

SONARQUBE



sonarqube 

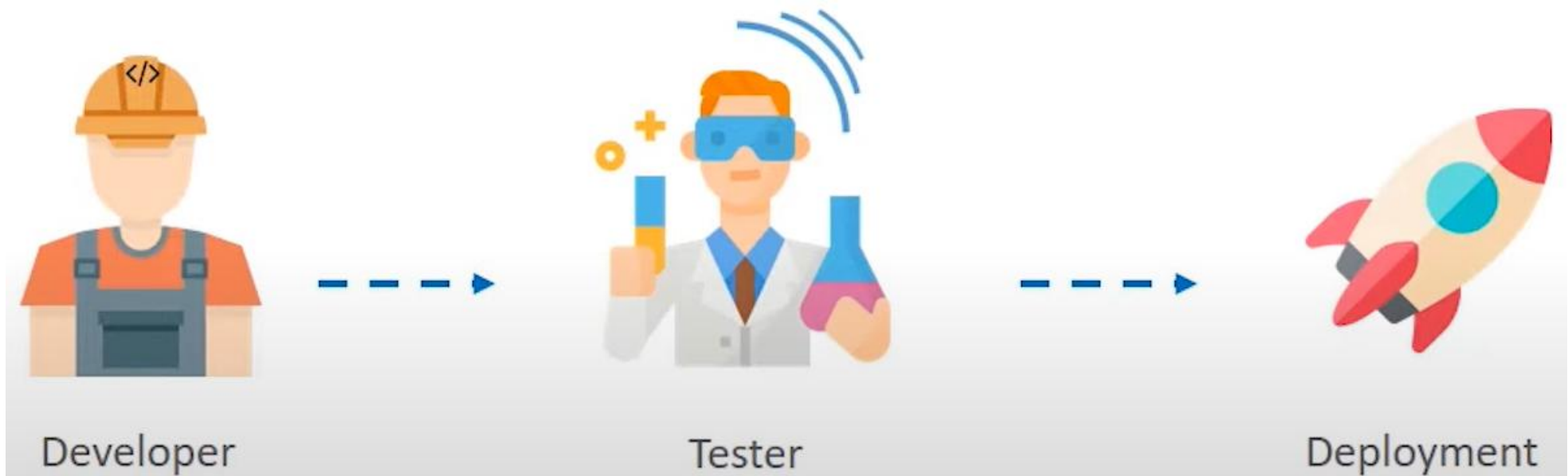
The SonarQube logo, which consists of the word "sonarqube" in a bold, lowercase sans-serif font. To the right of the text is a blue icon consisting of three concentric, curved lines that resemble a sonar wave or a signal.

PLAN DU COURS

- Tests dynamiques et Tests statiques
- C'est quoi SONARQUBE
- Caractéristiques de SONARQUBE
- Installation de SONARQUBE (à partir d'une Image Docker)
- Utilisation de Sonarqube
- Compréhension des résultats d'analyse de code

Tests dynamiques et Tests statiques

- Les **tests** font partie de cycle de vie du développement d'une application donnée.
- Les tests visent à s'assurer que le code qui sera déployé est de **bonne qualité, sécurisé** et ne présente pas de **bugs** (environ **30%** du temps de développement doit être consacré aux tests).

