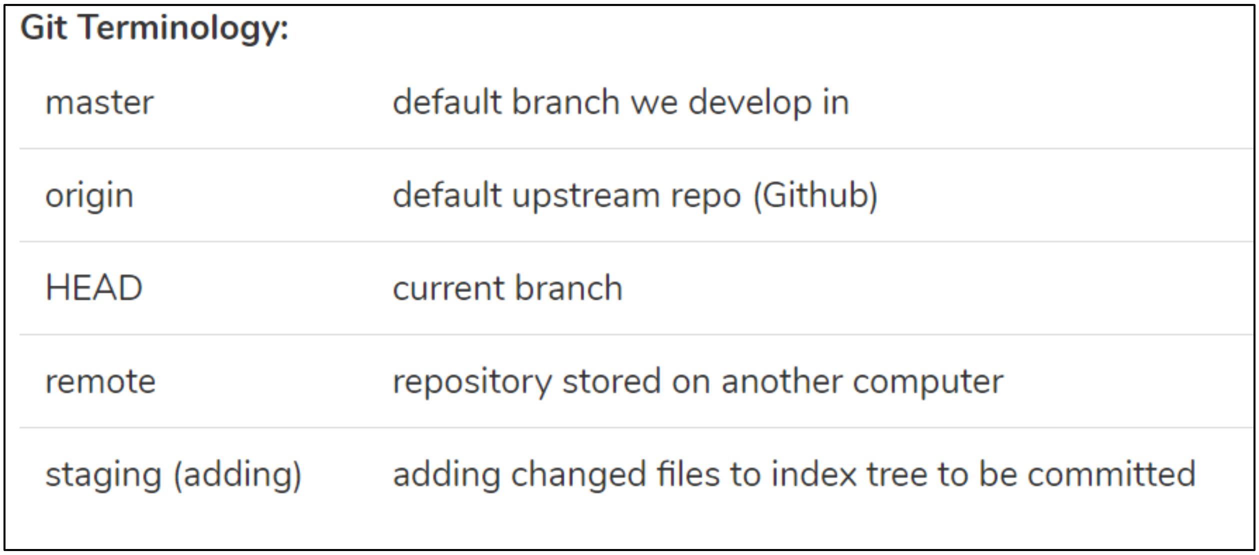
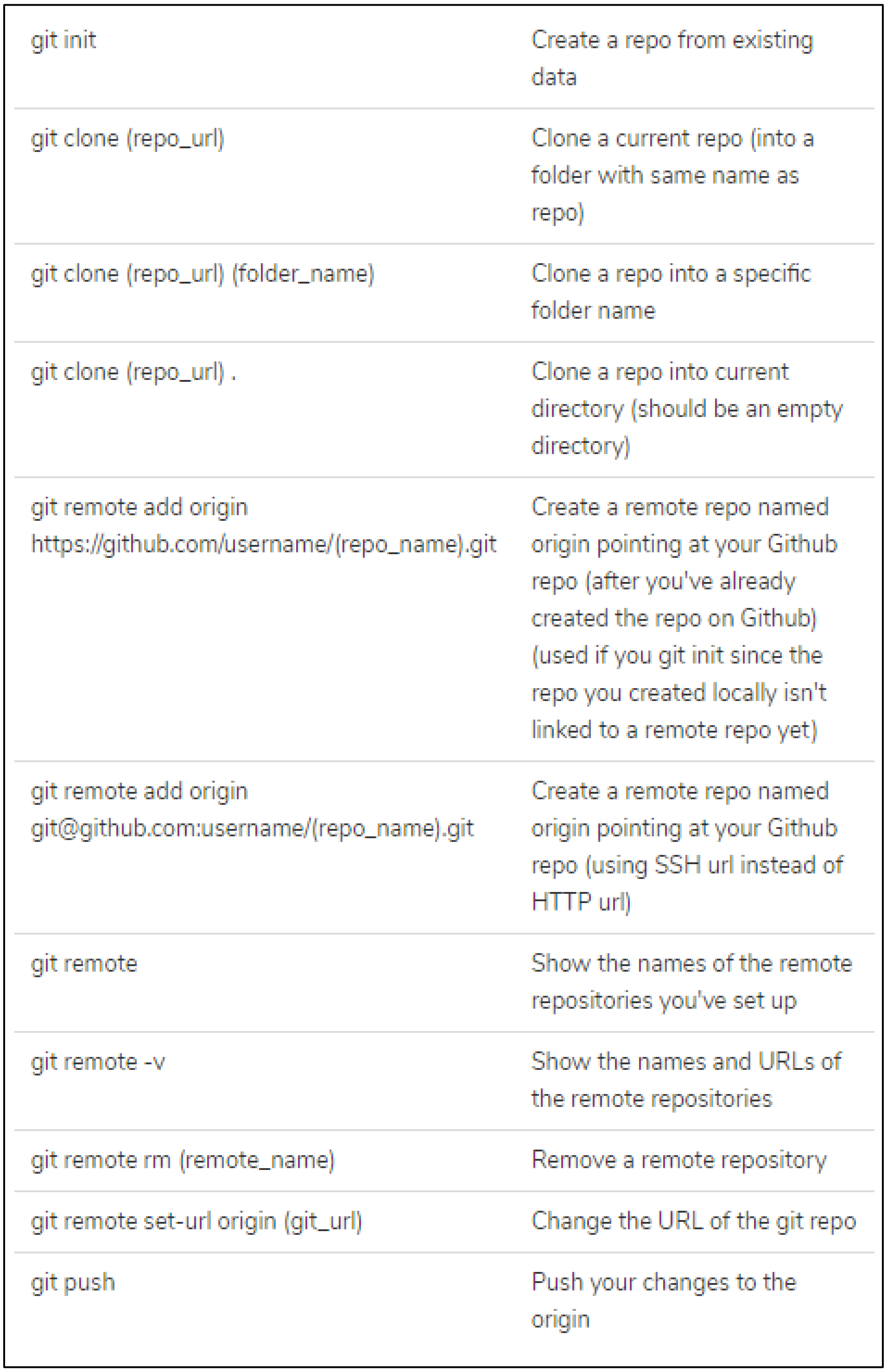
# Git Cheat Sheet



