

TP 01 - Git, GitHub & GitHub Desktop

Objectifs du TP

- L'utilisation du Git et du GitHub
- L'utilisation du GitHub Desktop

Documentation supplémentaire

- [GitHub QuickStart](#)
- [GitHub Desktop](#)
- [Git en bref](#)
- [Git cheatsheet](#)

Git et GitHub

Git est un système de gestion de code source et de versionnage qui permet le travail sur un projet software. Cela fonctionne généralement sur la ligne de commande; pour Linux et MacOS, il n'y a aucun problème mais les utilisateurs de Windows ne peuvent pas utiliser "cmd" et il y a besoin d'installer "Git Bash".

GitHub est une plateforme en ligne basée sur Git, que les développeurs peuvent utiliser pour stocker et versionner leur code source. Git est l'utilitaire utilisé, et GitHub est le serveur et l'application Web sur lesquels il s'exécute.

Création d'un compte GitHub (si vous n'en avez pas déjà un)

Si vous avez déjà un compte sur GitHub, assurez-vous d'avoir une **photo** avec vous sur votre profil, **votre prénom et votre nom**.

Si vous n'avez pas un compte, allez sur [GitHub](#). Vous pouvez en créer un, en utilisant l'adresse de e-mail universitaire ou l'adresse personnelle.

Recommandation Si vous utilisez votre adresse e-mail personnelle, vous pouvez également ajouter votre adresse e-mail universitaire pour bénéficier du pack étudiant GitHub. GitHub offre aux étudiants de nombreux avantages qui sont normalement payés. Si vous souhaitez l'utiliser, vous pouvez suivre les étapes de ce [document](#).

Comment créer un compte?

Saisissez un nom d'utilisateur(username), une adresse e-mail et un mot de passe pour votre compte. Pour valider votre compte, accédez à votre e-mail(inbox). Vous trouverez un e-mail expliquant comment valider le compte nouvellement créé. Vérifiez section spam, aussi au cas où vous n'auriez rien reçu dans votre inbox.

Installer Git Bash

Vous devez aller sur [Git](#) et téléchargez la version de Git compatible avec votre OS. Après ça, suivez les étapes d'installation de Setup.

Réglages de base pour Git

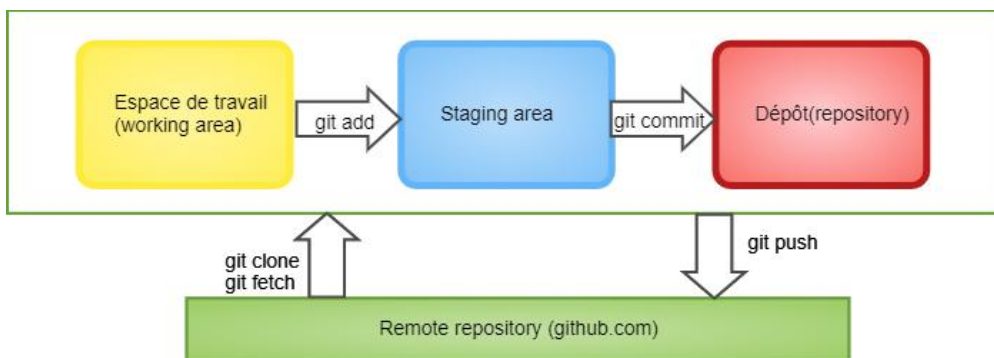
La première étape pour utiliser Git est de configurer votre nom et votre adresse e-mail avec les commandes suivantes:

```
user:~$ git config --global user.name "Prenome Nume"
```

```
user:~$ git config --global user.email "adresa_de_email@example.com"
```

Vous devez remplacer "Prenome Nume" avec votre prenom et nom "adresa_de_email@example.com" avec votre adresse e-mail!

La structure générale du Git



Working area est le repository dans lequel vous travaillez, il contient de nombreux fichiers, bien que tous ne puissent pas faire partie du projet final.

Staging area c'est comme un espace incomplet, c'est le lieu où vous pouvez ajouter la version d'un ou plusieurs fichiers que vous souhaitez enregistrer, c'est-à-dire faire partie de votre projet.

