

GIT & GITHUB

"Maîtrisez Git et GitHub : les outils incontournables pour collaborer efficacement et propulser vos projets vers le succès !"

Presentation by Josias Nteme





**QU'EST-CE
QUE GIT ?**



GIT

Git est un système de contrôle de version

Il est devenu un outil standard dans le développement logiciel, permettant de gérer les versions d'un projet, de suivre toutes les modifications, et de collaborer avec d'autres développeurs de manière structurée.

Contrairement à d'autres systèmes comme SVN ou CVS, Git est entièrement distribué, ce qui signifie que chaque utilisateur possède une copie complète de l'historique du projet.



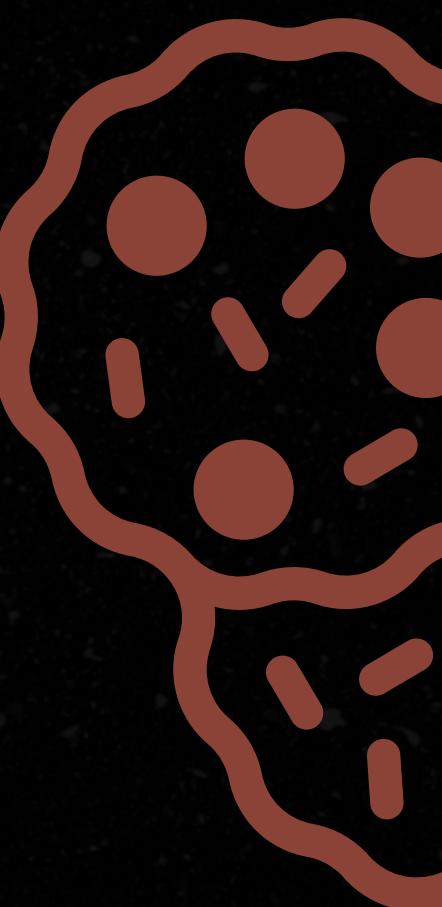
POURQUOI GIT EST-IL IMPORTANT ?

Git permet de suivre toutes les modifications faites sur un projet, offrant un historique détaillé de chaque changement.

Grâce à ses fonctionnalités de branchement et de fusion, Git facilite la collaboration en permettant à chaque développeur de travailler sur différentes fonctionnalités sans perturber le code principal.

Enfin, Git est fiable : avec chaque utilisateur possédant une copie complète du projet, les risques de perte de données sont considérablement réduits.

GitHub



QU'EST-CE QUE GITHUB ?



GitHub est une plateforme en ligne qui utilise Git pour héberger des projets de code. Il ajoute des fonctionnalités de collaboration comme la gestion des issues, les pull requests, et des outils de documentation.

En plus d'héberger des projets open-source, GitHub permet aussi aux entreprises et aux équipes de gérer des projets de manière privée et collaborative.

Utiliser GitHub simplifie la gestion des projets collaboratifs et facilite la visibilité et la contribution aux projets open-source.

CYCLE DE VIE D'UN FICHIER DANS GIT

Lorsqu'on travaille avec Git, chaque fichier passe par plusieurs étapes ou statuts dans son cycle de vie :

UNTRACKED

le fichier n'est pas encore suivi par Git.

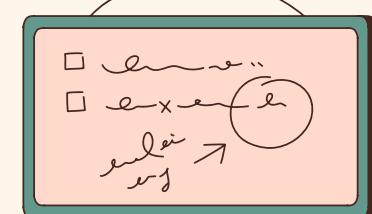
STAGED

le fichier est prêt à être enregistré dans le prochain commit.

COMMITTED

le fichier est enregistré dans l'historique du projet.

Cette gestion des statuts permet de travailler efficacement et de gérer les changements en plusieurs étapes.



