# Git, GitLab & GitHub

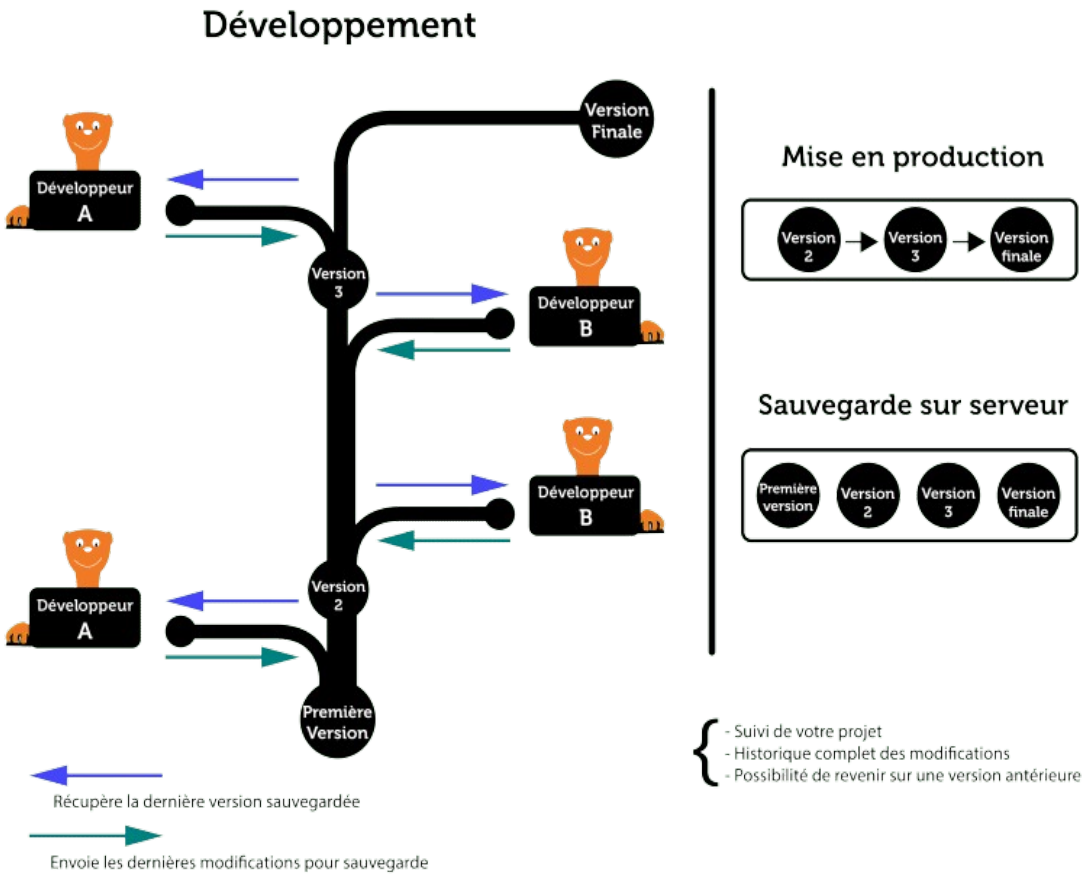
## git, c'est quoi?

Git est un outil qui permet de conserver l'historique complet des modifications effectuées sur votre projet, que l'on nomme version.

Plusieurs versions de votre projet sont sauvegardées sur un serveur. Il est possible à tout moment de revenir sur une version antérieure.

Plusieurs développeurs peuvent travailler en parallèle sur une ou plusieurs tâches/fonctionnalités. Chacun se base sur la dernière version sauvegardée. Leur travail est compartimenté et permet ainsi une plus grande flexibilité concernant les développements.

Lorsque toutes les modifications sont validées, il y a fusion des versions.



## 1. Présentation

Git est un **système de contrôle de version décentralisé**. C'est un logiciel **libre et gratuit**, créé en 2005 par Linus Torvalds, auteur du noyau Linux. Il s'agit du logiciel de gestion de versions le plus populaire dans le développement logiciel et web, qui est utilisé par des dizaines de millions de personnes, sur tous les environnements (Windows, Mac, Linux).

Il permet de **suivre les modifications apportées aux fichiers et de coordonner le travail entre plusieurs personnes**. Git est devenu l'un des outils les plus populaires parmi les développeurs en raison de sa flexibilité, de sa performance et de ses capacités de branchement et de fusion.

