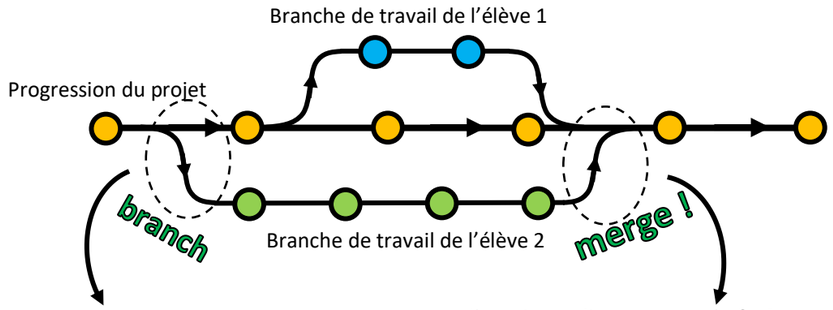
git et GitHub pour la gestion de projet

Partie 1 : principes et mise en œuvre des branches

git

'' Logiciel de gestion de versions (ou VCS en anglais, pour version control system) est un logiciel qui permet de stocker un ensemble de fichiers en conservant la chronologie de toutes les modifications qui ont été effectuées dessus. Il permet notamment de retrouver les différentes versions d'un lot de fichiers connexes.'' *(Wikipedia.fr)*

Principe général de fonctionnement de git



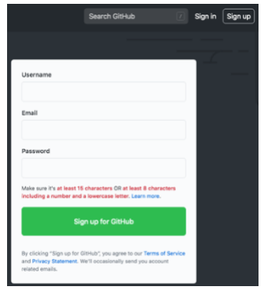
Chaque collaborateur peut créer une copie du projet dans une « branche » et peut ainsi **travailler sans impacter le « master » (ou main) du projet.**

Il peut ensuite proposer de **fusionner son travail avec le « main »** du projet. La fusion s'effectue après concertation du groupe de travail (propriétaire du repo).

GitHub

GitHub est un service d'hébergement utilisant le logiciel **Git**.
La plateforme représente la plus vaste plateforme de projets au monde. C'est une filiale de Microsoft depuis 2018.
Il est accessible par l'URL : http://www.github.com
Réseau social « numérique » utilisé par de nombreuses entreprises.

Se créer un compte sur la plateforme GitHub.com

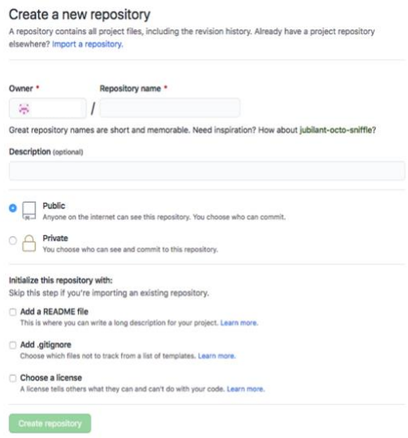


Cliquer sur « **Sign up** » et compléter les informations demandées.

GitHub

Créer un repository

En cliquant sur **Create repository**, on accède au paramétrage de notre premier projet. On choisit son nom, son statut **public/privé** (modifiable ensuite). On peut rajouter un fichier ReadMe (en markdown) de présentation.
Une fois créé, on accède au paramétrage du *repo* en cliquant sur *Settings*.
Dans ce menu, on ajoute des collaborateurs au projet qui pourront créer des *branchs* et proposer des *commits*.
On peut également activer la création de *GitHub Pages* qui permet d'héberger des pages web statiques liées au projet.



Installer GitHub Desktop

Originellement Git utilise des lignes de commande à saisir dans le Terminal. Cependant au lycée, l'invite de commande Windows est désactivé. On peut utiliser **GitHub Desktop,** une GUI (Graphical User Interface), moins pointue mais qui propose les commandes principales qui suffiront pour débuter.

