Git et Github

Git est un système de contrôle de version distribué qui permet de suivre les modifications apportées aux fichiers et de coordonner le travail sur ces fichiers parmi plusieurs personnes. Il est principalement utilisé pour le développement de logiciels, mais peut être utilisé pour suivre les modifications de tout ensemble de fichiers.

5TTR 6TTR 6TQ Découverte

Git et GitHub

Git est un système de contrôle de version distribué qui permet de suivre les modifications apportées aux fichiers et de coordonner le travail sur ces fichiers parmi plusieurs personnes. Il est principalement utilisé pour le développement de logiciels, mais peut être utilisé pour suivre les modifications de tout ensemble de fichiers.

GitHub est une plateforme en ligne qui utilise Git pour héberger des projets de développement de logiciels. Il offre des fonctionnalités supplémentaires comme la gestion des problèmes (issues), les demandes de tirage (pull requests), et bien plus encore.

Pourquoi utiliser Git et GitHub même quand on travaille seul ?

1. **Historique des modifications** : Git permet de garder un historique complet de toutes les modifications apportées à tes fichiers. Tu peux revenir à une version précédente à tout moment.

2. **Sauvegarde** : En utilisant GitHub, tu peux sauvegarder ton code en ligne, ce qui te protège contre la perte de données locale.

3. **Collaboration future** : Même si tu travailles seul pour le moment, tu pourrais avoir besoin de collaborer avec d'autres personnes à l'avenir. Git et GitHub facilitent cette collaboration.

4. **Gestion des branches** : Tu peux travailler sur différentes fonctionnalités ou corrections de bugs dans des branches séparées, ce qui te permet de garder ton code principal stable.

Les Fondements de Base de Git

Git est un système de **contrôle de version** qui permet de suivre les modifications apportées aux fichiers et de coordonner le travail sur ces fichiers parmi plusieurs personnes. Voici les concepts de base de Git et les différentes étapes pour envoyer son code.

1. Working Directory (Répertoire de travail)

C'est l'endroit où tu travailles réellement sur ton projet, avec tes fichiers et dossiers visibles sur ton ordinateur. Quand tu ouvres ton projet dans VS Code (ou un autre éditeur), tu es dans le **working directory**.

👉 Ici, tu peux créer, modifier ou supprimer des fichiers. Mais ces changements ne sont pas encore enregistrés par Git.

---

2. Zone de Staging (Index)

La zone de staging (ou **index**) est une **zone intermédiaire**.

- Après avoir modifié des fichiers dans ton **working directory**, tu choisis ceux que tu veux préparer pour le prochain enregistrement (commit).
- C'est comme faire un panier avant de passer à la caisse : tu mets de côté ce que tu veux valider dans l'historique.

Autrement dit :

- **Working directory** = espace de travail, fichiers modifiés en attente.
- **Staging area** = sélection des modifications prêtes à être sauvegardées.

Il est possible de simplifier en configurant ton éditeur (ex. VS Code) pour que toutes les modifications passent automatiquement par la zone de staging.

---

3. Dépôt (Repository)

Le dépôt Git est la **base de données** où Git enregistre l'historique complet du projet. Chaque commit correspond à une "photo" des fichiers choisis dans la zone de staging.

- Il peut être **local** (sur ton PC).
- Ou **distant** (***remote***) sur une plateforme comme GitHub pour partager et sauvegarder ton travail.

👉 Résumé en image mentale :

- **Working directory** : tu travailles et fais des brouillons.
- **Staging area** : tu choisis ce qui va être validé.
- **Repository** : tu ranges définitivement ton travail dans l'historique.

Les opérations fondamentales

4. Commit

Un commit est une **capture instantanée** des modifications apportées à tes fichiers. Chaque commit a un identifiant unique et un **message de commit** qui décrit les modifications. Les commits permettent de **suivre l'historique des modifications** et de **revenir à une version précédente** si nécessaire.

Faire un commit signifie enregistrer les modifications préparées (stagées) dans l'historique de ton projet. Il est parfois possible de *commiter* tous les changements sans les **stager** avant.

- C’est un peu comme faire un « **point de sauvegarde** » dans un jeu vidéo.

💡 Dans la plupart des outils de développement, il est possible de faire automatiquement un commit de fichiers qui n'ont pas été mis dans la zone de staging, ce qui correspond à faire un commit de tous les changements. C'est le cas dans Visual Studio code.

5. Push

Le push est l'action d'**envoyer tes commits locaux vers un dépôt distant** (comme GitHub). Cela permet de sauvegarder tes modifications en ligne et de les partager avec d'autres personnes.

- C’est comme envoyer ton colis à la poste pour qu'il soit disponible partout où tu le souhaites.

6. Pull

Le pull est l'action de **récupérer les modifications depuis un dépôt distant** et de les fusionner avec ton dépôt local. Cela permet de mettre à jour ton dépôt local avec les dernières modifications apportées par d'autres personnes.

© 2025 Mr. Lambrechts https://v2.ttrinfo.be 4 | 6

Étapes pour Envoyer Son Code

1. **Ajouter des Modifications à la Zone de Staging**
    - Lorsque tu modifies des fichiers, tu dois les **ajouter** (add) à la zone de staging pour les préparer à être commités. Cela te permet de choisir quelles modifications seront incluses dans le prochain commit.
2. **Créer un Commit (au plusieurs)**
    - Une fois que tu as ajouté les modifications à la zone de staging, tu peux créer un commit. Un commit enregistre les modifications dans l'historique du dépôt avec un message de commit qui décrit les modifications.
3. **Pousser les Commits vers le Dépôt Distant**

- Après avoir créé un ou plusieurs commits, tu peux les pousser vers le dépôt distant. Cela envoie tes modifications en ligne et les rend disponibles pour d'autres personnes.

4. **Tirer les Modifications depuis le Dépôt Distant**
    - Avant de commencer à travailler, il est bon de tirer les dernières modifications depuis le dépôt distant. Cela met à jour ton dépôt local avec les dernières modifications apportées par d'autres personnes.

Exemple de Workflow

1. **Ajouter des Modifications à la Zone de Staging**
    - tu modifies un fichier `index.html`.
    - tu ajoutes ce fichier à la zone de staging.
2. **Créer un Commit**
    - tu crées un commit avec un message comme "Mise à jour de la page d'accueil".
3. **Pousser les Commits vers le Dépôt Distant (*push*)**
    - tu pousses (*push**) tes commits vers le dépôt distant sur GitHub.
4. **Tirer les Modifications depuis le Dépôt Distant (*pull*)**
    - Avant de commencer à travailler sur une nouvelle fonctionnalité, tu tires (*pull*) les dernières modifications depuis le dépôt distant pour t'assurer que ton dépôt local est à jour.

En suivant ces étapes, tu peux gérer tes modifications de code de manière efficace et collaborative avec Git.

Synthèse des notions essentielles à maîtriser :

✅ **Git** permet de contrôler les versions de ton projet. ✅ **GitHub** est une plateforme pour stocker et partager ton code avec Git. ✅ Configure Git avec ton identité une seule fois. ✅ Le workflow principal : `git status`, `git add`, `git commit`, `git push`, `git pull`. ✅ Visual Studio Code facilite l'utilisation quotidienne de Git.

Vidéo à la une à regarder sur Youtube: ▶ http://youtu.be/r8jQ9hVA2qs