

Développement MAVEN

MAVEN



Objectifs pédagogiques

- Savoir structurer un projet autour de Maven et gérer les dépendances

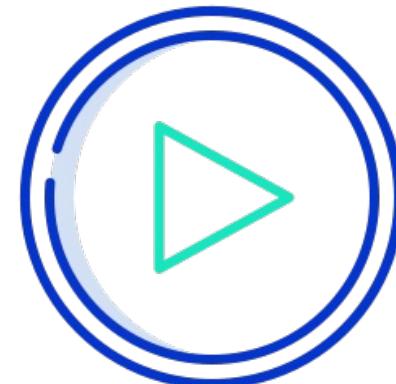


Sommaire

- Maven
- Présentation, historique
- Principes de Maven
- Installation
- Les versions
- Architecture
- Notion de build
- Le Project Object Model (POM)
- Principe de fonctionnement et d'exécution
- Goal
- Convention plutôt que configuration

Développement Maven

INTRODUCTION



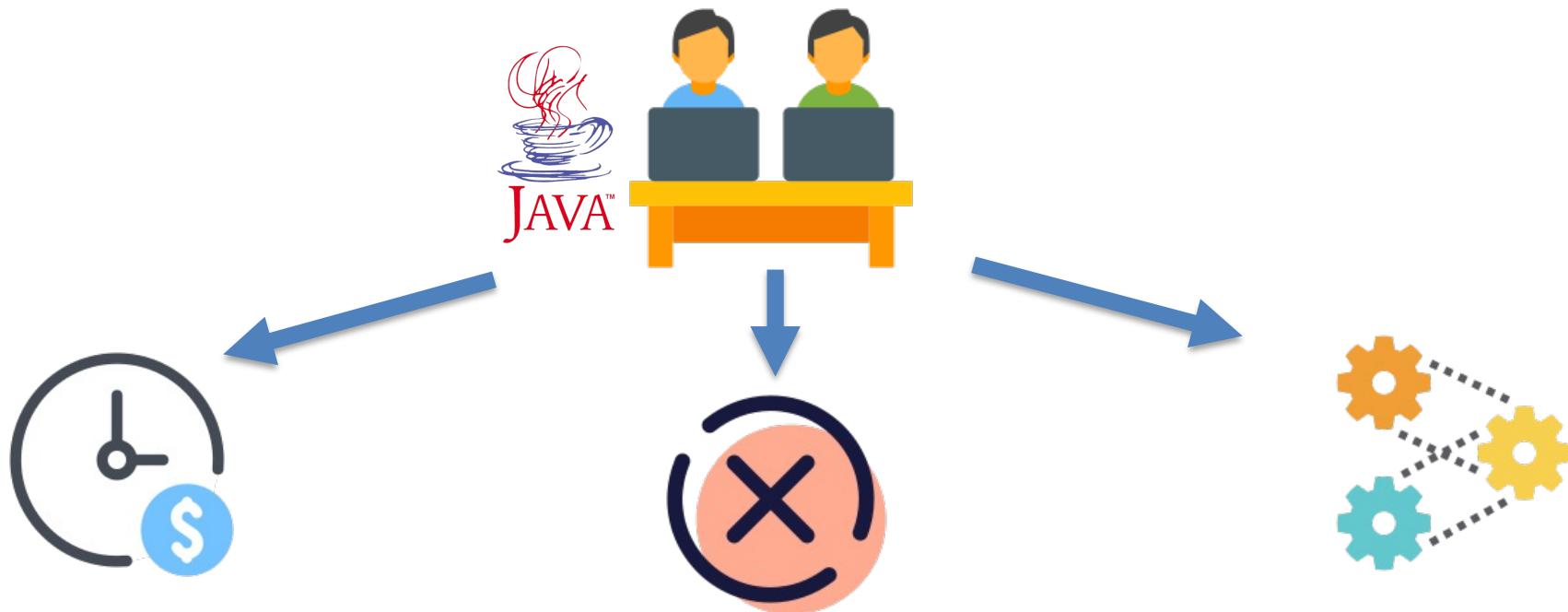
Introduction

Maven

- Outil de gestion de construction pour les projets Java
- Officiellement présenté en 2004 par Apache Software Foundation
- Créé par Jason Van Zyl (CTO & Fondateur de Sonatype, Plexus, Maven)
- Logiciel open-source, développé et maintenu par la communauté Apache Maven
- Utilisé pour automatiser la construction, la gestion des dépendances, les tests, la génération de documentation pour les projets Java

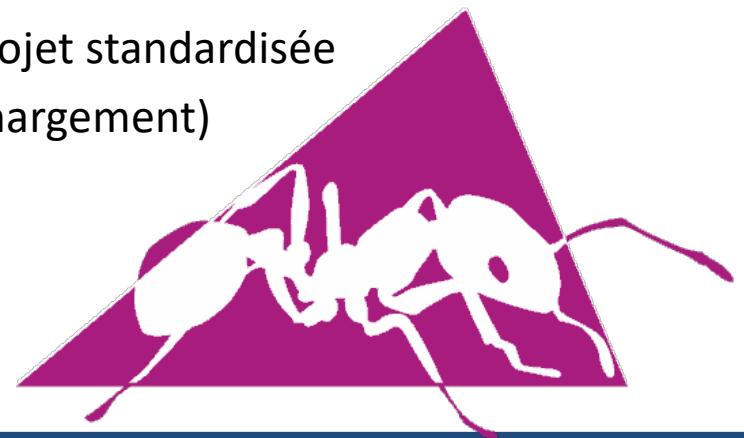
Historique 1/3

- **1990 : Une volonté forte d'évolution**
 - Les constructions des projets sont fastidieuses (manuelles, étape par étape)
 - Exécution de commandes pour compiler, gérer les dépendances/fichiers binaires
 - Approche manuelle pouvant causer des erreurs (temps et efforts conséquents)



Historique 2/3

- **2000 : Invention et utilisation de Apache Ant**
 - Outil de construction flexible et extensible
 - Basé sur un modèle de script XML personnalisé
 - Description des tâches de construction automatisées
- **2002 : Invention de Maven**
 - Cohérence des projets (Avec Ant, Chaque projet avait sa propre structure)
 - Réponds aux lacunes de Apache Ant
 - Configuration détectée par rapport à une structure de projet standardisée
 - Système de gestion de dépendances (résolution et téléchargement)
 - Facilite le concept de l'intégration continue (CI)



Historique 3/3

Maven a été créé pour résoudre les limitations d'Ant

- Simplification de la gestion des dépendances
- (CoC) Convention over Configuration
 - Configurations déduites en se basant sur une structure de projet standardisée
- Cohérence et réutilisation du code entre les projets
- Facilitation de l'intégration continue grâce à des fonctions intégrées

Développement Maven

PRINCIPES MAVEN

Principes de Maven 1/3

Convention over Configuration

- Maven favorise l'utilisation de conventions plutôt que de configurations explicites
- Utilise une structure de projet standardisée/des conventions pour réduire la complexité
- **Permet aux développeurs de se concentrer sur le code plutôt que sur la configuration**

Gestion des dépendances

- Maven fournit un système de gestion des dépendances robuste
- Les dépendances sont spécifiées dans le fichier de configuration **pom.xml**
- Résout les dépendances, les télécharger à partir de **repositories** et les inclue dans le projet
- Assure la cohérence entre les environnements de développement

Principes de Maven 2/3

Cycle de vie de construction prédéfini

- Définition d'un cycle de vie avec des phases prédéfinies
- Chaque phase est associée à un objectif spécifique
- Chaque phase est exécutable via une commande Maven standard
- Facilite l'automatisation et les tâches de construction

Gestion des plugins

- Utilisation de plugins pour étendre les fonctionnalités du processus de construction
- Plugins utilisables pour générer des rapports, exécuter des tests, déployer les artefacts
- Maven offre la possibilité de créer ses propres plugins

Principes de Maven 3/3

Documentation complète

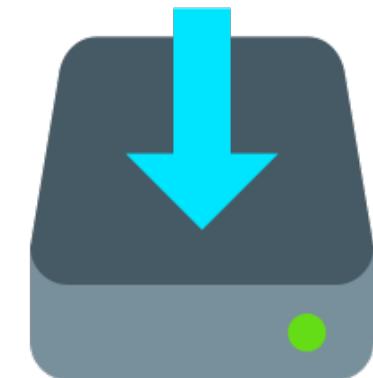
- Maven fournit des mécanismes pour documenter le projet sur plusieurs aspects
- **pom.xml** précise les dépendances, les plugins avec des commentaires et descriptions
- Maven propose des rapports pour générer des informations utiles sur le projet (**mvn site**)
- La génération de javadoc au format HTML est intégré nativement

Intégration continue et déploiement (CI/CD)

- Maven est compatible avec les serveurs d'intégration continue (**Jenkins, Bamboo, etc**)
- Les fonctionnalités (**tests, rapports, déploiement**) harmonisent la construction des pipeline

Développement JAVA

INSTALLATION & PROJET



Installation de Maven

- Maven est désormais installé avec votre IDE Java
- Sinon :
 - Maven est un outil Java, il nécessite le JDK (Java Development Kit)
 - <https://maven.apache.org/download.cgi>
 - Suivez les instructions d'installations spécifiques à votre système d'exploitation
 - Disponible sur Windows, MacOS, Linux, etc.
 - Une fois installé, vous pouvez vérifier la version de Maven avec :

```
mvn --version
```

Développement JAVA

LES VERSIONS



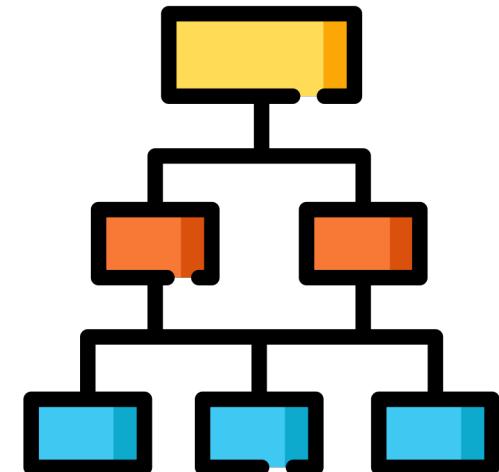
Les différentes versions Maven

- Maven 1.0 : Juillet 2004
- Maven 2.0 : Octobre 2005
- Maven 3.0 : Octobre 2009
- Maven 3.1 : Octobre 2013
- Maven 3.2 : Décembre 2013
- Maven 3.3 : Mars 2015
- Maven 3.4 : Mai 2016
- Maven 3.5 : Mai 2017
- Maven 3.6 : Février 2019
- Maven 3.6.1 : Octobre 2019
- Maven 3.6.2 à 3.6.5 : 2020
- Maven 3.8 : Avril 2021
- Maven 3.9 : Février 2023

<https://maven.apache.org/docs/history.html>

Développement JAVA

ARCHITECTURE



Architecture Maven 1/4

Repose sur plusieurs composants qui interagissent ensemble :

- pour gérer la construction, la gestion des dépendances et la documentation des projets
- **POM**
 - Fichier XML essentiel dans Maven qui décrit le projet et ses dépendances
 - Situé à la racine du projet
 - Fournit les instructions nécessaires à maven pour construire le projet
 - Spécifie les informations du projet (nom, version, dépendances, plugins, repositories)
- **Repositories**
 - Emplacements où les artefacts nécessaires au projet sont stockés et récupérés
 - Les artefacts sont les bibliothèques, plugins, etc
 - Maven utilise ces repositories pour résoudre les dépendances et télécharger les artefacts
 - Il existe un repository local et un repositories distants (tels que Maven Central)

Architecture Maven 2/4

- **Plugins**
 - Composants logiciels fournissant des fonctionnalités supplémentaires à Maven
 - Configurés dans le POM et utilisable pour des tâches pendant le cycle de vie du projet
 - Les plugins sont proposés par Maven ou développés spécifiquement pour le projet
- **Lifecycle**
 - Séquence prédéfinie de phases des tâches exécutées pour construire et gérer un projet
 - Chaque phase du cycle de vie est liée à un ensemble de tâches (**goals**) prédéfinies
 - Les tâches sont la compilation, les tests, l'empaquetage, l'installation, etc
 - Ces tâches sont exécutable en utilisant des commandes (comvn clean)

Architecture Maven 3/4

- **Gestion des dépendances**
 - Permet de spécifier les dépendances requises par le projet dans le POM
 - Maven résout automatiquement ces dépendances en se basant sur les informations du POM et les télécharge à partir des repositories spécifiés
 - Cela facilite le processus de gestion des dépendances et garantit que toutes les dépendances requises sont disponibles pour la construction du projet

Architecture Maven 4/4

- **Génération de la documentation**
 - Maven propose des plugins pour générer les rapports de documentation
 - documentation du code source, rapports de test, rapports de couverture, etc

Maven Javadoc Plugin : génère la documentation du code source au format Javadoc

Maven Surefire Plugin : exécute les tests unitaires et génère des rapports détaillés

Maven Failsafe Plugin : exécute les tests d'intégration et génère des rapports

Maven Jacoco Plugin : Génère des rapports de couverture de code

Maven Site Plugin : Génère un site web (doc, rapports de test/couvertures, dépendances, etc.)

