



## ÉCOLE MOHAMMADIA D'INGÉNIEURS

CI/CD — DEVOPS

SONARCLOUD + MAVEN/GITLAB CI (AVEC PIPELINE  
MULTI-PROJETS)

---

# Atelier 4 — Analyse Qualité (SonarCloud) & Intégration GitLab CI/CD

---

*Élèves :*

Sami FAOUZI

*Encadré par :*

Pr. ASMAA RETBI

2ème année Génie Informatique

14 décembre 2025

## Table des matières

<b>1</b>	<b>SonarCloud — Rappel (Diapo 55)</b>	<b>3</b>
<b>2</b>	<b>Pré-requis (optionnel) — Plugin SonarQube (capture)</b>	<b>3</b>
<b>3</b>	<b>Création du compte SonarCloud (depuis GitLab) — Diapo 57</b>	<b>3</b>
3.1	Connexion via GitLab . . . . .	3
3.2	Tableau de bord après connexion . . . . .	4
<b>4</b>	<b>Organisation SonarCloud (depuis un groupe GitLab) — Diapo 59</b>	<b>5</b>
4.1	Création du Personal Access Token GitLab (scope api) . . . . .	5
4.2	Création de l'organisation (Group ID + token) . . . . .	6
4.3	Organisation créée et connectée . . . . .	6
<b>5</b>	<b>Importer le projet Spring Boot existant — Diapo 60</b>	<b>7</b>
5.1	Liste des projets GitLab visibles dans SonarCloud . . . . .	7
5.2	Projet Spring Boot sélectionné . . . . .	7
<b>6</b>	<b>Configuration SonarCloud (Maven/GitLab CI) — Diapos 61 &amp; 62</b>	<b>8</b>
6.1	Variables CI/CD à ajouter dans GitLab . . . . .	8
6.2	Bloc YAML proposé par l'assistant SonarCloud . . . . .	10
<b>7</b>	<b>Adapter le fichier <code>.gitlab-ci.yml</code> — Diapo 63</b>	<b>10</b>
7.1	Fichier <code>.gitlab-ci.yml</code> (version utilisée) . . . . .	10
7.2	Capture du fichier modifié dans GitLab . . . . .	13
<b>8</b>	<b>Modification du code — Projet Spring Boot (Diapo 58)</b>	<b>14</b>
8.1	Classe Java volontairement “sale” (code smells) . . . . .	14
<b>9</b>	<b>Lancer l'analyse SonarCloud — Diapo 64</b>	<b>15</b>
9.1	Pipeline GitLab incluant le stage Sonar . . . . .	15
<b>10</b>	<b>Jenkins &amp; SonarCloud — Intégration dans Jenkins (Diapos 65 → 69)</b>	<b>15</b>
10.1	Étape 7 — Installation du plugin SonarQube dans Jenkins . . . . .	15
10.2	Étape 8 — Configurer SonarCloud dans Jenkins . . . . .	16
10.2.1	Création du credential SonarCloud (Secret text) . . . . .	16
10.2.2	Déclaration du serveur SonarCloud (SonarQube servers) . . . . .	16
10.3	Étape 9 — Ajouter un stage SonarCloud dans le Jenkinsfile . . . . .	17
10.4	Étape 10 — Lancer le pipeline Jenkins . . . . .	19
10.4.1	Log du stage SonarCloud (analyse réussie) . . . . .	20
10.5	Étape 11 — Analyse des résultats (SonarCloud) . . . . .	21
10.6	Comparaison synthétique : GitLab CI vs Jenkins (même analyse SonarCloud)	21

<b>11 Annexe — Rappel variables GitLab attendues</b>	<b>22</b>
--	-----------

## 1 SonarCloud — Rappel (Diapo 55)

SonarCloud analyse automatiquement la **qualité** et la **sécurité** du code :

- Bugs / Vulnérabilités
- Code Smells / Duplications
- Couverture de tests / Complexité

## 2 Pré-requis (optionnel) — Plugin SonarQube (capture)

Capture demandée dans certains supports : plugin SonarQube sélectionné.



FIGURE 1 – Plugin SonarQube (pré-requis selon configuration Jenkins).

## 3 Création du compte SonarCloud (depuis GitLab) — Diapo 57

### 3.1 Connexion via GitLab

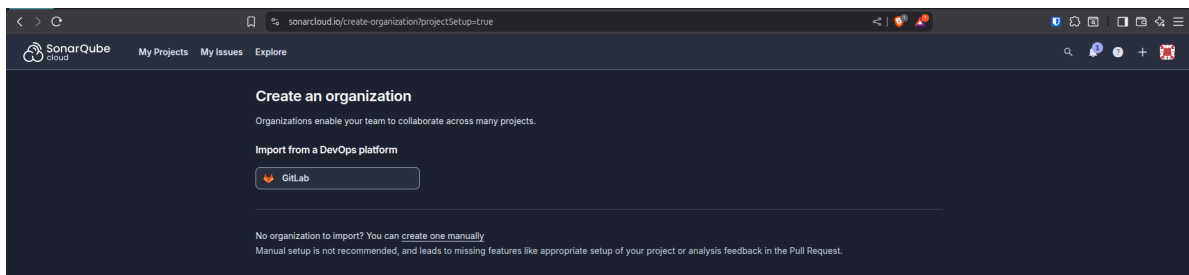


FIGURE 2 – Page de connexion SonarCloud avec l'option GitLab.

## 3.2 Tableau de bord après connexion

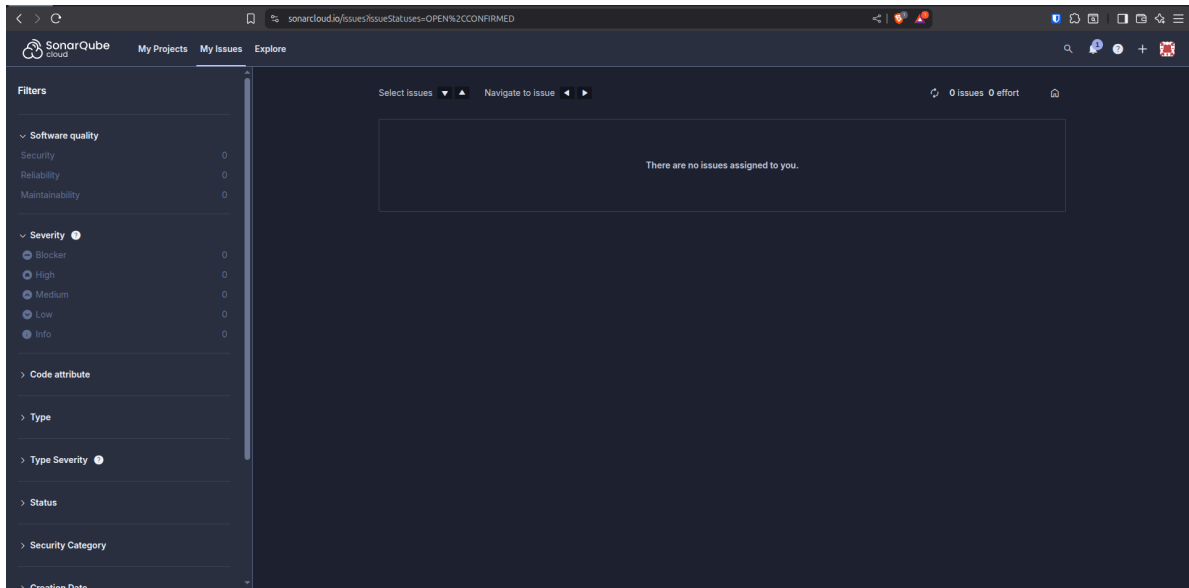
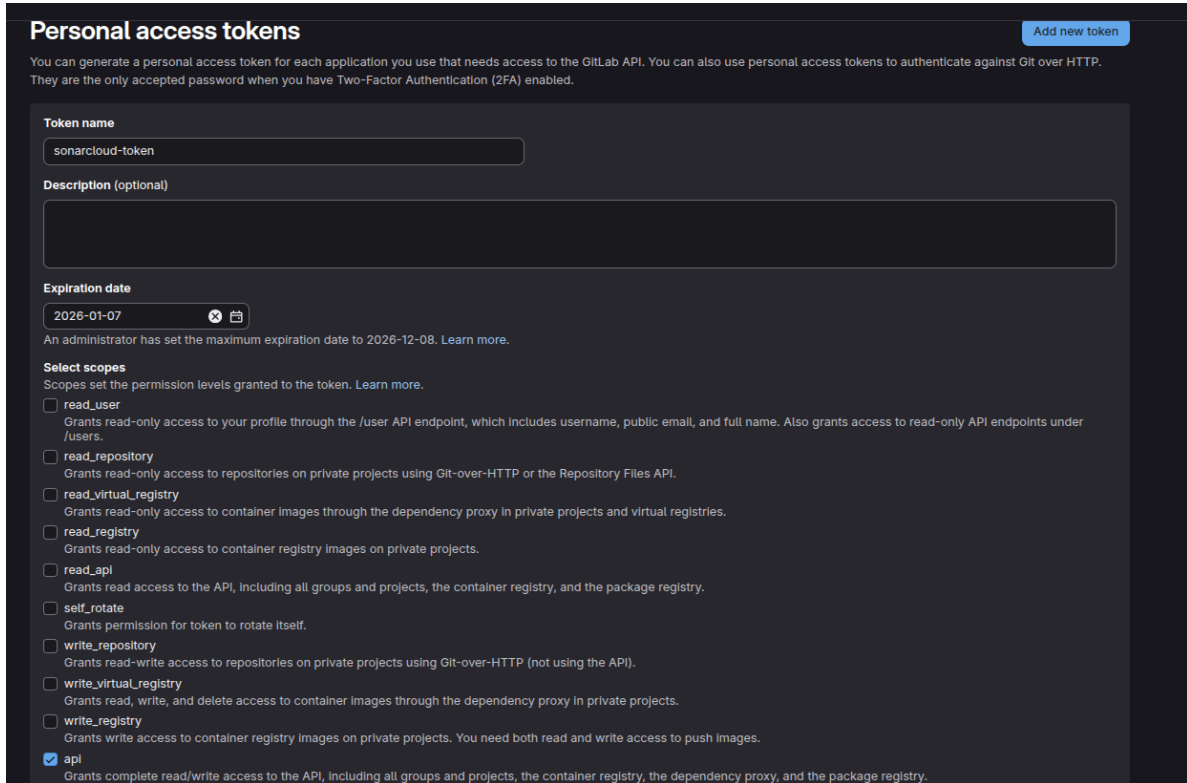


