Resolução de Conflitos

***É prudente criar um ramo temporário de backup e outro para efetuar o ajuste**
git switch -c hotfixes/<timestamp>/fix-branch-temp #temp
git switch -c hotfixes/<timestamp>/fix-branch-temp/backup #criar backup a partir do temp
#retorna ao temp
git switch hotfixes/<timestamp>/fix-branch-temp

Procura a HEAD desejada
git reflog

HEAD e HASH
HEAD é o commit atual ou estado, mas o ideal é usar a hash do commit
HEAD~1 é o pai do commit atual

cherry-pick

Trazer um commit específico de um branch qualquer, direto para o main
Caso de Uso: Quando um bug que afetava a main atual já foi solucionado em outra branch recente, mas é preciso inserir o trabalho efetuado em uma outra branch mais antiga.
#Ir para o main
git **cherry-pick** <hash do commit>

bisect

Voltar ao estado de commit
#Busca dentre determinados commits até encontrar o momento exato que determinada alteração foi aplicada
git bisect **start** #Inicializará a busca
#É preciso informar a ele um estado, ou commit cuja **parte do código que queremos não esteja boa**
#Começa informando onde é o estado ruim
git bisect **bad** HEAD

#Em seguida devemos informar o estado em que possivelmente estava bom, isto é, a partir de onde ele irá buscar o commit desejado. Estes serão os limites da busca.
git bisect **good** <hash do commit>
#Verifica as alteração no código na IDE e faz os testes
#Obteremos a mensagem de quantas revisões a mais há para testarmos depois desta, e ele nos mostrará o estado.
#E se ele ainda não encontrar a alteração desejada, continuaremos executando **bisect bad**
git bisect **bad**
#Verifica as alteração no código na IDE e faz os testes
#Ao encontrar o estado desejado, **copiar o hash mostrado no último BAD** (será usado mais a frente) e roda o **GOOD**
git bisect **good**
#Após o **bisect good,** verifique as alterações no código na IDE e faça os testes, caso ainda queira, pode fazer o **bisect bad** novamente.
#Para finalizar a busca, uma vez que o Git já nos entregou o hash e a descrição do commit, faremos o **reset**, com o qual voltaremos à main
git bisect **reset**

DICA:
#Com o hash do commit, poderíamos desfazer a alteração, analisar o porquê da inclusão dela, perguntar o que aconteceu à pessoa que realizou o commit. Para verificarmos todas as alterações referentes ao commit, aplicamos o comando
git show <hash do commit mostrado no último BAD>
#Para reverter, pode-se utilizar
git revert <hash do commit mostrado no último BAD>

#Em caso de aparecer uma mensagem de erro de CONFLITO, será preciso verificar os arquivos alterados (git status), pois pode haver conflito, e haverá linhas duplicadas escritas nos arquivos, e algumas linhas devem ser apagadas.

git add .
git commit -am "foo bar"

blame

#Verifica quem efetuou os commits em um arquivo
git blame <file>

Dica: Lembre-se do SCRUM, **não devemos apontar culpados, procurar resolver em equipe, e se possível, fazer críticas construtivas.**
O **objetivo do blame** é para saber **para quem perguntar** sobre determinado bloco de código que não entendemos.

checkout

***NÃO É RECOMENDADO**
Para retornar ao HEAD perdido e criar um branch temporário
git reflog #Procura a HEAD desejada
git **checkout** <hash> #Entra na HEAD desejada
git branch -m <none temp> #Renomeia, senão tudo será perdido ao retornar a outro ramo
#Efetua a verificação ou correção
git switch main #Entra em main
git merge temp #Mescla em main
git branch -d temp #Deletar temp

O **git checkout <file>** foi dividido em dois (git **restore** e git **switch**)
Não usar: git checkout <file>

restore

Retornar arquivo ou versões
git restore <file> #Retorna ao estado anterior ao último commit (**<file> ou .**)
git restore --source <hash> <file> #retorna ao estado que estava no hash indicado
git restore **--stage** --source <hash> <file> #mantém os arquivos atuais, mas apaga os commits

revert

git revert <hash>

reset

Exclua o commit mais recente, mantendo o trabalho que você realizou
git reset --soft HEAD~1

Exclua o commit mais recente, destruindo o trabalho que você realizou
*****git reset --hard HEAD~1*****

Total de commits
git rev-list --all --count