

Atelier Git

Tinaël Devresse

Par groupe de 2 ou plus, vous allez désormais vous exercer à l'utilisation de Git. Cet atelier a pour objectif de vous faire découvrir l'outil de manière pratique, en passant par toutes les étapes basiques de l'utilisation de l'outil.

Afin de bien comprendre les commandes basiques, il vous est demandé d'utiliser le Command Line Interface (CLI).

Dans un contexte professionnel, vous serez plus souvent invités à utiliser les logiciels de l'entreprise (intégration Git dans Visual Studio, GitKraken, etc.).

1. Installer Git

Afin d'installer Git (pour Windows), vous pouvez télécharger l'exécutable sur <https://git-scm.com/downloads> et suivre l'assistant d'installation. Les options par défaut sont adaptées à la plupart des utilisateurs.

Si vous avez un Mac ou un Linux, vous pouvez vous référer au tutoriel disponible ici: <https://www.atlassian.com/git/tutorials/install-git>.

2. Créer un compte GitHub

Afin de rendre votre futur dépôt disponible à vos camarades, chaque personne du groupe doit posséder un compte GitHub. Vous pouvez vous créer un compte en suivant le formulaire d'inscription à cette page-ci: <https://github.com/signup>.

3. Les démarches à réaliser

La suite de cet atelier va se reposer sur le contenu des tutoriels d'Atlassian. En effet, c'est une ressource très complète et correcte, autant en termes d'explications théoriques que de détails pratiques. Et comme je suis développeur, je ne réinvente pas la roue!

Le contexte est le suivant: vous créez un site web dont vous souhaitez partager le code source à vos amis. Votre site web se compose initialement d'une seule page HTML : `index.html`. Soyez inventifs mais ne partez pas trop loin dans la définition de votre page web. Après tout, on est là pour Git, pas pour votre page web! 😊

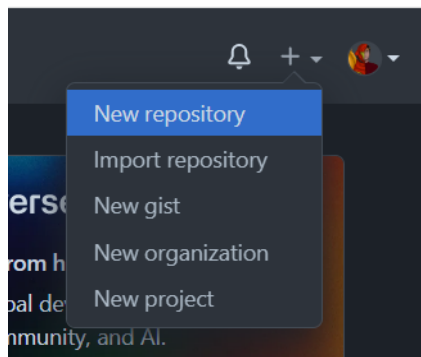
3.1. Gestion du dépôt

Afin de partager votre projet web à vos amis, vous devez d'abord en faire un dépôt. A cette fin, suivez les étapes suivantes et créez un dépôt!

3.1.1. Créer un dépôt

Puisque vous travaillez à plusieurs, vous utiliserez donc plusieurs ordinateurs. Pour cette première partie, **un** étudiant peut créer un dépôt distant. Les étapes de création d'un tel dépôt sont :

- vous connecter sur GitHub.com
- depuis la page d'accueil, vous pouvez cliquer sur le + en haut à droite et sélectionner le bouton "créer un dépôt"



- donner un nom à votre dépôt puis appuyer sur "créer"

Create a new repository


A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Repository template

Start your repository with a template repository's contents.


No template ▾


Owner * **Repository name ***

 HunteRoi ▾ / atelier-git ✓

Great repository names are short and memorable. Need inspiration? How about [animated-system?](#)

Description (optional)

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.


