Git - Command Line Guide

Anthony McGlone

May 10, 2022

Table of Contents

Chapter 1

Introduction

1.1 What is Git?

Git is a repository which is used to store files. Git is used by Software Engineers to store and version code for software releases. It's a powerful tool for collaborating on large projects.

Git versions the files within its repository, so every time a commit operation is made, the files in the repository are saved - a unique ID is also created and associated with the commit. This means that you can see the history of the changes in the repository. You can also delete the changes in the repository by going back to a specific commit.

In this guide you will see how the command line tool `git` is used to manage this repository.

1.2 Installing Git

1.2.1 MacOS

1. Open a terminal. Then install Homebrew (steps located: here).
2. Install Git by running `brew install git`
3. Open a terminal and run `git --version` (if Git is installed, the version number should be printed to console)

1.2.2 Windows

1. Install Git by downloading the Windows binary from git-scm (here)
2. Run the installer
3. Open a MS-DOS terminal and run `git --version` (if Git is installed, the version number should be printed to console)

1.2.3 Linux (Ubuntu)

1. Install Git by opening a terminal and running `sudo apt-get install git`
2. Run `git --version` (if Git is installed, the version number should be printed to console)

1.2.4 Linux (Red Hat)

1. Install git by opening a terminal and running `sudo yum install git`
2. Run `git --version` (if Git is installed, the version number should be printed to console)

Chapter 2

Basic Git command line operations

2.1 Setting up a local repository

First create a folder called `store`. Then open a terminal and navigate into that folder. Run this command:

```
git init
```

This command creates the repository and also creates a `.git` subdirectory. This subdirectory stores information about commits, and the location of your remote repository. This remote repository is stored on a server (hosted on the internet or on a company's internal network). To share your commits with others, you have to push your local commits to the remote repository. We'll see how to do this later.

Run the following command to verify that your local repository was set up:

```
git status
```

You should see the following text in your terminal:

```
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

Your repository is now successfully set up. The branch `master` is the main branch in your local repository. By default, it's the place where your files are stored. You can create other branches (basically copies of `master`) and work on edits there before saving them back into the `master` branch. For now, we'll just work with the `master` branch. We'll cover branching strategies and remote branches (stored in a remote repository) later.

2.2 Committing files into the `store` repository

First, create a `code.txt` file in the `store` folder. Then run `git status` in your terminal. You should see the following output:

```
Untracked files:
    (use "git add <file>..." to include in what will be committed)

        code.txt
```

The file is untracked, meaning Git doesn't see it yet. If you tried to do a commit operation now, Git wouldn't save the file or version it. Run the add operation below in your terminal to get Git to track the file.

```
git add code.txt
```