LEARNING MADE EASY

GitHub®

for dummies®
A Wiley Brand

Create a repository for your coding projects

Personalize your workflow with GitHub tools and integrations

Collaborate on open source software projects

Sarah Guthals, PhD

Phil Haack

Former Engineering Leaders at GitHub

GitHub®

for dummies®
A Wiley Brand

GitHub®

by Sarah Guthals, PhD
and Phil Haack

for dummies®
A Wiley Brand

GitHub® For Dummies®

Published by: **John Wiley & Sons, Inc.,** 111 River Street, Hoboken, NJ 07030-5774, www.wiley.com

Library of Congress Control Number: 2019938575

ISBN 978-1-119-57267-1 (pbk); ISBN 978-1-119-57266-4 (ePDF); ISBN 978-1-119-57265-7 (epub)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

Contents at a Glance

Table of Contents

Introduction

Welcome to the world of collaborative coding! Whether you're just starting your coding journey, building fairly complex programs, or building with a team of people, this book guides you in using one of the most used tools for collaborative code-writing; GitHub.com. With more than 31 million users and over 100 million repositories (projects) hosted, GitHub.com is the No. 1 place to build and collaborate on code.

About This Book

Though you spend many hours sitting at your computer, alone, debugging and writing code, the ideal coding team includes more than just you. Hundreds of developers spent more than four years building World of Warcraft before its first release in 2004. Although occasionally you can build a big hit like Flappy Bird alone, in a couple of days, the norm for software development is that you will work with other coders, designers, testers, user experience experts, product managers, and sometimes hardware engineers to bring something to the hands of users.

When you're first starting out on complex coding projects, understanding effective ways to collaborate can be daunting. This book introduces you to the world of open source development (the epitome of collaboration), as well as effective ways to work with one other person — or even yourself over the course of many years! (I don't know about you, but Sarah from three years ago knows stuff that Sarah from today can't remember, and Sarah from today has more experience than Sarah from three years ago.)

GitHub For Dummies is written as a reference guide. Each part introduces you to a different aspect of collaborative coding. So if you're experienced in using GitHub, but you're new to the open source community, you can jump to Part 5 and skip some of the GitHub basics.

As you explore each part of this book, keep the following points in mind:

- Words that are being defined appear in *italic*.
- Code and URLs (web addresses) are shown in `monofont`.
- Command sequences using onscreen menus use the command arrow. For example, when working in Scratch, you can open a new project as follows: From the menu bar, choose File ➪ New.
- The figures you see in this book use Mac and Chrome. We provide some tips when what you see on a Windows PC may be different, but you should see the same things, regardless of which Internet browser you use.

Foolish Assumptions

In this book, I make some assumptions that very well may be foolish, about you, your coding experience, and your goals.

- You're interested in and have had some experience with coding. You don't have to be an expert coder, but you have made a Hello World application (or the equivalent) in at least one programming language.
- You have patience and determination and are resourceful. When you're presented with a challenge, you can Google your way to finding a solution. This book guides you through GitHub.com as it exists at the time of writing it, but new features and workflows are being created, and part of your collaborative coding journey is to discover how to use those new features as they become available.
- You have experience with a keyboard and mouse on either Mac or Windows PC and have access to one of those machines.
- You're capable of using an Internet browser, such as Safari, Chrome, or Firefox, and you can type a URL to access a website, such as GitHub.com.
- You know how to install applications on your computer. Although we guide you through anything that is unique to the setup, you should know how to download and install an application without step-by-step guidance.

Icons Used in This Book

Throughout the margin of this book are small images, known as *icons.* These icons mark important tidbits of information:

TIP

The Tip icon identifies places where we offer additional tips for making this journey more interesting or clear. Tips can start you on a rabbit hole down another workflow, not covered in this book, or cover some neat shortcuts that you may not have known about.

REMEMBER

The Remember icon bookmarks to important ideas to help you work more effectively throughout this book.

