utbm
université de technologie
Belfort-Montbéliard

Rapport sur les commande GIT

ABDELAZIZ Hassan Ouamr

TPLO43 Autonme 2019

Sommaire

Les différents commandent suivis sur le tutoriel

- git commit
- git branch
- git checkout
- git cherry-pick
- git reset
- git revert
- git rebase
- git merge

Introduction

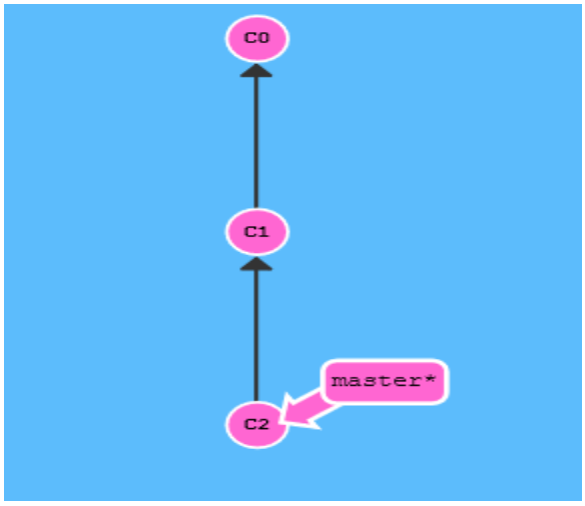
Git est un système de contrôle de version open-source qui a été lancé par Linus Trovalds-la même personne qui a créé Linux.

Il existe une large variété de commandes git disponibles dans le mode bac à sable. Grace au tutoriel sur les commandes, nous avons étudié quels que commandes ci-dessous.

Les différents commandes utilise et leurs fonctionnement

- ## git commit

La **commande git commit** permet de **valider les modifications apportées** au HEAD. Notez que tout commit ne se fera pas dans le dépôt distant. Les commits sont très légers et passer de l'un à l'autre est très rapide !
A chaque fois quand on exécute la commande **git commit,** elle fera une modification au niveau de parent.

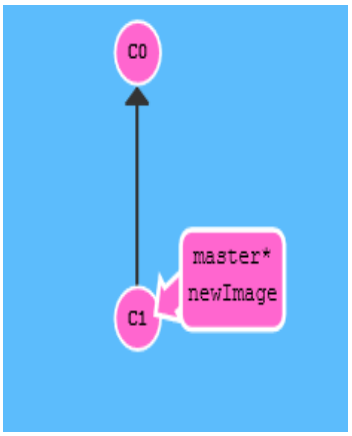


git commit

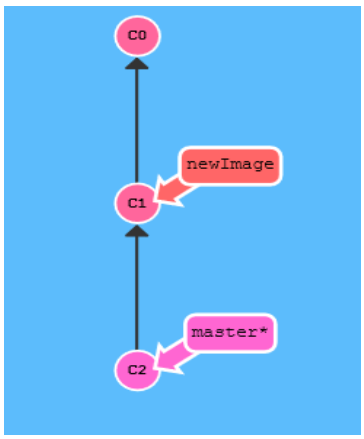
- ## git branche

Les branches sur git sont légers et sont des références sur un commit spécifique.

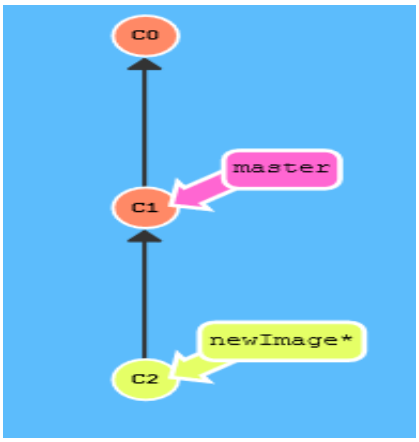
La commande **git branch newImage** nous donne une nouvelle branche qui se réfère à la commit C1.



