esprit
Se former autrement
HONORIS UNITED UNIVERSITIES

DevOp

sonarqube

Chapitre 4 : SonarQube

ESPRIT – UP ASI (Architecture des Systèmes d'Information)

Bureau E204

United Nations Educational, Scientific and Cultural Organization · UNESCO Chair "Project-based learning" ESPRIT School of engineering, Tunisia

EUR-ACE® Délivrée par la Commission des Titres d'Ingénieur

CONFÉRENCE DES GRANDES ÉCOLES

cdio CONCEIVE DESIGN IMPLEMENT OPERATE

Plan du cours

- Introduction 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 **Travail à faire**
- (Analyser le code d'un projet sur Git avec SONARQUBE Jenkins)
-
-
-

en utilisant

Introduction

Speedier time to market with higher returns on investment (ROI)

Early Detection and Fixing of Issues

Greater Collaboration, Team work, and Efficiency

Lower Failure Rate of New Releases and Shorter Lead Time Between Fixes

Problématique

Comment s'assurer que le code des logiciels est de haute qualité dès le début du développement et comment garantir que les applications restent fiables à long terme ?

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 **exempt de bugs** (environ 30% du temps de développement doit être consacré aux tests).

Développeur *Testeur* *Déploiement*

Tests dynamiques

- **Les tests dynamiques** consistent à exécuter le code pour repérer les erreurs, les bugs et évaluer les performances à l'aide de différentes méthodes telles que *les tests unitaires*, *les tests d'intégration*, *les tests de régression* et *les tests de charge*.

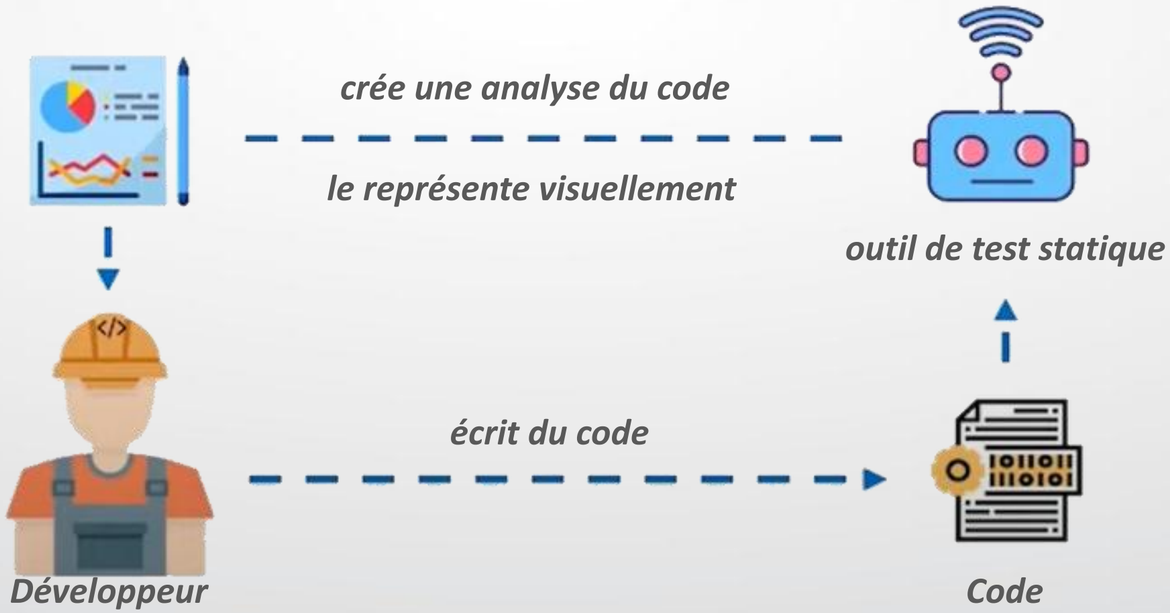
Tests statiques

- **Les tests statiques** sont réalisés sans exécuter le code. Ils inspectent le code source, la conception et la documentation pour détecter des problèmes potentiels, comme *les violations de normes de codage*, *les erreurs de syntaxe*, *les incohérences de conception*, etc.

→ **SonarQube** est un exemple d'outil d'analyse statique de code utilisé pour effectuer ces tests.

SonarQube

- **SonarQube** est un outil de test statique, open-source, qui analyse le code source pour détecter et corriger les problèmes de qualité, tels que les bugs, les vulnérabilités de sécurité et les mauvaises pratiques de codage. Il permet de
- ***mesurer la qualité du code source en continu*** (revue de code automatique).



