**Sigles, Abréviations et définitions de concepts :**

SaaS : Un service SaaS (Software as a Service), ou logiciel en tant que service, est un modèle de distribution de logiciels où l’application est hébergée sur des serveurs distants (dans le cloud) et accessible via Internet, généralement avec un navigateur web ( Selon IBM).

DVCS (Distributed Version Control System): C’est un système de gestion de versions où chaque utilisateur possède une copie complète du dépôt, y compris son historique.

Cloud : Le cloud computing permet aux clients d’utiliser une infrastructure et des applications via Internet, sans les installer ni les maintenir sur site. Il repose sur la virtualisation des ressources informatiques, permettant une mise à l’échelle dynamique et une utilisation optimisée des ressources (selon IBM).

DevOps : C’ est un ensemble de pratiques qui combine le développement logiciel (Dev) et les opérations informatiques (Ops).

L’objectif de DevOps est de réduire le cycle de développement logiciel, améliorer la qualité du code, accélérer les mises en production et renforcer la collaboration entre les équipes de développement et d’exploitation (selon IBM).

CI (Continuous Integration) : L’intégration continue consiste à automatiser les tests et la fusion du code dès qu’un développeur effectue une modification dans un projet (Selon IBM).

CD : Déploiement Continu (Continuous Delivery / Continuous Deployment)

Continuous Delivery : le code est préparé automatiquement pour le déploiement (mais le déploiement est déclenché manuellement).

Continuous Deployment : le code est automatiquement déployé dès qu’il a passé les tests avec succès.

Terminal : Le terminal est une interface en ligne de commande qui permet à un utilisateur d’interagir directement avec le système d’exploitation en tapant des commandes textuelles (selon Linux Foundation)

Débogueur : Un débogueur (ou debugger en anglais) est un outil utilisé par les développeurs pour analyser, tester et corriger les erreurs (ou bugs) dans un programme informatique. Il permet d’exécuter le code étape par étape, d’observer la valeur des variables, de poser des points d’arrêt (breakpoints), et de suivre le flux d’exécution du programme (selon Microsoft).

IDE (Integrated Development Environment) c’est un logiciel qui fournit aux développeurs un ensemble d’outils intégrés pour écrire, tester, déboguer et compiler du code dans un seul environnement (selon IBM).

Open Source : Un logiciel open source est un programme informatique dont le code source est librement accessible, modifiable et redistribuable par quiconque, sous certaines conditions définies par une licence libre (comme la licence MIT, GPL, Apache, etc.). Open Source Initiative

**Introduction :**

Bien avant nous, dans les années 1970, apparu le besoin d’outils de gestion de version. Lorsqu’un projet implique plusieurs développeurs ou lorsqu’un code évolue rapidement, il devient essentiel de sauvegarder l’historique des modifications, de travailler en parallèle sans écraser les contributions des autres, de revenir à une version stable après une erreur, de tracer les auteurs de chaque modification.

Et c’est en ayant ces objectifs en tête que Marc Rochkind créa, en 1972, SCCS (Source Code Control System) le premier outil reconnu de gestion de version [11].

Cependant, cet outil n’est limité qu’à la gestion des versions des fichiers sources Unix.

Ainsi, plusieurs outils de plus en plus performant et modernes furent créés au fil du temps :

* 1982 : RCS (Reversion Control System) ;
* 1990 : CVS (Concurrent Versions System) ;
* 2000 : Subversion (SVN) ;

Ce n’est qu’en 2005 que la communauté Linux, plus précisément Linus Trovalds le fondateur de Linux, développa **Git**, un gestionnaire de version qui inclut une gestion décentralisée très performante.

Git boosta l’environnement de la gestion de version et par ricochet la gestion des projets logiciels.

Puis à partir de 2008, pour héberger et collaborer autour de projets utilisant Git, dans l’optique d’élargir sa dimension sociale, des plateformes basées sur Git furent lancée. Ce sont GitHub et GitLab.

