Git 2 : gérer le travail à plusieurs

Ces exercices vont vous guider pour faire une gestion très basique des erreurs qu'on peut voir dans git, notamment en travaillant à plusieurs dans le même dépôt. Ils servent également à montrer l'intérêt du *workflow* de git pour (r)attraper des erreurs. Les exercices sont scénarisés pour créer des erreurs et les résoudre par la suite.

Les exercices ici suivent une scénarisation un peu précise. Il est demandé de ne pousser (*push*) que lorsque c'est indiqué dans la feuille, pour le bon fonctionnement des exercices.

Exercice 1 Avant de commencer

Cette feuille d'exercices est prévue pour être faite sur un dépôt bidon. Il est prévu de faire les exercices de manière guidée. Les différentes (nouvelles) commandes illustrées seront :

- `git checkout`
- `git reset`
- `git stash`

Exercice 2 Créer un README avec un peu de contenu

On va se placer dans un dépôt "propre", c'est-à-dire sans modification en cours. Nous travaillons dans un dépôt de test, les modifications ne sont donc pas spécialement importantes. Faites un checkout du dossier courant pour annuler toutes les modifications depuis le dernier commit.

Solution :

```
git checkout .
```

Allez sur `https://www.lipsum.com` et créez quelques paragraphes de *lorem ipsum*. Copiez-collez ces paragraphes à la fin de votre `README.md`. Poussez vos changements.

Solution :

```
git add README.md
git commit -m "modif readme."
git push
```

Exercice 3 Créer un conflit

Exercice 3.a Simuler le commit d'une autre personne

Après avoir poussé vos changements, modifiez le fichier `REAMDE.md` via l'interface web : supprimez le premier paragraphe de *lorem ipsum*.

À ce moment, vous avez une désynchronisation en votre dépôt sur votre machine et le dépôt en ligne : vous êtes en retard de 1 *commit*. Cela simule le cas réel de quelqu'un qui a fait une modification pendant que vous étiez en train de travailler.

Pour le moment, on va laisser cette modification tranquille, elle va revenir rapidement.

Exercice 3.b Faire des modification en conflit

Ajoutez une ligne à la fin de votre readme avec la commande `echo`. Tentez de pousser vos changements. Si vous avez bien fait, vous aurez un message d'erreur vous indiquant que vous ne pouvez pas pousser.

Solution :

```
echo -e "\nUne nouvelle ligne à la fin du readme." » README.md
git add README.md
git commit -m "modif readme"
git push # va donner une erreur
```

Exercice 4 Débloquer la situation

Exercice 4.a Premier temps : annuler le commit

Tant que le dossier local aura au moins un `commit` d'avance, il sera impossible de récupérer les modifications du dépôt en ligne. Dans un premier temps, on va donc annuler la mise-en-place, tout en gardant les changements.

Récupérez les métadonnées du dépôt en ligne puis vérifiez le status de votre dépôt pour savoir de combien de `commits` vous êtes en avance. Une fois que vous avez le nombre de commits d'avance que vous avez, faites un reset de ce nombre de commits pour retourner au dernier commit en commun entre le dépôt en ligne et votre dossier local.

Solution :

```
git fetch # récupérer les métadonnées
git status # voir le nombre de commits de différences
git reset HEAD~1 # ou un autre nombre si vous avez plus d'un commit de différence
```

Exercice 4.b Deuxième temps : mettre ses modifications de côté

Vos modifications sont encore présente dans votre dossier, ce qui peut empêcher de récupérer les modifications en ligne. Pour pouvoir récupérer les modifications en ligne, on va dans un premier temps mettre de côté les modifications.

On peut voir nos modifications courante (mais pas `add`) en regardant le status du dépôt.

Utilisez la commande `git stash` de la bonne façon pour mettre de côté vos modifications.

Solution :

```
git status # vérifier que nos modifications sont encore là
git diff # voir les modifications réelles, au cas où
git stash # ou un autre nombre si vous avez plus d'un commit de différence
```

Exercice 4.c Troisième temps : resynchroniser le dépôt et réappliquer les changements

À présent, on peut vérifier le status de notre dépôt et se convaincre qu'il est "propre", donc bon à être synchronisé avec le dépôt.

Avant de faire quoi que ce soit, listez les modifications dans votre *stash* et visualisez-les pour voir ce qui serait appliqué (afficher le diff).

Finalement, récupérez les modifications en ligne.

Solution :

```
git status # vérifier que le dépôt est "propre"
git stash list # voir la liste des modifications conservées dans le stash
git stash show -p 0 # voir le diff du dernier élément du stash
git pull
```

Exercice 4.d Dernier temps : resynchroniser le dépôt et réappliquer les changements

À présent que nous sommes "synchrones" avec le dépôt en ligne, affichez à nouveau les changements qui seraient appliqués avec le dernier *stash* pour voir s'il s'agit bien des mêmes changements que précédemment.

Appliquez les changements. Voilà, vous vous en êtes sorti ! Vous pouvez à présent pousser vos modifications.

Solution :

```
git stash show -p 0
git stash pop
git add .
git commit -m "modif readme"
git push
```

Exercice 5 Mot de fin

— Toujours récupérer les métadonnées avant de *pull*.
— On peut mettre de côté et réappliquer les changements que nous avons en conflit en quelques commandes rapides (avec de l'habitude, c'est très rapide).
— Ce n'est qu'une façon de faire ! Il reste plein de cas à voir, certains seront vus au S2.
— On verra également au S2 comment on peut gérer d'autres cas de conflits.