Développement Maven

NOTIONS DE BUILD

Notion de Build 1/4

Maven fournit un système de construction automatisé

- **Nettoyage (mvn clean)**
 - Nettoie les artefacts générés lors des compilations précédentes (**target**)
- **Validation (mvn validate)**
 - Vérifie si le projet respecte les conditions préalables à sa construction (**pom.xml**)
 - Vérifie la structure du projet, des dépendances, des configurations, etc
- **Compilation (mvn compile)**
 - Étape de compilation du code situé dans le *package src/main/java*
 - Les résultats de la compilation sont placés dans le dossier **target/classes**

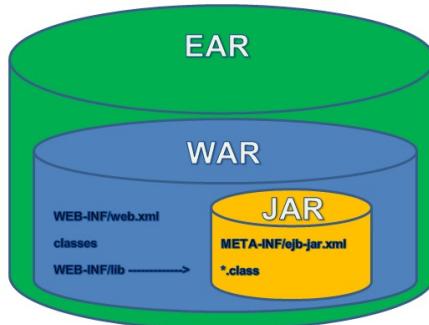
Notion de Build 2/4

- **Testing (mvn test)**

- Compile et exécute les fichiers de test du répertoire **src/test/java**
- Les résultats sont affichés dans la sortie console
- Les rapports de tests sont générés dans le dossier **/target/surefire-reports**

- **Empaquetage (mvn package)**

- Rassemble les fichiers compilés et les éventuelles ressources dans une archive
- Les formats sont : **Jar (Java Archive)**, **War (Web Archive)** ou **EAR (Enterprise Archive)**
- Les rapports de tests sont générés dans le dossier **/target/surefire-reports**



Notion de Build 3/4

- **Verification (mvn verify)**
 - Vérifie la qualité du projet après sa compilation et son empaquetage
 - Valide les règles de codage, l'analyse statique du code, les tests d'intégration
 - Permet de s'assurer que le projet respecte les normes de qualités et fiabilités
- **Installation (mvn install)**
 - Consiste à installer l'artefact dans le référentiel maven local
 - Le projet sera copié dans le dossier **.m2/repository/**
 - Permet de le rendre disponible pour d'autres projets locaux en tant que dépendances
- **Génération du site (mvn site)**
 - Permet de générer le site Maven contenant la doc, les rapports de test/couverture, etc
 - Le site Maven (pages html) est consultable dans le dossier **target/site**

Notion de Build 4/4

- **Déploiement (mvn deploy)**
 - Permet de déployer l'artefact binaire dans un référentiel Maven distant ou sur un serveur
 - Implique de rajouter une configuration pour le déploiement

Ces phases permettent de gérer efficacement le cycle de vie d'un projet

Développement JAVA

POM.XML



Le POM (Project Object Model) 1/3

- Représente la configuration de votre projet
 - Situé à la racine du projet (obligatoirement pour faire fonctionner Maven)
- Fichier XML qui décrit les détails du projet
 - Groupe du projet (**groupId**)
 - Nom du projet (**artifactId**)
 - Version
 - Dépendances
 - Plugins
 - etc
- Centralise la configuration du projet et assure la cohérence des dépendances

Le POM (Project Object Model) 2/3

Lors de la création d'un projet, les informations suivantes sont demandées :

- **groupId** est un identifiant unique pour une organisation
 - Souvent basé sur le nom de domaine inversé de l'organisation
- **artifactId** est le nom unique du projet
 - Correspond au nom du projet
- **version** est la version actuelle du projet
 - Utilisée pour suivre les différentes versions d'un projet au fil du temps

Le POM (Project Object Model) 3/3

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>
  <groupId>fr.sauvageboris</groupId>
  <artifactId>demo</artifactId>
  <version>1.0-SNAPSHOT</version>
  <name>demo</name>
  <packaging>war</packaging>

  <dependencies></dependencies>

  <build>
    <plugins></plugins>
  </build>

</project>
```

Les version 1/2

Maven permet de préciser la version de notre livrable dans le **pom.xml**

- Il existe plusieurs versions possibles

```
...  
<version>1.0-SNAPSHOT</version>
```

SNAPSHOT : Version en développement (Peut contenir des fonctionnalités en cours ou des corrections de bugs)

ALPHA : version expérimentale (Peut contenir des fonctionnalités non finalisées, des bogues/limitations)

BETA : version proche de la stabilité (Peut contenir des bogues avant d'être publiée en tant que version stable)

M1 / M2 : indique la version de jalon d'un projet (Milestone)

RELEASE : version stable et prête pour une utilisation générale (versions finales ou majeures d'un logiciel)

Les version 2/2

- Numéros de version conventionnels :
 - 1.0 : version initiale du projet
 - 1.0.1 : première mise à jour mineure
 - 1.1 : première mise à jour majeure
 - 2.0 : version majeure suivante
 - 2.1.3 : deuxième mise à jour mineure de la version 2.1
- Préfixes et suffixes :
 - 1.0-SNAPSHOT : version en cours de développement, pas encore publiée
 - 1.0-beta1 : version bêta 1
 - 1.0-rc1 : version de candidate à la sortie (release candidate) 1
 - 1.0-alpha, 1.0-alpha2, etc. : versions alpha (versions expérimentales)
 - 1.0-M1, 1.0-M2, etc. : versions de jalon (milestone versions)
 - 1.0-RELEASE : version finale/stable

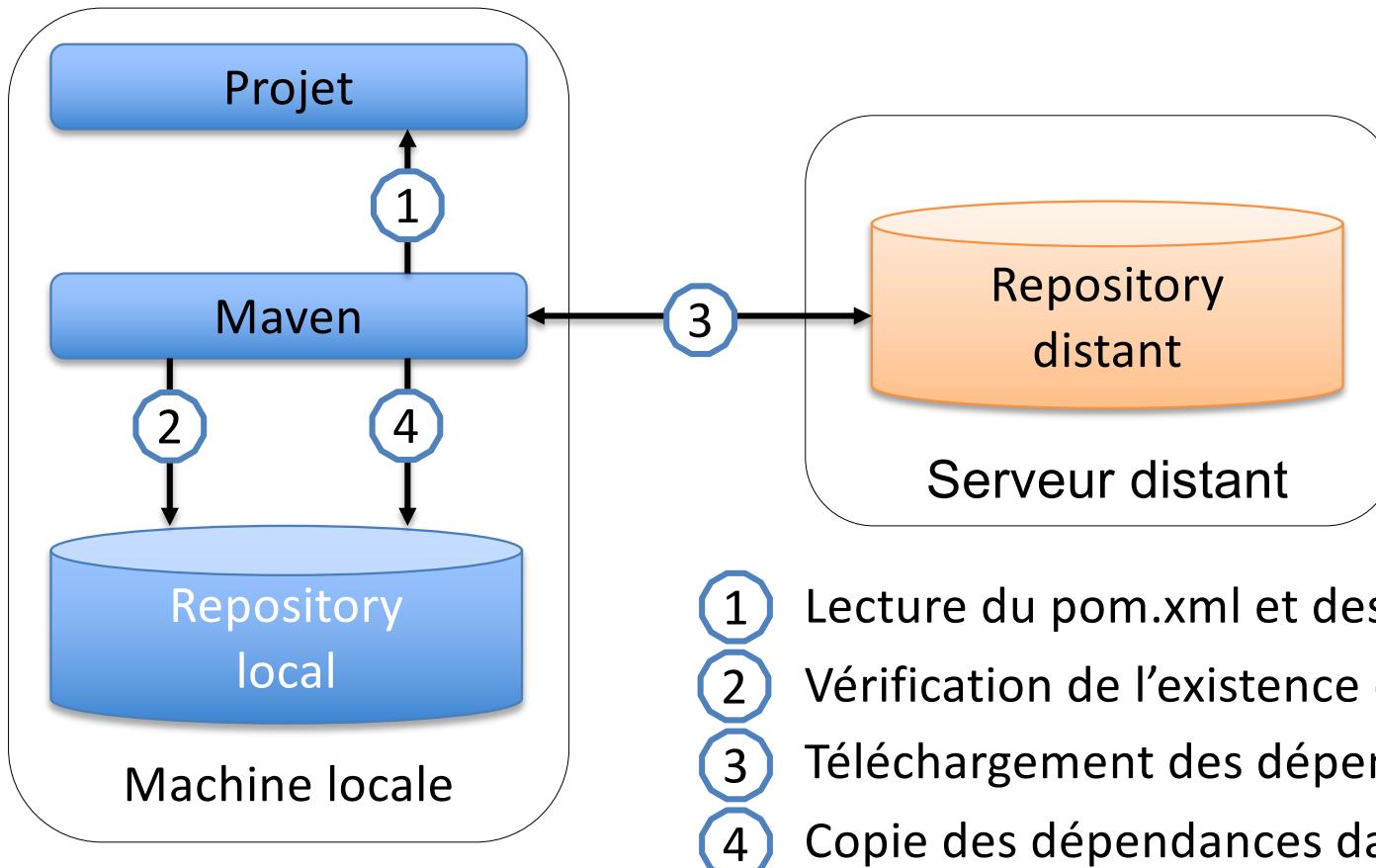
```
...<version>1.0-SNAPSHOT</version>...
```

Développement JAVA

REPOSITORY LOCAL ET DISTANT

Repository local et distant 1/2

- Maven utilise deux repository (local et distant)



Repository local et distant 2/2

- Le repository local
 - Situé sur votre machine locale (**.m2/repository**)
 - Stocke les dépendances des projets
 - Si une dépendance est introuvable, Maven télécharge depuis le répertoire distant
- Le repository distant
 - Serveur distant contenant les artefacts partagés par la communauté Maven
 - Configurés dans le fichier **settings.xml**
 - Le repository principal par défaut est le Maven Central Repository
 - Plusieurs repositories distants sont configurable dans le même **settings.xml**
 - Les repositories distants sont des serveurs Maven privés comme Jcenter, JitPack

Développement JAVA

LES COMMANDES MAVEN

Générer un projet 1/2

- **Maven permet de générer la structure de base d'un projet Java**
 - Avec les dossier et fichiers nécessaires

```
mvn archetype:generate -DgroupId=fr.sauvageboris -DartifactId=demo  
-DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
```

archetype:generate : Ce plugin archetype permet de générer un nouveau projet à partir d'un archetype

-DgroupId=fr.sauvageboris : identifiant du groupe du projet

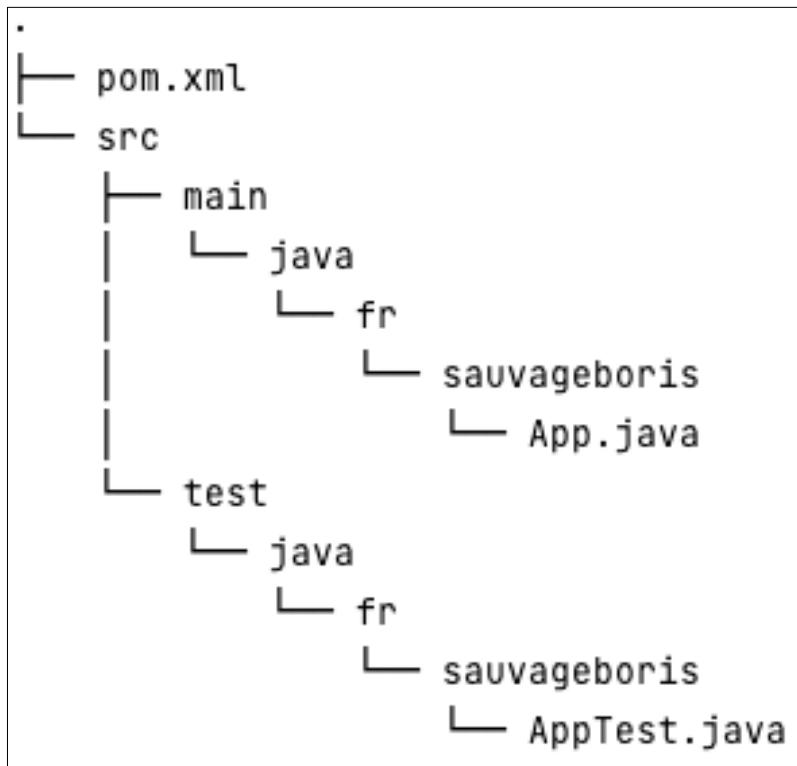
-DartifactId=demo : l'identifiant de l'artefact du projet

-DarchetypeArtifactId=maven-archetype-quickstart : spécifie l'identifiant de l'archetype utilisé

-DinteractiveMode=false : Désactive le mode interactif (Maven ne posera pas de questions)

Générer un projet 2/2

- Le projet généré aura la structure du projet ci-dessous :



Empaqueter un projet

- **Maven permet d'empaqueter un projet avec la phase package :**

```
mvn package
```

- En cas d'erreur de compilation à cause de votre version Java, ajouter le plugin :

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.8.1</version>
      <configuration>
        <source>1.8</source>
        <target>1.8</target>
      </configuration>
    </plugin>
  </plugins>
</build>
```



Exécuter un Jar

- Java permet d'exécuter des fichiers jar :

```
java -jar target/demo-1.0-SNAPSHOT.jar
```

- En cas d'erreur de lancement à cause d'un manifest absent, rajouter la configuration :

```
<build>
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-jar-plugin</artifactId>
  <version>3.2.0</version>
  <configuration>
    <archive>
      <manifest>
        <mainClass>fr.sauvageboris.App</mainClass>
      </manifest>
    </archive>
  </configuration>
</plugin>
</build>
```

