Question sur git

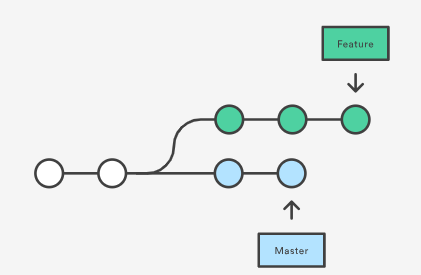
# Questions générales

1. **Expliquez pourquoi on utilise un système de gestion du code source et des outils de collaboration :**
   1. Pour pouvoir :
      1. Faire des retours en arrière sur des fichiers.
      2. Faire des *backups* plus facilement/rapidement.
      3. Voir les différences entre plusieurs versions de son code.
      4. Voir l’historique des modifications d’un projet.
      5. Partager ses fichiers avec d’autres personnes.
      6. Travailler à plusieurs en même temps sur des mêmes fichiers.
      7. Développer plusieurs fonctionnalités simultanément lorsqu’un système de branches est possible (comme avec Git).
2. **Est-ce que git est un système de gestion de versions centralisé ou décentralisé? Quelle est la différence entre les deux? Nommez un avantage et un inconvénient pour chacun des deux types systèmes :**
   1. Git est un système de gestion de versions décentralisé.
   2. Différence : Avec un système de gestion de versions centralisé, une copie du projet existe quelque part (généralement sur un serveur), et les développeurs y *commit* directement leurs changements, alors qu’avec un système décentralisé, les développeurs clonent une copie du *repo* sur leur poste.
   3. Décentralisé :
      1. Avantage : Permet (à l’exception des pull/push) de travailler sans connexion au réseau.
      2. Désavantage : Nécessite plus d’espace disque pour le développeur.
   4. Centralisé :
      1. Avantage : Plus facile à comprendre.
      2. Désavantage : Ne permet pas de travailler sans connexion au réseau.

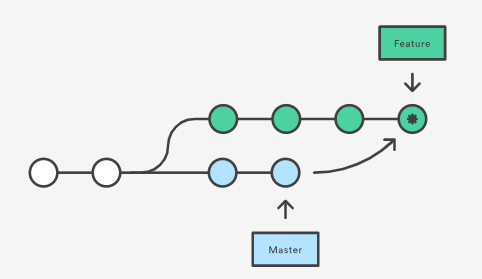
# Branching et Merge

1. **Quelle est la différence entre git merge et git rebase? Expliquez dans vos propres mots et illustrez votre réponse avec une image (originale ou tirée d'Internet, avec citation) :**

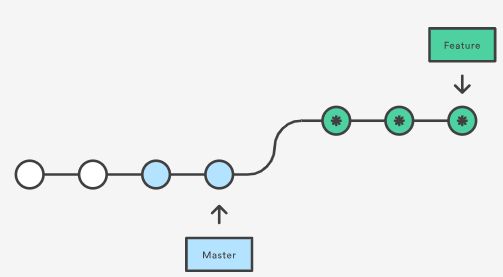
Situation initiale :



* 1. Merge : Crée un nouveau commit qui applique les changements d’une branche sur l’autre branche.



* 1. Rebase : Déplace l’endroit où la branche a été créée afin d’incorporer les nouveaux changements.

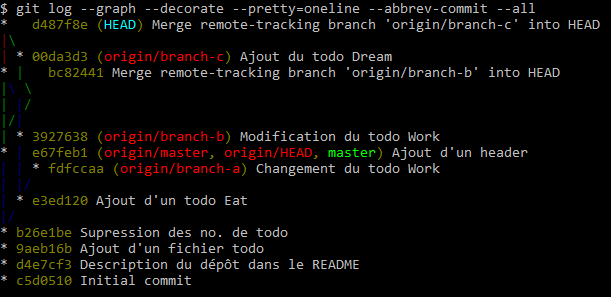


* 1. Les images sont tirées de https://www.atlassian.com/git/tutorials/merging-vs-rebasing

1. **Nommez trois situations (trois commandes git) qui peuvent mener à un merge conflict :**
   1. Deux personnes ont changé la même ligne, mais pas de la même manière.
   2. Une personne a modifié un fichier, alors qu’une autre personne a supprimé ce fichier.
   3. Une personne a ajouté une ligne juste avant ou juste avant une ligne qui a été supprimé par une autre personne.
   4. merge/pull : des modifications dans deux branches différentes affectent le même fichier au même endroit (même ligne).
   5. rebase : même raison qu’en a.
   6. stash pop/apply : Des changements non sauvegardés peuvent être écrasés par les changements présents dans le stash, si les modifications sont faites aux mêmes fichiers aux mêmes lignes.

# Exercices

1. **Est-ce qu'il y a un conflit entre branch-a et master? Pourquoi?**
   1. Non, car aucun message n’a été indiqué. Les fichiers n’ont donc probablement pas été modifié au même endroit.
2. **Est-ce qu'il y a un conflit entre branch-a et branch-c? Pourquoi?**
   1. Non, car aucun message n’a été indiqué. Les fichiers n’ont donc probablement pas été modifié au même endroit.
3. **Est-ce qu'il y a un conflit entre branch-a et branch-b? Pourquoi?**
   1. Oui, car le fichier “todo.txt” a été modifié au même endroit (ligne 2) dans les deux branches (Have fun vs. Work Hard)
4. **Qu'arrive-t-il si on supprime la branche branch-a, branch-b ou branch-c? Est-ce souhaitable?**
   1. On perdrait les changements en cours dans ces branches. Si un développeur s’est créé une branche pour ajouter une feature, puis se rend compte que cette feature n’est plus à faire, il pourrait supprimer la branche. Donc dans certains cas, ça pourrait être souhaitable.
5. **Faites un rebase de branch-b sur branch-a. Lister les commandes utilisées, assumant que votre HEAD est présentement sur master. S'il y a un conflit, nommez également les étapes utilisées pour le résoudre (commandes git, édition de fichier, utilisation d'outils graphiques de merge, etc.) :**
   1. git checkout branch-a
   2. git rebase branch-b
   3. Conflit :
      1. J’ouvre le fichier en conflit (todo.txt) dans un éditeur de texte (ex: notepad).
      2. Je décide de conserver les deux versions de la ligne (donc work hard ET have fun)
      3. Je supprime les lignes contenant les indicateurs de changement (”<<<<<< HEAD”, “=======” et “>>>>>> Changement du todo Work”).
      4. Je sauvegarde le fichier.
      5. git add todo.txt
      6. git rebase --continue
6. **Ensuite, lister les commandes pour merger branch-b sur master puis branch-c sur master. Ajouter une image de votre graphe de commits après toutes ces opérations. Cette image peut simplement être une capture d'écran de la sortie de git log --graph --decorate --pretty=oneline --abbrev-commit -–all :**
   1. git checkout master
   2. git merge branch-b
   3. git merge branch-c



1. **Maintenant, qu'arrive-t-il si on supprime la branche branch-a, branch-b ou branch-c? Est-ce souhaitable?**
   1. Il n’est pas souhaitable de supprimer la branche branche-a, car il n’y aura alors aucune liaison qui pourra retrouver l’historique de cette branche. Elle disparaîtrait avec tous ses commits. Pour les branches branche-b et branche-c, alors il n’y a aucun problème, puisque l’historique de ces branches est lié à celle de la branche principale (master) grâce au merge. Il serait donc effectivement souhaitable de les supprimer si on s’attend à ce que plus personne n’y travaille. Sinon, on peut simplement la recréer par la suite.