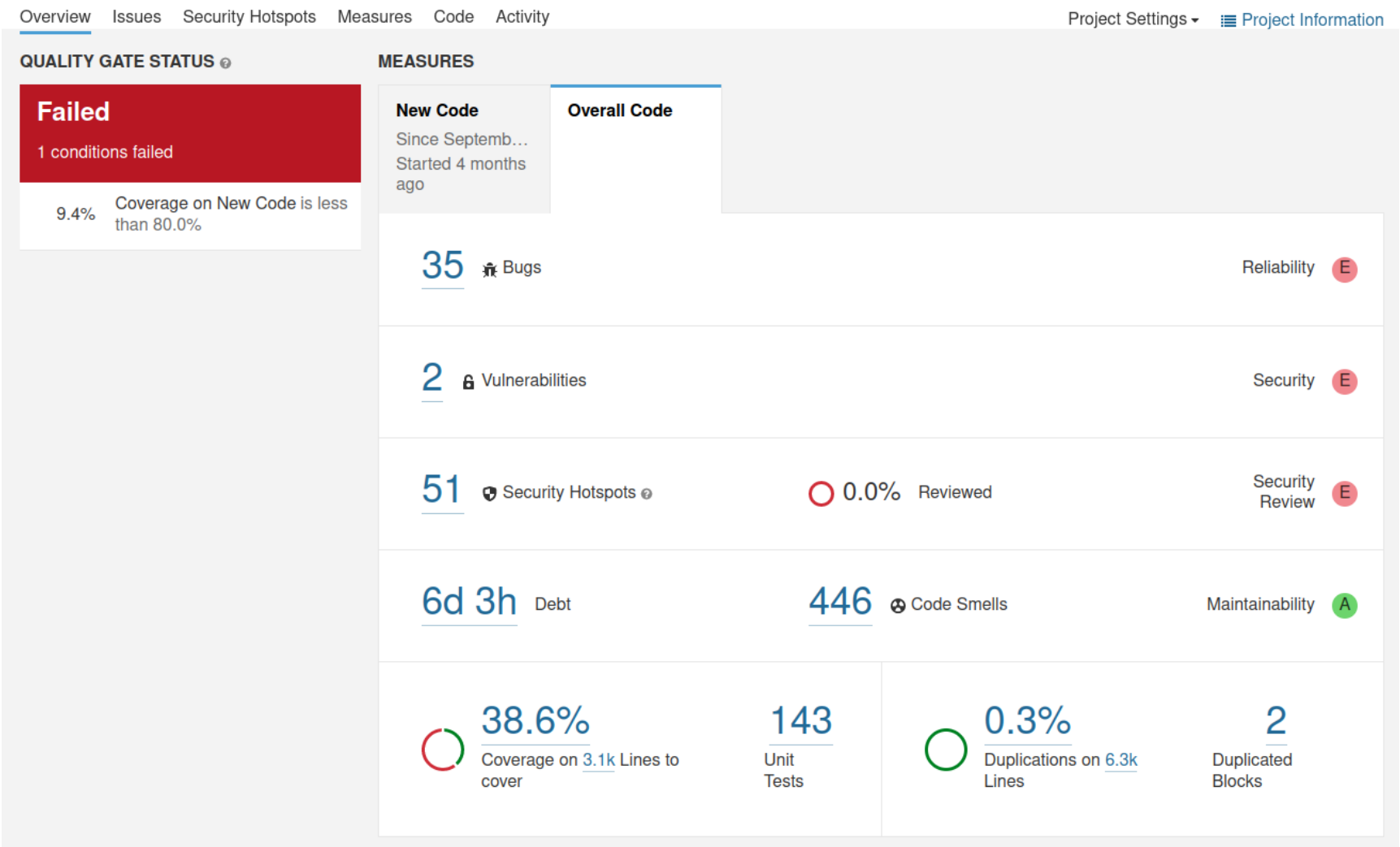
Copy

Il faudra corriger ce script : vous trouverez un projet `Gradle_Sonar_Java_Template` dans notre GitLab, afin d'avoir un exemple pour `app/build.gradle` et `.gitlab-ci.yml`

Il n'y a plus qu'à faire un commit pour que le résultat de l'analyse soit disponible sous SonarQube



Exemple d'un résultat d'analyse



SonarQube peut fonctionner avec Jacoco

Jacoco est un outil qui mesure la couverture des tests des projets Java

Première étape : ajout de la fonctionnalité dans le build de Gradle

Example 1. Applying the JaCoCo plugin

Kotlin | Groovy

build.gradle

```
plugins {
    id 'jacoco'
}
```

If the Java plugin is also applied to your project, a new task named `jacocoTestReport` is created. By default, a HTML report is generated at `layout.buildDirectory.dir("reports/jacoco/test")`.

