
sonarqube

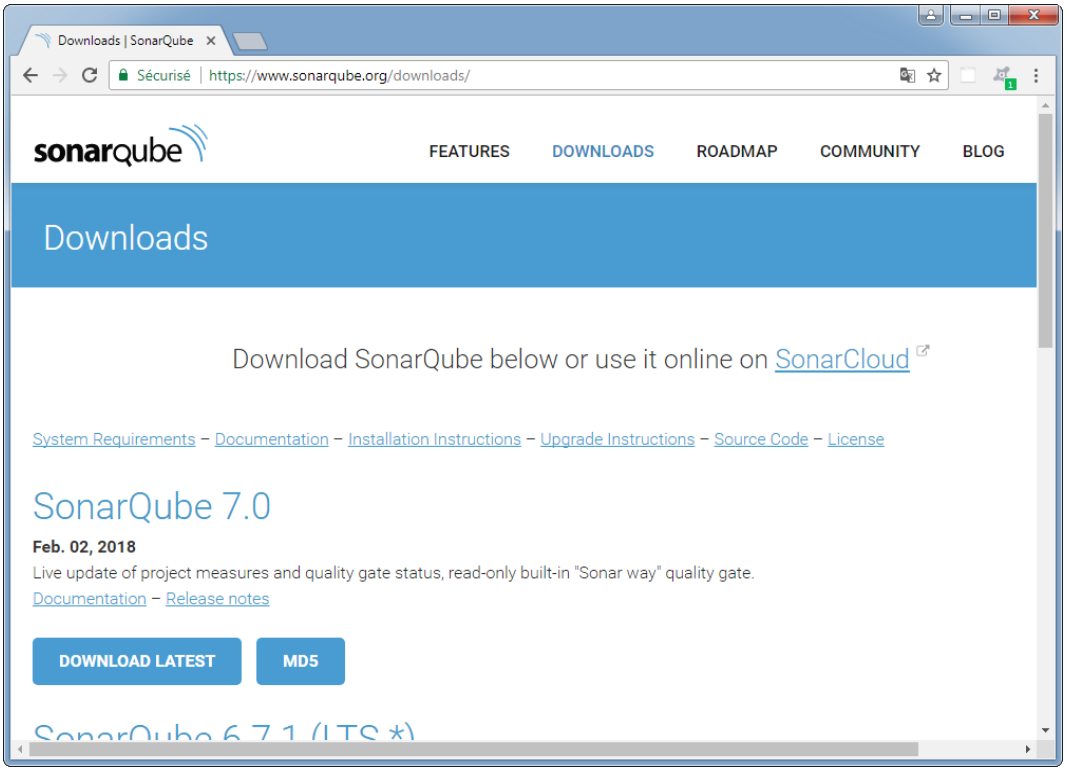
Prérequis

- Java : JDK 1.8 ou plus récent.
- RAM : 256 Mb
- Espace disque : 1 Gb.
- Système d'exploitation : Aucun prérequis.
    - Disponible sous Windows, Linux, MAC, etc…