FIGURE 3 – Tableau de bord SonarCloud (après connexion).

## 4 Organisation SonarCloud (depuis un groupe GitLab) — Diapo 59

### 4.1 Création du Personal Access Token GitLab (scope api)



**Personal access tokens** Add new token

You can generate a personal access token for each application you use that needs access to the GitLab API. You can also use personal access tokens to authenticate against Git over HTTP. They are the only accepted password when you have Two-Factor Authentication (2FA) enabled.

**Token name**  
sonarcloud-token

**Description (optional)**

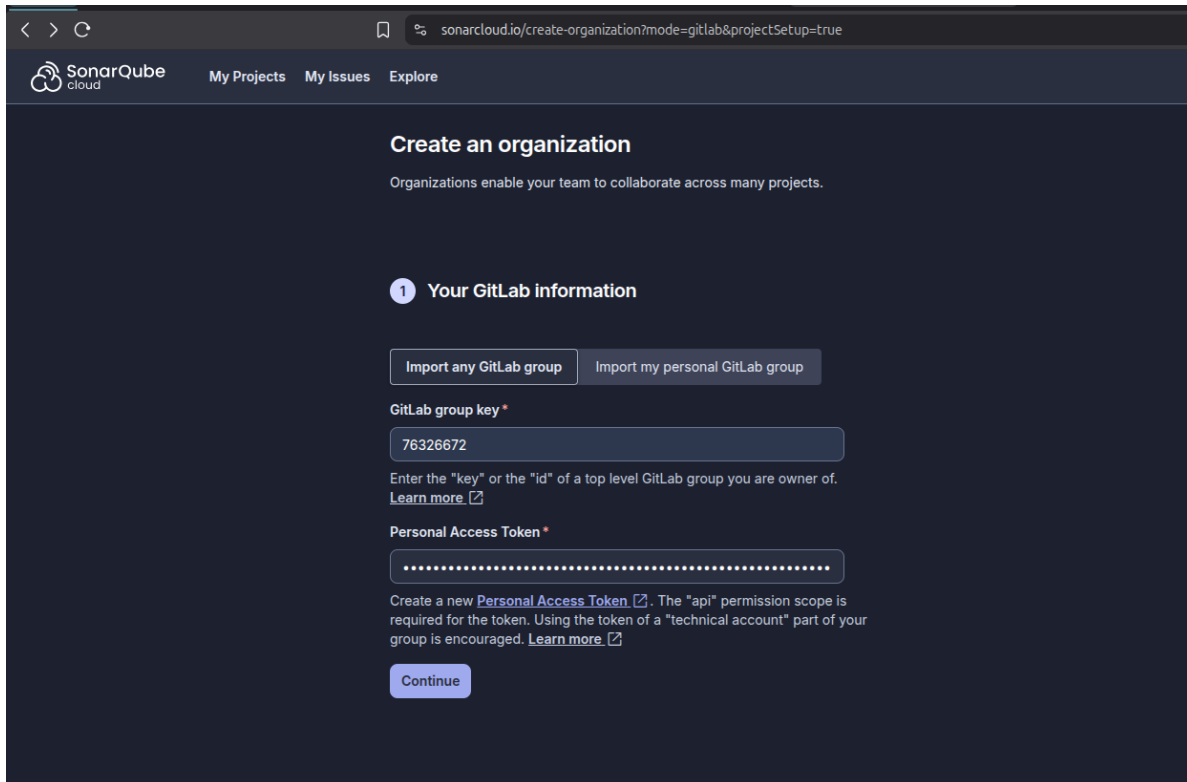
**Expiration date**  
2026-01-07 ✕ 📅  
An administrator has set the maximum expiration date to 2026-12-08. [Learn more.](#)

**Select scopes**  
Scopes set the permission levels granted to the token. [Learn more.](#)

- ☐ **read\_user**  
Grants read-only access to your profile through the /user API endpoint, which includes username, public email, and full name. Also grants access to read-only API endpoints under /users.
- ☐ **read\_repository**  
Grants read-only access to repositories on private projects using Git-over-HTTP or the Repository Files API.
- ☐ **read\_virtual\_registry**  
Grants read-only access to container images through the dependency proxy in private projects and virtual registries.
- ☐ **read\_registry**  
Grants read-only access to container registry images on private projects.
- ☐ **read\_api**  
Grants read access to the API, including all groups and projects, the container registry, and the package registry.
- ☐ **self\_rotate**  
Grants permission for token to rotate itself.
- ☐ **write\_repository**  
Grants read-write access to repositories on private projects using Git-over-HTTP (not using the API).
- ☐ **write\_virtual\_registry**  
Grants read, write, and delete access to container images through the dependency proxy in private projects.
- ☐ **write\_registry**  
Grants write access to container registry images on private projects. You need both read and write access to push images.
- ☒ **api**  
Grants complete read/write access to the API, including all groups and projects, the container registry, the dependency proxy, and the package registry.

FIGURE 4 – Création du token GitLab (scope api) utilisé pour l'organisation SonarCloud.

## 4.2 Création de l'organisation (Group ID + token)



sonarcloud.io/create-organization?mode=gitlab&projectSetup=true

SonarQube cloud My Projects My Issues Explore

### Create an organization

Organizations enable your team to collaborate across many projects.

1 Your GitLab information

Import any GitLab group Import my personal GitLab group

GitLab group key \*

76326672

Enter the "key" or the "id" of a top level GitLab group you are owner of. [Learn more](#)

Personal Access Token \*

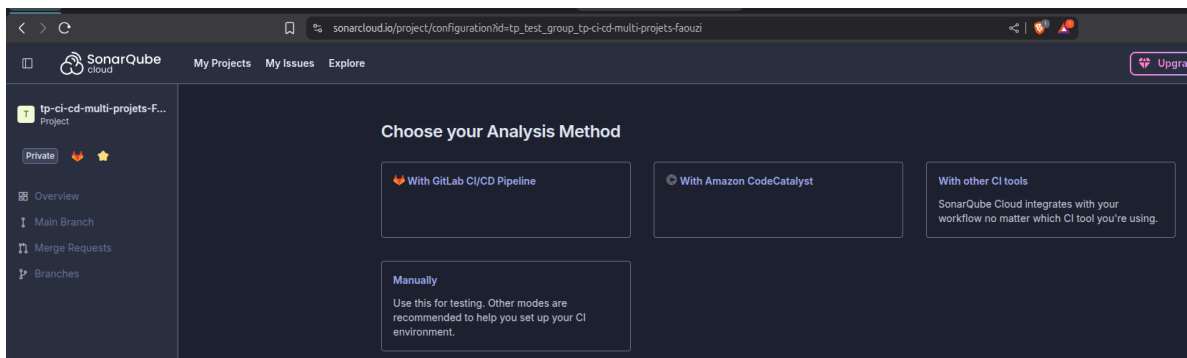
.....

