Git Cheat Sheet

Setup

```
git config --global user.name "<firstname lastname>"
git config --global user.email "<valid-email>"
```

set name and email to be associated with version history

```
git config --global colors.ui auto
```

set automatic command line coloring

```
git config --global alias.<alias-name> "<git-command>"
```

define a shortcut/alias for a git command

```
git config --system core.editor <editor>
```

set default text editor used by commands

```
git config --global --edit
```

open global configuration file in text editor

```
git init
```

initialize an existing directory as git repository

```
git clone <url>
```

retrieve a repository from a hosted location via URL

```
git clone --branch <branch-name> <url>
```

clone a specific branch from the remote repository

Gitignore

```
logs/
.notes
pattern*/
```

save a file with desired patterns as `.gitignore` with either direct string matches or wildcard globs

```
git config --global core.excludesfile <file>
```

system wide ignore pattern for all local repositories

Rewrite history

```
git rebase <branch>
```

apply any commits of current branch ahead of specified one

```
git reset --hard <commit>
```

clear staging area, rewrite working tree from specified commit

Stage & snapshot

```
git status
```

show modified files in cwd, staged for next commit

```
git add <file>
```

add a file as it looks now to your next commit (stage file)

```
git reset <file>
```

unstage a file while retaining the changes in cwd

```
git diff
```

show diff between cwd and staging area

```
git diff --staged
```

show diff between staged changes and last commit

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

commit your staged content as a new commit snapshot

```
git commit --amend
```

replace the last commit with the staged changes and last commit combined. use with nothing staged to edit the last commit's message

```
git clean -f
```

remove unversioned files

Working with branches

```
git branch
```

list your branches. a * marks the currently active branch

```
git branch <branch-name>
```

create a new branch at the current commit

```
git checkout
```

switch to another existing branch and check it out into cwd

```
git checkout -b <branch-name>
```

create and check out a new branch with branch-name

```
git merge <branch-name>
```

merge the specified branch's history into the current one

```
git branch-d <branch-name>
```

delete the specified branch

Temporary commits / stashes

Temporarily store modified, tracked files in order to change branches

```
git stash
```

saved modified and staged changes

```
git stash list
```

list stack-order of stashed file changes

```
git stash pop
```

apply and remove most recent stash from stash stack

```
git stash drop
```

discard most recent stash of stash stack

Path changes

```
git rm <file>
```

delete the file from project and stage the removal for commit

```
git mv <existing path> <new path>
```

change an existing file path and stage the move

```
git log --stat -M
```

show all commit logs with indication of any paths that moved

Inspect & compare

```
git log
```

show commit history for currently active branch

```
git log branchB..branchA
```

show the commits on branchA that are not on branchB

```
git log --follow <file>
```

list version history for a file, including renames

```
git diff <first-branch>..<second-branch>
```

show the diff between two branches

```
git show <commit>
```

show metadata and content changes of the specified commit

```
git blame <file>
```

show who edited each line of a file

Remote repositories

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

add a git URL as an alias

```
git fetch <alias>
```

fetch down all the branches from that git remote

```
git merge <alias>/<branch>
```

merge a remote branch into your current branch to bring it up to date

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

transmit local branch commits to the remote repository branch

```
git pull
```

fetch and merge any commits from the tracking remote branch

Glossary

`HEAD:` representing your cwd, the HEAD pointer can be moved to different branches, tags, or commits when using `git checkout`

`index:` different name for the staging area