Git Training

Ross Dahlke

Table of contents

Intro to Git Training

Welcome to Git training! Git is one of the most powerful and useful programming fundamentals. However, it is seldom taught in school, especially as a part of a social science research programs. In this document, I hope to get you started on your own Git journey. This document will certainly not cover everything related to Git but just a few commands can provide **a lot** of value. When starting out, I found it most useful to use it alongside a tutorial until I really felt comfortable with it. I hope that this document can be that tutorial for you.

Why Git?

Git is a version control system that makes programming faster, more collaborative, and improves data integrity. The big idea is that you can periodically save your code in a certain spot so that you, or a collaborator, can easy track changes and restore older versions. It also allows you get experimental. If you want to try out a crazy idea but you don't want to mess up the existing code, no problem! That's what Git is here for.

Git is all about connecting versions of code that are stored in the central repository (let's just call it the cloud) and your local computer.

If you want another explanation of Git, my programmer friends who are way smarter than me swear by the video Git for Ages 4 And Up

Git Clients

There are a variety of Git "clients" out there, including GitHub, GitKraken, Bitbucket, etc. In this tutorial, I will be showing how to use GitHub with the *command line*. GitHub does have a desktop application which can abstract the use of the command line, but I personally prefer being a little bit more "hands-on" to what is happening under the hood. If there's interest, I'm happy to create a tutorial using the GitHub Desktop app. Other Git storage systems, for example, Bitbucket, can be used similarly this tutorial.

GitHub

Starting with GitHub

1. Create a GitHub account
2. Configure your SSH key

    1. Open your terminal and check to see if you already have an SSH key using the command `ssh-add -l`
    2. If you do not, use the command

        ```
        ssh-keygen -t ed25519 -C "your_email@example.com"
        ```

    3. Press enter to use the default location
    4. Enter secure passphrase (or click enter for none)
    5. Use the command `cat .ssh/id_ed25519.pub` to show your public key
    6. Copy that key to your clipboard
    7. Log into GitHub
    8. Click on settings
    9. Select SSH and GPG keys
    10. Click New SSH Key
    11. Give your key a title
    12. Paste your public key into the Key field
    13. Click Add SSH Key