Create a new [Personal Access Token](#). The "api" permission scope is required for the token. Using the token of a "technical account" part of your group is encouraged. [Learn more](#)

Continue

FIGURE 5 – Création de l'organisation SonarCloud liée au groupe GitLab.

## 4.3 Organisation créée et connectée



sonarcloud.io/project/configuration?id=tp\_test\_group\_tp-ci-cd-multi-projets-faouzi

SonarQube cloud My Projects My Issues Explore Upgrade

tp-ci-cd-multi-projets-F... Project

Private

Overview Main Branch Merge Requests Branches

### Choose your Analysis Method

With GitLab CI/CD Pipeline

With Amazon CodeCatalyst

With other CI tools  
SonarQube Cloud integrates with your workflow no matter which CI tool you're using.

Manually  
Use this for testing. Other modes are recommended to help you set up your CI environment.

FIGURE 6 – Organisation SonarCloud créée et reliée à GitLab.

## 5 Importer le projet Spring Boot existant — Diapo 60

### 5.1 Liste des projets GitLab visibles dans SonarCloud

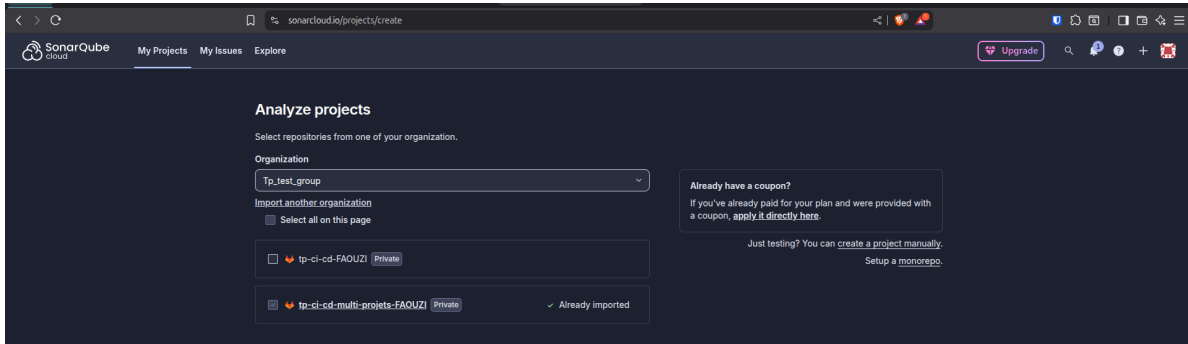


FIGURE 7 – Liste des projets GitLab détectés dans SonarCloud.

### 5.2 Projet Spring Boot sélectionné

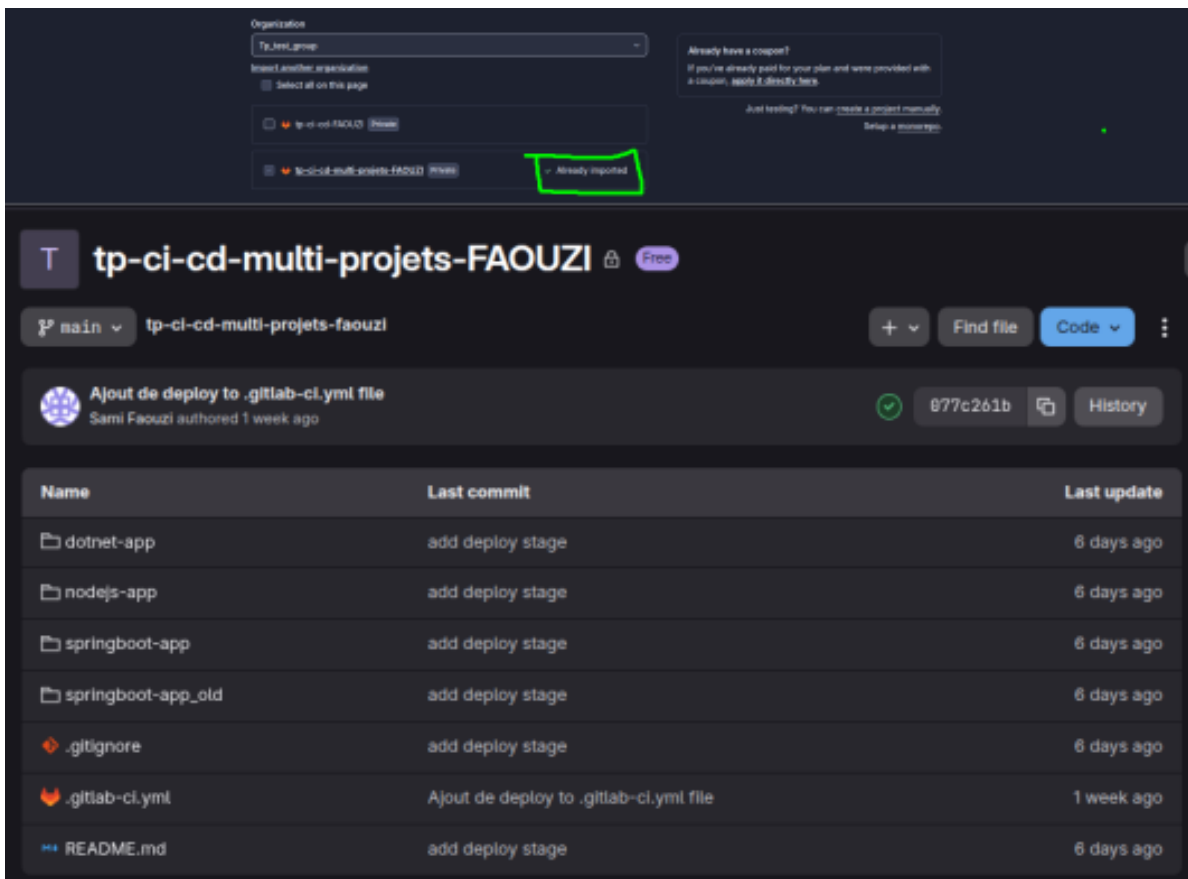


FIGURE 8 – Sélection du projet Spring Boot à analyser dans SonarCloud.