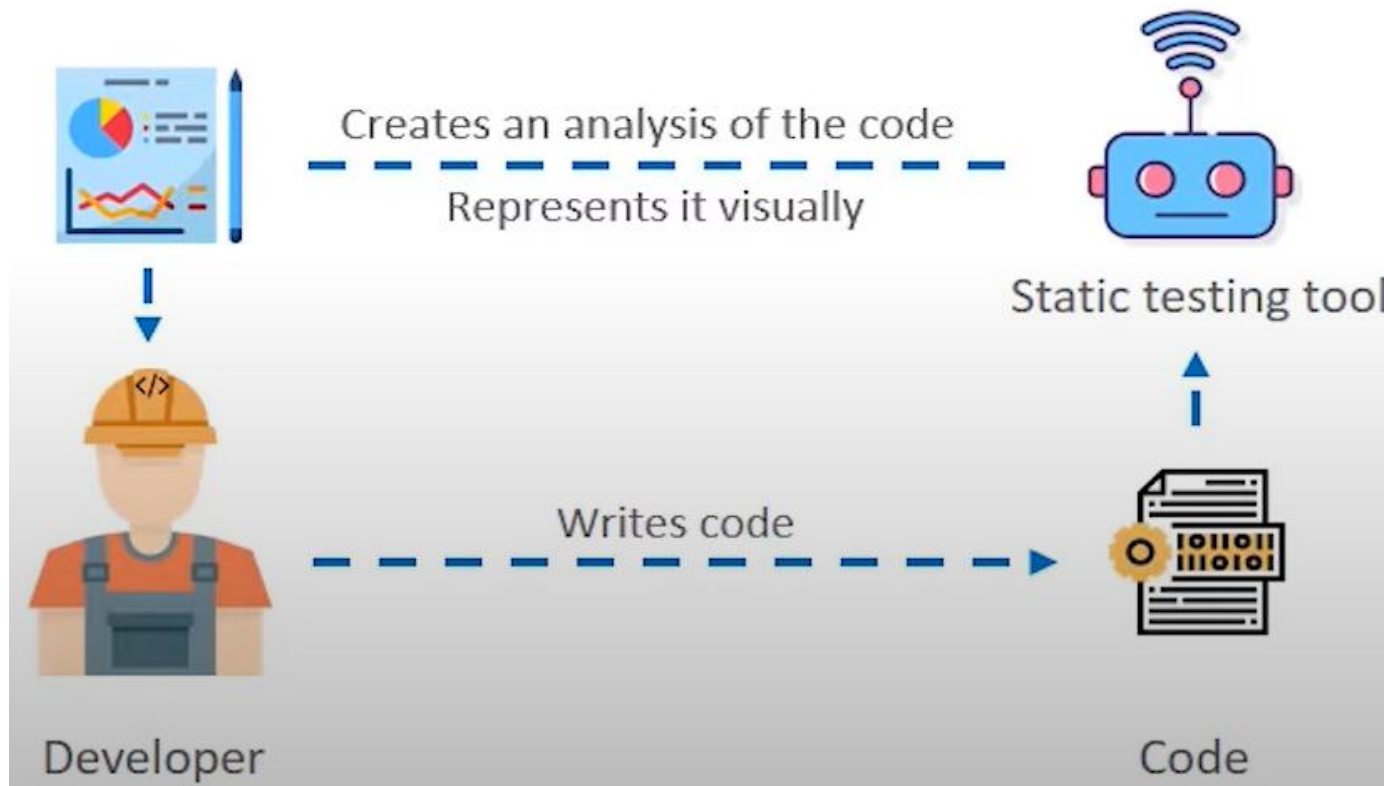


Tests dynamiques et Tests statiques

- Les **tests dynamiques** : Ces tests sont faits alors que l'application tourne, pour détecter les **dysfonctionnements** (fonctionnalités mal implémentées, DB inaccessible, ...) : Tests Unitaires (JUnit par exemple).
- Les **tests statiques** : Ces tests sont faits sur le code source, avant de l'exécuter. Il s'agit d'analyser le code pour détecter les écarts aux **bonnes pratiques de développement** (absence de logs, absence de commentaires. SONARQUBE fait ce type de tests.

SONARQUBE

- **SONARQUBE** est un outil de test statique, open source, utilisé pour analyser la qualité du code source, selon des règles prédéfinies. Il permet donc l'inspection en continue de la qualité de votre code (Code Review automatique).

