



Module 5

Intégration d'outils avancés

Contenu du module

- Un serveur central pour le stockage des livrables
 - La génération des livrables avec Jenkins
 - Intégration et configuration de Jenkins avec un serveur Archiva, Nexus
- Les outils de test de la qualité de code
 - La suite d'outils SonarQube
- Le déploiement dans des conteneurs
 - Principes de la conteneurisation
 - Mise en œuvre de Docker
 - Support de Docker dans Jenkins

Le stockage des livrables générés

- Les livrables générés par l'intégration continue doivent être archivés.
 - Ils peuvent être des dépendances pour d'autres projets !
- Une plateforme d'intégration continue doit donc prévoir un serveur **référentiel de livrables**.
- Il existe aujourd'hui sur le marché principalement trois logiciels qui assurent ce rôle :
 - Apache Archiva
 - Artifactory
 - Sonatype Nexus

L'exemple de Nexus Repository

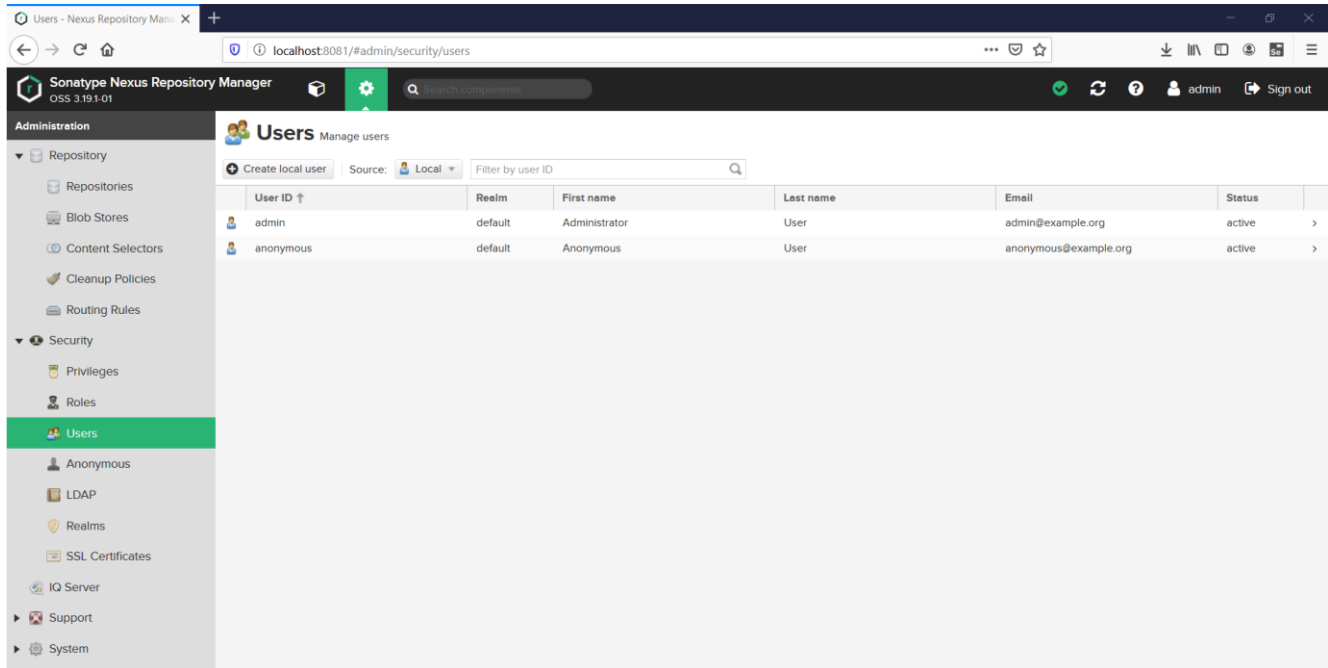
- **Nexus Repository** est le gestionnaire de référentiels développé par la société Sonatype.
- Il ne nécessite pas de base de données pour gérer tous les artefacts dans les référentiels mais utilise le système de fichiers standard.
- Nexus Repository gère un index en temps réel sur chaque référentiel afin de rendre optimale la recherche d'artefacts.
- Il existe deux éditions du produits :
 - **Nexus Repository OSS** sous licence Open Source.
 - **Nexus Repository Pro** sous licence commerciale.
- <https://blog.sonatype.com>