NOTE

While tests should be executed before generation of the report, the `jacocoTestReport` task does not depend on the `test` task.

Depending on your usecases, you may want to always generate the `jacocoTestReport` or run the `test` task before generating the report explicitly.

Example 2. Define dependencies between code coverage reports and test execution

Kotlin | Groovy

build.gradle

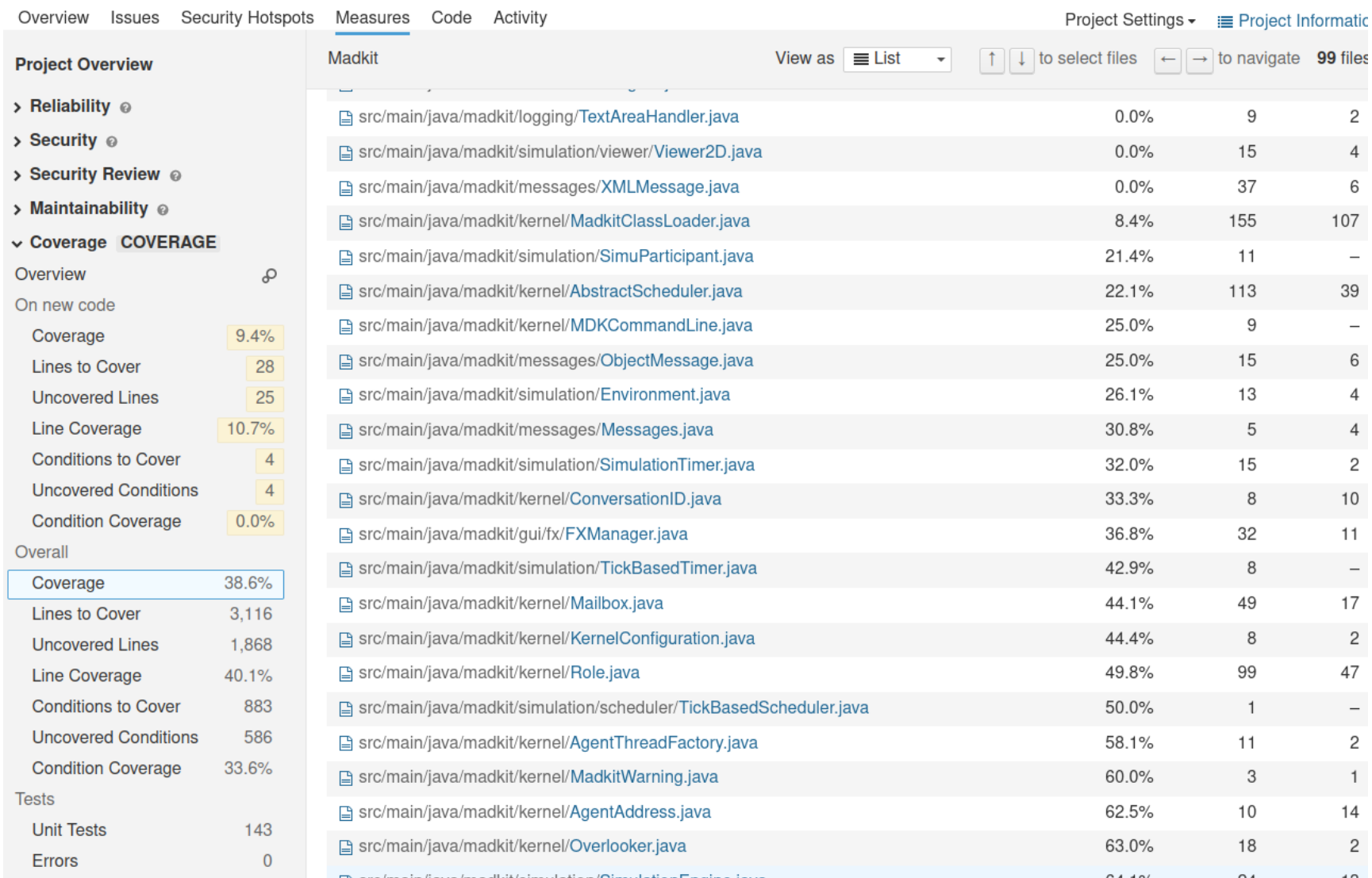
```
test {
    finalizedBy jacocoTestReport // report is always generated after tests r
}
jacocoTestReport {
    dependsOn test // tests are required to run before generating the repor
}
```