Repository c'est tout ce qui se trouve dans la *staging area* et représentera une nouvelle version du projet.

Repository

Le projet est stocké dans un **repository software**. Le repository contient des fichiers de projet: code source, fichiers de configuration. Il est généralement accompagné d'un fichier **README.md** qui contient des informations sur le projet: quel est le but du projet, comment il est compilé, sur quelles plates-formes il s'exécute. Le repository est de deux types: **locale** et **remote**. Ils peuvent être interconnectés et se référer en fait au même projet. Le référentiel local est celui que nous avons sur notre ordinateur, tandis que celui distant est stocké sur un serveur (dans notre cas GitHub). C'est juste une différence de perspective entre les deux, ils ne diffèrent pas techniquement. Habituellement, dans un projet Git / GitHub, il existe un repository central (remote) et plusieurs repositories secondaires (locaux), un pour chaque développeur de l'équipe de projet. Parmi les opérations les plus importantes avec un repository sont: init, fork, clone.

Des commandes de base

git status

Affiche l'état des modifications locales.

git add.

Ajouter des modifications à staging area.

git commit -m "message about the changes made"

Ajouter les modifications déjà du staging are dans le repository local.

git push

"Pousser" les changements de local vers github(remote repository)

git pull

Prenez les changements de remote repository.

git clone

Cloner un repository dans un nouveau répertoire.

Recommandation Pour mieux comprendre comment on utilise ces commandes, lire la documentation qui se trouve au début de TP!

Commit vs Push

Chaque commit regroupe un ensemble d'ajouts et de modifications effectués par un développeur de projet. En ayant le commits dans le repository on peut gérer le projet plus facilement, c'est à dire:

- revenir à un commit précédent (souvent même le dernier) si les derniers changements "gâchent" le projet
- pour voir qui est l'auteur de certains changements
- pour créer une branche de développement séparée à partir d'un commit précédent, sur laquelle essayer une nouvelle fonctionnalité, sans affecter le reste du projet.

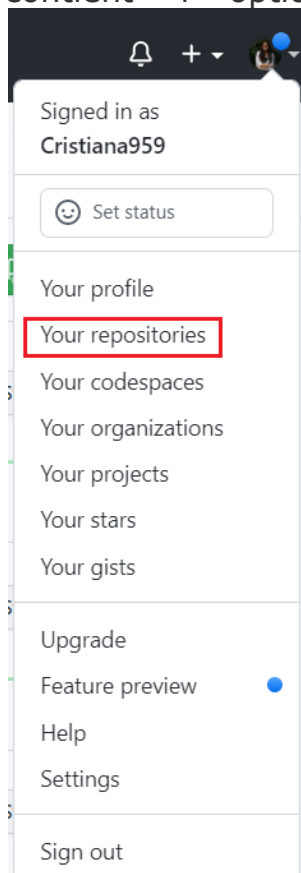
Enregistrer ces modifications signifie créer un **commit** dans le repository.

Quand on fait commit, il sera conservé dans le repository local du Git, pas dedans le repository remote du Git. Sans mise à jour le repository remote, les autres collègues ne pourront pas voir les changements que nous avons apportés. Nous voulons donc que les modifications apportées localement se retrouvent également remote. C'est-à-dire, les commits du repository local dans le repository remote. On va réaliser la publication par l'operation **push**.

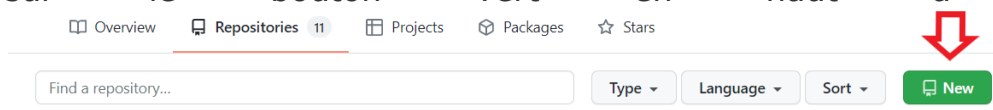
Créer un repository sur GitHub

Premièrement, vous devez connecter à GitHub.

- Ensuite, cliquez sur la flèche dans le menu en haut à droite et voyez quelque chose de similaire à l'image suivante.
- Cliquez sur *Your profile* . C'est ici que nous pouvons voir nos contributions sur GitHub. En haut de l'écran, nous verrons un menu horizontal qui contient 4 options: Overview, Repositories, Projects și Packages.