**Git Terminology:**

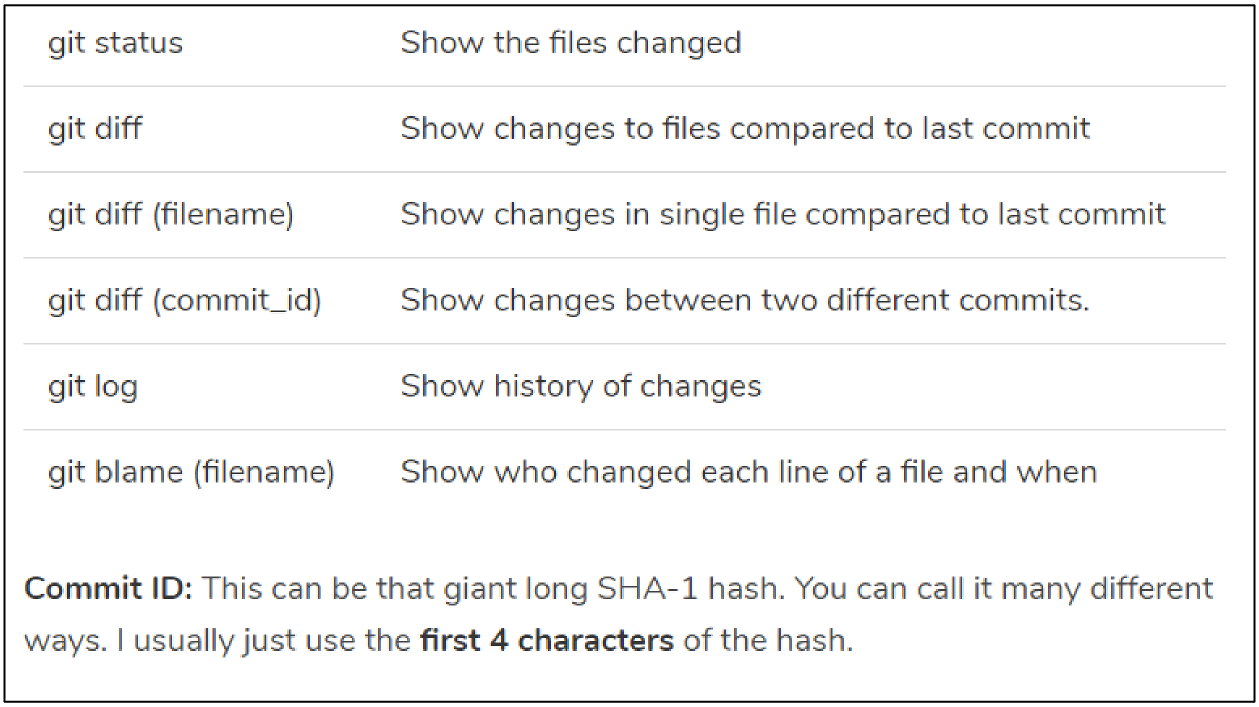
| | |
|---|---|
| master | default branch we develop in |
| origin | default upstream repo (Github) |
| HEAD | current branch |
| remote | repository stored on another computer |
| staging (adding) | adding changed files to index tree to be committed |