SonarQube peut fonctionner avec Jacoco

Deuxième étape : modification du script de CI. Exemple :

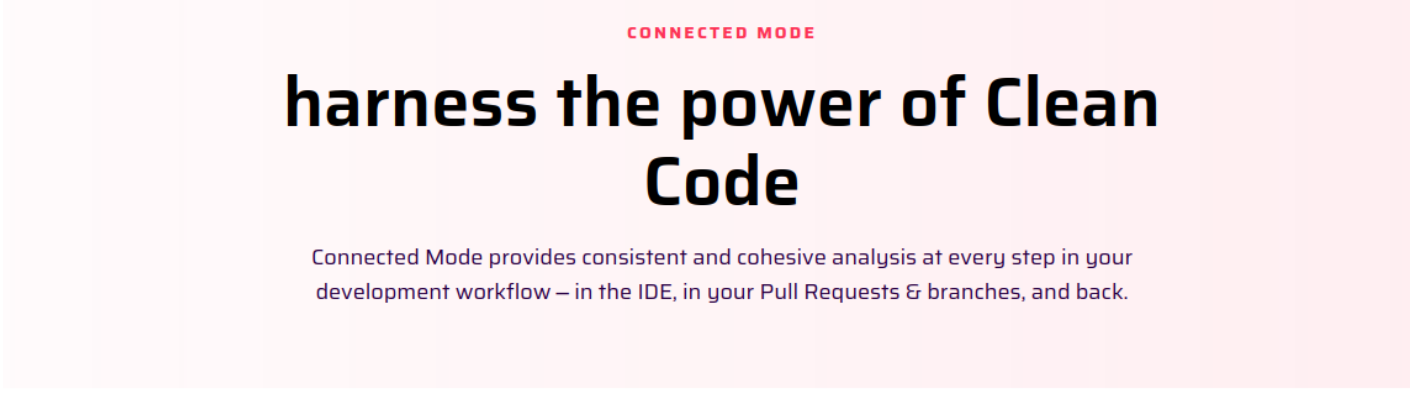
```yaml
stages:
  - test
  - check

before_script:
  - GRADLE_USER_HOME="$(pwd)/.gradle"
  - export GRADLE_USER_HOME

test:
  image: gradle:jre11-slim
  stage: test
  script: ./gradlew test

sonarqube-check:
  stage: check
  image: gradle:jre11-slim
  variables:
    SONAR_USER_HOME: "${CI_PROJECT_DIR}/.sonar"  # Defines the location of the analysis task cache
    GIT_DEPTH: "0"  # Tells git to fetch all the branches of the project, required by the analysis task
  cache:
    key: "${CI_JOB_NAME}"
    paths:
      - .sonar/cache
  script: ./gradlew sonar
  allow_failure: true
```

