git --everything-is-local

Search entire site...

- About
  - Branching and Merging
  - Small and Fast
  - Distributed
  - Data Assurance
  - Staging Area
  - Free and Open Source
  - Trademark
- Documentation
  - Reference
  - Book
  - Videos
  - External Links
- Downloads
  - GUI Clients
  - Logos
- Community

This book is available in English.

Full translation available in

български език,
Deutsch,
Español,
Français,
Ελληνικά,
日本語,
한국어,
Nederlands,
Русский,
Slovenščina,
Tagalog,
Українська
简体中文,

Partial translations available in

Čeština,
Македонски,
Polski,
Српски,
Ўзбекча,
繁體中文,

Translations started for

azərbaycan dili,

Беларуская,
فارسی,
Indonesian,
Italiano,
Bahasa Melayu,
Português (Brasil),
Português (Portugal),
Svenska,
Türkçe.

The source of this book is hosted on GitHub.
Patches, suggestions and comments are welcome.

Chapters ▾

2nd Edition

A1.1 Appendix A: Git in Other Environments - Graphical Interfaces

If you read through the whole book, you've learned a lot about how to use Git at the command line. You can work with local files, connect your repository to others over a network, and work effectively with others. But the story doesn't end there; Git is usually used as part of a larger ecosystem, and the terminal isn't always the best way to work with it. Now we'll take a look at some of the other kinds of environments where Git can be useful, and how other applications (including yours) work alongside Git.

Graphical Interfaces

Git's native environment is in the terminal. New features show up there first, and only at the command line is the full power of Git completely at your disposal. But plain text isn't the best choice for all tasks; sometimes a visual representation is what you need, and some users are much more comfortable with a point-and-click interface.

It's important to note that different interfaces are tailored for different workflows. Some clients expose only a carefully curated subset of Git functionality, in order to support a specific way of working that the author considers effective. When viewed in this light, none of these tools can be called "better" than any of the others, they're simply more fit for their intended purpose. Also note that there's nothing these graphical clients can do that the command-line client can't; the command-line is still where you'll have the most power and control when working with your repositories.

`gitk` and `git-gui`

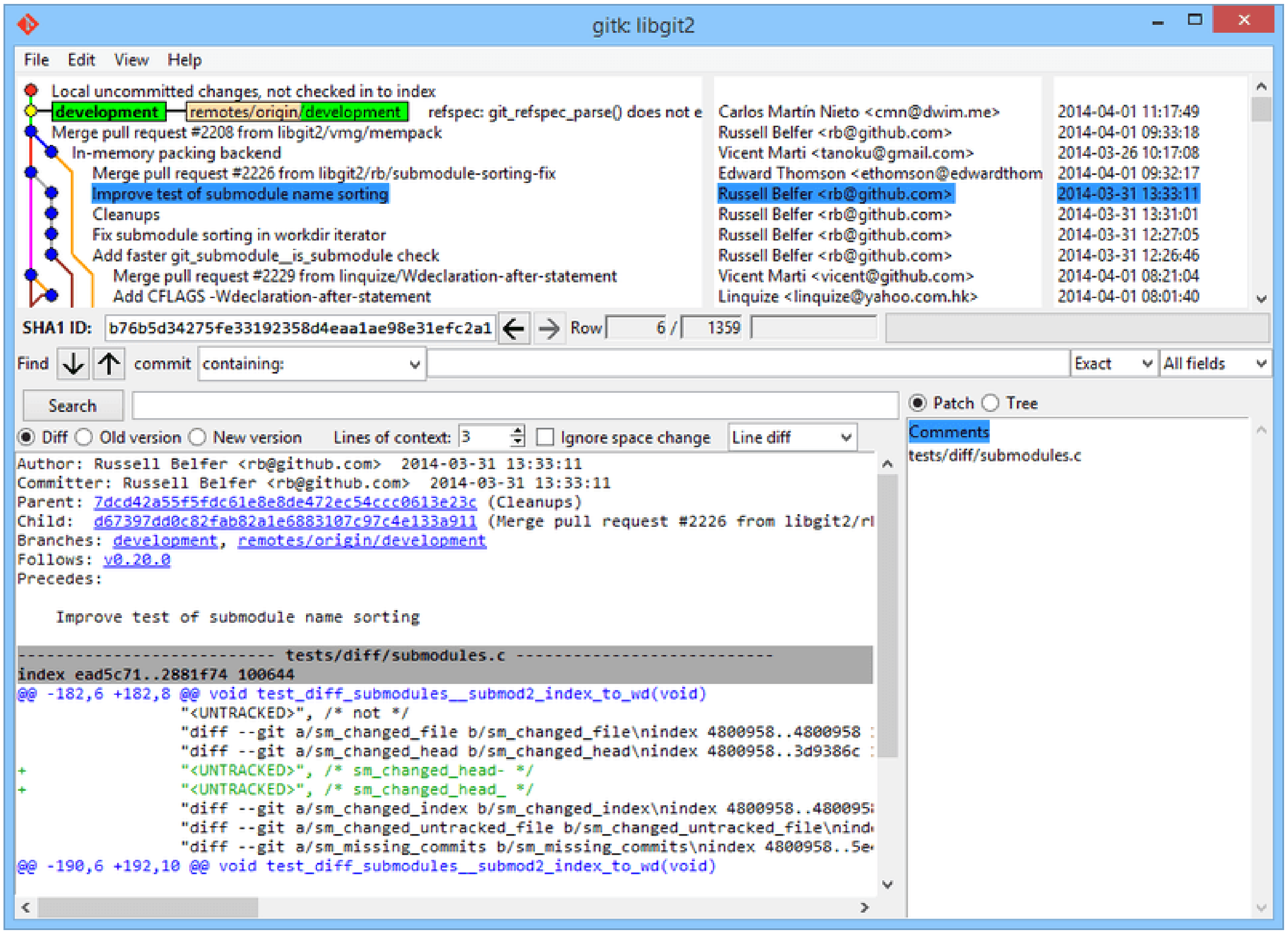
When you install Git, you also get its visual tools, `gitk` and `git-gui`.

