git et les branches
Simuler le travail en groupe, mais seul(e)

Mot d'introduction

L'objectif de cette feuille d'exercices est de se préparer au travail à plusieurs sur le même dépôt en mimant le travail en groupe sur plusieurs branches. Ce travail individuel sert à bien comprendre comment le contenu d'un dépôt fonctionne sur plusieurs branches et comment communiquer avec les différentes branches.

Attention : cette feuille d'exercices TD crée volontairement une situation de conflit de merge, il est donc prévu d'avoir un "problème" à un moment. Lisez bien ce que vous dit git car il vous donne des informations.

Pré-requis

— Avoir un compte Gitlab avec une clé publique bien configurée
— Pour indiquer qu'une branche appelée <name> que vous avez créée localement doit être stockée sur un dépôt distant, votre *premier* push sur cette branche sera la commande `git push --set-upstream origin <name>`, qui indique que la branche <name> doit être stockée sur le dépôt distant. Pour push les tags, utilisez la commande `git push origin --tags`.
— installer un outil qui vous permettra de visualiser des différences entre fichiers : Visual Studio Code fait l'affaire, mais `meld` est préféré pour sa légereté malgré ses limites. Il s'installe avec la commande `sudo apt install meld` sur Ubuntu.

Exercice 1 Mise en place des données initiales

Pour débuter votre périple, vous commencerez par récupérer une copie d'un dépôt qu'on vous donnera :

1. Allez sur `https://gitlab.com/plurital1/PPE2-2425-TP1`
2. Créez un **fork** de ce dépôt (bouton en haut à droite) : un **fork** est une copie du dépôt de quelqu'un d'autre sur votre compte. Cela vous permet d'y faire des modifications (c'est un peu comme une branche, mais à l'échelle de l'utilisateur(ice))
3. Une fois le fork fait, clonez-le sur votre machine. Si vous forkez `https://gitlab.com/plurital1/PPE2-2425-TP1` et que votre nom d'utilisateur est `USER`, le dépôt que vous clonez aura pour adresse `https://gitlab.com/USER/PPE2-2425-TP1`
4. sur ce dépôt, vous devriez trouver deux fichiers : un `readme.md` et un fichier `dormeur.txt` ;
5. jetez un œil au log afin de voir ce qui a été fait sur le dépôt avant de le forker

Exercice 2 Créer deux branches avec des modifications différentes

Pour cet exercice, on travaillera sur le fichier `dormeur.txt`. Gardez-le ouvert sur un éditeur et regardez-le régulièrement pour voir son contenu changer au fil de l'exercice.

Dans cet exercice, il faudra créer deux branches à partir de la branche `main` et modifier `dormeur.txt` dans chacune d'entre elles :

1. Créez la branche `casse` depuis `main` et basculez dessus.
   — modifiez `dormeur.txt` de façon à remettre les majuscules à chaque première lettre de chaque vers du poème ;
   — une fois cette modification faite, committez et pushez (sur le dépôt en ligne) le travail fait sur la branche `casse`.
2. Créez la branche `numerotation` (sans accents de préférence) depuis `main` et basculez dessus.
   — modifiez `dormeur.txt` de façon à mettre début du premier (et uniquement du premier) vers de chaque strophe le numéro de la strophe. Par exemple, il faudra indiquer « 1 : » à la première strophe, « 2 : » à la deuxième, etc ;
   — une fois cette modification faite, committez et pushez (sur le dépôt en ligne) le travail fait sur la branche `numerotation`.

Exercice 3 Marcher dans le merge, ça porte bonheur !

Il est grand temps de mettre en commun les travaux effectués sur chacune des branches !

1. restez (ou revenez) sur la branche `numerotation` ;
2. lancez un merge des modifications de `casse` ;
3. aïe ! Vous avez un conflit de merge ! Utilisez un outil de comparaison de textes pour le régler au mieu. Typiquement, ces outils vous afficheront trois colonnes : celle du milieu est l'état du fichier avant les modifications faite dans chacune des branches (avant la divergence), et les deux autres seront la version locale et celle qu'on cherche à merger. [1]
4. utilisez l'outil en question pour intégrer les modifications faites des deux côtés (possibilité de travail manuel non nulle) ;
5. une fois que vous avez un fichier qui ressemble à ce que vous voulez, sauvegardez ;
6. vérifiez le statut de votre dépôt : un commit tout prêt devrait vous attendre (à quoi peut-on le remarquer ?), pushez-le ;
7. retournez sur la branche `main` et mergez cette fois-ci les modifications présentes sur la branche `numerotation`. Que se passe-t-il ? Faites le nécessaire pour pusher les modifications.

Exercice 4 Et maintenant, on documente le tout sur un coin propre.

La dernière étape va être de documenter ce que vous avez fait sur une branche spécifique, appelée `doc` :

1. on veut retourner dans le passé, à une version où `dormeur.txt` n'existe pas, comment faire ? Si jamais cela vous bloque, restez sur `main` vous pouvez aussi supprimer le fichier `dormeur.txt` (mais seulement si vous bloquez !).
2. Git vous affiche en toute logique un avertissement, quel est-il ? Cherchez un peu de documentation.
3. Créez et basculez sur la branche `doc`
4. Créez à présent le fichier `journal.md` et remplissez-le avec le travail que vous avez fait. Pushez, c'est fini côté git !

Exercice 5 Rendre le travail

Une fois le travail terminé, envoyez sur icampus, dans une archive (ex : .zip) :

— Les screenshots de vos solutions LearnGitBranching (pensez à faire `echo "<votrenom>"` au début !)
— Créer le tag "seance1" sur la branche main à la fin du travail
— Le lien vers votre dépôt git après le fork

1. Les visualisations peuvent varier !