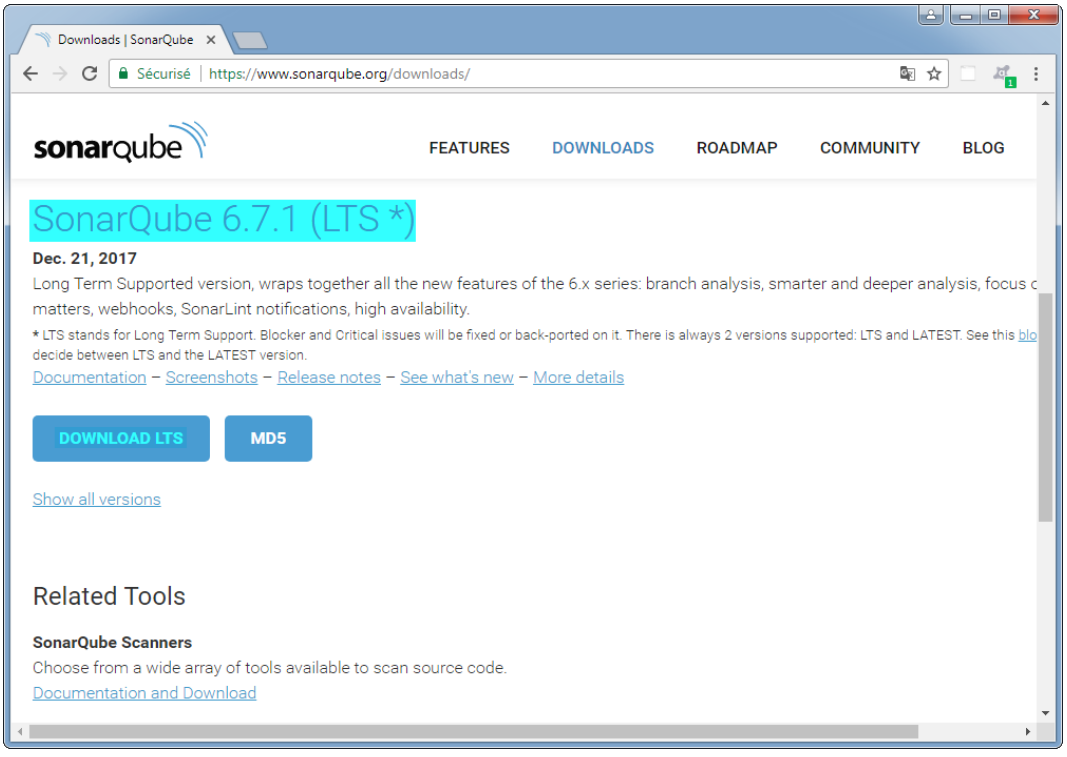
I. Installation du serveur sonarqube

Etape 1: Téléchargement de sonarqube

- Télécharger sonarqube depuis son site web officiel https://www.sonarqube.org/downloads/



- Choisir une version Long-term Support (LTS) de type Generic Java package.



Etape 2: Installation de sonarqube

- Télécharger le binaire du sonarqube

```
#  wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-5.4.zip
```

- Décompresser et copier le dossier sonarqube-* dans le dossier /opt

```
# unzip sonarqube-5.4.zip
# mv sonarqube-5.4 /opt/sonarqube-5.4
```

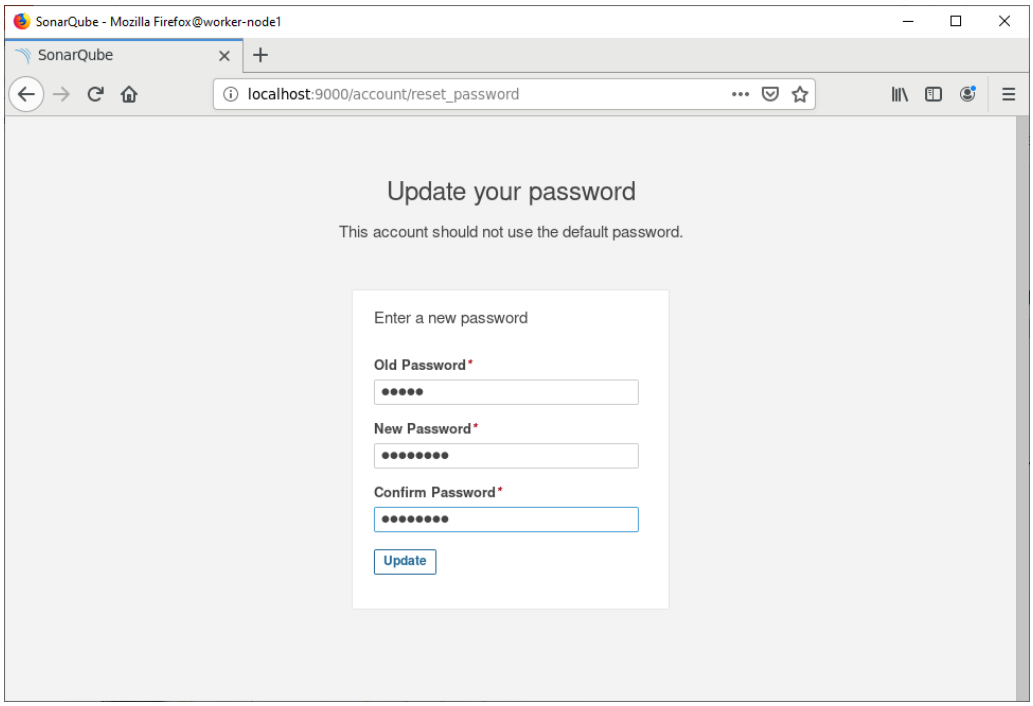
- Démarrer le serveur sonarqube

```
# /opt/sonarqube-5.4/bin/linux-x86-64/sonar.sh  start
```

```
[root@worker-node1 linux-x86-64]# ./sonar.sh start
Starting SonarQube...
Started SonarQube.
[root@worker-node1 linux-x86-64]# ./sonar.sh status
SonarQube is running (41522).
```

