GitHub
GIT CHEAT SHEET

Git is the free and open source distributed version control system that's responsible for everything GitHub related that happens locally on your computer. This cheat sheet features the most important and commonly used Git commands for easy reference.

INSTALLATION & GUIS

With platform specific installers for Git, GitHub also provides the ease of staying up-to-date with the latest releases of the command line tool while providing a graphical user interface for day-to-day interaction, review, and repository synchronization.

GitHub for Windows
https://windows.github.com

GitHub for Mac
https://mac.github.com

For Linux and Solaris platforms, the latest release is available on the official Git web site.

Git for All Platforms
http://git-scm.com

SETUP

Configuring user information used across all local repositories

`git config --global user.name "[firstname lastname]"`
set a name that is identifiable for credit when review version history
`git config --global user.email "[valid-email]"`
set an email address that will be associated with each history marker
`git config --global color.ui auto`
set automatic command line coloring for Git for easy reviewing

SETUP & INIT

Configuring user information, initializing and cloning repositories

`git init`
initialize an existing directory as a Git repository
`git clone [url]`
retrieve an entire repository from a hosted location via URL

STAGE & SNAPSHOT

Working with snapshots and the Git staging area

`git status`
show modified files in working directory, staged for your next commit
`git add [file]`
add a file as it looks now to your next commit (stage)
`git reset [file]`
unstage a file while retaining the changes in working directory
`git diff`
diff of what is changed but not staged
`git diff --staged`
diff of what is staged but not yet committed
`git commit -m "[descriptive message]"`
commit your staged content as a new commit snapshot

BRANCH & MERGE

Isolating work in branches, changing context, and integrating changes

`git branch`
list your branches. a * will appear next to the currently active branch
`git branch [branch-name]`
create a new branch at the current commit
`git checkout`
switch to another branch and check it out into your working directory
`git merge [branch]`
merge the specified branch's history into the current one
`git log`
show all commits in the current branch's history

INSPECT & COMPARE

Examining logs, diffs and object information

`git log`	show the commit history for the currently active branch
`git log branchB..branchA`	show the commits on branchA that are not on branchB
`git log --follow [file]`	show the commits that changed file, even across renames
`git diff branchB...branchA`	show the diff of what is in branchA that is not in branchB
`git show [SHA]`	show any object in Git in human-readable format

SHARE & UPDATE

Retrieving updates from another repository and updating local repos

`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

TRACKING PATH CHANGES

Versioning file removes and 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

REWRITE HISTORY

Rewriting branches, updating commits and clearing 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