Desktop

Utilisation basique de Desktop

GitHub desktop demande de se loguer sur le compte GitHub soit au démarrage soit dans le menu **options (ou préférences)**.
Il est nécessaire alors de créer une copie locale des fichiers. On a 3 possibilités :
- **créer un *repo* local** : un dossier local est créé, envoyé ensuite en ligne.
- **cloner un *repo* existant en ligne** : copier les fichiers du projet en ligne dans un dossier du disque dur local,
- **Ajouter un *repo* existant** : fait le lien entre une copie locale d'un *repo* et le compte GitHub en créant un *repo* contenant les fichiers.

Pour avoir accès aux dossiers et aux fonctions Git, il est nécessaire de s'identifier en saisissant vos identifiants dans le menu **options (Windows)** ou **préférences (Mac).**

Il faut voir **GitHub Desktop** comme un programme de centralisation et de synchronisation des modifications effectuées sur le *repo* de travail :
- **il identifie** les fichiers modifiés, les modifications apportées (sur les scripts de codes essentiellement), les dossiers et les fichiers ajoutés ou supprimés,
- **il permet de créer des branches**, de créer des *commits*, de les envoyer en ligne et de proposer d'appliquer ces modifications à la branche principale (*Pull Request).*

Il ne permet pas d'éditer directement les fichiers, il est nécessaire d'utiliser un IDE ou le programme d'édition habituel (Word, SW, Matlab,…)

Parlez-vous Git ?

- **Repository (« *Repo* »)** : dossier qui contient tous les fichiers sources et suit toutes les modifications apportées en créant un historique au fil du temps. Un projet est un *repo*.
- **Branch** : nouvelle version du *repo* créée par chaque développeur afin d'isoler son travail des autres. Bref, une ligne de développement indépendante.
- **Cloner un repo** : action de créer une copie du projet sur le disque de l'ordinateur (en local) afin de le modifier. Un dossier par *repo* est créé.
- **Commit** : action d'enregistrer les modifications dans le *repo* local. Toujours illustrée par un message décrivant les modifications apportées. Conserve un historique des modifications et permet de revenir en arrière.
- **Push** : action d'envoyer vers le *repo* distant (branch ou main), les modifications effectuées dans le *repo* local (clone).
- **Merge** : action de fusionner une branche avec la branche principale « master ».
- **Fetch :** action de récupérer toutes les données de la branche courante qui n'existent pas encore dans votre version locale mais ces données ne seront pas fusionnées avec la branche locale (il faut effectuer un *merge* ensuite).
- **Pull :** action qui regroupe les commandes *fetch* et *merge.* Elle télécharge les données des *commits* qui n'ont pas encore été récupérées dans votre branche locale puis fusionne ces données (équivalent à *Fetch* si les modifications apportées ne sont pas trop lourdes sinon peux engendrer des conflits).
- **Fork** : Action qui consiste à cloner le *repo* directement en ligne. On réalise un *fork* dans le but de modifier le projet en cours et de proposer ses modifications au propriétaire du *repo.*

Pourquoi utiliser GitHub pour gérer un projet ?

- **permet un suivi des tâches de chaque collaborateur**,
- La fonction *Pull Request* pour effectuer un *merge* est l'occasion pour le groupe de travail de **réaliser des revues de projet régulières pour valider les modifications apportées,**
- **permet de mettre en œuvre les méthodes agiles** : le développement par branches permet une **réactivité maximale dans l'adaptation des caractéristiques du projet** dans le cas de redéfinition du (ou des) besoins client,
- un ***wiki*** (création de pages collaboratives) intégré permet de développer un rapport de projet commun en Markdown, langage balisé libre, simple,
- GiHub permet d'héberger des pages web statiques de présentation du projet directement à partir du *repo*,
- GitHub propose son propre outil de gestion de projet basé sur la méthode Kanban (voir Partie 2).

