TP : Utilisation de Git Rebase

Prérequis :

Avoir Git installer sur votre machine

Avoir une base sur l'utilisation de Git

1. Configuration initiale :

Tout d'abord, créez un nouveau répertoire pour votre TP et initialisez un dépôt git.

mkdir git-rebase-tp

cd git-rebase-tp

git init

2. Créez une base de travail :

Créez un fichier et effectuez quelques commits :

echo "Ligne 1" > fichier.txt

git add fichier.txt

git commit -m "Ajout de la ligne 1"

Répétez l'opération pour quelques autres lignes.

3. Créez une nouvelle branche :

Créez une nouvelle branche et ajoutez quelques commits.

git checkout -b nouvelle-branche

echo "Ligne A" >> fichier.txt

git commit -am "Ajout de la ligne A"

echo "Ligne B" >> fichier.txt

git commit -am "Ajout de la ligne B"

4. Faites des modifications sur la branche principale :

Retournez sur la branche master (ou main, selon la configuration de votre Git) et effectuez d'autres commits.

git checkout master

echo "Ligne 2" >> fichier.txt

git commit -am "Ajout de la ligne 2"

5. Rebasez la nouvelle-branche :

Revenez sur votre nouvelle-branche et utilisez rebase :

git checkout nouvelle-branche

git rebase master

Il est possible que vous rencontriez des conflits. Si c'est le cas, Git vous indiquera les fichiers qui posent problème. Résolvez les conflits manuellement, puis continuez le rebase :

git add fichier.txt

git rebase --continue

6. Vérifiez l'historique :

Utilisez git log pour voir l'ordre des commits. Vous verrez que les commits de la nouvelle-branche ont été déplacés "au-dessus" des commits de la branche master.

Cas d'utilisation :

Garder une branche propre : Si vous travaillez sur une fonctionnalité ou un bugfix pendant un certain temps, rebase vous permet de vous assurer que vous travaillez toujours par rapport à la dernière version de la branche principale (par exemple, pour intégrer les derniers correctifs de sécurité).

Éviter les "merge commits" : Certains projets préfèrent avoir un historique linéaire sans "merge commits". Rebase vous permet d'atteindre cet objectif.

Réécrire l'historique : Avec rebase, notamment l'option -i pour un rebase interactif, vous pouvez réécrire l'historique, fusionner des commits, modifier des messages de commit, etc.

Extraire un ensemble spécifique de commits : Dans les grands dépôts, il peut parfois être utile de prendre un ensemble de commits et de les appliquer à une autre branche. Rebase peut faciliter cette tâche.

Attention : N'utilisez jamais rebase sur des commits qui ont été poussés sur un dépôt distant et partagés avec d'autres personnes. Cela réécrirait l'historique public, ce qui est généralement une mauvaise idée. Utilisez rebase principalement pour des commits qui n'ont pas encore été partagés.