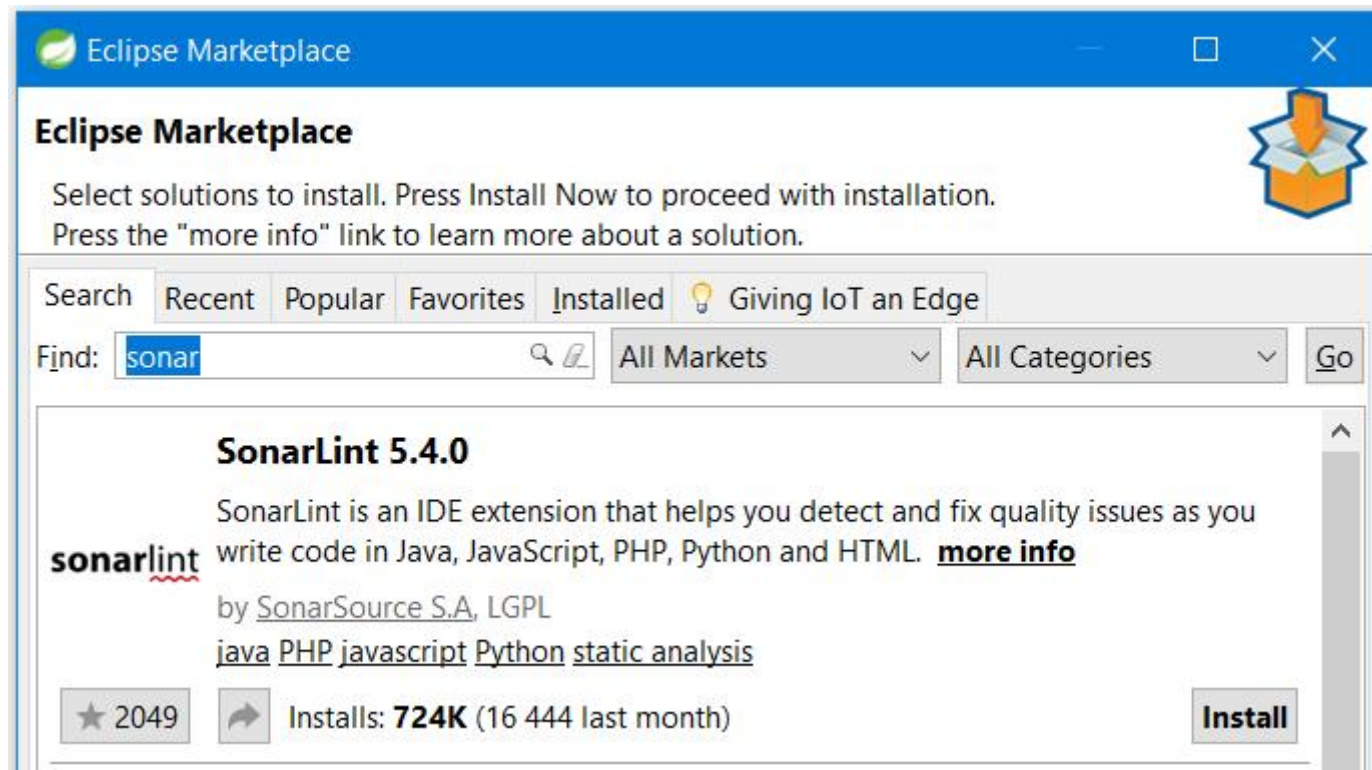


Caractéristiques de SONARQUBE

- SONARQUBE peut être utilisé avec une vingtaine de langages (Java, .Net (C#), Python, PHP, Cobol, JavaScript, ...)
- Il permet de détecter **les défauts de codage** (code jamais utilisé, dupliqué, sans commentaires, sans tests unitaires, sans gestion d'exception, non sécurisé,).
- Il nous permet de choisir **les règles à activer** lors de l'analyse de notre code.
- SONARQUBE peut être installé en mode **standalone**, ou en tant que **plugin** intégré à un IDE comme **STS** (Eclipse).

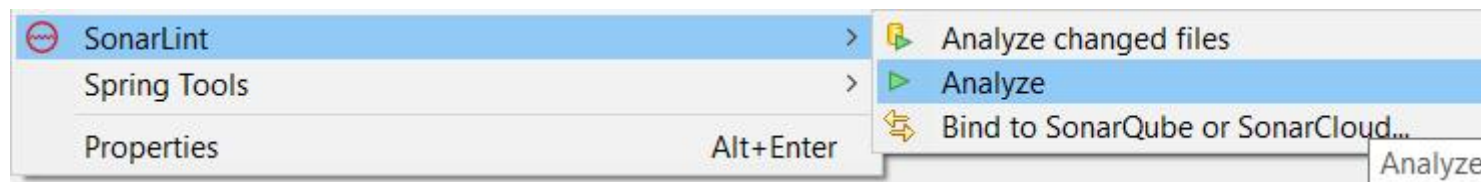
PLUGIN SONARQUBE pour STS

- Dans STS, aller dans Help -> Eclipse Marketplace, chercher « sonar », installer le plugin « SonarLint », accepter la licence, accepter de redémarrer STS