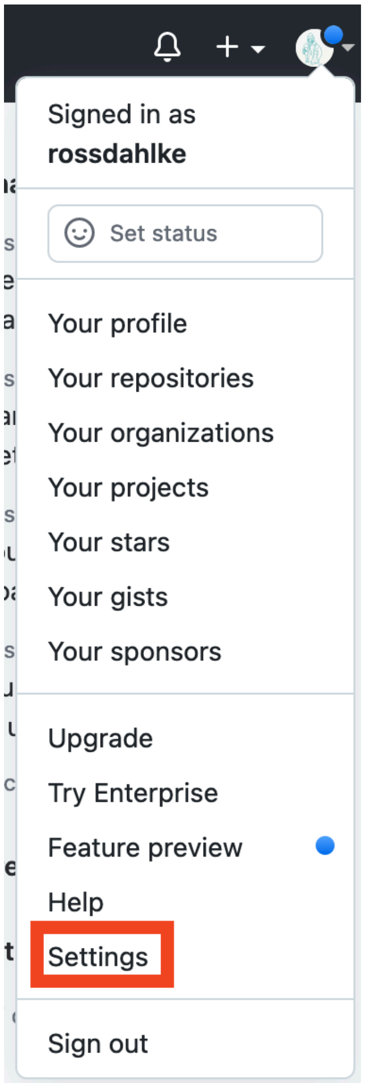


Figure 1: settings

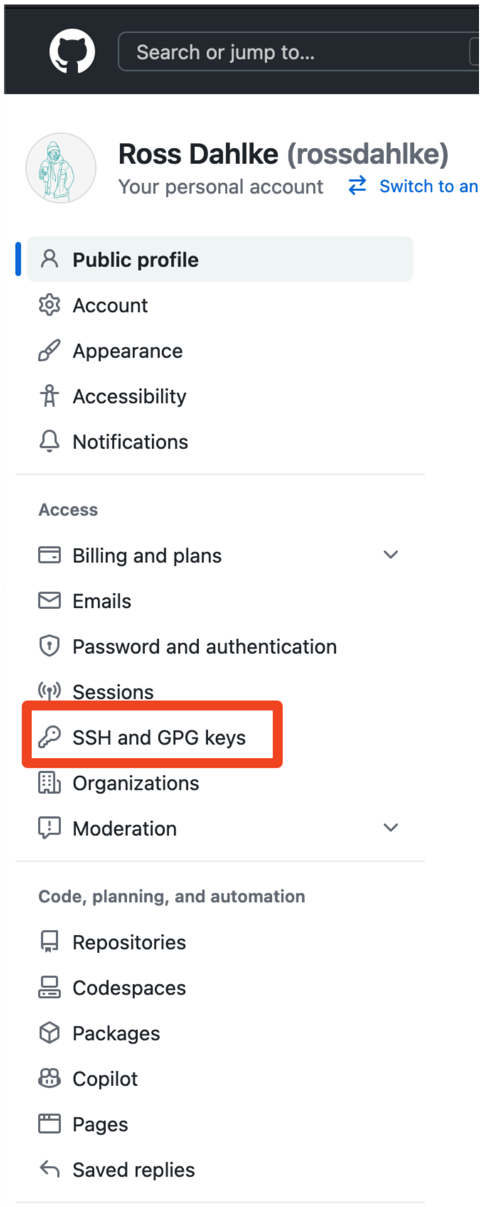


Figure 2: ssh keys button

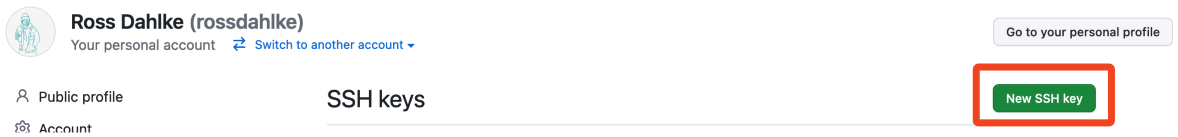


Figure 3: new ssh key

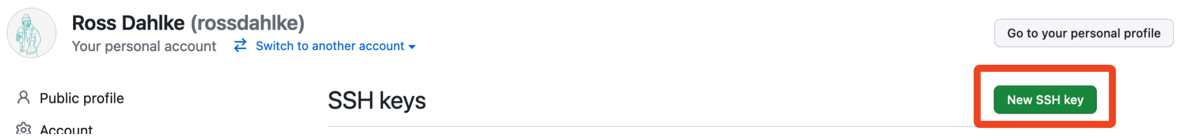


Figure 4: ssh fields

Creating a new repository

1. Go to your GitHub homepage
2. On the left menu "Top Repositories" click the "New" button

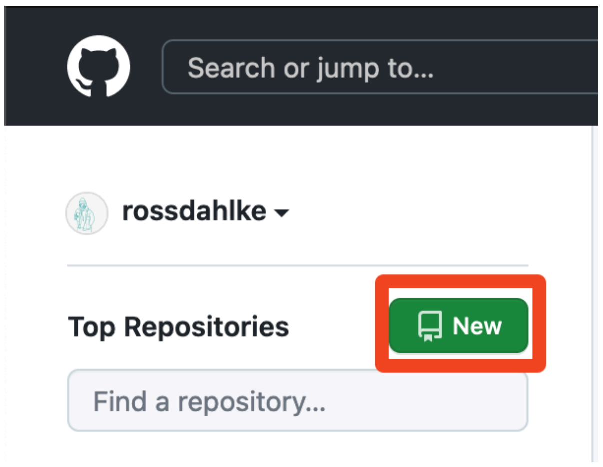


Figure 5: Create a new repository

3. Give your repository a name and fill out the remaining details
4. Congratulations, you just made a new repository! Right now, the repository only exists in the cloud.

Downloading a repository

1. Go to the GitHub page of your repository (e.g., github.com/rossdahlke/git-training)
2. If you do not have the repository downloaded on your local computer, click on the "Code" dropdown
3. Click on "SSH" and copy the git address (e.g., "git@github.com:rossdahlke/git-training.git")
4. Open your terminal on your local computer or IDE

