2
Cloning a Repo Written by Chris Belanger & Bhagat Singh

The preceding chapter took you through a basic crash course in Git, and got you right into using the basic mechanisms of Git: cloning a repo, creating branches, switching to branches, committing your changes, pushing those changes back to the remote and opening a pull request on GitHub for your changes to be reviewed.

That explains the *how* aspect of Git, but, if you've worked with Git for any length of time (or haven't worked with Git for any time at all), you'll know that the *how* is not enough. It's important to also understand the *why* of Git to gain not just a better understanding of what's going on under the hood, but also to understand how to fix things when, not if, your repository gets into a weird state.

So, first, you'll start with the most basic aspect of Git: getting a repository copied to your local system via **cloning**.

What is cloning?

Cloning is exactly what it sounds like: creating a copy, or clone, of a repository. A Git repository is nothing terribly special; it's simply a directory, containing code, text or other assets, that tracks its own history. Then there's a bit of secure file transfer magic in front of that directory that lets you sync up changes. That's it.

A Git repository tracks the history of all changes inside the repository through a hidden **.git** directory that you usually don't ever have to bother with — it's just there to quietly track everything that happens inside the repository. You'll learn more about the structure and function of the hidden .git directory later on in this book.

So since a Git repository is just a special directory, you could, in theory, effect a pretty cheap and dirty clone operation by zipping up all the files in a repository on your friend's or colleague's workstation and then emailing it to yourself. When you extract the contents of that zipped-up file, you'd have an exact copy of the repository on your computer.

However, emailing things around can (and does) get messy. Instead, many organizations make use of online repository hosts, such as GitHub, GitLab, BitBucket or others. Some organizations even choose to self-host repositories, but that's outside the scope of this book. For now, you'll stick to using online hosts — in this example, GitHub.

Using GitHub

GitHub, at its most basic level, is really just a big cloud-based storage solution for repositories, with account and access management mixed in with some collaboration tools. But you don't need to know about all the features of GitHub to start working with repositories hosted on GitHub, as demonstrated in the Git crash course in the previous chapter.

Cloning from an online repository is a rather straightforward operation. To get started, you simply need the following things:

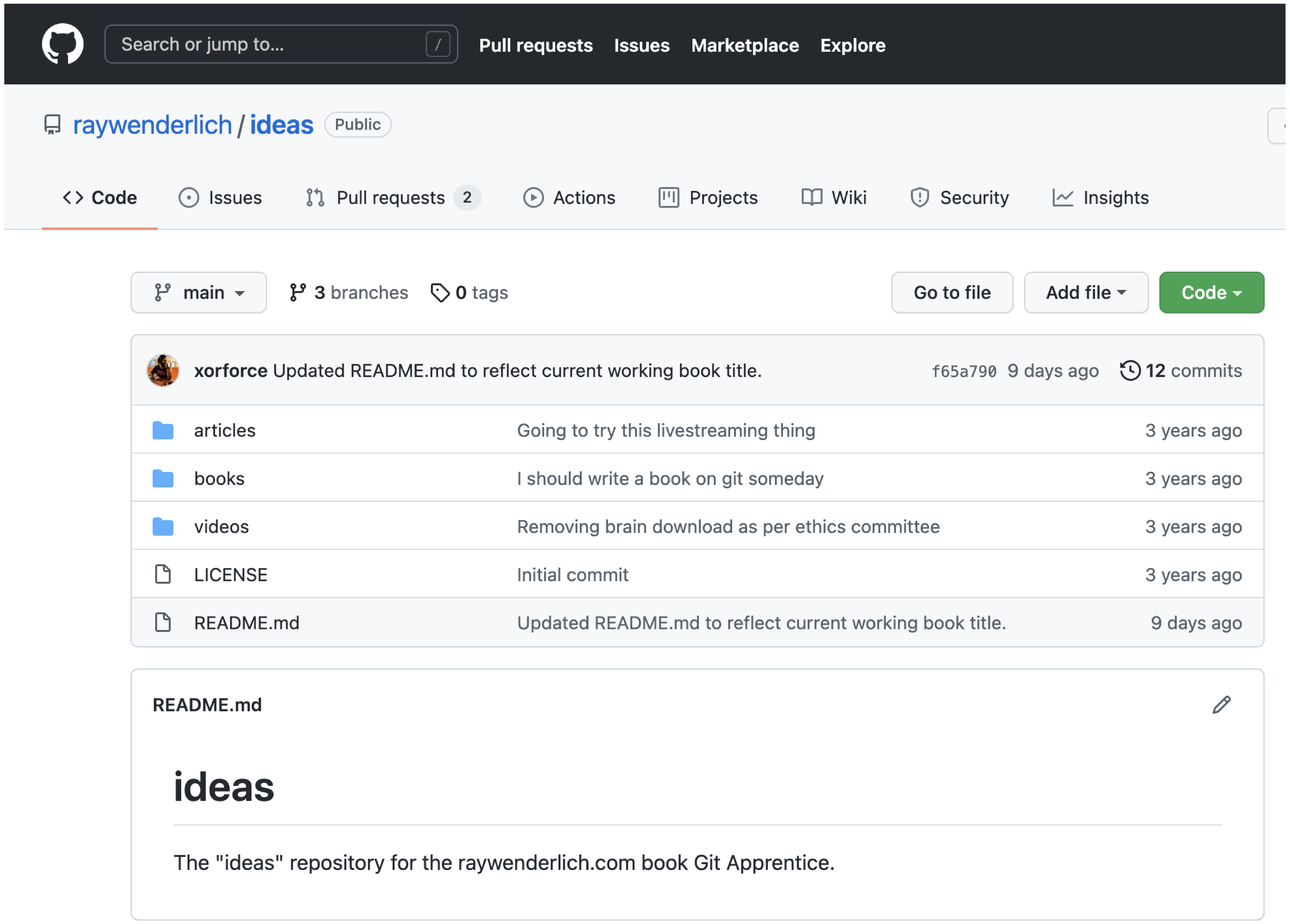
- A working installation of Git on your local system.
- The remote URL of the repository you want to clone.
- Any credentials for the online host.

Note: It *is* generally possible to clone repositories without using credentials, but you won't be able to propagate the changes you make on your local copy back to the online host.

The GitHub repository homepage

There's a repository already set up on GitHub for you to clone, so you first need to get the remote URL of the repository.

To start, navigate to https://github.com/raywenderlich/ideas and log in with your GitHub username and password. If you haven't already set up an account, you can do so now.



xorforce Updated README.md to reflect current working book title. f65a790 9 days ago 12 commits

articles	Going to try this livestreaming thing	3 years ago
books	I should write a book on git someday	3 years ago
videos	Removing brain download as per ethics committee	3 years ago
LICENSE	Initial commit	3 years ago
README.md	Updated README.md to reflect current working book title.	9 days ago

README.md

ideas

The "ideas" repository for the raywenderlich.com book Git Apprentice.

The main page for the ideas repository.

Once you're on the homepage for the repository, have a look at the list of files and directories listed on the page. These lists and directories represent the contents of the repository, and they are the files that you'll clone to your local system.