- On va appuyer sur Repositories. Nous allons maintenant voir la liste complète des dépôts que nous avons. Pour en créer un nouveau, appuyez sur le bouton vert en haut à droite New.



- Il est maintenant temps de donner un nom au projet et vous pouvez décider s'il doit être public (visible par tous les utilisateurs) ou privé (visible uniquement par nous et les collaborateurs potentiels du projet). Un formulaire similaire à celui de l'image ci-dessous apparaîtra. Pour ce tutoriel, nous allons créer un référentiel public.
- Cliquez sur *Create repository*. Le nom du repository est SDE-TP2

Repository template

Start your repository with a template repository's contents.

No template ▾

Owner *

Cristiana959 ▾

Repository name *

SDE2-TP2 ✓

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

☐ Choose a license

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

Create repository

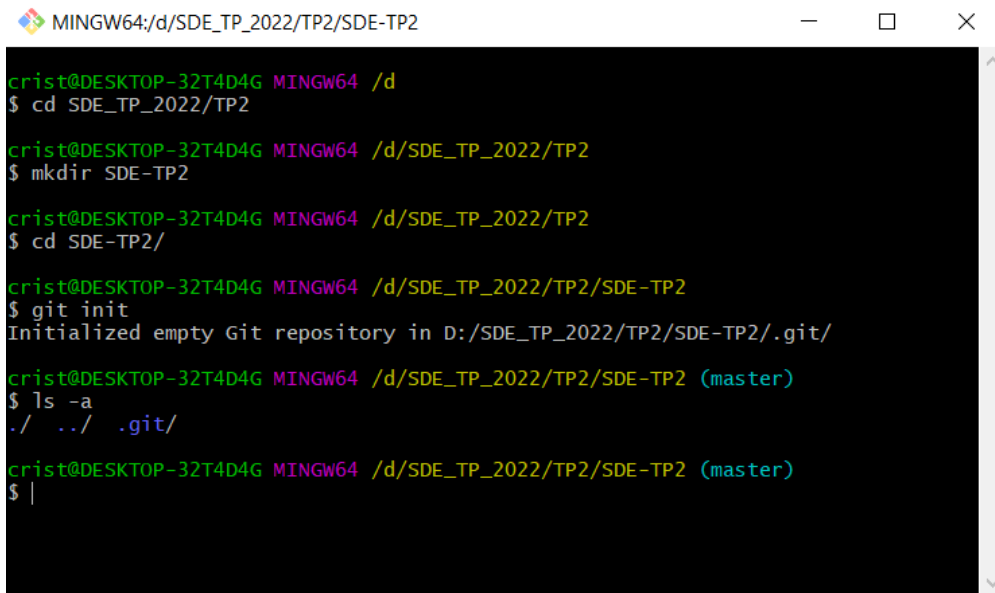


Créer un repository local

Dans le répertoire personnel, nous allons créer un répertoire dans lequel on va initialiser le repository Git.

Après avoir tapé les commandes suivantes, vous aurez une sortie similaire à celle de l'image ci-dessous.

```
user:~$ mkdir SDE2-TP2
user:~$ cd SDE2-TP2
user:~/SDE2-TP2$ git init
user:~/SDE2-TP2$ ls -a
```



```
MINGW64:/d/SDE_TP_2022/TP2/SDE-TP2
cris@DESKTOP-32T4D4G MINGW64 /d
$ cd SDE_TP_2022/TP2

cris@DESKTOP-32T4D4G MINGW64 /d/SDE_TP_2022/TP2
$ mkdir SDE-TP2

cris@DESKTOP-32T4D4G MINGW64 /d/SDE_TP_2022/TP2
$ cd SDE-TP2/

cris@DESKTOP-32T4D4G MINGW64 /d/SDE_TP_2022/TP2/SDE-TP2
$ git init
Initialized empty Git repository in D:/SDE_TP_2022/TP2/SDE-TP2/.git/

cris@DESKTOP-32T4D4G MINGW64 /d/SDE_TP_2022/TP2/SDE-TP2 (master)
$ ls -a
./ ../ .git/

cris@DESKTOP-32T4D4G MINGW64 /d/SDE_TP_2022/TP2/SDE-TP2 (master)
$ |
```

On a initialisé le repository local en utilisant la commande **git init**, à partir du répertoire créé nommé *SDE2-TP2*. En même temps, on a utilisé la commande *ls -a* pour afficher les répertoires cachés, car le répertoire **.Git** est un répertoire caché.