Exemple de résultat obtenu pour la couverture dans SonarQube



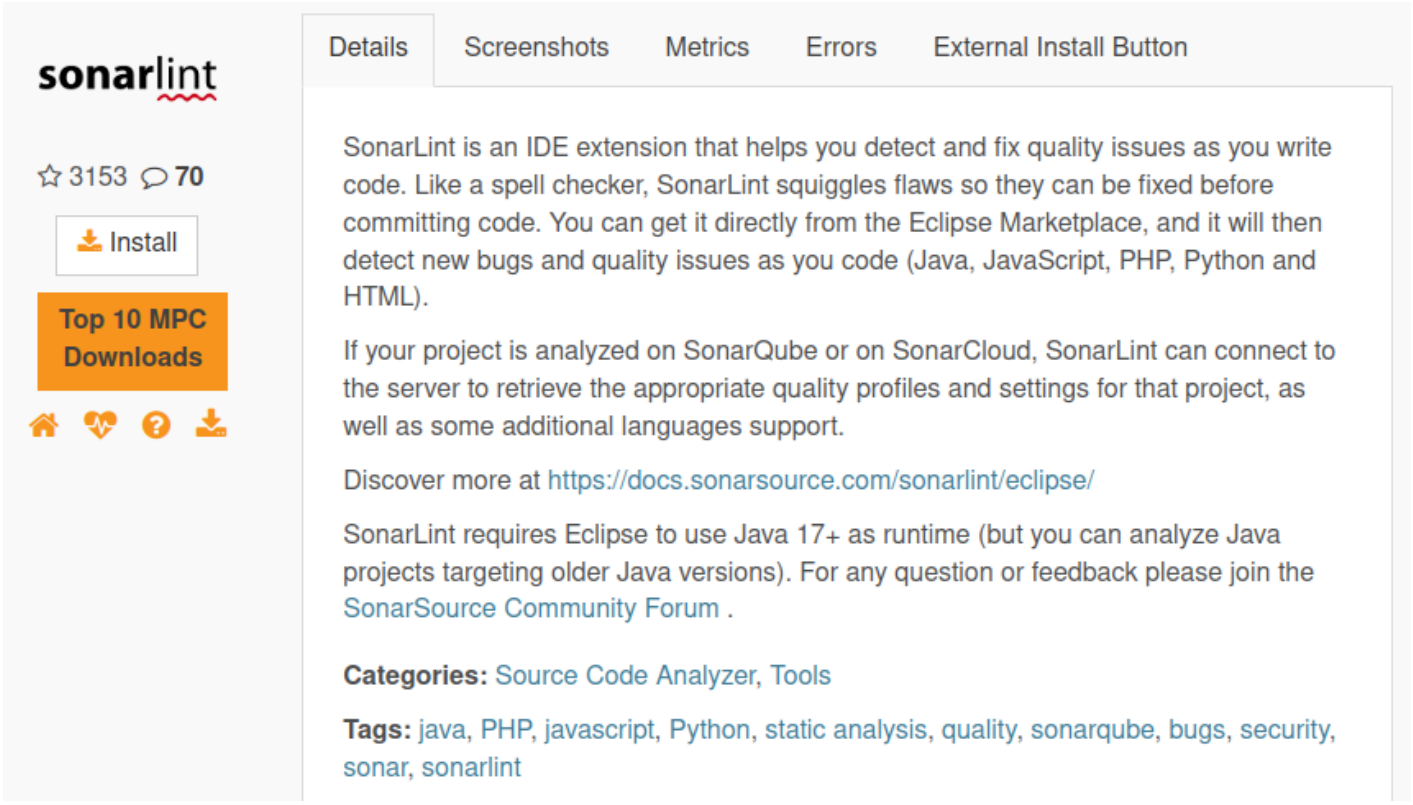
SonarLint

Plugin pour IDE qui permet de vérifier les règles sonarqube pendant l'édition du code source.