PLUGIN SONARQUBE pour STS

- Ouvrir un de vos projets sur STS, bouton droit et choisir « Sonar Lint » -> Analyze, voir le résultat :

A screenshot of the SonarLint Report tab in STS. The tab shows a list of 6 items with columns for 'Resource' and 'Description'. The items are listed in a table format.

Resource	Description
Calcul.java	Complete the task associated to this TODO comment.
Calcul.java	Replace this use of System.out or System.err by a logger.
Calcul.java	Remove this unused "j" local variable.
Calcul.java	Remove this useless assignment to local variable "j".
CalculTest.java	Remove this assertion from production code.
CalculTest.java	This block of commented-out lines of code should be removed.

SONARQUBE : INSTALLATION

- Connectez-vous à votre VM Centos7 et lancer un client ssh (vagrant up, vagrant ssh).
- Récupérer l'image Docker 'sonarqube:8.9.7-community' de Dockerhub
- Attention : version **8.9.7-community** et non la latest.
- (faites un chmod auparavant pour éviter les problèmes de droits d'accès) :

```
[vagrant@localhost ~]$ sudo chmod 666 /var/run/docker.sock  
[vagrant@localhost ~]$ docker pull sonarqube:8.9.7-community
```

SONARQUBE : LANCEMENT

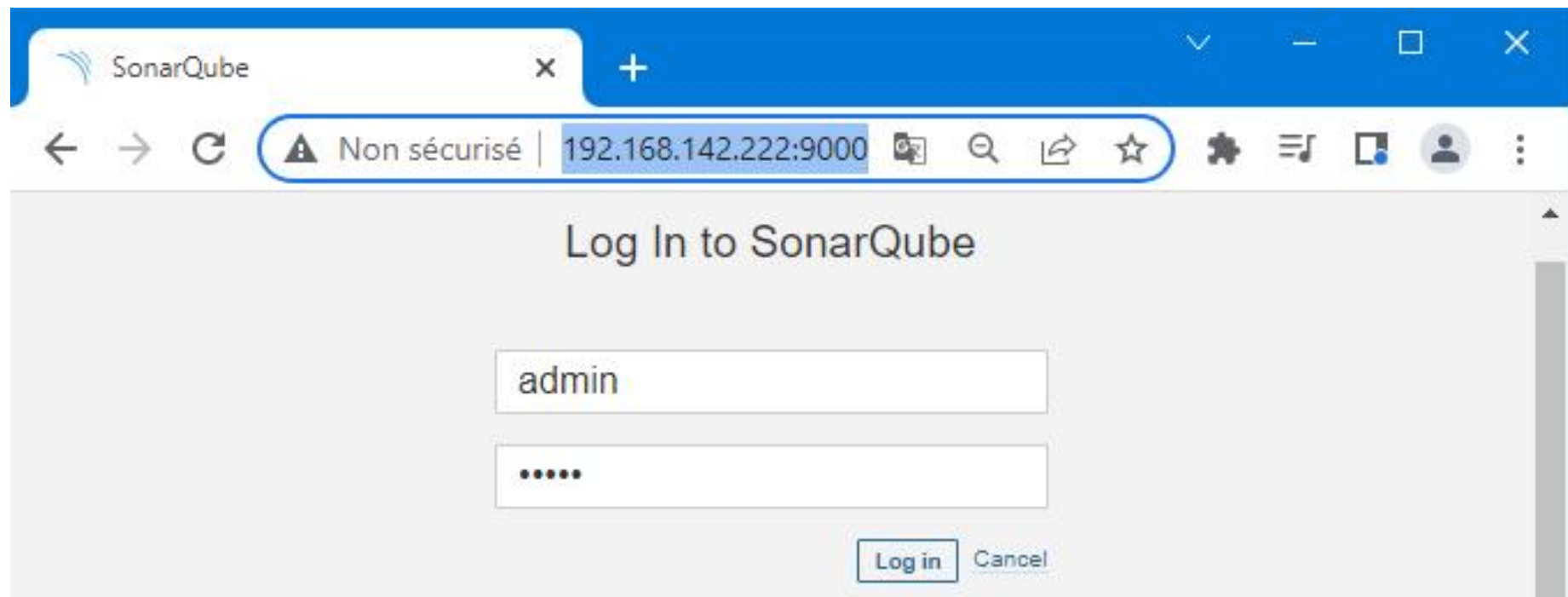
- Lancer Sonarqube : `docker run -p 9000:9000 sonarqube:8.9.7-community`

```
2022.09.27 20:14:36 INFO app[][o.s.a.SchedulerImpl] Process[ce] is up
2022.09.27 20:14:36 INFO app[][o.s.a.SchedulerImpl] SonarQube is up
```

