**Pull. Fetch, Rebase**

1. **Pull**

**git pull -r**

This performs a special type of merge called a “rebase”, which makes sure that your project history is  
very simple. If you instead run git pull (without the -r option), a complex history will be recorded.

*"You should always commit before pulling"*

**git pull REMOTE-NAME BRANCH-NAME**

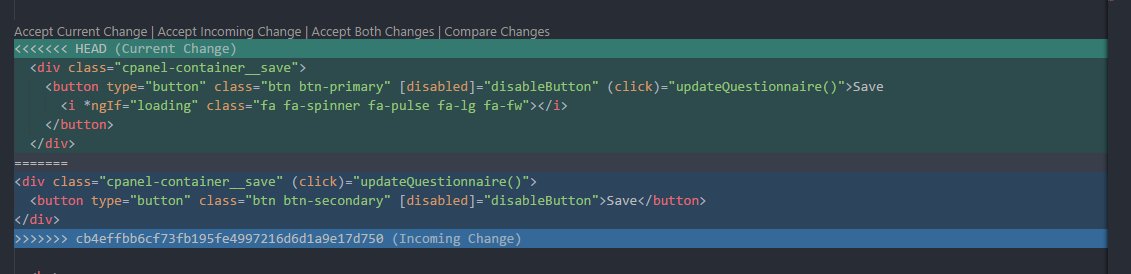
to pull from a specific branch (remote-name is most often "origin", which is the url of the remote repo)

Conflicts after pulling:

1. Solve conflicts through CLI

> cat path/to/file.c # view the current conflicts  
more code  
more code  
<<<<<<< HEAD  
code added by you  
=======  
code added by colleague  
>>>>>>> I, your colleague, changed file.c. This is the commit message  
more code  
more code  
> nano path/to/file.c # resolve all the merge conflicts by editing the file   
> git add path/to/file.c # mark it as resolved  
> git status # are all the conflicting files marked as resolved?  
> git diff --cached # Make sure the code builds and works.  
> git rebase --continue # tell git that you are done

1. Solve using IDE UIs.



Current change: your changes

Incoming change: other developer's changes