Setting Up Git

Foundations Course

Introduction

Git is a very popular version control system. You'll become very familiar with this piece of software throughout TOP, so don't worry too much about understanding it at this point. There are many lessons focused on Git later in the curriculum.

GitHub is a service that allows you to upload your code using Git and to manage your code with a nice web interface. GitHub and Git are not the same thing or even the same company.

Step 1: Install Git

Click the Operating System you have chosen below:

▶ **Linux**

▶ **MacOS**

▶ **Chrome OS/CloudReady**

Step 2: Configure Git and GitHub

Step 2.1: Setup Git

For Git to work properly, we need to let it know who we are so that it can link a local Git user (you) to GitHub. When working on a team, this allows people to see what you have committed and who committed each line of code.

The commands below will configure Git. Be sure to enter your own information inside the quotes (but include the quotation marks)!

```
git config --global user.name "Your Name"
git config --global user.email "yourname@example.com"
```

GitHub recently changed the default branch on new repositories from `master` to `main`. Change the default branch for Git using this command:

```
git config --global init.defaultBranch main
```

To enable colorful output with `git`, type

```
git config --global color.ui auto
```

To verify that things are working properly, enter these commands and verify whether the output matches your name and email address.

```
git config --get user.name
git config --get user.email
```

macOS Users: Run these two commands to tell git to ignore .DS_Store files, which are automatically created when you use Finder to look into a folder. .DS_Store files are invisible to the user and hold custom attributes or metadata

(like thumbnails) for the folder, and if you don't configure GitHub to ignore them, pesky .DS_Store files will show up in your commits.

```
echo .DS_Store >> ~/.gitignore_global
git config --global core.excludesfile ~/.gitignore_global
```

Step 2.2: Create a GitHub Account or Sign In

Go to GitHub.com and create an account! If you already have an account, sign in. You do not need to use the same email address you used before, but it might be a good idea to use the same one to keep things simple.

Step 2.3: Create an SSH Key

An SSH key is a cryptographically secure identifier. It's like a really long password used to identify your machine. GitHub uses SSH keys to allow you to upload to your repository without having to type in your username and password every time.

First, we need to see if you have an Ed25519 algorithm SSH key already installed. Type this into the terminal and check the output with the information below:

```
ls ~/.ssh/id_ed25519.pub
```

If a message appears in the console containing the text "No such file or directory", then you do not yet have an Ed25519 SSH key, and you will need to create one. If no such message has appeared in the console output, you can proceed to step 2.4.

To create a new SSH key, run the following command inside your terminal. The `-C` flag followed by your email address ensures that GitHub knows who you

are.

Note: The angle brackets (`< >`) in the code snippet below indicate that you should replace that part of the command with the appropriate information. Do not include the brackets themselves in your command. For example, if your email address is `odin@theodinproject.com`, then you would type `ssh-keygen -t ed25519 -C odin@theodinproject.com`. You will see this convention of using angle brackets to indicate placeholder text used throughout The Odin Project's curriculum and other coding websites, so it's good to be familiar with what it means.

```
ssh-keygen -t ed25519 -C <youremail>
```

- When it prompts you for a location to save the generated key, just push `Enter`.
- Next, it will ask you for a password; enter one if you wish, but it's not required.

Step 2.4: Link Your SSH Key with GitHub

Now, you need to tell GitHub what your SSH key is so that you can push your code without typing in a password every time.

First, you'll navigate to where GitHub receives our SSH key. Log into GitHub and click on your profile picture in the top right corner. Then, click on `Settings` in the drop-down menu.

Next, on the left-hand side, click `SSH and GPG keys`. Then, click the green button in the top right corner that says `New SSH Key`. Name your key something that is descriptive enough for you to remember where it came from. Leave this window open while you do the next steps.

Now you need to copy your public SSH key. To do this, we're going to use a command called `cat` to read the file to the console. (Note that the `.pub` file extension is important in this case.)

```
cat ~/.ssh/id_ed25519.pub
```

Highlight and copy the output, which starts with `ssh-ed25519` and ends with your email address.

Now, go back to GitHub in your browser window and paste the key you copied into the key field. Then, click `Add SSH key`. You're done! You've successfully added your SSH key!

Step 2.5 Testing your key

Follow the directions in this article from GitHub to verify your SSH connection **(Don't forget to omit the `$` when you copy and paste the code!)**. You should see this response in your terminal: **Hi username! You've successfully authenticated, but GitHub does not provide shell access.** Don't let GitHub's lack of providing shell access trouble you. If you see this message, you've successfully added your SSH key and you can move on. If the output doesn't correctly match up, then try going through these steps again or come to the Discord chat to ask for help.

Step 3: Let us know how it went!

You've completed the basic installations section, good job! As you progress through the Paths there will be other tools to install, so keep an eye out!

You probably felt like you were way in over your head, and you probably didn't understand much of what you were doing. That's 100% normal. Hang in there.

You can do this! And we've got your back.

Additional Resources

This section contains helpful links to related content. It isn't required, so consider it supplemental.

- Understanding SSH Key Pairs SSH is a secure network protocol that uses an implementation of public-key cryptography, also known as asymmetric cryptography. Having a basic understanding of how it works can help you understand what an SSH key is all about.
- Asymmetric Encryption - Simply explained a short video explaining Asymmetric Encryption.

View Course

Login to track progress

Next Lesson

Improve this lesson on GitHub

Have a question?

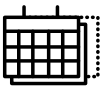
Chat with our friendly Odin community in our Discord chatrooms!

Open Discord

Are you interested in accelerating your web development learning experience?

Get started

Thinkful



5-6 months

Job Guarantee

1-on-1 Mentorship