Ces deux plateformes offrent propose plusieurs fonctionnalités toutes indispensables les unes que les autres.

Plus loin encore, aujourd’hui, les editeur de texte comme Visual Studio Code intègrent nativement Git.

VS Code en plus de ses fonctions d’écriture et de gestion efficace du code reconnait automatiquement les projets Git, offre la possibilité de se connecter à un compte GitHub ou utiliser des extensions GitHub et GitLab.

Dans la suite de ce document, nous parleront plus en détail de Git (Caractéristiques, avantages et lacunes), des plateformes affiliées à Git et de VS Code.

1. **Git :** 
   1. **Définition :**

Git est un système de contrôle de version distribué gratuit et open source conçu pour gérer tout, des petits aux très grands projets, avec rapidité et efficacité [1].

Git a été développé en 2005 par la communauté des développeurs Linux (et en particulier Linus Torvalds, le créateur de Linux), qui a choisi de façonner son propre outil en s’appuyant sur les enseignements tirés de l’utilisation de BitKeeper (un DVCS, gestionnaire de version) [3].

Depuis sa création en 2005, Git a évolué et gagné en maturité pour devenir simple d’utilisation tout en conservant ses qualités initiales. Il est incroyablement rapide, très efficace pour les grands projets [3].

* 1. **Objectifs de Git:**

L’objectif principal de Git est de permettre à un ou plusieurs développeurs de travailler sur un même projet tout en gardant un historique complet de toutes les modifications effectuées.

Cela rend les projets plus sûrs, puisque l’on peut revenir en arrière à n’importe quel moment, fusionner les changements des différentes branches du projet et éviter les conflits.

Git permet ainsi de **gérer efficacement de grands projets**, tout en permettant aux développeurs **de collaborer facilement.**

* 1. **Caractéristiques de Git :**
* **Décentralisation :**

La plupart des outils de gestion de version comme Subversion et CVS fonctionnent en mode centralisé. C’est-à-dire que les changements effectués par les utilisateurs sont envoyés à un serveur, et c’est sur ce dernier que les autres développeurs vont se synchroniser pour récupérer les changements [1].

**Git** lui, est un **gestionnaire de version décentralisée**, c’est-à-dire qu’il n’est pas nécessaire de disposer d’un serveur maître pour l’utiliser. Chacun des utilisateurs peuvent se synchroniser entre eux[1].

* **Fiabilité :**

Chaque utilisateur dispose d’une copie complète du dépôt (historique complet du projet).

Ainsi, même en cas de défaillance d’un serveur central ou d’une panne réseau, il est toujours possible de restaurer tout le projet à partir d’une copie locale.

Git a été conçu pour résister aux erreurs de disque, aux coupures réseau et aux défaillances système, garantissant ainsi la continuité du travail des développeurs.

Git n’est donc pas seulement un outil pratique pour gérer les versions, c’est aussi un outil sûr et fiable pour protéger les projets contre la perte de données et les attaques[2].

* 1. **Vocabulaire de Git :**

Voici quelques termes fondamentaux à connaître pour bien utiliser Git :

* **Dépôt** (repository) :

C’est l’endroit où est stocké le projet avec tout son historique. Il peut être local (sur ton ordinateur) ou distant (sur un serveur comme GitHub ou GitLab).

* **Commit :**

C’est un enregistrement d’une série de modifications dans le projet. Chaque commit contient un message descriptif expliquant ce qui a changé.

* **Branche** (branch) :

Une branche est une copie parallèle du projet. Elle permet de travailler sur des fonctionnalités ou des corrections sans affecter directement la version principale.

* **Fusion** (merge) :

C’est l’action de combiner les changements de plusieurs branches ensemble.

* **Clone** :

Créer une copie locale d’un dépôt distant.

* 1. **Lacunes de Git :**

Bien que Git prenne en charge le développement collaboratif dès sa sortie d’usine, de nombreux utilisateurs trouvent sa nature distribuée difficile à configurer. La plupart des organisations préfèrent disposer d’un référentiel unique, centralisé et partagé, auquel les développeurs peuvent facilement se connecter. C’est précisément ce que proposent GitHub et d’autres [5].