WARNING

The Warning icon helps protect you from common errors and may even give you tips to undo your mistakes.

Beyond the Book

In addition to what you're reading right now, this product also comes with a free access-anywhere Cheat Sheet that covers common commands and GitHub actions. To get this Cheat Sheet, simply go to `www.dummies.com` and search for "GitHub For Dummies Cheat Sheet" in the Search box.

Other online resources also pair with this book:

- Online articles covering additional topics are available at `www.dummies.com/extras/github`.
- Updates to this book, if any, can be found at `www.dummies.com/extras/github`.
- GitHub Learning Labs are free, guided tutorials that can be installed and found at `https://lab.github.com`.

Where to Go from Here

GitHub is a tool used by millions of developers. The workflows that you discover in this book is just the beginning. As you become a more experienced coder, begin to collaborate on more elaborate projects, or join different companies and teams, you may encounter new workflows that use these tools in different ways. You should feel empowered to explore! Visit `https://help.github.com` or `https://guides.github.com` for guidance and don't forget to follow the blog at `https://blog.github.com/` to stay up to date with all of the new features!

1

Getting Started with GitHub.com

IN THIS PART . . .

Discover how to use Git on your local computer to track changes in your project.

Sign up for a free GitHub.com account.

Explore GitHub.com resources and features.

Install GitHub Desktop to manage the link between your local and remote projects.

Install the GitHub Atom editor as a lightweight option for coding.

Prepare for creating your own projects and contributing to others.

IN THIS CHAPTER

- » Getting familiar with GitHub
- » Discovering Git
- » Signing up with GitHub.com
- » Exploring helpful resources

Chapter 1 Understanding the Git in GitHub

Whether you're an experienced coder or a newbie starting out, learning how to work with others on code is critical to succeeding in the software industry. Millions of people around the world work together to build software, and GitHub is one of the largest tools to support a collaborative workflow. This chapter introduces you to the core tools you need to write code with other people.

Introducing GitHub

GitHub creates an environment that allows you to store your code on a remote server, gives you the ability to share your code with other people, and makes it easy for more than one person to add, modify, or delete code to the same file and project, while keeping one source of truth for that file (phew!). So what does that all actually mean? One of my favorite ways of explaining GitHub.com to folks who are new to the tool is to compare it to Google Docs — a place online where you can write code with other people and not have to email different versions back and forth.

What makes GitHub work behind the scenes is Git.

Understanding Version Control

Version control systems (also known as source control management, or SCM) are software that keep track of each version of each file in a coding project, a timestamp for when that version was created, and the author of those changes.

TIP

Writing code is an iterative process. For example, when you're building a website, you first may want to get some basic structure up before adding all your content. The best thing to do is to create a version of your website each time you have something that works. That way, as you experiment with the next piece, if something breaks, you can just go back to your previous version and start over.

SCMs enable coders to make mistakes without worrying that they'll have to completely start over. Think of it like being able to click Undo, but instead of undoing each key press, you can undo an entire piece of the project if you decide you don't like it or it doesn't work.

The basic workflow of coding with version control system support is as follows:

1. **Create a project, typically in a folder on your computer.**
2. **Tell your version control system of choice to track the changes of your project/folder.**
3. **Each time your project is in a working state, or you're going to walk away from it, tell your version control system of choice to save it as the next version.**
4. **If you ever need to go back to a previous version, you can ask your version control system to revert to whichever previous version you need.**

You can use a version control system if you're working alone on your own computer, but it gets even more interesting when you begin working with other people. (For more on working with other people, see the section "Git version control," later in this chapter).

For more information about version control, visit `https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control`.

Git Version Control

GitHub, as the same would suggest, is built on Git. Git is a type of version control system, and it is free and open source, which means that anyone can use it, build on top of it, and even add to it.

GitHub products make using Git easy, but if you're curious, you can also use Git to track your solo projects on your computer. You can find a brief introduction to local Git commands for solo projects in the section "Try simple Git on the terminal".

