History!

Table of contents

1. Viewing the Commit History

After you have created several commits, or if you have cloned a repository with an existing commit history, you'll probably want to look back to see what has happened.

1.1. git log

The most basic and powerful tool to do this is the **git log** command.

```
git log
```

By default, with no arguments, git log lists the commits made in that repository in reverse chronological order; that is, the most recent commits show up first.

A huge number and variety of options to the git log command are available to show you exactly what you're looking for. Here, we'll show you some of the most popular.

1.2. git log -p

One of the more helpful options is -p or --patch, which shows the difference (the patch output) introduced in each commit.

You can also limit the number of log entries displayed, such as using -2 to show only the last two entries.

```
git log -p -2
```

This option displays the same information but with a diff directly following each entry. This is very helpful for code review or to quickly browse what happened during a series of commits that a

collaborator has added.

1.3. git log -stat

You can also use a series of summarizing options with git log. For example, if you want to see some abbreviated stats for each commit, you can use the --stat option:

```
git log --stat
```

The --stat option prints below each commit entry a list of modified files, how many files were changed, and how many lines in those files were added and removed. It also puts a summary of the information at the end.

1.4. git log --pretty

Another really useful option is --pretty. This option changes the log output to formats other than the default.

```
git log --pretty=oneline
```

The most interesting option value is format, which allows you to specify your own log output format. This is especially useful when you're generating output for machine parsing — because you specify the format explicitly, you know it won't change with updates to Git:

```
git log --pretty=format:"%h - %an, %ar : %s"
git log --pretty=format:"%h %s" --graph
```