`gitk` is a graphical history viewer. Think of it like a powerful GUI shell over `git log` and `git grep`. This is the tool to use when you're trying to find something that happened in the past, or visualize your project's history.

Gitk is easiest to invoke from the command-line. Just `cd` into a Git repository, and type:

```
$ gitk [git log options]
```

Gitk accepts many command-line options, most of which are passed through to the underlying `git log` action. Probably one of the most useful is the `--all` flag, which tells gitk to show commits reachable from *any* ref, not just HEAD. Gitk's interface looks like this:



```
Local uncommitted changes, not checked in to index
development   remotes/origin/development   refspec: git_refspec_parse() does not e   Carlos Martín Nieto <cmn@dwim.me>   2014-04-01 11:17:49
Merge pull request #2208 from libgit2/vmg/mempack   Russell Belfer <rb@github.com>   2014-04-01 09:33:18
  In-memory packing backend   Vicent Marti <tanoku@gmail.com>   2014-03-26 10:17:08
    Merge pull request #2226 from libgit2/rb/submodule-sorting-fix   Edward Thomson <ethomson@edwardthom   2014-04-01 09:32:17
    Improve test of submodule name sorting   Russell Belfer <rb@github.com>   2014-03-31 13:33:11
    Cleanups   Russell Belfer <rb@github.com>   2014-03-31 13:31:01
    Fix submodule sorting in workdir iterator   Russell Belfer <rb@github.com>   2014-03-31 12:27:05
    Add faster git_submodule__is_submodule check   Russell Belfer <rb@github.com>   2014-03-31 12:26:46
      Merge pull request #2229 from linquize/Wdeclaration-after-statement   Vicent Marti <vicent@github.com>   2014-04-01 08:21:04
      Add CFLAGS -Wdeclaration-after-statement   Linquize <linquize@yahoo.com.hk>   2014-04-01 08:01:40

SHA1 ID: b76b5d34275fe33192358d4eaa1ae98e31efc2a1   Row 6 / 1359
Lines of context: 3

tests/diff/submodules.c
```

```
Author: Russell Belfer <rb@github.com>  2014-03-31 13:33:11
Committer: Russell Belfer <rb@github.com>  2014-03-31 13:33:11
Parent: 7dcd42a55f5fdc61e8e8de472ec54ccc0613e23c (Cleanups)
Child:  d67397dd0c82fab82a1e6883107c97c4e133a911 (Merge pull request #2226 from libgit2/rl
Branches: development, remotes/origin/development
Follows: v0.20.0
Precedes:

    Improve test of submodule name sorting

--------------------------- tests/diff/submodules.c ---------------------------
index ead5c71..2881f74 100644
@@ -182,6 +182,8 @@ void test_diff_submodules__submod2_index_to_wd(void)
                 "<UNTRACKED>", /* not */
                 "diff --git a/sm_changed_file b/sm_changed_file\nindex 4800958..4800958 
                 "diff --git a/sm_changed_head b/sm_changed_head\nindex 4800958..3d9386c 
+                "<UNTRACKED>", /* sm_changed_head- */
+                "<UNTRACKED>", /* sm_changed_head_ */
                 "diff --git a/sm_changed_index b/sm_changed_index\nindex 4800958..480095
                 "diff --git a/sm_changed_untracked_file b/sm_changed_untracked_file\nind
                 "diff --git a/sm_missing_commits b/sm_missing_commits\nindex 4800958..5e
@@ -190,6 +192,10 @@ void test_diff_submodules__submod2_index_to_wd(void)
```

