

# Introduction à Git et GitHub

## 1 Git

Git est un logiciel de gestion de versions décentralisé (distributed version control). Comme tous les logiciels de ce type, son but premier est de permettre à plusieurs utilisateurs de travailler sur un ensemble de documents (en général, les sources d'une application informatique, et par extension les documents en texte brut), de modifier ces derniers et de partager leurs modifications, tout en gardant une trace des différentes révisions (et donc de pouvoir retrouver une version antérieure du document).

ID	Message	Author	Authored Date	Committer
42b4d4d	merge branch 'master' of ssh://abercot@git.isara.fr:2222/isara-net/projects.git	abercot	3 hours ago	abercot
78c3b6b	évaluation module BDD	abercot	5 hours ago	abercot
a5d5d4f	FE : affichage du site d'origine des élèves et regroupement selon ce critère dans les interventions des experts Programme Fx	chertel	5 hours ago	chertel
a5d5d4f	percoureur	abercot	5 hours ago	abercot
c5987a	[francois.francos] : réécriture du dialogue de réservation.	francois	22 hours ago	francois
3068145	[francois.francos] : on booking : réservation de salles	francois	4 days ago	chertel
a408053	insertion intervention salles	francois	4 days ago	chertel
2033101	[francois.francos] : Merge remote-tracking branch 'origin/master' into francois_wordcloud	francois	23 hours ago	francois
8d200f	Application PSE21AL	abercot	23 hours ago	abercot
3d3c05b	api 44 mobility	abercot	25 hours ago	abercot
08b5d05	Fonctionnalité application m2so étudiant	francois	25 hours ago	francois
77a3b03	gestion du word cloud étudiants	francois	25 hours ago	francois
4011381	contribution de la liste du nuage	francois	4 days ago	francois
a07b6d3	interface de saisie de mots	francois	4 days ago	francois
501a45d	dialogue de saisie pour les word TAG	francois	4 days ago	francois
03b23b	Word TAG : DAO + manager + saisie des mots	francois	4 days ago	francois
50703af	Word tag DAO	francois	4 days ago	francois
e262113	Merge branch 'master' of ssh://abercot@git.isara.fr:2222/isara-net/projects.git	abercot	4 days ago	abercot
6c744b6	mobilité.m	abercot	4 days ago	abercot
a891206	percoureur	abercot	4 days ago	abercot
07b3d13	Force la suppression des inscriptions péda même avec des notes.	chertel	5 days ago	chertel
a5d5d4f	Correction installation Erreur	chertel	5 days ago	chertel
1c1b69f	reporteur ITECH	abercot	5 days ago	abercot
2bc356a	asc	abercot	5 days ago	abercot
ec5d8e1	asc	abercot	6 days ago	abercot
27250ef	exports percoureur	abercot	6 days ago	abercot
1a77a08	mq bulletin server	abercot	6 days ago	abercot
a5d5d4f	Merge branch 'master' of ssh://abercot@git.isara.fr:2222/isara-net/projects.git	abercot	6 days ago	abercot
1a77a08	évaluation ASC	abercot	6 days ago	abercot

FIGURE 1 – Différents "commits" pour Isar@net

Le contributeur doit gérer les modifications des documents à 4 niveaux différents :

- Le dossier de travail (**Working Tree**) : dossier de la machine du contributeur qui contient les fichiers en cours de modification. Il n'y a aucune notion de gestion de versions à ce niveau.
- La zone de transit (**Staging Area**) : elle contient des liens vers les documents du Working Tree que le contributeur veut "commiter" (enregistrer une nouvelle version).
- Le dépôt (**Repository**) local : c'est l'ensemble des révisions des différents documents connues localement par le contributeur.
- Le ou les dépôts distants, contenant l'ensemble des révisions connues sur d'autres machines.