## Starting a Repo init / clone / remote



| | |
|---|---|
| git init | Create a repo from existing data |
| git clone (repo_url) | Clone a current repo (into a folder with same name as repo) |
| git clone (repo_url) (folder_name) | Clone a repo into a specific folder name |
| git clone (repo_url) . | Clone a repo into current directory (should be an empty directory) |
| git remote add origin https://github.com/username/(repo_name).git | Create a remote repo named origin pointing at your Github repo (after you've already created the repo on Github) (used if you git init since the repo you created locally isn't linked to a remote repo yet) |
| git remote add origin git@github.com:username/(repo_name).git | Create a remote repo named origin pointing at your Github repo (using SSH url instead of HTTP url) |
| git remote | Show the names of the remote repositories you've set up |
| git remote -v | Show the names and URLs of the remote repositories |
| git remote rm (remote_name) | Remove a remote repository |
| git remote set-url origin (git_url) | Change the URL of the git repo |
| git push | Push your changes to the origin |

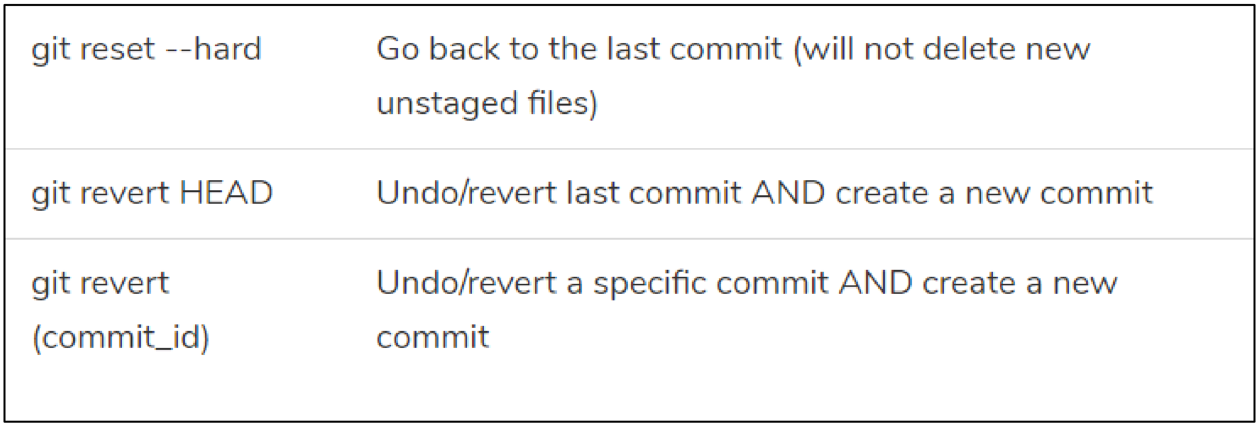
## Showing changes status / different / log / blame



| | |
|---|---|
| git status | Show the files changed |
| git diff | Show changes to files compared to last commit |
| git diff (filename) | Show changes in single file compared to last commit |
| git diff (commit_id) | Show changes between two different commits. |
| git log | Show history of changes |
| git blame (filename) | Show who changed each line of a file and when |