Git ne dispose d’aucun outil intégré de gestion ou d’authentification des utilisateurs. Si un utilisateur distant accède à votre installation Git locale, il a un contrôle total sur votre dépôt et l’historique des commits, ce qui signifie qu’il peut écraser des commits, supprimer des branches et corrompre complètement votre système de contrôle de version [5].

1. **GitHub :**

GitHub est une plateforme basée sur le cloud où vous pouvez stocker, partager et travailler avec d’autres pour écrire du code. Elle facilite la collaboration entre développeurs grâce à plusieurs fonctionnalités [4].

* 1. **Avantages de GitHub :**

En plus de rendre Git plus facile à utiliser, GitHub fournit également une variété de fonctionnalités supplémentaires que vous n'obtenez pas avec une installation simple de Git.

GitHub fournit une authentification à deux facteurs et des services d’autorisation basés sur des jetons qui régissent ce qu’un utilisateur peut faire sur un référentiel Git.

Lorsqu’un écart existe entre les besoins d’une organisation et les fonctionnalités offertes par l’outil Git, des fournisseurs comme GitHub interviennent pour combler ce manque.

* 1. **Lacunes de GitHub :**

Malgré ses nombreux atouts**,** GitHub présente également des lacunes (limites) [6]:

* **Fonctionnalités CI/CD moins complètes que GitLab**

Bien que GitHub propose GitHub Actions pour l’intégration et le déploiement continus, ces fonctionnalités sont moins centralisées et intégrées que celles de GitLab, qui offre une plateforme DevOps complète par défaut.

* **Pas d’auto-hébergement natif**

Contrairement à GitLab, GitHub ne permet pas l’auto-hébergement. Les entreprises ou institutions souhaitant héberger leurs propres dépôts doivent se tourner vers GitHub Enterprise Server, qui est complexe à mettre en place et payant [6].

* **Interface parfois trop simplifiée**

L’interface de GitHub, pensée pour être intuitive, peut limiter la personnalisation ou l’accès rapide à certaines fonctionnalités avancées, notamment pour les utilisateurs expérimentés.

* **Certaines fonctionnalités avancées sont payantes**

Les fonctions de sécurité, d’analyse de code ou de gestion d’équipe avancée sont réservées aux plans payants (Pro, Team, Enterprise) [6].

* 1. **Les fonctionnalités principales de GitHub :**

GitHub possède beaucoup de fonctionnalités dont les principales sont :

* **Hébergement de dépôts Git**

GitHub permet d’héberger des projets versionnés avec Git, qu’ils soient publics ou privés.

* **Pull Requests**

Outil collaboratif permettant de proposer des modifications, de les discuter et de les intégrer dans la branche principale.

* **Issues (Tickets)**

Système de suivi des bugs, tâches, ou suggestions, utilisé pour gérer un projet efficacement.

* **Actions GitHub (CI/CD)**

Automatisation des tests, des déploiements et autres tâches grâce à l’intégration continue.

* **Wiki**

Chaque dépôt peut contenir une documentation collaborative via un wiki intégré.

* **Pages GitHub**

Fonctionnalité permettant d’héberger des sites web statiques directement à partir d’un dépôt.

* **Forks**

Permet de copier un projet pour proposer des améliorations ou créer une version dérivée.

* **Gestion des équipes et des permissions**

Contrôle des accès selon les rôles (lecture, écriture, admin) pour sécuriser la collaboration.

* **Graphiques et statistiques**

Visualisation des contributions, de l’activité et de la structure du projet.

GitHub a développé un écosystème complet d’outils DevOps autour de Git, promettant d’améliorer la productivité des développeurs, des équipes logicielles et des organisations.