☐ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: None ▾

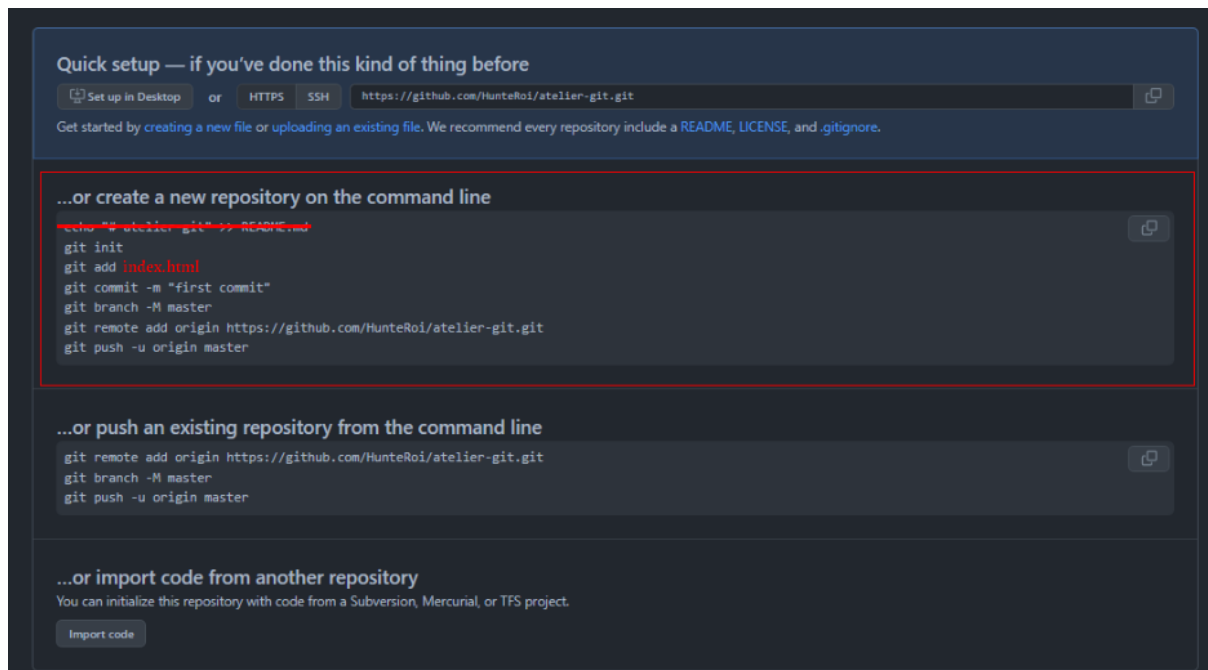
Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

License: None ▾

 You are creating a public repository in your personal account.

Create repository

Une fois le dépôt distant créé, la seconde étape sera d'y pousser votre projet web. Pour se faire, il faut suivre le premier encadré affiché sur GitHub qui vous explique comment créer un dépôt local et le lier avec un dépôt distant pour ensuite y pousser du contenu:



N'hésitez pas à regarder les détails de la création d'un dépôt sur Atlassian (<https://www.atlassian.com/git/tutorials/setting-up-a-repository/git-init>) si vous vous posez des questions sur les commandes utilisées!

N'oubliez pas que votre projet doit comporter une page intitulée `index.html` ! Votre premier commit doit donc envoyer sur votre dépôt distant votre première version de ce fichier HTML.

Attention!

Lors de la création d'un dépôt distant, seul le propriétaire a les pouvoirs de modification. Afin de partager cette lourde responsabilité, il ne faut surtout pas oublier d'inviter vos camarades à collaborer avec vous! Suivez le tutoriel suivant pour cette étape: <https://docs.github.com/en/account-and-profile/setting-up-and-managing-your-personal-account-on-github/managing-access-to-your-personal-repositories/inviting-collaborators-to-a-personal-repository>

3.1.2. Récupérer un dépôt

Pour récupérer un dépôt existant, vous pouvez le cloner. Cette étape est également expliquée dans Atlassian (<https://www.atlassian.com/git/tutorials/setting-up-a-repository/git-clone>).

3.2. Changements

À partir d'ici, tout le monde a une copie du projet sur son ordinateur, et ce même projet existe sur GitHub. C'est le moment d'apporter des modifications! L'un de vous va modifier le fichier HTML pour y ajouter du contenu.

