

## Exercice 2 : partagez votre projet avec GITHUB

Nous allons à présent voir comment partager votre code via le service GitHub et les commandes « `git push` » et « `git pull` ». Cela vous permettra notamment d'y accéder depuis n'importe où (e.g., depuis votre ordinateur professionnel et votre ordinateur personnel), mais aussi de « l'ouvrir » à une personne tierce (ici votre binôme de TP) . Nous travaillerons pour cela à la fois en ligne de commande, et via le site GitHub : <https://github.com/>.

**1. Commencez par vous créer un compte GitHub si vous n'en disposez pas.**

Pour cela il suffit de vous rendre sur le site web (<https://github.com>), de choisir un identifiant et un mot de passe et de donner une adresse mail.

**2. Créer un nouveau « repository » que vous appellerez « tp-git\_votre-nom », destiné à accueillir le projet de l'exercice 1.**

Notez que ce n'est pas strictement nécessaire de leur donner le même nom, mais ça simplifiera les choses.

Laissez le « public » et ne demandez pas à GitHub d'inclure dès maintenant de `README.md`

**3. Nous allons à présent synchroniser votre projet local et le repository GitHub.**

Pour cela :

- Récupérer l'URL du projet sur GitHub, qui devrait être du type [https://github.com/votre-identifiant/tp-git\\_votre-nom.git](https://github.com/votre-identifiant/tp-git_votre-nom.git)
- Ajouter un serveur distant « remote » dans votre projet (en local) via la commande « `git remote` », en l'occurrence, par l'appel « `git remote add origin URL` ».
- « Pousser » votre projet sur le serveur distant (i.e., GitHub) via la commande « `git push` », en l'occurrence par l'appel « `git push origin master` ».
- Vérifier que le repository a bien été mis à jour sur GitHub (moyennant une éventuelle actualisation de la page).

**4. Nous allons à présent voir comment mettre à jour le dépôt local.**

Pour cela :

- Modifier le fichier « `info.txt` » via le site web (e.g., rajouter une information). Notez que pour enregistrer cette modification, vous devez faire un nouveau « `commit` » (via GitHub). Vous pouvez le vérifier dans l'historique des commit.
- Retournez dans votre terminal et faites un « `git status` ». Voyez-vous les modifications apportées ?
- Pour mettre à jour votre version locale il faut utiliser la commande « `git pull` », en l'occurrence « `git pull origin master` ». Vérifier que le fichier a bien été mis à jour.

5. **Nous allons à présent faire l'inverse : modifier le dépôt distant à partir d'une modification faite localement.**

Pour cela :

- a. Modifier le fichier « `info.txt` » en local et l'enregistrer sur le dépôt (i.e, via « `git add` » + « `git commit` »)
- b. Retourner sur l'interface GitHub et afficher le fichier « `info.txt` » : a-t-il été mis à jour ?
- c. Vous constaterez que ce n'est pas le cas ce qui est corroboré par le fait que l'historique des commit n'a pas évolué (alors qu'il compte un commit de plus en local) : comparer le résultat de « `git log` » avec ce qui est référencé sur GitHub.
- d. Pour soumettre les modifications sur le serveur distant, il faut passer par la commande « `git push` », en l'occurrence par l'appel « `git push origin master` ». Vérifier sur l'interface GitHub que le fichier a été mis à jour (ainsi que l'historique des commit).

6. **Nous allons maintenant illustrer comment travailler à plusieurs.**

Pour cela :

- a. Ajouter votre binôme collaborateur de votre projet, et accepter d'être collaborateur du projet de votre binôme (cela passe par un email de validation)
- b. Récupérer le projet de votre binôme via la commande « `git clone` » en utilisant l'URL transmise par votre binôme
- c. Modifier (dans l'image de son projet) son fichier « `info.txt` » (e.g., en y ajoutant vos informations personnelles ou en modifiant les siennes). Enregistrer les changements (add + commit) et « pousser » sur le serveur distant (i.e., via « `git push origin master` »).
- d. Vérifier que vos modifications ont bien été prises en compte sur l'interface GitHub de votre binôme.
- e. A l'inverse, une fois que votre binôme a bien « poussé » ses modifications sur votre dépôt GitHub, mettez à jour votre version locale (i.e., sur votre PC) via la commande « `git pull origin master` ».

**Pour aller plus loin :**

1. **En pratique, on passe souvent directement par GitHub pour créer un projet.**

Pour cela :

- a. Créer un nouveau projet nommé « **tp-git\_votre-nom\_repo-2** » via votre compte GitHub. Laissez le public, mais demandez à GitHub d'y inclure un **README.md**
- b. « Clonez » le projet localement via la commande « **git clone** »
- c. Ajouter un fichier (ici « **toto.txt** ») au projet et « poussez » le sur le serveur distant. Notez que par défaut, quand on « clone » un dépôt distant, git nous crée un « remote » appelé « origin » liée à notre branche « master » : il faut donc utiliser la commande « **git push origin master** ».

2. **En pratique on risque de rencontrer des incompatibilités ou conflits (même quand on travaille seul sur un projet) si on modifie les fichiers en utilisant à la fois l'interface web et le système de fichier local.**

Ce serait par exemple le cas si on modifiait le README.md via le site web et le reste des fichiers en local. Heureusement, GIT permet de gérer cela assez simplement :

- a. Placez-vous dans le dépôt que vous venez de créer et modifier le fichier **README.md** via l'interface GitHub.
- b. Modifier le fichier que vous avez ajouté (ici « **toto.txt** ») dans votre terminal, puis enregistrer (add + commit) les modifications.
- c. Jusque-là tout va bien. Essayer maintenant de « pousser » votre modification sur le serveur distant via la commande « **git push** ». Git refuse car la version distante est plus à jour que votre version locale.
- d. La solution consiste à mettre à jour votre version locale via la commande « **git pull** »
  - Notons qu'avant de tout rapatrier d'office on peut aussi faire un « **git fetch** » -- en l'occurrence « **git fetch origin master** » -- pour rapatrier en local la nature des changements effectués et les explorer avant de les implémenter pour de bon via « **git pull** ».
- e. La force de Git est d'avoir gardé en mémoire le commit que vous vouliez faire. Pour s'en convaincre, on peut lancer la commande « **git status** » qui nous dit que notre version locale est « en avance » par rapport à la version distante de pas 1 mais 2 commits.
  - Ce deuxième commit est en fait lié au fait d'avoir dû passer par « git pull » pour synchroniser les deux versions, et qui a donc laissé une trace dans l'historique des commit (voir « **git log** » pour s'en convaincre).
- f. Notre commit étant toujours en cours, il suffit cette fois de « pousser » nos modifications via la commande « **git push** » (en l'occurrence « **git push origin master** ») pour les envoyer sur le serveur distant. Vérifier que c'est bien le cas.

**Pour aller moins loin...**

Notez que toutes ces opérations peuvent être réalisées sans ligne de commande, via un éditeur graphique appelé « GitHubDesktop » : <https://desktop.github.com/>. Installez-le et « poussez » un nouveau fichier sur GitHub via cette application.