Atelier avancé n°2 : Découverte de Git, GitHub & GitLab

Présentation de Git, GitHub et GitLab :

Git :
Git est un système de contrôle de version open source. Concrètement, c'est un outil qui te permet de traquer tous les fichiers de ton projet. Chaque modification de fichier est alors détectée par Git et versionnée dans une version instantanée. Un historique de modification va être disponible sur ton projet. Tu vas pouvoir le consulter et pourquoi pas même revenir en arrière dans le temps !
Pour en savoir plus : https://www.jesuisundev.com/comprendre-git-en-7-minutes/

Concrètement, voici pourquoi il est fortement conseillé d'utiliser Git :

- Permet la modularisation de son projet : Git permet une modularisation aisée de son projet. Que ce soit pour un projet de petite, moyenne ou grande taille, le besoin est sans cesse présent de développer des fonctionnalités en parallèle. Git permet à l'aide des *branches* de facilement atteindre ses fins.

- Permet d'annuler ses erreurs : dans votre éditeur de code, vous avez l'habitude d'annuler vos modifications (CTRL+Z, CTRL+Y, …). Mais dès que vous fermez l'éditeur, impossible d'annuler ce que vous venez à peine de modifier. Git apporte une solution avec les *commits* avec la possibilité de passer d'un *commit* à un autre.

- Permet de travailler en mode déconnecté : nous n'avons pas internet à tout moment. Nous ne sommes pas au boulot tout le temps. Mais quand on utilise Git, puisqu'on a tout le code sur notre poste de travail, on peut continuer le développement partout où on sera.

- Permet d'éviter les pertes de données : si vous travaillez à plusieurs sur un projet, concilier vos modifications –quand on n'utilise pas un outil moderne– est très critique. Il est difficile à vue d'œil d'identifier les modifications apportées par un collaborateur, de comparer deux fichiers, ou de faire l'intégration de ces modifications. Avec Git, vous identifiez clairement les modifications des autres collaborateurs, ce qu'ils ont ajouté ou retranché, et pourquoi ils l'ont fait. Et lorsque ultérieurement une modification devient critique, vous pouvez repartir en arrière sur une version préalable du fichier grâce à l'historique des modifications.

GitHub et Gitlab :

Il est important de comprendre la différence entre Git et Github ou Gitlab. Git est un outil qui permet de gérer différents projets en les envoyant sur un serveur. Ce dernier est connecté à l'ordinateur d'autres développeurs qui envoient leur code et récupèrent le vôtre. Toute personne qui travaille sur un projet est connectée avec les autres, tout est synchronisé.

Quant à Github et Gitlab, il s'agit d'un logiciel, ou plateforme. Leur rôle est d'héberger ces différents projets qui utilisent Git. Concrètement, ils proposent une interface graphique qui simplifie l'utilisation.

Pour en savoir plus sur les différence entre Github et Gitlab : https://www.ionos.fr/digitalguide/sites-internet/developpement-web/gitlab-vs-github/

Travail à faire :

Pour commencer vous allez installer git dans votre environnement de travail (dossier créé lors du premier atelier), puis vous allez utiliser GitHub pour visualiser cet environnement.

1) Démarrer le terminal Windows PowerShell puis utiliser la commande **wsl**.
2) Vérifier que git est installé en utilisant la commande **git --version**. Si git n'est pas installé, utiliser la commande **apt install git** pour l'installer.
3) Créer vous un compte GitHub sur ce site : https://github.com/
4) Utiliser les commandes pour fournir à Git votre username et email :
    - **git config --global user.name "votre_username"**
    - **git config --global user.email "votre_email"**
5) Utiliser la commande **git config --list** pour vérifier que les informations saisis sont correctes.
6) En utilisant le terminal et la commande **cd** déplacez vous jusqu'au dossier tdpython1 sur votre bureau que vous avez créé dans le précédent atelier avancé.
7) Utiliser la commande **git init MyGit** pour créer votre premier repository Git
8) Utiliser la commande **ls** et vérifier la bonne installation de votre repository Git
9) Utiliser la commande **touch README** pour créer un fichier texte
10) Utiliser la commande **nano README** pour éditer le fichier texte et écrire le message suivant dans le fichier : "Ceci est mon premier Git Repository".
11) Par la suite déplacez un ou plusieurs fichiers Python dans le dossier MyGit
12) Pour "push" les fichiers dans votre repository il faut commencer par préparer votre commit en utilisant la commande add sur les fichiers que vous souhaitez transférer :
    - **git add README**
    - **git add fichier1.py**
    - …
13) Ensuite il vous suffit d'utiliser la commande **git commit -m "Ceci est mon premier commit"** pour transférer les fichiers.
14) Rendez vous sur votre compte GitHub et créer un nouveau repository avec **le même nom** que le repository que vous avez créé sur votre bureau, ici "MyGit".
15) Par la suite générer un token sécurisé sur Git en allant dans le menu : https://github.com/settings/tokens. Sauvegarder ce token dans un fichier texte pour ne pas le perdre.
16) Utiliser la commande **git remote add origin https://token@github.com/user_name/MyGit.git** pour connecter le git présent dans

votre dossier pythontd1 à votre repository sur GitHub. Penser à remplacer token user_name par votre nom d'utilisateur au niveau de l'URL ci-dessus.

17) Enfin, utiliser la commande **git push origin master** pour transférer votre fichiers sur GitHub
18) Vérifier que les fichiers ont été transférés sur GitHub

Travail à faire :

En utilisant les mêmes commandes et en vous aidant d'internet configurer un autre repository git avec GitLab dans un dossier pythontd2 sur votre Bureau.