Installation de Nexus Repository

- Télécharger l'archive de Nexus Repository
 - <https://fr.sonatype.com>
- Décompresser l'archive
- Une fois l'archive décompressée, le serveur peut se lancer à partir de l'exécutable présent dans le répertoire **bin/** de l'arborescence d'installation :
 - Windows
 - nexus.exe /run
 - Unix/Linux
 - ./nexus run
- Le serveur se met en écoute sur le port 8081.
 - L'interface d'administration est alors disponible à l'adresse <http://<serveur-nexus>:8081>
 - Notes :
 - Le compte par défaut (administrateur) est utilisable avec l'identifiant **admin**
 - Il est indiqué comment trouver le mot de passe par défaut.
 - Il sera nécessaire de modifier le mot de passe à la première connexion.

Configuration de Nexus

- Avant de pouvoir utiliser le référentiel de livrables Nexus Repository, il est nécessaire de créer un compte utilisateur.
 - Ce dernier va servir à Jenkins pour se connecter à Nexus afin d'y publier les livrables



The screenshot shows the Sonatype Nexus Repository Manager web interface. The browser address bar displays `localhost:8081/#admin/security/users`. The page title is "Sonatype Nexus Repository Manager OSS 3.19.1-01". The left sidebar shows the "Administration" menu with "Users" selected. The main content area is titled "Users" and "Manage users". It includes a "Create local user" button, a "Source" dropdown set to "Local", and a "Filter by user ID" search bar. Below this is a table listing users:

User ID ↑	Realm	First name	Last name	Email	Status
admin	default	Administrator	User	admin@example.org	active
anonymous	default	Anonymous	User	anonymous@example.org	active

Publier un livrable dans Nexus

- Il faut créer un dépôt pour un format donné sur le serveur Nexus
 - Maven, NPM, RAW, ...
- Via l'API REST de Nexus
- Via Maven et le goal deploy
 - La configuration du dépôt doit être spécifiée dans le `pom.xml`
- Via une tâche spécifique d'un projet Free-style utilisant l'un des plugins Jenkins pour Nexus :
 - Nexus Artifact Uploader
 - Nexus Platform

Adaptation du projet Maven

- Jenkins va utiliser la phase **deploy** du cycle de vie par défaut de Maven pour publier vers Nexus Repository.
 - Il est donc nécessaire d'adapter le POM du projet pour indiquer les informations de localisation du serveur.
- Il faut commencer par créer une entrée **<server>** dans le fichier **M2_HOME/conf/settings.xml** :

```
<settings>
...
<server>
  <!-- L'id doit impérativement être 'nexus' -->
  <id>nexus</id>
  <username>deployment</username>
  <password>secret</password>
</server>
...
</settings>
```


Configuration du projet Maven

- Le POM du projet doit ensuite être adapté pour indiquer où se trouvent les emplacements de stockage des livrables en développement (SNAPSHOT), et des livrables versionnés (RELEASE).

```
<distributionManagement>
  <repository>
    <id>nexus</id>
    <name>Référentiel Releases</name>
    <url>http://serveur-nexus:8081/repository/maven-releases</url>
    <layout>default</layout>
  </repository>
  <snapshotRepository>
    <id>nexus</id>
    <name>Référentiel Snapshots</name>
    <url>http://serveur-nexus:8081/repository/maven-snapshots</url>
  </snapshotRepository>
</distributionManagement>
```

SonarQube

- **SonarQube** (précédemment **Sonar**) est un logiciel libre permettant de mesurer la qualité du code source en continu.
- Support de plus de vingt-cinq langages
 - Java, C, C++, Objective-C, C#, PHP, Flex, Groovy, JavaScript, Python, PL/SQL, COBOL...
- Permet le reporting sur :
 - L'identification des duplications de code
 - La mesure du niveau de documentation
 - Le respect des règles de programmation
 - La détection des bugs potentiels
 - L'évaluation de la couverture de code par les tests unitaires
 - L'analyse de la répartition de la complexité
 - L'analyse du design et de l'architecture d'une application
- Analyses entièrement automatisées. Intégration avec :
 - Maven, Ant, Gradle
 - Serveurs d'intégration continue (Jenkins, Hudson...).
- Extensible par des plugins