NB : Pour apporter des modifications sur la configuration globale du serveur sonarqube, éditer le fichier **`/opt/sonarqube-8/conf/sonar.properties`**

- Testez la connexion au serveur sonarqube via l'url : http://<IP>:9000
- Utiliser le compte administrateur par défaut pour s'authentifier : admin/admin puis initialiser le mot de passe



- Créer le projet Java « App_Test_Maven_Sonarqube »
- Ajouter le plugin sonar et dans le fichier pom.xml

```
<plugin>
  <groupId>org.sonarsource.scanner.maven</groupId>
  <artifactId>sonar-maven-plugin</artifactId>
  <version>3.0</version>
</plugin>
<plugin>
  <groupId>org.jacoco</groupId>
  <artifactId>jacoco-maven-plugin</artifactId>
  <version>0.8.6</version>
</plugin>
```

- Modifier le code source en ajoutant du mauvais code
- Compiler le projet en précisant le goal **`sonar:sonar`**
- Accéder au tableau de bord du Sonarqube et vérifier les problèmes
- Corriger les erreurs et vérifier le changement de qualité de code sur Sonarqube

Exercice :

- Créer le projet Java « App_Test_Maven_Sonarqube »

```
mvn archetype:generate
```

- Ajouter le plugin sonar et dans le fichier pom.xml

```
        <plugin>
          <groupId>org.sonarsource.scanner.maven</groupId>
          <artifactId>sonar-maven-plugin</artifactId>
          <version>3.7.0.1746</version>
        </plugin>
        <plugin>
          <groupId>org.jacoco</groupId>
          <artifactId>jacoco-maven-plugin</artifactId>
          <version>0.8.6</version>
        </plugin>
```

- Modifier le code source en ajoutant du mauvais code

```
package org.gk.cursusdevops;

import java.util.*;

        public class App  {

        public App(){
                System.out.println("C'est le constructeur ....");
        }
        public void methodeA(){
                        String chaine="";
                List<Integer> list = new ArrayList<Integer>();
                list.add(1);
                list.add(2);
                list=null;
                list.add(3);
                Object obj = getInfo();
                System.out.println(obj.toString());
        }
        public Object getInfo(){
                return null;
        }

    public static void main( String[] args ) {
        System.out.println( "Hello World!" );
        App app = new App();
        app.methodeA();
    }
}
```

- Compiler le projet en précisant le goal :sonar

```
# mvn clean sonar:sonar
```

- Corriger les erreurs et vérifier le changement de qualité de code sur Sonarqube

Page App.java

```
import java.util.*;
public class App{
        public App(){
                //System.out.println("C'est le constructeur ....");
        }
        public void methodeA(){
                //String chaine="";
                List<Integer> list = new ArrayList<Integer>();
                list.add(1);
                list.add(2);
                list.add(3);
                Object obj = getInfo();
                if(obj!=null){
                     System.out.println(obj.toString());
                }
        }
        public Object getInfo(){
                return null;
        }
        public static void main( String[] args ) {
        //System.out.println( "Hello World!" );
        App app = new App();
        app.methodeA();
        }
}
```

Page TestApp.java

```
package org.gk.cursusdevops;
import static org.junit.Assert.assertTrue;
import org.junit.Test;

public class AppTest{
    @Test
    public void shouldAnswerWithTrue(){
        App a =new App();
        a.methodeA();
        a.getInfo();
        assertTrue( true );
    }
}
```