Init

L'opération init est locale et a le rôle d'initialiser un repository local vide. L'initialisation du repository local signifie création, dans le répertoire choisi de l'environnement pour pouvoir travailler sur un projet logiciel versionné Git. Cette opération conduit à la création d'un répertoire nommé **.git** dans lequel sont des données supplémentaires sur le référentiel, appelées metadata.

Connecter les deux repositories

Avant, on a créé un repository local et un repository distant, et pour travailler avec eux, on doit les interconnecter, en utilisant la commande:

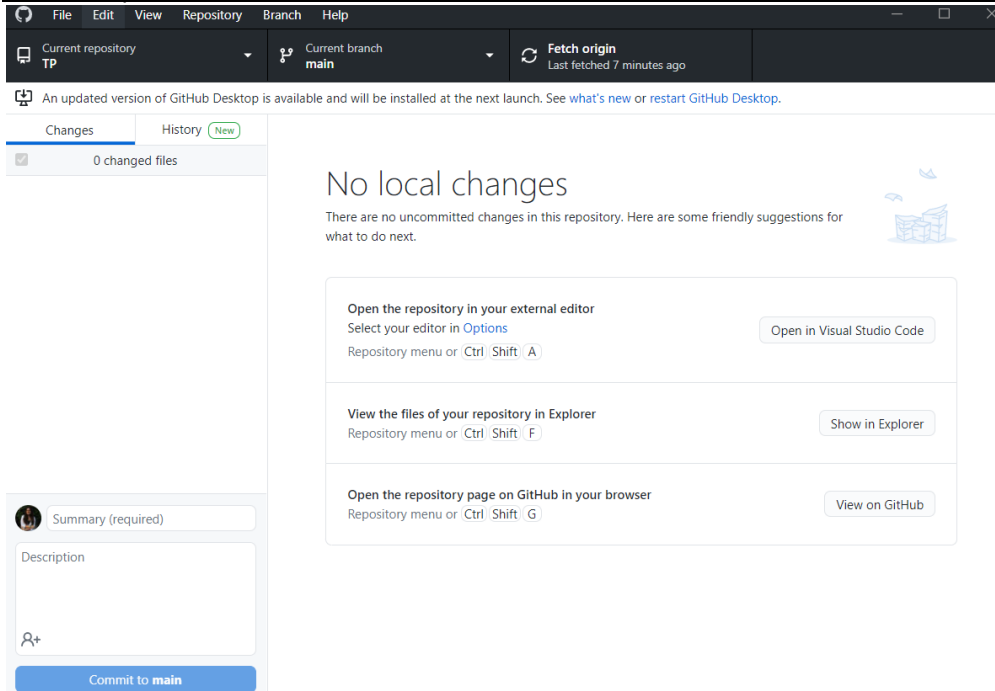
```
user:~/SDE-TP2$ git remote add origin https://github.com/{username}/SDE2-TP2.git
```

{username} est l'utilisateur sur GitHub.