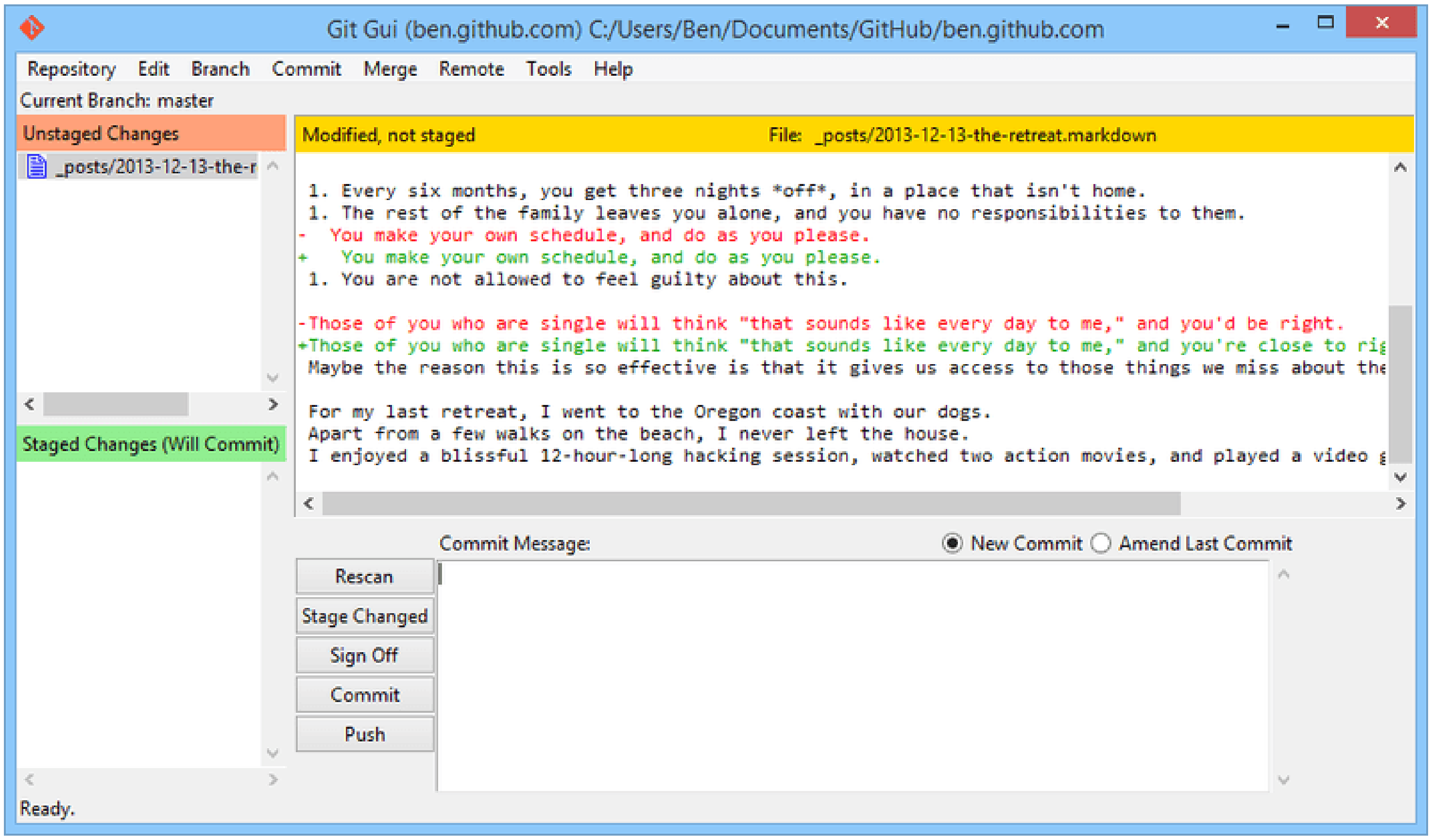
Figure 151. The `gitk` history viewer

On the top is something that looks a bit like the output of `git log --graph`; each dot represents a commit, the lines represent parent relationships, and refs are shown as colored boxes. The yellow dot represents HEAD, and the red dot represents changes that are yet to become a commit. At the bottom is a view of the selected commit; the comments and patch on the left, and a summary view on the right. In between is a collection of controls used for searching history.

`git-gui`, on the other hand, is primarily a tool for crafting commits. It, too, is easiest to invoke from the command line:

```
$ git gui
```

And it looks something like this:


Git Gui (ben.github.com) C:/Users/Ben/Documents/GitHub/ben.github.com
Repository Edit Branch Commit Merge Remote Tools Help
Current Branch: master
Unstaged Changes
_posts/2013-12-13-the-r
Staged Changes (Will Commit)
Modified, not staged
File: _posts/2013-12-13-the-retreat.markdown
Commit Message:
New Commit
Amend Last Commit
Rescan
Stage Changed
Sign Off
Commit
Push
Ready.

```
 1. Every six months, you get three nights *off*, in a place that isn't home.
