Git Cheat Sheet

Definitions

Working directory	The folder containing the files that you work and that are managed by git. The changes made in the working directory will have to be added later on in the index. The working directory can contain files not managed by git, that is, files that are ignored (.gitignore file) or files not added to the index voluntarily.
Index	Staging area where to put changes to be committed to the repository. What is the reason behind this *intermediary* step required before committing? It allows to easily build the *perfect* commit. For instance, you work on a feature, and you find a bug that you feel compelled to correct. You want to commit only the correction of the bug so you add it to the index – without adding anything else – and you commit the correction.
Commit	Full snapshot of the content of your working directory at a certain point in time.
Branch	Branches are just pointers to commits. That pointer moves to the latest commit in that branch as you commit in that branch. The default branch is arbitrarily named *master.* Branching in git is easy and painless. Git is the "all you can branch" buffet of the versioning tools.

Commands

`git init`	**Initializes git repository** Adds a .git directory in the current folder That directory contains all the meta data and data git uses to get the job done. Git does not create directories (cf. *.svn* folders...) in each sub-directories it handles.
`git add <filename>`	Adds the specified file **from the working directory to the git's index**
`git add -p`	Same as previous but will **show and ask confirmation** of each hunks to add When requested for approval, hit ? to get the definition of each suggested action.
`git add .`	**Adds recursively all the file** of a working directory to git's index Will ignore the files matching the rules of the .gitignore file
`git status`	**Shows the content that is about to be committed, from the data in the index.** If some contents have not been added (to the index) yet, a message will be displayed.
`git commit`	**Commits changes that are in the index** Opens text editor in order enter description of the commit
`git commit -m '<commit desc>'`	Same as previous, except it **allows to enter commit message without opening a text editor**
`git diff`	Shows the **differences between the index and the working directory**
`git diff --staged`	Shows the **differences between the last commit and what I'm about to commit** Shows the content in the index that is ready to be committed
`git diff <branch name or SHA1 of a commit> <branch name or SHA1 of a commit>`	**Shows the differences between two arbitrary commits** Commit can be indicated as a branch name (which is a pointer to a commit) or by the commit's 40 characters SHA1.
`git branch`	**Shows all branches contained in the repository** Indicates the current working branch with a * in front
`git branch <branch name>`	**Creates a new branch** with the branch name specified Indicates the current working branch with a * in front
`git branch -v`	**Shows branches and the commits they point to**
`git branch -d <branch name>`	**Deletes the branch specified** Allows to do housekeeping, and deletes a branch that is not needed anymore since we've completed the development for this branch. SUPER SAFE COMMAND: it won't let you lose work. It only allows to delete a branch only if the content of that branch is in the master branch somehow. If it contains uncommitted

	work, it will not be deleted and will suggest to use "-D" if you really want to do that.
`git checkout <branch name>`	**Changes the current working branch to the branch specified** If the specified branch is not pointing at the same commit that the previous branch was pointing to, git will reload the content of the working directory in order to reflect the content of the commit that the specified branch is pointing to. Git will put in the working directory the files contained in the commit BUT will not remove files from the working directory that were not added yet. Tip: To avoid creating a mess, don't switch branches unless "git status" shows no change to be committed or "git stash" your changes before checking out another branch.
`git checkout -b <branch name>`	**Creates a new branch and makes it the current branch**
`git stash`	**Put the content of the working directory aside** in order to allow you to work on something else (e.g. another branch) and come back to it later. More on stash basics at: http://book.git-scm.com/4_stashing.html
`git reset --hard`	**USE WITH CARE** **Delete uncommitted changes**
`git reflog`	**Shows the history , for the last 30 days.**
`git merge <name of branch>`	**Merges the current branch with the specified branch** Changes the content of the working directory to reflect the merge.

Git Basic Steps

1. Make changes to working directory
2. Stage those changes to the index
3. Commit the current state of the index

File tracking

New files must be explicitly added.
.gitignore file can be added at the root of the working directory
E.g.: *.log to ignore .log files

Sources

Mastering Git Basics By Tom Preston-Werner (co-founder of github.com)
http://ontwik.com/git-github/mastering-git-basics-by-tom-preston-werner/

git
http://danielmiessler.com/study/git

Tech Talk: Linus Torvalds on git (git creator)
http://www.youtube.com/watch?v=4XpnKHJAok8