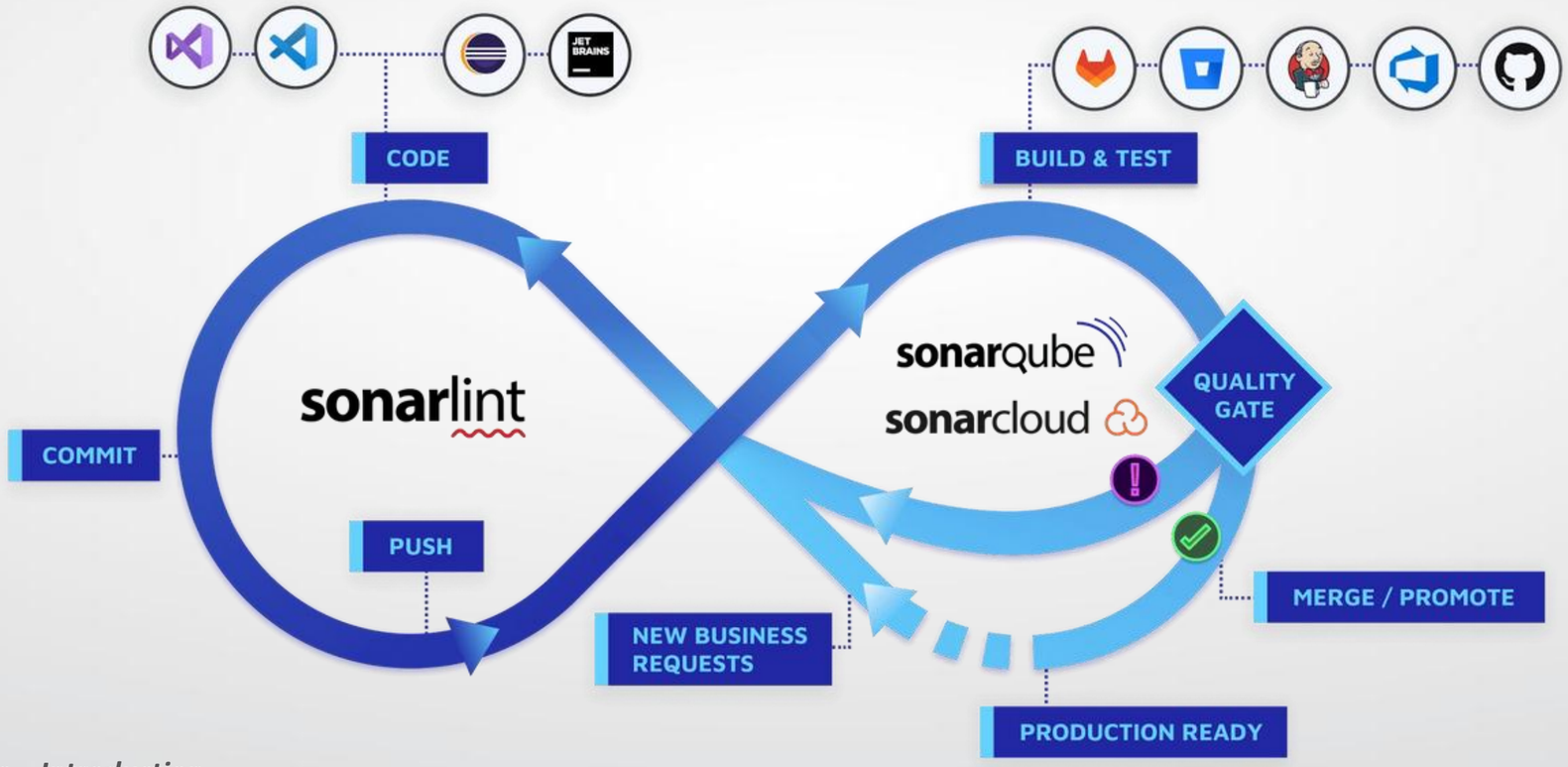
SonarQube - Caractéristiques

Les principales fonctionnalités de sonarqube

- Analyse du code source. Prise en charge de nombreux langages (Java, .Net (C#),
- Python, PHP, JavaScript, **Détection de problèmes dans le code**, comme les bugs…).
- et les vulnérabilités. Mesure de la qualité du code avec des métriques.
- Intégration avec des outils de gestion de projets et de développement. Création
- de rapports détaillés sur la qualité du code et son évolution. **Personnalisation**
- **des règles de qualité**. Support de l'intégration continue (CI/CD).
-
-