git branch newImage



git checkout newImage

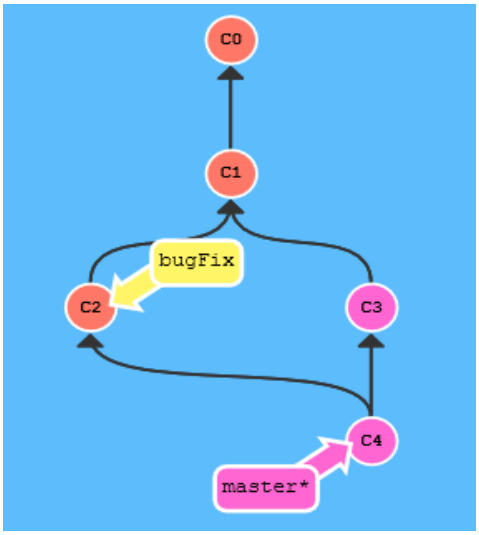


git commit

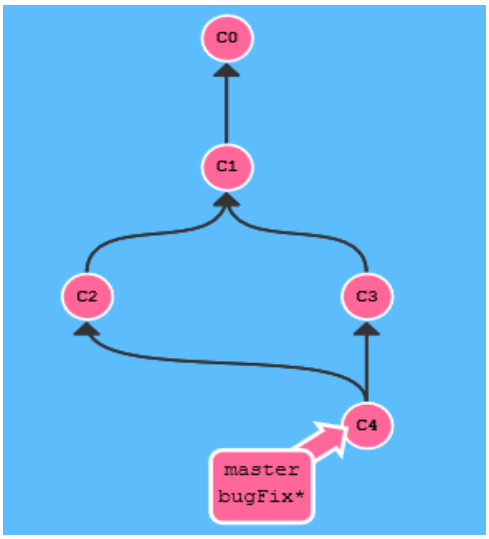
- # Branches et Merges

  Cette commande permet de combiner deux branches parents. La commande qui permet de fusionner les deux parents est :

- **git merge bugFix (ma branche)**



git merge bugFix

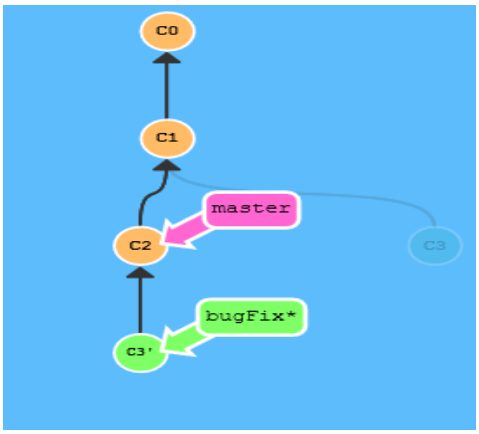


git merge bugFix

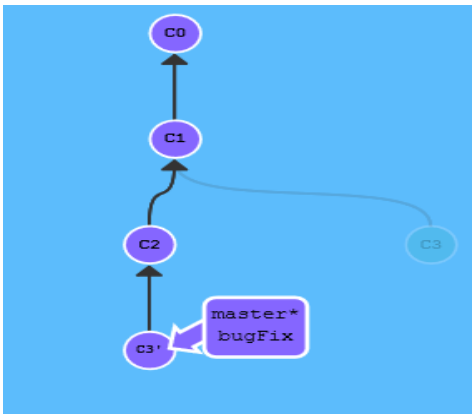
- # git Rebase

  Cette commande permet de combiner les contenus de deux branches. **Rebase** prend un ensemble de commits, les "recopie", et les ajoute en bout de chaîne à un autre endroit.