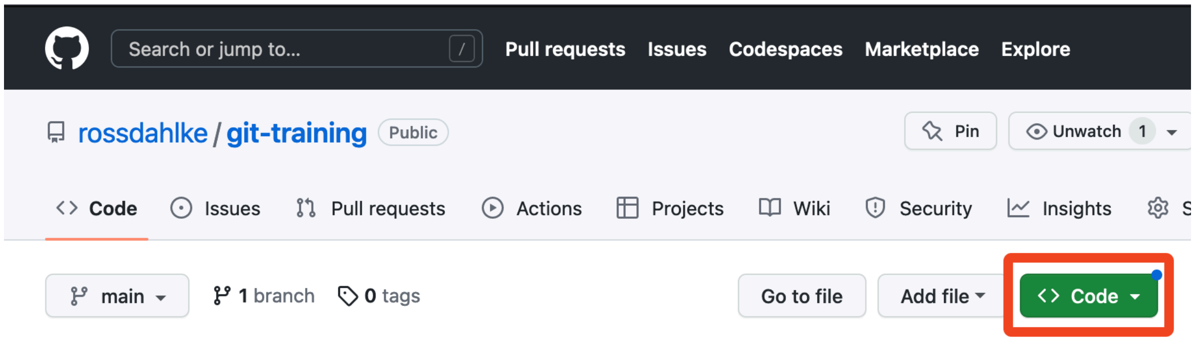
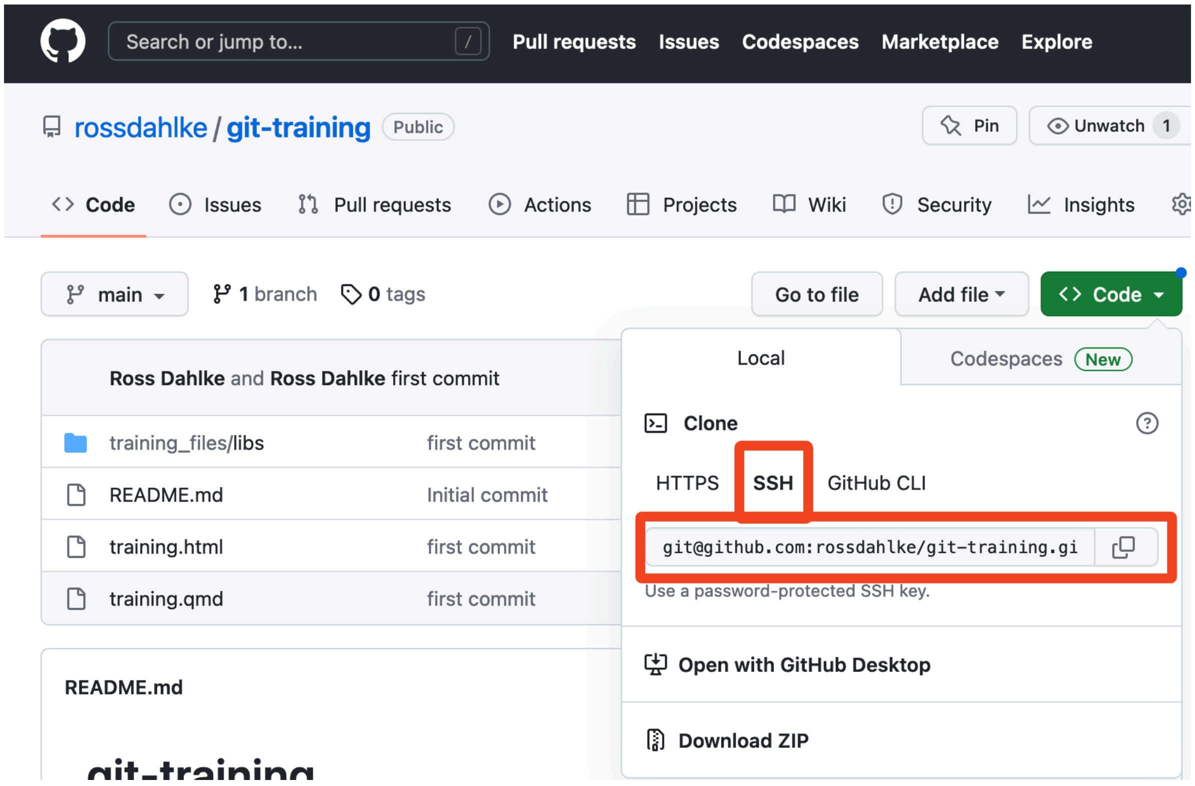


Figure 6: Code Button



```
git@github.com:rossdahlke/git-training.gi
```

Figure 7: ssh

5. Use `ls` and `cd` to navigate to the folder that you want the repository to live in
6. In the terminal use the `git clone` function to "clone" the repository down to your local machine
    1. e.g., `git clone git@github.com:rossdahlke/git-training.git`
7. Now, that repository should exist on your local computer
8. Make any changes you want your want to the code

Sending a repository back up

1. Once you've made the changes you want to the file, you can "commit" and "push" those changes back up to GitHub
2. Open your terminal on your local computer or use the terminal in your IDE
3. Use `ls` and `cd` to navigate to your Git repository
4. Check on the status of the files in your repository using the `git status` command
    1. Files in red are new/ edited files that are currently untracked
    2. Files in green are new/ edited files that are being tracked
5. Add the files you wish to "commit" up using the `git add` function
    1. e.g., `git add training.html`
6. You can check on the status of your files again using `git status`, notice that the files you just added should be green now
7. Commit the changes you made using the `git commit -m` command followed by a message in quotes (e.g., `git commit -m "first commit"`)
8. Push your changes back to GitHub using the `git push` command
9. Check on GitHub to make sure your changes are there
10. Now, other people can go and `git clone` your files and push them back

Just remember the steps of "add, commit, push"!

Getting updates to a repository that someone else pushed

1. If you already have a repository cloned onto your local computer and want to get the latest version of the repo (i.e., after someone else pushed up their changes) use the `git pull` command