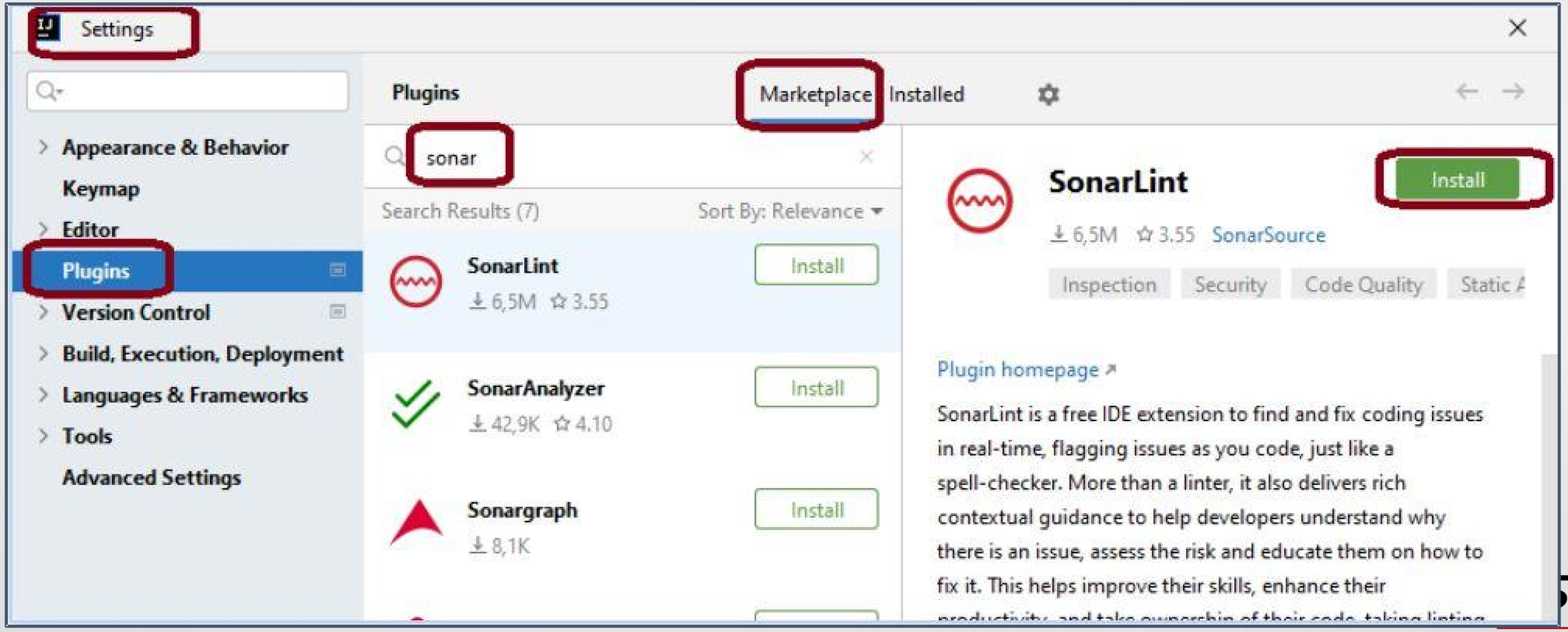
SonarQube - Plugin IDE

› **SonarQube** peut être installé de manière indépendante (en mode **standalone**) ou intégré en tant que **plugin** dans un environnement de développement (IDE).



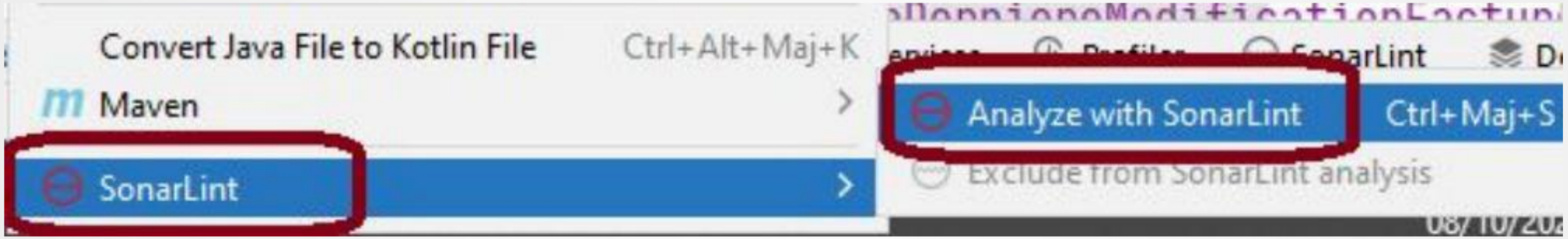
SonarQube - Plugin IntelliJ

- DansIntelliJ, vous pouvezaccéderà*Setting->Plugins->Marketplace*, puis recherchez « Sonar », installez le plugin « **SonarLint** », acceptez l'installation et acceptez de redémarrer IntelliJ à la fin du processus d'installation.