Ajouter une fonction mehodeB avec un mauvais code

```
        public void methodeB(){
        String myString = null;
        System.out.println("Equal? " + myString.equals("foo"));
        return null;
        }
```

Correction : System.out.println("Equal?" + "foo".equals(myString));

- ⇨ Cela empêche le déclenchement d'exceptions de pointeur nul, car un littéral de chaîne ne peut jamais être nul par définition.
- ⇨ Télécharger le binaire du sonarqube

II. Intégration de Sonarqube avec Jenkins

Etape 1 : Configuration du SonarScanner

- Télécharger le binaire du SonarScanner depuis l'URL : https://docs.sonarqube.org/latest/analysis/scan/sonarscanner/

```
# wget https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cli-4.5.0.2216-linux.zip
```

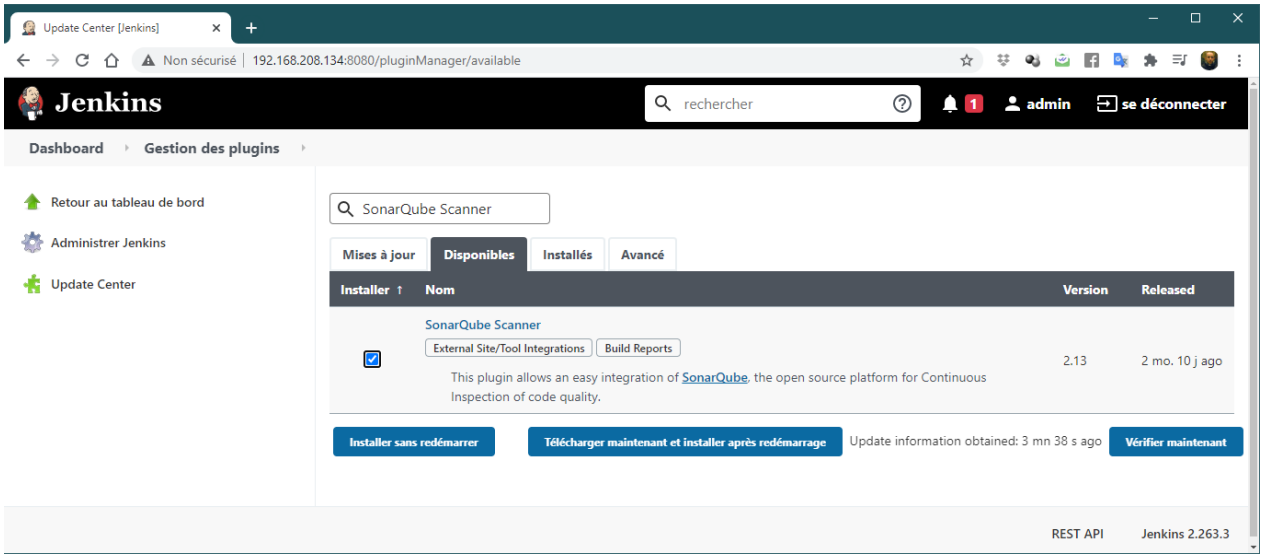
- Décompresser et copier le dossier sonar-scanner-cli* dans le dossier /opt

```
# unzip sonar-scanner-cli-4.5.0.2216-linux.zip
# mv sonar-scanner-4.5.0.2216-linux/ /opt/sonar-scanner-4.5
```

Etape 2 : Intégration de Sonarqube avec Jenkins

- Installez git plugin (Sans redémarrer)