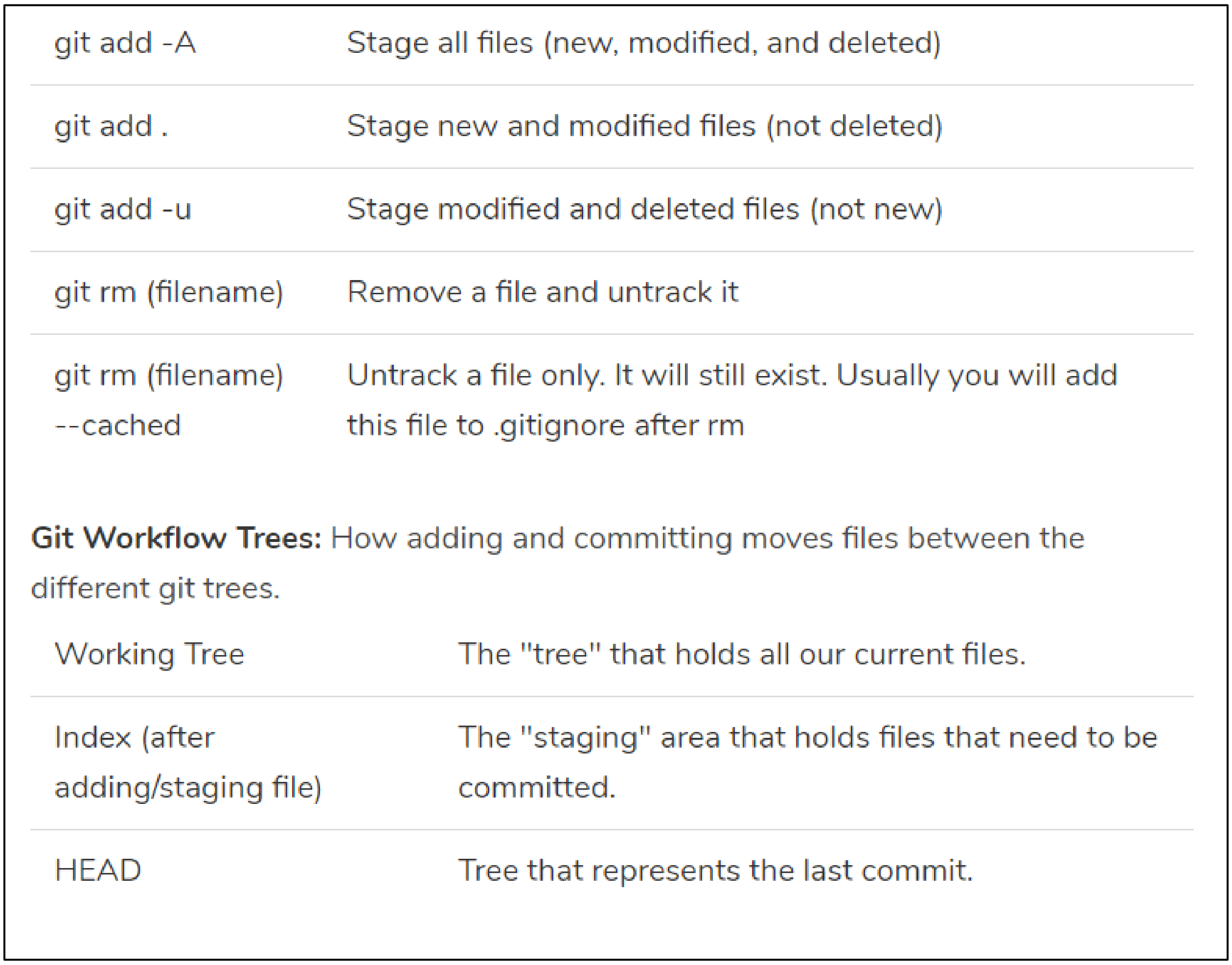
**Commit ID:** This can be that giant long SHA-1 hash. You can call it many different ways. I usually just use the **first 4 characters** of the hash.

## Undoing changes reset / revert



| | |
|---|---|
| git reset --hard | Go back to the last commit (will not delete new unstaged files) |
| git revert HEAD | Undo/revert last commit AND create a new commit |
| git revert (commit_id) | Undo/revert a specific commit AND create a new commit |