 1. The rest of the family leaves you alone, and you have no responsibilities to them.
-  You make your own schedule, and do as you please.
+   You make your own schedule, and do as you please.
 1. You are not allowed to feel guilty about this.

-Those of you who are single will think "that sounds like every day to me," and you'd be right.
+Those of you who are single will think "that sounds like every day to me," and you're close to rig
 Maybe the reason this is so effective is that it gives us access to those things we miss about the

 For my last retreat, I went to the Oregon coast with our dogs.
 Apart from a few walks on the beach, I never left the house.
 I enjoyed a blissful 12-hour-long hacking session, watched two action movies, and played a video g
```

Figure 152. The `git-gui` commit tool

On the left is the index; unstaged changes are on top, staged changes on the bottom. You can move entire files between the two states by clicking on their icons, or you can select a file for viewing by clicking on its name.

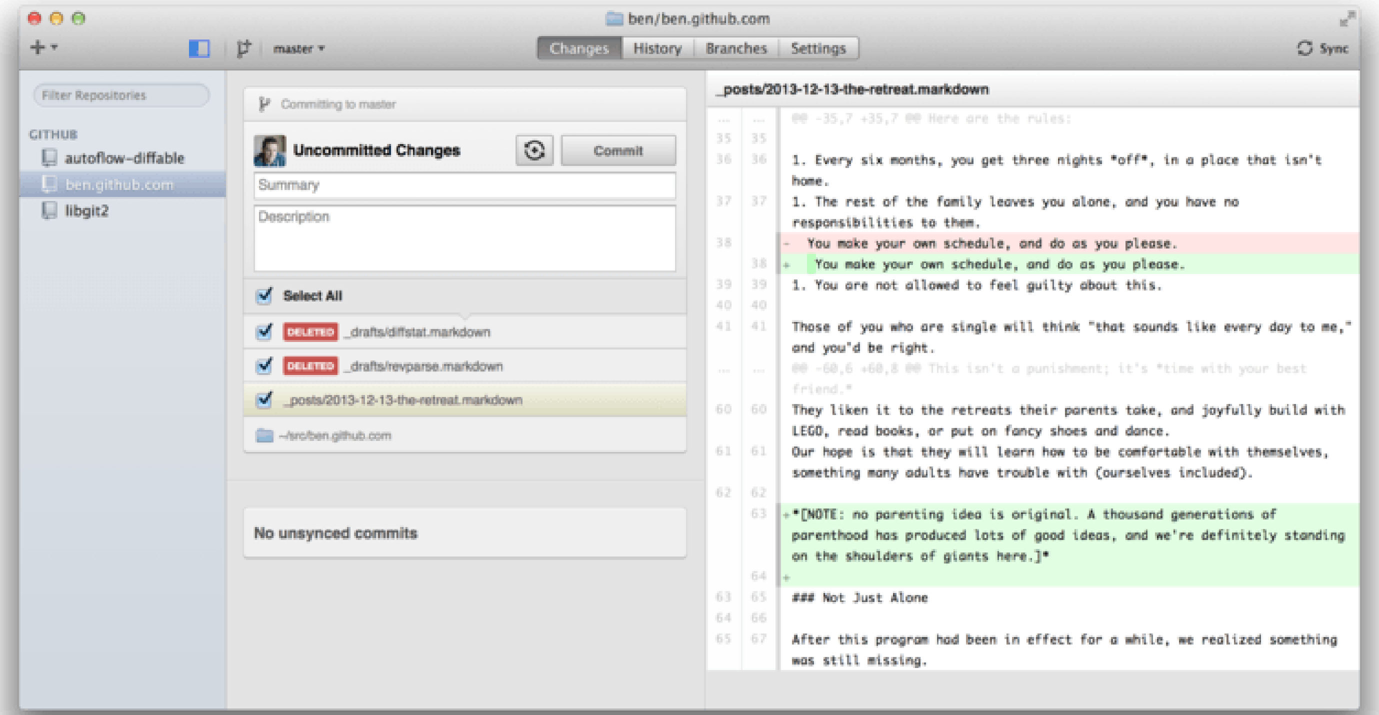
At top right is the diff view, which shows the changes for the currently-selected file. You can stage individual hunks (or individual lines) by right-clicking in this area.

