Git/GitHub Cheatsheet

Help

```
$ git --help
```

In-depth help for all git commands, up to date with your version. You can add ”–help” in front of any command to get the help page for the command.

Process

1. Clone repository
2. Fetch remote branches
3. Stage changes
4. Pull changes (if remote repo is shared)
5. Commit staged changes
6. Push commit to remote

Terminology

Directory

The equivalent of a ”folder” on your computer. Directory is the term we use whenever we discuss Unix-related things.

Repository

A container where git stores files and file histories. Also called ”repo” for short.

Branch

A stored version of your repository code. A repository can have as many branches as need be.

Remote

A repository on another computer (usually a server). Remotes usually act as backup repos. GitHub is a remote repository host!

Staging

Preparing which files you want to save. See the ”Stage a commit” section.

Commit

A saved state of all currently staged files. Commits are used to track different states your code is in throughout a project's lifetime.

Push

Saving commit(s) in a remote repository.

Merge Conflicts

Conflicting changes between two files that are meant to be the same one. If two people make different edits to the same file, it usually results in a merge conflict.

Merge

A method of combining conflicting edits in two versions of code. There are many merge algorithms, many of which attempt to combine code automatically (GitHub uses merge-ort by default). But manual intervention is sometimes needed.

*Pull Request

A repository change request. Pull requests are an exclusively GitHub term, for whenever someone may want to request for changes in a repo without needing push access.

Good Practices

Commit often

Git works wonderfully as a library for your code history. The more commits you make, the easier it will be to find or revert back to a previous commit.

Write a .gitignore

gitignore files are what git uses to prevent files from being staged. They prevent git from staging files defined in the file.

*Setup branch permissions

Branch permissions prevent erroneous pushes to branches within GitHub. Bad pushes can often mess up team development on a project, and is always discouraged. Since most repos regard master/main to be their central branch for a repo, it is also the best candidate for branch protections.

Write a README.md file

A README should detail the contents of your repository. A good README should make your code easy to follow, and easy to get working/deploy.

Commands

Clone

Clone a repository into a new directory.

```
$ git clone --help
$ git clone <ssh-url>
```

Initialize

Start terminal in current tag.

```
$ git init
```

Fetch

Fetches branch from a remote.

```
$ git fetch --help
$ git fetch
$ git fetch <remote> <branch>:<local branch>
$ git checkout <branch>
$ git checkout -b <name>
$ git switch <branch>
```

Stage

Prepare a file, for staging.

```
$ git add --help
$ git add
$ git add <file>
$ git add <directory>
```

Commit

Create a commit, and attach a commit message.

```
$ git commit --help
$ git commit
$ git commit -m "<message>"
```

Push

Push code to a remote repository.

```
$ git push --help
$ git push
$ git push <remote> <branch>
```

Pull

Pull code from a remote repository

```
$ git pull --help
$ git pull
$ git pull <remote> <branch>
```

Remote

Augment remotes for the current repo.

```
$ git remote --help
$ git remote add <remote>
$ git remote rename <remote>
$ git remote remove <remote>
```

Merge

Combines the contents of two branches/repositories.

```
$ git merge --help
$ git merge
$ git remote <branchA> <branchB>
```

Rebase

Rewrite the git commit history according to your needs. It is often done to combine git history, or to remove commits.

```
$ git rebase --help
$ git rebase
$ git rebase -i <branch>
```

*Terminology specific to GitHub

Emil Kovacev, MakeOpenSource,
https://workshop.makeopensource.org