Frequently used git commands

git clone https://github.com//your_account/your_project //this command should be invoked only once. It basically downloads the entire remote repository i.e the repository from GitHub.

git fetch origin // Updates downloaded remote repository if the remote repository is ahead in commits.

git merge <**branch**> // merges the local branch named "branch" with the current branch that you are on.

Note: Before merging, first view what branch you currently are by invoking the **git branch** command, and make sure you are currently on the intended branch. This caution is due to the nature of the merge command. It merges the provided branch with the currently checked out branch.

git merge origin/branch_a //merges the remote branch named "branch_a" with the current branch that you are on.

Note: Before merging, first view what branch you currently are by invoking the **git branch** command, and make sure you are currently on the intended branch. This caution is due to the nature of the merge command. It merges the provided branch with the currently checked out branch.

git push origin <**branch_a**> //. The provided name "branch_a" is the local branch's name. Upon invoking this command, the remote branch that matches this name is replaced by the provided branch. In other words, replaces the remote branch named "some_branch_a" with the local branch named "some_branch_a"

Source: https://github.com/sanskarbhusal/git-commands

During conflict, try these based on scenario

If git yells "conflict" during push or merge, just abort the merge using:

git merge --abort // Cancels the merge process.

, and then try one of these commands below based on the scenario.

git reset --hard <branch> // The current branch that you are on will get reset to match the state of the specified local branch named "branch"

git reset --hard origin/<remote_branch> // The current branch that you are on will get reset to match the state of the specified remote branch named "remote_branch"

git push origin <branch> --force // replace the remote branch "main" with the local branch named "branch". **--force** flag is explicitly tell git to delete the remote branch named "branch" and replaces with local branch named "branch"

Source: https://github.com/sanskarbhusal/git-commands

git push origin <branch> --force-with-lease // similar to forced push but does it safely. If the remote branch is behind your local branch in commit i.e no one has pushed any changes since you last fetched and merged from that remote branch "branch". Use this if you don't want to delete the commits of others from the remote, because **--force** is kind of rude in the sense that it deletes the specified branch from remote and replaces it with your local one, resulting in loss of contribution of other collaborators in the team.

The above commands replace the entire branch. If you want to replace only specific files, try this instead:

git checkout origin/<branch> -- path/to/file //replaces the 'file" of current branch with the "file" from remote branch named "branch"

This will replace the specified "file" on your current branch with the one from the specified remote branch. The specified remote branch needn't be the same as the one you are currently checked out on your local repo. You can specify any remote branch and be on a different branch on your local, it will still copy and replace the specified file from the specified remote branch.

To do local to local replacement, use this:

git checkout <branch> -- path/to/file //replaces the "file" of current branch with the "file" of the specified local branch named "branch"

*[Note that, in few of the commands that we saw above, not including "origin/" inferred that the specified branch is the local branch and including "origin/" before branch name inferred to remote branch. Keyword "origin", by default, is a reference to the remote repository that you clone using **git clone** .]*

Source: https://github.com/sanskarbhusal/git-commands

*!!! Be aware that the above commands are meant to replace, not merge. The merge strategy used by **git merge** command is a bit confusing and tedious to work with and not even necessary if you are a solo developer on your project. **git merge** command only benefits collaborative projects where more than one developer pushes changes on the same repo. So, during merge conflicts, replace the branch or conflicting file using above commands as you see fit. But, don't use it heavily, 'cause it might develop the habit of completely replacing conflicting files and branches, which won't be appreciated in a collaborative project, once you join a company or start contributing to an open source code base. Use them carefully.*

If you are in the middle of git merge because of the conflict, you can also do this.

git checkout --ours . //keep the current branch's version of the conflicted file

or,

git checkout --theirs . //keep the other branch's version of the conflicted file

*Note that the above command will only work when you are in the middle of the merge i.e you fired "**git merge …..**" and git presented you with conflict and you haven't fired "**git merge --abort**" yet, which means you are stuck in the middle of the merge.*

Miscellaneous commands

git init // to convert the current working folder into a local git repo

Source: https://github.com/sanskarbhusal/git-commands

git branch -M <new_branch_name> // to rename current branch name from whatever it is currently named to “new_branch_name”.

git remote add origin https://github.com/user_name/repository_name.git // to set the url of the remote repository with the alias “origin”. This allows you to run commands such as **git push origin, git fetch origin**, etc.

git remote remove origin // to remote the alias “origin”

Source: https://github.com/sanskarbhusal/git-commands