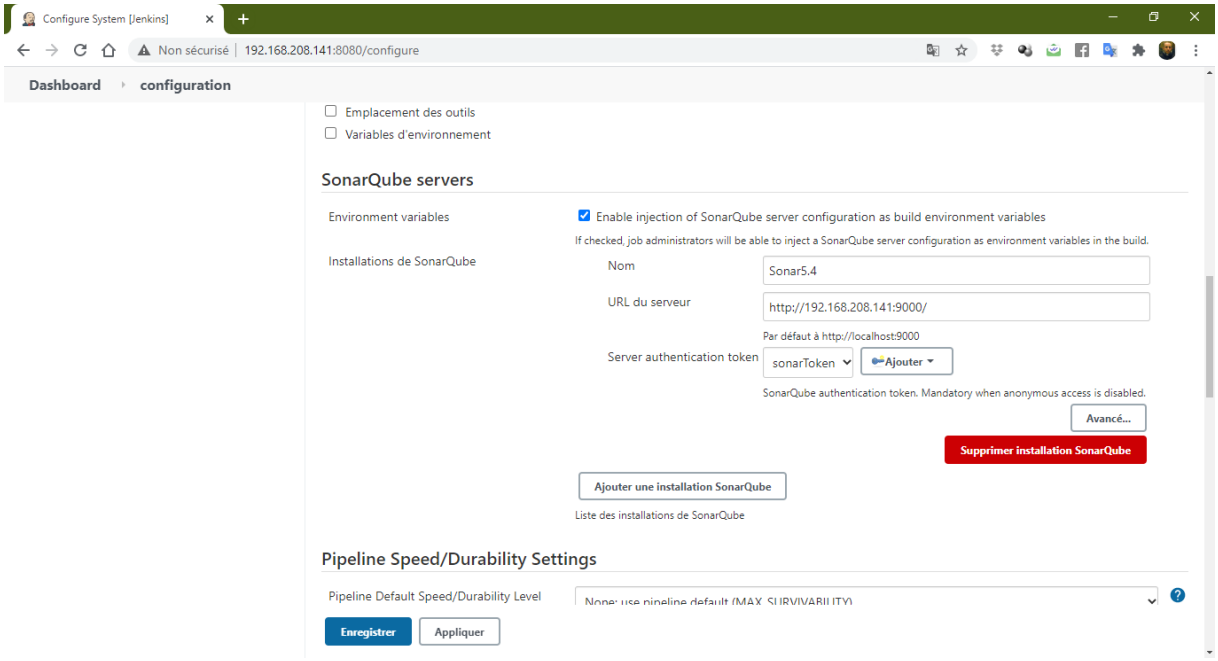
`Administrer Jenkins > Gestion de plugins > Disponibles > SonarQube Scanner`



- Configurez de serveur SonarQube:
  - `Administrer Jenkins > Configurer le système > SonarQube servers`
  - Faites défiler jusqu'à la section de configuration SonarQube, cliquez sur Ajouter SonarQube et ajoutez les valeurs qui vous sont demandées.

Remarque : Le jeton d'authentification du serveur doit être créé en tant qu'informations d'identification 'Secret Text'.

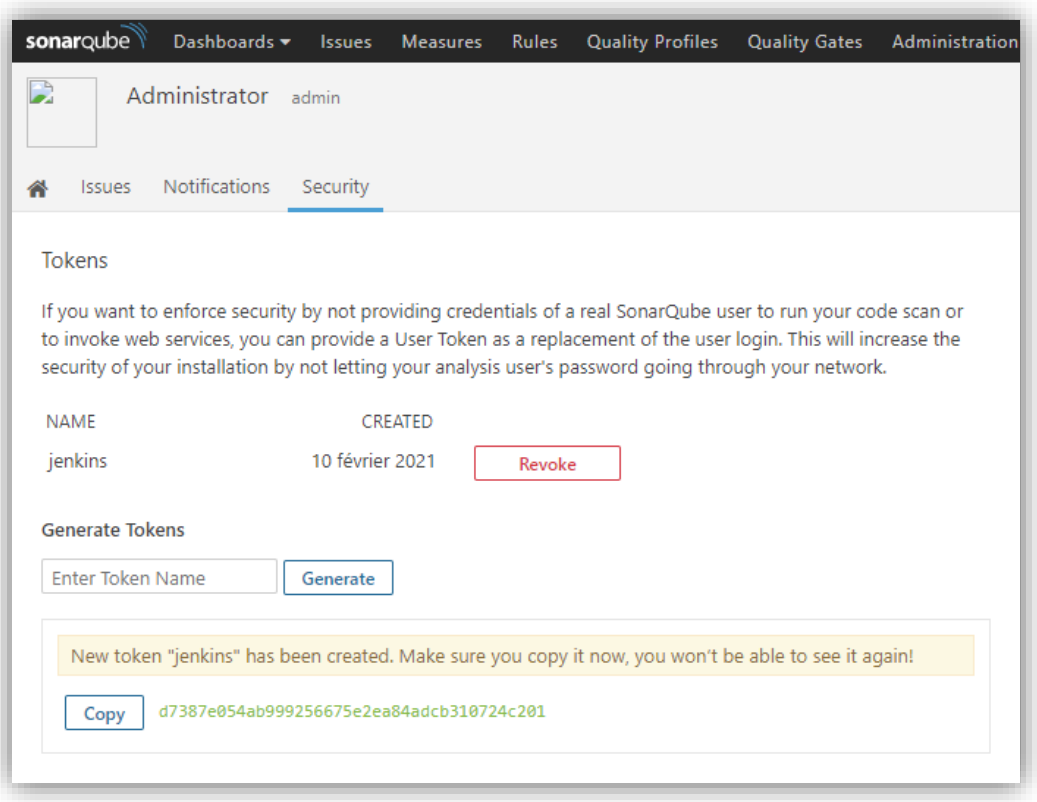
- Cochez la case « Enable injection of SonarQube server configuration as build environment variables »



