Setting up Git

Step One: Sign Up for GitHub

Here comes the easy part: make yourself a GitHub account signing up on the front page. After completing the form, GitHub will sign you in and take you to your empty news feed. In the middle of the page, you'll see the boot camp (pictured to the right). We're going to go through it to set up your account and, later, create your first repository. Click on "Set Up Git" to get started.

Step Two: Install Git

Download Git

GitHub exists because of a version control application called git. The site is based around how git works, and git is pretty old. It runs via the command line and has no fancy graphical user interface. Since it's made to manage code you wrote, this shouldn't sound too scary. (Of course, as previously mentioned, GitHub did make wonderful software to allow you to use their service without the command line but that won't help you too much unless you know the basics.)

Git works by reading a local code repository (just a folder containing code for your project) on your computer and the mirroring that code elsewhere (in this case, GitHub's servers). Initially we'll commit (i.e. send) your entire local repository to GitHub, but that's just a one-time affair. As you continue to work on your code, you'll simply commit changes. GitHub will then keep track of the changes you made, creating different versions of files so you can revert back to old ones if you want (or just keep track of those changes for other reasons). This is primarily why you'd want to use a version control system like git on your own, but additional benefits surface when using git to manage code with other people working on your project. When multiple developers commit code with git, GitHub becomes a central repository where all the code that everyone's working on can stay in sync. You'll commit your changes, and other developers will pull them (i.e. sync them to their local repository). You'll do the same with their code.

Git makes this all happen, so you need to download the latest version and install it. On OS X, you'll just install the command line app. On Windows, you'll get a few more items. We'll discuss how they work in the next step.

Step Three: Set Up Git

To set up git, you need to make your way into the command line. On OS X, that means launching the Terminal app (Hard Drive -> Applications -> Utilities -> Terminal) and on Windows that means launching the Git Bash app you just installed—not the Windows command prompt. When you're ready, tell git your name like this:

```
git config --global user.name "Your Name Here"
```

For example, mine would look like this because I'm using a test account for this example:

```
git config --global user.name "Adam Dachis"
```

You can put in any name you like, but afterwards you'll need to input your email and that email must be the email you used when signing up for GitHub:

```
git config --global user.email "your_email@youremail.com"
```

If, for whatever reason, you signed up for GitHub with the wrong email address, you'll need to change it.

Step Four: Create Your First Repository

Now that you've made it this far, you can actually use GitHub! As a first order of business, we're going to create a repository (or "repo" for short). Head on over to GitHub and click the "New Repository" button on the top right of your account page. (Note: If you're still displaying the GitHub bootcamp section, it'll show up underneath it.)

When creating a repository you have a few things to decide including it's name and whether it'll be publicly accessible or not. Choosing a name should be pretty simple because you likely already have a name for your project. If you're just following along for learning purposes, use "Hello-World." Why "Hello-World" and not "Hello World"? Because spaces and special characters will cause problems. Keep it simple and easy to type in the command line. If you want to include a more complex name, you can add it to the optional description field beneath the name field.

If you're creating an open-source project, you want a public repository. If you want to code by yourself or share only with specific people, a private repository will do. Make the choice that works best for you and your project.

When you're all done, you can click the "Create repository" button but you might want to do one other thing first: check the "Initialize this repository with a README" checkbox. Why? All repositories require a README file. Ideally that file would contain a little information about your project, but you might not want to deal with that right now. By initializing the repository with a README, you'll get an empty README file that you can just deal with later. For the purposes of this tutorial, we're going to leave the box unchecked because, in the next section, we're going to create a README file from scratch to practice committing (sending) it to GitHub.

Step Five: Make Your First Commit

When you send files to GitHub, you commit them. To practice, we're going to initialize your local repository and create a README file to commit as practice. Before you start, you need to know where your local code repository is on your computer and how to access it via the command line. In this tutorial, we're going to assume there's a directory called "Hello-World" in your computer's home folder. If you need to create one, just run this command (same for Git Bash on Windows and OS X's terminal):

```
mkdir ~/Hello-World
```

Now change to that directory using the cd (change directory) command:

```
cd ~/Hello-World
```

In case you were wondering, the ~ represents your home directory in Git Bash and Terminal. It's simply shorthand so you don't have to type it all out (which would look more like /Users/yourusername/). Now that your repository is ready, type this:

```
git init
```

If you already had a repository ready to go, you'd just need to cd to that directory and then run the git init command in there instead. Either way, your local repository is ready to go and you can start committing code. But wait, you don't have anything to commit! Run this command to create a README file:

```
touch README
```

Let's take a break for a second and see what just happened. Go into the home folder on your computer and look at the Hello-World folder (or look at whatever folder you're using for a local repository). You'll notice a README file inside, thanks to the command you just ran. What you won't see is a .git folder, but that's because it's invisible. Git hides it in there, but because you ran the git init command you know it exists. If you're skeptical, just run the ls command in Git Bash/Terminal to display a list of everything in the current directory (which, if you're following along, is your local repository).

So how does git know we want to commit this README file we just created? It doesn't, and you have to tell it. This command will do the trick:

```
git add README
```

If you want to add other files to commit, you'll use the same command but replace README with the name of a different file. Now, run this command to commit it:

```
git commit -m 'first commit'
```

While the other commands were pretty straightforward, the commit command has a little more going on so let's break it down. When you type git, that's just telling the command line that you want to use the git program. When you type commit, you're telling git you want to use the commit command. Everything that follows those two thing count as options. The first, -m, is what's known as a flag. A flag specifies that you want to do something special rather than just run the commit command. In this case, the -m flag means "message" and what follows it is your commit message (in the example, 'first commit'). The message isn't absolutely necessary (although you'll usually need to provide one), but simply a reference to help you differentiate the various versions of a file (or files) you commit to your repository.

Your first commit should go by in a split second because you haven't actually uploaded anything yet. To get this empty README file to GitHub, you need to push it with a couple of commands. Here's the first:

```
git remote add origin https://github.com/yourusername/Hello-World.git
```

You need to replace "yourusername" with—you guessed it—your GitHub username. For me, it'd look like this:

```
git remote add origin https://github.com/gittest1040/Hello-World.git
```

This command tells git where to send your Hello-World repository. Now all you need to do is send it:

```
git push origin master
```

Once you run that command, everything (in this case, just your README file) will make it's way over to GitHub. Congratulations on your first commit!

Learning More

Using GitHub requires more than just committing a README file, but these basics should give you a good grasp on how to interact with the git app and the service. Now that you know how GitHub works at its core, you can use the GitHub apps to manage your code instead if you prefer. If you want to learn more about GitHub, there are some great tutorials. For starters, take a look at how to fork a repository and LockerGnome's GitHub guide.

Note: http://lifehacker.com/5983680/how-the-heck-do-i-use-github