clean

Cleans the working tree by recursively removing files that are not under version control, starting from the current directory.

status

Displays paths that have differences between the index file and the current HEAD commit, paths that have differences between the workspace and the index file, and paths in the workspace that are not tracked by git.

diff

Displays the differences not added to the index.

add <file... or dir...>

Adds the current content of new or modified files to the index, thus staging that content for inclusion in the next commit. Use ‘add --interactive‘ to add the modified contents in the workspace interactively to the index.

add -u

Adds the current content of modified (NOT NEW) files to the index. This is similar to what ’git commit -a’ does in preparation for making a commit.

rm <file(s)...>

Remove a file from the workspace and the index.

mv <file(s)...>

Move file in the workspace and the index.

checkout <files(s)... or dir>

Updates the file or directory in the workspace. Does NOT switch branches.

diff <commit or branch>

View the changes you have in your workspace relative to the named <commit>. You can use HEAD to compare it with the latest commit, or a branch name to compare with the tip of a different branch

commit -a [-m 'msg']

Commit all files changed since your last commit, except untracked files (ie. all files that are already listed in the index). Remove files in the index that have been removed from the workspace.

reset --hard

Matches the workspace and index to the local tree. WARNING: Any changes to tracked files in the working tree since commit are lost. Use this if merging has resulted in conflicts and you'd like to start over. Pass ORIG_HEAD to undo the most recent successful merge and any changes after.

checkout <branch>

Switches branches by updating the index and workspace to reflect the specified branch, <branch>, and updating HEAD to be <branch>.

checkout -b <name of new branch>

Create a branch and switch to it

merge <commit or branch>

Merge changes from <branch name> into current branch.

Use '--no-commit' to leave changes uncommitted.

rebase <upstream>

Reverts all commits since the current branch diverged from <upstream>, and then re-applies them one-by-one on top of changes from the HEAD of <upstream>.

cherry-pick <commit>

Integrate changes in the given commit into the current branch.

revert <commit>

Reverse commit specified by <commit> and commit the result. This requires your working tree to be clean (no modifications from the HEAD commit).

reset HEAD <file(s)...>

Remove the specified files from the next commit. Resets the index but not the working tree (i.e., the changed files are preserved but not marked for commit) and reports what has not been updated.

reset --soft HEAD^

Undo the last commit, leaving changes in the the index.

diff --cached [<commit>]

View the changes you staged vs the latest commit. Can pass a <commit> to see changes relative to it.

commit [-m ’msg’]

Stores the current contents of the index in a new commit along with a log message from the user describing the changes.

commit --amend

Modify the last commit with the current index changes.

log

Show recent commits, most recent on top. Options:

‘--decorate‘ with branch and tag names on appropriate commits

‘--stat‘ with stats (files changed, insertions, and deletions)

‘--author=<author>‘ only by a certain author

‘--after="MMM DD YYYY"‘ ex. ("Jun 20 2008") only commits after a certain date

‘--before="MMM DD YYYY"‘ only commits that occur before a certain date

‘--merge‘ only the commits involved in the current merge conflicts

diff <commit> <commit>

View the changes between two arbitrary commits

branch

List all existing branches. Option -r causes the remote-tracking branches to be listed, and option -a shows both.

branch -d <branch>

Delete an specified branch. Use -D to force.

stash list

List the stashes that you currently have.

stash show [<stash>]

Show the changes recorded in the stash as a diff between the stashed state and its original parent. When no <stash> is given, shows the latest one.

stash drop [<stash>]

Remove a single stashed state from the stash list. When no <stash> is given, it removes the latest one.

stash clear

Remove all the stashed states. Note that those states will then be subject to pruning, and may be impossible to recover.

stash save [<msg>]

Save your local modifications to a new stash, and run git reset --hard to revert them. The <msg> part is optional and gives the description along with the stashed state. For quickly making a snapshot, you can omit both "save" and <msg>.

stash apply [<stash>]

Move changes from the specified stash into the workspace. The latest stash is the default.

stash pop

Applies the changes from the last (or specified) stash and then removes the given stash.

stash branch <branchname> [<stash>]

Creates and checks out a new branch named <branchname> starting from the commit at which the <stash> was originally created, applies the changes recorded in <stash> to the new working tree and index.

If that succeeds, and <stash> is a reference of the form stash@{<revision>}, it then drops the <stash>. When no <stash> is given, applies the latest one.

This is useful if the branch on which you ran git stash save has changed enough that git stash apply fails due to conflicts. Since the stash is applied on top of the commit that was HEAD at the time git stash was run, it restores the originally stashed state with no conflicts.

clone <repo>

Download the repository specified by <repo> and checkout HEAD of the master branch.

pull <remote> <refspec>

Incorporates changes from a remote repository into the current branch. In its default mode, ‘git pull‘ is shorthand for ‘git fetch‘ followed by ‘git merge FETCH_HEAD‘.

reset --hard <remote>/<branch>

Reset local repo and working tree to match a remote branch. Use ‘reset --hard origin/master‘ to throw away all commits to the local master branch. Use this to start over on a failed merge.

branch --track <new> <remote/branch>

Create a new local branch that tracks a remote branch.

fetch <remote> <refspec>

Download objects and refs from another repository.

push

update the server with your commits across all branches that are *COMMON* between your local copy and the server.Local branches that were never pushed to the server in the first place are not shared

push <remote> <branch>

Push new (or existing) branch to remote repository

push <remote> <branch>:<branch>

Push new branch to remote repository with a different name

branch -r

List remote branches

push <remote> :<branch>

Remove a remote branch. Literally "push nothing to this branch"