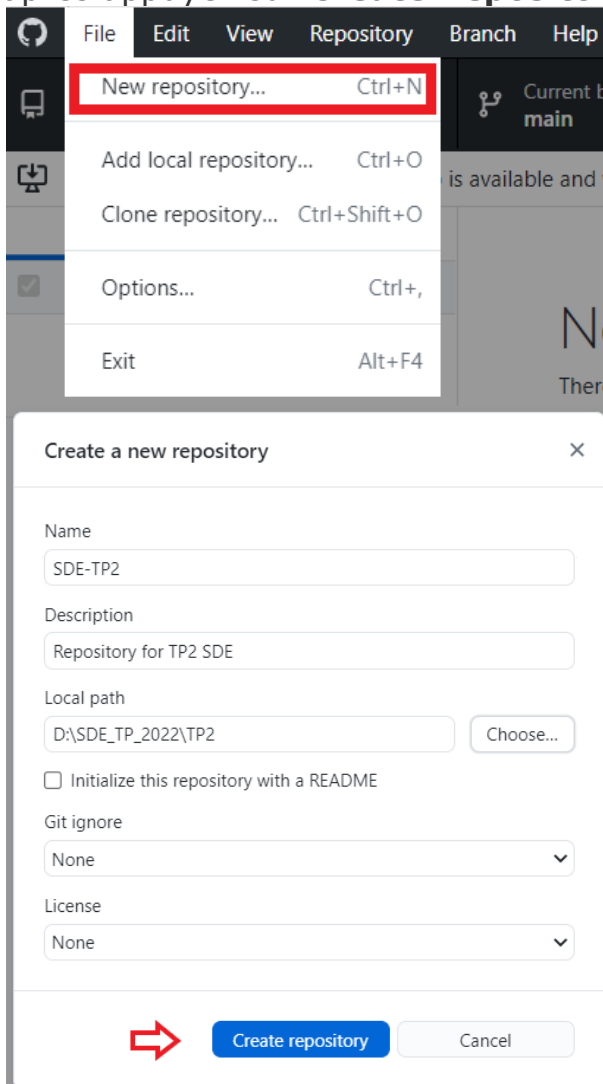
Connecter les deux repositories signifie le réglage du repository origin, c'est-à-dire le repository remote, au repository local.

L'installation de GitHub Desktop

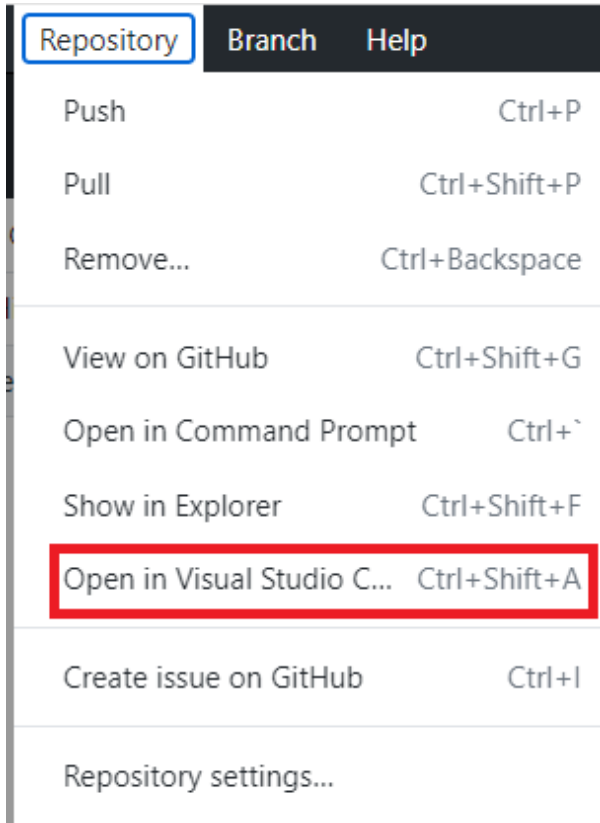
Premièrement, vous devez aller sur [GitHub Desktop](#) et téléchargez l'outil. Après, vous pouvez connecter avec votre compte GitHub.