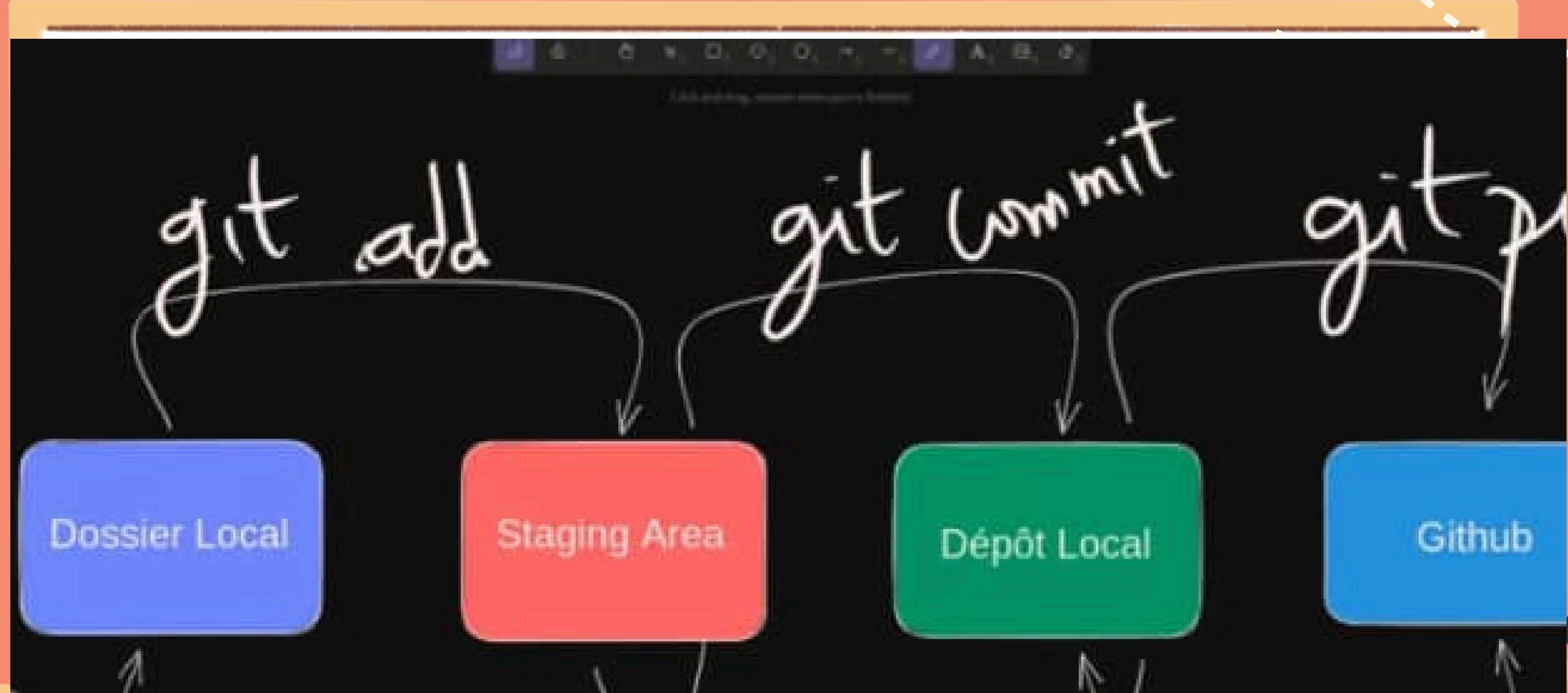
COMMANDES DE BASE DE GIT

Voici les commandes essentielles pour bien démarrer avec Git :

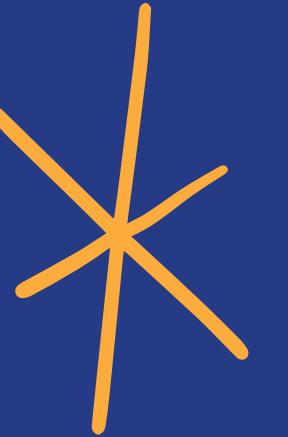
- git init : Initialise un nouveau dépôt Git dans le dossier actuel.
- git status : Vérifie l'état actuel des fichiers, indiquant ceux qui sont suivis ou non.
- git add : Ajoute les fichiers pour le prochain commit.
- git commit : Enregistre les modifications dans le dépôt.
- git log : Affiche l'historique des commîts.

Avec ces commandes de base, vous pouvez déjà commencer à travailler efficacement avec Git.





Branches dans Git



Une branche est une version parallèle d'un projet, permettant d'expérimenter ou de développer de nouvelles fonctionnalités sans affecter la version principale.

Les branches sont très utiles pour le développement collaboratif car elles permettent de travailler sur des tâches spécifiques, puis de fusionner les changements dans le code principal une fois qu'ils sont prêts.

Par exemple, vous pourriez créer une branche pour développer une nouvelle fonctionnalité, tout en gardant la branche principale stable et prête à être déployée.



Travailler avec GitHub

Pour collaborer sur GitHub, commencez par créer un dépôt en ligne et ajoutez-le comme dépôt distant de votre projet local.

Utilisez les commandes git push pour envoyer vos changements vers GitHub et git pull pour récupérer les changements des autres collaborateurs.

