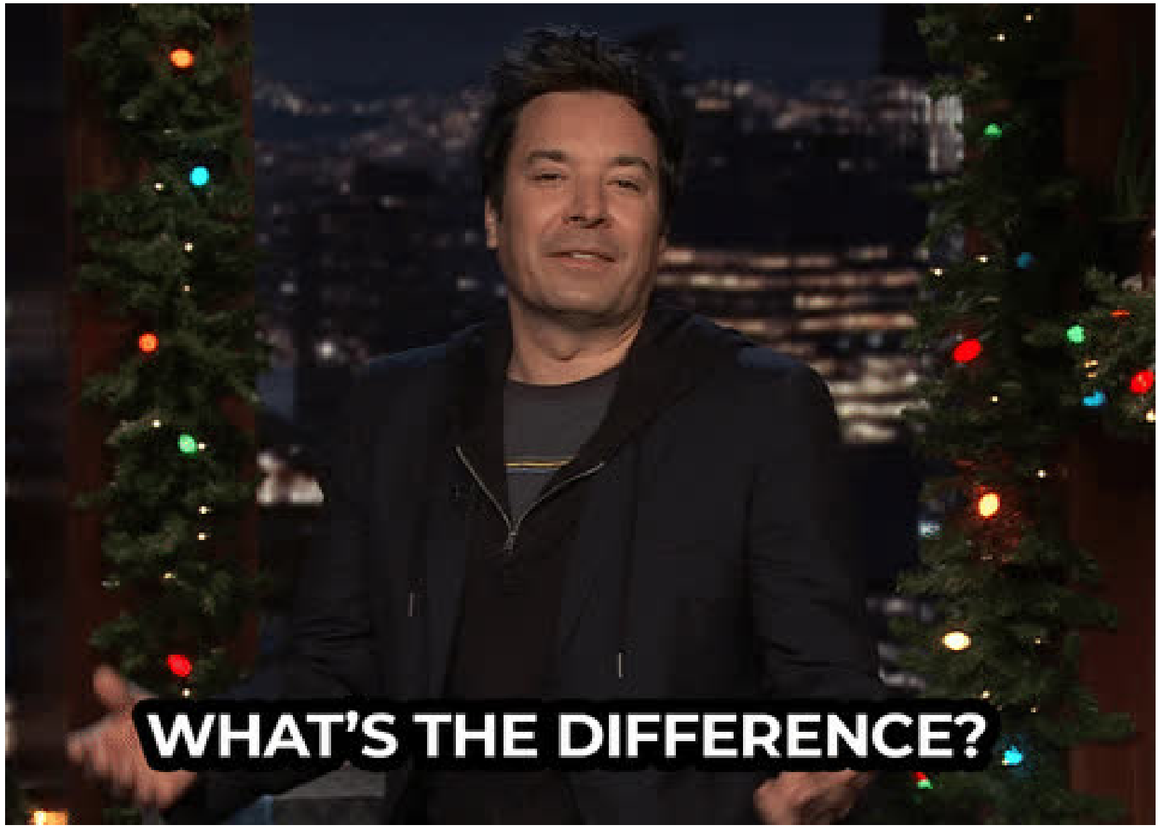
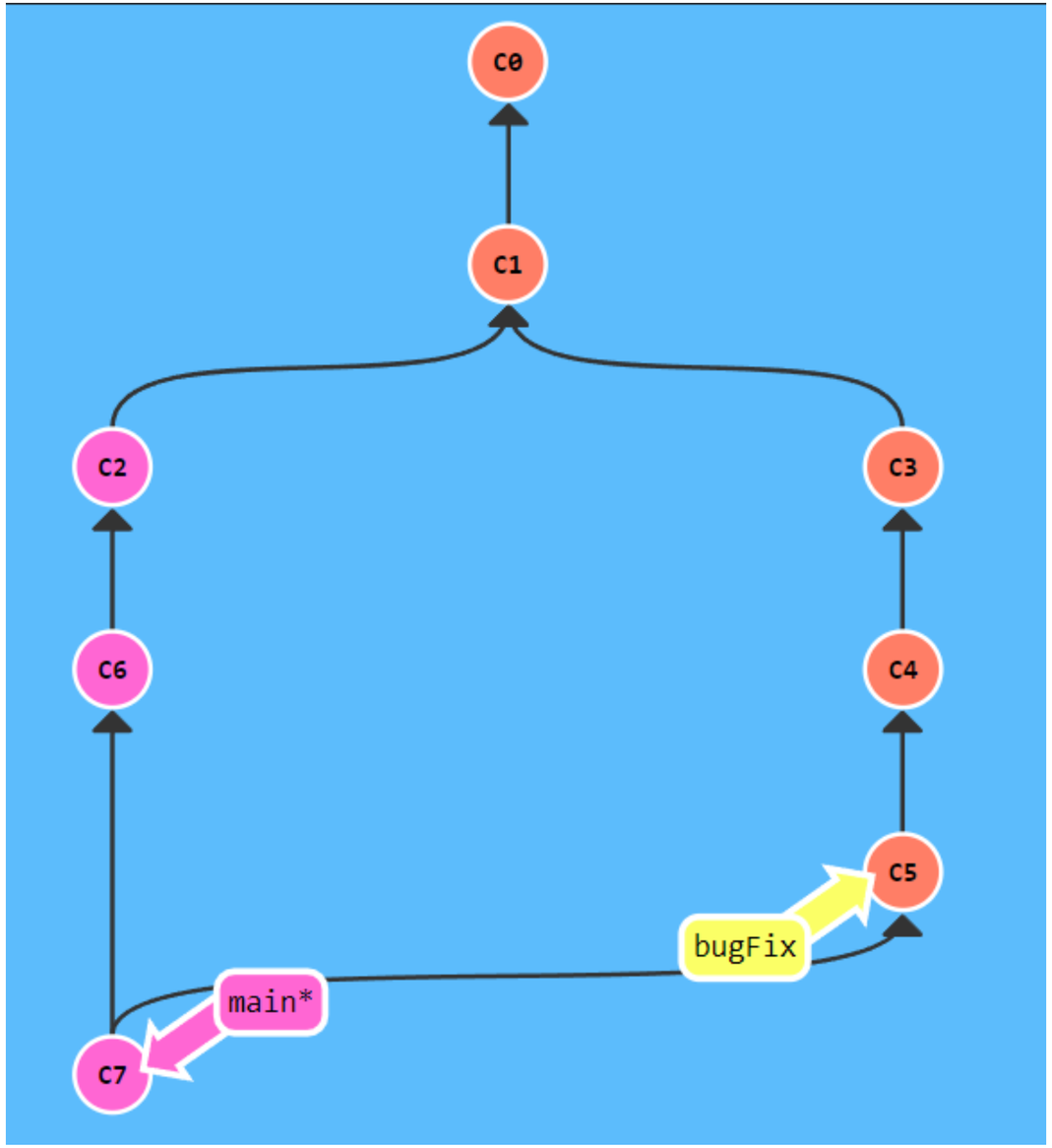
Bien comprendre la différence entre git merge et git rebase

