TP Git

Travaux Pratiques

1 Prise en main de git

1.1 Création de compte sur GitHub

GitHub est un service Web d'hébergement de dépôts distants, utilisant git. GitHub comporte des fonctionnalités supplémentaires destinées à la collaboration, telles que le suivi des bugs, les demandes d'ajout de fonctionnalités, ou la gestion de tâches...À noter qu'il existe d'autres sites Web d'hébergement basés sur git, tels que GitLab, ou BitBucket.

Allez sur `https://github.com/` et suivez les instructions pour la création et l'activation de votre compte. Sur un terminal, spécifiez l'adresse mail avec laquelle vous allez effectuer 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 "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 a accès à votre code (droits de lecture). Terminez en cliquant sur "Create repository".

Suivez les instructions fournies par GitHub pour créer votre copie locale du dépôt. Dans l'ordre :

1. `git init` permet d'initialiser un dépôt local vide dans votre répertoire courant ;
2. `git add` permet d'indexer un fichier ou un répertoire ;
3. `git commit` permet de valider les modifications sur votre dépôt local ;
4. `git remote add <nom> <url>` : permet de lier votre dépôt local (initié par init) à un dépôt distant (`url` sur GitHub) ;
5. `git push` permet d'envoyer les modifications indexées au dépôt distant.

1.3 Premiers commits

Vous voulez mettre sous version l'exercice 2 du TP1.

1. Copiez l'ensemble du répertoire, y compris les exécutables du TP1, dans votre répertoire où le dépôt git a été initialisé.
2. Utilisez les commandes `git status` et `git diff` en vous aidant de la documentation au besoin. Que constatez-vous ?
3. Indexez l'ensemble des fichiers (y compris les exécutables), puis vérifiez l'indexation avec `git status`.
4. Poussez les modifications sur votre dépôt distant.

Ces étapes sont un minimum pour sauvegarder votre travail sur le serveur distant. Il est important de ne pas oublier de faire un `git status` à chaque étape : un fichier que vous ne désiriez pas ajouter peut être ajouté par mégarde.

1.4 Supprimer des fichiers dans le dépôt distant

Nous avons mis l'ensemble du projet sous git. Il s'agit d'une mauvaise pratique : les exécutables et les fichiers résultant de la compilation (`.o`) ne doivent pas être mis sous version. Il est alors possible de supprimer des fichiers indexés en utilisant :

```
git rm --cached fichier
```

Attention : Ne pas oublier l'option `--cached`, sinon votre fichier sera aussi supprimé de votre dépôt local !

Question Supprimez le/les exécutables de votre dépôt distant, puis commitez et poussez vers votre dépôt distant. N'oubliez pas `git status` pour vérifier les modifications apportées, à chaque étape.

Une autre option, est de définir, à la racine de votre dépôt local, un fichier nommé `.gitignore`. Ce fichier contient l'ensemble des fichiers qui ne doivent pas être ajoutés à votre dépôt distant.

Question En utilisant des expressions régulières, définissez un fichier `.gitignore` qui permette d'ignorer les fichiers avec l'extension `.o`, les fichiers temporaires, ainsi que l'exécutable dans le dépôt. Commitez et poussez vos modifications. Comment cela simplifie-t-il l'ajout de nouveaux fichiers à votre projet ?

2 Collaboration entre différents utilisateurs

Le but de cet exercice est de vous initier à l'utilisation de git en tant qu'outil de collaboration.

2.1 Manipulation d'un dépôt existant

Récupérez le dépôt à l'adresse `https://github.com/fgrelard/GLGit`.
Forkez le dépôt en suivant ces étapes :

1. Sur GitHub, cliquez sur l'icône du fork.
2. Une copie est ajoutée sur votre espace GitHub. Clonez-la afin d'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 à partir de la ligne de commande ou 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 donnée dans la partie "Issues" du dépôt distant de `fgrelard`.

2.2.1 Identification du commit coupable

Votre but est d'identifier le commit ayant induit le bug.
Question Utilisez `log`, `diff`, `checkout` et `gitk` afin d'identifier le commit responsable de l'apparition du bug.