## Staging files add / rm

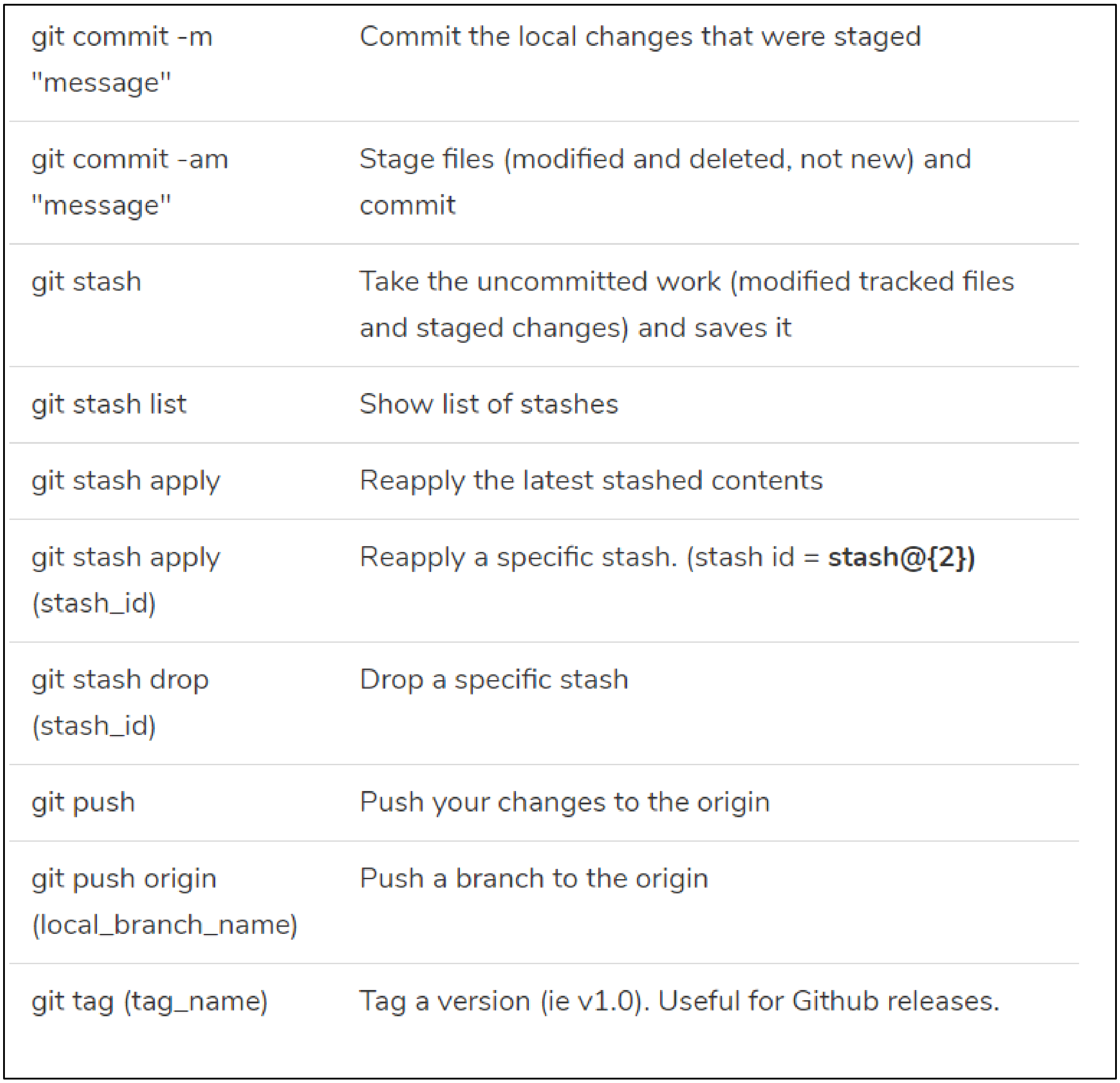


| | |
|---|---|
| git add -A | Stage all files (new, modified, and deleted) |
| git add . | Stage new and modified files (not deleted) |
| git add -u | Stage modified and deleted files (not new) |
| git rm (filename) | Remove a file and untrack it |
| git rm (filename) --cached | Untrack a file only. It will still exist. Usually you will add this file to .gitignore after rm |

