My **Git** Commands

By cfrBernard

Start a Project

Create a local repo (omit <directory> to init the current directory)

```
$ git init <directory>
```

Download a remote repo

```
$ git init <directory>
```

Branches

List all local branches. Add -r flag to show all remote branches. Add -a flag for all branches

```
$ git branch
```

Create a new branch

```
$ git branch <new-branch>
```

Switch to a branch & uptdate the working directory

```
$ git checkout <branch>
```

Create a new branch and switch to it

```
$ git checkout -b <new-branch>
```

Delete a merged branch

```
$ git branch -d <branch>
```

Delete a branch, whether merged or not

```
$ git branch -D <branch>
```

Tag Management

Add a tag to current commit

```
$ git tag <tag-name>
```

Add an annotated tag to current commit

```
$ git tag -a <tag-name> -m <text>
```

Push a tag

```
$ git push origin <tag-name>
```

Make a Change

Add a file to staging

```
$ git add <file>
```

Stage all files

```
$ git add .
```

Commit all staged files to git

```
$ git commit -m “commit message”
```

Add all changes made to tracked files & commit

```
$ git commit -am “commit message”
```

Stashing

Store modifier & staged changes. To inclued untracked files, add -u flag. For untracked & ingored flies, add -a flag

```
$ git stash
```

As above, but add a comment

```
$ git stash save “comment”
```

Partial stash. Stash just a single file, a collection of files, or individual changes from within files

```
$ git stash -p
```

List all stashes

```
$ git stash list
```

Re-apply the stash without deleting it

```
$ git stash apply
```

Re-apply the stash at index 2, then delete it from the stash list. Omit stash@{n} to pop the most recent stash

```
$ git stash pop stash@{2}
```

Show the diff summary of stash 1. Pass the -p flag to see the full diff

```
$ git stash apply
```

Delete stash at index 1. Omit stash@{n} to delete last stash made

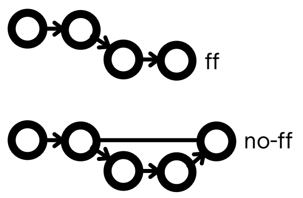
```
$ git stash drop stash@{1}
```

Delete all stashes

```
$ git stash clear
```

Merging

Merging branch A into branch B. Add --no-ff option for no-fast-forward merge



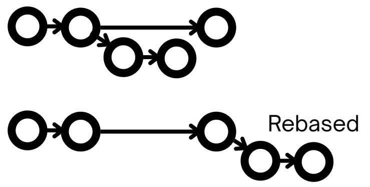
```
$ git checkout <branchB>
$ git merge <branchA>
```

Merge & squash all commits into one new commit

```
$ git commit -m “commit message”
```

Rebasing

Rebase feature branch onto main (to incorporate new changes made to main). Prevents unnecessary merge commits into feature, keeping history clean



```
$ git checkout feature
$ git rebase main
```

Interatively clean up a branches commits before rebasing onto main

```
$ git rebase -i main
```

Interatively rebase the last 3 commits on current branch

```
$ git rebase -i Head~3
```

Undoing Things

Move (&/or rename) a file & stage move

```
$ git mv <existing_path> <new path>
```

Remove a file from working directory & staging area, then stage the removal

```
$ git rm <file>
```

Remove from staging area only

```
$ git rm --cached <file>
```

View a previous commit (READ only)

```
$ git checkout <commit_ID>
```

Create a new commit, reverting the changes from a specified commit

```
$ git revert <commit_ID>
```

Go back to a previous commit & delete all commit ahead of it (revert is safer). Add --hard flag to also delete workspace changes (BE CAREFUL)

```
$ git reset <commit_ID>
```

Review your Repo

List new or modified files not yet committed

```
$ git status
```

List commit history, with respective IDs

```
$ git log --oneline
```

Show changes to unstaged files. For changes to staged files, add --cached option

```
$ git diff
```

Show changes between two commits

```
$ git diff commit1_ID commit2_ID
```

Synchronizing

Add a remote repo

```
$ git remote add <alias> <url>
```

View all remote connections. Add -v flag to view urls

```
$ git remote
```

Remove a connection

```
$ git remote remove <alias>
```

Rename a connection

```
$ git remote rename <old> <new>
```

Fetch all branches from remote repo

```
$ git fetch <alias>
```

Fetch a specific branch

```
$ git fetch <alias> <branch>
```

Fetch the remote repo’s copy of the current branch, then merge

```
$ git pull
```

Move (rebase) your local changes onto the top of new changes made to the remote repo (for clean, linear history)

```
$ git pull --rebase <alias>
```

Upload local content to remote repo

```
$ git push <alias>
```

Upload to a branch

```
$ git push <alias> <branch>
```

Submodules

Add a submodule

```
$ git submodule add <url> <path>
```

Init submodule after clone

```
$ git submodule update --inti
–recursive
```

Update submodules to latest remote

```
$ git submodule update --remote
--merge
```

Pull repo + submodules in one go

```
$ git pull –recurse-submodule
```

Push including submodules

```
$ git push --recurse-submodules=on-
demand
```

Remove a submodule

```
$ git submodule deinit -f <path>
$ rm -rf .git/modules/<path>
$ git rm -f <path>
```