## Getting and Creating Projects

There are two ways to get a Git repository. One is to copy it from an existing repository on the network or elsewhere and the other is to create a new one in an existing directory.

### git init

To take a directory and turn it into a new Git repository so you can start version controlling it, you can simply run git init.

### git clone

The git clone command is actually something of a wrapper around several other commands. It creates a new directory, goes into it and runs git init to make it an empty Git repository, adds a remote (git remote add) to the URL that you pass it (by default named origin), runs a git fetch from that remote repository and then checks out the latest commit into your working directory with git checkout.

## Basic Snapshotting

For the basic workflow of staging content and committing it to your history, there are only a few basic commands.

### git add

The git add command adds content from the working directory into the staging area (or “index”) for the next commit. When the git commit command is run, by default it only looks at this staging area, so git add is used to craft what exactly you would like your next commit snapshot to look like.

### git status

The git status command will show you the different states of files in your working directory and staging area. Which files are modified and unstaged and which are staged but not yet committed. In its normal form, it also will show you some basic hints on how to move files between these stages.

### git diff

The git diff command is used when you want to see differences between any two trees. This could be the difference between your working environment and your staging area (git diff by itself), between your staging area and your last commit (git diff --staged), or between two commits (git diff master branchB).

### git difftool

The git difftool command simply launches an external tool to show you the difference between two trees in case you want to use something other than the built in git diff command.

### git commit

The git commit command takes all the file contents that have been staged with git add and records a new permanent snapshot in the database and then moves the branch pointer on the current branch up to it.

use the -a flag to skip the git add step in daily workflows and use the -m flag to pass a commit message in on the command line instead of firing up an editor.

### git reset

The git reset command is primarily used to undo things, as you can possibly tell by the verb. It moves around the HEAD pointer and optionally changes the index or staging area and can also optionally change the working directory if you use --hard. This final option makes it possible for this command to lose your work if used incorrectly, so make sure you understand it before using it.