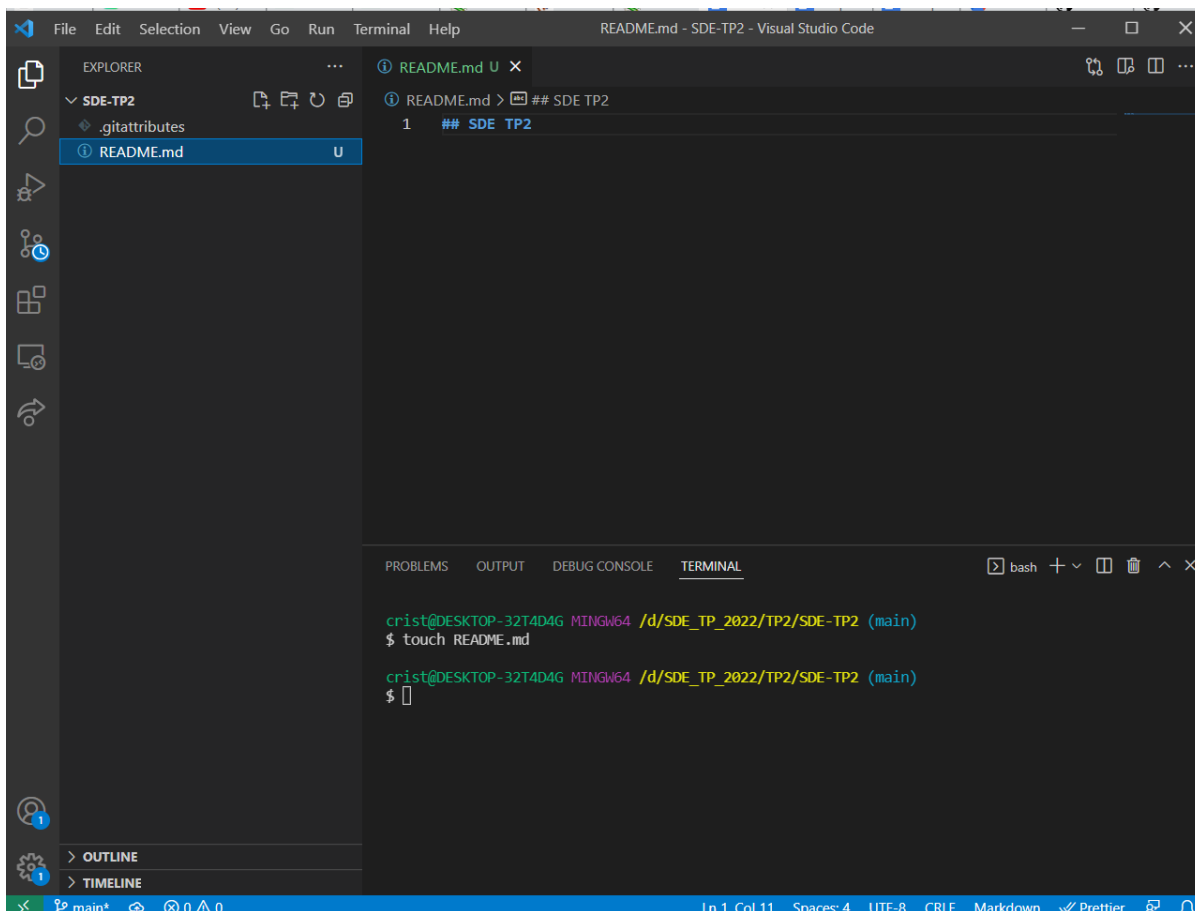
Vous pouvez créer un repository local directement dans GitHub Desktop en appuyant sur **File** → **New Repository**. Complétez les champs avec les informations nécessaires, après appuyez sur **Create Repository**.