3.2.1. Sauvegarder vos changements

Cette même personne va devoir sauvegarder ses changements pour les partager aux autres. Pour sauvegarder ces changements sur votre copie du dépôt, suivez la partie concernant “git add” et “git commit” sur Atlassian

(<https://www.atlassian.com/git/tutorials/saving-changes>).

N’oubliez pas de faire un “git push” par la suite, pour envoyer vos changements sur le dépôt distant (<https://www.atlassian.com/git/tutorials/syncing/git-push>) !

3.2.2. Récupérer les derniers changements

Pour les autres étudiants, c’est le moment de récupérer ce qui a été sauvegardé. Afin de récupérer le contenu fraîchement poussé, il faut réaliser un “git pull”. Cela est expliqué dans la catégorie “synchronisation” sur Atlassian:

<https://www.atlassian.com/git/tutorials/syncing/git-pull>.

4. Collaboration

Vous avez expérimenté la création, le clonage et la modification dans un dépôt distant. Cela dit, en règle générale, il n’y a pas qu’un seul développeur qui travaille sur le projet pendant que les autres attendent la fin du travail du premier. Tout le monde bosse, parfois même sur les mêmes fichiers!

4.1. Conflit de fusion

Lorsque plusieurs personnes travaillent de leur côté sur un même fichier, il est possible d’avoir un conflit. Ce phénomène s’appelle un *merge conflict*. C’est un problème connu et facile à éviter lorsqu’une équipe travaille main dans la main en utilisant un VCS¹.

Vous allez expérimenter un conflit de fusion et allez tenter de le corriger. Pour se faire, chaque étudiant va modifier le fichier index.html de son côté. Vous allez ensuite recommencer la partie 3.2.1 Sauvegarder vos changements à tour de rôle.

La seconde personne qui tentera d’envoyer ses changements sur le dépôt distant tombera dans le cas d’un conflit de fusion. Vous trouverez plus d’informations sur comment régler votre souci ici: <https://www.atlassian.com/git/tutorials/using-branches/merge-conflicts>.

4.2. Créer une branche

Une fois le conflit réglé, regardons à un sujet important: créer des branches! Cette partie sur la création et l’utilisation de branches est expliquée sur Atlassian à cette page

<https://www.atlassian.com/git/tutorials/using-branches>.

Votre but ici est de créer chacun une branche et d’ajouter un fichier html différent puis de l’ajouter comme lien (en utilisant l’élément HTML ancre <a>) dans votre index.html.

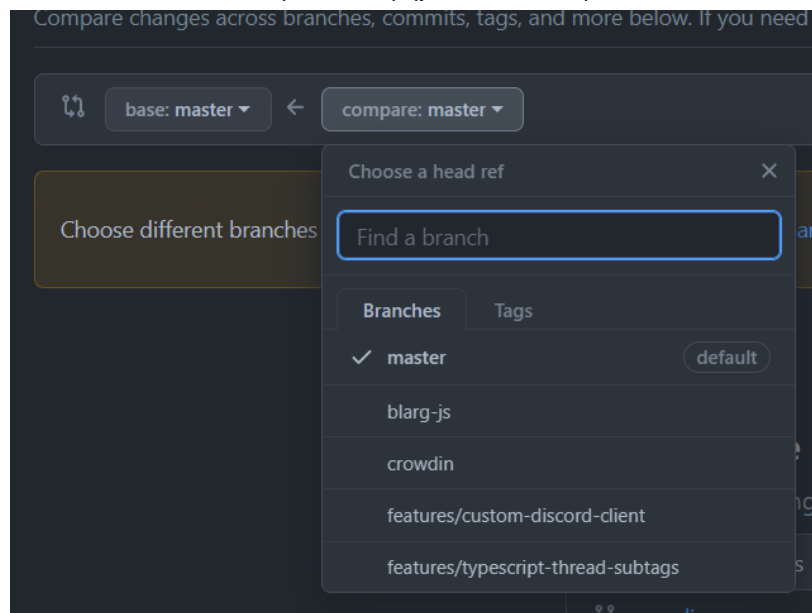
¹ VCS : Version Control System

4.3. Pull Request

La théorie relative aux pull requests (PR) est expliquée sur la page suivante:

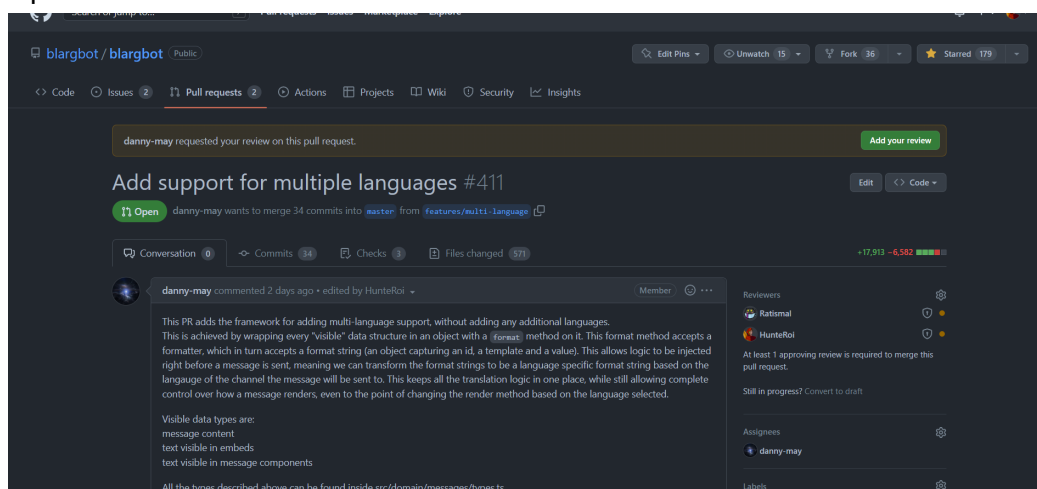
<https://www.atlassian.com/git/tutorials/making-a-pull-request>. En ce qui concerne la pratique pure et dure sur GitHub, voici les quelques étapes à suivre:

- sur la branche que vous avez créé au point 4.2, ajoutez vos changements (`git branch` suivi d'un `git checkout` pour la création, suivis de `git add` et de `git commit`);
- pousser cette branche sur le dépôt distant (`git push`);
- se rendre sur la page de son dépôt distant, dans l'onglet "Pull Requests";
- créer une nouvelle PR en ayant comme source votre branche (partie "compare") et comme cible la branche *master* (ou *main*) (partie "base").



Et voilà! Le tour est joué. Il ne reste plus qu'à laisser vos collègues se charger de réaliser une relecture, et peut-être de gérer toute la partie "fusion"!

Voici à quoi cela ressemble sur GitHub:



5. Tout ce qui n'a pas été abordé

Beaucoup de choses n'ont pas été abordées. Non pas par manque d'intérêt ou d'importance, mais bien parce que Git est un outil complet, et GitHub (comme tous ses "semblables") est très fourni également, et qu'il m'est donc impossible de vous expliquer tout en quelques heures.

Ci-dessous, une petite liste non-exhaustive de points importants sur lesquels vous devriez prendre le temps de vous documenter et faire vos propres essais:

- le "git commit --amend" (<https://www.atlassian.com/git/tutorials/rewriting-history>)
- les 3 types de "merge commits" (<https://rietta.com/blog/github-merge-types/>)
- le "stash" (<https://www.atlassian.com/git/tutorials/saving-changes/git-stash>)
- le "rebasing" (<https://www.atlassian.com/git/tutorials/merging-vs-rebasing>)
- les métadonnées d'un dépôt
(<https://www.atlassian.com/git/tutorials/inspecting-a-repository>)
- les "hooks" (<https://www.atlassian.com/git/tutorials/git-hooks>)

Vous pouvez également refaire l'entièreté de ce labo en utilisant un client Git comme GitKraken (vous verrez, c'est beaucoup plus rapide!).