SonarQube - Plugin IntelliJ

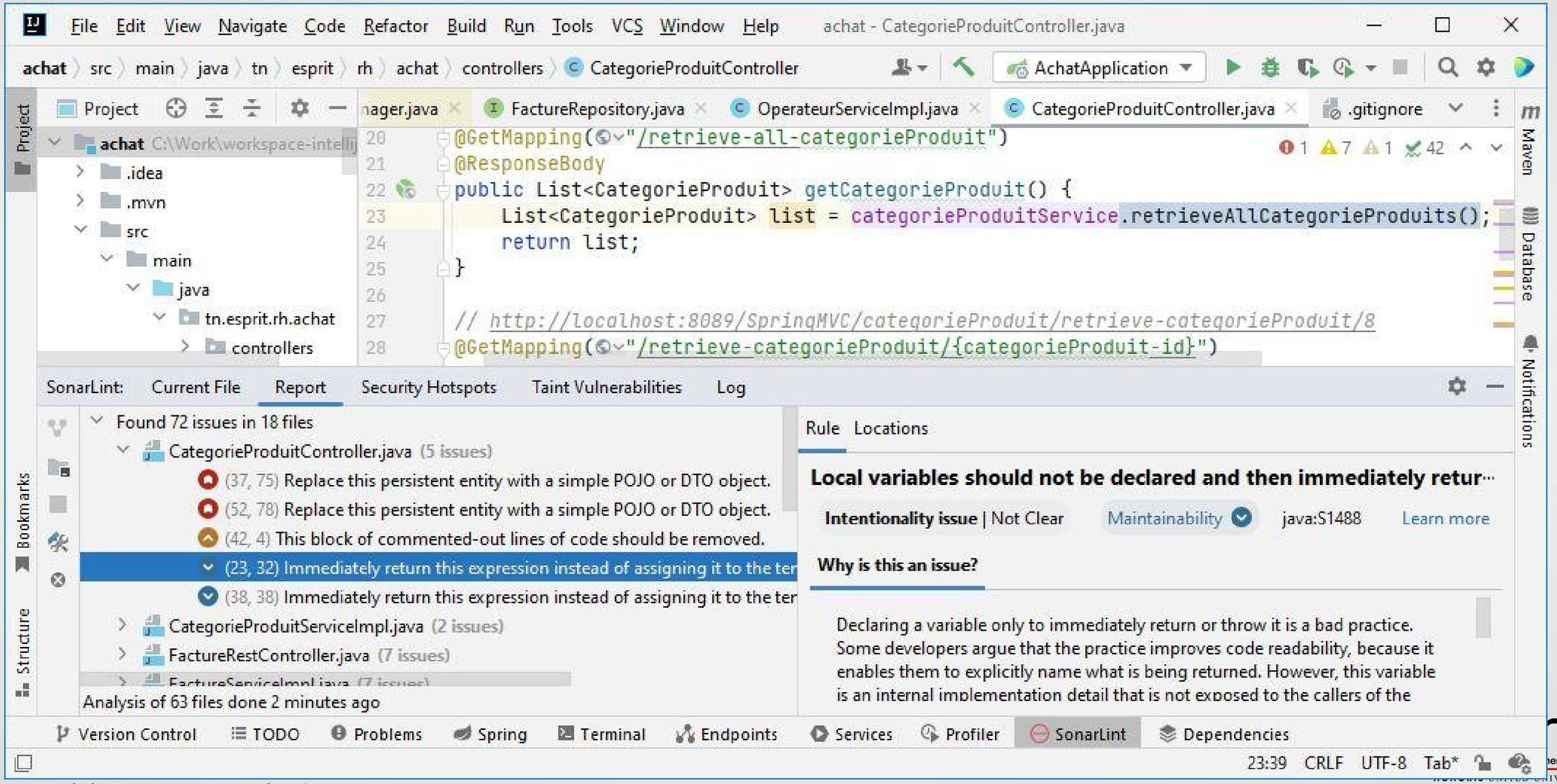
- Ouvrez l'un de vos projets dans IntelliJ, faites un clic droit et sélectionnez *SonarLint -> Analyze with SonarLint*, puis examinez les résultats



sonarlint

SonarQube - Plugin IntelliJ

› Résultat :