Le dépôt distant agit comme une référence commune pour tous les membres de l'équipe, garantissant que chacun travaille sur la version la plus récente du code.

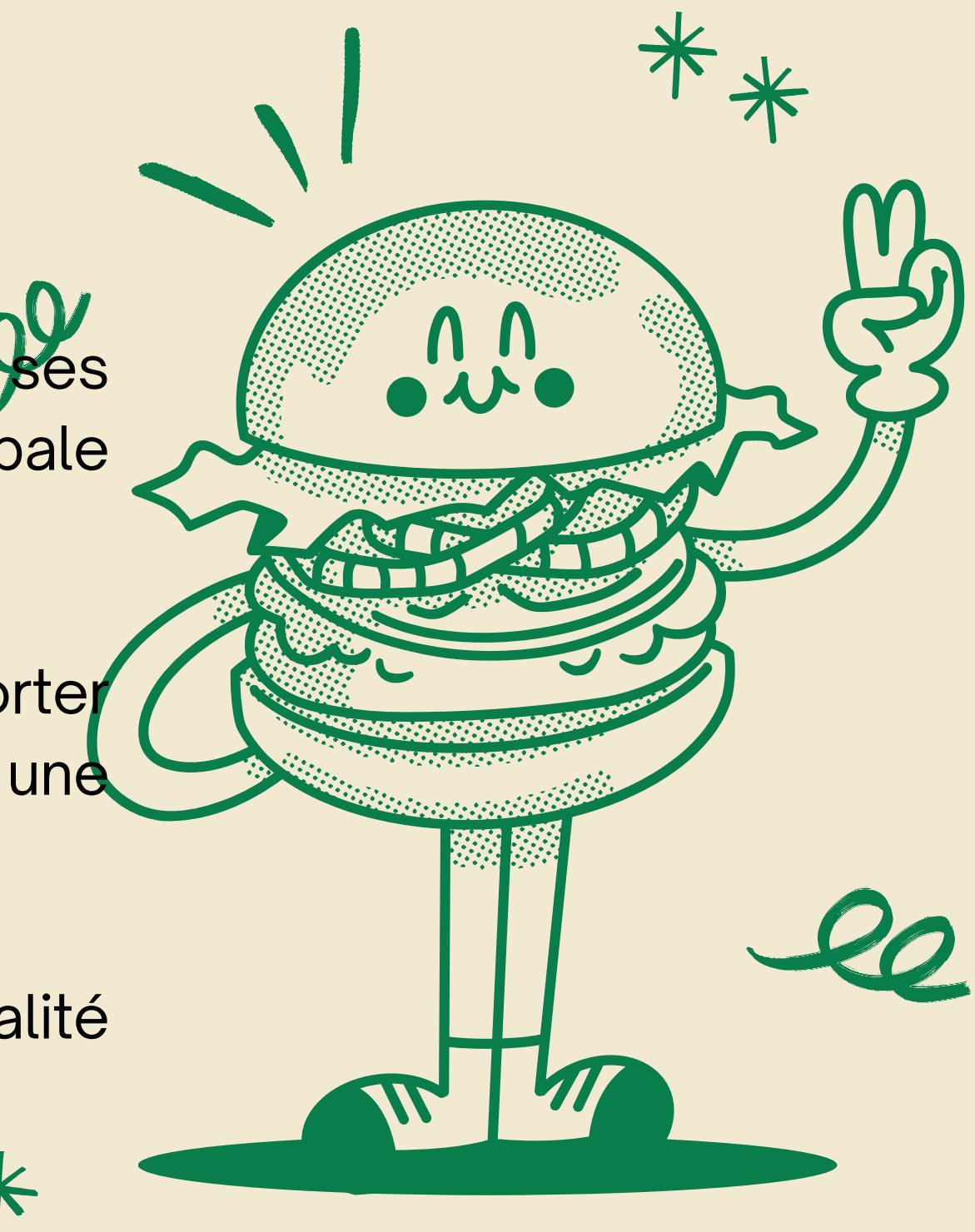


Pull Requests et Collaboration

Une Pull Request (PR) permet à un développeur de proposer ses modifications pour qu'elles soient intégrées dans la branche principale d'un projet.

Le processus de PR implique de créer une nouvelle branche, d'y apporter les modifications, de pousser la branche vers GitHub, et enfin d'ouvrir une PR pour que d'autres membres de l'équipe puissent réviser le code.

Les PR permettent aussi d'engager des discussions et d'assurer la qualité du code en facilitant les révisions.

