1 Prise en main de Git

1.1 Création d'un compte sur GitHub

GitHub est un service web d'hébergement de dépôts distants utilisant Git. Il offre des fonctionnalités supplémentaires destinées à la collaboration, telles que le suivi des bugs, les demandes d'ajout de fonctionnalités ou encore la gestion de tâches.

À noter qu'il existe d'autres plateformes d'hébergement basées sur Git, comme **GitLab** ou **BitBucket**.

Rendez-vous sur `https://github.com/` et suivez les instructions pour créer et activer votre compte. Dans un terminal, spécifiez l'adresse e-mail que vous utiliserez pour vos commits :

```
git config - -global user.email "votre_email@example.com"
```

1.2 Création d'un dépôt

Nous allons maintenant créer un dépôt distant. Sur GitHub :
— Cliquez sur le "+" en haut à droite de la page, puis sur **New repository**.
— Donnez un nom à votre dépôt, par exemple `PremierDepot`, ainsi qu'une description.
— Par défaut, votre dépôt est **public**, c'est-à-dire que tout le monde peut consulter votre code.
— Terminez en cliquant sur **Create repository**.
Ensuite, suivez les instructions fournies par GitHub pour créer votre copie locale du dépôt. Dans l'ordre :

1. `git init` : initialise un dépôt local vide dans votre répertoire courant ;
2. `git add` : indexe un fichier ou un répertoire ;
3. `git commit` : valide les modifications sur votre dépôt local ;
4. `git remote add <nom> <url>` : lie votre dépôt local à un dépôt distant (URL sur GitHub) ;
5. `git push` : envoie les modifications indexées vers le dépôt distant.

1.3 Premiers commits

Vous souhaitez mettre sous gestion de version l'un des travaux pratiques précédents.

1. Copiez l'ensemble du répertoire, y compris les exécutables du TP4, dans le répertoire où le dépôt Git a été initialisé.
2. Utilisez les commandes `git status` et `git diff`. Que constatez-vous ?
3. Indexez tous les fichiers (y compris les exécutables), puis vérifiez l'indexation avec `git status`.
4. Poussez les modifications vers votre dépôt distant.

Ces étapes constituent le minimum pour sauvegarder votre travail sur le serveur distant. Il est important d'exécuter `git status` à chaque étape : un fichier non souhaité peut être ajouté par erreur.

1.4 Suppression de fichiers dans le dépôt distant

Nous avons mis l'ensemble du projet sous Git, ce qui constitue une mauvaise pratique. Les exécutables et les fichiers issus de la compilation (`.o`) ne doivent pas être suivis par Git.
Vous pouvez supprimer des fichiers déjà indexés à l'aide de :

```
git rm --cached nom_du_fichier
```

Attention : N'oubliez pas l'option `-cached`, sinon le fichier sera également supprimé de votre dépôt local !

Supprimez les exécutables de votre dépôt distant, puis validez et poussez vos modifications. N'oubliez pas d'utiliser `git status` pour vérifier les changements à chaque étape.

Une autre approche consiste à créer, à la racine de votre dépôt local, un fichier nommé `.gitignore`. Ce fichier contient la liste des fichiers à exclure du suivi Git.
À l'aide d'expressions régulières, créez un fichier `.gitignore` qui ignore :

— les fichiers avec l'extension `.o` ; les fichiers temporaires ; l'exécutable du dépôt.
Validez et poussez vos modifications. Expliquez en quoi cette approche simplifie l'ajout de nouveaux fichiers.

2 Collaboration entre différents utilisateurs

Le but de cet exercice est de vous initier à l'utilisation de Git comme **outil de collaboration**.

2.1 Manipulation d'un dépôt existant

Récupérez le dépôt à l'adresse suivante : `https://github.com/fgrelard/GLGit`

1. Sur GitHub, cliquez sur l'icône du **Fork**.
2. Une copie du dépôt est ajoutée à votre espace GitHub. Clonez-la pour obtenir un dépôt local.

Il s'agit d'un programme en **C** dédié à la manipulation de voies métaboliques et d'enzymes. L'utilisateur peut spécifier leur nom depuis la ligne de commande ou à partir d'un fichier.

2.2 Processus de résolution de bugs

Un bug s'est glissé dans le programme après un commit. La description du bug est disponible dans la section **Issues** du dépôt distant de `fgrelard`.

2.2.1 Identification du commit fautif

Votre objectif est d'identifier le commit ayant introduit le bug.
Utilisez les commandes `git log`, `git diff`, `git checkout` et `gitk` pour repérer le commit responsable.

2.2.2 Création d'une branche

Lorsqu'il s'agit de corriger un bug complexe ou d'ajouter une fonctionnalité, il est préférable de créer une **branche dédiée**. Cela permet de maintenir une version stable séparée d'une version en développement.

1. Créez une branche locale nommée `fix_suppression` :

```
git branch fix_suppression
```

2. Poussez cette branche sur le dépôt distant :

```
git push -u origin fix_suppression
```

3. Déplacez-vous sur la nouvelle branche :

```
  git checkout fix_suppression
```

   Les fichiers modifiés sur cette branche ne seront pas affectés sur `master`.

2.2.3 Résolution du bug

1. Modifiez le fichier concerné par le bug en le réinitialisant à une version précédente à l'aide de la variable `HEAD`.
2. Validez (`git commit`) puis poussez (`git push`) vos modifications sur la branche `fix_suppression`.

2.2.4 Validation et suppression de la branche

1. Vérifiez que la correction du bug n'a pas introduit d'autres erreurs en testant le programme.
2. Fusionnez ensuite la branche `fix_suppression` avec `master`, puis supprimez-la :

```
git checkout master
git merge fix_suppression
git branch -d fix_suppression
git push origin --delete fix_suppression
```