- Optionnel : C'est possible de lancer Sonar en background (option -d ou un docker start sur le conteneur).

SONARQUBE : UTILISATION

- Sur votre machine Windows, aller à l'url `http://<ip-vm>:9000` et se connecter avec `admin / admin` :



- Changer le mot de passe à sonar par exemple (`admin / sonar`).

SONARQUBE : UTILISATION

- **Attention** : Si vous arrêtez le conteneur sonarcube (CTRL S ou en arrêtant la VM), il ne faut pas lancer un docker run sur l'image sonarcube, car cela créera un nouveau conteneur. Il faut juste faire :

```
> Sélection vagrant@localhost:~  
[vagrant@localhost ~]$ sudo chmod 666 /var/run/docker.sock  
[vagrant@localhost ~]$ docker ps -a  
CONTAINER ID        IMAGE                                     COMMAND  
ba73cc1e77eb        sonarcube:8.9.7-community             "bin/run.s  
...
```

- Le chmod vous permet d'utiliser les commandes docker sans sudo
- Puis faire un docker start id-conteneur-sonarcube et attendre quelques minutes le temps que tout se lance (Elasticsearch, Sonarcube, ...) :

```
[vagrant@localhost ~]$ docker start ba73cc1e77e  
ba73cc1e77eb  
[vagrant@localhost ~]$
```

SONARQUBE : UTILISATION

- Puisque nous n'avons pas de projets à analyser sur notre VM, nous allons utiliser Jenkins, pour récupérer un projet de Git, puis l'analyser avec Sonarqube :
- Se connecter à **Jenkins** via l'url `http://<ip-vm>:8080`
- Sur le **pipeline** Jenkins déjà créé, récupérer le code de votre projet de Github en ajoutant le stage suivant dans le script Groovy (mettez l'URL de votre projet ou testez avec le repo Git ci-dessous) :

```
stage ('GIT') {  
    steps {  
        echo "Getting Project from Git";  
        git "https://github.com/mhassini/timesheet-ci.git";  
    }  
}
```

SONARQUBE : UTILISATION

- Sur le même pipeline, lancer les commandes Maven **clean** et **compile** pour compiler le code de votre projet récupéré de Git :

```
stage('MVN CLEAN') {  
    steps {  
        sh 'mvn clean'  
    }  
}  
  
stage('MVN COMPILE') {  
    steps {  
        sh 'mvn compile'  
    }  
}
```