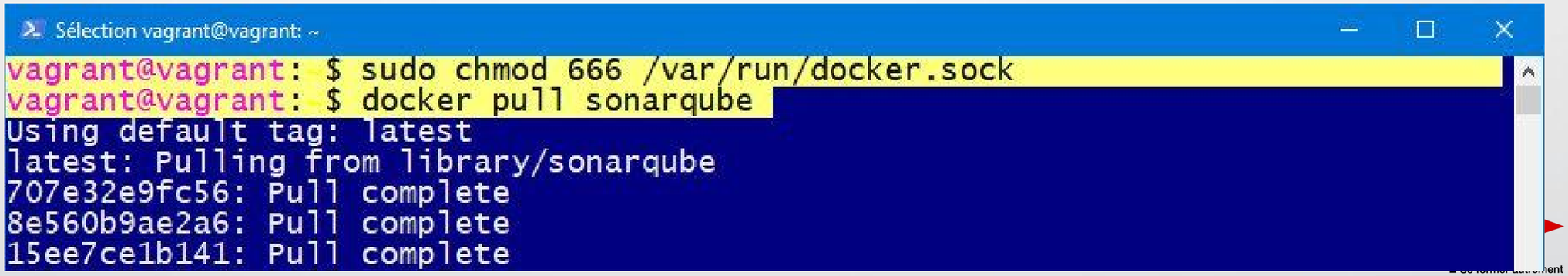
SonarQube - Installation

1. **Téléchargezl'image DockerdeSonarQube :**

› Connectez-vous à votre machine virtuelle Ubuntu en utilisant un client SSH. Pour ce faire, démarrez VirtualBox, ouvrez une fenêtre PowerShell et exécutez les commandes vagrant `up et vagrant ssh.`

› Assurez-vous d'avoir effectué un chmod au préalable pour éviter les problèmes de droits d'accès. Utilisez la commande docker pull pour télécharger l'image SonarQube depuis le

› Docker Hub.



SonarQube - Lancement

2. ExécutezleconteneurSonarQube :

›Vous pouvez maintenant lancer un conteneur SonarQube en utilisant la commande `docker run`.

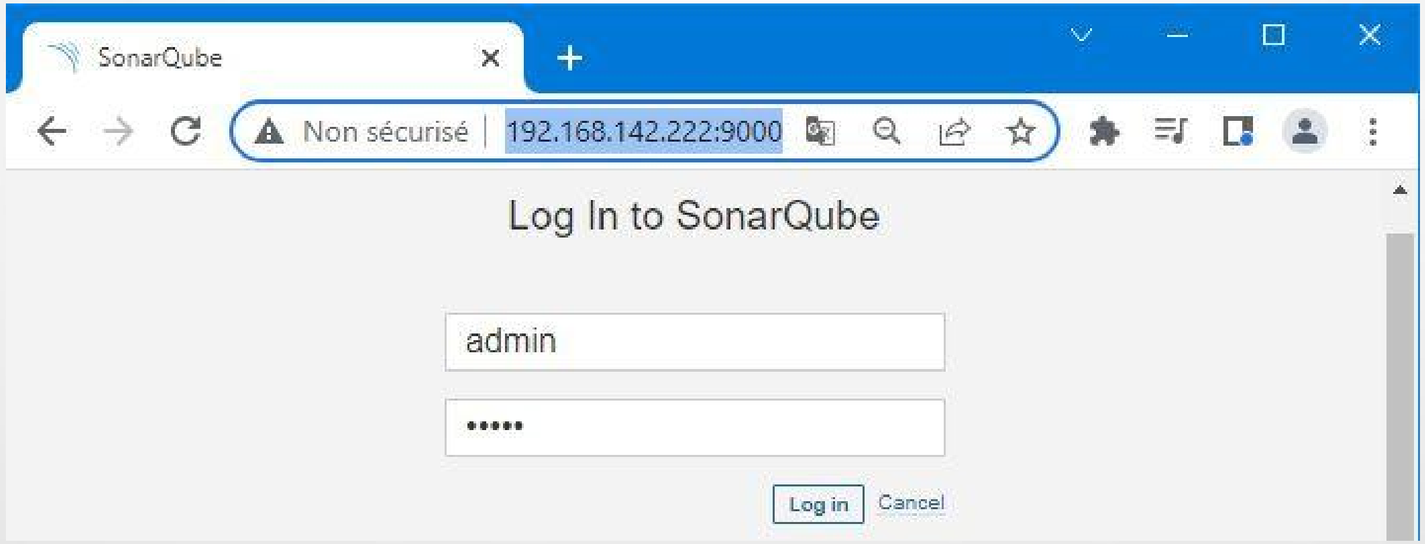
```
vagrant@vagrant:~$ docker run -d --name sonarqube -p 9000:9000 -p 9092:9092 sonarqube
```

-d : Lance le conteneur en en arrière-plan --name sonarqube : Attribue un nom au conteneur, ici "*sonarqube*" -p 9000:9000 -p 9092:9092 : Associe les ports du conteneur aux ports correspondants de l'hôte, permettant ainsi d'accéder à l'interface web et au broker de messagerie.