**Git Workflow Trees:** How adding and committing moves files between the different git trees.

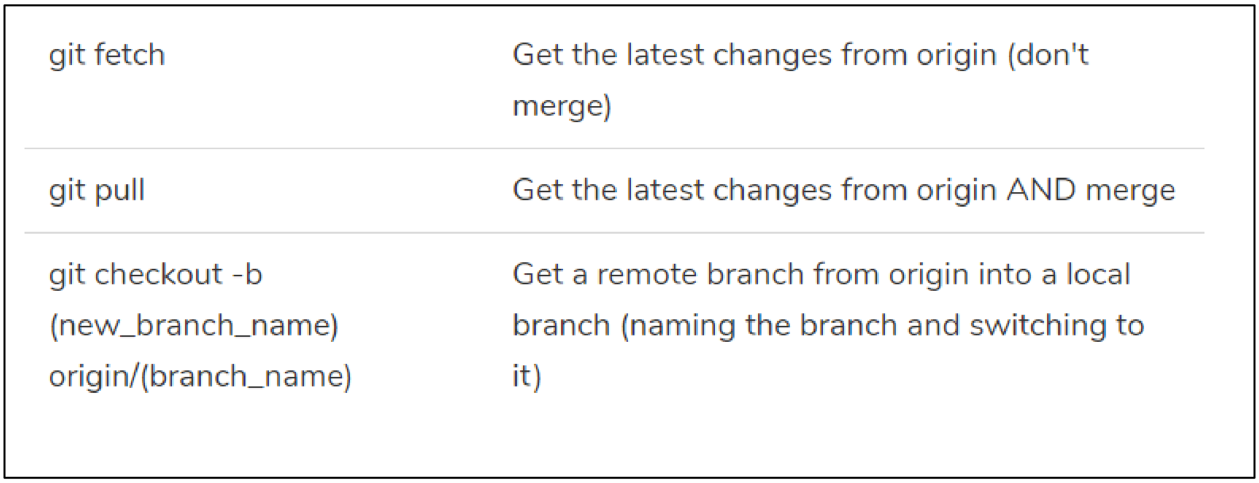
| | |
|---|---|
| Working Tree | The "tree" that holds all our current files. |
| Index (after adding/staging file) | The "staging" area that holds files that need to be committed. |
| HEAD | Tree that represents the last commit. |

## Publishing commit / stash / push



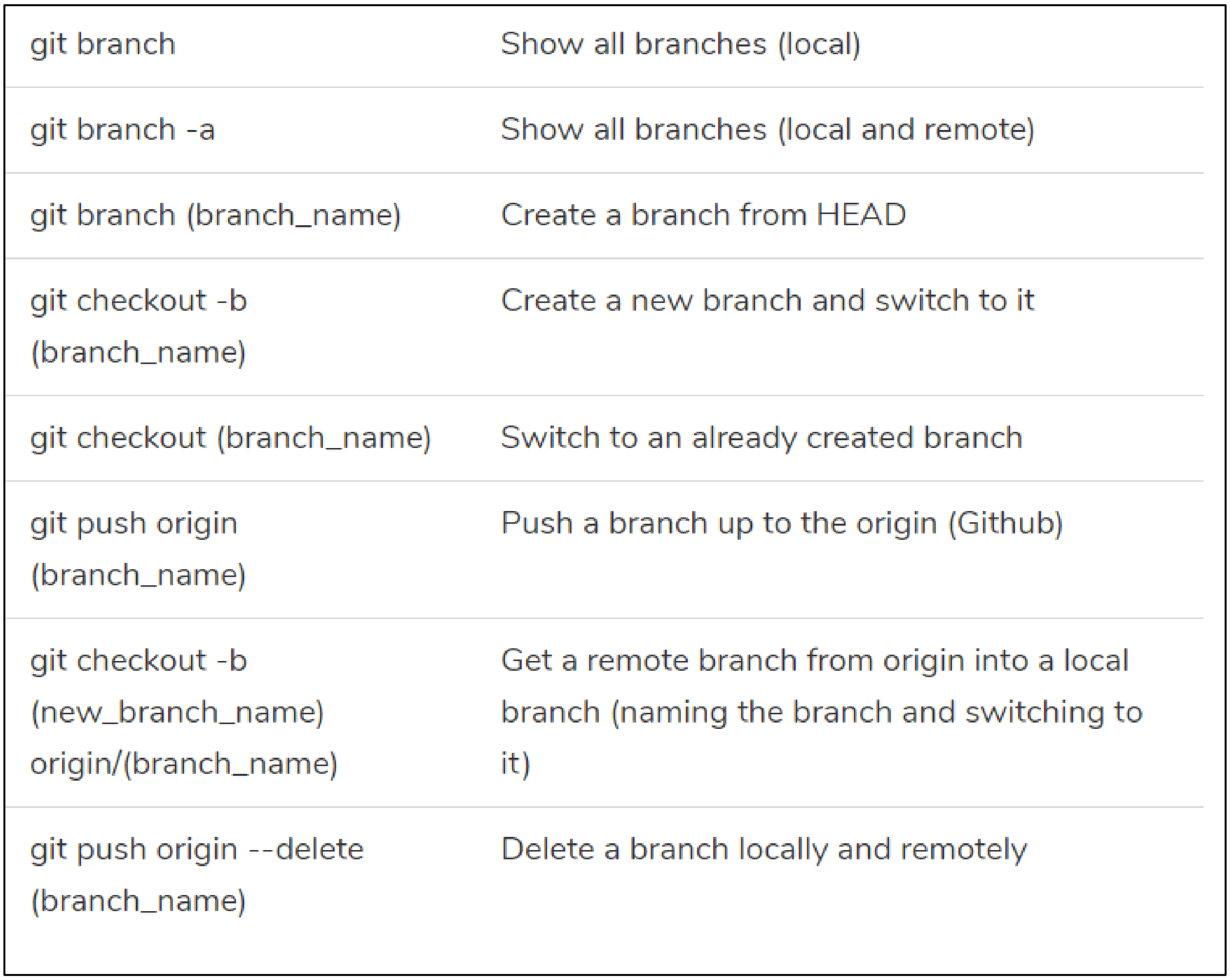
| | |
|---|---|
| git commit -m "message" | Commit the local changes that were staged |
| git commit -am "message" | Stage files (modified and deleted, not new) and commit |
| git stash | Take the uncommitted work (modified tracked files and staged changes) and saves it |
| git stash list | Show list of stashes |
| git stash apply | Reapply the latest stashed contents |
| git stash apply (stash_id) | Reapply a specific stash. (stash id = **stash@{2})** |
| git stash drop (stash_id) | Drop a specific stash |
| git push | Push your changes to the origin |
| git push origin (local_branch_name) | Push a branch to the origin |
| git tag (tag_name) | Tag a version (ie v1.0). Useful for Github releases. |