SONARQUBE : UTILISATION

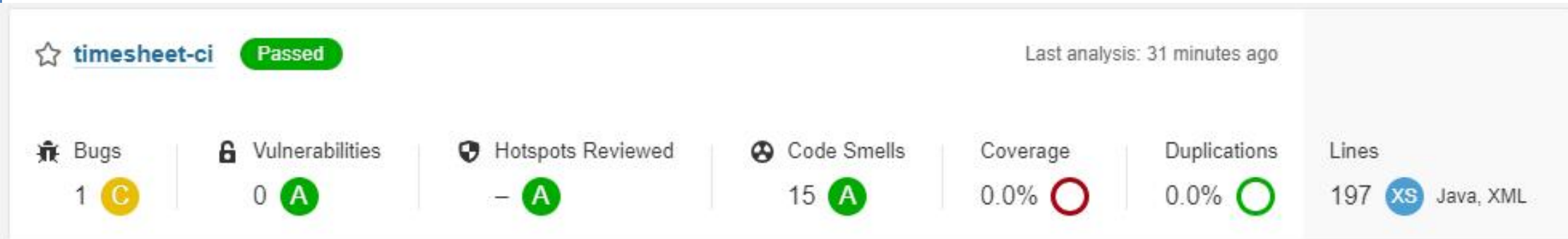
- Sur le même pipeline, lancer la commandes Maven d'analyse de code **sonar:sonar** pour analyser la qualité de votre code et envoyer le rapport au serveur Sonarqube (Mettez le mot de passe de votre Sonarqube) :

```
stage('MVN SONARQUBE') {  
    steps {  
        sh 'mvn sonar:sonar -Dsonar.login=admin -Dsonar.password=sonar' ????  
    }  
}
```

- **Lancer le Job via Jenkins :**

SONARQUBE : ANALYSE DES RÉSULTATS

- Aller dans <http://<ip-vm>:9000> et regarder le résultat de l'analyse de votre projet :



- Sur l'interface ci-dessus, il suffit de cliquer sur le projet « timesheet » pour accéder aux détails de toute cette analyse :
- Sonarqube détecte automatiquement le **langage (Java + XML)** dans notre cas)
- Sonarqube vérifie si le développeur a mis du **code dupliqué** dans plusieurs endroits (source d'erreur) : 0% dans notre cas




SONARQUBE : ANALYSE DES RÉSULTATS

- **Bug** : Un code qui peut engendrer un mauvais comportement de l'application (un null pointer exception par exemple), dans notre cas :

```
@Override
public User retrieveUser(String id) {
    l.info("in retrieveUser id = " + id);
    //User u = userRepository.findById(Long.parseLong(id)).orElse(null);
    //int i = 1/0;
    User u = userRepository.findById(Long.parseLong(id)).get();

    l.info("user returned : " + u);
    return u;
}
```

Call "Optional#isPresent()" before accessing the value. Why is this an issue?




 Bug ▾  Major ▾  Open ▾ Not assigned ▾ 10min effort [Comment](#)

```
l.info("user returned : " + u);
return u;
}
```




- **Vulnerability** : Faille de sécurité dans notre code.
- **Hotspots Reviewed** : Code à revoir pour être sûr que ce n'est pas une faille de sécurité.

SONARQUBE : ANALYSE DES RÉSULTATS

- Sonarqube affiche si le code a été exécuté par les outils de tests (comme JUnit). Sonarqube ne fait pas l'analyse lui-même mais se base sur d'autres outils comme JaCoCo (c'est pour cela qu'on a 0% de **Coverage** puisque JaCoCo n'est pas ajouté à notre projet).
- **Code Smells** : Ce n'est pas un bug, mais c'est un code qui peut retarder l'équipe de développement ou l'équipe de support quand ils essaient de comprendre ou modifier le code (exemple : beaucoup de commentaires ou des imports non utilisés) :

☐ This block of commented-out lines of code should be removed. [Why is this an issue?](#)
 Code Smell ▾  Major ▾  Open ▾ Not assigned ▾ 5min effort [Comment](#)

 src/.../java/tn/esprit/spring/services/UserServiceImpl.java

☐ Remove this unused import 'javax.transaction.Transactional'. [Why is this an issue?](#)
 Code Smell ▾  Minor ▾  Open ▾ Not assigned ▾ 2min effort [Comment](#)

SONARQUBE

Passed

All conditions passed.

New Code

Overall Code

1  Bugs


Reliability 

0  Vulnerabilities

Security 

0  Security Hotspots 

— Reviewed

Security Review 

1h 32min Debt

15  Code Smells

Maintainability 

 0.0%
Coverage on 49 Lines to cover

—
Unit Tests

 0.0%
Duplications on 197 Lines

0
Duplicated Blocks