* 1. **Corrélation entre Git-GitHub :**

Lorsque l’on télécharge des fichiers sur GitHub, ils sont stockés dans un « référentiel Git ». Cela signifie que lorsque l’on apporte des modifications (ou « commits ») aux fichiers dans GitHub, Git commencera automatiquement à suivre et à gérer ces modifications[4].

Il existe de nombreuses actions liées à Git que l’on peut effectuer sur GitHub directement dans le navigateur, telles que la création d’un référentiel Git, la création de branches, le téléchargement et l’édition de fichiers.

* 1. **Différences entre Git et GitHub :**

La principale différence entre Git et GitHub est que Git est un outil de contrôle de version gratuit et open source que les développeurs installent localement sur leurs ordinateurs personnels, tandis que GitHub est un service en ligne payant (et/ou gratuit) conçu pour exécuter Git dans le cloud [5]..

Git est un logiciel. GitHub est un service SaaS en ligne. Malgré cette distinction, Git et GitHub ne sont pas des offres concurrentes. Au contraire, elles fonctionnent ensemble et se complètent [5].

1. **GitLab :**

GitLab est une plateforme DevOps complète qui permet de gérer l’ensemble du cycle de vie du développement logiciel, de la planification à la mise en production [6].

Tout comme GitHub, GitLab repose sur Git pour le suivi de versions, mais il offre aussi des fonctionnalités intégrées.

1. **Avantages de GitLab :** 
   1. **Plateforme DevOps tout-en-un :**

GitLab regroupe dans un seul outil :

* Gestion de versions (Git),
* Gestion de projets (issues, tableaux Kanban),
* Intégration et déploiement continus (CI/CD),
* Sécurité du code (audit, scan de vulnérabilités),
* Supervision et suivi des performances.
  1. **CI/CD intégré**

GitLab propose une solution native pour l’intégration continue (CI) et le déploiement continu (CD), permettant d’automatiser les tests, la construction et le déploiement des applications sans outil externe.

* 1. **Auto-hébergement :**

Contrairement à GitHub (qui est principalement cloud), GitLab peut être installé sur vos propres serveurs, ce qui est très apprécié pour les projets internes ou sensibles.

* 1. Outils de collaboration intégrés
* Suivi des tickets,
* Revue de code,
* Wiki intégré,
* Tableaux de bord d’activité.

1. **Lacunes ou limites de GitLab :**

Malgré ses nombreux avantages, GitLab présente quelques inconvénients :

* **Consommation de ressources** : L’installation auto-hébergée de GitLab nécessite des ressources serveurs importantes (RAM, CPU, stockage).
* **Courbe d’apprentissage :** Pour les débutants, l’interface et les nombreuses fonctionnalités peuvent sembler complexes.
* **Performance lente** sur certaines configurations (surtout en auto-hébergement).
* **Fonctionnalités avancées payantes :** Certaines fonctions professionnelles (monitoring avancé, sécurité renforcée) ne sont disponibles que dans les éditions premium/ultimate.

1. **Différence entre Git et GitLab, GitHub et GitLab :** 
   1. **Git Vs GitLab :**

Git est l’outil de versionnage, tandis que GitLab est une plateforme qui utilise Git pour offrir des services autour du développement logiciel [6].

Comme pour GitHub, GitLab est une plateforme web complète qui repose sur Git. Plus loin encore, elle permet non seulement d’heberger des dépôts Git mais aussi de gérer l’ensemble du cycle DevOps : planification de projet, suivi des bugs, intégration continue (CI), déploiement continu (CD), sécurité, collaboration en équipe, etc.

* 1. **GitLab Vs GitHub :**

GitLab et GitHub sont tous deux des plateformes d’hébergement de dépôts Git, mais ils présentent des différences importantes [6]:

* **Origine** : GitHub appartient à Microsoft, alors que GitLab est développé par GitLab Inc.
* **Hébergement** : GitHub est principalement basé sur le cloud, tandis que GitLab peut être auto-hébergé (installé localement dans une entreprise).
* **Fonctionnalités** **DevOps** : GitLab propose une chaîne DevOps complète intégrée (CI/CD, gestion de projet, sécurité, etc.). GitHub offre aussi ces fonctions, mais souvent via des outils externes ou via GitHub Actions pour CI/CD.
* **Licence** : GitLab propose une version open source (Community Edition), ce qui permet plus de personnalisation alors que GitHub est propriétaire.