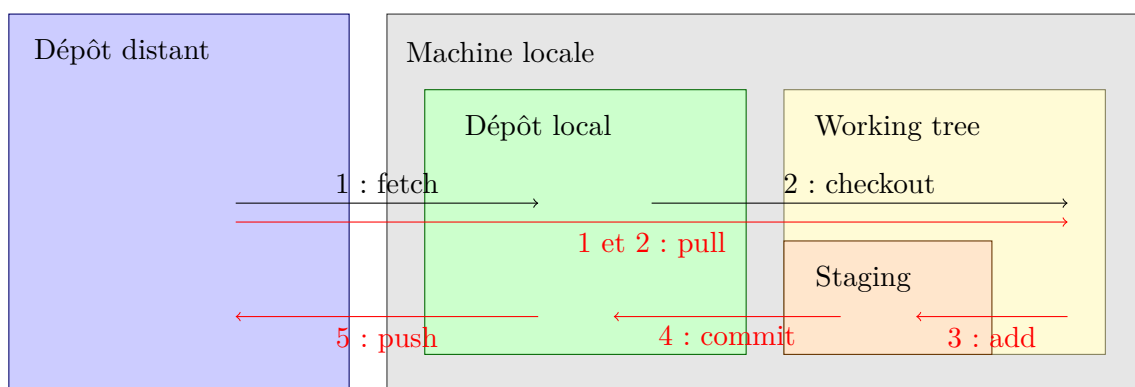


FIGURE 2 – Différents niveaux de dépôts et principales commandes de Git

Typiquement, un contributeur va récupérer à partir d'un dépôt distant la dernière version d'un document dans son dépôt local et dans son dossier de travail (**pull**), modifier ce document, le "marquer" comme modifié et prêt à être archivé (**add**), réaliser l'enregistrement de la nouvelle version dans son dépôt local (**commit**) et éventuellement pousser ces modifications dans un dépôt distant (**push**) chez un autre contributeur ou un serveur.

Avant de poursuivre, vérifiez que Git 2.21.0 (ou plus récent, disponible ici <https://git-scm.com/>) est installé sur votre machine. Pour cela, ouvrez un shell/invite de commandes (sous Windows10, tapez "cmd" dans la zone de recherche) et tapez la commande suivante :

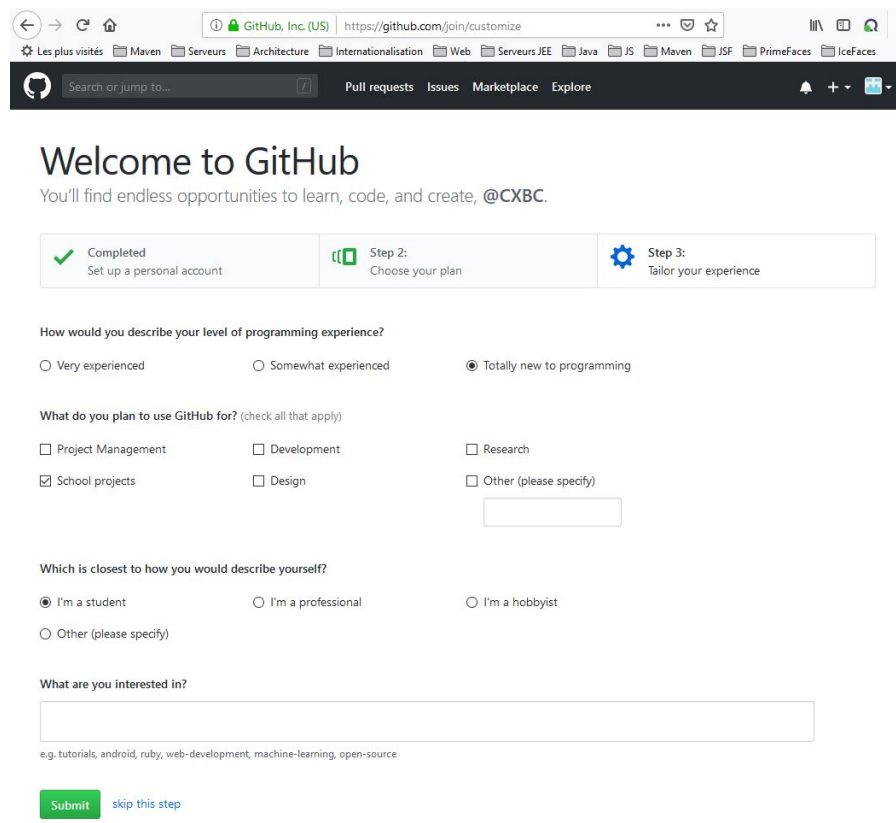
```
git --version
```

## 2 GitHub

### 2.1 Ouverture d'un compte GitHub

GitHub est un service Web d'hébergement et de gestion de développement de logiciels, utilisant le logiciel de gestion de versions Git. Nous allons donc l'utiliser pour héberger et partager les différents dépôts pour la suite des TD.

