Git Bash Cheat Sheet Cheat Sheet

by hol via cheatography.com/204235/cs/43643/

Key Definitions

This cheat sheet contains git-specific terms and jargon. Here's a run-down of all the terms you may encounter:

Basic Definitions

• **Local repo or repository:** A local directory containing code and files for the project

• **Remote repository:** An online version of the local repository hosted on services like GitHub and GitLab

• **Cloning:** The act of making a clone or copy of a repository in a new director¸

• **Commit:** A snapshot/save of the project you can come back to

• **Branch:** A copy of the project used for working in an isolated environment without affecting the main project

• **Git merge:** The process of combining two branches together

More advanced definitions

`.gitignore` file: A file that lists other files you want git not to track (e.g. data folders, outputs, and any local files that shouldn't be seen by the public.

• **Staging area:** a cache that holds changes you want to commit next

• **Git stash:** another type of cache that holds unwanted changes you may want to come back later

• **HEAD:** a reference name for the latest commit. `HEAD~n` syntax is used to refer to older commits (e.g. `HEAD~2` refers to the second-to-last commit)

Starting a project

`git init [project name]`	Create a new local repository in the current directory names **[project name]**.
`git clone [project ssh url]`	Downloads a project with the entire history using the **SSH URL** from GitLab.

Make a Change

`git status`	Displays the status of your working directory. Options include new, staged, and modified files. It will retrieve branch name, current commit identifier, and changes pending commit.
`git add [file]`	Add a file to the staging area. Use. in place of the full file path to add all changed files from the current directory down into the directory tree.
`git add .`	Stage all files

Make a Change (cont)

`git commit -m " commit messag - e"`	Create a new commit from changes added to the staging area. The commit must have a message!

Branches

`git branch`	List all local branches. Add `-r` flag for all remote branches. `a` flag for all branches.
`git branch new-branch`	Create a new branch called 'new-branch'
`git check out new-branch`	Switch to 'new-branch' branch & update the working directory
`git check out -b new-branch`	Create a new branch 'new-branch- and switch to it
`git check out -D new-branch`	Delete 'new-branch'

Merging

`git checkout b` `git merge a`	Merge branch `a` into branch `b`. Add `--no-ff` option for no fast-forward merge

Undoing Things

`git mv [existing_path] [new_path]`	Move (&/or rename) a file & stage move.
`git rm [file]`	Remove from staging area only.
`git checkout [commi t_ID ]`	View a previous commit (READ only)
`git revert [commi t_ID]`	Create a new commit, reverting the changes from a specified commit
`git reset [commi t_ID]`	Go back to a previous commit & delete all commits ahead of it (revert is safer). Add `--hard` flag to also delete workspace changes (BE CAREFUL)

Cheatography

Git Bash Cheat Sheet Cheat Sheet

by hol via cheatography.com/204235/cs/43643/

Share & Update

`git fetch`	Downloads commits and files from a remote repo into your local repo (no merge).
`git pull`	Downloads content from a remote repo an updates local repo to match content (git fetch + git merge).
`git push`	Upload local content to remote repo.

.gitignore Patterns

`*data/`	Ignore any directory named **data**.
`confidential.txt/`	Ignore file named **confidential.txt**.
`*.csv`	Ignore any file with the format **.csv**
`git config --global core.e xcl ude sfile [fi le]`	System wide ignore pattern for all local repos.

As best practice you should not upload any data or outputs (html, figs) to git. You can find some useful tempates at GitHub .gitignore Templates.

NB: You should not hold any sensitive data on GitLab as they may be accessed by others