Cheat sheet

diff and patch Cheat Sheet

diff

diff is used to find differences between two files. On its own, it's a bit hard to use; instead, use it with diff -u to find lines which differ in two files

diff -u

diff -u is used to compare two files, line by line, and have the differing lines compared side-by-side in the same output.

Patch

Patch is useful for applying file differences. See the below example, which compares two files. The comparison is saved as a .diff file, which is then patched to the original file!

There are some other interesting patch and diff commands such as patch -p1, diff -r !

Check them out in the following references:

- http://man7.org/linux/man-pages/man1/diff.1.html
- http://man7.org/linux/man-pages/man1/patch.1.html

Initial Git Cheat Sheet

The Linux kernel documentation itself, as well as impassioned opinions from other developers.

You can check out "Setting your email in Git" and "Keeping your email address private" on the GitHub help site for how to do this.

Command	**Explanation & Link**
git commit -a	Stages files automatically
git log -p	Produces patch text
git show	Shows various objects
git diff	Is similar to the Linux `diff` command, and can show the differences in various commits
git diff --staged	An alias to --cached, this will show all staged files compared to the named commit
git add -p	Allows a user to interactively review patches to add to the current commit

git mv	Similar to the Linux `mv` command, this moves a file
git rm	Similar to the Linux `rm` command, this deletes, or removes a file

There are many useful git cheat sheets online as well. Please take some time to research and study a few, such as this one
https://training.github.com/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 is effectively used to switch branches.

git reset 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.

git commit --amend is used to make changes to commits after-the-fact, which can be useful for making notes about a given commit.

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 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
git branch <name>	Creates the branch
git branch -d <name>	Deletes the branch
git branch -D <name>	Forcibly deletes the branch
git checkout <branch>	Switches to a branch.
git checkout -b <branch>	Creates a new branch and switches to it.
git merge <branch>	Merge joins branches together.
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.

Git status: This displays paths that have differences between the index file and the current HEAD commit; paths that have differences between the working tree and the index file; and paths in the working tree that are not tracked by Git. You can view the status of the working tree using the command: [git status]

Git log: This lists the commits done in the repository in reverse chronological order; that is, the most recent commits show up first. This command lists each commit with its SHA-1 checksum, the author's name and email, date, and the commit message.

Git branch: Branches are a part of the everyday development process on the master branch. Git branches effectively function as a pointer to a snapshot of your changes. When you want to add a new feature or fix a bug, no matter how big or small, you spawn a new branch to encapsulate your changes. This makes it difficult for unstable code to get merged into the main codebase.