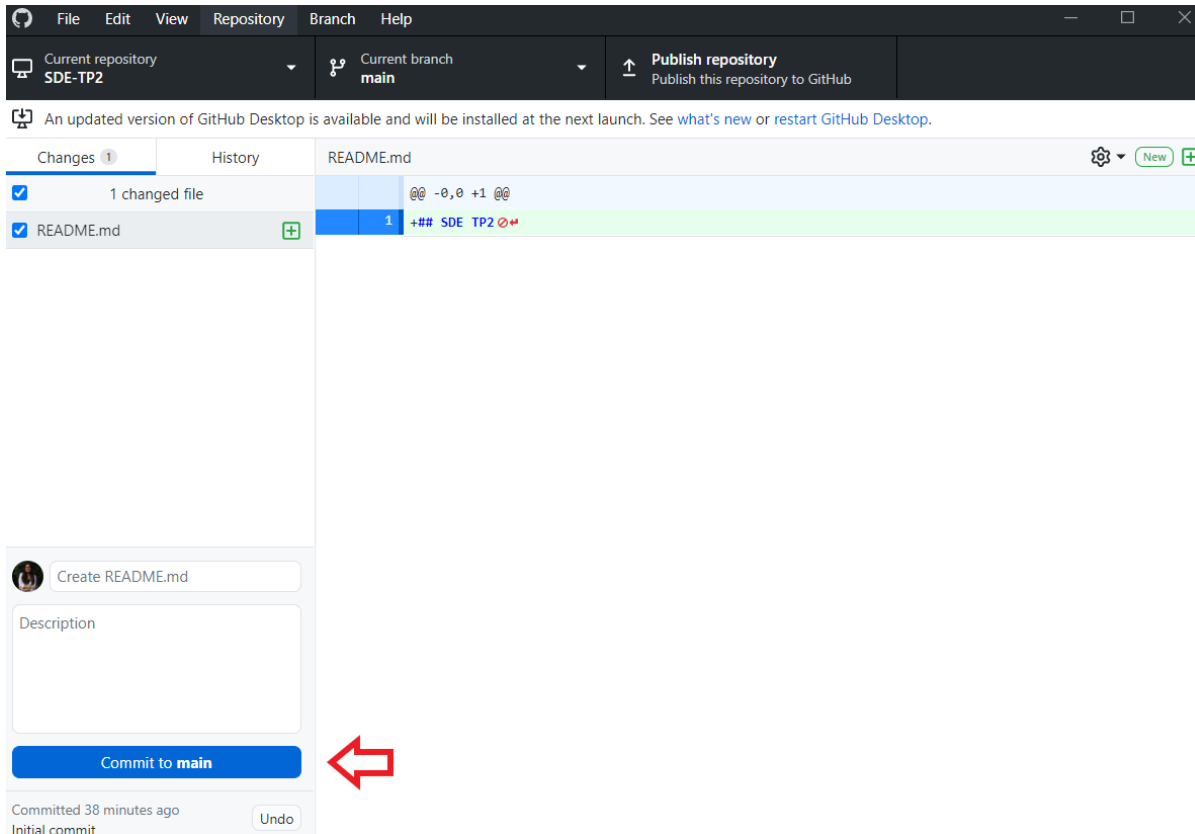
Vous pouvez ouvrir votre repository dans l'IDE de votre choix (par exemple VS Code) pour ajouter de nouvelles files ou pour changer des files. Les modifications apparaissent immédiatement dans GitHub Desktop.



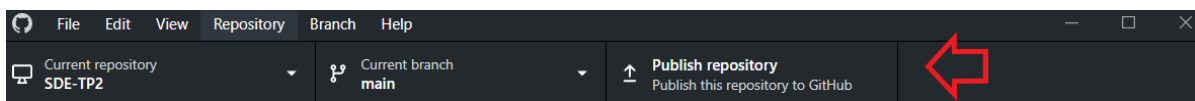
On va ajouter un fichier de README.



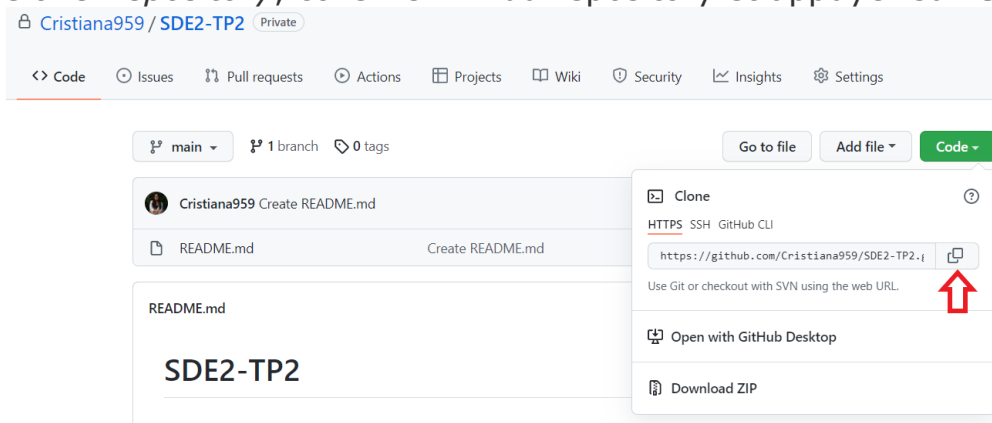
On peut voir que nos modifications ont été actualisés immédiatement dans GitHub Desktop. On ajoute un message pour notre commit et on appuie sur Commit to main pour ajouter le file à notre repository.

