To go to the project directory

Linux

$ cd /home/user/my\_project

MacOS:

$ cd /Users/user/my\_project

Windows:

$ cd C:/Users/user/my\_project

type:

$ git init

to start version-controlling existing files

$ git add \*.c

$ git add LICENSE

$ git commit -m 'Initial project version'

to get a copy of an existing Git repository

git clone.

to clone the repository into a directory named something other than libgit2, you can specify the new directory name as an additional argument:

$ git clone https://github.com/libgit2/libgit2 mylibgit



### Checking the Status of Your Files

to determine which files are in which state

git status

### Tracking New Files

begin tracking a new file

git add.

To begin tracking the README file

$ git add README

### Short Status

To get a far more simplified output

git status -s or git status --short

### Ignoring Files

You can create a file listing patterns to match them named

.gitignore.

### Viewing Your Staged and Unstaged Changes

git diff

// It’s important to notice that

//git diff

// it does not show us the changes did, since your last commit

### Committing Your Changes

$ git commit

### Skipping the Staging Area

To skip the staging area add

-a

option to the

git commit

command do Git automatically

### Removing Files

Type

git rm

### Moving Files

$ git mv file\_from file\_to

### Viewing the Commit History

To get the project, run

$ git clone https://github.com/schacon/simplegit-progit

When you run git log in this project, you should get output that looks something like this:

$ git log

commit ca82a6dff817ec66f44342007202690a93763949

Author: Scott Chacon <schacon@gee-mail.com>

Date: Mon Mar 17 21:52:11 2008 -0700

Change version number

commit 085bb3bcb608e1e8451d4b2432f8ecbe6306e7e7

Author: Scott Chacon <schacon@gee-mail.com>

Date: Sat Mar 15 16:40:33 2008 -0700

Remove unnecessary test

commit a11bef06a3f659402fe7563abf99ad00de2209e6

Author: Scott Chacon <schacon@gee-mail.com>

Date: Sat Mar 15 10:31:28 2008 -0700

Initial commit

to see some abbreviated stats for each commit

--stat

To change the log output to formats other than the default

--pretty.

**How to use output**

%H Commit hash

%h Abbreviated commit hash

%T Tree hash

%t Abbreviated tree hash

%P Parent hashes

%p Abbreviated parent hashes

%an Author name

%ae Author email

%ad Author date (format respects the --date=option)

%ar Author date, relative

%cn Committer name

%ce Committer email

%cd Committer date

%cr Committer date, relative

%s Subject

**Options**

-p Show the patch introduced with each commit.

--stat Show statistics for files modified in each commit.

--shortstat Display only the changed/insertions/deletions line from the --stat command.

--name-only Show the list of files modified after the commit information.

--name-status Show the list of files affected with added/modified/deleted information as well.

--abbrev-commit Show only the first few characters of the SHA-1 checksum instead of all 40.

--relative-date Display the date in a relative format (for example, “2 weeks ago”) instead of using the full date format.

--graph Display an ASCII graph of the branch and merge history beside the log output.

--pretty Show commits in an alternate format. Options include oneline, short, full, fuller, and format (where you specify your own format).

--oneline Shorthand for --pretty=oneline --abbrev-commit used together.

### Limiting Log Output

### To take a number of useful limiting options

git log .

To find the last commit that added or removed a reference to a specific function

$ git log -S function\_name

| [**Options to limit the output of**](https://git-scm.com/book/en/v2/ch00/limit_options)  -<n> Show only the last n commits  --since, --after Limit the commits to those made after the specified date.  --until, --before Limit the commits to those made before the specified date.  --author Only show commits in which the author entry matches the specified string.  --committer Only show commits in which the committer entry matches the specified string.  --grep Only show commits with a commit message containing the string  -S Only show commits adding or removing code matching the string |
| --- |

### Undoing Things

--amend

To take your staging area and uses it for the commit

### Unmodifying a Modified File

git status , Explain you the chanages

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git checkout -- <file>..." to discard changes in working directory)

modified: CONTRIBUTING.md

and you can discard some changes that you did

$ git checkout -- CONTRIBUTING.md

$ git status

On branch master

Changes to be committed:

(use "git reset HEAD <file>..." to unstage)

renamed: README.md -> README

## **Tagging**

// Git has the ability to name a specific points in a the history of a repository. Usually are called .V1.0, v2.0 and so on).

### Listing Your Tags

type

git tag (with optional -l or --list):

$ git tag

v1.0

v2.0

### Creating Tags

There are two type

**lightweight** and **annotated**.

### Annotated Tags

This is easy, just type -a when you run the tag command:

$ git tag -a v1.4 -m "my version 1.4"

$ git tag

v0.1

v1.3

v1.4

**To see the tag data along with the commit that was tagged**

git show

### Lightweight Tags

### just provide a tag name:

$ git tag v1.4-lw

$ git tag

v0.1

v1.3

v1.4

v1.4-lw

v1.5

### Sharing Tags

you can run

git push origin <tagname>.

### Deleting Tags

you can use

git tag -d <tagname>.

### Checking out Tags

you can use

git checkout

Git Basics or Git aliases

// This is to simplify commands when you do not want to writte it all

$ git config --global alias.co checkout

$ git config --global alias.br branch

$ git config --global alias.ci commit

$ git config --global alias.st status