Feys Dylan
BeCode.org


WHAT'S THE DIFFERENCE?

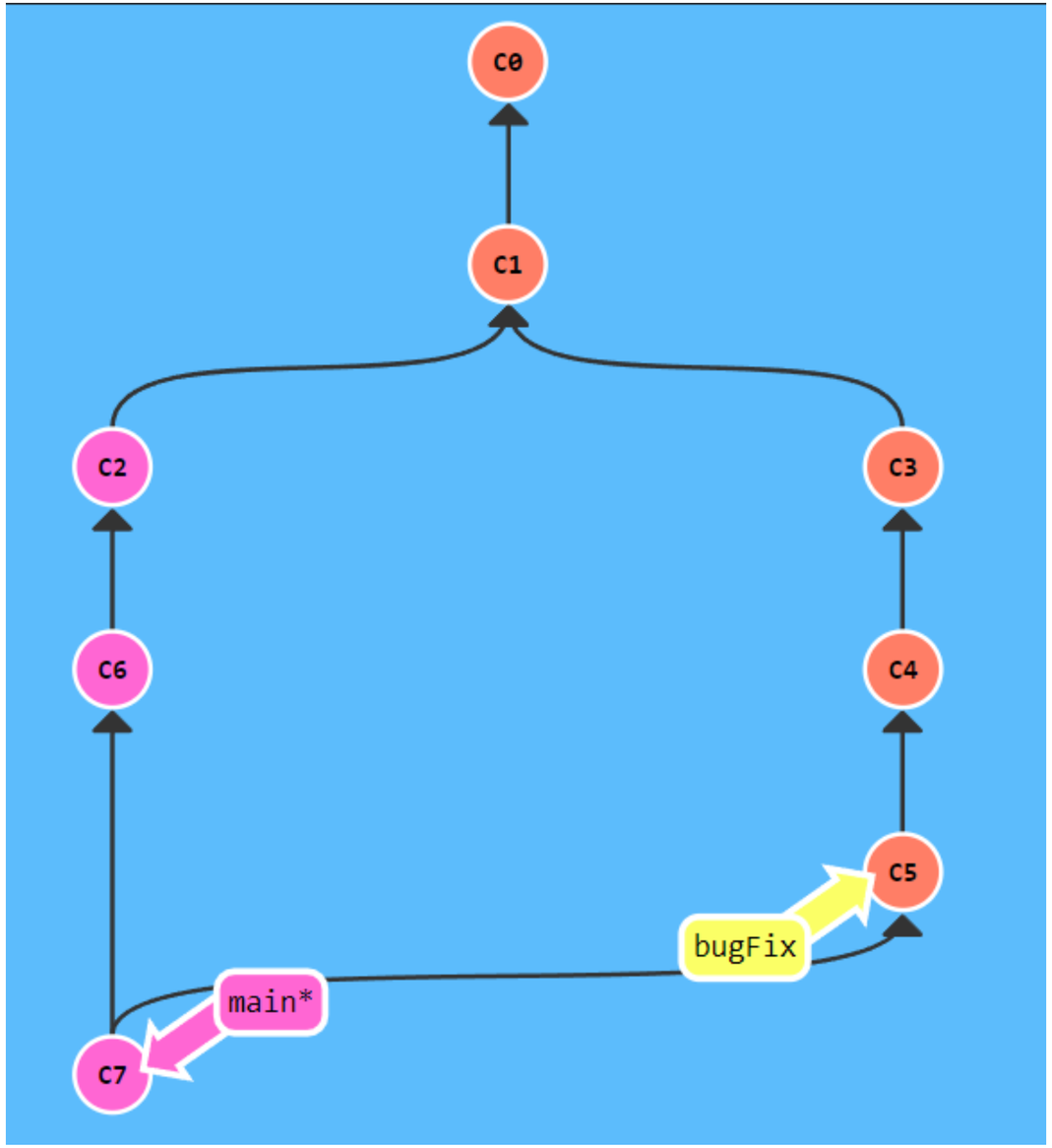
git merge et **git rebase** sont des concepts
assez flous pour la plupart des dev

De manière général, git merge et git rebase servent à fusionner des branches vers la branche principal