Afficher les dépendances

- Il est possible d'afficher une arborescence détaillée des dépendances :

```
mvn dependency:tree -Dverbose
```

```
MacBook-Pro-de-Boris:demo sauvageb$ mvn dependency:tree -Dverbose
[INFO] Scanning for projects...
[INFO]
[INFO] -----< fr.sauvageboris:demo >-----
[INFO] Building demo 1.0-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- maven-dependency-plugin:2.8:tree (default-cli) @ demo ---
[INFO] fr.sauvageboris:demo:jar:1.0-SNAPSHOT
[INFO] \- junit:junit:jar:4.13.2:test
[INFO]   \- org.hamcrest:hamcrest-core:jar:1.3:test
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 0.834 s
[INFO] Finished at: 2023-05-09T18:55:00+02:00
[INFO] -----
```

Rafraîchir Pom.xml

- **Les dépendances Maven sont automatiquement actualisées**
 - Maven détecte automatiquement les changements lors de l'exécution d'une commande
- **Il est possible de rafraîchir immédiatement des dépendances et configurations :**
 - Vérifie si des MAJ sont disponibles pour les dépendances et de les télécharger si nécessaire

```
mvn clean install -U
```

```
mvn clean install --update-snapshots
```

Développement JAVA

DÉCRIRE SON PROJET

Introduction

Décrire son projet

- Dans le POM (Project Object Model), il est intéressant de décrire son projet :

```
<!-- Informations générales du projet -->
<name>exercice2</name>
<description>Description du projet nommé exercice 2</description>
<url>http://www.mon-site.fr/mon-app</url>

<!-- Informations sur l'organisation qui gère le projet -->
<organization>
    <name>Mon entreprise</name>
    <url>http://www.mon-entreprise.fr</url>
</organization>

<!-- Les licences utilisées dans le cadre d'une prestation de service par exemple -->
<licenses>
    <license>
        <name>Apache License, Version 2.0</name>
        <url>https://www.apache.org/licenses/LICENSE-2.0.txt</url>
    </license>
</licenses>
```

Développement JAVA

CI/CD

Introduction

CI/CD : Continuous Integration/Continuous Deployment

- Approche qui vise à automatiser le processus de livraison des logiciels
 - Pratique utilisée dans l'industrie du développement logiciel
- Améliore la qualité, la rapidité et la fiabilité des déploiements logiciels

Intégration Continue 1/2

Consiste à intégrer régulièrement et fréquemment le code dans un référentiel partagé

Les étapes sont souvent les suivantes :

1. Écriture du code source via un IDE (Eclipse, IntelliJ, etc)
2. Maven récupère les dépendances depuis Nexus Repository
3. Envoie du code sur le serveur Gitlab
4. Le serveur d'intégration continue Gitlab CI/CD détecte des changements sur Gitlab
5. Jenkins déclenche un pipeline (compilation, tests, analyse statique)
6. Le serveur SonarQube récupère les analyses statiques

CI/CD

Intégration Continue 2/2

Nexus



Code source



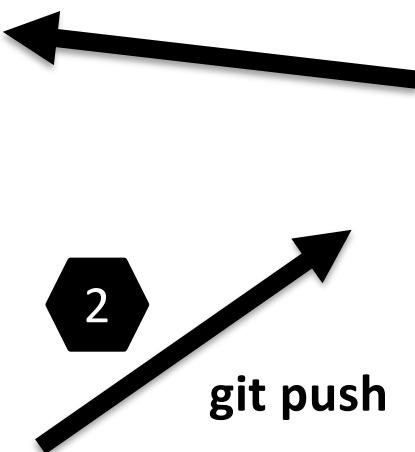
1



Serveur Gitlab

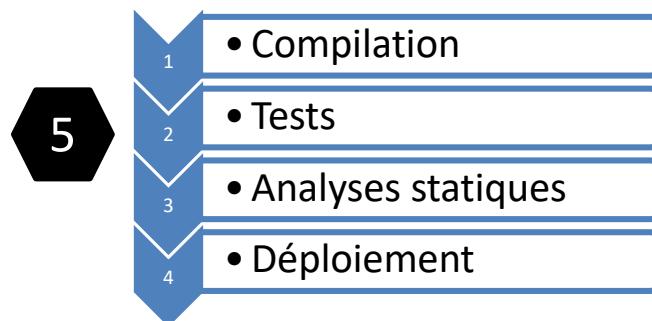
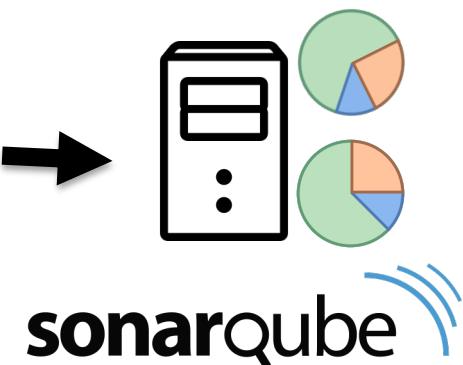


3



git push

4



Déploiement Continu 1/2

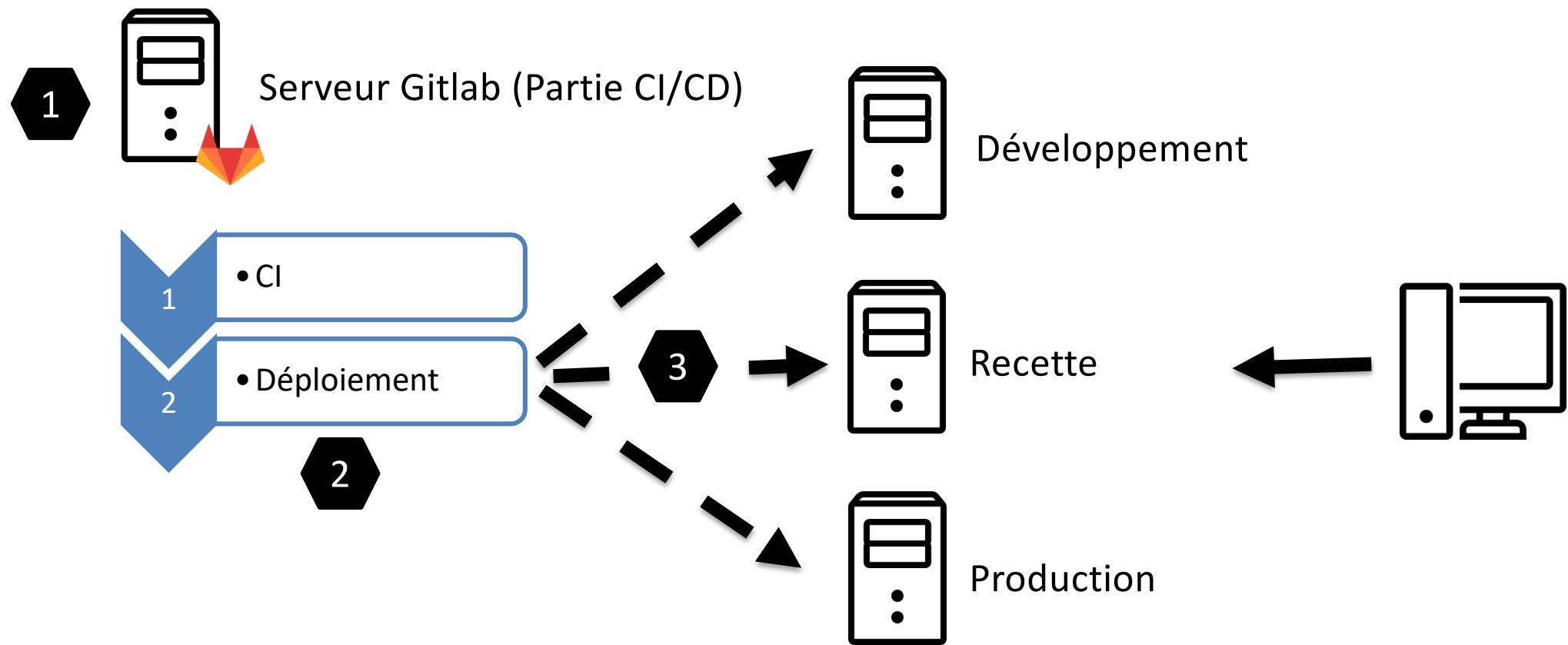
Consiste à automatiser la livraison du logiciel aux environnements de production

Les étapes sont souvent les suivantes :

1. L'intégration continue s'est déroulée avec succès
2. Le code est transformé en livrable (.jar ou .war ou .ear ou image docker)
3. Le livrable est déployé vers un environnement de test, pré-production, production
4. Des outils comme Ansible, Docker ou Kubernetes peuvent être utilisés

CI/CD

Intégration Continue 2/2



Développement JAVA

NEXUS



Présentation

- Plateforme de gestion de dépôts
- Permet d'héberger des artefacts
 - Ces artefacts sont des composants, générés par exemple au build d'un projet



Développement JAVA

SONARQUBE

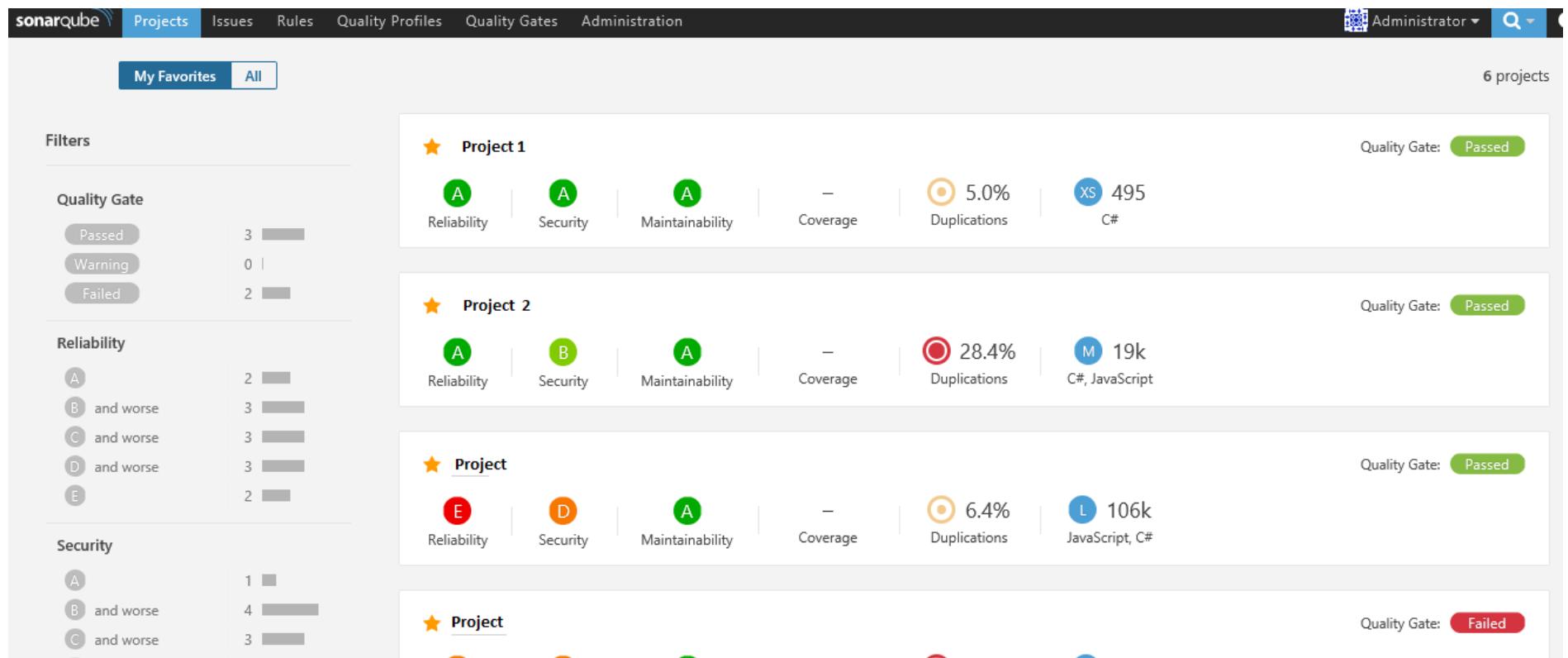


Présentation 1/2

- Plateforme d'analyse statique du code open-source
- Offre des fonctionnalités avancées pour l'inspection continue de la qualité du code
- Facilite l'identification et la résolution des problèmes de code



Présentation 2/2



Analyse statique du code 1/4

- Effectue une analyse statique du code source
 - pour détecter les violations des règles de qualité et de sécurité
- Identifie les erreurs, les vulnérabilités et les mauvaises pratiques de codage
- Permet de maintenir un code propre, lisible et maintenable