Dans un premier temps, créez un compte gratuit sous Github en cliquant sur **Sign up** sur la page d'accueil du site (<https://github.com/>).



The screenshot shows the GitHub sign-up page at <https://github.com/join/customize>. The page is titled "Welcome to GitHub" and includes a progress bar with three steps: "Step 1: Completed Set up a personal account", "Step 2: Choose your plan" (current step), and "Step 3: Tailor your experience". The "Step 2" section contains several questions:

- How would you describe your level of programming experience?**
  - ☐ Very experienced
  - ☐ Somewhat experienced
  - ☒ Totally new to programming
- What do you plan to use GitHub for? (check all that apply)**
  - ☐ Project Management
  - ☐ Development
  - ☐ Research
  - ☒ School projects
  - ☐ Design
  - ☐ Other (please specify)
- Which is closest to how you would describe yourself?**
  - ☒ I'm a student
  - ☐ I'm a professional
  - ☐ I'm a hobbyist
  - ☐ Other (please specify)
- What are you interested in?**  
  
e.g. tutorials, android, ruby, web-development, machine-learning, open-source

At the bottom, there are two buttons: "Submit" (green) and "skip this step" (blue).

FIGURE 3 – Dernière étape de la création du compte GitHub

Après la confirmation de l'adresse email, vous pourrez accéder à l'écran d'accueil du site, à différents tutoriaux, à la possibilité de créer de nouveaux dépôts, etc.

## 2.2 Création et utilisation d'un premier dépôt

Créez un dépôt IsaraFirstRepo dans GitHub en gardant les paramètres par défaut.

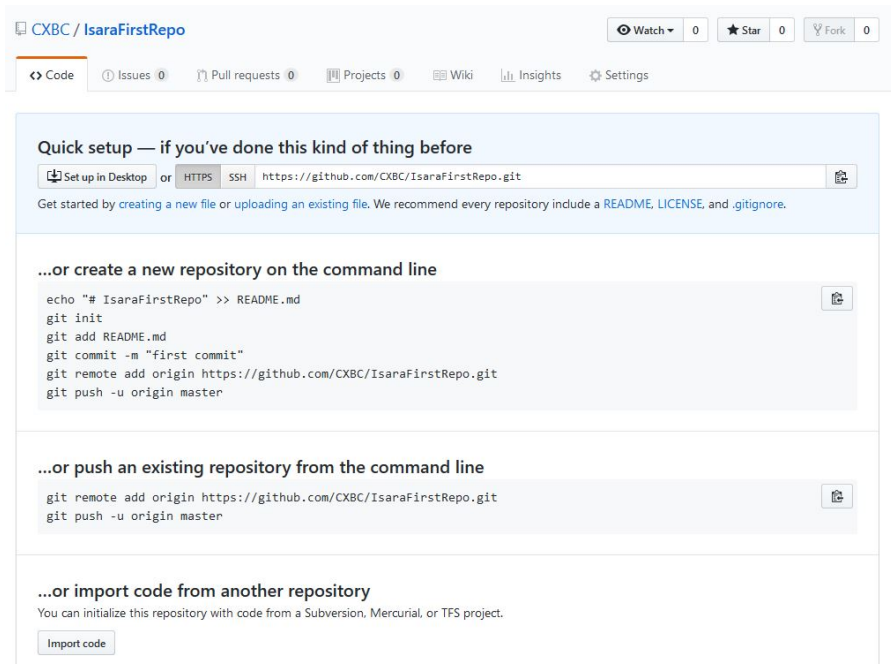


FIGURE 4 – Nouveau dépôt créé

Dans le schéma de principe de Git, ce dépôt GitHub correspond à un dépôt distant. Il faut donc désormais créer un dépôt local (sur le disque de votre machine) par clonage de ce dépôt. Dans une invite de commandes, créez un nouveau dossier et y cloner le dépôt distant avec les commandes suivantes :

```
mkdir TD
cd TD
git clone https://github.com/{GitHubUser}/IsaraFirstRepo.git
```

Un nouveau sous-dossier est créé avec le nom du dépôt. Comme cela est d'ailleurs indiqué sur la page d'accueil de votre dépôt sous GitHub, il est conseillé d'ajouter un fichier Markdown *README.md* pour donner des informations générales sur votre dépôt.