L’avantage de **rebase** est de permettre d'obtenir une simple séquence linéaire de commits.



git rebase



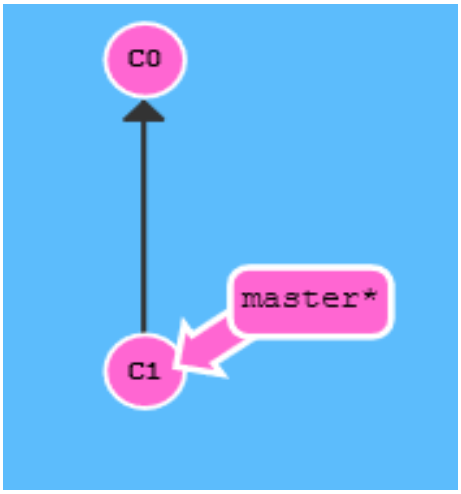
git rebase

- # Se déplacer dans Git
- # HEAD

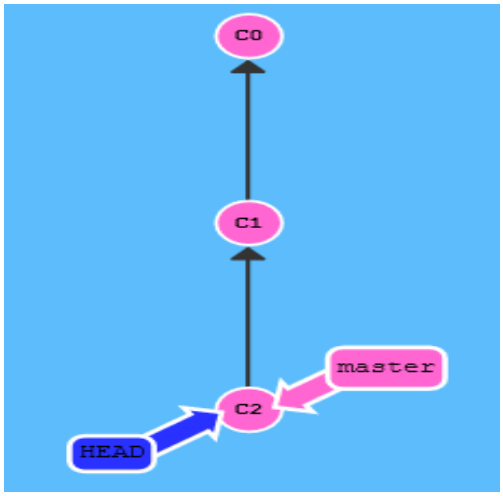
HEAD est le nom symbolique pour le commit sur lequel nous nous situons actuellement plus simplement c'est le commit sur lequel nous travaillons.

HEAD pointe toujours sur le commit le plus récent dans l'arbre des commits. La plupart des commandes git qui modifient l'arbre des commits vont commencer par modifier HEAD.

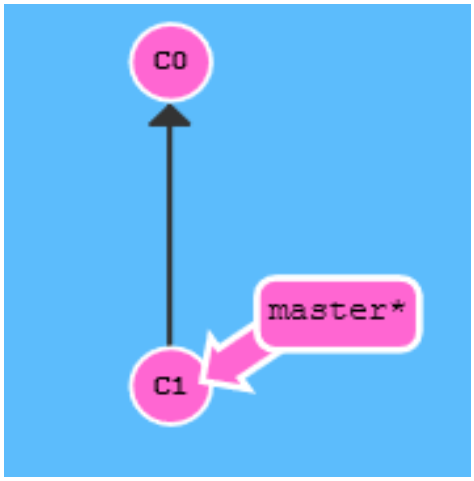
Normalement HEAD pointe sur le nom d'une branche (comme bugFix). Quand vous effectuez un commit, le statut de bugFix est modifié et ce changement est visible par le biais de HEAD.



