Introduction à Git pour la Collaboration en Bioinformatique

Communauté Bioinformatique de l'UCAD : **Les Biosciens**

1 Configuration initiale de Git

À faire une seule fois pour identifier l'auteur des modifications :

```
git config --global user.name "Prnom Nom"
git config --global user.email "adresse@email.com"
```

2 Initialiser un dépôt Git (optionnel)

Si vous créez un projet local sans GitHub :

```
git init
```

Cette commande crée un dossier caché `.git` qui initialise le suivi de version.

3 Cloner un dépôt GitHub existant

Via HTTPS (recommandé pour débuter)

```
git clone https://github.com/nom-utilisateur/nom-depot.git
```

Via SSH (plus sécurisé, nécessite une clé SSH)

```
git clone git@github.com:nom-utilisateur/nom-depot.git
```

Différences :

— **HTTPS** : plus simple, mais demande votre identifiant/token à chaque `push`.
— **SSH** : plus fluide une fois configuré (clé publique/privée).

4 Commandes Git de base

— `git status` : voir les changements dans le dépôt.
— `git add fichier` : ajouter un fichier à l'index.
— `git add .` : ajouter tous les fichiers modifiés.
— `git commit -m "Message"` : enregistrer les changements localement.

5 Branches Git

— `git branch` : lister les branches existantes.
— `git checkout -b nom-branche` : créer et basculer vers une nouvelle branche.
— `git checkout main` : revenir à la branche principale.

6 Pousser et tirer des modifications

— `git push origin nom-branche` : envoyer les changements vers GitHub.
— `git pull origin main` : récupérer les changements du dépôt distant.

7 Authentification : Token HTTPS et Clé SSH

a. Cloner avec un token (HTTPS)

Depuis août 2021, GitHub exige un **token d'accès personnel** au lieu de votre mot de passe.

1. Aller sur github.com/settings/tokens :Sur ton git personnel.
2. Cliquer sur **Generate new token** (classic ou fine-grained).
3. Donner une description, choisir une durée et cocher les droits (ex. `repo`).
4. Copier le token et le garder précieusement.
5. Lors du `git clone` ou `git push`, entrez :
   — Votre `nom d'utilisateur GitHub`
   — Le **token** comme mot de passe

b. Cloner avec une clé SSH (recommandé pour les utilisateurs réguliers)

Étapes pour générer une clé SSH :

```
ssh-keygen -t ed25519 -C "votre@email.com"
```

1. Appuyez sur Entrée pour accepter le chemin par défaut.
2. Ajouter la clé au SSH agent :

```
eval "$(ssh-agent -s)"
ssh-add ~/.ssh/id_ed25519
```

3. Copier la clé publique :

```
cat ~/.ssh/id_ed25519.pub
```

4. Aller sur GitHub : **Settings** → **SSH and GPG keys**
5. Cliquer sur **New SSH key**, coller la clé et enregistrer.

Clonage avec SSH ensuite :

```
git clone git@github.com:utilisateur/depot.git
```

8 Ajouter des collaborateurs sur GitHub

1. Aller sur la page du dépôt GitHub.
2. Cliquer sur **Settings** → **Collaborators** ou **Manage access**.
3. Cliquer sur **Add people**.
4. Entrer le nom d'utilisateur GitHub.
5. Choisir un rôle :
   — **Read** : lecture seule.
   — **Write** : peut pusher/modifier le code.
   — **Admin** : peut modifier le dépôt et ajouter des personnes.
6. Le collaborateur doit accepter l'invitation.

9 Conseils de collaboration

— Créez toujours une branche pour vos modifications.
— Commitez souvent avec des messages clairs.
— Utilisez des **pull requests** pour intégrer vos changements.
— Évitez de travailler directement sur `main`.

10 Utilisation de plusieurs branches et fin de projet

Travail avec plusieurs branches

Travailler avec plusieurs branches permet de séparer les fonctionnalités, les corrections de bugs ou les expérimentations du code principal. Voici comment organiser le travail :

— Créez une branche pour chaque fonctionnalité ou tâche :

```
git checkout -b nouvelle-fonction
```

— Faites vos modifications, puis committez :

```
git add .
git commit -m "Ajout de la nouvelle fonctionnalit"
```

— Poussez votre branche vers GitHub :

```
git push origin nouvelle-fonction
```

— Créez une **pull request** (PR) sur GitHub pour proposer l'intégration dans `main`.

Fin de projet : fusion et nettoyage

Une fois que toutes les fonctionnalités ont été validées :

1. Revenez sur la branche principale :

```
git checkout main
git pull origin main
```

2. Fusionnez les branches une à une :

```
git merge nom-branche
```

3. En cas de conflit, Git vous indique les fichiers concernés. Ouvrez-les, corrigez-les, puis :

```
git add fichier-conflit
git commit
```

4. Supprimez les branches fusionnées (localement et à distance) :

```
git branch -d nom-branche % local
git push origin --delete nom-branche % distant
```

5. Enfin, taguez la version stable si besoin :

```
git tag -a v1.0 -m "Version finale"
git push origin v1.0
```

Bonnes pratiques :

— Ne fusionnez pas dans `main` sans revue de code via PR.
— Nettoyez les branches obsolètes pour garder un dépôt propre.

11 Ressources utiles

— Guide Git simplifié en français
— Documentation Git officielle
— Tutoriel GitHub

Contact : Albert NDOUR – Ing. Bioinformatique
`albertndour96@gmail.com`