1. **Visual Studio Code (VS Code) :**

Annoncé le 29 Avril 2015 puis publié le 18 novembre 2015, VS Code est un éditeur de code source gratuit et open source, développé par Microsoft, conçu pour être à la fois léger, rapide et puissant. Il est disponible sur Windows, macOS et Linux [7].

C’est un éditeur qui supporte plusieurs langages ( JavaScript, HTML, CSS, Python, C++, Java….etc).

Grâce à une large bibliothèque d’extensions, VScode est l’environnement idéal pour tout programmeur ou développeur.

1. **Avantages de VS Code :**

* **Gratuit et open source**

VS Code est totalement gratuit et son code source est disponible en open source, ce qui en fait un outil accessible à tous.

* **Multiplateforme**

Il fonctionne sur Windows, macOS et Linux, ce qui le rend compatible avec tous les environnements de développement.

* **Léger et rapide**

Contrairement aux IDE lourds, VS Code est rapide à lancer et n’alourdit pas le système.

* **Éditeur polyvalent**

Il supporte des dizaines de langages (JavaScript, Python, Java, C++, PHP, HTML/CSS…) avec la coloration syntaxique, l’autocomplétion et la navigation dans le code [7][8].

* **Extensions puissantes**

Une immense bibliothèque d’extensions permet d’ajouter des fonctionnalités : GitHub, GitLab, Docker, Live Server, Prettier, etc.

Intégration native de Git

VS Code permet de gérer les dépôts Git (commits, branches, push/pull…) sans quitter l’éditeur.

* **Terminal intégré**

Il dispose d’un terminal intégré, ce qui évite d’alterner entre l’éditeur et la console.

* **Débogueur intégré**

Le débogueur visuel permet d’exécuter pas à pas le code, poser des points d’arrêt, inspecter les variables, etc.

* **Interface personnalisable**

L’utilisateur peut adapter l’apparence (thèmes, icônes) et l’agencement selon ses préférences.

* **Communauté active**

Grâce à une grande communauté, de nombreuses extensions, thèmes et ressources sont disponibles.

1. **Lacunes de VS Code :**

VScode présente aussi certaines lacunes :

* **VS Code n’est pas un IDE complet :**

Contrairement à un IDE comme IntelliJ ou Visual Studio, VS Code est un éditeur de texte enrichi, ce qui signifie qu’il faut installer des extensions pour des fonctionnalités avancées (tests, déploiement, gestion de projets complexes…).

* **Performances sur gros projets**

Sur des projets très volumineux ou complexes, VS Code peut devenir lent, notamment en raison des nombreuses extensions actives.

* **Fonctionnalités inégales selon les langages**

Certaines fonctionnalités (débogage, autocomplétion, etc.) sont plus développées pour certains langages (comme JavaScript ou Python) que pour d’autres.

* **Dépendance aux extensions tierces**

Beaucoup de fonctionnalités importantes dépendent d’extensions développées par des tiers, ce qui peut poser des problèmes de compatibilité ou de sécurité si elles ne sont plus maintenues.

1. **Relation entre VScode et Git, VScode et Github, VScode et GitLab :** 
   1. **VScode et Git :**

Certaines relations existent entre VScode et Git :