## 6 Configuration SonarCloud (Maven/GitLab CI) — Diapos 61 & 62

### 6.1 Variables CI/CD à ajouter dans GitLab

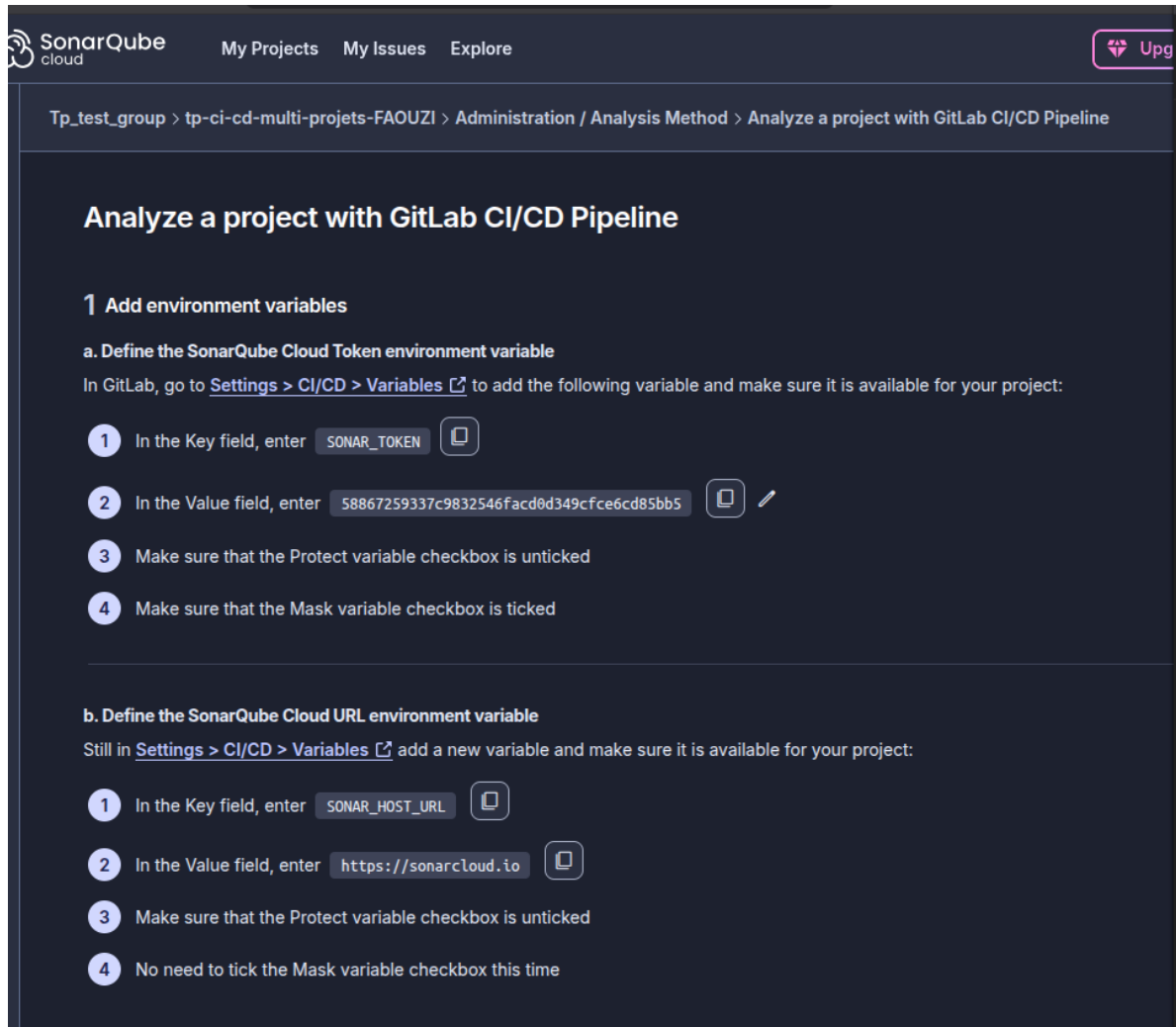
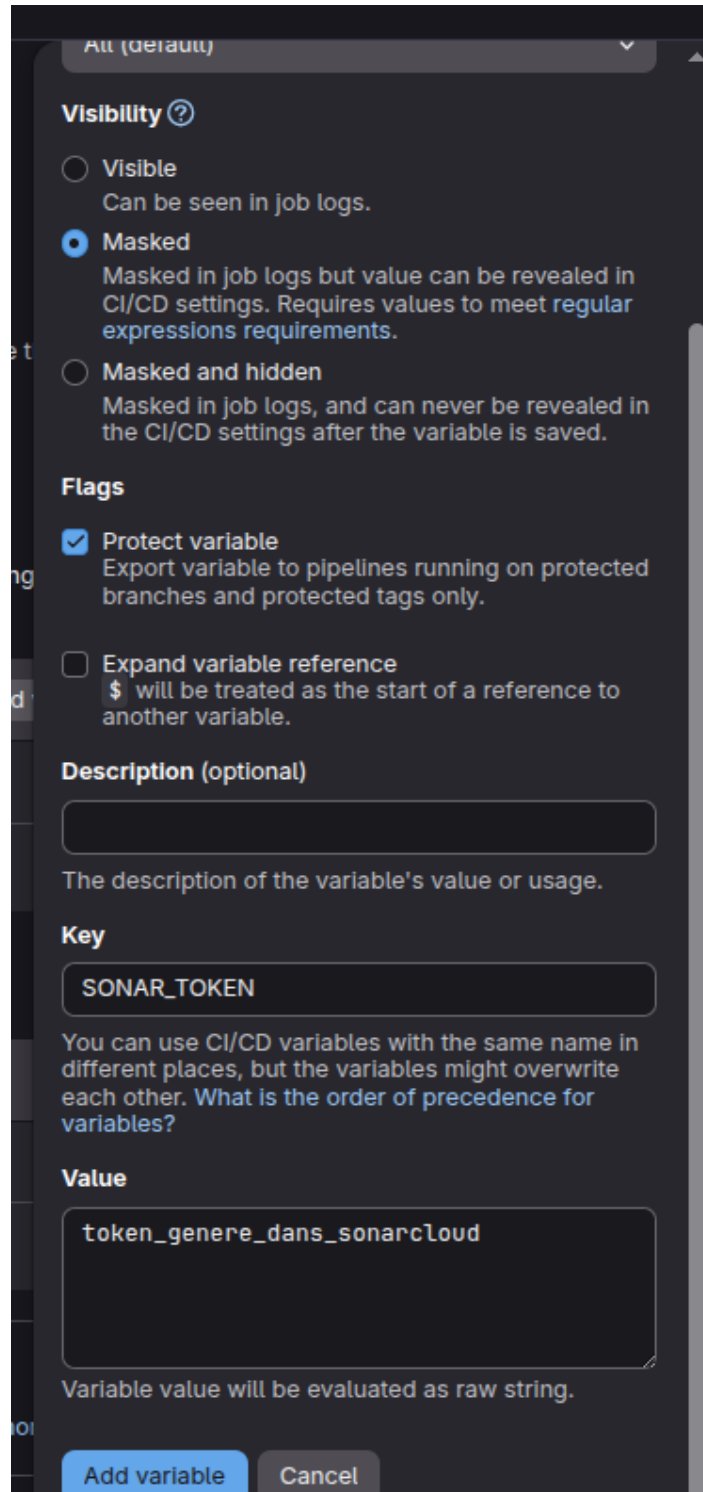


FIGURE 9 – Variables recommandées par SonarCloud : `SONAR_TOKEN`, `SONAR_HOST_URL`, `SONAR_ORGANIZATION`.



Alt (default)

**Visibility ?**

- ☐ Visible  
Can be seen in job logs.
- ☒ Masked  
Masked in job logs but value can be revealed in CI/CD settings. Requires values to meet **regular** expressions requirements.
- ☐ Masked and hidden  
Masked in job logs, and can never be revealed in the CI/CD settings after the variable is saved.

**Flags**

- ☒ Protect variable  
Export variable to pipelines running on protected branches and protected tags only.
- ☐ Expand variable reference  
\$ will be treated as the start of a reference to another variable.

**Description (optional)**

The description of the variable's value or usage.

**Key**

SONAR\_TOKEN

You can use CI/CD variables with the same name in different places, but the variables might overwrite each other. What is the order of precedence for variables?

**Value**

token\_genere\_dans\_sonarcloud

Variable value will be evaluated as raw string.

Add variable Cancel

FIGURE 10 – Configuration des variables CI/CD dans GitLab (SONAR\_TOKEN, SONAR\_HOST\_URL, SONAR\_ORGANIZATION).