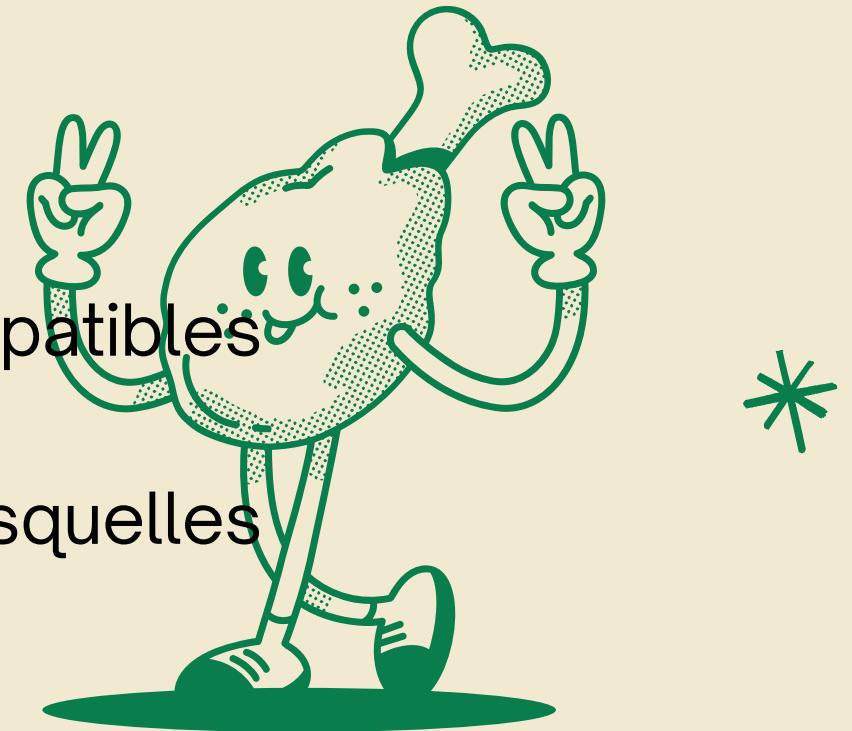




Résolution des Conflits

Les conflits de fusion surviennent lorsque deux modifications incompatibles sont apportées à une même partie du code.

Pour résoudre un conflit, il faut comparer les modifications, choisir lesquelles conserver et faire un nouveau commit pour enregistrer la résolution.



* Bien que les conflits puissent être gênants, Git offre des outils efficaces pour les identifier et les résoudre rapidement, ce qui est essentiel dans le développement collaboratif.

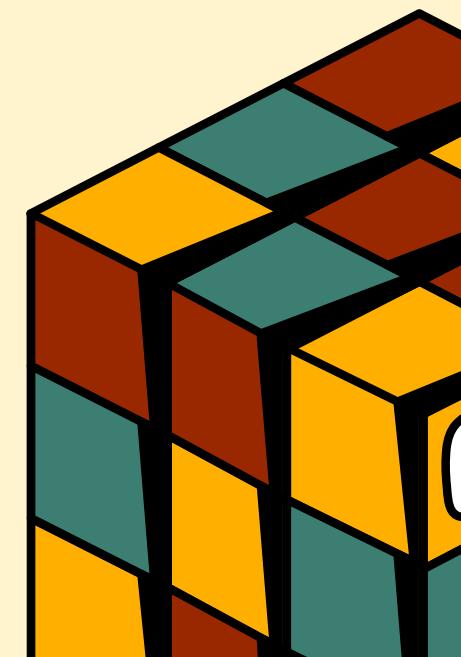


Bonnes Pratiques et Conseils

Quelques bonnes pratiques pour mieux utiliser Git et GitHub :

1. Écrire des messages de commit clairs pour faciliter le suivi des modifications
2. Utiliser un workflow adapté au projet, comme Git Flow ou GitHub Flow, pour structurer le développement.
3. Documenter le code et utiliser les issues sur GitHub pour suivre les tâches.

En suivant ces pratiques, vous pouvez maintenir un projet organisé et faciliter le travail collaboratif.





Conclusion

En conclusion, nous avons vu comment Git permet de gérer les versions d'un projet et comment GitHub facilite la collaboration sur le code.

Pour approfondir vos connaissances, je vous recommande de consulter la documentation officielle de Git et les guides GitHub, ainsi que de pratiquer avec des tutoriels en ligne.

N'hésitez pas à poser des questions et à explorer davantage ces outils pour perfectionner votre gestion de projets !