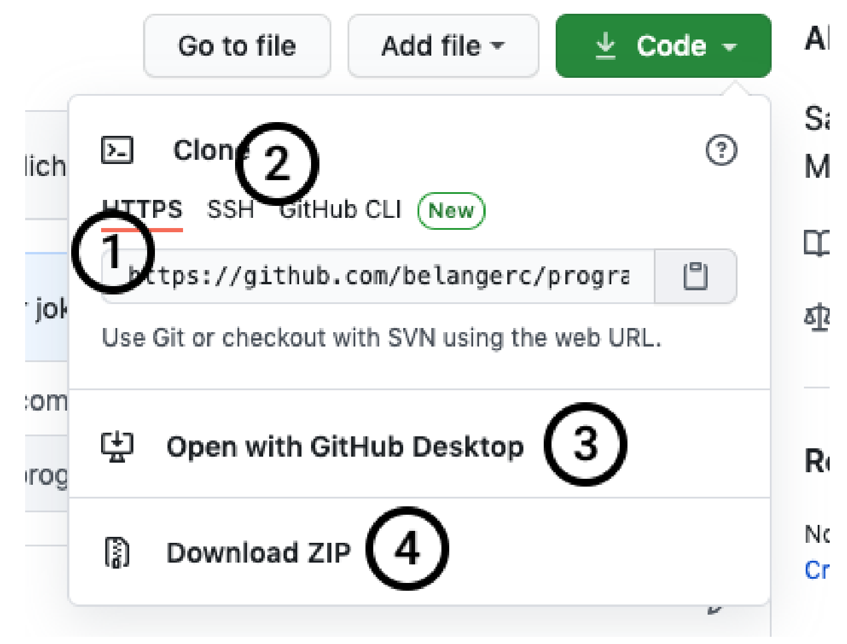
But where do you find the remote URL of the repository to clone it? Like many things in Git (and with computers, in general), there are multiple ways to clone a repository. In this chapter, you'll use the easiest and most common cloning method, which starts on the GitHub repository homepage.

Finding the repository clone URL

Look for and click on the **Code** button on the repository homepage.

The 'Clone or download' button displays the various cloning options for a repository.

The little pop-up dialog gives you a few options to get a repository cloned to your local system:



Use Git or checkout with SVN using the web URL.

The cloning options for at GitHub repository.

. This is the main HTTPS URL for the repository. This is the URL that you'll use in this chapter to clone from the command line `git` client.

. You can also use SSH to clone a repository. Clicking this link lets you toggle between using SSH and HTTPS to work with the repository.

. If you have the GitHub Desktop app installed, you can use the **Open in Desktop** button to launch GitHub Desktop and clone this repository all in one step.

. If you just want a zipped copy of what's in the repository (but not all the repository bits itself), the **Download ZIP** button will let you do this.

For now, copy the HTTPS URL that you see in the dialog via the little clipboard icon button to the right of the URL. This places a copy of the HTTPS URL in your clipboard so that you can paste it into your command line later.

Cloning on the command line

Now, go to your command prompt. Change to a suitable directory where you want your repositories to live. In this case, I'll create a directory in my home directory named **GitApprentice** where I would like to locally store all of the repos for this book.

Execute the following command to create the new directory:

```
mkdir GitApprentice
```

Now, execute the following command to see the listing of files in the directory (yours will be different than shown below):

```
ls
```

I see the following directories on my system, and there's my new **GitApprentice** directory:

```
~ $ ls
Applications            Downloads               Music
Dropbox                         Pictures                Library
Public                          Desktop                 GitApprentice
Documents                       Movies
```

Execute the following command to navigate into the new directory:

```
cd GitApprentice
```

You're now ready to use the command line to clone the repository.

Enter the following command, but don't press the **Enter** key or **Return** key just yet:

```
git clone
```

Now, press the **Space bar** to add one space character and paste in the URL you copied earlier, so your command looks as follows:

```
git clone https://github.com/raywenderlich/ideas.git
```

Now, you can press **Enter** or **Return** to execute the command.

You'll see a brief summary of what Git is doing below:

```
~/GitApprentice $ git clone https://github.com/raywenderlich/ideas.git
Cloning into 'ideas'...
remote: Enumerating objects: 52, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 52 (delta 0), reused 2 (delta 0), pack-reused 49
Receiving objects: 100% (52/52), 6.26 KiB | 173.00 KiB/s, done.
Resolving deltas: 100% (22/22), done.
```

Execute the `ls` command to see the new contents of your **GitApprentice** directory:

```
~/GitApprentice $ ls
ideas
```

Use the `cd` command, followed by the `ls` command, to navigate into the new **ideas** directory and see what's inside:

```
~/GitApprentice $ cd ideas
~/GitApprentice/ideas $ ls
LICENSE         README.md       articles        books           videos
```

So there's the content from the repository. Well, the *visible* content at least. Run the `ls` command again with the `-a` option to show the hidden .git directory discussed earlier:

```
~/GitApprentice/ideas $ ls -a
.               .git            README.md       books
..              LICENSE         articles        videos
```

Aha — there's that magical **.git** hidden directory. Take a look at what's inside.