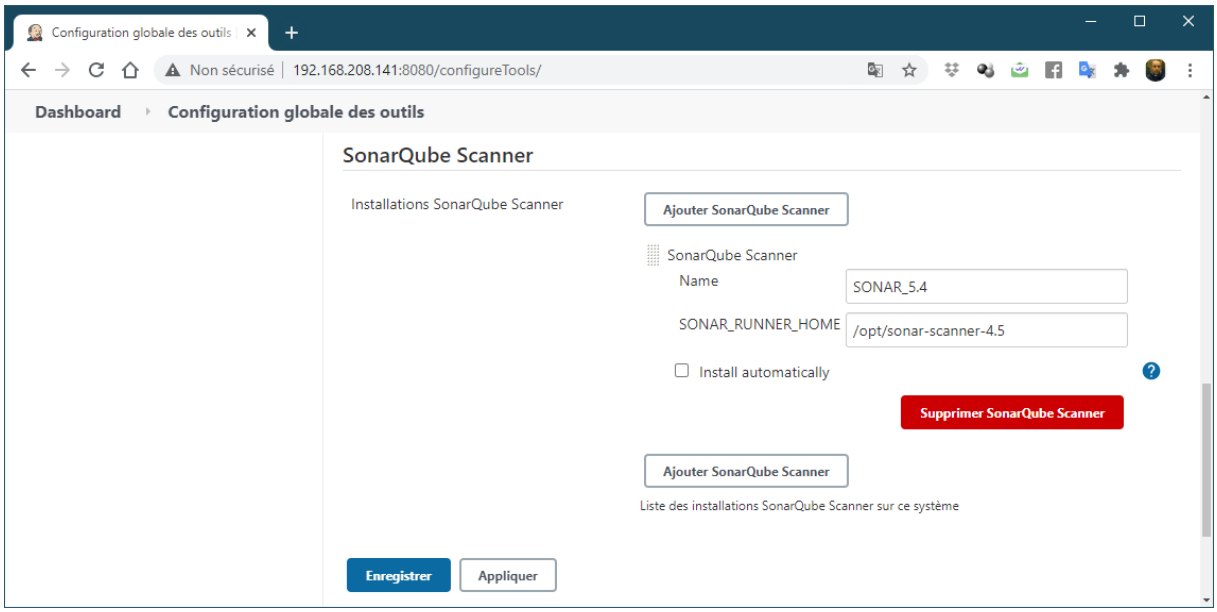
Générer un jeton

Vous pouvez générer de nouveaux jetons dans :

