1 Help

`git help <command>`	Get help for a specific git command

2 Configuration

`git config --list` See current settings
`git config --global user.name "<name>"` Set user name
`git config --global user.email "<email>"` Set email

Use `--global` to set the configuration for all projects. If `git config` is used without `--global` and run inside a project directory, the settings are set for the specific project.

3 Stuff

`git config core.filemode false` Make git ignore file modes

This option is useful if the file permissions are not important to us, for example when we are on Windows.

`git init` Initialize a git repository for existing code
`git clone https://github.com/user/repository.git` Clone remote repo in new directory with same name as repository (you can then change this to whatever you want).
`git clone https://github.com/user/repository.git .` Current directory

4 Viewing and merging with remotes

`git remote -v` View remote urls
`git remote set-url origin http//github.com/repo.git` Change origin url
`git remote add remote-name https://github.com/user/repo.git` Add remote
`git pull origin master` Update and merge your current branch with a remote; origin is the remote repository, and master the remote branch. (If you don't want to merge your changes, use git fetch)

5 Local files and changes

`git diff`	See non-staged (non-added) changes to existing files (Note that this does not track new files.)
`git diff --cached`	See staged (added), non-commited changes
`git diff origin/master`	See differences between local changes and master (Note that origin/master is one local branch, a shorthand for refs/remotes/origin/mast which is the full name of the remote-tracking branch.)
`git diff COMMIT1_ID COMMIT2_ID`	See differences between two commits
`git diff --name-only COMMIT1_ID COMMIT2_ID`	See the files changed between two commits
`git diff-tree --no-commit-id --name-only -r COMMIT_ID`	See the files changed in a specific commit
`git show --pretty="format:" --name-only COMMIT_ID`	See the files changed in a specific commit
`git diff --cached origin/master`	See diff before push
`git show COMMIT_ID`	See details (log message, text diff) of a commit
`git rm removeme.txt tmp/crap.txt`	Remove files
`git mv file_oldname.txt file_newname.txt`	Move files
`git commit --amend -m "New commit message"`	Change message of last commit

6 View local changes

`git log`	Recent commit history
`git log -2`	Commit history for the last two commits
`git log -p -2`	Commit history for the last two commits, with diff
`git log --pretty=oneline`	Commit history printed in single lines

7 Revert one commit, push it

```
git revert dd61ab21
git push origin master
```

8 Revert to the moment before one commit

`git reset 56e05fced`	reset the index to the desired tree
`git reset --soft HEAD@{1}`	move the branch pointer back to the previous HEAD
`git reset --hard`	Update working copy to reflect the new commit

9 Undo last commit

`git reset --soft HEAD~1`	preserving local changes
`git reset --hard HEAD~1`	without preserving local changes
`git reset [--mixed HEAD~1 | HEAD~1]`	preserving local changes in index

10 Undo non-pushed commits

```
git reset origin/master
```

11 Reset to remote state

```
git fetch origin
git reset --hard origin/master
```

12 Branches

`git branch` See local branches
`git branch` -a See all branches
`git checkout master` Create a branch
`git branch new-branch-name` (Here master is the starting point for the new branch. Note that with these 2 commands we don't move to the new branch, as we are still in master and we would need to run git checkout new-branch-name. The same can be achieved using one single command: git checkout -b new-branch-name)
`git checkout new-branch-name` Checkout a branch
`git cherry -v master` See commit history for just the current branch (master is the branch you want to compare)
`git checkout master` Merge branch commits
`git merge branch-name` (Here we are merging all commits of branch-name to master.)
`git merge branch-name --no-commit --no-ff` Merge a branch without committing
`git diff branch-name` See differences between the current state and a branch

`git diff branch-name path/to/file` See differences in a file, between the current state and a branch
`git branch -d new-branch-name` Delete a branch
`git push origin new-branch-name` Push the new branch
`git fetch origin` Get all branches

13 Patches

`git diff > patch-issue-1.patch`	Make some changes, create a patch
`git add newfile; git diff --staged > patch-issue-2.patch`	Add a file and create a patch
`git add newfile`	Add a file, make some changes, and create a patch
`git diff HEAD > patch-issue-2.patch`	
`git format-patch COMMIT_ID`	Make a patch for a commit
`git format-patch HEAD 2`	Make patches for the last two commits
`git format-patch origin/master`	Make patches for all non-pushed commits
`git format-patch --binary --full-index origin/master`	Create patches that contain binary content
`git apply -v patch-name.patch`	Apply a patch
`git am patch1.patch`	Apply a patch created using format-patch

14 Tags

`git tag 7.x-1.3` Create a tag Remove from repository all locally deleted files
`git push origin 7.x-1.3` Push a tag

15 Other/Unorganized

`git rev-parse --show-toplevel`	Get the git root directory
`git rm $(git ls-files --deleted)`	Remove from repository all locally deleted files
`git clean -f`	Delete all untracked files Including directories:
`git clean -f -d`	Preventing sudden cardiac arrest:
`git clean -n -f -d`	
`Show total file size difference between two commits`	Short answer: Git does not do that. Long answer: See http://stackoverflow.com/a/10847242/1391963
`git reset HEAD file.txt`	Unstage (undo add) files:
`git describe --tags ‘git rev-list --tags --max-count=1‘`	See closest tag
`screen`	Have git pull running every X seconds, with GNU Screen
`for((i=1;i<=10000;i+=1)); do sleep 30 && git pull; done`	Use Ctrl+a Ctrl+d to detach the screen.
`history | grep git`	See previous git commands executed
`grep ’ˆgit’ /root/.bash_history`	See previous git commands executed See recently used branches (i.e. branches ordered by most recent commit)

`git for-each-ref --sort=-committerdate refs/heads/ | head`	Tar project files, excluding .git directory
`cd ..`	
`tar cJf project.tar.xz project/ --exclude-vcs`	Tar all locally modified files
`git diff --name-only | xargs tar -cf project.tar -T -`	Look for conflicts in your current files
`grep -H -r "<<<" *`	
`grep -H -r ">>>" *`	
`grep -H -r '^=======$' *`	There's also git-grep.
`patch < file.patch`	Apply a patch not using git