

Git Cheat Sheet

Setup

Set the name and email that will be attached to your commits and tags

```
$ git config --global user.name "Danny Adams"
$ git config --global user.email "my-email@gmail.com"
```

Start a Project

Create a local repo (omit <directory> to initialise the current directory as a git repo)

```
$ git init <directory>

Download a remote repo

$ git clone <url>
```

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"
```

Basic Concepts

main: default development branch
origin: default upstream repo
HEAD: current branch
HEAD^: parent of HEAD
HEAD~4: great-great grandparent of HEAD

By @DoableDanny

Branches

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

```
$ git branch

Create a new branch

$ git branch <new-branch>

Switch to a branch & update 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

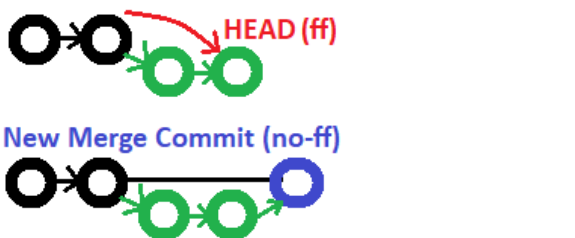
$ git branch -D <branch>

Add a tag to current commit (often used for new version releases)

$ git tag <tag-name>
```

Merging

Merge branch a into branch b. Add --no-ff option for no-fast-forward merge



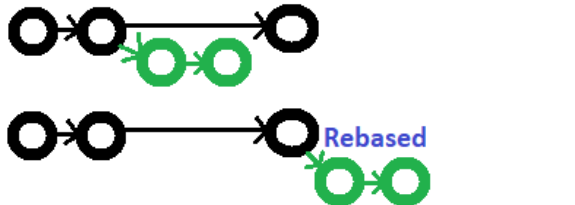
```
$ git checkout b
$ git merge a

Merge & squash all commits into one new commit

$ git merge --squash a
```

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

Iteratively clean up a branches commits before rebasing onto main

$ git rebase -i main

Iteratively 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 commits ahead of it (revert is safer). Add --hard flag to also delete workspace changes (BE VERY 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
```

Stashing

Store modified & staged changes. To include untracked files, add -u flag. For untracked & ignored files, 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 show stash@{1}
```

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

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

Delete all stashes

$ git stash clear
```

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 (no merge)

$ 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 (can then pull request)

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