- Utilisateur> Mon compte> Sécurité



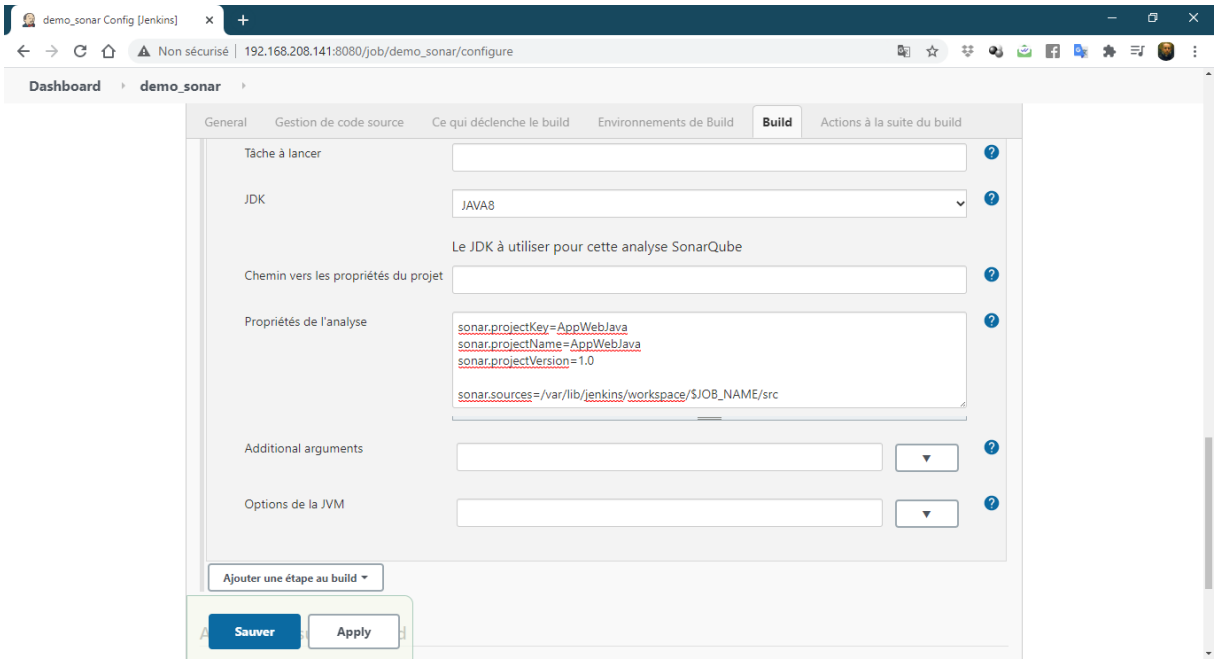
- Configurez de SonarQube Scanner:
    - Administrer Jenkins > Configuration globale des outils > SonarQube Scanner



- Analyse d'un projet Java
    - Créez un nouvel item de type « projet free-style »
    - Dans la section « Build », Ajouter une étape au build de type : « ***Lancer une analyse avec SonarQube Scanner*** »
    - Remplir les propriétés du projet à analyser :
        - sonar.projectKey
        - sonar.projectName
        - sonar.projectVersion
        - sonar.sources

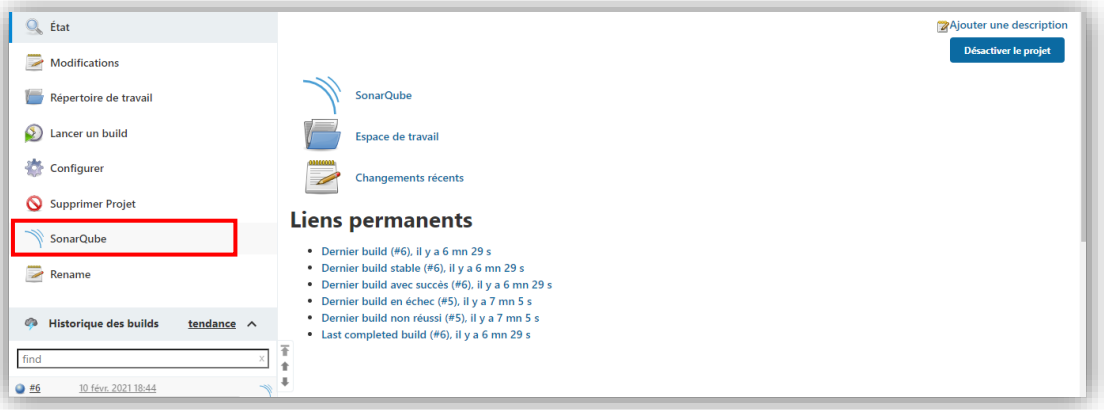
Exemple :