At the bottom right is the message and action area. Type your message into the text box and click “Commit” to do something similar to `git commit`. You can also choose to amend the last commit by choosing the “Amend” radio button, which will update the “Staged Changes” area with the contents of the last commit. Then you can simply stage or unstage some changes, alter the commit message, and click “Commit” again to replace the old commit with a new one.

`gitk` and `git-gui` are examples of task-oriented tools. Each of them is tailored for a specific purpose (viewing history and creating commits, respectively), and omit the features not necessary for that task.

GitHub for macOS and Windows

GitHub has created two workflow-oriented Git clients: one for Windows, and one for macOS. These clients are a good example of workflow-oriented tools – rather than expose *all* of Git’s functionality, they instead focus on a curated set of commonly-used features that work well together. They look like this:

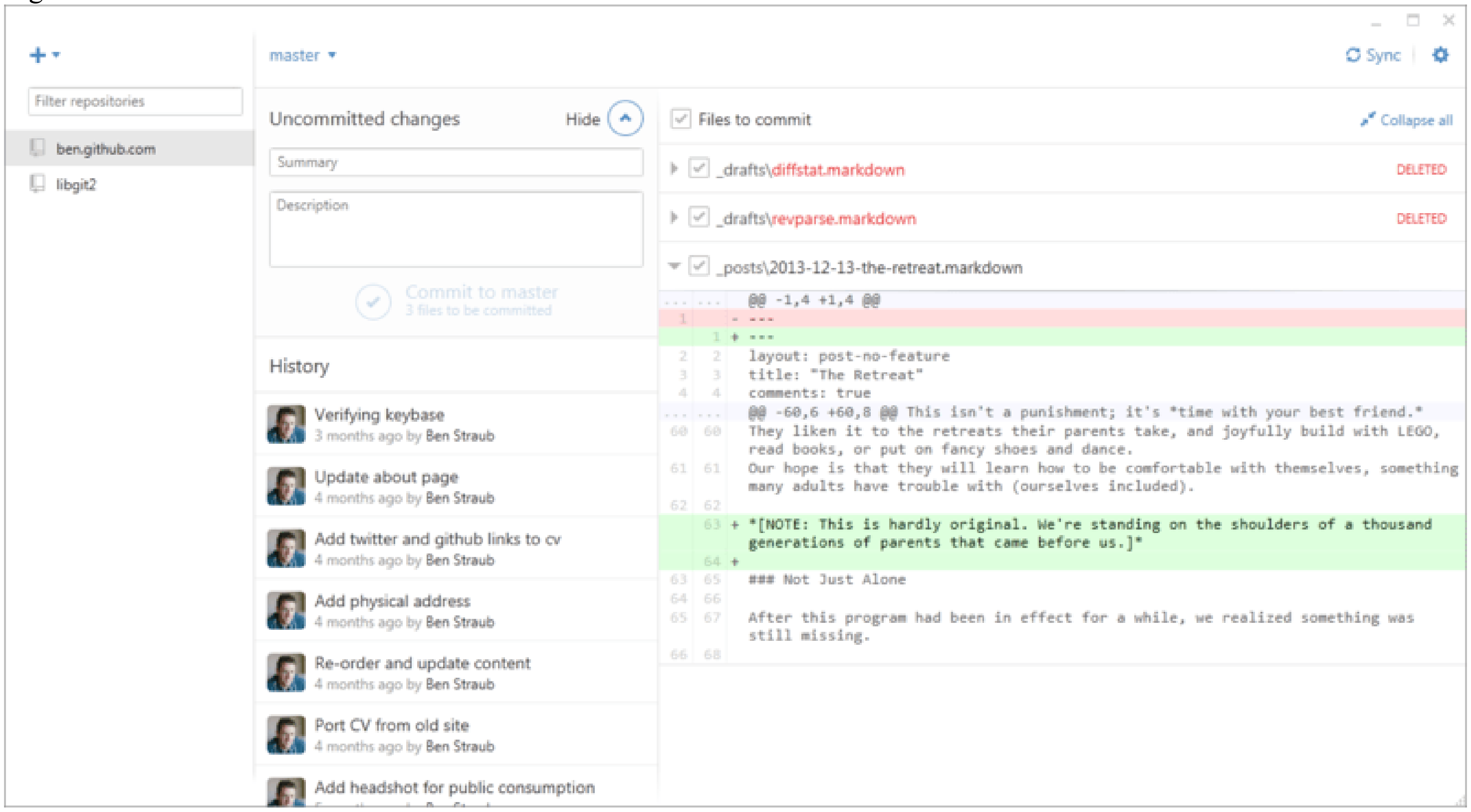


DELETED _drafts/diffstat.markdown
DELETED _drafts/revparse.markdown
_posts/2013-12-13-the-retreat.markdown
~/src/ben.github.com

_posts/2013-12-13-the-retreat.markdown

```
@@ -35,7 +35,7 @@ Here are the rules:

1. Every six months, you get three nights *off*, in a place that isn't
home.
1. The rest of the family leaves you alone, and you have no
responsibilities to them.
- You make your own schedule, and do as you please.
+  You make your own schedule, and do as you please.
1. You are not allowed to feel guilty about this.

Those of you who are single will think "that sounds like every day to me,"
and you'd be right.
@@ -60,6 +60,8 @@ This isn't a punishment; it's *time with your best
friend.*
They liken it to the retreats their parents take, and joyfully build with
LEGO, read books, or put on fancy shoes and dance.