Créez, par exemple avec Notepad++, un nouveau fichier texte *README.md* dans le dossier de votre dépôt (qui correspond en fait au Working Tree de votre dépôt) et tapez-y simplement `"*Hello World!*"`.

Dans l'invite de commande de votre machine, tapez ensuite :

```
cd IsaraFirstRepo
git add README.md
git commit -m "Premier commit pour le README.md"
git push
```

Actualisez la page d'accueil Web de votre dépôt sous GitHub : le fichier *README.md* y apparaît bien et son contenu, formaté (GitHub reconnaît le format *.md*), est même affiché comme description de votre dépôt.

Vous avez édité localement ce fichier et utilisé les commandes de Git pour le transférer sur votre dépôt distant sous GitHub, mais, dans le cas d'un fichier Markdown et en particulier pour le *README.md*, vous auriez pu aussi modifier directement ce fichier sous GitHub, dans l'interface Web. Faites le à présent, puis récupérer localement la dernière version du fichier en tapant dans une invite de commande :

git pull

Nous avons découvert les principales commandes de Git dans un cas simple mais l'intérêt n'est évidemment pas de modifier un document d'information en local avant de le partager : de nombreux services de partages et d'édition de documents en ligne existent avec un "workflow" plus simple.

Mais ce type de processus est fondamental dans le cas d'un travail collaboratif sur un logiciel, projet contenant des milliers de fichiers en général compilés, assemblés, exécutés et/ou interprétés localement, dont les sources doivent être sauvegardées et versionnées avec les contributions des différents développeurs travaillant parfois dans des pays différents.

L'usage de ces différentes commandes Git peut être simplifié par l'utilisation d'une interface graphique, soit indépendante (Git Extensions par exemple dans le cas général, GitHub Desktop pour des projets hébergés sous GitHub, etc.), soit intégrée à un environnement de développement (Visual Studio, Eclipse, R Studio, etc.).

### 2.3 Fork d'un dépôt d'un autre utilisateur

GitHub permet également de créer facilement un nouveau dépôt à partir d'un projet mis à disposition gratuitement par un autre utilisateur. On parle alors de **fork** ou d'embranchement. Le nouveau dépôt devient indépendant du premier au moment du fork, il ne s'agit pas d'une collaboration au premier projet.

Dans le navigateur où votre session GitHub est ouverte, tapez l'adresse :

<https://github.com/CedricBerteletti/IsaraS8OpenDeepLearning>

Vous visualisez alors le dépôt dans lequel se trouve certains scripts Python que nous allons utiliser dans les prochains TD (sur Python et le Deep Learning). Créez un fork de ce dépôt en cliquant sur le bouton idoine : vous obtenez alors un nouveau dépôt contenant tous les fichiers du dépôt d'origine, et même l'historique des modifications effectuées sur ces fichiers.

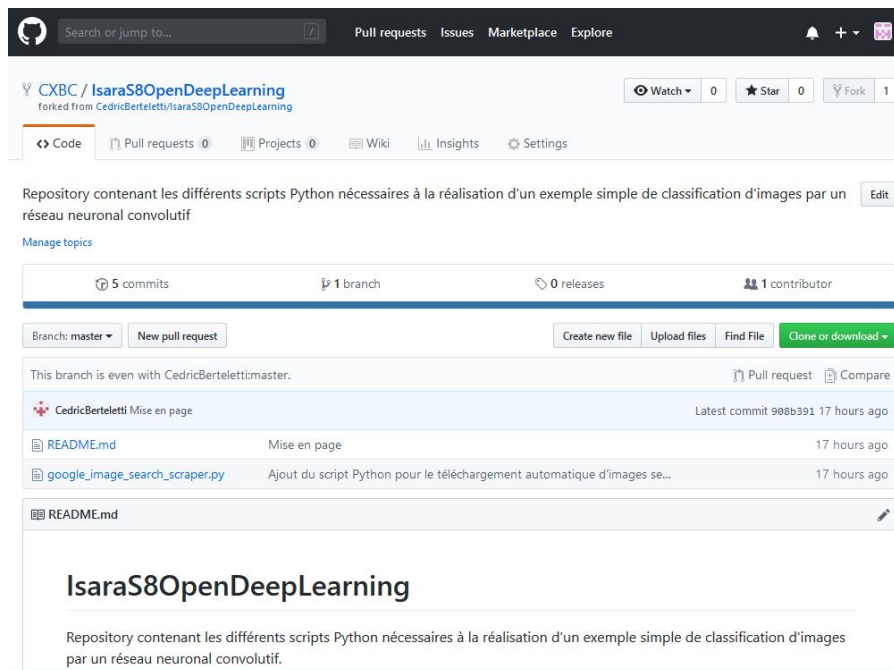


FIGURE 5 – Nouveau dépôt obtenu après le fork

Vous pouvez maintenant créer un clone local de ce dépôt distant GitHub comme dans 2.2, modifier le script à volonté, voire l'exécuter dans votre environnement Python (dans un prochain TD).