Try simple Git on the terminal

With the help of Git for Windows, using the terminal on Mac, Windows, or Linux computers is exactly the same. A *terminal* is an application that enables you to interact with your computer in a text-based way — in other words, instead of double-clicking and dragging, you type commands to navigate your computer.

If you're on Mac or Linux, a terminal is already installed on your computer. If you're using a Windows computer, install Git for Windows, which is what this section assumes you've installed. Just go to `https://gitforwindows.org` and click Download to gain access to Git Bash, an emulator that allows you to interact with Git just like you would on a Linux or Mac terminal. You also get Git GUI, which gives you a user interface for almost all Git commands you might type into Git Bash, and shell integration so that you can quickly open Git Bash or Git GUI from any folder.

TIP

Many developers on Windows prefer to use PowerShell as their terminal environment. You can use Git within PowerShell, but setting that up properly is outside the scope of this book. However, Phil, one of the authors, has a handy guide to setting this up at `https://haacked.com/archive/2011/12/13/better-git-with-powershell.aspx`.

First, find the Terminal application:

- » On Mac, you can click the magnifying glass at the top right of your desktop, type **Terminal**, select the terminal from the list of applications, and press Enter or click it.
- » On Linux, press Ctrl-Alt-T all at the same time, and the terminal window opens.
- » On Windows, click the Windows menu in the bottom right of your desktop, search **Git Bash,** select the Git Bash application from the list of search results, and press Enter or click it.

When the application opens, type `git --version` in the terminal. If you have Git installed, you should see a version number, as shown in the following code (the `$` should already be on the line, so you do not need to type it). Otherwise, you can follow the instructions on `https://git-scm.com/book/en/v2/Getting-Started-Installing-Git`.

WARNING

When using the command line, you have to be very careful about exactly what you're typing. In the following code, the first instruction is for you to type `git --version`. You should note that a space appears between `git` and the rest of the instruction but no other spaces. You should also note the two dashes before the word `version`. They can be easy to miss, so be careful!

For Mac or Linux, you should see something like this:

```
$ git --version
git version 2.16.3
$
```

For Windows, you should see something like this:

```
$ git --version
git version 2.20.1.windows.1
$
```

Next, using the terminal, go to your desktop and create a new folder called git practice. To do this, you should type the following commands:

```
$ cd ~/Desktop
$ mkdir git-practice
$ cd git-practice
$
```

If you type `pwd`, you should see that you are now in the folder git-practice, which is on your desktop. It might look something like this:

```
$ pwd
$ /Users/sguthals/Desktop/git-practice
$
```

```
Now, you can tell git to track this folder using the init command.
$ git init
Initialized empty Git repository in /Users/sguthals/Desktop/git-practice
$
```

Then make sure that you have a clean folder. You can check with the `status` command:

```
$ git status
On branch master

No commits yet

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

Then, you can create a file to have Git start tracking and confirm the file is in the folder:

```
$ echo "practicing git" > file.txt
$ ls
file.txt
$
```

On Mac, you can open this folder in a Finder with the `open <path>` command:

```
$ open .
$
```

On Linux, you can open this folder with the `nautilus <path>` command:

```
$ nautilus .
$
```

On Windows, you can open this folder with the `explorer <path>` command:

```
$ explorer .
$
```

In this example, we put `.` as the `<path>` for each command. `.` tells the terminal to open the current folder. You could also use a different path with these commands to open other folders.

After the folder is open, double-click the file called `file.txt`, and the file opens with TextEdit on Mac, gedit on Linux, and Notepad on Windows. You can see that the words "practicing git" are actually there.

Close the file. Now, you can tell Git that you want to save this as a particular version. Back in the terminal:

```
$ git add file.txt
$ git commit -m "Adding my file to this version"
[master (root-commit) 8d28a21] Adding my file to this version
 1 file changed, 1 insertion(+)
 Create mode 100644 file.txt
$ git status
On branch master
nothing to commit, working tree clean
$
```

You can make a change to your file in the text file. Open the file again, change the text to say "Hi! I'm practicing git today!" and then click File➪ Save and close the text application.

When you go back to the Terminal to check the status of your project again, you should see that Git has noticed that the file has changed:

