Advanced Git Cheat Sheet

| **Command** | **Explanation & Link** |
| --- | --- |
| git commit -a | [Stages files automatically](https://git-scm.com/docs/git-commit#Documentation/git-commit.txt---all) |
| git log -p | [Produces patch text](https://git-scm.com/docs/git-log#_generating_patch_text_with_p) |
| git show | [Shows various objects](https://git-scm.com/docs/git-show) |
| git diff | [Is similar to the Linux `diff` command, and can show the differences in various commits](https://git-scm.com/docs/git-diff) |
| git diff --staged | [An alias to --cached, this will show all staged files compared to the named commit](https://git-scm.com/docs/git-diff) |
| git add -p | [Allows a user to interactively review patches to add to the current commit](https://git-scm.com/docs/git-add) |
| git mv | [Similar to the Linux `mv` command, this moves a file](https://git-scm.com/docs/git-mv) |
| git rm | [Similar to the Linux `rm` command, this deletes, or removes a file](https://git-scm.com/docs/git-rm) |

There are many useful git cheatsheets online as well. Please take some time to research and study a few, such as [this one](https://github.github.com/training-kit/downloads/github-git-cheat-sheet.pdf).

.gitignore files

.gitignore files are used to tell the git tool to intentionally ignore some files in a given Git repository. For example, this can be useful for configuration files or metadata files that a user may not want to check into the master branch. Check out more at: <https://git-scm.com/docs/gitignore>.

A few common examples of file patterns to exclude can be found [here](https://gist.github.com/octocat/9257657).

# Git Revert Cheat Sheet

[git checkout](https://git-scm.com/docs/git-checkout) is effectively used to switch branches.

[git reset](https://git-scm.com/docs/git-reset#_examples) basically resets the repo, throwing away some changes. It’s somewhat difficult to understand, so reading the examples in the documentation may be a bit more useful.

There are some other useful articles online, which discuss more aggressive approaches to [resetting the repo](https://jwiegley.github.io/git-from-the-bottom-up/3-Reset/4-doing-a-hard-reset.html).

[git commit --amend](https://git-scm.com/docs/git-commit#Documentation/git-commit.txt---amend) is used to make changes to commits after-the-fact, which can be useful for making notes about a given commit.

[git revert](https://git-scm.com/docs/git-revert) makes a new commit which effectively rolls back a previous commit. It’s a bit like an undo command.

There are a [few ways](https://git-scm.com/book/en/v2/Git-Basics-Undoing-Things) you can rollback commits in Git.

There are some interesting considerations about how git object data is stored, such as the usage of sha-1.

Feel free to read more here:

* <https://en.wikipedia.org/wiki/SHA-1>
* <https://github.blog/2017-03-20-sha-1-collision-detection-on-github-com/>

# Git Branches and Merging Cheat Sheet

| **Command** | **Explanation & Link** |
| --- | --- |
| git branch | [Used to manage branches](https://git-scm.com/docs/git-branch) |
| git branch <name> | [Creates the branch](https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging) |
| git branch -d <name> | [Deletes the branch](https://git-scm.com/docs/git-branch#Documentation/git-branch.txt--D) |
| git branch -D <name> | [Forcibly deletes the branch](https://git-scm.com/docs/git-branch#Documentation/git-branch.txt--D) |
| git checkout <branch> | [Switches to a branch.](https://git-scm.com/docs/git-checkout) |
| git checkout -b <branch> | Creates a new branch and [switches to it](https://git-scm.com/docs/git-checkout#Documentation/git-checkout.txt--bltnewbranchgt). |
| git merge <branch> | [Merge joins branches together](https://git-scm.com/docs/git-merge). |
| git merge --abort | If there are merge conflicts (meaning files are incompatible), --abort can be used to abort the merge action. |
| git log --graph --oneline | [This shows a summarized view of the commit history for a repo](https://git-scm.com/book/en/v2/Git-Basics-Viewing-the-Commit-History). |

# Basic Interaction with GitHub Cheat-Sheet

There are various remote repository hosting sites:

* [GitHub](http://github.com/)
* [BitBucket](https://bitbucket.org/product)
* [Gitlab](https://gitlab.com/).

Follow the workflow at <https://github.com/join> to set up a free account, username, and password. After that, [these steps](https://help.github.com/articles/create-a-repo/) will help you create a brand new repository on GitHub.

Some useful commands for getting started:

| **Command** | **Explanation & Link** |
| --- | --- |
| git clone URL | [Git clone is used to clone a remote repository into a local workspace](https://git-scm.com/docs/git-clone) |
| git push | [Git push is used to push commits from your local repo to a remote repo](https://git-scm.com/docs/git-push) |
| git pull | [Git pull is used to fetch the newest updates from a remote repository](https://git-scm.com/docs/git-pull) |

This can be useful for keeping your local workspace up to date.

* <https://help.github.com/en/articles/caching-your-github-password-in-git>
* <https://help.github.com/en/articles/generating-an-ssh-key>

# Git Remotes Cheat-Sheet

| **Command** | **Explanation & Links** |
| --- | --- |
| git remote | [Lists remote repos](https://git-scm.com/docs/git-remote) |
| git remote -v | [List remote repos verbosely](https://git-scm.com/docs/git-remote#Documentation/git-remote.txt--v) |
| git remote show <name> | [Describes a single remote repo](https://git-scm.com/docs/git-remote#Documentation/git-remote.txt-emshowem) |
| git remote update | [Fetches the most up-to-date objects](https://git-scm.com/docs/git-remote#Documentation/git-remote.txt-emupdateem) |
| git fetch | [Downloads specific objects](https://git-scm.com/docs/git-fetch) |
| git branch -r | [Lists remote branches](https://git-scm.com/docs/git-branch#Documentation/git-branch.txt--r); can be combined with other branch arguments to manage remote branches |

# Conflict Resolution Cheat Sheet

Merge conflicts are not uncommon when working in a team of developers, or on Open Source Software. Fortunately, GitHub has some good documentation on how to handle them when they happen:

* <https://help.github.com/en/github/collaborating-with-issues-and-pull-requests/about-merge-conflicts>
* <https://help.github.com/en/github/collaborating-with-issues-and-pull-requests/resolving-a-merge-conflict-using-the-command-line>

You can also use [git rebase branchname](https://git-scm.com/book/en/v2/Git-Branching-Rebasing) to change the base of the current branch to be branchname

The git rebase command is a lot more powerful.  Check out [this link](https://git-scm.com/book/en/v2/Git-Tools-Rewriting-History) for more information.

# Git Fork and Pull Request Cheat Sheet

Check out the following link for more information:

* <https://help.github.com/en/articles/about-pull-request-merges>

# More Information on Code Reviews

Check out the following links for more information:

* <http://google.github.io/styleguide/>
* <https://help.github.com/en/articles/about-pull-request-reviews>
* <https://medium.com/osedea/the-perfect-code-review-process-845e6ba5c31>
* <https://smartbear.com/learn/code-review/what-is-code-review/>

# Additional Tools

Check out the following links for more information:

* <https://arp242.net/diy.html>
* <https://help.github.com/en/articles/closing-issues-using-keywords>
* <https://help.github.com/en/articles/setting-guidelines-for-repository-contributors>
* <https://www.infoworld.com/article/3271126/what-is-cicd-continuous-integration-and-continuous-delivery-explained.html>
* <https://stackify.com/what-is-cicd-whats-important-and-how-to-get-it-right/>
* <https://docs.travis-ci.com/user/tutorial/>
* <https://docs.travis-ci.com/user/build-stages/>