<u>**Updating and Getting code fetch / pull**</u>



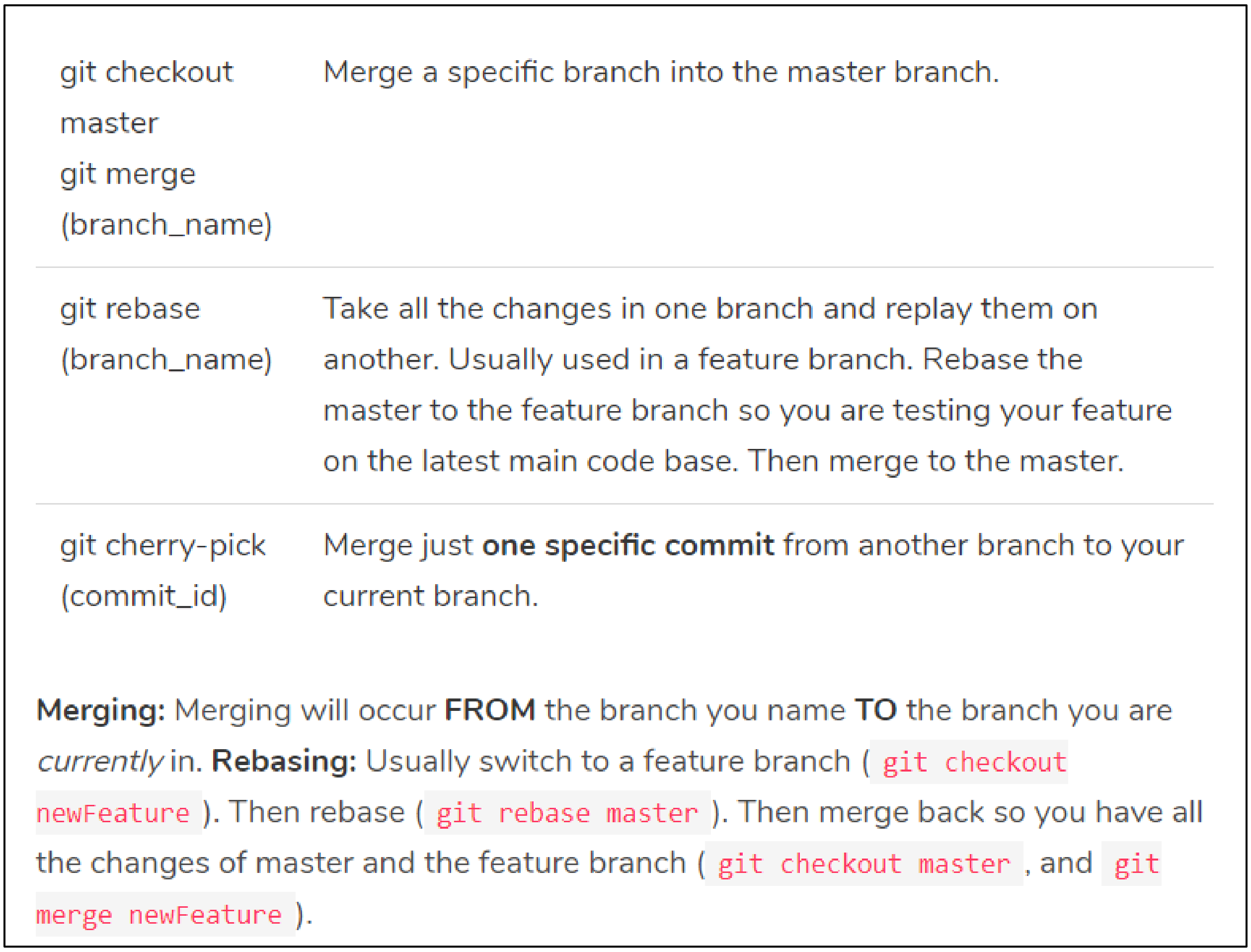
| | |
|---|---|
| git fetch | Get the latest changes from origin (don't merge) |
| git pull | Get the latest changes from origin AND merge |
| git checkout -b (new_branch_name) origin/(branch_name) | Get a remote branch from origin into a local branch (naming the branch and switching to it) |