```
$ git status
On branch master
Changed not staged for commit:
  (use "git add <file..." to update what will be committed)
  {use "git checkout -- <file>..." to discard changed in working directory)

        modified:    file.txt

no changed added to commit (use "git add" and/or "git commit -a")
$
```

`Commit` this version of your file again and notice that Git recognizes that everything has been saved to a new version:

```
$ git add file.txt
$ git commit -m "I changed the text"
[master 6d80a2a] I changed the text
 1 file changed, 1 insertion(+), 1 deletion(-)
```

```
$ git status
On branch master
nothing to commit, working tree clean
$
```

TIP

If your terminal starts to get too cluttered, you can type `clear` to clear some space and make it more visually appealing. Don't worry; you can always scroll up and see everything you typed earlier!

Say that you actually want to go see the original change; when you added "practicing git". First, get the `log` of all the `commits` you have made:

```
$ git log
commit 6d80a2ab7382c4d308de74c25669f16d1407372d (HEAD -> master)
Author: sguthals <sguthals@github.com>
Date:   Sun Dec 9 08:54:11 2018 -0800

    I changed the text

commit 8d28a21f71ec5657a2f5421e03faad307d9eec6f
Author: sguthals <sguthals@github.com>
Date:   Sun Dec 9 08:48:01 2018 -0800

    Adding my file to this version
$
```

Then ask Git to show you the first `commit` you made (the bottom most one). Make sure that you're typing your unique `commit hash`. In this book, the hash starts with `8d28a2`. Make sure you type the entire hash that appears in your Git log:

TIP

Instead of typing the entire hash (and possibly having a typo), you can highlight the hash with your mouse, right-click and choose copy, and then after `git checkout`, you can right-click and choose Paste. Using the keyboard shortcuts Ctrl+C or ⌘-C doesn't work

```
$ git show 8d28a21f71ec5657a2f5421e03faad307d9eec6f
commit 8d28a21f71ec6567a2f5421e03faad307d9eec6f
Author: sguthals <sarah@guthals.com>
Date:   Sun Dec 9 08:48:01 2018 -0800

    Adding my file to this version

diff --git a/file.txt b/file.txt
new file mode 100644
```

```
index 0000000..849a4c7
--- /dev/null
+++ b/file.txt
@@ -0,0 +1 @@
+practicing git
$
```

You can see that `practicing git` was added to the file in that original commit.

For more information on how to use git on the command line, check out the following resources:

- » The GitHub Git Cheat Sheet at `https://services.github.com/on-demand/downloads/github-git-cheat-sheet.pdf`
- » The Visual Git Cheat Sheet at `http://ndpsoftware.com/git-cheatsheet.html`
- » The Git Docs page at `https://git-scm.com/doc`

Another couple resources for learning and understanding Git is `https://learngitbranching.js.org` and `http://git-school.github.io/visualizing-git`, which enable users on Windows to experience a similar workflow because they're visualizations hosted on a website. The first link, learninggitbranching.js.org, is a good self-guided set of exercises, while the second link, git-school, is best used for folks who have a decent understanding of Git and want to explore what will happen in different scenarios, or for folks who have a more expert Git user guiding them.

Git branching by collaborator

Git is different from other version control systems because it has fast branching, shown in Figure 1-1. *Branching* is a Git function that essentially copies code (each `branch` is a copy of the code), allows you to make changes on a specific copy, and then merges your changes back into the main (`master`) branch.

When you're writing code, you will add files and commit changes to your `master` branch. Figure 1-1 outlines a specific workflow where two people are collaborating on the same file. Person 1 creates a new branch called `MyBranch` and makes some changes to the file. Person 2 also creates a new branch called `YourBranch` and makes some changes to the same file. You can see this change in box #1.