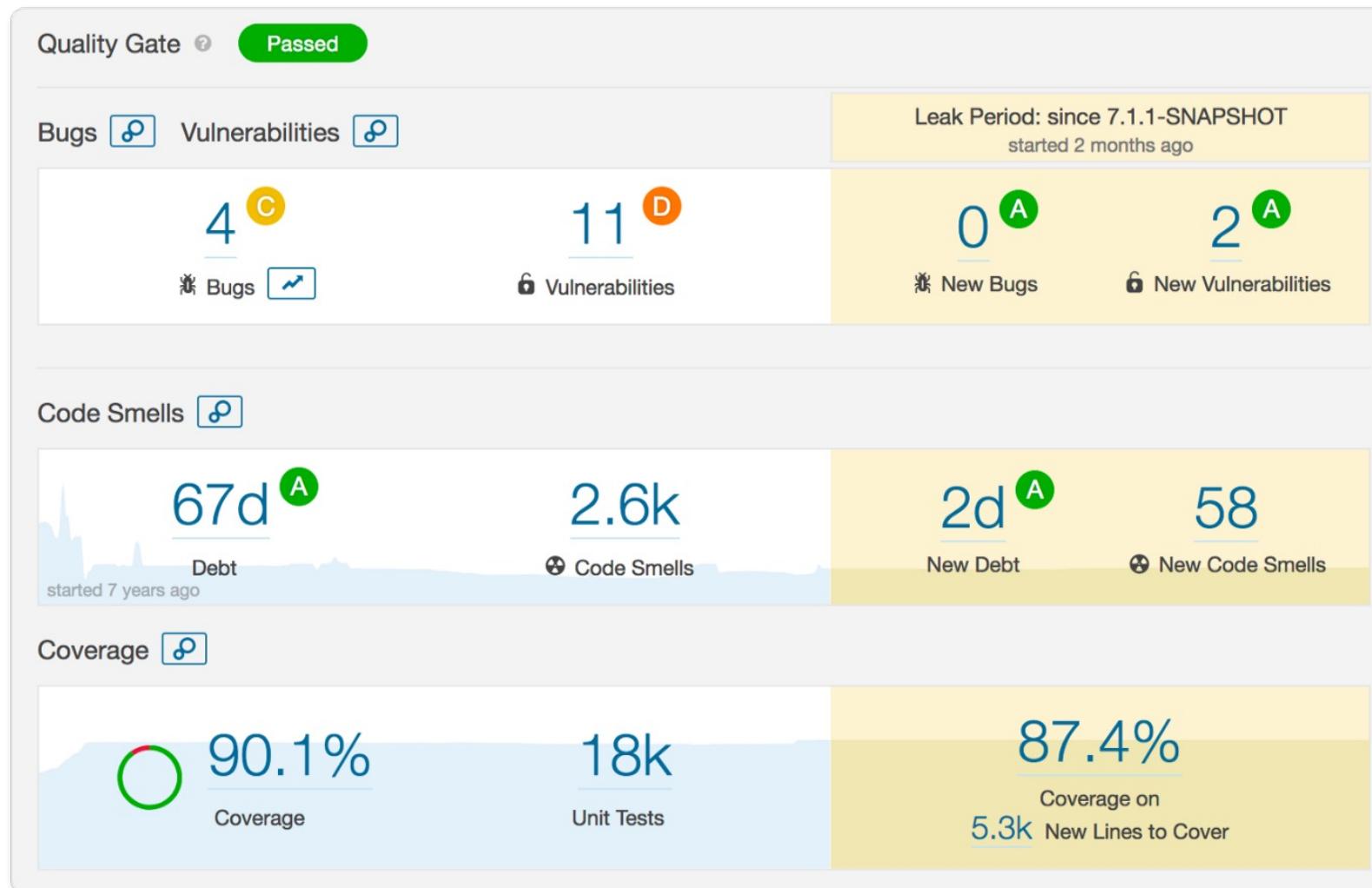


Analyse statique du code 2/4

- Génère des tableaux de bord et des rapports détaillés sur la qualité du code
- Fournit des indicateurs clés
 - tels que la couverture de code, la duplication, la complexité cyclomatique, etc
- Aide à surveiller l'amélioration continue de la qualité du code

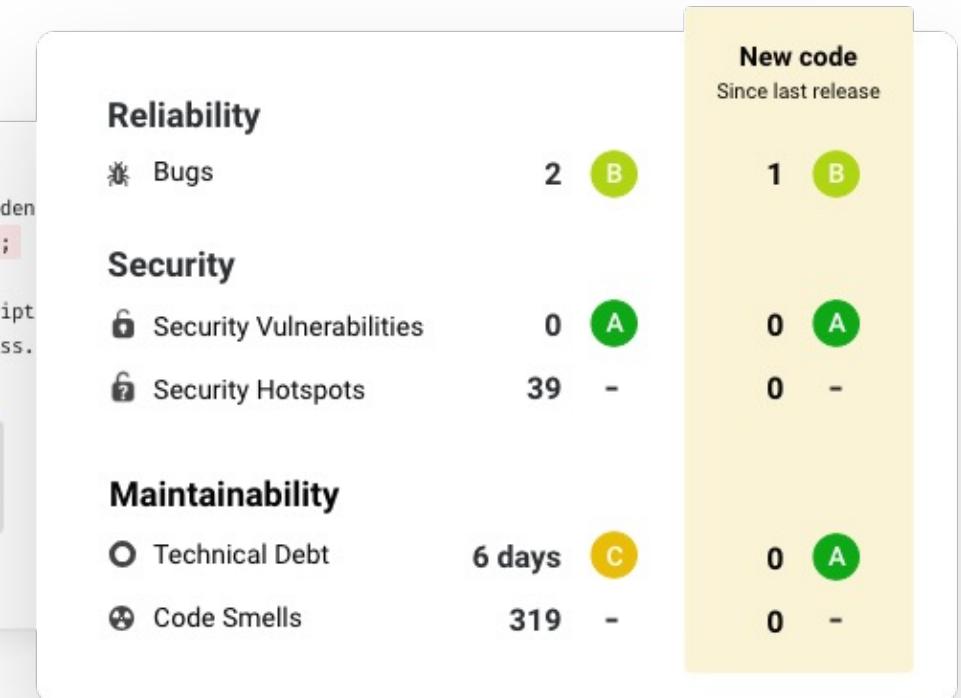


Analyse statique du code 3/4



Analyse statique du code 4/4

```
246 if (Provider.class == roleTypeClass) {  
247     Type providedType = ReflectionUtils.getLastTypeGenericArgument(dependen  
2 Class providedClass = 1 ReflectionUtils.getTypeClass(providedType);  
249  
250     if (this.componentManager.hasComponent(providedType, dependencyDescript  
251         || 3 providedClass.isAssignableFrom(List.class) || providedClass.  
  
A "NullPointerException" could be thrown; "providedClass" is nullable here.  
Bug Major cert, cwe  
252         continue;  
253     }  
254 }
```



Intégration continue

- S'intègre facilement aux outils d'intégration continue
 - tels que Jenkins, GitLab CI/CD, et Bamboo
- Automatise les analyses de code lors des builds
 - feedback rapide sur la qualité du code
- Facilite l'intégration de l'analyse de qualité dans vos pipelines CI/CD



Gestion des problèmes de sécurité

- Propose des règles de sécurité prédéfinies
 - pour identifier les vulnérabilités connues dans le code source
- Aide à prévenir les failles de sécurité et les problèmes de conformité
- Permet de renforcer la sécurité du code tout au long du cycle de développement



Avantages de SonarQube

- Amélioration de la qualité du code et de la maintenabilité
- Réduction des bugs et des vulnérabilités de sécurité
- Gain de temps en identifiant rapidement les problèmes de code
- Aide à respecter les bonnes pratiques de développement
- Facilite la collaboration entre les développeurs et les équipes de qualité



Développement JAVA

GITLAB



Présentation 1/3

- Initialement pensé comme un outil de collaboration pour le développement informatique
 - La première version a été développée par les développeurs Ukrainiens Valery Sizov et Dmitriy Zaporozhets
- À l'origine, il n'existait qu'une version entièrement gratuite et open-source de Gitlab
- En 2014, la version EE (Enterprise Edition) devient payante même si son code source est toujours public
- Il est devenu un outil permettant d'améliorer le workflow des équipes et d'optimiser le cycle de vie DevOps

Présentation 2/3

- Gitlab est disponible en 2 versions
 - **CE : Community Edition** – gratuite – license MIT
 - **EE : Enterprise Edition** – payante - license propriétaire avec différents niveaux de souscription
- L'édition Enterprise est construite au-dessus de l'édition Community
- On peut télécharger librement l'une ou l'autre de ces versions
 - Cependant, il est nécessaire de payer pour accéder à certaines fonctionnalités sur EE
- L'utilisation de l'édition Enterprise sans souscription revient à utiliser la Community
 - On pourra débloquer les fonctionnalités supplémentaires via le paiement d'un abonnement

Présentation 3/3

- Application Web développé en langage Ruby par GitLab Inc.
 - <http://www.gitlab.com/gitlab-org/gitlab>
 - Dernière version: 16.10 (21 mars 2024)
- L'Enterprise Edition propose des fonctionnalités supplémentaires
- Voici les fonctionnalités de l'open core :
 - gestion du cycle de vie de projets git
 - gestion des participants aux projets et leurs droits (rôles, groupes, etc.).
 - Suivi des Issues pour lister les bugs
 - Suivi du temps
 - gérer la communication entre ces participants (intégration de Mattermost – Slack Like)
 - Création d'un Wiki pour la documentation
 - proposer des Merge Requests pour fusionner les branches
 - Intégration continue et Livraison continue (CI / CD)

Les installations

- En mode SAAS, il convient de créer un compte sur <https://gitlab.com/>
 - Au besoin, vous pourrez payer selon les fonctionnalités souhaités
- En mode On-Premise (sur site), plusieurs possibilités sont disponibles
 - Installation d'un package d'installation selon la distribution linux
 - Installation sur un cluster Kubernetes
 - Installation sur une infrastructure de cloud (AWS, GoogleCloudPlateform , Azure)
- Selon la méthode souhaitée, la documentation liste les opération à effectuer
- <https://about.gitlab.com/install/ce-or-ee/>

Inscription sur Gitlab

- Crédation d'un compte possible depuis le site
 - https://gitlab.com/users/sign_up
- Après validation, connexion possible
 - https://gitlab.com/users/sign_in



GitLab.com

Nom d'utilisateur ou adresse de courriel principale

Mot de passe

 [Mot de passe oublié ?](#)

Se souvenir de moi

Page d'accueil

The screenshot shows the GitLab homepage with a dark theme. The top navigation bar includes a logo, a search bar, and links for 'Your work' and 'Projects'. The main sidebar on the left is titled 'Your work' and lists various options: Projects (selected), Groups, Issues, Merge requests, To-Do List, Milestones, Snippets, Activity, Workspaces, Environments, and Help. The main content area is titled 'Projects' and shows a list of 5 projects under 'Yours'. The projects listed are 'SPRING_Book_API' and 'NODE_JS_Book_API', both owned by 'Boris Sauvage'. A red box highlights the 'New project' button in the top right corner, and another red box highlights the project list. Red arrows point from the text annotations to these specific areas.

Création d'un nouveau projet

Explore projects New project

La liste des projets

Navigation principale

Project Name	Owner	Last updated
SPRING_Book_API	Boris Sauvage	1 month ago
NODE_JS_Book_API	Boris Sauvage	1 month ago

Détail d'un projet Gitlab 1/4

The screenshot shows a GitLab project page for 'SPRING_Book_API'. On the left is a sidebar with various project management and configuration options. The main area displays the project name 'SPRING_Book_API' in a large header, with a red arrow pointing to it. Below the header is a commit history showing a single update to '.gitlab-ci.yml'. A red box highlights the 'Notifications', 'Star', and 'Fork' counts in the top right. To the right of the commit history is a table of files with their last commits and updates. A large orange arrow points from the text 'Visibilité du projet' to the top right corner of the page.

Visibilité du projet

Name	Last commit	Last update
.mvn/wrapper	Initialized from 'Spring' project template	4 months ago
src	Initialized from 'Spring' project template	4 months ago
.gitignore	Initialized from 'Spring' project template	4 months ago
.gitlab-ci.yml	Update .gitlab-ci.yml file	1 month ago
CONTRIBUTING.md	Initialized from 'Spring' project template	4 months ago
Dockerfile	Initialized from 'Spring' project template	4 months ago
LICENSE	Initialized from 'Spring' project template	4 months ago
README.md	Initialized from 'Spring' project template	4 months ago
mvnw	Initialized from 'Spring' project template	4 months ago

Project information

- 5 Commits
- 1 Branch
- 0 Tags
- 2.4 MiB Project Storage
- 1 Environment

README
MIT License
CONTRIBUTING
CI/CD configuration
+ Add CHANGELOG
+ Add Kubernetes cluster
+ Add Wiki
+ Configure Integrations

Created on

Détail d'un projet Gitlab 2/4

The screenshot shows the GitLab interface for the project 'SPRING_Book_API'. A red arrow points to the 'master' dropdown menu in the top navigation bar.