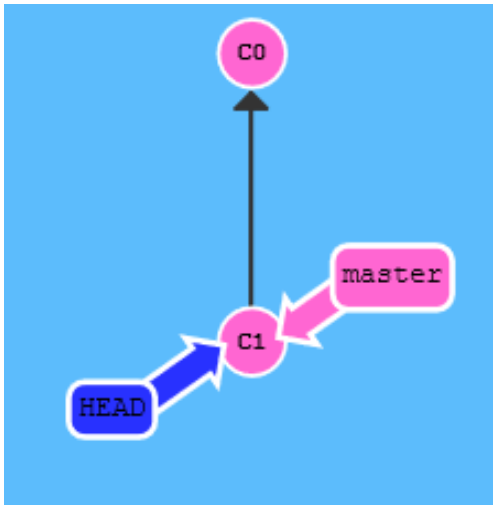
Avant la commande



git checkout C1 ; git checkout master ; git commit ; git checkout C2

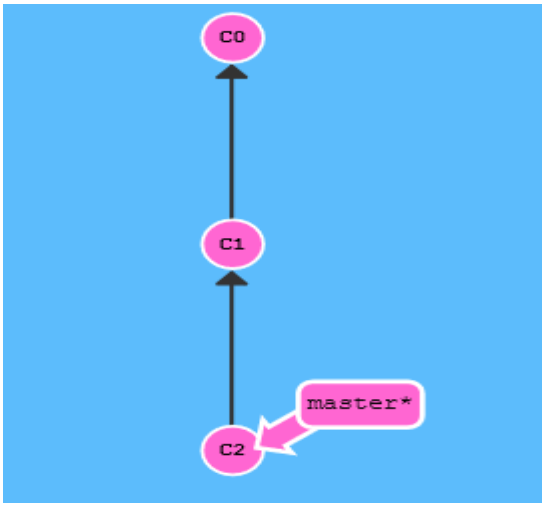


Avant la commande

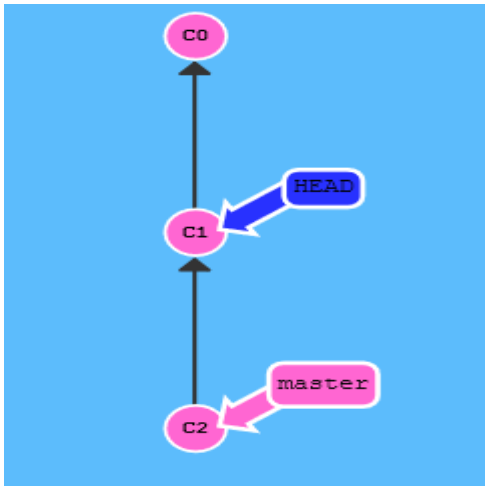


git checkout C1

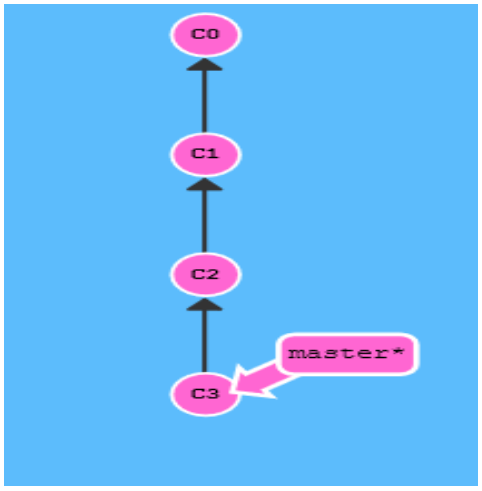
- **Les commits relatifs sont puissants, et on va en introduire deux simples ici :**
  - Revenir d'un commit en arrière avec ^
  - Revenir de plusieurs en arrière avec ~**<num>**

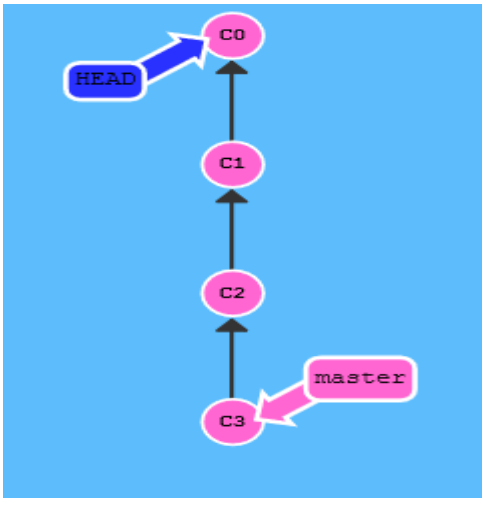


Avant la commande



git checkout master^





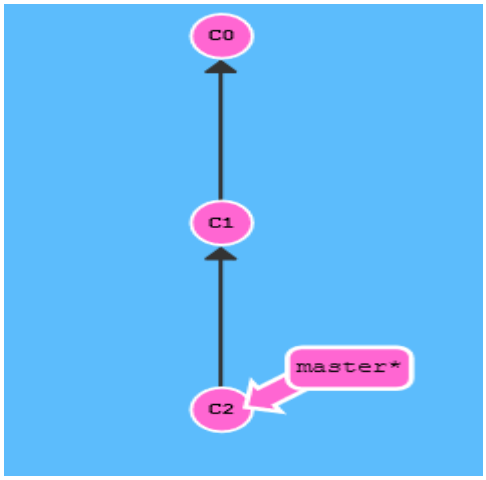
Avant cette commande
git checkout C3 ; git checkout HEAD^ ;git checkout HEAD^ ; git checkout HEAD^ ;

git checkout C3 ; git checkout HEAD^ ;git checkout HEAD^ ;