GitLab et GitHub sont tous deux basés sur le système de contrôle de version distribué Git. Elles offrent des **fonctionnalités supplémentaires** pour la **gestion de projets et la collaboration**. Ces outils sont essentiels pour les développeurs modernes, permettant une meilleure gestion du code source, de la collaboration en équipe et de l'intégration continue.
Les deux plateformes se **ressemblent beaucoup**. Elles fonctionnent toutes deux sur des serveurs Linux, sont dotées d'un gestionnaire de problèmes et proposent un large éventail d'intégrations et d'outils d'importation tiers. Elles disposent également toutes deux d'interfaces en ligne de commande (CLI) pour les développeurs avancés, et proposent également des interfaces web pour les nouveaux programmeurs. Toutefois les deux plateformes adoptent des approches de développement différentes (intégration continue/livraison continue (CI/CD) et DevOps ).
Environ 77 % des développeurs utilisent régulièrement GitHub, contre 40 % pour GitLab.

### Fonctionnalités principales de Git :

- **Contrôle de version distribué** : Chaque utilisateur possède une copie complète de l'historique du projet, ce qui permet de travailler hors ligne et de sauvegarder les versions du projet localement ;
- **Branchement et fusion** : Permet de créer des branches pour développer des fonctionnalités indépendamment, puis de les fusionner une fois terminées ;
- **Historique complet** : Enregistre chaque modification avec un identifiant unique, ce qui permet de revenir à des versions antérieures si nécessaire ;
- **Collaboration** : Facilite le travail d'équipe en permettant à plusieurs développeurs de travailler sur le même projet simultanément.

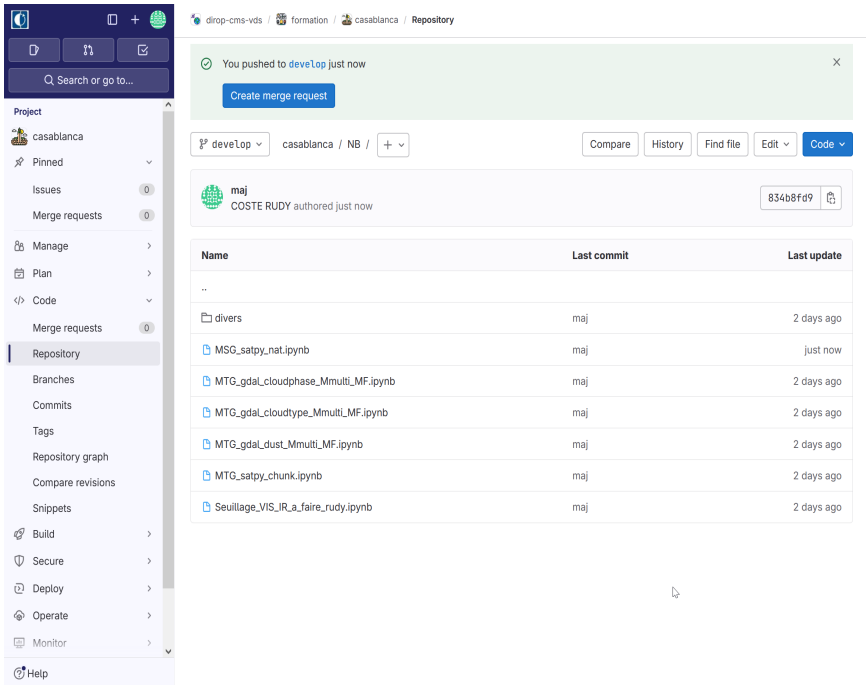
# 2. GitLab

## 2.1. Qu'est-ce que GitLab ?

GitLab est arrivé plus tard que GitHub (2011 vs 2008). C'est une plateforme de DevOps complète, fournissant des fonctionnalités pour le développement, la sécurité et l'exploitation. Lancé en 2011 par Dmitriy Zaporozhets et Valery Sizov, GitLab est conçu pour offrir une solution tout-en-un pour la gestion du cycle de vie des applications, de la planification au déploiement.
L'édition Community de GitLab reste gratuite et open source.

## 2.2. Fonctionnalités Principales

- **CI/CD** : Intégration et déploiement continus pour automatiser les tests et les déploiements ;
- **Gestion de Projet** : Tableau de bord Kanban, gestion des problèmes, jalons, et suivi du temps ;
- **Sécurité** : Audit de sécurité, tests de sécurité automatisés, et gestion des vulnérabilités ;
- **Collaboration** : Revue de code, discussions sur les merge requests, et gestion des utilisateurs ;
- **Auto DevOps** : Automatisation des pipelines CI/CD sans configuration préalable.