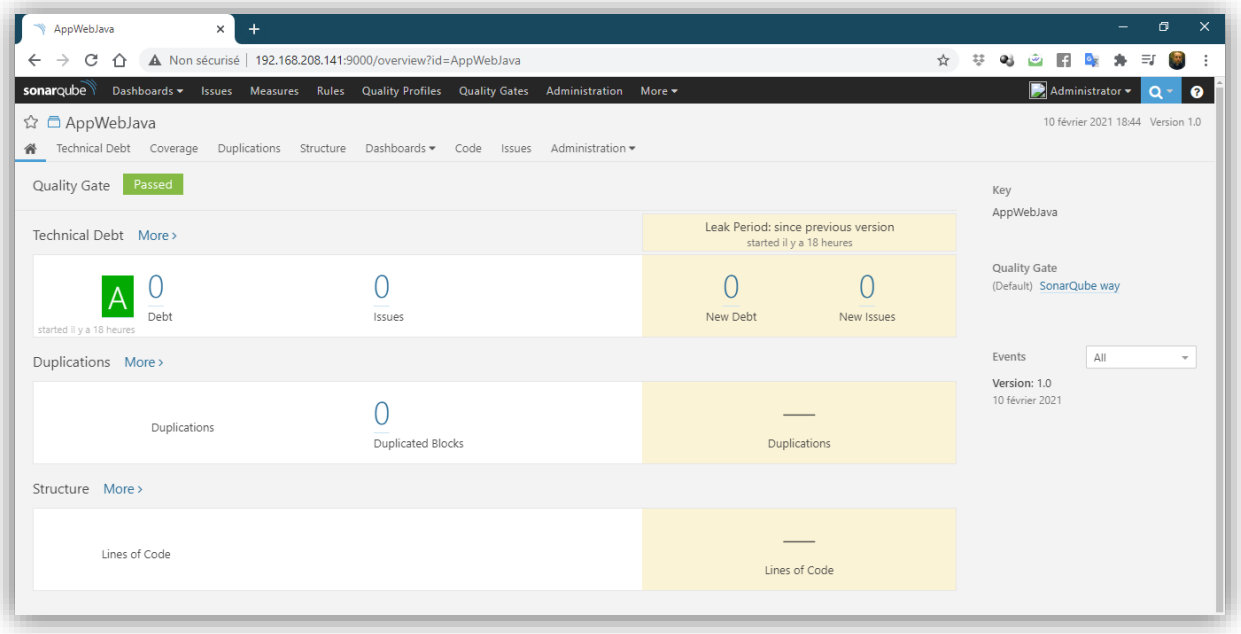
```
sonar.projectKey=AppWebJava
sonar.projectName=AppWebJava
sonar.projectVersion=1.0

sonar.sources=/var/lib/jenkins/workspace/$JOB_NAME/src
```



- Lancez le build pour démarrer l'analyse de votre projet.

- Pour vérifier le rapport d'analyse, cliquez sur l'icon SonarQube dans le tableau de bord du projet




AppWebJava
Non sécurisé | 192.168.208.141:9000/overview?id=AppWebJava
sonarqube
Dashboards
Issues
Measures
Rules
Quality Profiles
Quality Gates
Administration
More
Administrator
AppWebJava
10 février 2021 18:44 Version 1.0
Technical Debt
Coverage
Duplications
Structure
Dashboards
Code
Issues
Administration
Quality Gate
Passed
Technical Debt More ›
Leak Period: since previous version
started il y a 18 heures
A
0
Debt
started il y a 18 heures
0
Issues
0
New Debt
0
New Issues
Duplications More ›
Duplications
0
Duplicated Blocks
—
Duplications
Structure More ›
Lines of Code
—
Lines of Code
Key
AppWebJava
Quality Gate
(Default) SonarQube way
Events
All
Version: 1.0
10 février 2021