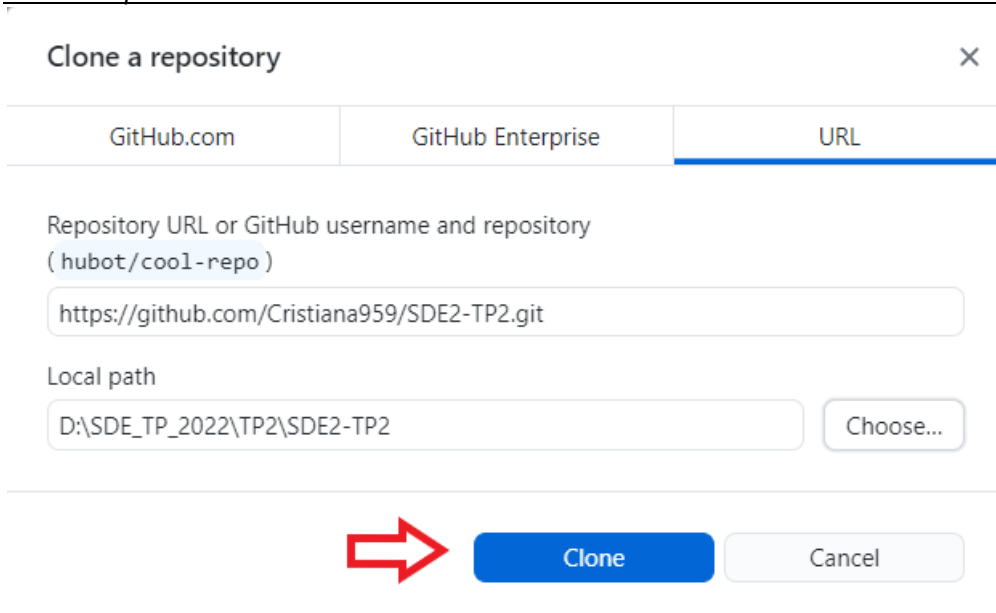


Pour publier le repo sur GitHub on fait click sur Publish repository.



Vous pouvez aussi cloner un repository existant et lui modifier en cliquant sur *File* → *Clone Repository*, coller le link du repository et appuyer sur **Clone**.





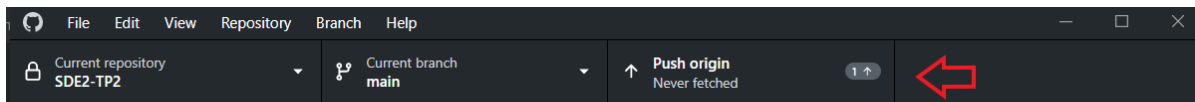
Clone a repository

GitHub.com GitHub Enterprise URL

Repository URL or GitHub username and repository
(hubot/cool-repo)

Local path

Après que vous avez fait tous les modifications, vous faisons un commit et push pour publier les modifications dans le repository remote.



Exercices

1. Créez un compte GitHub.
2. Installez Git et GitHub Desktop.
3. Créez le repository SDE-TP2 en utilisant GitHub Desktop.
4. Créez le repository SDE-TP2-2 en utilisant GitHub, fait un clone et ajoute un fichier *hello.py*. Publiez les modifications sur GitHub