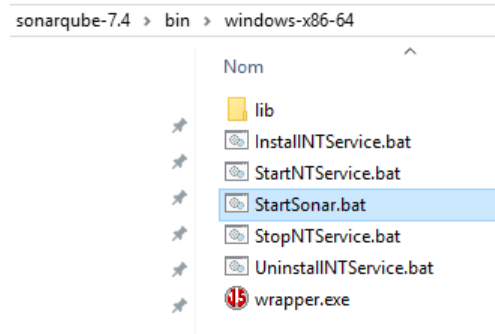
Mise en œuvre de SonarQube

- L'utilisation de SonarQube pour l'analyse de rapport suppose la mise en œuvre :
 - Du serveur **SonarQube**, qui génère les rapports agrégés à partir de différentes source de données
 - De l'analyseur **SonarQube Scanner** qui analyse les données et le code afin de permettre la production des rapports

Installation du serveur SonarQube

- Prérequis
 - Installation d'un JDK
- Téléchargement de l'archive ZIP du serveur à partir de <http://www.sonarqube.org/downloads/>
- Décompression de l'archive **sonarqube-x.x.x.zip**
- Le lancement du serveur se fait à partir de
 - sonarqube-x.x.x/bin/<platform>/StartSonar.xxx

» Sous Windows



Le serveur SonarQube

- Une fois démarré, le serveur propose une interface de gestion à l'adresse
 - `http://<serveur>:9000`

The screenshot displays the SonarQube web interface. At the top, a dark navigation bar contains the SonarQube logo, links for Projects, Issues, Rules, Quality Profiles, and Quality Gates, a search bar with the placeholder text 'Search for projects, sub-projects and files...', and a 'Log in' button. Below the navigation bar, the main content area is divided into two sections. The left section, titled 'Continuous Code Quality', features a 'Log in' button and a 'Read documentation' button. The right section, titled 'Projects Analyzed', shows a large '0' and a list of metrics: Bugs, Vulnerabilities, and Code Smells. Below these sections, a 'Multi-Language' section lists 20+ supported programming languages: Java, C/C++, C#, COBOL, ABAP, HTML, RPG, JavaScript, TypeScript, Objective C, XML, VB.NET, PL/SQL, T-SQL, Flex, Python, Groovy, PHP, Swift, and Visual Basic. At the bottom, a 'Quality Model' section explains the three types of issues: Bugs (code that is demonstrably wrong or highly likely to yield unexpected behavior), Vulnerabilities (raised on code that is potentially vulnerable to exploitation by hackers), and Code Smells (will confuse maintainers or give them pause, measured primarily in terms of the time they will take to fix).

SonarQube Scanner & Jenkins

- L'intégration de SonarQube Scanner et Jenkins peut se faire de différentes manières
 - Installation locale de SonarQube Scanner et invocation en ligne de commande depuis un projet Free-style
 - Utilisation du plugin Jenkins **SonarQube Scanner**
 - <https://docs.sonarqube.org/latest/analysis/scan/sonarscanner-for-jenkins/>