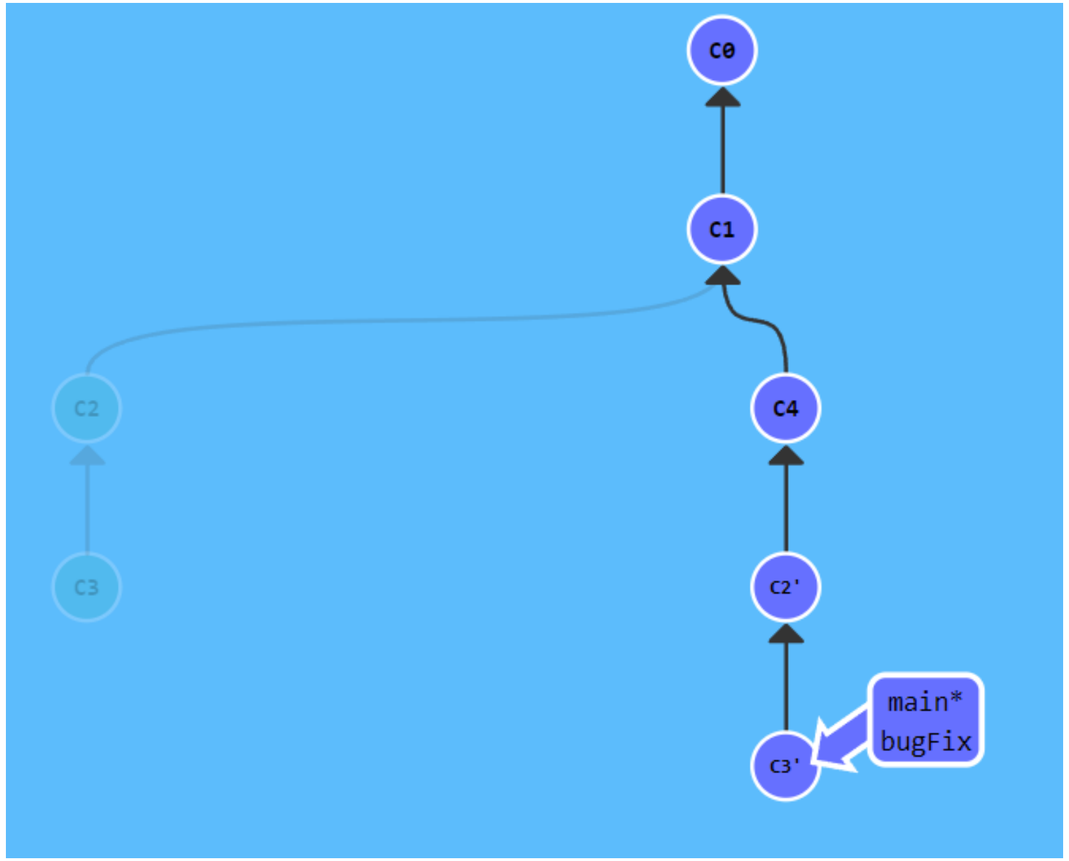
Comment fonctionne git merge ?

- Localisation des points de fusion
- Création d'un commit de fusion
- Résolution des conflits (si nécessaire)
- Validation de la fusion
- MàJ de l'historique
- Nettoyage des références



