

Partie 1 : principes et mise en œuvre des branches

et pour la gestion de projet

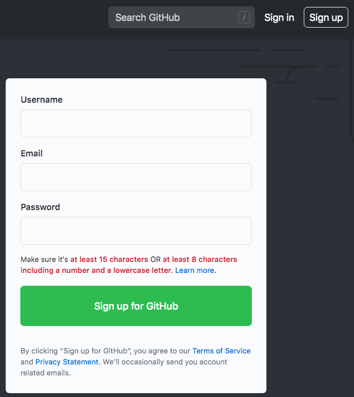


**Installer** [**GitHub Desktop**](https://desktop.github.com/) **et** [**Atom**](https://atom.io/)

Originellement Git utilise des [lignes de commande](https://gist.github.com/aquelito/8596717) 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.



**Se créer un compte sur la plateforme** [**GitHub.com**](http://www.github.com/)



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

‘’ 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)*



**Utilisation basique de**

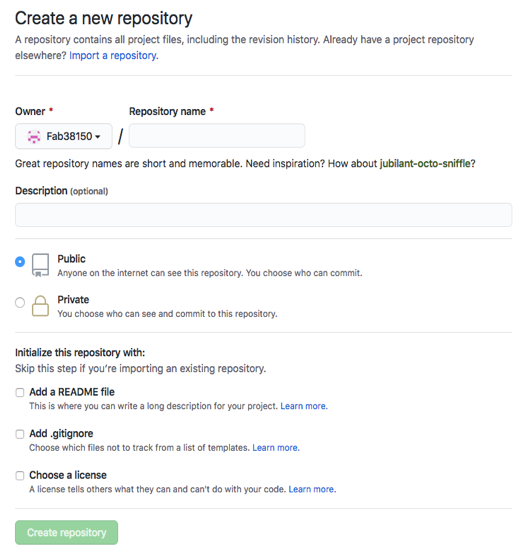
GitHub desktop demande de se loguer sur le compte GitHub soit au démarrage soit dans le menu **options (ou préférences)**.

Il est nécessaire alors de créer une copie locale des fichiers. On a 3 possibilités :

- créer un *repo* local qui sera ensuite envoyé en ligne,

- Cloner un *repo* existant en ligne : copier les fichiers du projet en ligne dans un dossier du disque dur,

- 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.



**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.

* **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.*

**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>

Il suffit de se créer un compte en cliquant sur **Sign up.**

**Glossaire**

branch

merge !

**Principe général de fonctionnement de**



Progression du projet

Branche de travail de l’élève 1

Branche de travail de l’élève 2

Chaque collaborateur peut créer une copie du projet dans une « branche » et peut ainsi **travailler sans impacter sur 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.