## 6.2 Bloc YAML proposé par l'assistant SonarCloud

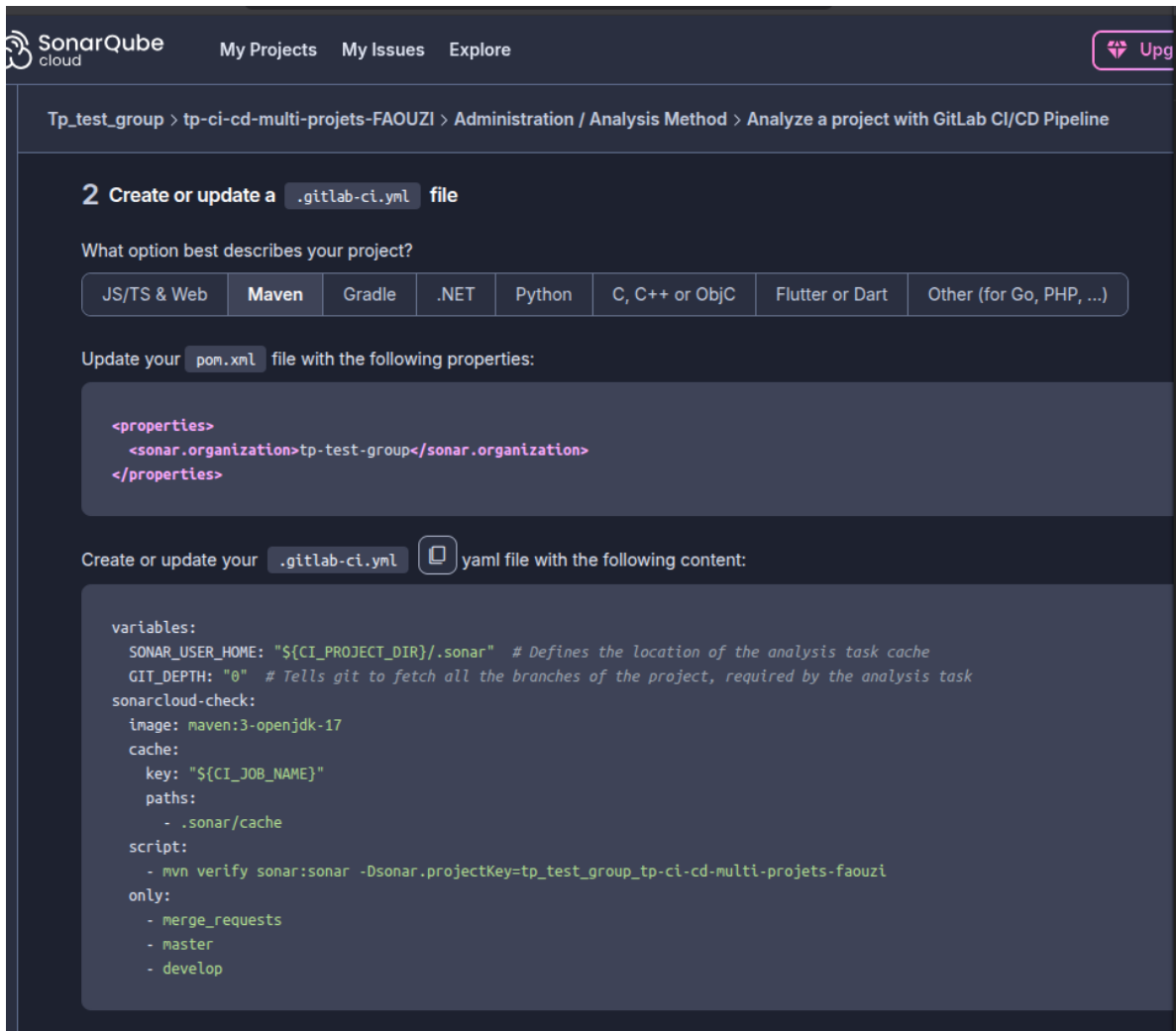


FIGURE 11 – Bloc YAML proposé par SonarCloud (Maven/GitLab CI).

## 7 Adapter le fichier `.gitlab-ci.yml` — Diapo 63

### 7.1 Fichier `.gitlab-ci.yml` (version utilisée)

Le fichier suivant correspond au format demandé (stages build/test/sonar/deploy) et évite l'erreur Maven `No plugin found for prefix '-Dsonar.host.url=...'` en utilisant un bloc YAML plié >.

Listing 1 – Fichier `.gitlab-ci.yml` (pipeline multi-projets + SonarCloud + déploiement Tomcat)

```
stages:
  - build
  - test
  - sonar
  - deploy
```

```
# ----- JOBS DE BUILD -----

build-springboot:
  stage: build
  image: maven:3.9-eclipse-temurin-17
  script:
    - echo "====_Dbut_du_build_Spring_Boot_===="
    - cd springboot-app
    - mvn -B clean package -DskipTests
    - echo "====_Fin_du_build_Spring_Boot_===="
  artifacts:
    paths:
      - springboot-app/target/*.war
    expire_in: 1 week

build-nodejs:
  stage: build
  image: node:20-alpine
  script:
    - echo "====_Dbut_du_build_Node.js_===="
    - cd nodejs-app
    - npm install
    - npm run build
    - echo "====_Fin_du_build_Node.js_===="

build-dotnet:
  stage: build
  image: mcr.microsoft.com/dotnet/sdk:8.0
  script:
    - echo "====_Dbut_du_build_.NET_===="
    - cd dotnet-app/demoapp
    - dotnet restore
    - dotnet build --configuration Release
    - echo "====_Fin_du_build_.NET_===="

# ----- JOBS DE TEST -----

test-springboot:
  stage: test
  image: maven:3.9.9-eclipse-temurin-21-alpine
  script:
    - echo "-----_Dbut_des_tests_Spring_Boot_-----"
```

```

- cd springboot-app
- mvn test
- echo "-----Fin des tests Spring Boot-----"

test-nodejs:
  stage: test
  image: node:20-alpine
  script:
    - echo "-----Dbut des tests Node.js-----"
    - cd nodejs-app
    - npm test
    - echo "-----Fin des tests Node.js-----"

test-dotnet:
  stage: test
  image: mcr.microsoft.com/dotnet/sdk:8.0
  script:
    - echo "-----Dbut des tests .NET-----"
    - cd dotnet-app/demoapp
    - dotnet test
    - echo "-----Fin des tests .NET-----"

# ----- JOB SONARCLOUD (MAVEN) -----

variables:
  SONAR_USER_HOME: "${CI_PROJECT_DIR}/.sonar"
  GIT_DEPTH: "0"

sonarcloud-check:
  stage: sonar
  image: maven:3.9.9-eclipse-temurin-17
  cache:
    key: "${CI_JOB_NAME}"
    paths:
      - .sonar/cache
  script:
    - echo "====Lancement de l'analyse SonarCloud===="
    - >
    mvn -f springboot-app/pom.xml -B
    -DskipTests
    -Dsonar.projectKey=tp_test_group_tp-ci-cd-multi-projets-faouzi
    -Dsonar.organization=$SONAR_ORGANIZATION
    -Dsonar.host.url=$SONAR_HOST_URL

```

```

-Dsonar.token=$SONAR_TOKEN
clean verify sonar:sonar
- echo "====_Fin_de_l'analyse_SonarCloud_===="
only:
- main

# ----- DPLOIEMENT TOMCAT -----

deploy-tomcat:
  stage: deploy
  needs:
    - job: build-springboot
  artifacts: true
  script:
    - echo "====_Dbut_Deploy_Tomcat_===="
    - ls -lh springboot-app/target/*.war
    - sudo cp springboot-app/target/*.war /var/lib/tomcat10/webapps/springboot-
      ci.war
    - sudo systemctl restart tomcat10
    - echo "====_Fin_Deploy_Tomcat_===="
  only:
    - main
  tags:
    - local-shell

```

## 7.2 Capture du fichier modifié dans GitLab

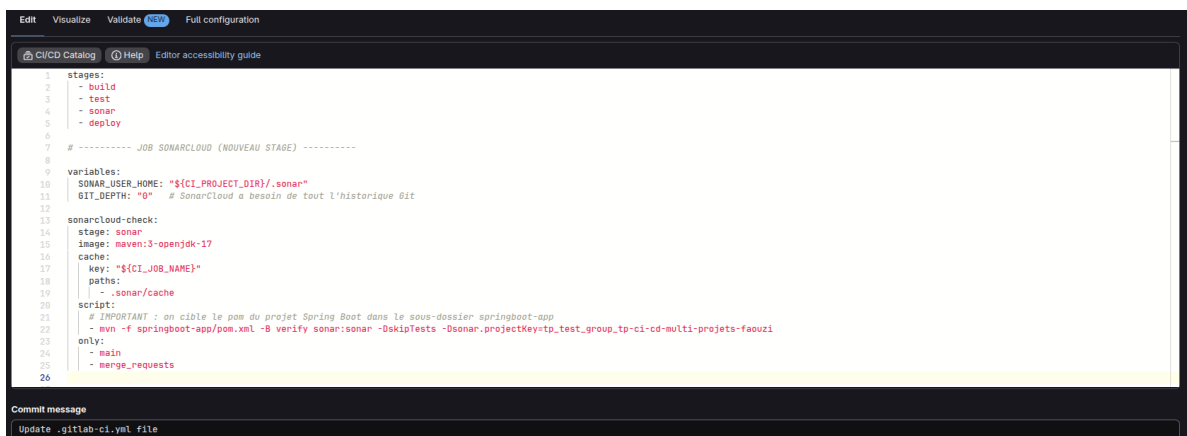


FIGURE 12 – Capture du fichier `.gitlab-ci.yml` modifié (GitLab).

## 8 Modification du code — Projet Spring Boot (Diapo 58)

### 8.1 Classe Java volontairement “sale” (code smells)

Création d’une classe avec :

- un champ non utilisé;
- une duplication volontaire;
- aucun test unitaire associé à cette classe.