Exploring the .git directory

Use the `cd` command to navigate into the .git directory:

```
cd .git
```

Execute the `ls` command again to see what dark magic lives inside this directory. This time, use the `-F` option so that you can tell which entities are files and which are directories:

```
ls -F
```

You'll see the following:

```
~/GitApprentice/ideas/.git $ ls -F
HEAD            config          hooks/          info/           objects/        refs/
branches/       description     index           logs/           packed-refs
```

So it's not quite the dark arts, I'll admit. But what *is* here is a collection of important files and directories that track and control all aspects of your local Git repository. Most of this probably won't make much sense to you at this point, and that's fine. As you progress through this book, you'll learn what most of these bits and pieces do.

For now though, leave everything as-is; there's seldom any reason to work at this level of the repository. Pretty much everything you do should happen up in your working directory, not in the .git subfolder.

So backtrack up one level to the the working directory for your repository with the `cd` command:

```
cd ..
```

You're now back up in the relative safety of the top level of your repository. For now, it's enough to know where that .git directory lives and that you really don't have a reason to deal with anything in there right now.

Forking

You've managed to make a clone of the **ideas** repository, but although **ideas** is a public repository, the **ideas** repository currently belongs to the raywenderlich organization. And since you're not a member of the raywenderlich organization, the access control settings of the **ideas** repository mean that you won't be able to push any local changes you make back to the server. Bummer.

But with most public repositories, like **ideas**, you can create a remote copy of the repository up on the server under your own personal user space. You, or anyone you grant access to, can then clone that copy locally, make changes and push those changes back to the remote copy on the server. Creating a remote clone of a repository is known as **forking**.

First, you'll need to rid your machine of the existing local clone of the **ideas** repository. It's of little use to you in its current state, so it's fine to get rid of it.

First, head up one level, out of your working directory, by executing the following command:

```
cd ..
```

You should be back up at the main **GitApprentice** directory:

```
~/GitApprentice $
```

Now, get rid of the local clone with the `rm` command, and use the `-rf` options to recursively delete all subdirectories and files, and to force all files to be deleted:

```
rm -rf ideas
```

Execute `ls` to be sure the directory is gone:

```
~/GitApprentice $ ls
~/GitApprentice $
```

Looks good. You're ready to create a fork of the raywenderlich **ideas** repository… which leads you to your challenge for this chapter!

Challenge

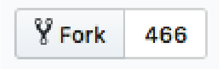
Challenge: Fork on GitHub and create a local clone

The goal of this challenge is twofold:

. Create a fork of the **ideas** repository under your own user account on GitHub.

. Clone the forked repository to your local system.

Navigate to the homepage for the **ideas** repository at https://github.com/raywenderlich/ideas. In the top right-hand corner of the page, you'll see the **Fork** button. That's your starting point.



The 'Fork' button lets you create a remote copy of a repository.

The steps to this challenge are:

. Fork the **ideas** repository under your own personal user account.

. Find the clone URL of your new, forked repository.

. Clone the forked **ideas** repository to your local system.

. Verify that the local clone created successfully.

. Bonus: Prove that you've cloned the fork of your repo and not the original repository.

If you get stuck, you can always find the solution to this challenge under this chapter's **projects/challenge** folder inside the materials you downloaded for this book.

Key points

Cloning creates a local copy of a remote Git repository.
Use `git clone` along with the clone URL of a remote repository to create a local copy of a repository.
Cloning a repository automatically creates a hidden **.git** directory, which tracks the activity on your local repository.
Forking creates a remote copy of a repository under your personal user space.

Where to go from here?

Once you've successfully completed the challenge for this chapter, head into the next chapter where you'll learn about the `status`, `diff`, `add` and `commit` commands. You'll also learn just a bit about how Git actually tracks the changes that you make in the local copy of your repository.

3. Committing Your Changes 1. A Crash Course in Git

Have a technical question? Want to report a bug? You can ask questions and report bugs to the book authors in our official book forum here.