2.2.2 Création d'une branche

Afin de résoudre un bug complexe ou d'ajouter des fonctionnalités, il est souvent nécessaire de modifier plusieurs parties du code. On crée donc une branche, où l'on fait tous les commits dédiés à la résolution du bug. L'idée est de maintenir une version stable, dans la branche `master`, séparée de la version en développement, qui peut contenir des bugs.

Questions En vous aidant de la documentation :

1. Créer une branche locale nommée `fix_suppression` avec la commande `git branch`.
2. Poussez la branche locale sur le dépôt distant avec la commande `git push`.
3. Déplacez-vous sur la branche locale avec `git checkout`. Les fichiers modifiés sur cette branche ne seront pas modifiés sur `master`.

2.2.3 Résolution du bug

On peut ensuite résoudre le bug.

1. Modifiez le fichier concerné par le bug en le réinitialisant à une de ses versions précédentes, en utilisant la varible `HEAD`.
2. Commitez, puis poussez vos modifications **sur la branche** `fix_suppression`.

2.2.4 Validation et suppression de la branche

1. Assurez-vous que la résolution du bug n'a pas induit d'autres bugs dans l'exécution du programme, en le testant.
2. Vous pouvez ensuite fusionner la branche `fix_suppression` dans `master` et la supprimer :
   (a) déplacez-vous sur la branche `master`
   (b) fusionnez la branche `fix_suppression` dans master avec `git merge`
   (c) supprimez la branche locale avec `git branch -d`
   (d) supprimez la branche distante avec `git push -d`

2.2.5 Pull requests

Votre correction du bug peut intéresser l'auteur original du projet, `fgrelard`. Sur GitHub, ouvrez une pull-request. Les pull-request sont un ensemble de commits qui peuvent être intégrés directement par l'auteur du projet dans son dépôt distant, et sont ainsi un outil puissant pour travailler à plusieurs.

3 Fusion et conflits

Le but de cet exercice est de manipuler les fusions (merge) de commits. Une fusion peut être conflictuelle lorsque deux utilisateurs travaillent en parallèle sur les mêmes lignes du même fichier.

3.1 Importation du projet

1. Utiliser l'outil d'import de GitHub disponible à l'adresse : `https://github.com/new/import`. Cet outil permet de faire l'importation d'un projet existant. Entrez l'URL suivante dans "Your old's repository URL" : `https://github.com/githubschool/on-demand-merge-conflict.git`. Une copie du dépôt doit être faite sur faite sur votre espace GitHub, de façon similaire à un fork
2. Clonez ce dépôt en local.

Ce dépôt contient du code permettant d'afficher un CV en ligne.

3.2 Première fusion

Vous voulez changer le titre et la description de votre CV.

1. Créez une branche nommée `mytitle`
2. Modifiez les lignes 2 et 3 du fichier `_config.yml` en spécifiant le titre et la description voulue.
3. Commitez et poussez vers le dépôt distant, en spécifiant le nom de la branche courante.
4. Déplacez-vous sur la branche `master`
5. Fusionnez (merge) les modifications avec la commande `git merge`, puis commitez et poussez.

La branche `master` contient maintenant les modifications effectuées sur la branche `mytitle`. Supprimez la branche `mytitle` sur le dépôt local et distant (cf. exercice précédent).

3.3 Fusion conflictuelle

Vous voulez spécifier votre nom et vos informations personnelles.

1. Déplacez-vous sur la branche `username-config`
2. Sur cette branche, ouvrez le fichier `_config.yml`
3. Les lignes 12 à 19 doivent être modifiées avec vos informations personnelles. Sauvegardez vos modifications.
4. Commitez puis poussez vos modifications.
5. Déplacez-vous sur la branche `master` et fusionnez la branche `username-config` à l'aide de `merge`.

Un conflit apparaît, puisque l'utilisateur Bob Belcher a modifié les mêmes lignes que vous sur la branche `master`. Ouvrez le fichier `_config.yml` à nouveau, le fichier doit se présenter de la façon suivante :

```
<<<<<<< commit Bob
# version de Bob
=======
# ma version
>>>>>>>
```

Modifiez le fichier pour ne garder que votre version : à ce stade, le conflit est considéré comme résolu. Enfin, commitez puis poussez vers votre dépôt distant.

4 Pour aller plus loin

Des exercices complémentaires sur la manipulation des branches, très clairs et bien conçus : `http://learngitbranching.js.org/`.