SonarQube - Utilisation

3. Accédez àl'interfaceweb deSonarQube:

›Vous pouvez maintenant accéder à l'interface web de SonarQube en ouvrant votre navigateur et en visitant http://<ip-vm>:9000. Vous devrez vous connecter avec les informations d'identification par défaut (admin/admin).



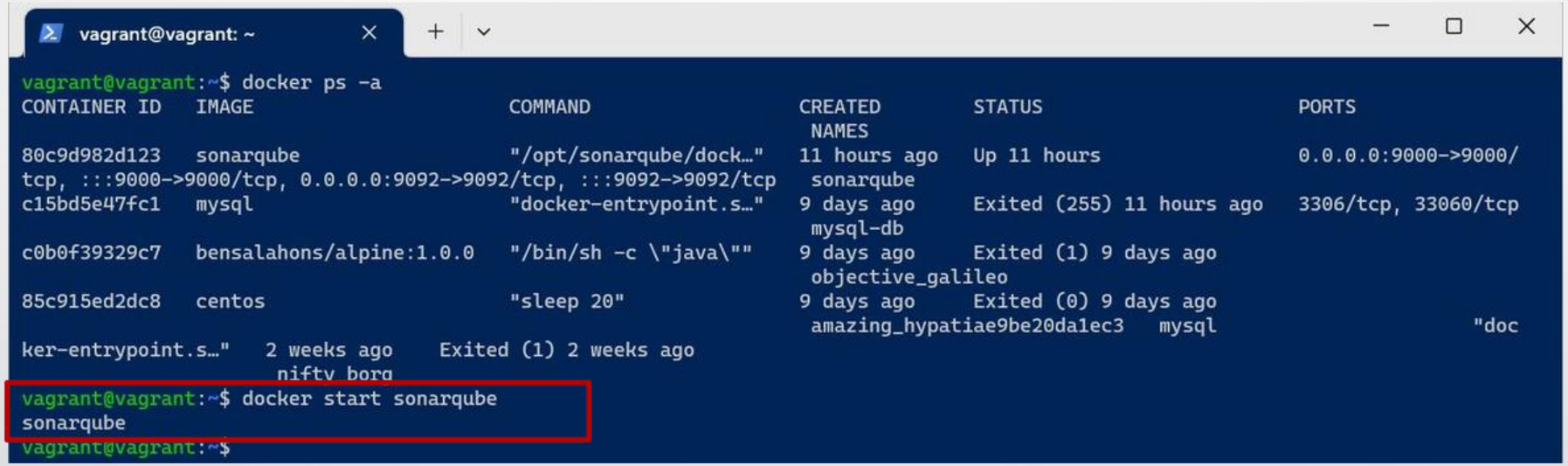
› Vous pouvez changer le mot de passe par sonar par exemple : (admin/**sonar**).

SonarQube - Utilisation

ATTENTION :

›Si vous arrêtez le conteneur SonarQube (en utilisant CTRL+C ou en arrêtant la VM), évitez de lancer à nouveau docker run sur l'image sonarqube, car cela créerait un nouveau conteneur.

Au lieu de cela, utilisez simplement la commande suivante pour redémarrer le conteneur existant : `docker start <NAME_CONTAINER ou ID_CONTAINER>`



SonarQube – Utilisation avec Jenkins

- Pour mettre en place l'analyse d'un projet Git avec SONARQUBE en utilisant Jenkins, suivez les étapes suivantes :

  - Accédez à Jenkins via l'URL http://<ip-vm>:8080

  - Dans le pipeline Jenkins existant, ajoutez un nouveau stage au script Groovy pour récupérer le code de votre projet depuis GitHub (assurez-vous de spécifier l'URL de votre projet)

```
stage ('GIT') {
    steps {
        echo "Getting Project from Git";
        ....................................
    }
}
```

SonarQube – Utilisation avec Jenkins

- Exécutez les commandes Maven **clean** et **compile** pour nettoyer et compiler le code du projet que vous avez récupéré depuis Git (utilisez la commande **sh ' '** correspondante).

```
stage('MVN CLEAN') {
    steps {
        sh '.................'
    }
}

stage('MVN COMPILE') {
    steps {
        ......................
    }
}
```