## 2.4 Travail collaboratif avec GitHub

Formez un binôme : nous allons simuler un travail collaboratif simple sur le premier script Python que nous utiliserons : *image\_search\_scraper.py*.

Ce script nous servira à télécharger automatiquement des images sur Internet en se basant sur certains mots clés correspondant aux différentes catégories d'un thème général (par exemple, "les animaux du Kenya"), définis par cette ligne du script :

```
query = ["gazelle thomson", "gazelle grand", "monkey", "giraffe",  
        "lion", "leopard", "elephant", "rhinoceros", "hyppopotame"]
```

Choisissez un thème commun avec votre binôme puis chacun indépendamment 1 ou 2 catégories de ce thème. Déterminez enfin sur lequel de vos deux dépôts vous allez travailler. Dans les options de ce dépôt, ajoutez comme collaborateur l'autre élève.

Chaque binôme va ensuite remplacer les différents animaux dans le tableau passé à *query* par les 2 ou 3 catégories choisies en respectant le workflow défini dans la *figure 2* : s'assurer d'avoir la dernière version avec un **pull**, modifier le fichier et publier les modifications, localement et à distance.

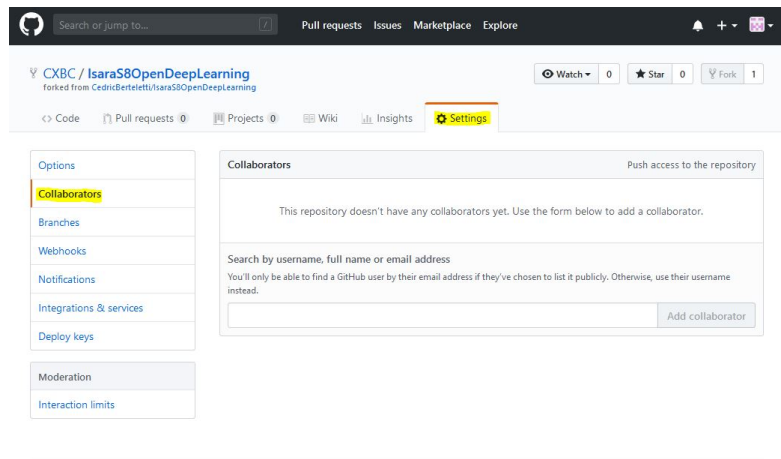


FIGURE 6 – Ajout d'autres utilisateurs GitHub au projet

## 3 Pour aller plus loin

### 3.1 Résolution des conflits

Dans une situation de collaboration plus complexe, il est possible que plusieurs collaborateurs modifient en parallèle un même fichier, *ie* qu'un second utilisateur ait réalisé un **pull** avant le **push** avec les mises à jour du premier utilisateur.

Dans ce cas, par rapport à la version précédente sur le dépôt distant, il existe deux versions concurrentes sur les machines des deux utilisateurs, ce qui crée un conflit.

Le second collaborateur doit alors résoudre ces conflits en comparant les différentes versions et en intégrant ses modifications en respectant celles de son collègue (par exemple avec un éditeur de texte permettant d'afficher et comparer deux versions d'un document, tel que WinMerge).

### 3.2 Gestion des branches

Enfin, il est souvent confortable, notamment pour des modifications lourdes ou longues, de travailler dans une **branche** indépendante de la branche principale des révisions (en générale appelée **master**). On travaille alors sur cette branche sans impact sur le travail de ses collègues et, lorsque l'on est sûr des modifications à apporter, on les fusionne avec la branche principale (**merge**).

Pour approcher ces concepts et les manipuler un peu, vous pouvez compléter au moins les quatre premiers "niveaux" (Séquence d'introduction de l'onglet Main) de ce tutoriel en ligne : <https://learngitbranching.js.org/>.

### 3.3 Expérimenter avec le support de Git dans Visual Studio Code