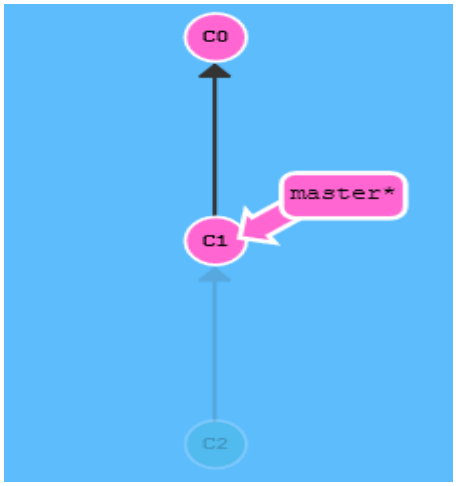
- **git reset et git revert**

Il y a principalement deux façons d'annuler des changements avec Git :

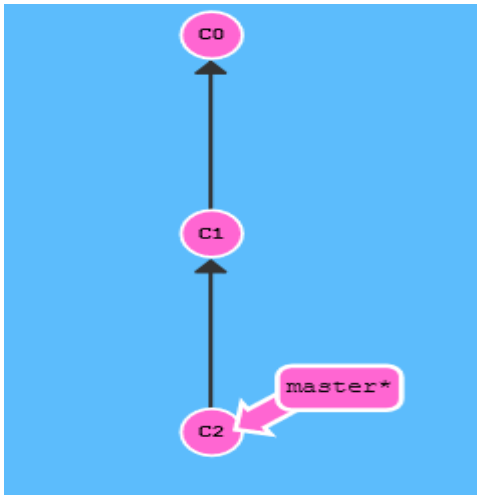
- **git reset**
- **git revert**



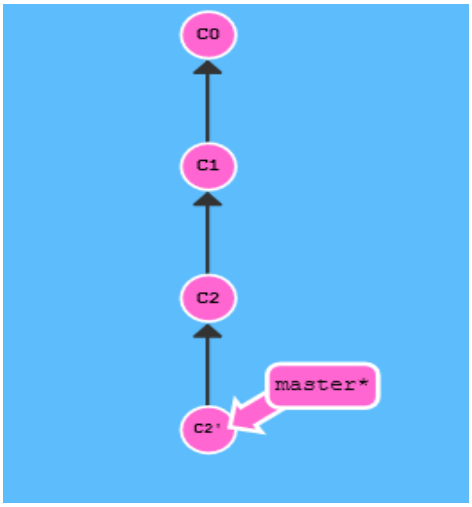
Avant la commande git resert HEAD~1



git resert HEAD~1



Avant la commande git resert HEAD



git resert HEAD

Déplacer votre travail

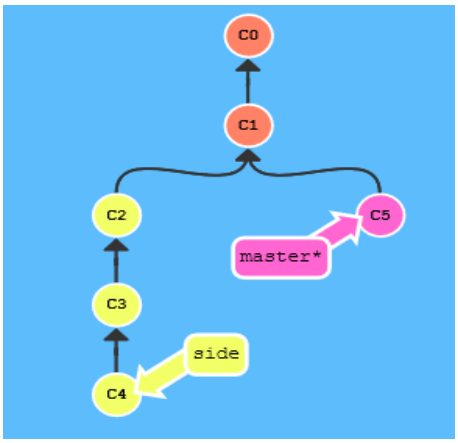
Nous allons maintenant se focaliser sur le déplacement de travail.