Listing 2 – BadQualityService.java (champ non utilisé + duplication)

```
package com.test;

// Classe volontairement "sale" pour SonarCloud
public class BadQualityService {

    // Champ non utilis
    private String unusedField;

    public int computeScoreV1(int a, int b) {
        int sum = a + b;
        if (sum > 10) {
            return sum;
        } else {
            return sum + 10;
        }
    }

    // Mthode quasi-identique => duplication volontaire
    public int computeScoreV2(int a, int b) {
        int sum = a + b;
        if (sum > 10) {
            return sum;
        } else {
            return sum + 10;
        }
    }
}
```

## 9 Lancer l'analyse SonarCloud — Diapo 64

### 9.1 Pipeline GitLab incluant le stage Sonar

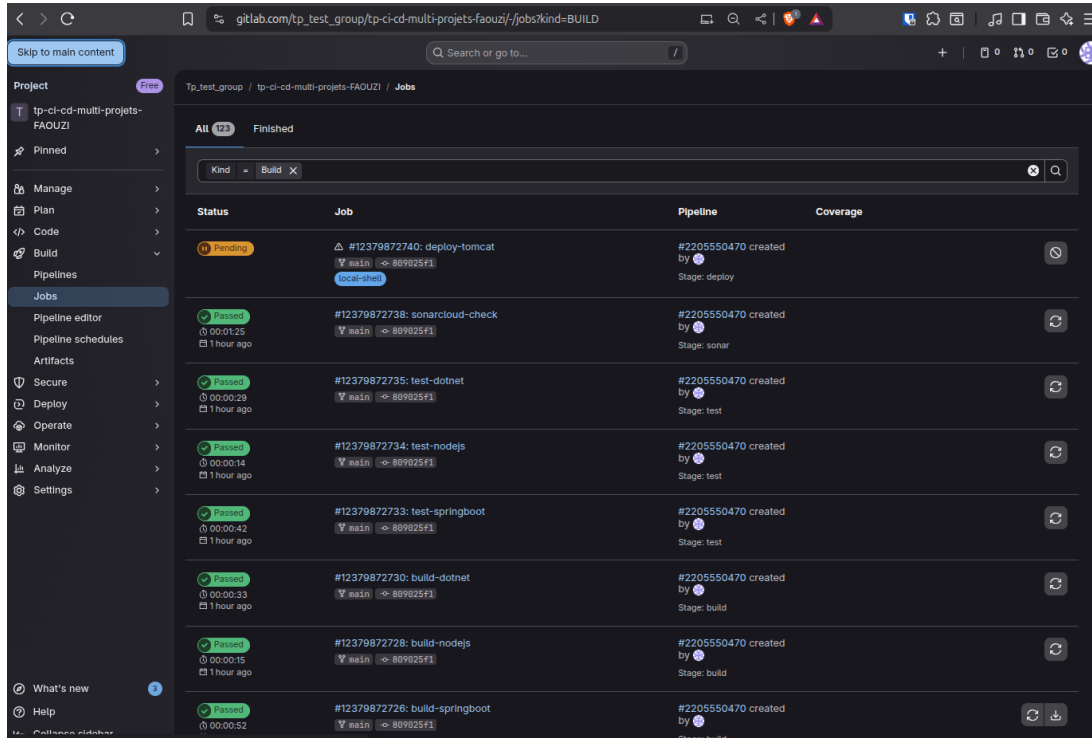


FIGURE 13 – Connexion SonarCloud réussie via GitLab.

## 10 Jenkins & SonarCloud — Intégration dans Jenkins (Diapos 65 → 69)

### 10.1 Étape 7 — Installation du plugin SonarQube dans Jenkins

Installation du plugin **SonarQube Scanner for Jenkins** via : [Manage Jenkins](#) → [Plugins](#) → recherche puis installation.



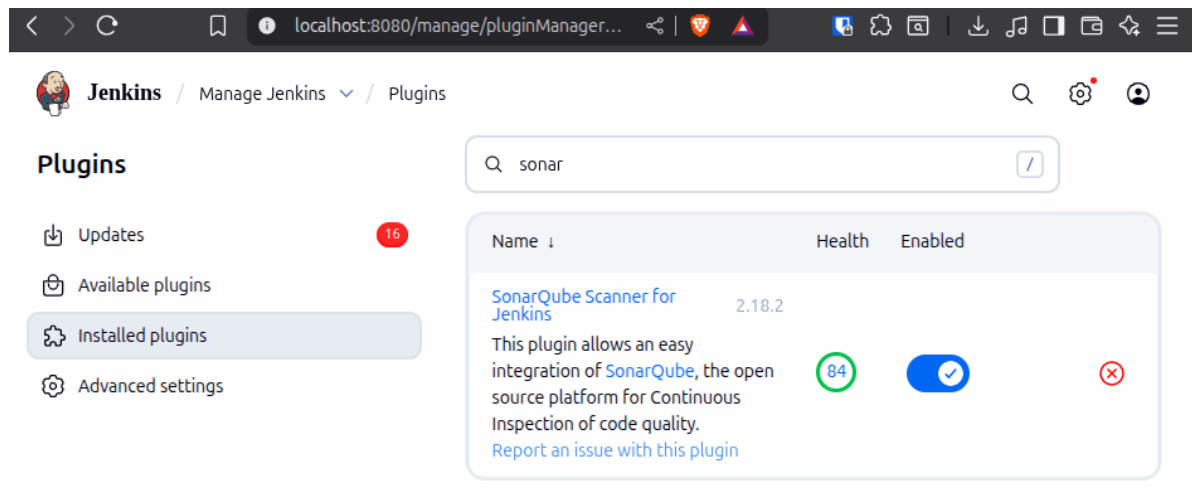


FIGURE 14 – Installation du plugin SonarQube Scanner dans Jenkins (installation réussie).

## 10.2 Étape 8 — Configurer SonarCloud dans Jenkins

### 10.2.1 Création du credential SonarCloud (Secret text)

Ajout du token SonarCloud en tant que credential **Secret text** dans : [Manage Jenkins](#) → [Credentials](#) → [Global](#) → [Add Credentials](#).

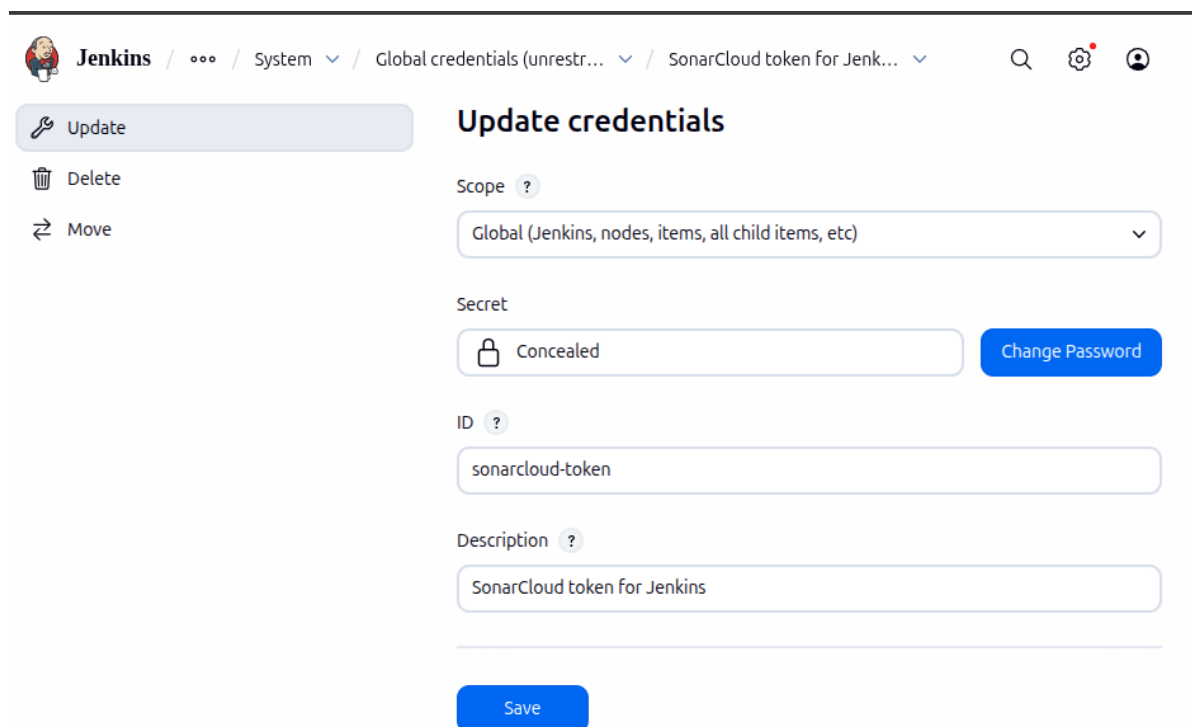


FIGURE 15 – Credential SonarCloud (Secret text) créé dans Jenkins.