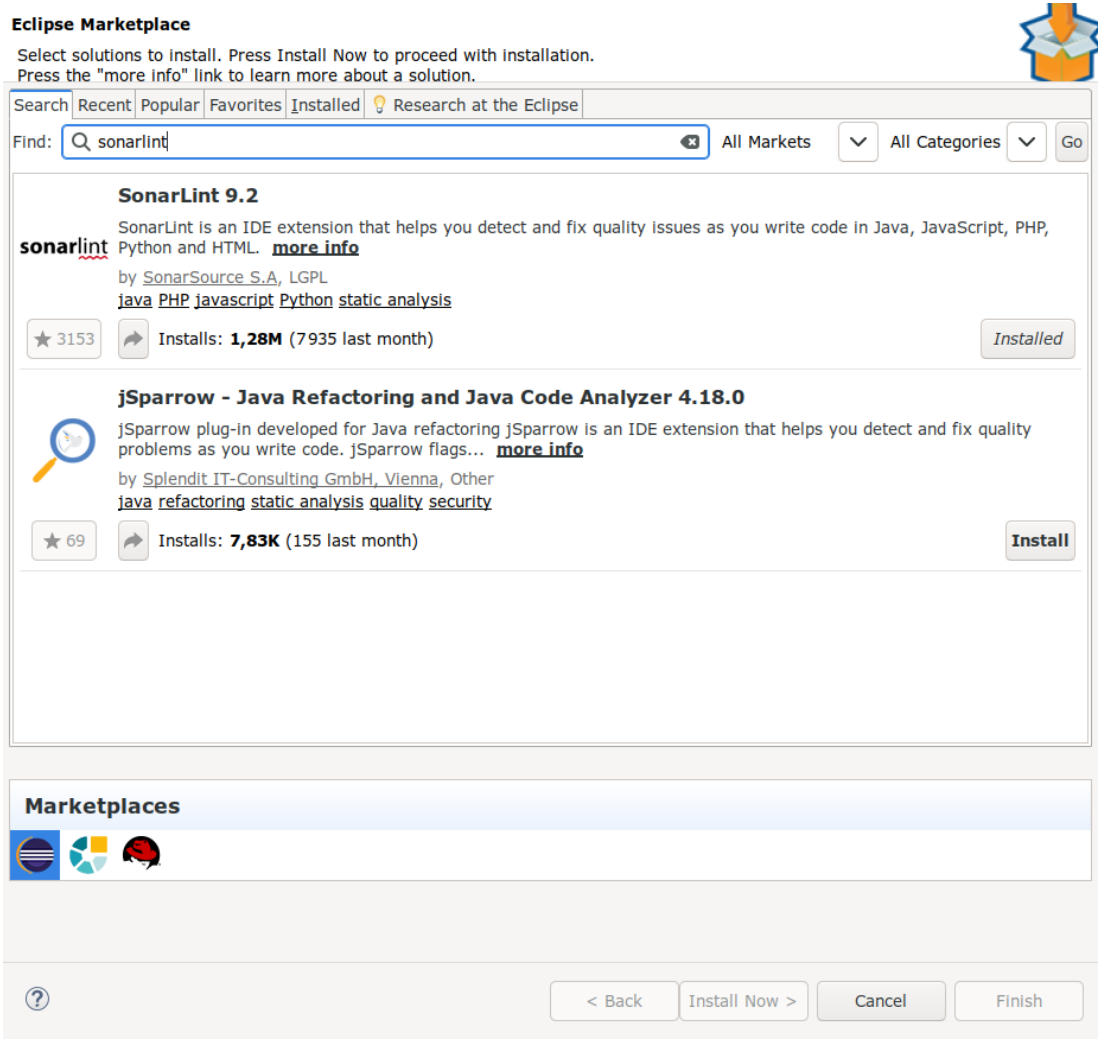
SonarLint

Plugin pour Eclipse :



SonarLint

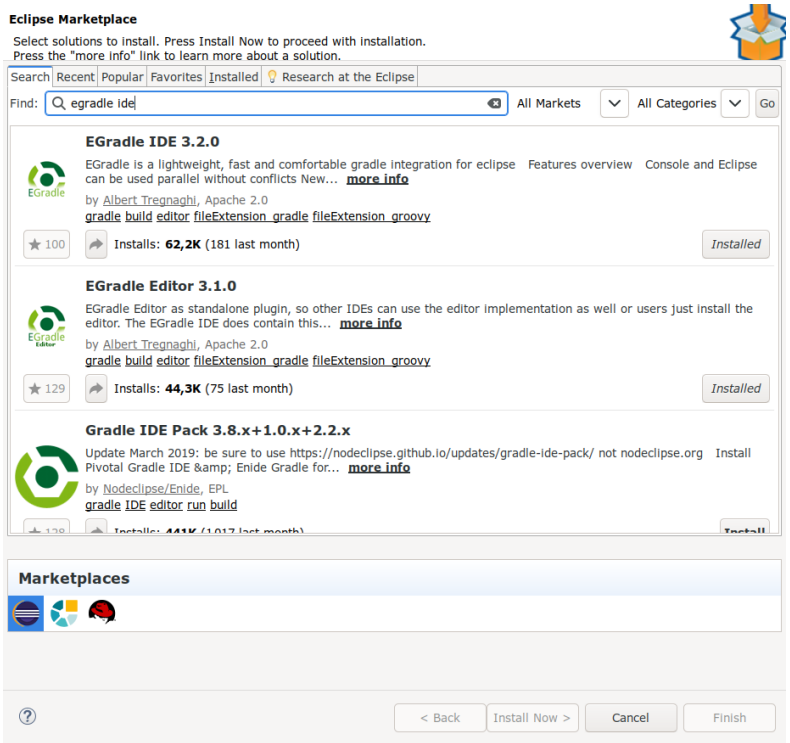
Depuis le marketplace sous Eclipse :



