1 Qu'est-ce que c'est, à quoi ça sert ? :

- GIT est un système de gestion de versions, qui permet de stocker de manière optimisée et sécurisée des fichiers et toutes leurs modifications dans le temps.
- GIT est décentralisé, et permet à de multiples personnes de travailler ensemble sur le même projet, puis d'intégrer harmonieusement les travaux de chacun.
- GitLab, c'est une alternative open-source à la référence **GitHub**, idéale pour des usages privés ou plus avancés, et qui propose beaucoup plus de fonctionnalités.

2 Fonctionnement :

GIT maintient dans votre projet des fichiers d'index qui stockent chaque modification de fichier, ainsi qu'un lien vers la version précédente.
Ainsi, à chaque "commit" (modification de fichiers), vous enregistrez les nouvelles modification dans l'index.
Et vous pouvez revenir en arrière à tout moment, ou créer des "branches" qui vous permettront de travailler en parallèle sur une nouvelle version du fichier.

3 Acceuil du Projet :

Cette page permet d'avoir toutes les informations sur le projet. Le fichier README.md est automatiquement affiché et doit permettre de comprendre les tenants et les aboutissants du projet, ainsi que les moyens pour installer le projet ou collaborer. Si le projet vous intéresse, mettez-le en favori pour le retrouver facilement sur la page d'accueil ou dans le menu"

4 Fonctionnement

Pour démarrer/travailler sur un projet, le schéma GitLab est toujours le même :

4.1 Via terminal :

0. On va d'abord **cloner** le projet (copier en local)
   `$ git clone https://gitlab.etu.umontpellier.fr/scattering-team/scattering-m1phynum.git`
1. Si cela est déjà fait, on va récupérer la dernière version en date du fichier sur lequel on travail **fetch** [1] / **pull** [2]
   `$ git fetch origin main` `$ git pull origin main`
2. Une fois qu'on à modifier nos fichiers, on les **ajoute** à la "liste" de modifications
   `$ git add fichier1.py fichier2.html`
3. On sauvegarde les modifications avec un **commit**
   `$ git commit -m 'résumé des modifications' (' ' obligatoires)`
4. On **soumet** nos modifications aux projets GitLab
   `$ git push main origin`

4.2 Via interface graphique VSCode :

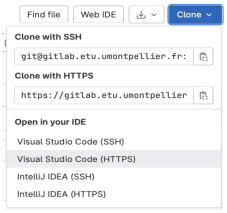

FIGURE 1 – Ouvrir dans VSCode via HTTPS

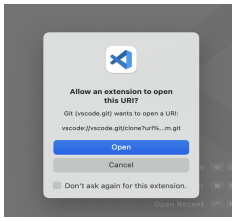

FIGURE 2 – On valide l'ouverture

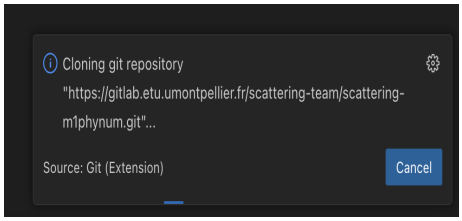

FIGURE 3 – Un prompt nous notifie du **clonage**

1. git fetch va récupérer toutes les données des commits effectués qui n'existent pas encore dans votre version, ces données ne seront pas fusionnées avec votre branche locale, vous devrez utiliser ensuite la commande git merge
2. git pull est en fait la commande qui regroupe les commandes git fetch suivie de git merge.

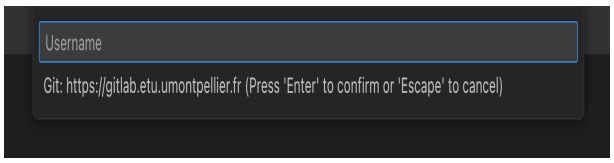

FIGURE 4 – Authentifiez-vous avec votre adresse mail umontpellier

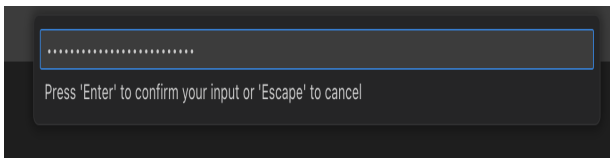

FIGURE 5 – Ensuite, utilisez votre mot de passe ENT.

FIGURE 6 – Une fois cloné, vous devez utiliser le bouton **Pull** pour obtenir la dernière version à jour

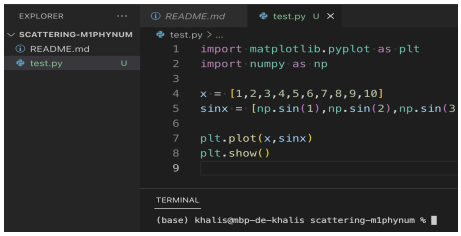

FIGURE 7 – Une fois vos fichiers créés/modifiés le vert indique une sauvegarde locale

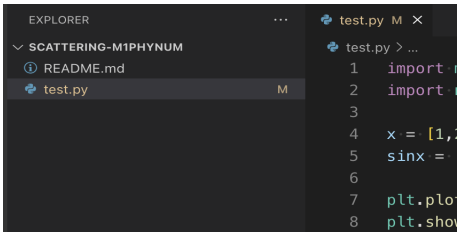

FIGURE 8 – le jaune indique qu'ils ne sont pas sauvegardés du tout !

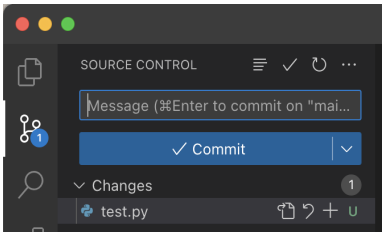

FIGURE 9 – Vous pouvez sauvegarder les modifications avec la section **commit** à gauche

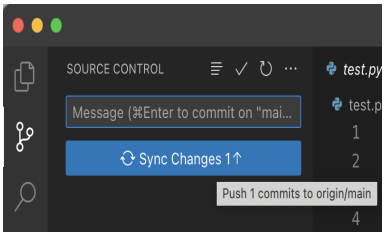

FIGURE 10 – Ensuite, soumettez le commit en utilisant le bouton 'Sync Changes' **(Push)**

Name	Last commit	Last update
README.md	production du readme.md initial	1 hour ago
test.py	validation test	just now

FIGURE 11 – Tada : Le fichier apparait bien sur le repo original

5 Demande de fusion :

Lorsque vous modifiez des fichiers sur un projet qui ne vous appartient pas, vous pouvez soumettre vos changements pour validation au propiétaire du projet.
Ceci permet de faire une revue, à l'issue de laquelle le propriétaire pourra directement intégrer vos changements dans le projet.