Our hope is that they will learn how to be comfortable with themselves,
something many adults have trouble with (ourselves included).

+*[NOTE: no parenting idea is original. A thousand generations of
parenthood has produced lots of good ideas, and we're definitely standing
on the shoulders of giants here.]*
+
### Not Just Alone

After this program had been in effect for a while, we realized something
was still missing.
```

Figure 153. GitHub for macOS



_drafts\diffstat.markdown DELETED
_drafts\revparse.markdown DELETED
_posts\2013-12-13-the-retreat.markdown

```
@@ -1,4 +1,4 @@
- ---
+ ---
layout: post-no-feature
title: "The Retreat"
comments: true
@@ -60,6 +60,8 @@ This isn't a punishment; it's *time with your best friend.*
They liken it to the retreats their parents take, and joyfully build with LEGO,
read books, or put on fancy shoes and dance.
Our hope is that they will learn how to be comfortable with themselves, something
many adults have trouble with (ourselves included).

+ *[NOTE: This is hardly original. We're standing on the shoulders of a thousand
generations of parents that came before us.]*
+
### Not Just Alone

After this program had been in effect for a while, we realized something was
still missing.
```

Verifying keybase
3 months ago by Ben Straub
Update about page
4 months ago by Ben Straub
Add twitter and github links to cv
4 months ago by Ben Straub
Add physical address
4 months ago by Ben Straub
Re-order and update content
4 months ago by Ben Straub
Port CV from old site
4 months ago by Ben Straub
Add headshot for public consumption

Figure 154. GitHub for Windows

They are designed to look and work very much alike, so we'll treat them like a single product in this chapter. We won't be doing a detailed rundown of these tools (they have their own documentation), but a quick tour of the "changes" view (which is where you'll spend most of your time) is in order.