Branche en cours de consultation

Name	Last commit	Last update
.mvn/wrapper	Initialized from 'Spring' project template	4 months ago
src	Initialized from 'Spring' project template	4 months ago
.gitignore	Initialized from 'Spring' project template	4 months ago
.gitlab-ci.yml	Update .gitlab-ci.yml file	1 month ago
CONTRIBUTING.md	Initialized from 'Spring' project template	4 months ago
Dockerfile	Initialized from 'Spring' project template	4 months ago
LICENSE	Initialized from 'Spring' project template	4 months ago
README.md	Initialized from 'Spring' project template	4 months ago
mvnw	Initialized from 'Spring' project template	4 months ago

Project information

- 5 Commits
- 1 Branch
- 0 Tags
- 2.4 MiB Project Storage
- 1 Environment

Actions

- README
- MIT License
- CONTRIBUTING
- CI/CD configuration
- Add CHANGELOG
- Add Kubernetes cluster
- Add Wiki
- Configure Integrations

Created on

Détail d'un projet Gitlab 3/4

The screenshot shows the GitLab interface for the project 'SPRING_Book_API'. The top navigation bar includes a search bar, a pinned button, and a 'Code' dropdown. The main header displays the project name 'SPRING_Book_API' and the title 'Dernier commit'. A red arrow points to the first commit entry in the list below. The commit details show 'Update .gitlab-ci.yml file' by 'Boris Sauvage' from '1 month ago'. The commit hash is 'c70a543a'. To the right, there's a 'Project information' sidebar with metrics like 5 commits, 1 branch, 0 tags, and 2.4 MiB of storage.

Name	Last commit	Last update
.mvn/wrapper	Initialized from 'Spring' project template	4 months ago
src	Initialized from 'Spring' project template	4 months ago
.gitignore	Initialized from 'Spring' project template	4 months ago
.gitlab-ci.yml	Update .gitlab-ci.yml file	1 month ago
CONTRIBUTING.md	Initialized from 'Spring' project template	4 months ago
Dockerfile	Initialized from 'Spring' project template	4 months ago
LICENSE	Initialized from 'Spring' project template	4 months ago
README.md	Initialized from 'Spring' project template	4 months ago
mvnw	Initialized from 'Spring' project template	4 months ago

Détail d'un projet Gitlab 4/4

The screenshot shows a GitLab project page for 'SPRING_Book_API'. The top navigation bar includes a search bar, pinned issues, and a 'Code' button highlighted with a red arrow. The main content area displays a commit history and a detailed file list. A large red arrow points to the file list table.

Cloner le projet

Name	Last commit	Last update
.mvn/wrapper	Initialized from 'Spring' project template	4 months ago
src	Initialized from 'Spring' project template	4 months ago
.gitignore	Initialized from 'Spring' project template	4 months ago
.gitlab-ci.yml	Update .gitlab-ci.yml file	1 month ago
CONTRIBUTING.md	Initialized from 'Spring' project template	4 months ago

Project information

- 5 Commits
- 1 Branch
- 0 Tags
- 2.4 MiB Project Storage
- 1 Environment

Files

- README
- MIT License
- CONTRIBUTING
- CI/CD configuration
- Add CHANGELOG

La liste des fichiers sur la branche sélectionnée

Les outils du projet

The screenshot shows a sidebar menu titled "Project" with a list of tools. Each tool is associated with a red horizontal bar of varying lengths. To the right of each bar is a description in orange text.

- S SPRING_Book_API Le code source, les commits, les branches, ...
- Pinned
- Issues Board des développements en cours, le backlog, ...
- Merge requests Gestion des fusions de branches
- Manage
- Plan Milestones, wiki
- Code
- Build Outil d'intégration et de déploiement continu
- Secure
- Deploy
- Operate
- Monitor
- Analyze
- Settings Paramétrage des branches, droits, ...

Graph des commits

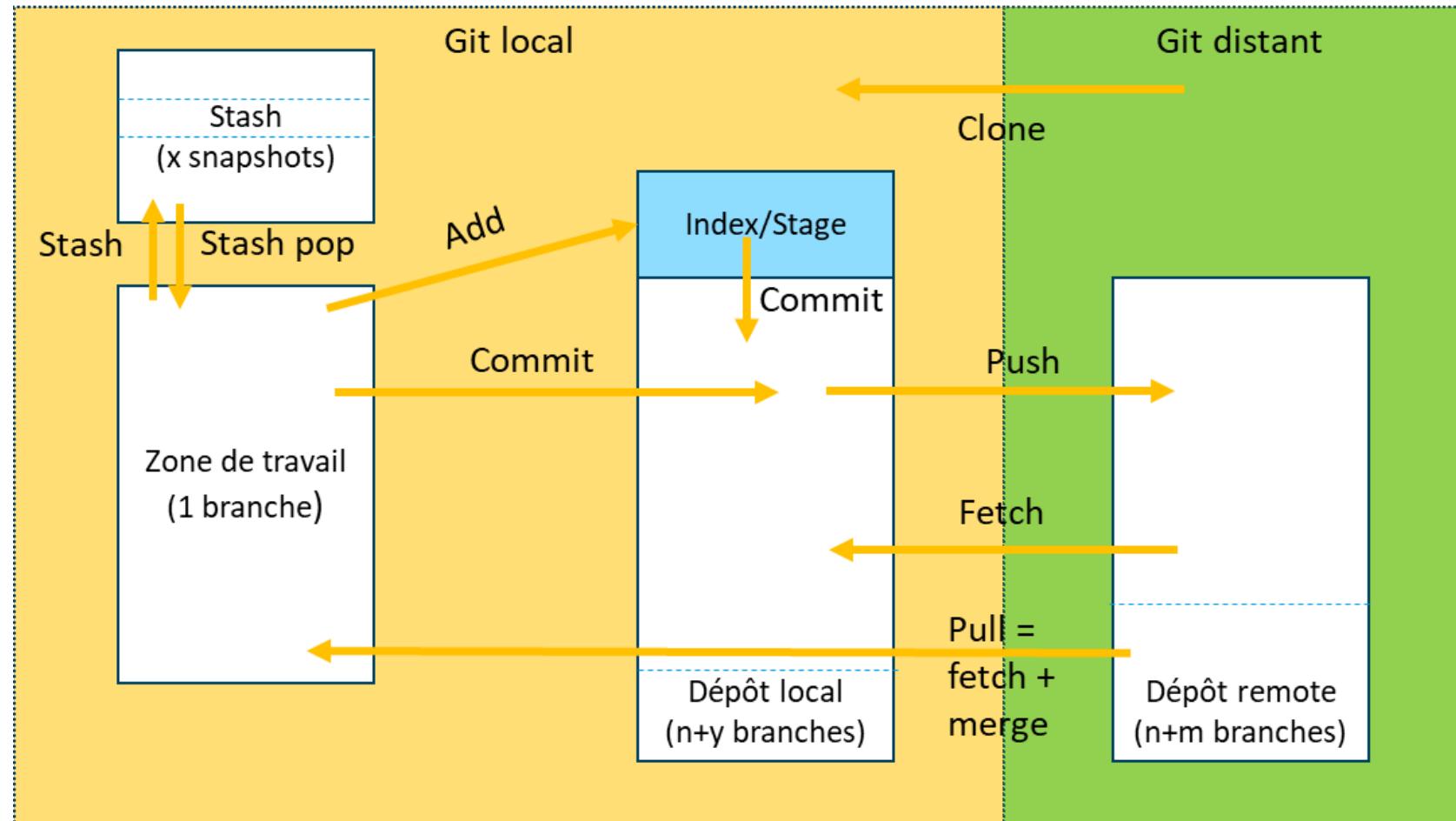
The screenshot shows the GitLab interface for a project named "SPRING_Book_API". The left sidebar has a "Repository graph" section with a red box around the "Compare revisions" button, and a red arrow points from this button to the main commit graph area. The main area displays a timeline of commits on the "master" branch. The commits are listed from top to bottom:

- Update .gitlab-ci.yml file
- Ajout d'un job sonar (fournit par SonarCloud)
- Ajout d'un job sonar
- Update pom.xml (sonarqube)
- Initialized from 'Spring' project template

The commits are dated from January 22, 2019, down to November 2, 2018.

Outil de comparaison des versions

Résumé



Solutions similaires

- Travis CI : <https://travis-ci.com/>
- CircleCI : <https://circleci.com/>
- Bamboo : <https://www.atlassian.com/software/bamboo>
- Azure DevOps : <https://azure.microsoft.com/services/devops/>
- GitHub Actions : <https://github.com/features/actions>
- TeamCity : <https://www.jetbrains.com/teamcity/>
- Bitbucket Pipelines : <https://bitbucket.org/product/features/pipelines>

Développement

GITLAB & GIT

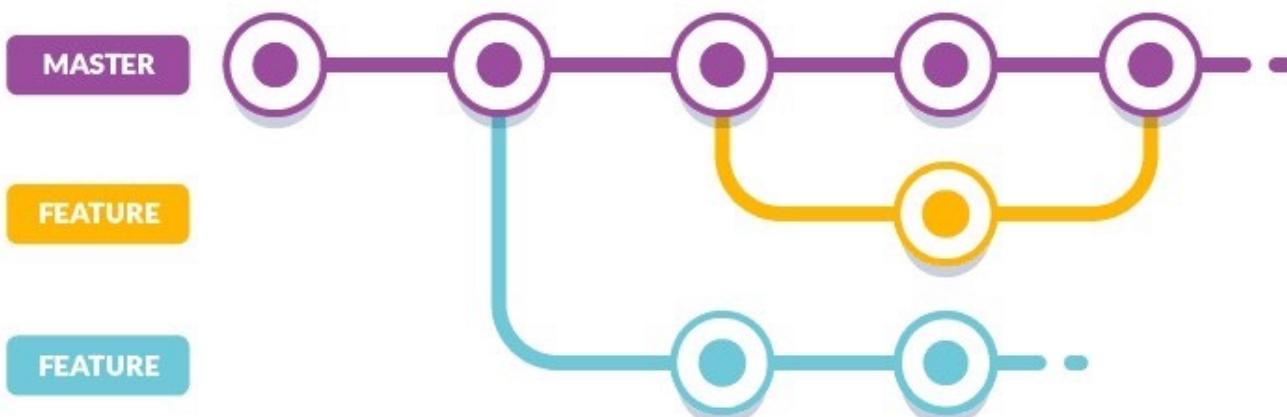
Les workflows

- Un dépôt Git peut vite devenir compliqué à maintenir sans règles imposées
- **Solution : Suivre un workflow de développement**
 - Pour unifier les pratiques au sein d'une même équipe
 - Pour simplifier la gestion du dépôt
 - Pour connaître en temps réel l'état de son dépôt (les fonctionnalités en cours de développement, les branches pouvant être supprimées, ...)
- **Exemples de workflow de développement**
 - **Feature Branch Workflow** : la simplicité, penser petit et synchronisation régulière
 - **GitFlow** : une solution **robuste** s'adaptant à tous les contextes
 - **Fork & Merge** : chacun son dépôt distant, utilisé pour l'**open-source**

Feature Branch Workflow 1/2

- Workflow idéal lorsqu'il n'y a pas besoin de gérer des releases ou des versions
 - Fonctionne surtout en équipe réduite
- **Concept**
 - Une seule branche principale : main
 - Chaque **fonctionnalité** fait l'objet d'une branche **feature** tirée de la main
- **Principes**
 - Tout ce qui est sur main est **déployable**
 - Chaque branche doit avoir un nom significatif
 - Commit et push réguliers
 - **Merge request (Pull request pour Github)** à la fin d'une feature
 - Déployer directement après **merge**