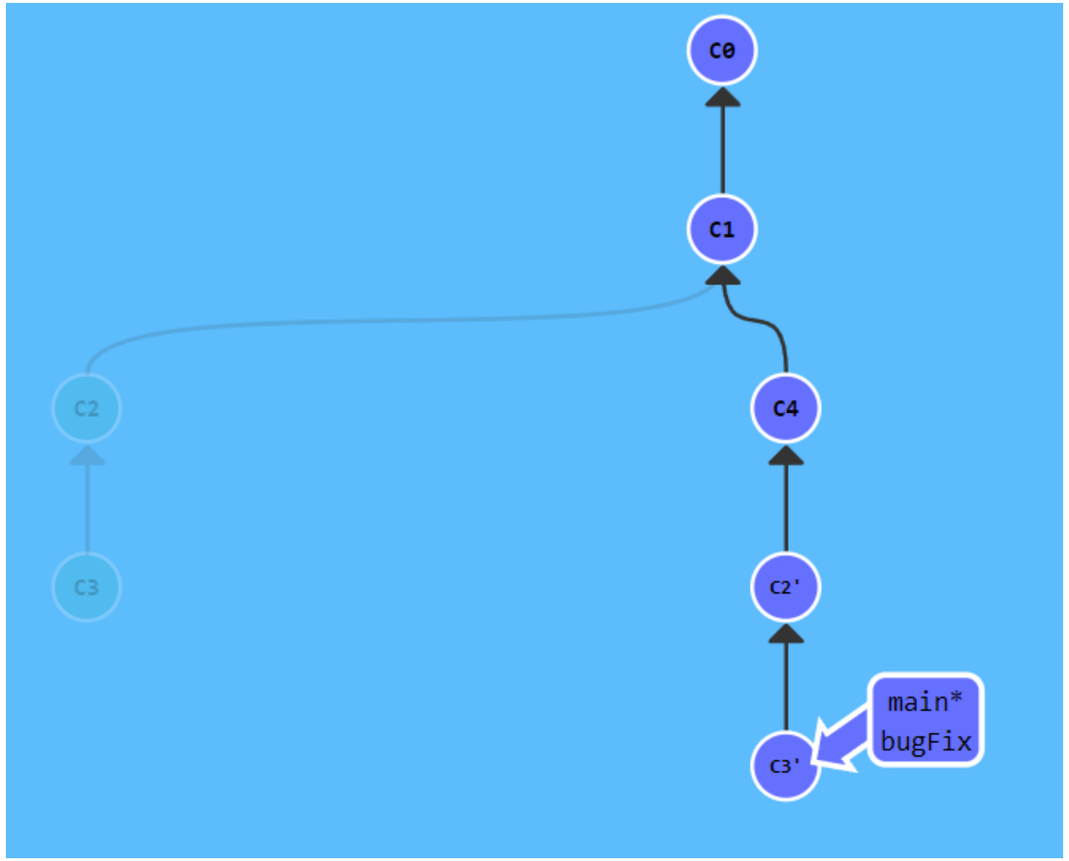
Comment fonctionne git merge ?

En résumé :
fusionne les deux branches en créant un nouveau commit
Le tout en gardant l'historique



Comment fonctionne git rebase ?

- Détermination des différences
- Sauvegarde des commits
- Déplacement de la branche actuelle
- Réinitialisation de la branche
- Réapplication des commits
- Gestion des conflits (si nécessaire)
- Validation de la réapplication
- MàJ de l'historique
- Nettoyage des références



Comment fonctionne git rebase ?

En résumé :
réapplique les commits d'une branche sur une autre branche en modifiant l'historique des commits pour paraître comme s'ils avaient été faits sur la branche cible dès le début.

Il existe une autre commande magique ! Git rebase -i nous permet d’être les gardiens du temps

Comment fonctionne git rebase -i ?

- Après avoir taper la commande, Git ouvre un éditeur de texte.
- Chaque commit est présenté avec une commande par défaut (généralement pick)
- Plusieurs commandes disponibles
- Modification de l’ordre des commits
- Sauvegarde des modifications
- Traitement des commits
- Pause pour édition ou fusion
- Résolution des conflits
- Continuation du rebase

Comment fonctionne git rebase -i ?

En résumé :
git rebase -i offre un contrôle précis sur la réécriture de l'historique des commits, permettant à l'utilisateur de modifier leur ordre, les messages, de les fusionner et de supprimer des commits indésirables, le tout de manière interactive.

Dans quel cas est-il plus intéressant d'utiliser git merge ?

- Intégration de branches distinctes
- Préservation de l'historique d'origine
- Traitement automatique des conflits
- Facilité de la collaboration

Dans quel cas est-il plus intéressant d'utiliser git rebase ?

- Nettoyage de l'historique
- Intégration de vos modifications avec la branche principale
- Evitement de commit de fusion inutile
- Réduction de l'encombrement de l'historique

En conclusion :

Git merge est idéal pour fusionner les branches de façon non linéaire et préserver l'historique.

Git rebase est utile pour réorganiser l'historique des commits de manière linéaire et propre. Permet de replacer la branche sur la pointe d'une autre branche en évitant les commits de fusion.

En conclusion :

Dans la réalité, l'utilisation de l'un ou l'autre varie en fonction :

- des préférences du dev / de la boite
- de la situation