À propos d'Eclipse

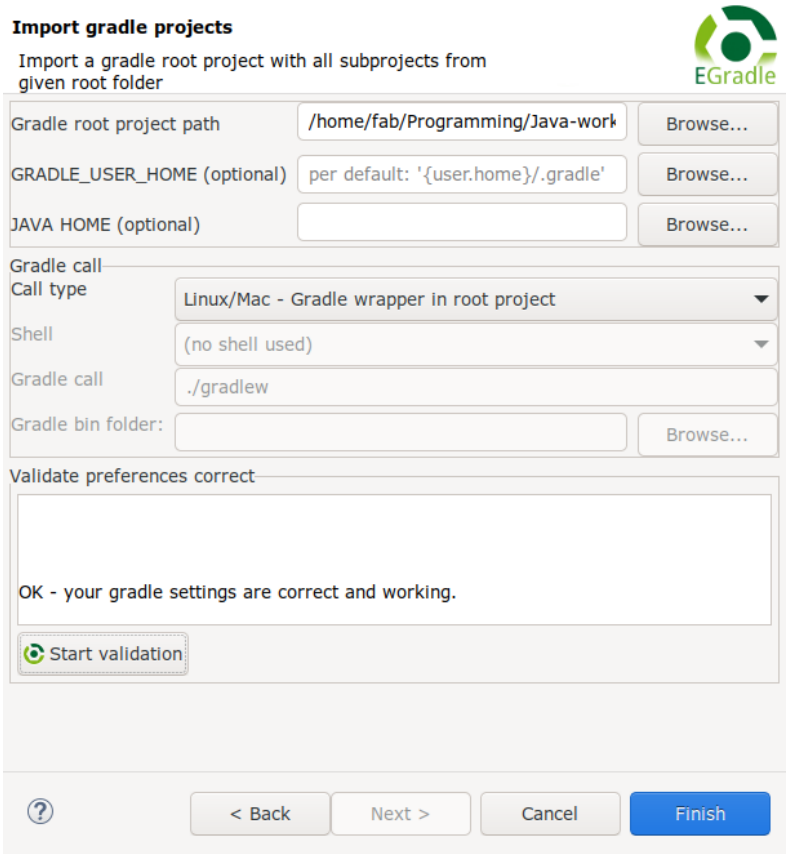
Pour importer des projets Gradle proprement :

1. Ajoutez le plugin `'eclipse'` au fichier `build.gradle` de votre projet
2. Ajoutez le plugin `'EGradle IDE'` à Eclipse



EGradle

Puis faire, file -> import

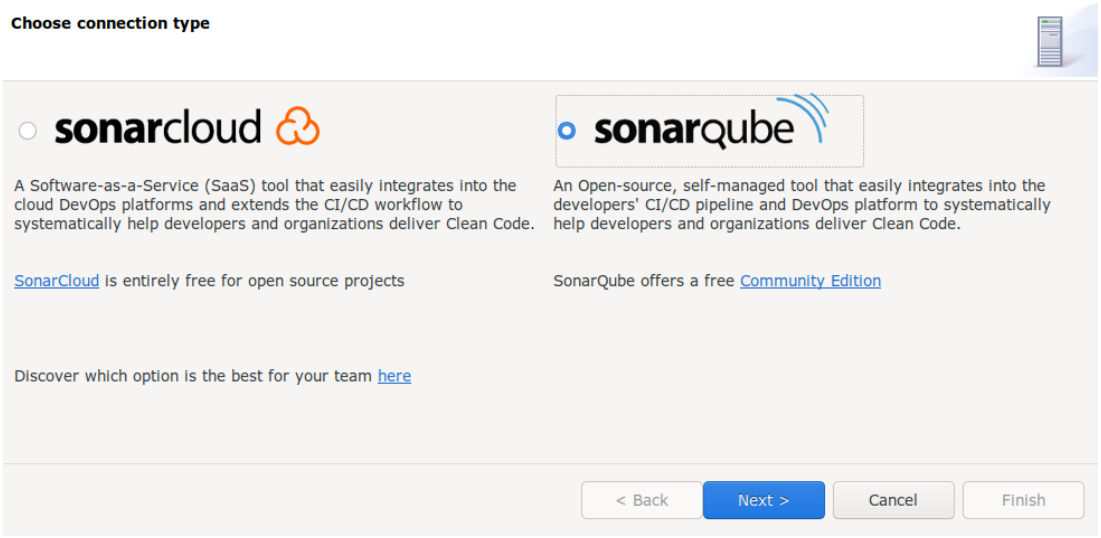


Sélectionnez ensuite le répertoire du projet et vérifiez le avec *Start validation*

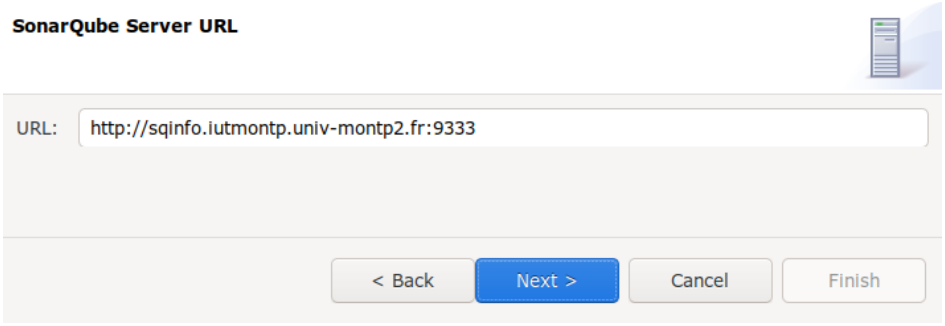
Une fois SonarLint installé

On peut lier SonarLint à un serveur SonarQube, pour recevoir des notifications dans l'IDE ▶ Connected mode