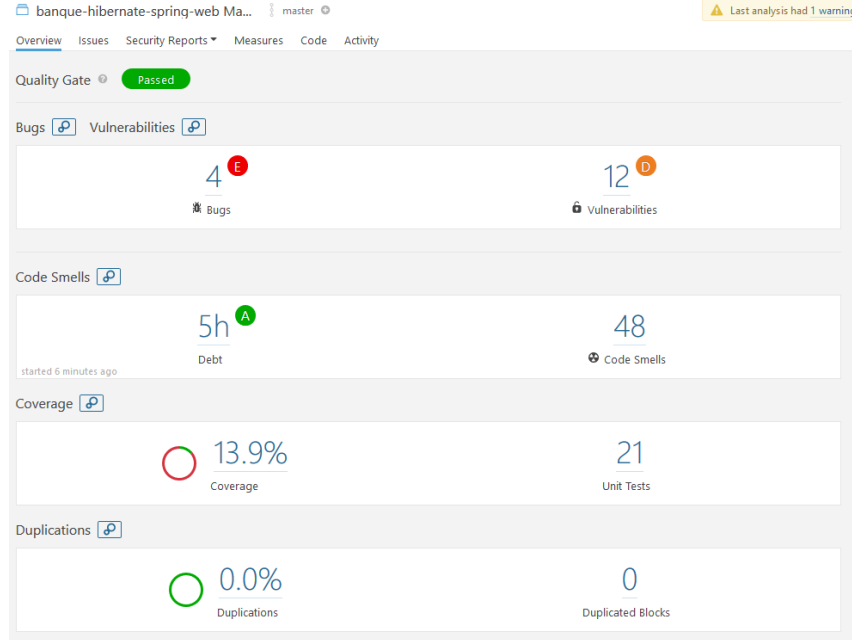
SonarQube et les projets Maven

- Une fois le serveur SonarQube prêt, il faut donner la possibilité à Maven de communiquer avec lui pour permettre la publication des rapports
- Edition du fichier **settings.xml** de Maven
 - Dans la section **profiles** du fichier, ajouter

```
<profile>
  <id>sonar</id>
  <activation>
    <activeByDefault>true</activeByDefault>
  </activation>
  <properties>
    <sonar.host.url>
       http://localhost:9000
    </sonar.host.url>
  </properties>
</profile>
```

- Puis lancer la commande **mvn sonar:sonar**
 - Le projet doit apparaître dans le serveur SonarQube

Le projet dans SonarQube



- **JaCoCo** et **SonarQube** s'intègrent évidemment avec les serveurs d'intégration continue

La conteneurisation

- Fourniture d'un environnement équivalent aux machines virtuelles (isolation, configuration...) mais moins volumineux en espace disque
 - Seules l'application et ses dépendances (autres applications et bibliothèques) sont dans l'image.
- Invariance
 - Chaque exécution d'un container depuis la même image crée les mêmes processus dans le même état de départ (si on n'utilise pas de volume).
- Environnement de développement/test plus facile à distribuer que les VM
- En production
 - Déploiement, initialisation et mise à jour de version rapide
 - Souvent présent dans un système de déploiement « DevOps »
 - Mise à l'échelle (augmentation/réduction du nombre d'instances d'un container) simple dans un cluster

Docker

- Docker est une plate-forme de conteneurs logiciels virtualisés qui permet d'empaqueter des applications et leurs dépendances systèmes afin de les exécuter sur n'importe quel serveur Linux ou Windows.
 - Technique qui permet d'exécuter des processus de façon « isolée »
 - Fonctionnalité de LXC (container Linux Standard) dont Docker étend le format grâce à une API de haut niveau
- L'approche d'une architecture applicative en containers est celle des micro-services.
- <https://www.docker.com/>



Pourquoi déployer dans Docker ?

- Pour permettre d'exécuter des tests où le déploiement de l'application dans son environnement est nécessaire
 - Tests d'intégration
 - Liaison avec une base de données, Serveur Web, ...
 - Tests fonctionnels
 - Tests d'IHM Web : L'application doit être disponible en HTTP !
- Pour faciliter l'initialisation et le nettoyage de l'environnement de test
 - Chaque conteneur est créé avec le même état de départ
 - Le jeu de données peut donc être intégré dans le conteneur
 - La suppression du conteneur implique la perte des données modifiées par le test
- Pour livrer un conteneur autonome en (pré) production
 - Vers une architecture Cloud ou On-Premise

Déployer une application dans Docker

- Déployer des applications construites avec Jenkins dans un conteneur
- Prérequis :
 - Disposer d'une installation de Docker
- Intégration entre Jenkins et Docker peut se faire :
 - En ligne de commande (avec un projet "free-style")
 - Grâce au plugin « Docker Pipeline » dans le cas d'un projet de pipeline
- Pour créer un job de déploiement d'un conteneur Docker, il faut :
 - Une étape qui crée l'image Docker publiant l'application
 - Un fichier Dockerfile qui décrit cette image (dans le SCM !)
 - La commande Docker qui construit l'image
 - `docker build -t <image:tag> <build_context>`
 - Une étape qui lance le conteneur avec l'application
 - Exécution de la commande Docker démarrant le conteneur

Travaux Pratiques



www.eni-service.fr