Help Topic: The Git Commit Command

Spring 2022

Oscar Viquez, oviquezr@mit.edu
Michael Benjamin, mikerb@mit.edu
Department of Mechanical Engineering
MIT, Cambridge MA 02139

The Git `commit` Command

The `git commit` command records changes from your working copy to the repository. A log message must be provided. The message can be given on the command line, from a file, or an editor may be launched as the commit proceeds. More info on `git commit` can always be found by Googling "Git book" and reading the full PDF online free, or just typing `git help commit` anytime on the command line.

Basic usage of the `git commit` and `git push` commands

The basic syntax for the `git commit` command is as follows.

```
$ git commit -m "brief commit message"
```

The `git commit` command does *not* require a network connection. With this command, the changes previously staged with `git add` and `git rm` are now formally recorded in the local copy of the repository's records. To send the committed changes to the server, an additional command is needed:

```
$ git push
```

The `git push` command *does* require a network connection to the server, and you will be prompted for a password (unless you have something like ssh keys in use).

The `-m "brief message"` option is essentially allowing the commit to proceed without any textual log file comment. This is generally not recommended for large projects, where more detailed commit documentation is often preferred. However, for individual repositories with small incremental changes per commit, the brief messages are often sufficient.

Collecting the commit message from a file

In addition to brief one-line comments, Git supports longer commit for instances where more verbose change tracking is needed. One of the solutions to manage this is to collect the message from a file, using the following command:

```
$ git commit -F <file>
```

Configuring Git with an editor of your choosing

Another approach to handling long commit messages is to prompt the user with a text editor window, where the user can proceed to type the desired details about the commit. By default, Git uses whatever the user has set as the default text editor – this can be done via one of the shell environment variables `VISUAL` or `EDITOR`. Otherwise, Git falls back to the vi editor to create and edit your commit messages. To change that default to something else, you can use the core.editor setting:

```
$ git config --global core.editor emacs
```

If you're a `bash` user, you can set the `EDITOR` environment variable by adding the following lines to your `.bashrc` file, with the editor of your choosing:

```
export EDITOR=emacs
```

To check the prevailing value of an environment variable, e.g., `EDITOR`, just type this on the command line: `echo $EDITOR`.