TP git

Afficher la liste des branches :

$ git branch

Créer une nouvelle branche

$ git branch <nom\_nouvelle\_branche>

Pour créer une nouvelle branche à un autre emplacement :

$ git branch <nom\_nouvelle\_branche> <revision>

Ces commandes ne changent pas la branche courante. Pour créer une nouvelle branche et en faire la branche courante, utilisez plutôt :

git checkout -b <nom\_nouvelle\_branche> *# ou*

$ git checkout -b <nom\_nouvelle\_branche> <revision>

#### Changer de branche

Cette opération consiste à modifier la copie de travail pour la mettre dans le même état que le sommet d’une branche.

$ git checkout <branche>

##### **À propos de git checkout**

La commande git checkout est utilisée dans divers contextes, qui rendent difficile à percevoir sa cohérence interne.

La fonction première de cette commande est de *modifier l’état de la copie de travail*. Selon ses arguments, elle a des effets supplémentaires :

* un branche : changer la branche courante
* une révision : passer en mode *detached HEAD*

Exercice

1. Dans le dépôt que vous avez créé au premier exercice, créez une branche nommée style, et placez-vous dans cette branche.
2. Modifiez la feuille de style (par exemple pour changer la couleur de fond) et commitez vos changements.
3. Revenez sur la branche master. Constatez que vos changements de style ont disparu (pour l’instant).
4. Dans la branche master, modifiez ou ajoutez du contenu au fichier HTML, et commitez vos modifications.
5. Revenez sur la branche style. Constatez que vos changements de style ont réapparu, mais que vos dernières modifications dans le fichier HTML ont, elles, disparu.
6. Modifiez à nouveau la feuille de style (par exemple pour changer la police) et commitez vos changements.

#### Fusionner deux branches

L’opération de **fusion** (en anglais *merge*) permet d’intégrer les modifications d’une branche dans une autre

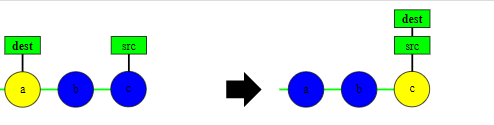
GIT permet également de fusionner plus de deux branches dans une même opération, mais nous n’irons pas jusque là dans ce cours.

Il y a deux situations possibles, selon les positions relatives de la branche à fusionner (source) et de la branche destination.

##### **Fusion sans commit**

Si la branche destination est contenue dans la branche source,

la fusion a simplement pour effet de déplacer le sommet de la branche cible.

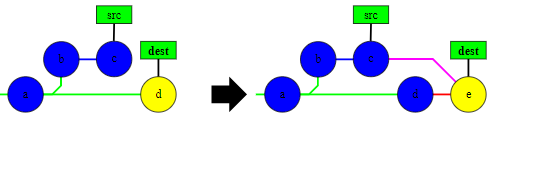


##### **Fusion avec commit**

Si la branche destination et la source ont divergé,

la fusion crée un nouveau commit intégrant les modifications des deux branches ;

ce commit devient le sommet de la branche destination.



git merge <branche>

##### **Exercice**

1. Nous allons maintenant fusionner la branche style (créée à l’exercice précédent) avec la branche master.
2. Placez-vous dans la branche master, et appliquez la méthode de votre choix (ligne de commande ou interface graphique) pour y fusionner la branche style.
3. Constatez que toutes vos modifications (contenu HTML et style) sont maintenant visibles.
4. Constatez également (dans l’historique) que le commit ainsi créé a *deux* commit parents.

## Gérer les conflits

### Motivation[¶](https://perso.liris.cnrs.fr/pierre-antoine.champin/enseignement/intro-git/#id19)[§](https://perso.liris.cnrs.fr/pierre-antoine.champin/enseignement/intro-git/slides/index.html#id19)

La fusion de branches est automatiquement gérée par GIT lorsque les modifications des deux branches portent sur :

* des fichiers différents, ou
* des parties distinctes des mêmes fichiers texte

#### Exemple géré par GIT

Branche 1 :

- La première ligne

+ La première ligne modifiée

La deuxième ligne

La troisième ligne

Branche 2 :

La première ligne

La deuxième ligne

- La troisième ligne

+ La troisième ligne modifiée

Fusion :

La première ligne modifiée

La deuxième ligne

La troisième ligne modifiée

#### Exemple non géré par GIT

Branche 1 :

- La première ligne

+ La première ligne modifiée

La deuxième ligne

- La troisième ligne

+ La troisième ligne modifiée

Branche 2 :

- La première ligne

+ La première ligne changée

La deuxième ligne

La troisième ligne

On a donc un **conflit** lorsque les deux branches modifient :

* un même fichier binaire, ou
* la même partie d’un fichier texte.

Dans ce cas, le conflit doit être résolu à la main avant de pouvoir créer le commit de fusion

#### Fichiers comportant un conflit

Les fichiers texte comportant un conflit sont automatiquement modifiés pour :

* inclure les modifications non conflictuelles, et
* faire apparaître les deux versions concurrentes pour les modifications conflictuelles.

<<<<<<< HEAD

La 1e ligne modifiée

=======

La 1e ligne changée

>>>>>>> src

La 2e ligne

La 3e ligne modifiée

Les fichiers binaires ne sont pas modifiés

#### Résolution du conflit

Une fois les fichiers en conflit corrigés, on peut résoudre le conflit :

* En ligne de commande :
* $ git commit -a

Le nouveau commit aura pour parents les sommets des branches fusionnées.

#### Abandon

On peut également décider d’abandonner la fusion :

* En ligne de commande :
* $ git merge --abort

##### **Exercice**

1. Créez un nouveau dépot, et ajoutez-y un fichier conflit.txt contenant le texte suivant :
2. La première ligne
3. La deuxième ligne
4. La troisième ligne
5. Créez plusieurs branches, dans lesquelles vous modifierez différemment le fichier conflit.txt, en suivant les exemples ci-avant. Tentez ensuite de fusionner ces branches.
6. Lorsque GIT vous signale un conflit, constatez comment le fichier conflit.txt a été modifié, et résolvez le conflit.