### 10.2.2 Déclaration du serveur SonarCloud (SonarQube servers)

Configuration dans : [Manage Jenkins](#) → [Configure System](#) → [SonarQube servers](#).

## 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

SonarCloudServer

Server URL

Default is http://localhost:9000

https://sonarcloud.io

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled.

SonarCloud token for Jenkins

+

Add

Advanced

▼

FIGURE 16 – Section *SonarQube servers* remplie : Name = [SonarCloudServer](#), URL = <https://sonarcloud.io>, token associé.

## 10.3 Étape 9 — Ajouter un stage SonarCloud dans le Jenkinsfile

Le Jenkinsfile local a été modifié pour ajouter un stage **SonarCloud Analysis**. L'extrait ci-dessous correspond au stage d'analyse (exécuté dans le dossier [springboot-app](#)).

Listing 3 – Extrait du Jenkinsfile : stage SonarCloud Analysis

```

pipeline {
    agent any
    options { timestamps() }

    environment {
        SONAR_INSTANCE = 'SonarCloudServer'
        SONAR_ORG = 'tp-test-group'
        SONAR_PROJECT_KEY = 'tp_test_group_tp-ci-cd-multi-projets-faouzi'
        APP_DIR = 'springboot-app'
    }

    stages {
        stage('Checkout') {
            steps { checkout scm }
        }
    }
}
    
```

```

stage('Build') {
    steps {
        dir(env.APP_DIR) {
            script {
                if (isUnix()) {
                    sh 'chmod +x mvnw || true'
                    sh './mvnw -B -DskipTests clean verify'
                } else {
                    bat 'mvnw.cmd -B -DskipTests clean verify'
                }
            }
        }
    }
}

stage('SonarCloud Analysis') {
    steps {
        dir(env.APP_DIR) {
            withSonarQubeEnv(env.SONAR_INSTANCE) {
                script {
                    if (isUnix()) {
                        sh '''
                            chmod +x mvnw || true
                            ./mvnw -B -DskipTests sonar:sonar \
                                -Dsonar.projectKey=$SONAR_PROJECT_KEY \
                                -Dsonar.organization=$SONAR_ORG \
                                -Dsonar.host.url=https://sonarcloud.io \
                                -Dsonar.token=$SONAR_AUTH_TOKEN
                        '''
                    } else {
                        bat """
                            mvnw.cmd -B -DskipTests sonar:sonar ^
                                -Dsonar.projectKey=%SONAR_PROJECT_KEY% ^
                                -Dsonar.organization=%SONAR_ORG% ^
                                -Dsonar.host.url=https://sonarcloud.io ^
                                -Dsonar.token=%SONAR_AUTH_TOKEN%
                        """
                    }
                }
            }
        }
    }
}

```

```
}
}
}
```

**Remarque :** selon la version du plugin Maven, le log affiche souvent **BUILD SUCCESS** au lieu de **ANALYSIS SUCCESSFUL**. Dans les deux cas, cela indique que l'analyse SonarCloud s'est terminée correctement.

## 10.4 Étape 10 — Lancer le pipeline Jenkins

Lancement via : **Job** → **Build Now**. Les stages observés dans notre exécution apparaissent dans la vue *Stages*.

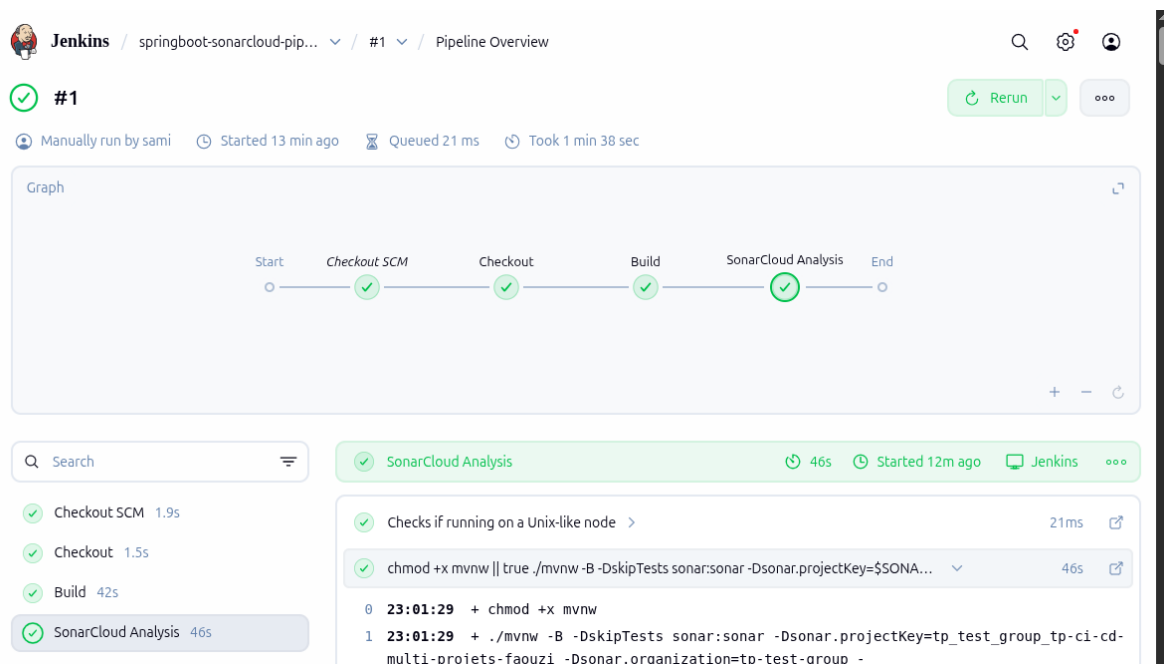


FIGURE 17 – Liste des stages du pipeline Jenkins (Checkout/Build/SonarCloud Analysis).

### 10.4.1 Log du stage SonarCloud (analyse réussie)

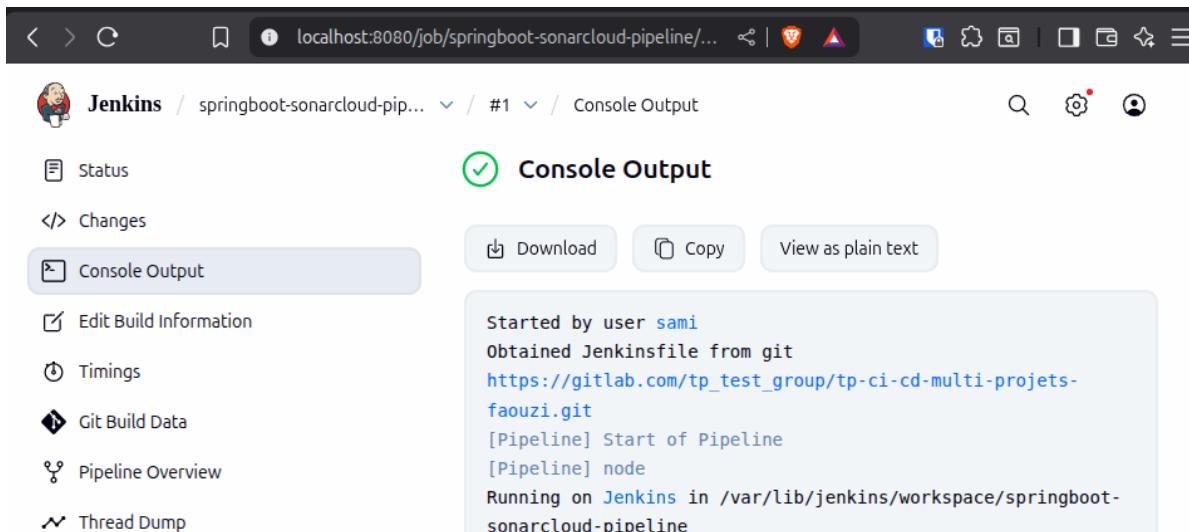


FIGURE 18 – Extrait du log du stage SonarCloud : exécution réussie (*BUILD SUCCESS*).