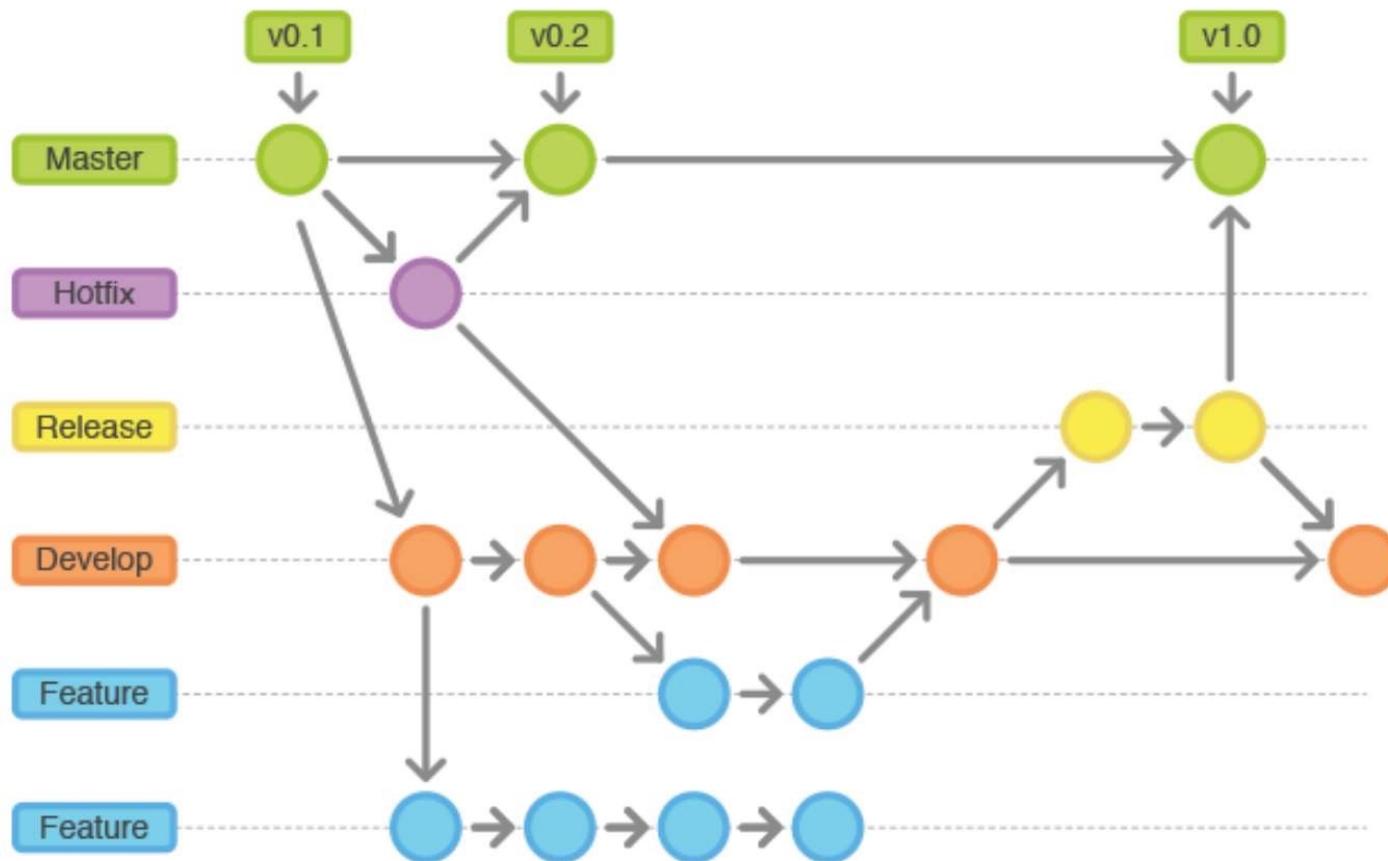
Feature Branch Workflow 2/2



GitFlow 1/2

- Pour les contextes d'équipes grandissantes, de méthode agile et de livraison continue
- **Concept**
 - Une branche pour la production : **main**
 - Une branche pour le développement : **develop**
 - Des branches pour la préparation des mises en production et la gestion des versions : **release**
 - Des branches pour le développement des fonctionnalités : **feature**
 - Des branches pour les corrections en production : **hotfix**
- **Principes**
- Tout ce qui est sur **main** est **déployable**
- Qualification continue sur la **develop** ou les **features**
- Qualification totale/recette sur les branches de **release**

GitFlow 2/2



Fork & Merge 1/2

- Idéal pour l'open source afin que n'importe qui puisse contribuer à un projet sans le corrompre
 - Plus complexe à mettre en place et à maintenir
- **Concept**
 - Aucun travail sur le dépôt principal
 - Fork du dépôt distant (git clone vers un autre dépôt distant)
 - Travail sur le fork qui peut régulièrement récupérer les modifications du dépôt principal
 - **Merge/Pull request** du fork à la fin avec discussions et merge par les maintainers du dépôt principal
- **Principe**
 - Tout ce qui est sur le dépôt principal est déployable et revu de tous

Fork & Merge 2/2



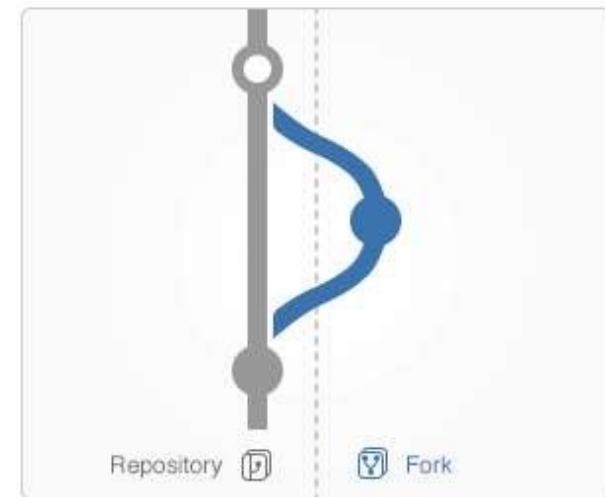
Fork

Develop features on a branch and create a pull request to get changes reviewed.



Discuss

Discuss and approve code changes related to the pull request.



Merge

Merge the branch with the click of a button.

Projets Gitlab

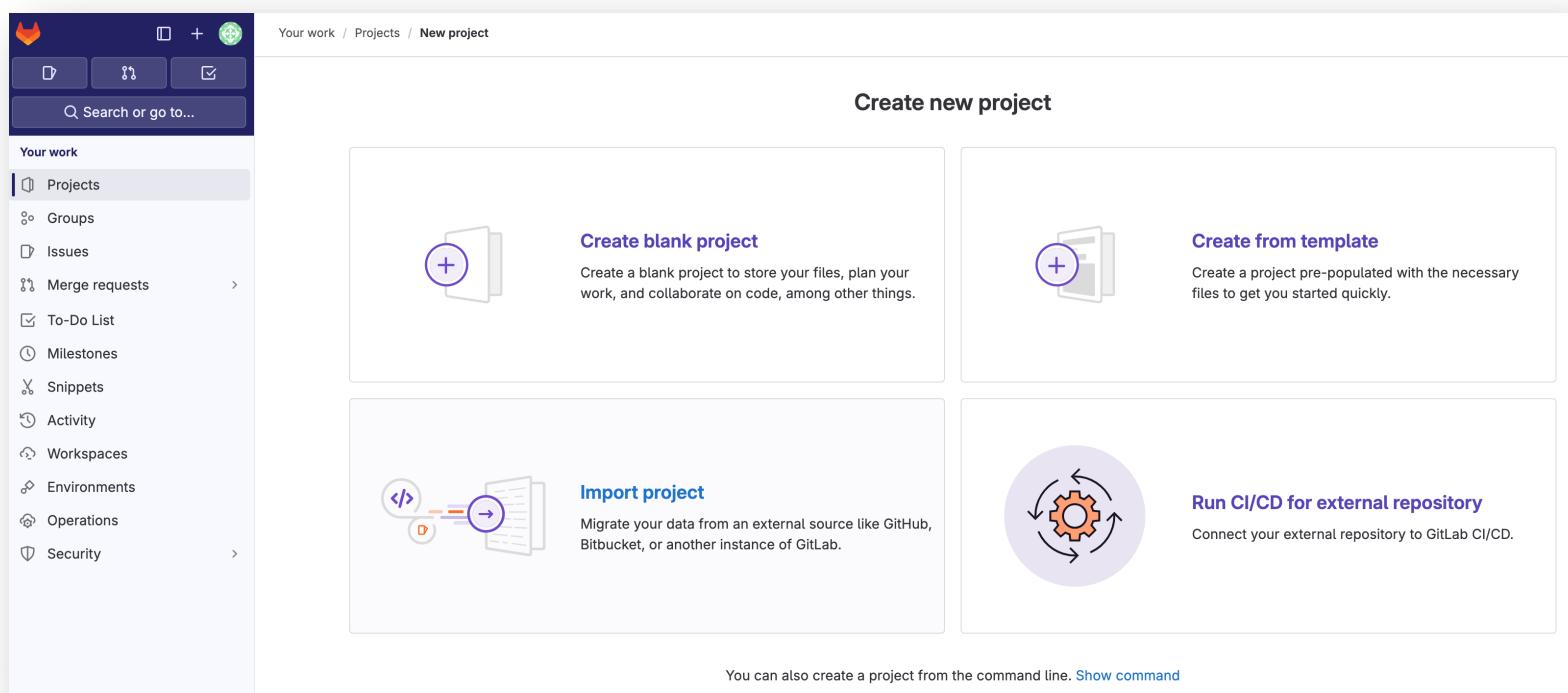
- Crédation de projets de développement versionnés
 - <https://gitlab.com/projects/new>

The screenshot shows the GitLab interface with the following details:

- Header:** Your work / Projects
- Search Bar:** Search or go to...
- Sidebar (Your work):** Projects (selected), Groups, Issues, Merge requests, To-Do List, Milestones, Snippets, Activity, Workspaces, Environments, Operations, Security.
- Top Right:** Explore projects, New project, Filter by name, Language, Last created.
- Section Headers:** Yours (3), Starred (0), Pending deletion.
- Buttons:** All, Personal.
- Project Listings:**
 - Boris Sauvage / SPRING_Book_API (Owner) - Updated 1 day ago
 - Boris Sauvage / NODE_JS_Book_API (Owner) - Updated 1 day ago

Nouveau Projet Gitlab 1/2

- Crédation de projets de développement versionnés
 - <https://gitlab.com/projects/new>



Nouveau Projet Gitlab 2/2

 **Create blank project**

Create a blank project to store your files, plan your work, and collaborate on code, among other things.

Project name
ExempleProjet

Must start with a lowercase or uppercase letter, digit, emoji, or underscore. Can also contain dots, pluses, dashes, or spaces.

Project URL / **Project slug**
https://gitlab.com/sauvageb/ / exempleprojet

Project deployment target (optional)
Select the deployment target ▾

Visibility Level ⓘ
 Private
Project access must be granted explicitly to each user. If this project is part of a group, access is granted to members of the group.
 Public
The project can be accessed without any authentication.

Project Configuration

Initialize repository with a README
Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.

Enable Static Application Security Testing (SAST)
Analyze your source code for known security vulnerabilities. [Learn more](#).

Create project **Cancel**

Premier commit

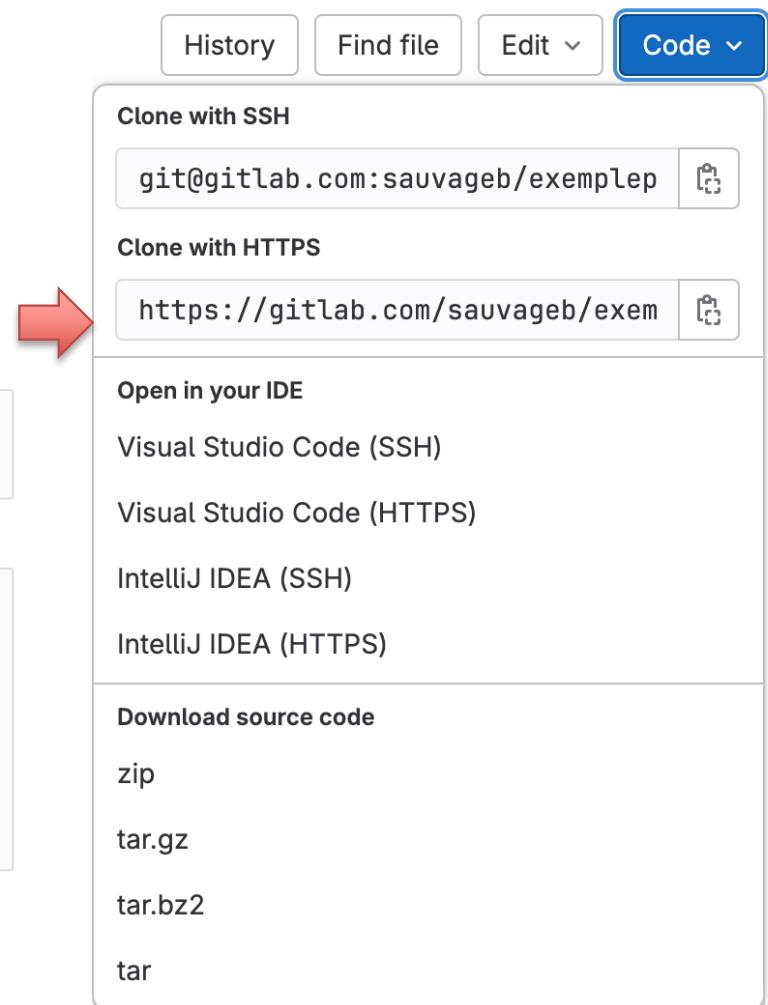
- Cloner votre dépôt distant depuis l'URL dans :
 - Clone with HTTPS

Git global setup

```
git config --global user.name "Boris Sauvage"  
git config --global user.email "sauvageboris.pro@gmail.com"
```

Create a new repository

```
git clone https://gitlab.com/sauvageb/exemple.git  
cd exemple  
git switch --create main  
touch README.md  
git add README.md  
git commit -m "add README"  
git push --set-upstream origin main
```



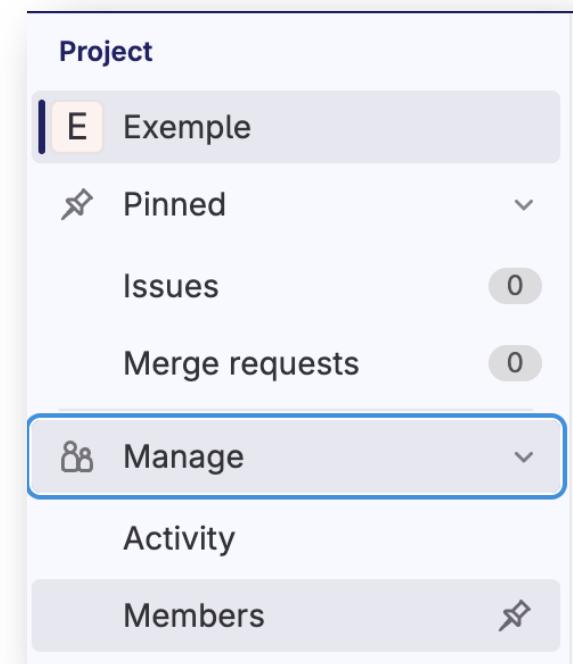
Développement

LA GESTION DU DÉPÔT AVEC GITLAB



Les collaborateurs 1/4

- Dans le cadre d'un projet, on distingue plusieurs acteurs avec leurs droits
- GitLab permet de gérer cet aspect en permettant :
 - D'ajouter des acteurs à un projet
 - D'affecter à chacun de ces acteurs des droits via un rôle
- Pour éditer la liste des membres d'un projet, il convient d'aller dans la section **Manage > Members**



Les collaborateurs 2/4

- Pour ajouter un utilisateur au projet, on dispose d'un formulaire à remplir via le bouton **Invite members**
 - On y indique l'utilisateur, son rôle, ainsi qu'éventuellement une date d'expiration de son accès
 - On peut aussi importer les membres d'un projet depuis un autre projet avec le bouton **Import from a project**

Invite members

You're inviting members to the **Exemple** project.

