Série de Travaux Dirigés (TD) sur Git
Gestion des branches et développement collaboratif avec Git

1 Développement avec des branches

La documentation officielle est disponible en local (git help <nom-option>) et en ligne (http ://git-scm.com/docs).
À partir d'une version donnée du code, on souhaite pouvoir poursuivre le développement principal, tout en s'autorisant à partir sur des développements annexes sur une période plus courte. Il peut typiquement s'agir d'une correction d'un bug majeur nécessitant plusieurs étapes et affectant plusieurs fichiers.

On considère le fichier nommé `lignes.txt` suivant :

```
igne01
igne02
ligne03
```

On constate qu'il y a une erreur (un bug) sur les deux premières lignes : il manque la lettre `l` devant `igne01` et `igne02`. Pour corriger cette erreur, nous allons travailler sur une branche secondaire afin de ne pas bloquer les développements sur la branche principale.

Exercice 2

1. Créer le dépôt, configurer-le avec votre nom et votre email, intégrer le fichier `lignes.txt` avec les 3 lignes indiquées et faire un premier commit avec comme commentaire `v0`.

2. Créer une branche `correction`, et afficher les branches disponibles sur le dépôt ainsi que votre état.

3. Sur quelle branche êtes-vous ?

4. Placez-vous sur la branche `correction`, et réafficher les branches disponibles. Quelles différences observez-vous ?

5. Dans quel état sont les fichiers ?

6. Vous êtes toujours sur la branche `correction` : éditez le fichier afin d'ajouter le `l` manquant sur la première ligne seulement. Commitez cette modification avec le message `lignes.txt : ligne01 corrigée`.

7. On suppose que le développement se poursuit sur la branche principale (`master`) : placez-vous sur cette branche. Consultez le contenu du fichier. La correction apportée ne doit pas apparaître car celle-ci a été faite sur la branche `correction`.

8. Ajoutez une nouvelle ligne au fichier (`ligne04`) et commitez cet ajout.

9. Revenez sur la branche `correction`, corrigez la `ligne02` du fichier texte et commitez la modification. Quel est le contenu du fichier texte ?

10. Nous allons intégrer les corrections apportées sur la branche `correction` à la branche principale :

    - positionnez-vous sur la branche `master` ;
    - affichez le contenu du fichier ;
    - intégrez les modifications de la branche `correction` avec la commande :

      ```
      /projet$ git merge correction -m "fusion".
      ```

11. Affichez l'état du fichier après modification : les deux premières lignes sont corrigées, et la `ligne04` doit apparaître (ajoutée en parallèle du correctif sur la branche `correction`).

12. Affichez graphiquement toutes les évolutions avec une interface graphique.

2 Développement collaboratif

Afin de se familiariser avec le développement collaboratif, nous allons considérer un projet avec trois dépôts (dans trois répertoires distincts) : deux dépôts locaux (`repo1` et `repo2` associés aux utilisateurs `nom1` et `nom2`) et un dépôt distant `remote`. Cette situation est illustrée par la figure 1.

Exercice 3

1. (a) Créez un répertoire `repo1`, allez dans ce répertoire (`cd repo1`), puis initialisez un dépôt Git, et configurez-le avec le nom `nom1` et l'email `nom1@email.com` (avec l'option `local`).

   (b) Créez un fichier `file.txt` avec une seule ligne `ligne01`. Ajoutez puis historisez ce fichier dans un commit commenté avec `first commit`.

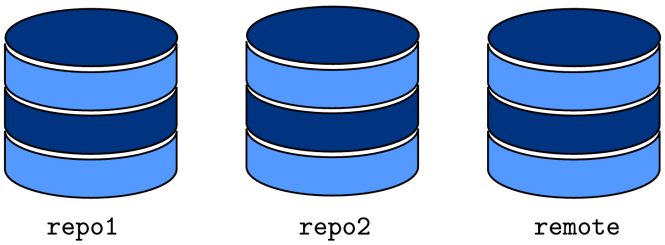

Figure 1: Développement collaboratif avec 3 dépôts.

(c) Affichez les logs (`git log`).

2. (a) Remontez dans l'arborescence.
   (b) Créez un répertoire `remote` puis allez dans ce répertoire. Initialisez le dépôt `remote` par clonage de `repo1` avec la commande suivante. Avant de taper cette commande, répondez à ces deux questions :
   - Que désigne le dernier point dans la commande précédente ?
   - Quel est l'objectif de l'option `-bare` de la commande `git clone` ?

   ```
   /remote$ git clone --bare chemin_vers_repo1 .
   ```

   (c) Affichez ensuite les logs de ce dépôt.

3. (a) Remontez dans l'arborescence.
   (b) Créez un répertoire `repo2` puis allez dans ce répertoire.
   (c) Initialisez le dépôt `repo2` par clonage du dépôt `remote`.
   (d) Configurez ensuite ce dépôt avec le nom `nom2` et l'email `nom2@email.com` (avec l'option `local`).
   (e) Affichez ensuite les logs.

4. (a) Depuis le répertoire `repo2` : affichez le contenu du fichier.
   (b) Ajoutez ensuite une ligne `ligne02` en fin de fichier : ajoutez puis commitez cette modification.
   (c) Affichez les logs. Quelles sont les versions présentes ? Qui les a réalisées ?

5. Mise à jour du dépôt distant (`remote`) depuis le dépôt courant `repo2` :
   (a) Restez dans le dépôt `repo2`.
   (b) Poussez (`push`) la modification précédente vers le dépôt `remote`.

6. Vérification que la mise à jour a bien été intégrée au dépôt `remote` :

(a) Placez-vous dans le répertoire `remote`.

(b) Consultez les logs.

(c) Pour quelle raison aucun fichier ne se trouve dans le répertoire de travail `remote` ?