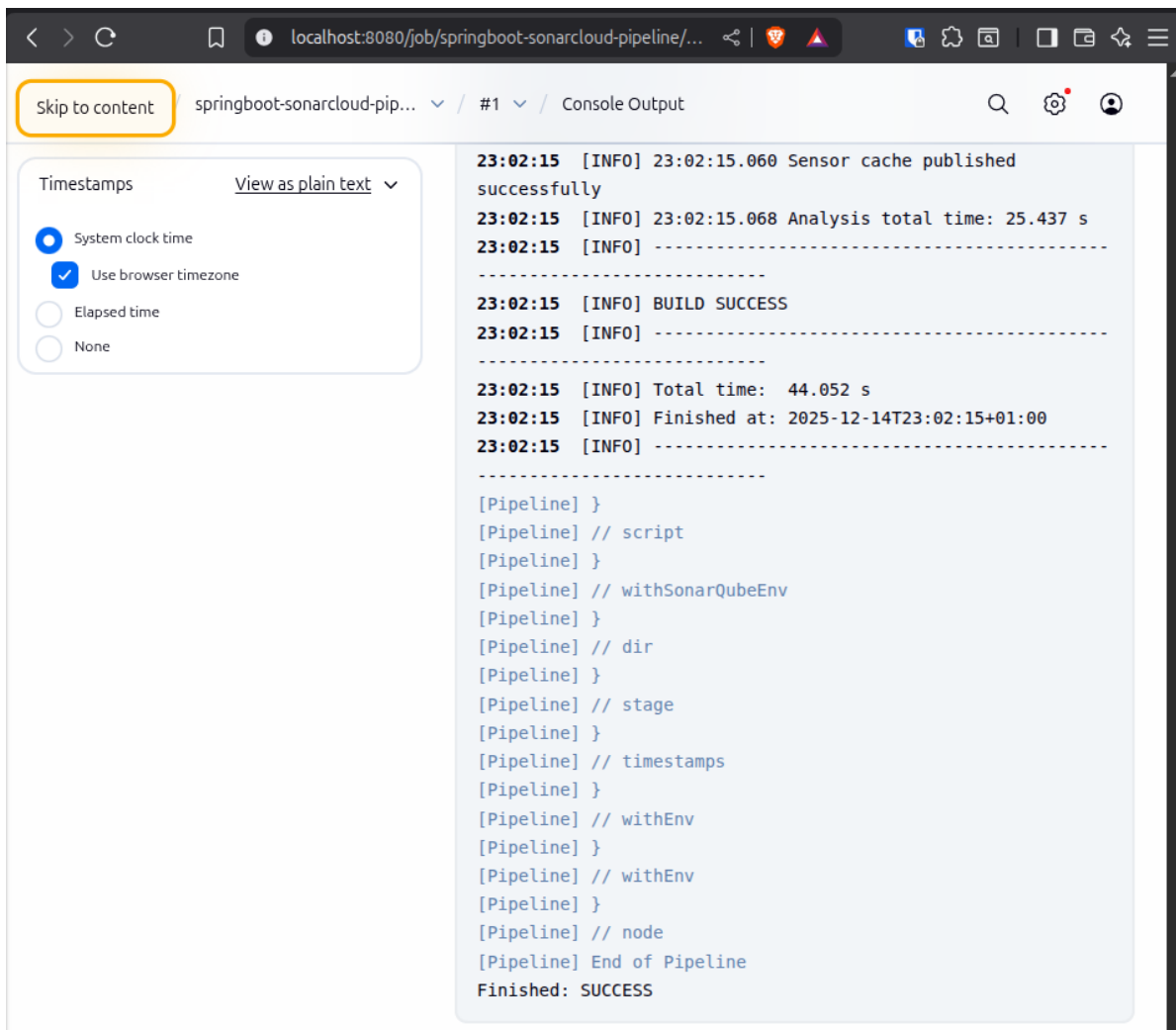


FIGURE 19 – Deuxième extrait du log confirmant la réussite du pipeline.

## 10.5 Étape 11 — Analyse des résultats (SonarCloud)

Après l'exécution du pipeline Jenkins, une nouvelle analyse apparaît sur SonarCloud. Cela confirme que Jenkins a bien envoyé l'analyse vers SonarCloud.

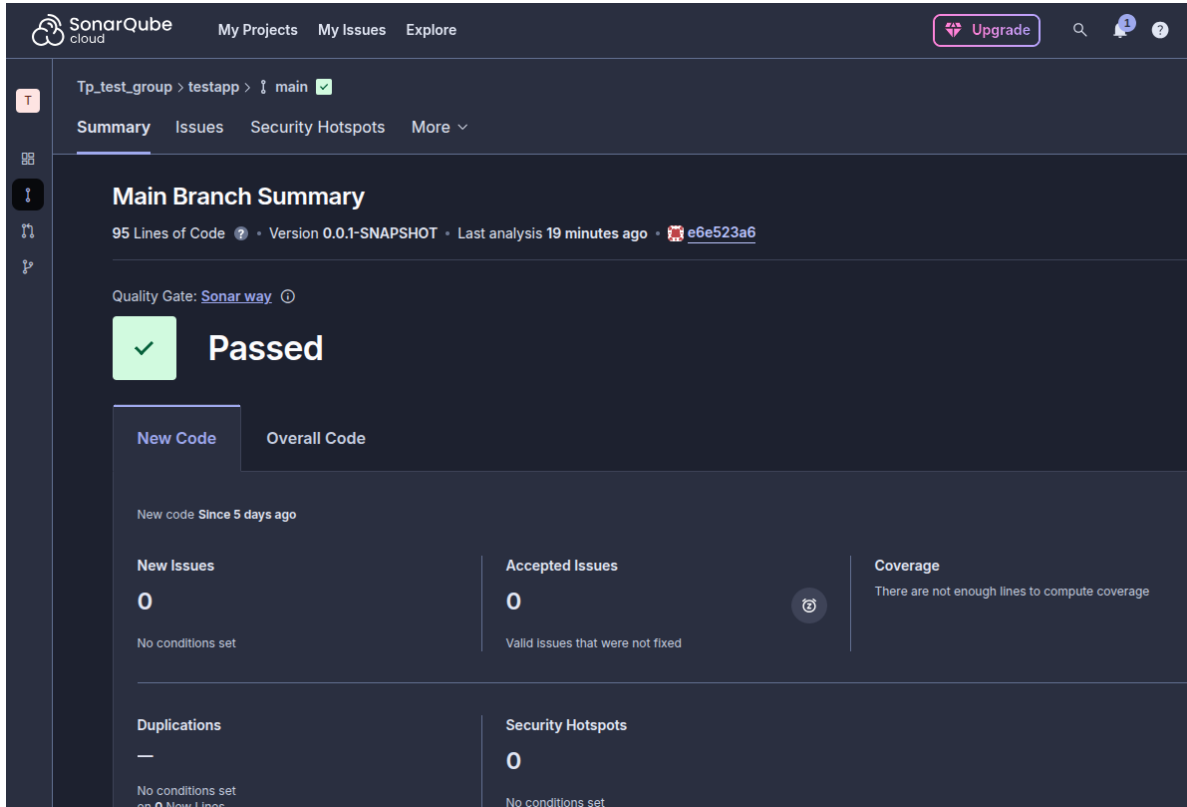


FIGURE 20 – Nouvelle analyse reçue et visible dans SonarCloud après exécution depuis Jenkins.

## 10.6 Comparaison synthétique : GitLab CI vs Jenkins (même analyse SonarCloud)

Une analyse SonarCloud a déjà été réalisée via GitLab CI. Les métriques SonarCloud restent cohérentes, car l'outil d'analyse est le même (SonarCloud) ; seule la plateforme CI change.

Critère	GitLab CI + SonarCloud	Jenkins + SonarCloud
Fichier pipeline	<a href="#">.gitlab-ci.yml</a>	<a href="#">Jenkinsfile</a>
Exécution	GitLab Runner	Jenkins (controller/agent)
Déclenchement	push/merge (CI)	manuel ou automatique (Build Now / webhook)
Résultats SonarCloud	Identiques (mêmes règles/projet)	Identiques (mêmes règles/projet)
Visualisation	Pipeline GitLab	Vue Stages + Console détaillée

TABLE 1 – Comparaison GitLab CI vs Jenkins pour l'analyse SonarCloud.

## 11 Annexe — Rappel variables GitLab attendues

Dans GitLab → **Settings** → **CI/CD** → **Variables** :

- **SONAR\_TOKEN** : token généré dans SonarCloud
- **SONAR\_HOST\_URL** : <https://sonarcloud.io>
- **SONAR\_ORGANIZATION** : clé de l'organisation SonarCloud