Username, name or email address

Select members or type email addresses

Select a role

Guest

[Read more](#) about role permissions

Access expiration date (optional)

 YYYY-MM-DD

[Cancel](#) **Invite**



Les collaborateurs 3/4

- Les rôles Gitlab définissent les droits qu'un utilisateur aura
- On distingue quatre rôles qui sont logiques dans leurs attributions :
 - **Guest** qui peut consulter et laisser des commentaires
 - **Reporter** qui peut gérer les issues et créer des **merge request**
 - **Developer** qui peut créer des branches, faire des push sur les branches non protégées, des merge requests et des tags
 - **Maintainer** qui peut faire des push sur les branches protégées, modifier les paramètres du projet, modifier les variables CI/CD
 - **Owner (propriétaire)** qui a tous les droits sur le projet
- La documentation liste les droits de chaque rôle :
 - <https://docs.gitlab.com/ee/user/permissions.html>

Les collaborateurs 4/4

- Sur la page des membres apparaissent également les membres du projet
 - Il est possible de modifier les droits d'un utilisateur ou sa date d'expiration d'accès
 - Il est également possible de supprimer un utilisateur du projet

Project members

You can invite a new member to **Exemple** or invite another group.

Members 2

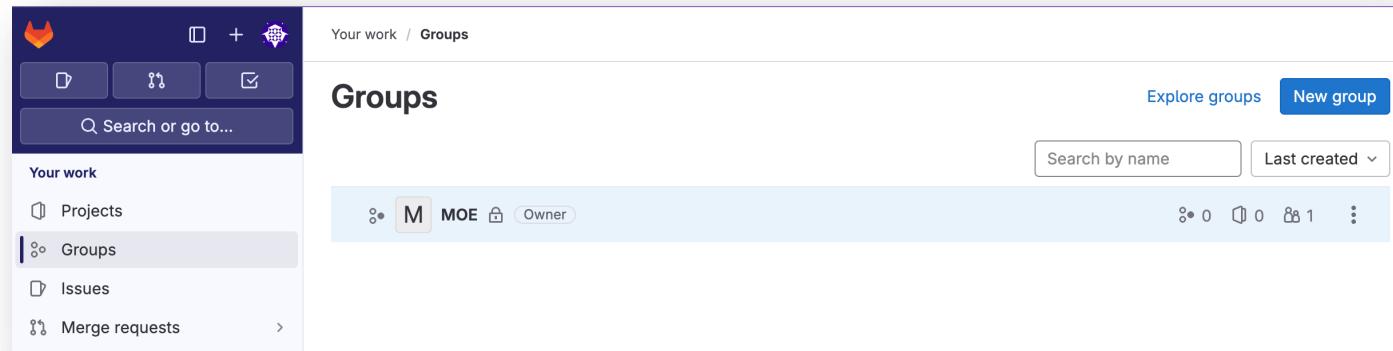
Filter members Account ▾ Account ▾

Account	Source	Max role	Expiration	Activity
 Boris Sauvage <small>(It's you)</small> @sauvageb	Direct member by Boris Sauvage	Owner	Expiration date <input type="button" value="▼"/>	User created: Oct 20, 2020 Access granted: Mar 07, 2024 Last activity: Mar 10, 2024
 JOHN DOE @john.doe	Direct member by Boris Sauvage	Guest ▾	Expiration date <input type="button" value="▼"/>	User created: Feb 19, 2020 Access granted: Mar 10, 2024 Last activity: Feb 19, 2020

- On dispose d'option de tri et de recherche dans le cas d'une liste importante de membres

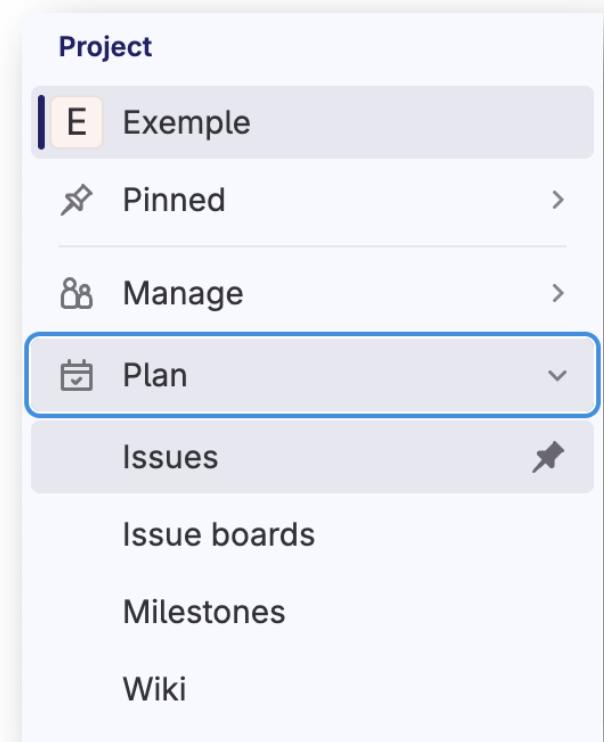
Les groupes

- Les groupes permettent de regrouper des projets ensemble
 - Gitlab offre la possibilité de créer également des sous-groupes
- L'utilisation des groupes facilite la gestion des utilisateurs
 - Un projet va hériter des membres du groupe qui le contient
- On retrouve la gestion des groupes dans le menu **Groups > Your Groups** en haut de l'écran
 - On n'y retrouvera la possibilité de créer un nouveau groupe et d'ajouter des membres ou encore de créer des sous-groupes



Les issues 1/7

- Le système d'issues de GitLab est très riche en fonctionnalités
 - Il permet de suivre les anomalies et les améliorations au sein d'un projet
- Chaque **issue** se voit attribuer un numéro unique au sein du projet
- À noter que, dans les projets **public** et **internal**, tous les utilisateurs peuvent créer des **issues**
 - Dans les projets privés, il convient d'être membre du projet en mode **Guest** pour créer une **issue**
- Dans un projet, on retrouve la gestion des **issues** dans la section **Issues** du projet



Les issues 2/7

- Un issue est constitué
 - D'un titre
 - D'un type
 - D'une description (texte riche)
 - On peut également indiquer
 - À qui est assignée cette **issue**
 - Un jalon (**milestone**)
 - Des étiquettes (**labels**)
 - Une date voulue de résolution (**due date**)

New Issue

Title (required)

Type [?](#)

Issue

Description

Preview | *B* *I* *₹* | *≡* *</>* *♂* *♀* *☰* *☷* *☷* *☷* | *田* *📎* *☒*

Write a description or drag your files here...

Switch to rich text editing

Add **description templates** to help your contributors to communicate effectively!

This issue is confidential and should only be visible to team members with at least Reporter access.

Assignee

Unassigned [Assign to me](#)

Milestone

Select milestone

Labels

Select label

[Create Issue](#) [Cancel](#)

Les issues 3/7

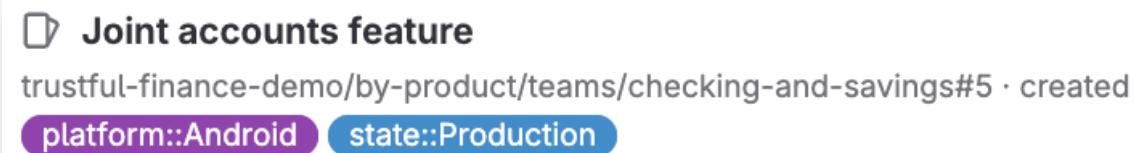
- Une fois l'**issue** créée, elle apparait dans la liste **Issues**

The screenshot shows a software interface for managing issues. At the top, there are buttons for 'Open' (1), 'Closed' (0), and 'All' (1). On the right, there are buttons for 'Bulk edit', 'New issue', and a three-dot menu. Below this is a search bar with a clock icon and the placeholder 'Search or filter results...', and a 'Created date' dropdown. The main list area contains one item:

⌚ Ajout d'une case à cocher "Payée" sur le formulaire d'une facture
#1 · created just now by Boris Sauvage · Mar 11, 2024

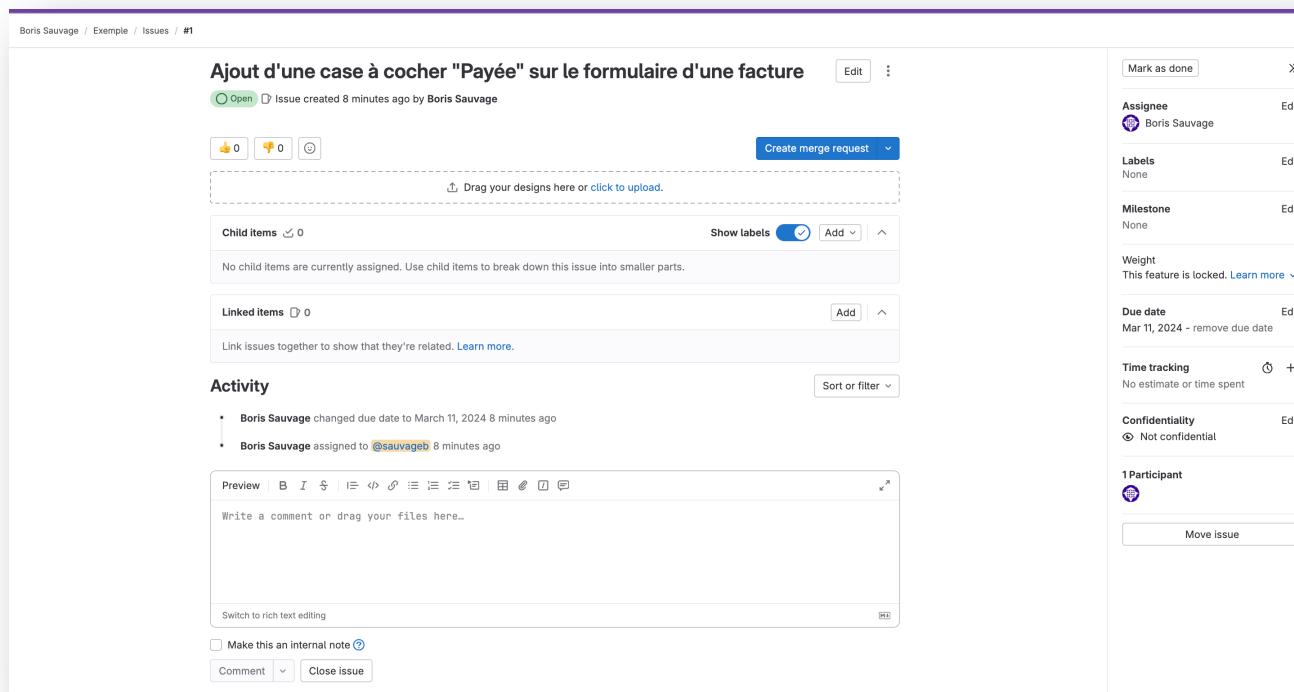
On the right side of the list item are icons for a globe, a document, and a number '0'.