- ## git Cherry-pick

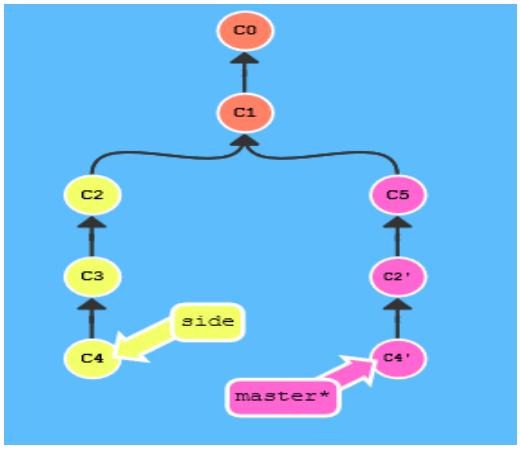
  La première commande de cette série est **git cherry-pick**. Elle a le prototype suivant :

  - **git cherry-pick<Commit1><Commit2><…>**

  C'est une manière simple de dire qu'on voudrait copier une série de commits en-dessous de notre emplacement actuel (HEAD).



Avant la commande git cherr-pick



git cherr-pick C2 C4

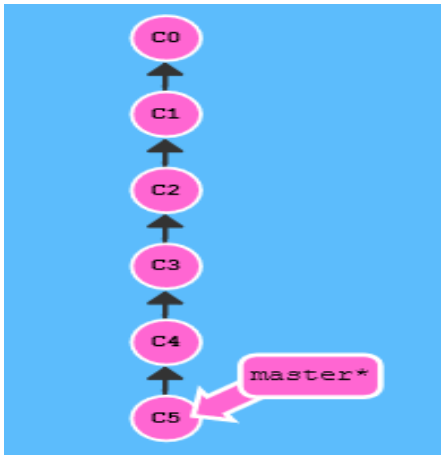
- ## git rebase

  Git cherry-pick est pratique quand vous savez exactement quels commits vous voulez (*et* que vous connaissez leurs identifiants.

  Dans le cas où nous ne connaissons pas les identifiants, nous allons utiliser la commande **git rebase.**

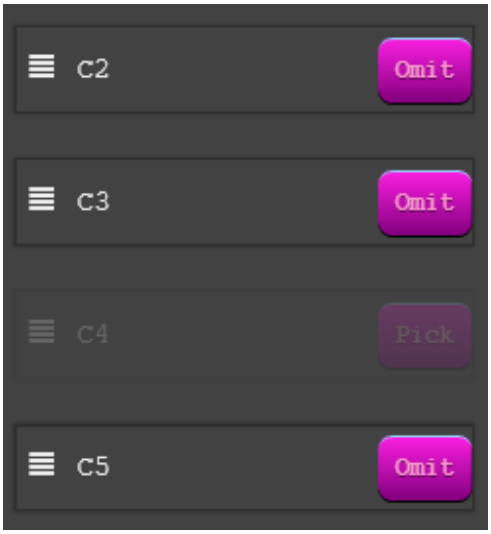
Quand le rebase interactif s'ouvre, vous avez la possibilité de faire 3 choses :

- Vous pouvez réarranger les commits simplement en changeant leur ordre dans l'interface graphique.
- Vous pouvez omettre certains commits. Cela est désigné par **pick**: cliquer sur pick désélectionne/resélectionne le commit.
- Enfin, vous pouvez écraser des commits.

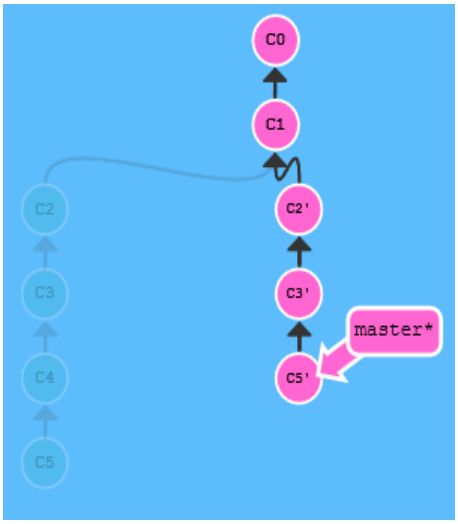


Avant la commande git rebase –i HEAD~4

git rebase –i HEAD~4



pick C4 confirmation



git rebase –i HEAD~4

Conclusion : Bref ce tutoriel nous a permis de vusialiser les commandes et il nous a permis aussi d'apprendre le fonctionnement des differents commandes.