## 2.3. Avantages de GitLab

- **Self-hosted : Peut être hébergé sur vos propres serveurs pour un contrôle total ;**
- Open Source : La version communautaire est open source, permettant des personnalisations et des extensions ;

- **Complétude** : Offre une solution intégrée pour tous les aspects du développement logiciel ;

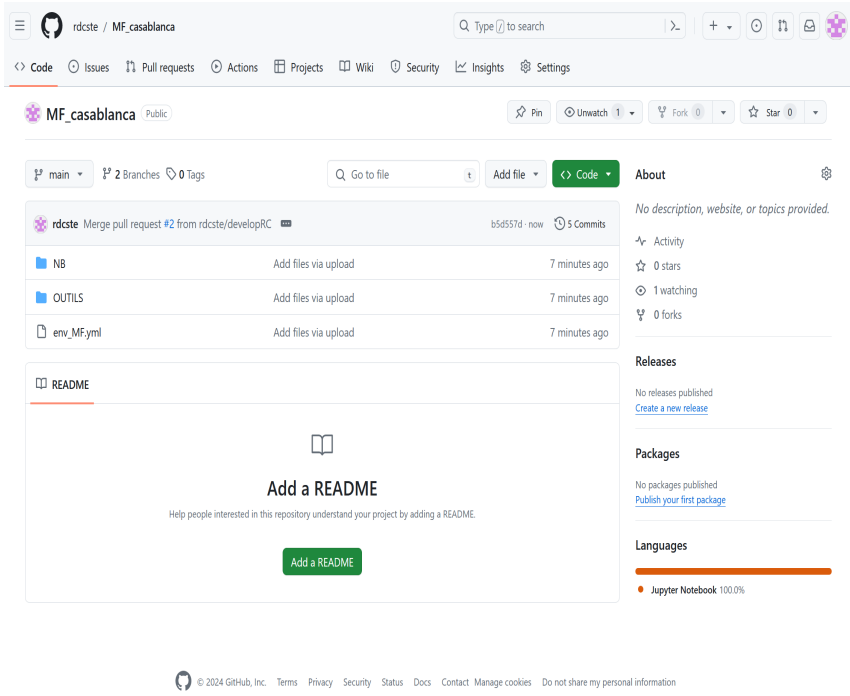
# 3. GitHub

## 3.1. Qu'est-ce que GitHub ?

GitHub est le plus ancien des services (2008) et a été acquis par Microsoft en 2018  Grâce à son avantage de pionnier, GitHub est devenu le point d'attache de nombreux dépôts de code libre. Il propose des référentiels gratuits limités et d'autres payants pour  les projets privés et commerciaux avec des fonctionnalités supplémentaires telles que des outils avancés de sécurité, une intégration CI/CD plus sophistiquée, et un support amélioré.

## 3.2. Fonctionnalités Principales

- **Repositories** : Hébergement de projets avec gestion des versions.
- **Pull Requests** : Facilite les revues de code et les discussions autour des modifications proposées ;
- **Actions** : CI/CD intégré pour automatiser les workflows.
- **GitHub Pages** : Hébergement gratuit de sites web statiques directement depuis un référentiel ;
- **Sécurité** : Analyses de sécurité, audits de dépendances, et alertes de vulnérabilités.




## 3.3. Avantages de GitHub

- **Communauté** : Une large communauté de développeurs et de projets open source ;
- **Intégrations** : Large écosystème d'intégrations avec des outils tiers.
- **Facilité d'utilisation** : Interface utilisateur intuitive et documentation exhaustive ;
- **Éducation** : Programmes pour les étudiants et les enseignants, ainsi que des ressources pédagogiques.

## 4. Commandes de Base

- **git clone <url-du-repo>** # Clone un référentiel existant. Copie tout le contenu du référentiel distant dans un nouveau répertoire local.

Pour aller plus loin

- git init # Initialise un nouveau référentiel Git.
- git checkout <nom-de-branche> # Bascule sur une branche existante
- git checkout -b <nom-de-branche> # Crée et bascule sur une nouvelle branche
- git branch #lister toutes les branches présentes dans le dépôt
- git config --global user.email "@mail" # configurer les préférences de l'utilisateur
- git config --global user.name "Prenom Nom" # configurer les préférences de l'utilisateur

- git add <fichier> # Ajoute des modifications à l'index (la zone de staging ). Cela prépare les fichiers pour le prochain commit
- git commit -m "message" # Enregistre les modifications dans l'historique.
- Git status # Affiche l'état des fichiers dans le répertoire de travail et l'index.
- git push # Envoie les modifications locales vers un référentiel distant.
- git pull # Récupère les modifications depuis un référentiel distant.



## Stratégie de branches