- Apparaissent dans cette liste les informations de base d'une **issue** avec ses étiquettes
 - Les versions payantes permettent d'utiliser les **scoped labels** définis avec les caractères :: sous la forme **clé::valeur**



Les issues 4/7

- Un clic sur une **issue** permet d'accéder à sa fiche avec un panneau extensible à droite de l'écran



Les issues 5/7

- La fiche de l'**issue** permet d'effectuer plusieurs actions, par exemple :
 - Ajouter des commentaires
 - Modifier les étiquettes
 - Modifier les membres en charge de cette **issue (Assignees)**
 - Fermer l'**issue**
- À noter que le tableau de bord **Issues > Board** permet de gérer ses **issues** sous forme de carte à déplacer
 - Le déplacement modifiera le statut de Les sous et ses étiquettes
- Il est possible d'offrir la possibilité de créer une **issue automatiquement** suite à l'envoi d'un mail
 - Activer et paramétrier Service Desk pour cela

Les issues 6/7

- Les messages de commit peuvent avoir un impact automatique sur le statut de certaines **issues**
 - On pourra par exemple fermer automatiquement une issue grâce à un commit
 - On pourra aussi lier un commit à une **issue** particulière
 - Des liens hypertextes seront alors présent au sein de GitLab pour accéder à l'issue depuis le commit, et inversement
 - On désignera une **issue** sous la forme de #3 pour l'issue numéro 3
- Pour fermer automatiquement une **issue**, on peut utiliser les expressions suivantes au sein des messages de commit

Available keywords:

- Close, Closes, Closed, Closing, close, closes, closed, closing
- Fix, Fixes, Fixed, Fixing, fix, fixes, fixed, fixing
- Resolve, Resolves, Resolved, Resolving, resolve, resolves, resolved, resolving
- Implement, Implements, Implemented, Implementing, implement, implements, implemented, implementing

Les issues 7/7

- Exemple d'utilisation au sein d'un message de commit :

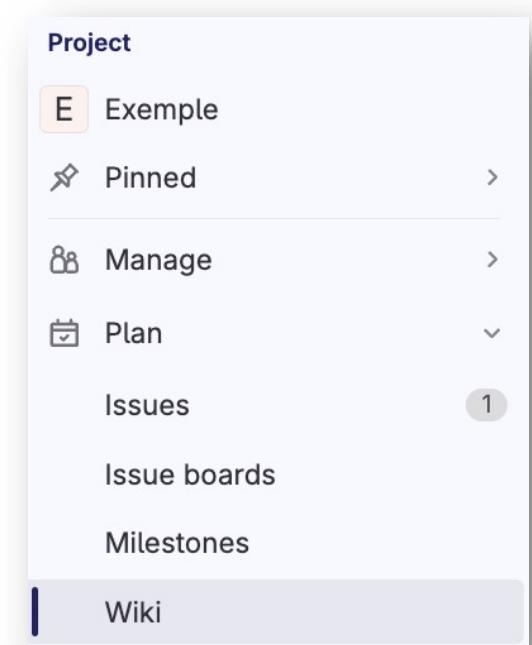
Message du commit

Fix #2 and Closes #1
Related to #3

- Les **issues 1 et 2** seront closes avec dans chacune d'elle un lien vers le commit associé
- Une note vers le **commit** sera ajouté dans l'**issue 3**

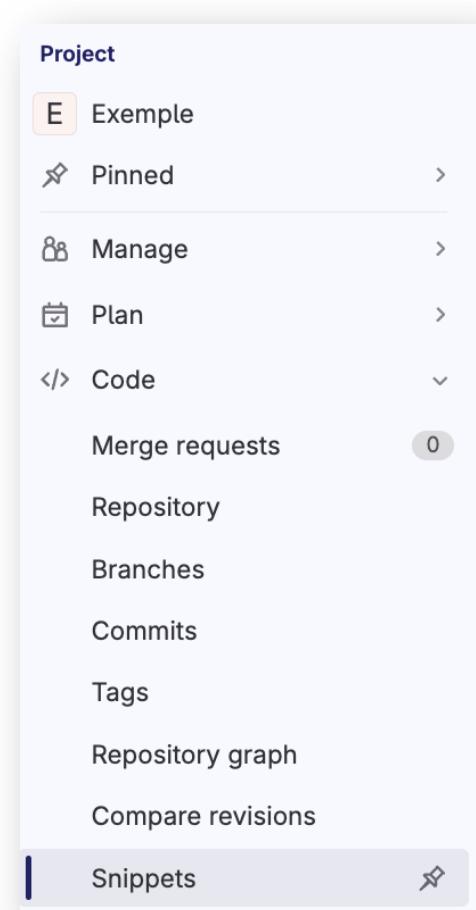
Le wiki

- Le wiki Gitlab permet d'élaborer de la documentation pour un projet
- On écrit des pages au format MarkDown et on peut ajouter des liens vers d'autres pages
 - La création d'une page wiki nécessite le rôle **Developer**
- L'édition d'une page wiki est riche en possibilité
 - La documentation liste toutes les possibilités de typographie et de mise en forme via le système MarkDown de Gitlab
 - <https://docs.gitlab.com/ee/user/markdown>
- On notera une partie spécifique aux références Gitlab qui permet de faire des liens vers des objets ou entité Gitlab (merge request, projet, utilisateur, snippet, issue, ...)
 - <https://docs.gitlab.com/ee/user/markdown#gitlab-specific-references>



Les snippets 1/3

- Le système de fragments (**snippets**) permet de partager des morceaux de code ou du texte (par exemple des commandes) avec d'autres utilisateurs
- Dans un projet, on retrouve la gestion des **snippets** dans la section **Snippets** du projet
- Un **Snippet** comprend :
 - Un titre
 - Une description en format *Markdown*
 - Plusieurs fichiers contenant du code ou du texte
 - Un accès (**private** – membres du projet, **internal** ou **public** – tout le monde)



Les snippets 2/3

- Exemple de création d'un Snippet

Les snippets 3/3

- Une fois créé, le **snippet** apparaît dans la liste des **snippets** du projet

The screenshot shows a snippet list with the following details:

- All 1 Private 1 Internal 0 Public 0
- New snippet button
- Snippet entry: Fragment #5 (\$3686346 · created 1 minute ago by Boris Sauvage)
- Statistics: 1 like, 0 comments, updated 1 minute ago

- On peut également accéder à la fiche d'un Snippet et le commenter

The screenshot shows a snippet detail page for "Fragment #5".

Header information:

- Authored just now by Boris Sauvage
- Edited just now
- Clone dropdown

Code editor:

```
example.js 110 B
1 const fruits = ['pomme', 'orange', 'banane', 'fraise'];
2
3 fruits.forEach(fruit => {
4   console.log(fruit);
5});
```

Interaction buttons:

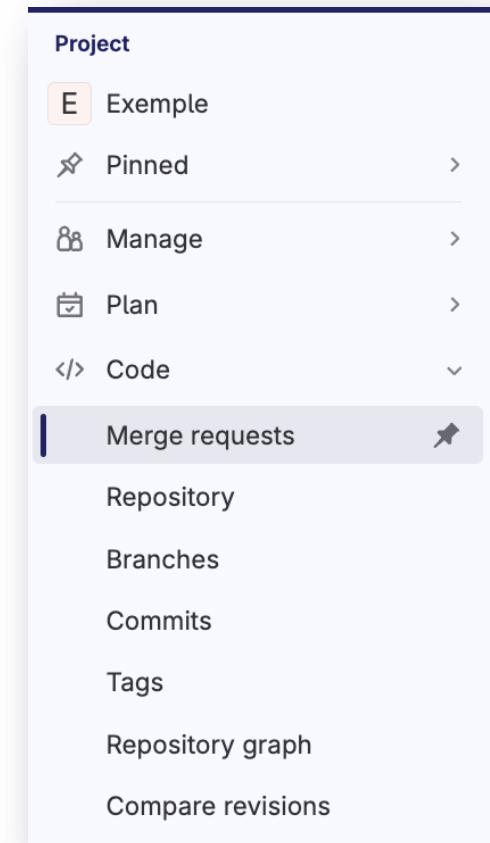
- Like 0
- Dislike 0
- Comment 0

Preview toolbar:

Write a comment or drag your files here...

Les merge requests 1/4

- Une **merge request (pull request dans Github)** permet de proposer des changements apportés dans une branche depuis notre branche
- Cette fonctionnalité permet de voir les changements apportés, mais aussi le statut du pipeline
 - Les collaborateurs du projet peuvent commenter ce qui est apporté avant que la **merge request** soit validée
- On créer une demande de fusion en allant dans le menu **Merge Request**, puis en cliquant sur **New Merge Request**



Les merge requests 2/4

- Une **merge request** est constitué d'une branche source et d'une branche cible qu'il faut sélectionner à sa création
 - La branche cible peut être une branche d'un autre dépôt dans le cadre fork

New merge request

Source branch

sauvageb/exemple test_branche

 Update test.js
Boris Sauvage authored Mar 10, 2024

 8ed2f58d 

Target branch

sauvageb/exemple main

 Update .gitlab-ci.yml file (Test coverage badge)
Boris Sauvage authored Mar 10, 2024

 2ab13ea5 

[Compare branches and continue](#)

- On peut ensuite paramétriser la **merge request** en lui donnant un titre, une description et d'autre paramètres

Les merge requests 3/4

- Dans la fenêtre récapitulative de la merge de Request, on retrouve le résultat de l'exécution du pipeline
- Certains peuvent approuver les modifications
- Un bouton **Merge** est présent afin de valider la **merge request** si tous les prérequis sont validés

Superbe fonction

Open Boris Sauvage requested to merge [test_branch](#) into [main](#) just now

[Overview](#) 0 [Commits](#) 0 [Pipelines](#) 0 [Changes](#) 1

[0](#) [0](#)

Pipeline #1207884796 passed
Pipeline passed for 8ed2f58d on [test_branch](#) 3 minutes ago

[Approve](#) Approval is optional [?](#)

Ready to merge!

Delete source branch Squash commits [?](#) Edit commit message
1 commit and 1 merge commit will be added to main.

[Merge](#)

Activity

All activity [↑](#)

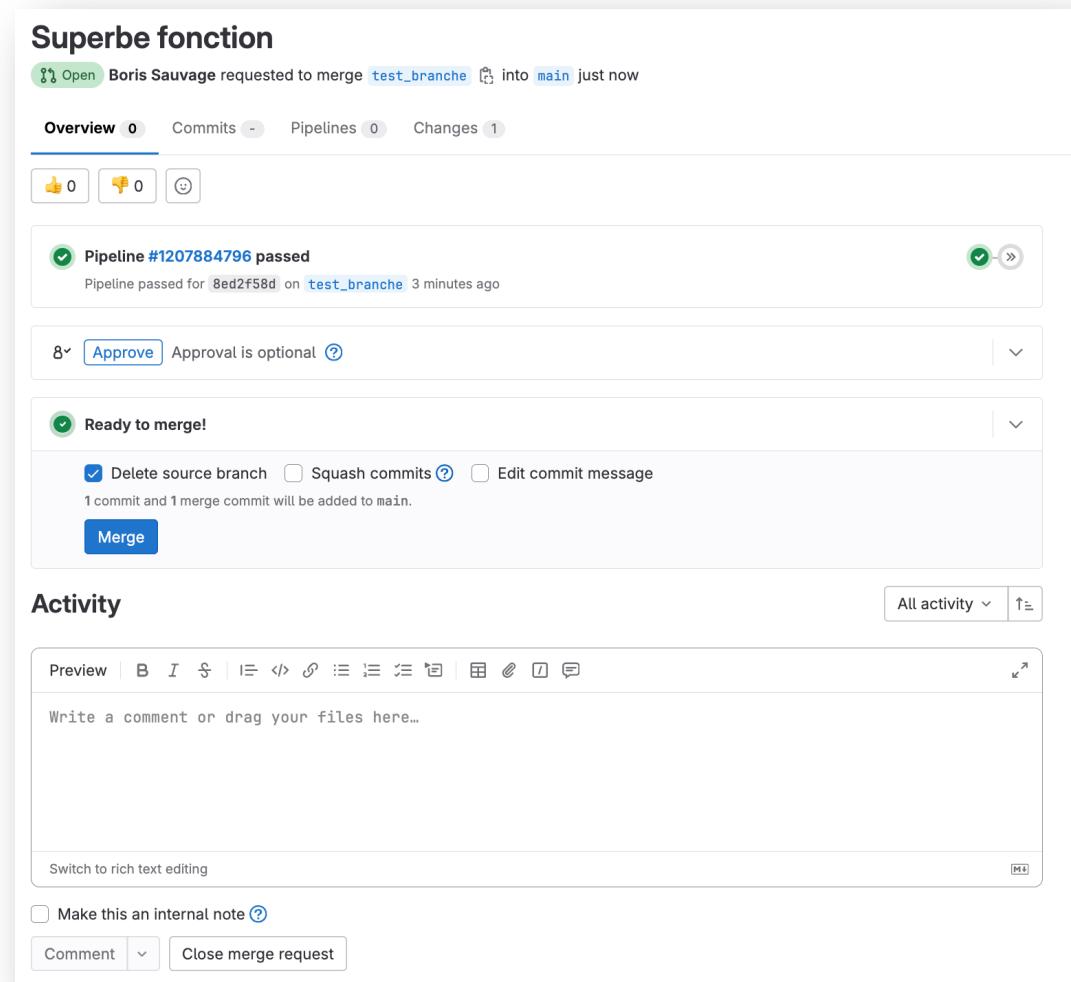
Preview | [B](#) [I](#) [S](#) | [IE](#) [C](#) [E](#) [R](#) [W](#) | [F](#) [O](#) [D](#) [M](#) [L](#) [C](#)

Write a comment or drag your files here...

Switch to rich text editing

Make this an internal note [?](#)

[Comment](#) [Close merge request](#)



Les merge requests 4/4

- Dans la partie **Settings > Repository**, section **Protected Branches**, on peut paramétrer les droits sur les branches
 - On peut partager une branche, ou faire en sorte qu'elle ne soit touché que par des **merge Request**

Protected branches

Keep stable branches secure and force developers to use merge requests. [What are protected branches?](#)

⚠️ Giving merge rights to a protected branch also gives elevated permissions for certain CI/CD features. [What are the security implications?](#)

Branch	Allowed to merge	Allowed to push and merge	Allowed to force push ?
main default	Maintainers ▼	Maintainers ▼	<input checked="" type="checkbox"/> X Unprotect

- Dans la partie **Settings > General**, section **Merge requests**, on a la possibilité de modifier certains paramètres liés aux **merge requests**
 - À partir des plans Premium de GitLab, on peut paramétrer des règles de validation (approvals)

- FIN -