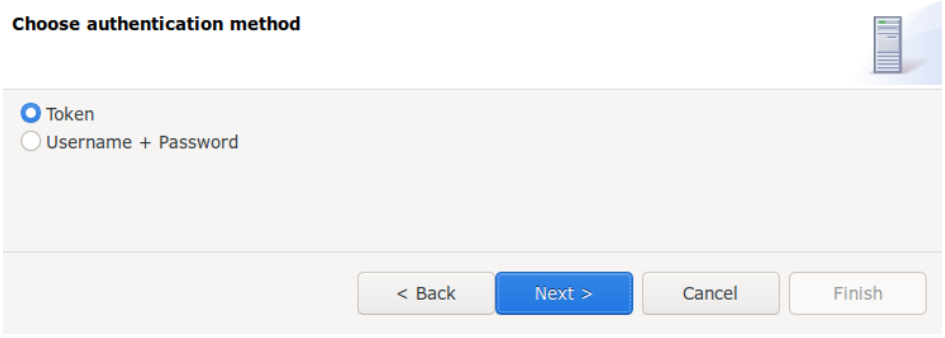
Dans le menu Sonarlint:



Bind du projet sur le SonarQube

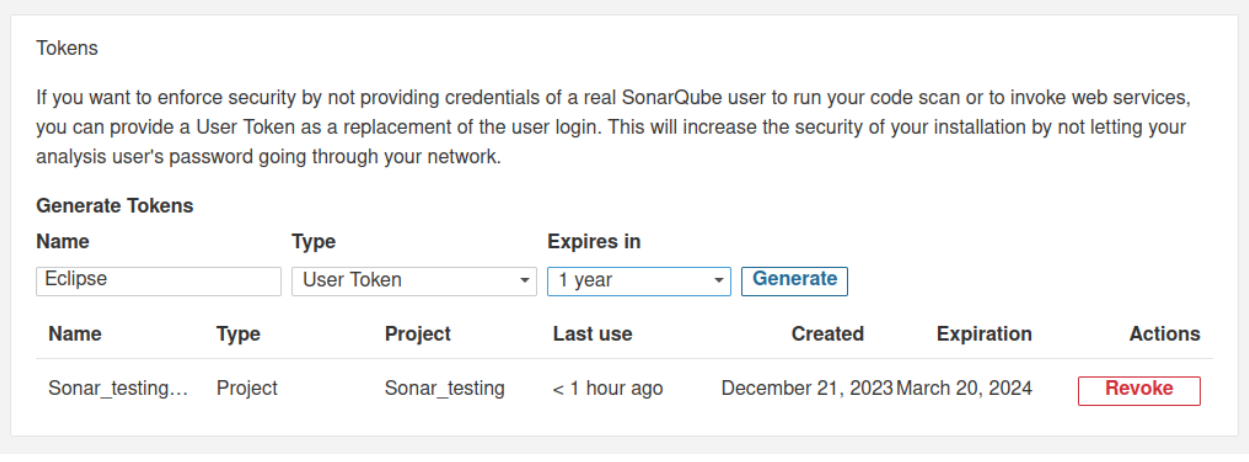


⇒

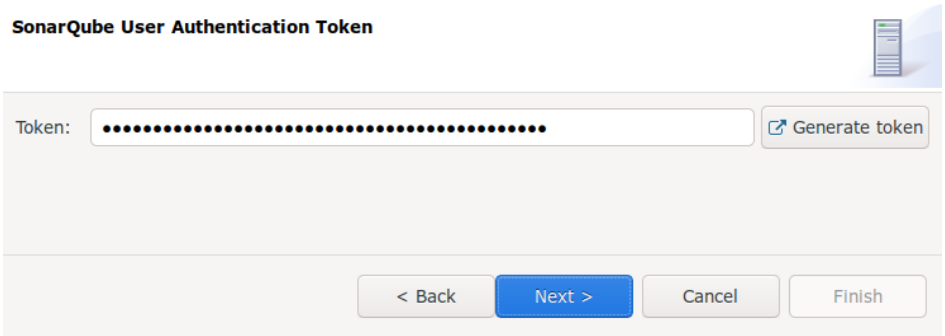


Génération du token sous SonarQube

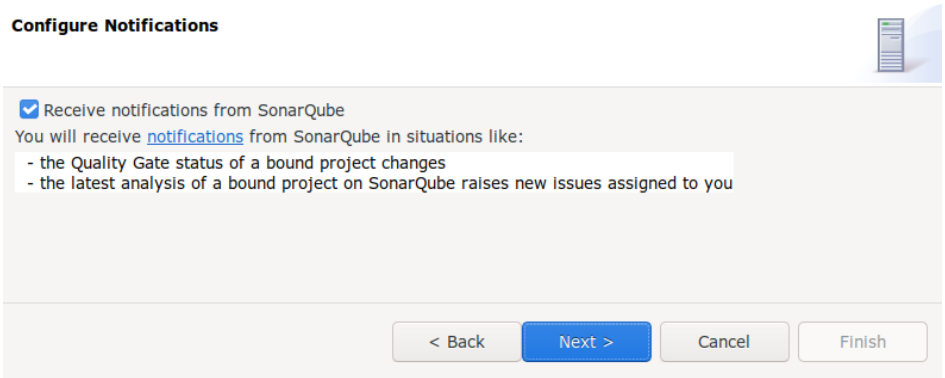
Dans votre profil -> Onglet Security:



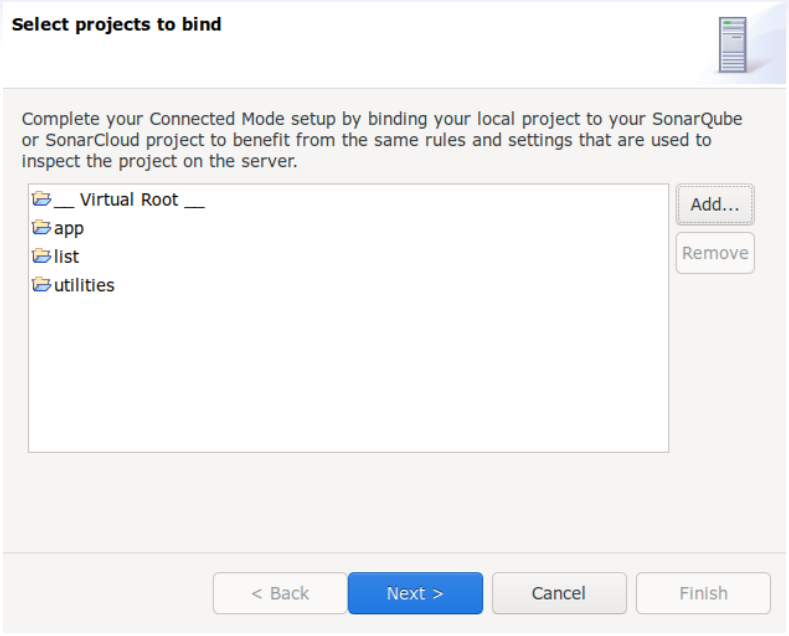
Bind du projet sur le SonarQube



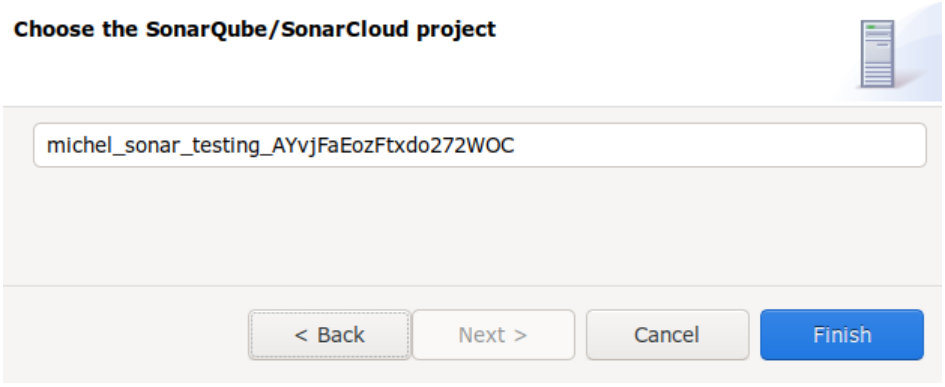
⇒



Bind du projet sur le SonarQube



⇒ Sélection du projet enregistré dans SonarQube



Conclusion

SonarQube

- Principe CaYC : Clean as You Code
- Permet de s'assurer du niveau de qualité du code : Quality Gates
- Peut être intégré dans un piepeline de CI
- Peut inclure la couverture des tests

SonarLint

- Plugin pour les IDE
- Peut être connecté à une instance de SonarQube
- Un must have !