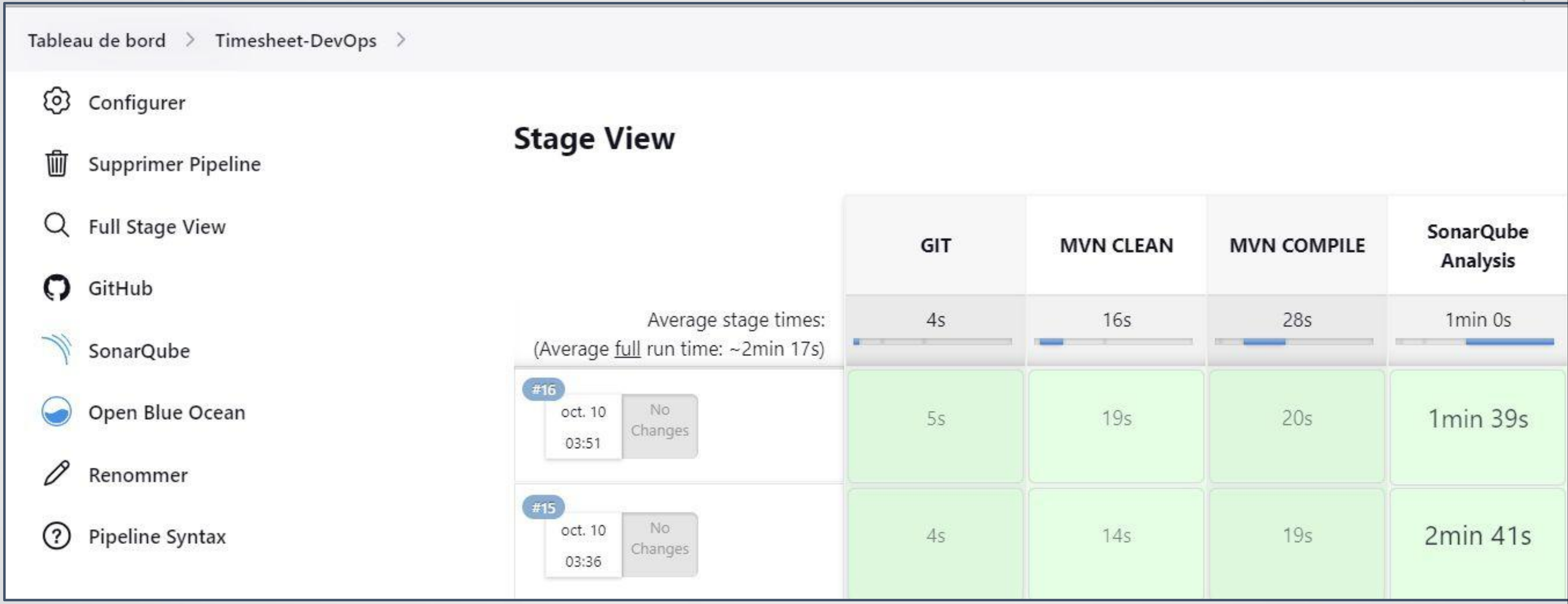
SonarQube – Utilisation avec Jenkins

- Lancez la commande Maven **sonar:sonar** pour analyser la qualité du code avec SonarQube et envoyez le rapport au serveur SonarQube (assurez-vous de fournir le mot de passe de votre instance SonarQube).

```
stage('MVN SONARQUBE') {
    steps {
        ..................................
    }
}
```

- Lancez le Job via **Jenkins**.