* Intégration native : VS Code intègre Git par défaut. Il permet d’exécuter les commandes Git (commit, push, pull, gestion des branches, etc.) directement depuis son interface graphique.
* Terminal intégré : permet d’utiliser Git en ligne de commande sans quitter l’éditeur.
  1. **VScode et Github :**
* **Extension GitHub officielle** : permettant de se connecter à GitHub, de cloner des dépôts, créer des branches, ouvrir des pull requests, commenter du code, etc.
* **GitHub Copilot** (extension payante) **:** fournit de l’autocomplétion intelligente basée sur l’IA.
* **Gestion simplifiée des dépôts distants :** on peut publier un projet local directement sur GitHub depuis l’interface de VS Code**.**
  1. **VScode et GitLab :**
* **Extensions GitLab  :** bien que non intégrée nativement, plusieurs extensions gratuites permettent d’utiliser GitLab depuis VS Code :
* Authentification à GitLab,
* Gestion des issues, merge requests,
* Visualisation des pipelines CI/CD.
* Compatibilité Git : comme GitLab repose sur Git, toutes les commandes Git exécutées via VS Code fonctionnent parfaitement avec des dépôts GitLab.

VS Code agit comme une interface moderne et puissante pour gérer les projets versionnés avec Git, qu’ils soient hébergés sur GitHub, GitLab ou ailleurs. Grâce à ses extensions, il facilite le travail collaboratif, le suivi de version et l’automatisation du développement.

Le développement moderne, qu’il soit web ou logiciel, repose fortement sur des outils de gestion de versions et de collaboration. Git, en tant que système de versionnage décentralisé, permet aux développeurs de travailler de manière efficace, sécurisée et collaborative. Des plateformes comme GitHub et GitLab viennent enrichir cet écosystème en offrant des fonctionnalités en ligne orientées vers la gestion de projets, l’intégration continue (CI/CD), la collaboration et le déploiement [4][6].

Quant à Visual Studio Code, il s’impose comme l’un des éditeurs de code les plus puissants et polyvalents. Grâce à ses intégrations transparentes avec Git, GitHub et GitLab, il facilite toutes les étapes du développement, du codage à la publication. L’ensemble de ces outils s’inscrit également dans une logique DevOps, promouvant l’automatisation, la qualité et la rapidité du développement [9][10].

Références :

1. Dr. XPOSE. Git  : gestionnaires de versions. Université Paris-Est Marne-la-Vallée. Disponible sur : <https://monge.univ-mlv.fr/~dr/XPOSE2008/git/gestionnaires.html>. Consulté le : 27 avril 2025.
2. Git. Git – documentation officielle. Disponible sur : <https://git-scm.com/>. Consulté le : 27 avril 2025.
3. Chacon S, Straub B. Pro Git. 2ᵉ éd. Apress ; 2014. Disponible sur : <https://git-scm.com/book/en/v2>. Consulté le : 27 avril 2025.
4. GitHub, Inc. GitHub Docs. Disponible sur : <https://docs.github.com/fr>. Consulté le : 28 avril 2025.
5. TheServerSide.com. Git vs. GitHub : What is the difference between them ? . TheServerSide ; 2023. Disponible sur : <https://www.theserverside.com/video/Git-vs-GitHub-What-is-the-difference-between-them>. Consulté le : 28 avril 2025.
6. GitLab, Inc. GitLab Documentation. Disponible sur : <https://docs.gitlab.com/ee/>. Consulté le : 28 avril 2025.
7. Microsoft. Visual Studio Code documentation [Internet]. Disponible sur : <https://code.visualstudio.com/docs>. Consulté le : 29 avril 2025.
8. Wikipedia . Visual Studio Code [Internet]. Wikipedia. Disponible sur : <https://en.wikipedia.org/wiki/Visual_Studio_Code>. Consulté le : 29 avril 2025.
9. Microsoft. What is Visual Studio Code ? [Internet]. Visual Studio Code Docs. Disponible sur : <https://code.visualstudio.com/docs>. Consulté le : 29 avril 2025.
10. IBM. What is DevOps ? [Internet]. IBM Cloud Learn Hub. Disponible sur : https://www.ibm.com/cloud/learn/devops-a-complete-guide. Consulté le : 29 avril 2025.
11. Wikipédia. Source Code Control System [Internet]. 2024 [cité 1 mai 2025]. Disponible sur :

https://fr.wikipdia.org/wiki/Source\_Code\_Control\_System