- On the left is the list of repositories the client is tracking; you can add a repository (either by cloning or attaching locally) by clicking the "+" icon at the top of this area.

- In the center is a commit-input area, which lets you input a commit message, and select which files should be included. On Windows, the commit history is displayed directly below this; on macOS, it's on a separate tab.
- On the right is a diff view, which shows what's changed in your working directory, or which changes were included in the selected commit.
- The last thing to notice is the "Sync" button at the top-right, which is the primary way you interact over the network.

Note You don't need a GitHub account to use these tools. While they're designed to highlight GitHub's service and recommended workflow, they will happily work with any repository, and do network operations with any Git host.

Installation

GitHub for Windows can be downloaded from https://windows.github.com, and GitHub for macOS from https://mac.github.com. When the applications are first run, they walk you through all the first-time Git setup, such as configuring your name and email address, and both set up sane defaults for many common configuration options, such as credential caches and CRLF behavior.

Both are "evergreen" – updates are downloaded and installed in the background while the applications are open. This helpfully includes a bundled version of Git, which means you probably won't have to worry about manually updating it again. On Windows, the client includes a shortcut to launch PowerShell with Posh-git, which we'll talk more about later in this chapter.

The next step is to give the tool some repositories to work with. The client shows you a list of the repositories you have access to on GitHub, and can clone them in one step. If you already have a local repository, just drag its directory from the Finder or Windows Explorer into the GitHub client window, and it will be included in the list of repositories on the left.

Recommended Workflow

Once it's installed and configured, you can use the GitHub client for many common Git tasks. The intended workflow for this tool is sometimes called the "GitHub Flow." We cover this in more detail in The GitHub Flow, but the general gist is that (a) you'll be committing to a branch, and (b) you'll be syncing up with a remote repository fairly regularly.

Branch management is one of the areas where the two tools diverge. On macOS, there's a button at the top of the window for creating a new branch:

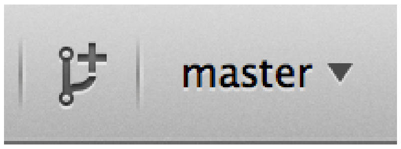


Figure 155. "Create Branch" button on macOS

On Windows, this is done by typing the new branch's name in the branch-switching widget:

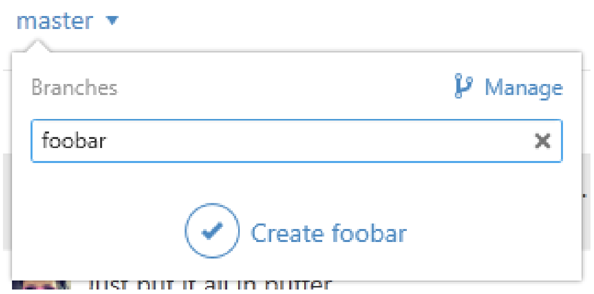


Figure 156. Creating a branch on Windows

Once your branch is created, making new commits is fairly straightforward. Make some changes in your working directory, and when you switch to the GitHub client window, it will show you which files changed. Enter a commit message, select the files you'd like to include, and click the "Commit" button (ctrl-enter or ⌘-enter).

The main way you interact with other repositories over the network is through the "Sync" feature. Git internally has separate operations for pushing, fetching, merging, and rebasing, but the GitHub clients collapse all of these into one multi-step feature. Here's what happens when you click the Sync button:

1. `git pull --rebase`. If this fails because of a merge conflict, fall back to `git pull --no-rebase`.

2. `git push`.

This is the most common sequence of network commands when working in this style, so squashing them into one command saves a lot of time.

Summary

These tools are very well-suited for the workflow they're designed for. Developers and non-developers alike can be collaborating on a project within minutes, and many of the best practices for this kind of workflow are baked into the tools. However, if your workflow is different, or you want more control over how and when network operations are done, we recommend you use another client or the command line.

Other GUIs

There are a number of other graphical Git clients, and they run the gamut from specialized, single-purpose tools all the way to apps that try to expose everything Git can do. The official Git website has a curated list of the most popular clients at https://git-scm.com/downloads/guis. A more comprehensive list is available on the Git wiki site, at https://git.wiki.kernel.org/index.php/Interfaces,_frontends,_and_tools#Graphical_Interfaces.

prev | next
About this site