SonarQube – Utilisation avec Jenkins



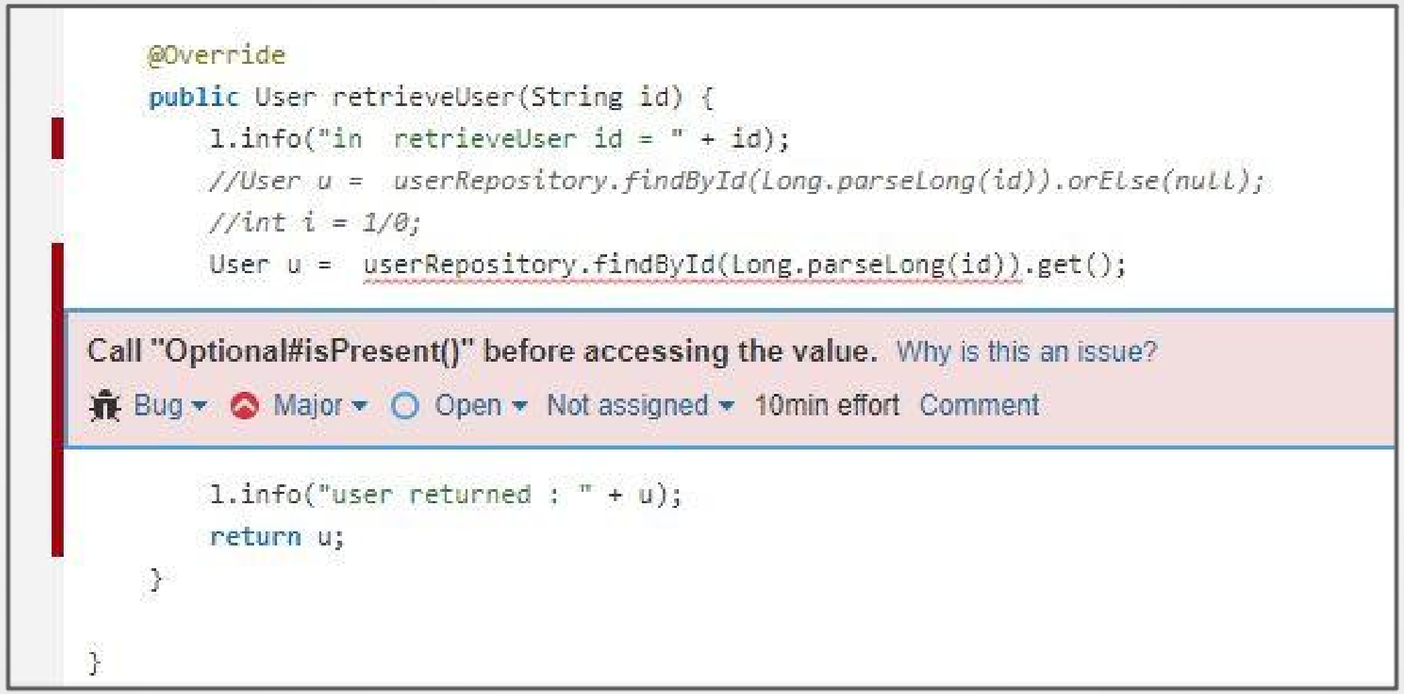
SonarQube – Analyse des résultats

- Accédezà*http://<adresse_ip_vm>:9000* et consultez les résultats de l'analyse de votre projet.

- Sur l'interface, cliquez simplement sur le projet "timesheet-devops" pour accéder aux détails complets de cette analyse. SonarQube détecte automatiquement le **langage** utilisé (**Java + XML**
- dans notre cas). SonarQube vérifie si le code contient du **code dupliqué** (ce qui peut être
- source d'erreurs). Dans notre cas, le pourcentage de duplications est de 0%.

SonarQube - Analyse des résultats

- **Bug:**Undéfaut dans lecodesusceptible de provoquer un comportement indésirable de l'application, tel qu'une exception de pointeur nul, dans notre situation :



- **Vulnerability :** Une faiblesse de sécurité dans notre code. **Hotspots Reviewed (Zones à**
- **vérifier) :** Parties du code qui nécessitent un examen pour s'assurer qu'elles ne présentent pas de failles de sécurité.

SonarQube – Analyse des résultats

- **SonarQube** indique si le code a été soumis à des outils de test, tels que JUnit.

- SonarQube lui-même n'effectue pas l'analyse, mais se repose sur d'autres outils, tels que JaCoCo. C'est pourquoi notre projet affiche un taux de couverture (**Coverage**) de 0 %, car JaCoCo n'a pas été intégré.

SonarQube – Analyse des résultats

- **Code Smells** (*Mauvaises pratiques de codage*) : Ce n'est pas une anomalie du code (bug), mais plutôt du code qui peut compliquer la tâche de l'équipe de développement ou de support, par exemple en raison de commentaires excessifs ou des imports inutilisés.

SonarQube – Analyse des résultats

Quality Gate
Passed

Enjoy your sparkling clean code!

New Code | Overall Code

Reliability
1 Bugs — C

Maintainability
30 Code Smells — A

Security
2 Vulnerabilities — D

Security Review
0 Security Hotspots — A

Coverage
0.0% Coverage
Coverage on 168 Lines to cover
– Unit Tests

Duplications
0.0% Duplications
Duplications on 547 Lines
0 Duplicated Blocks

"Apprendre par le projet, c'est découvrir par l'action, créer par la compréhension, et réussir par la persévérance."

OBJECTIFS DE DÉVELOPPEMENT DURABLE

nous supportons les objectifs de développement durable

ESPRIT – UP ASI (Architecture des Systèmes d'Information)

Bureau E204