## Branching branch / checkout



| | |
|---|---|
| git branch | Show all branches (local) |
| git branch -a | Show all branches (local and remote) |
| git branch (branch_name) | Create a branch from HEAD |
| git checkout -b (branch_name) | Create a new branch and switch to it |
| git checkout (branch_name) | Switch to an already created branch |
| git push origin (branch_name) | Push a branch up to the origin (Github) |
| git checkout -b (new_branch_name) origin/(branch_name) | Get a remote branch from origin into a local branch (naming the branch and switching to it) |
| git push origin --delete (branch_name) | Delete a branch locally and remotely |

# Integrating branches merge / rebase



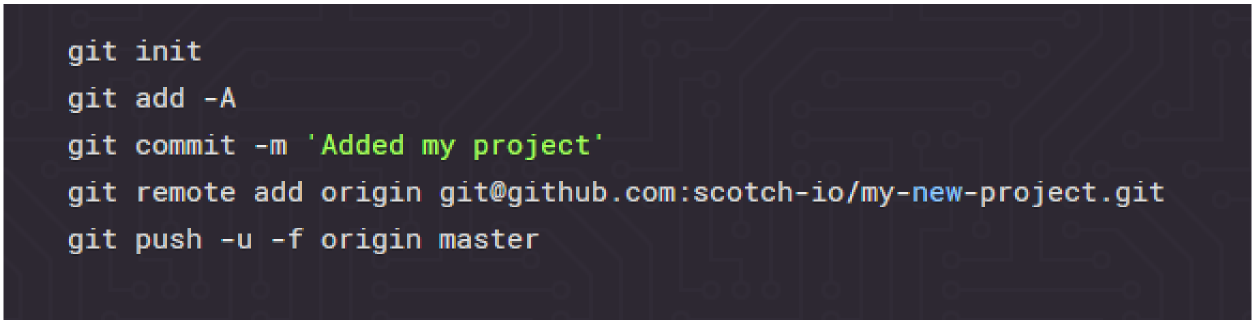
| | |
|---|---|
| git checkout master<br>git merge (branch_name) | Merge a specific branch into the master branch. |
| git rebase (branch_name) | Take all the changes in one branch and replay them on another. Usually used in a feature branch. Rebase the master to the feature branch so you are testing your feature on the latest main code base. Then merge to the master. |
| git cherry-pick (commit_id) | Merge just **one specific commit** from another branch to your current branch. |

**Merging:** Merging will occur **FROM** the branch you name **TO** the branch you are *currently* in. **Rebasing:** Usually switch to a feature branch ( `git checkout newFeature` ). Then rebase ( `git rebase master` ). Then merge back so you have all the changes of master and the feature branch ( `git checkout master` , and `git merge newFeature` ).

## Step 1: Create a Repo on GitHub

## Step 2: Initialize Git in the project folder

- **git init**
- **git add –A (all?*)**
- **git commit –m ‘initial message’**
- **git remote add origin <github:ssh?http>**
- **git push –u –f origin master**
  - **–u : sets the remote origin as the default**
  - **–f : stands for force that will overwrite in remote directory**



```
git init
git add -A
git commit -m 'Added my project'
git remote add origin git@github.com:scotch-io/my-new-project.git
git push -u -f origin master
```