IGNORING PATTERNS

Preventing unintentional staging or commiting of files

```
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

TEMPORARY COMMITS

Temporarily store modified, tracked files in order to change branches

`git stash`	Save modified and staged changes
`git stash list`	list stack-order of stashed file changes
`git stash pop`	write working from top of stash stack
`git stash drop`	discard the changes from top of stash stack

GitHub Education

Teach and learn better, together. GitHub is free for students and teachers. Discounts available for other educational uses.

✉ **education@github.com**

education.github.com

GitHub

Git Cheat Sheet

Git is the open source distributed version control system that facilitates GitHub activities on your laptop or desktop. This cheat sheet summarizes commonly used Git command line instructions for quick reference.

Install

GitHub for Windows
https://windows.github.com

GitHub for Mac
https://mac.github.com

Git for All Platforms
http://git-scm.com

Git distributions for Linux and POSIX systems are available on the official Git SCM web site.

Configure tooling

Configure user information for all local repositories

`$ git config --global user.name "[name]"`
Sets the name you want attached to your commit transactions

`$ git config --global user.email "[email address]"`
Sets the email you want attached to your commit transactions

`$ git config --global color.ui auto`
Enables helpful colorization of command line output

Create repositories

When starting out with a new repository, you only need to do it once; either locally, then push to GitHub, or by cloning an existing repository.

`$ git init`
Turn an existing directory into a git repository

`$ git clone [url]`
Clone (download) a repository that already exists on GitHub, including all of the files, branches, and commits

The .gitignore file

Sometimes it may be a good idea to exclude files from being tracked with Git. This is typically done in a special file named `.gitignore`. You can find helpful templates for `.gitignore` files at github.com/github/gitignore.

Branches

Branches are an important part of working with Git. Any commits you make will be made on the branch you're currently "checked out" to. Use `git status` to see which branch that is.

`$ git branch [branch-name]`
Creates a new branch

`$ git checkout [branch-name]`
Switches to the specified branch and updates the working directory

`$ git merge [branch]`
Combines the specified branch's history into the current branch. This is usually done in pull requests, but is an important Git operation.

`$ git branch -d [branch-name]`
Deletes the specified branch

Synchronize changes

Synchronize your local repository with the remote repository on GitHub.com

`$ git fetch`
Downloads all history from the remote tracking branches

`$ git merge`
Combines remote tracking branch into current local branch

`$ git push`
Uploads all local branch commits to GitHub

`$ git pull`
Updates your current local working branch with all new commits from the corresponding remote branch on GitHub. `git pull` is a combination of `git fetch` and `git merge`

GitHub Git Cheat Sheet

Make changes

Browse and inspect the evolution of project files

`$ git log`
Lists version history for the current branch

`$ git log --follow [file]`
Lists version history for a file, including renames

`$ git diff [first-branch]...[second-branch]`
Shows content differences between two branches

`$ git show [commit]`
Outputs metadata and content changes of the specified commit

`$ git add [file]`
Snapshots the file in preparation for versioning

`$ git commit -m "[descriptive message]"`
Records file snapshots permanently in version history

Redo commits

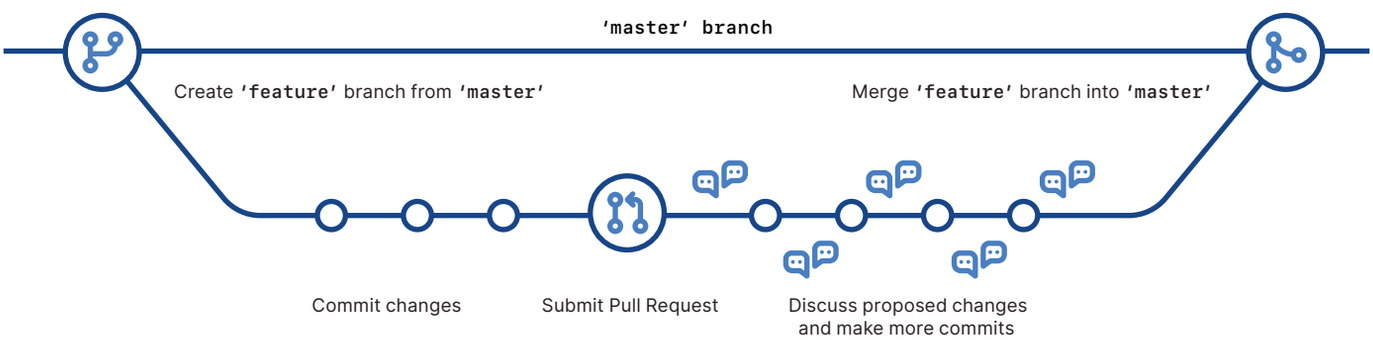
Erase mistakes and craft replacement history

`$ git reset [commit]`
Undoes all commits after [commit], preserving changes locally

`$ git reset --hard [commit]`
Discards all history and changes back to the specified commit

CAUTION! Changing history can have nasty side effects. If you need to change commits that exist on GitHub (the remote), proceed with caution. If you need help, reach out at github.community or contact support.

GitHub Flow



Glossary

`git:` an open source, distributed version-control system
`GitHub:` a platform for hosting and collaborating on Git repositories
`commit:` a Git object, a snapshot of your entire repository compressed into a SHA
`branch:` a lightweight movable pointer to a commit
`clone:` a local version of a repository, including all commits and branches
`remote:` a common repository on GitHub that all team member use to exchange their changes
`fork:` a copy of a repository on GitHub owned by a different user
`pull request:` a place to compare and discuss the differences introduced on a branch with reviews, comments, integrated tests, and more
`HEAD:` representing your current working directory, the HEAD pointer can be moved to different branches, tags, or commits when using `git checkout`

GitHub Training

Want to learn more about using GitHub and Git?
Email the Training Team or visit our web site for learning event schedules and private class availability.

services@github.com
services.github.com