

Guide GitHub · Premiers pas

Compte, dépôt du binôme, Codespaces, premier commit

Adrien Chiodo

Mercredi 7 octobre 2026

À QUOI SERT GIT

Git garde l'historique d'un dossier de travail : chaque version enregistrée porte une date, un auteur et un message. On sait qui a fait quoi, et quand ; on peut revenir en arrière ; on travaille à deux sans s'écraser. Dans un projet de données, l'historique prouve aussi ce qui a été décidé avant de voir les résultats. GitHub héberge ces dossiers et leur historique.

Trois mots. Le **dépôt** est le dossier du projet, avec son historique. Le **commit** est une photo datée du dépôt, accompagnée d'un message. Le **push** envoie vos commits sur GitHub, où votre binôme et l'intervenant les retrouvent.

1. CRÉER SON COMPTE

Chacun crée le sien, sur <https://github.com>, bouton **Sign up**.

- Une adresse mail que vous consultez : GitHub y envoie un code de vérification.
- Un nom d'utilisateur sobre : il apparaîtra sur votre travail, et plus tard devant des recruteurs.
- L'offre gratuite suffit. Si GitHub vous demande d'activer la double authentification, faites-le avec une application d'authentification.

2. CRÉER LE DÉPÔT DU BINÔME

Un seul membre du binôme le fait.

1. Ouvrez <https://github.com/Ad-C/mth101-modele>.
2. Cliquez sur **Use this template**, puis **Create a new repository**.
3. *Owner* : votre compte. *Repository name* : mth101.
4. Cochez **Private**, puis **Create repository**.

3. INVITER SON BINÔME ET L'INTERVENANT

1. Dans votre dépôt : **Settings**, puis **Collaborators**, puis **Add people**.
2. Ajoutez le nom d'utilisateur de votre binôme (les deux autres membres d'un trinôme), puis Ad-C.
3. Votre binôme accepte l'invitation, reçue par mail ou sur <https://github.com/notifications>.

4. OUVRIR LE DÉPÔT DANS CODESPACES

Codespaces ouvre votre dépôt dans le navigateur, dans l'éditeur VS Code, avec Python et ses bibliothèques déjà installés. Rien à installer.

1. Sur la page du dépôt : bouton vert **Code**, onglet **Codespaces**, puis **Create codespace on main**.
2. Le premier démarrage prend quelques minutes.
3. Si VS Code demande s'il peut faire confiance aux auteurs du dossier, acceptez : c'est votre dépôt.
4. Chaque membre du binôme ouvre son propre codespace, depuis son compte.

5. FAIRE UN COMMIT ET UN PUSH

VS Code suit la langue du navigateur : libellés en français, puis en anglais.

1. Ouvrez `PREMIERE-MISSION.md`, complétez votre partie, enregistrez (`Ctrl+S`, ou `Cmd+S` sur Mac).
2. Ouvrez le panneau **Contrôle de code source** (*Source Control*), l'icône des branches dans la barre de gauche.
3. Écrivez un message qui dit ce que vous avez fait.
4. Cliquez sur **Valider** (*Commit*), puis sur **Synchroniser les modifications** (*Sync Changes*) : c'est le push.
5. Sur github.com, ouvrez **Commits** : votre commit y figure, à votre nom.

Les mêmes gestes, dans le terminal :

```
git add PREMIERE-MISSION.md
git commit -m "Première mission : présentation du binôme"
git push
```

Les bonnes habitudes. Commitez souvent, avec un message clair. Avant de partir : commit, push, puis arrêt du codespace (<https://github.com/codespaces>, **Stop codespace**) ; ce qui n'est pas poussé reste dans le codespace. Jamais de mot de passe ni de clé d'API dans le dépôt. Codespaces est gratuit dans la limite d'environ 60 heures par mois.

SI ÇA BLOQUE

Codespaces ne s'ouvre pas. Modifiez le fichier sur github.com : crayon (*Edit this file*), modifications, puis **Commit changes** ; commit et push se font en un geste. Ou touchez `.` sur la page du dépôt : l'éditeur `github.dev` committe de même, sans Python.

github.com ne répond pas. Partage de connexion d'un téléphone ; à défaut, commit dès le retour d'une connexion, avant le vendredi 9 octobre.

Python sans Codespaces. Google Colab (<https://colab.research.google.com>, compte Google) ouvre les notebooks : *File*, *Upload notebook*. Rapatriez ensuite le résultat dans le dépôt.

Push refusé (*rejected*). Votre binôme a poussé entre-temps : **Synchroniser les modifications**, ou `git pull` puis `git push`. Si la synchronisation échoue à son tour (*divergent branches*), tapez une fois, dans le terminal du codespace, `git config --global pull.rebase false`, puis synchronisez de nouveau (FAQ du cours, question 3). Si vous avez modifié la même ligne tous les deux, appelez l'intervenant.