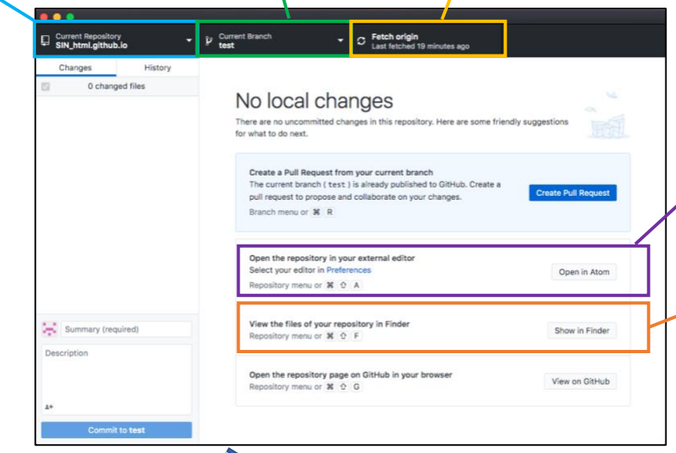
git

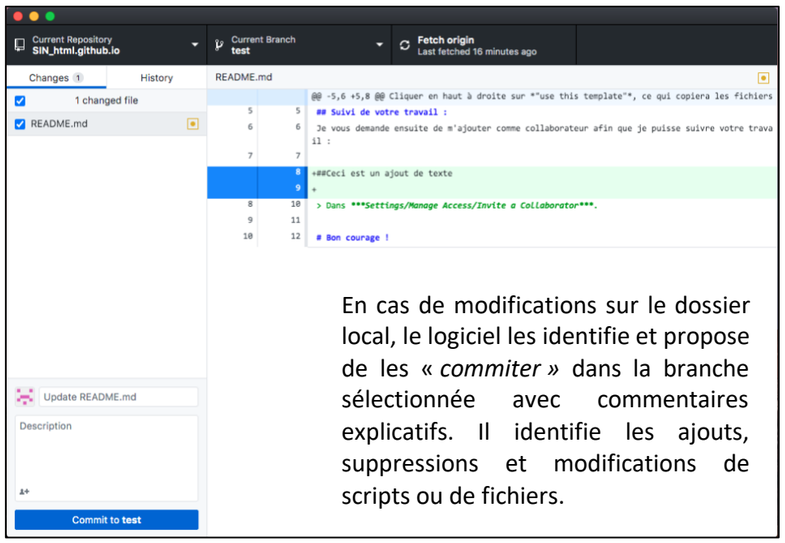
Interface de Desktop

Gestion des repositories

Gestion des branches

Mise à jour du clone local

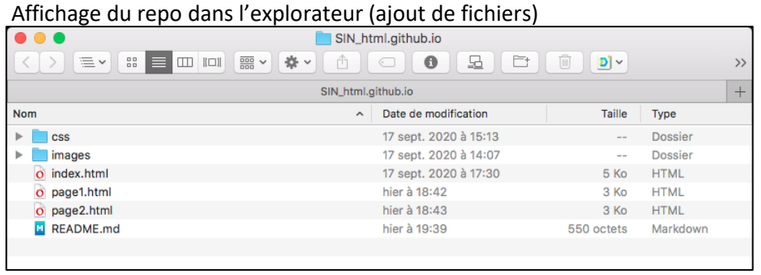




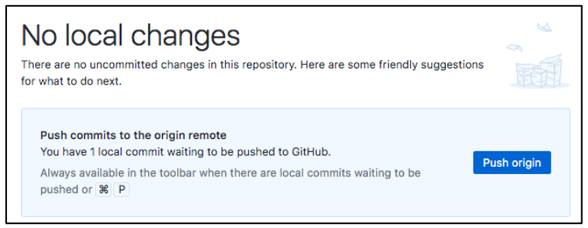
En cas de modifications sur le dossier local, le logiciel les identifie et propose de les « *commiter* » dans la branche sélectionnée avec commentaires explicatifs. Il identifie les ajouts, suppressions et modifications de scripts ou de fichiers.

Affichage du repo dans un IDE (scripts)

Affichage du repo dans l'explorateur (ajout de fichiers)



On peut envoyer le *commit* en ligne en cliquant sur *Push* (publication dans la branche sélectionnée) …



… ou annuler le ou les *commit effectués.*