HACK <HTTP://WWW.READWRITE.COM/HACK>

GitHub For Beginners: Don't Get Scared, Get Started

GitHub is more than just a programmer's tool. If you want to collaborate on anything, you should give it a try. Part 1 of a two-part look at getting started with GitHub.

Lauren Orsini <http://www.readwrite.com/author/lauren-orsini> *on* September 30, 2013

It's 2013, and there's no way around it: you need to learn how to use GitHub.

Why? Because it's a social network that has completely changed the way we work. Having started as a developer's collaborative platform, GitHub is now the largest online storage space of collaborative works that exists in the world. Whether you're interested in participating in this global mind meld or in researching this massive file dump of human knowledge, you need to be here.

See also: GitHub For Beginners: Commit, Push And Go
<http://readwrite.com/2013/10/02/gi
for-beginners-part-2>

Simply by being a member, you can brush elbows with the likes of **Google** <https://github.com/google> and **Facebook** <https://github.com/facebook> . Before GitHub existed, major companies created their knowledge mainly in private. But when you access their GitHub accounts, you're free to download, study, and build upon anything they add to the network. So what are you waiting for?

Looking For GitHub Answers

As embarrassing as it is to admit, this tutorial came into being because all of the “GitHub for Beginners” articles I read were way over my head. That’s probably because I don’t have a strong programming background, like most

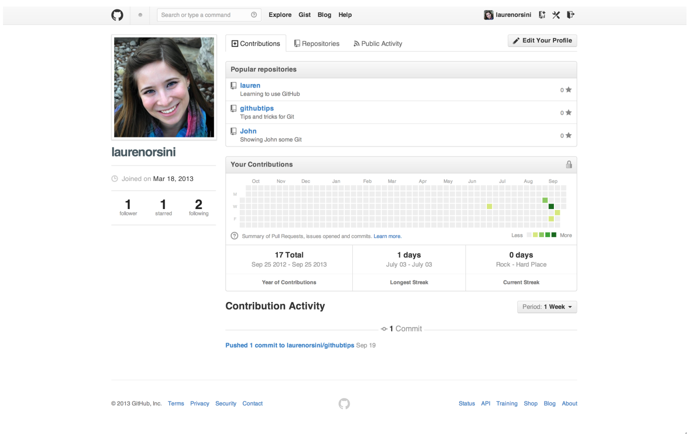
GitHub users. I couldn't identify with the way most tutorials suggest using GitHub, as a showcase for my programming work.

See also: Github's Tom Preston-Werner: How We Went Mainstream
<http://readwrite.com/2013/11/18/git
tom-preston-warner>

What you might not know is that there are plenty of reasons to use GitHub if you're not a programmer. According to GitHub's educational videos, any knowledge worker can benefit, with "knowledge worker" defined as most any profession that makes use of a computer.

If you've given up on understanding how to use GitHub, this article is for you.

One of the main misconceptions about GitHub is that it's a development tool, as much a part of coding as computer languages and compilers. However, GitHub itself isn't much more than a social network like Facebook or Flickr. You build a profile, upload projects to share and connect with other users by "following" their accounts. And while many users store programs and code projects, there's nothing preventing you from keeping text documents or other file types in your project folders to show off.



The author's GitHub page.

You may already have a dozen other social media accounts, but here's why you should be on GitHub anyway: it's got the best **Terms of Service** <https://help.github.com/articles/github-terms-of-service> agreement out of the bunch. If you check out Section F of the terms, you'll see that GitHub does everything in its power to ensure that you retain total ownership of any projects you upload to the site:

"We claim no intellectual property rights over the material you provide to the Service. Your profile and materials uploaded remain yours."

What's more, you can actually use GitHub without knowing ANY code at all. You don't really need a tutorial to sign up and click around. But I do think that there's merit to learning things the hard way first, by which I mean, with plain old coding in Git. After all, GitHub just happens to be one of the most effortless graphical interfaces for the Git programming language.

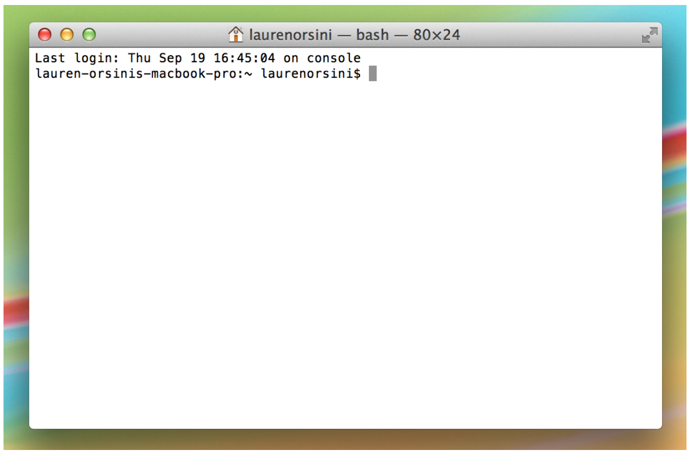
What Is Git?

Thank famed software developer **Linus Torvalds** <http://en.wikipedia.org/wiki/Linus_Torvalds> for Git, the software that runs at the heart of GitHub. (And while you're at it, go ahead thank him for the Linux operating system, too.) Git is version control software, which means it manages changes to a project without overwriting any part of that project. And it's not going away anytime soon, particularly since Torvalds and his fellow kernel developers employ Git to help develop the core kernel for Linux.

Why use something like Git? Say you and a coworker are both updating pages on the same website. You make your changes, save them, and upload them back to the website. So far, so good. The problem comes when your coworker is working on the same page as you at the same time. One of you is about to have your work overwritten and erased.

A **version control** <http://git-scm.com/video/what-is-version-control> application like Git keeps that from happening. You and your coworker can each upload your revisions to the same page, and Git will save two copies. Later, you can merge your changes together without losing any work along the way. You can even revert to an earlier version at any time, because Git keeps a "snapshot" of every change ever made.

The problem with Git is that it's so ancient that we have to use the command line—or Terminal if you're a Mac user—in order to access it, typing in snippets of code like '90s hackers. This can be a difficult proposition for modern computer users. That's where GitHub comes in.



```
Last login: Thu Sep 19 16:45:04 on console
lauren-orsinis-macbook-pro:~ laurenorsini$ 
```

The author's Terminal screen on a Mac.

GitHub makes Git easier to use in two ways. First, if you **download the**

GitHub software <http://github.com/> to your computer, it provides a visual interface to help you manage your version-controlled projects locally. Second, creating an account on GitHub.com brings your version-controlled projects to the Web, and ties in social network features for good measure.

You can browse other GitHub users' projects, and even download copies for yourself to alter and learn from. Other users can do the same with your public projects, and even spot errors and suggest fixes. Either way, no data is lost because Git saves a "snapshot" of every change.

While it's possible to use GitHub without learning Git, there's a big difference between using and understanding. Before I figured out Git I could use GitHub, but I didn't really understand why. In this tutorial, we're going to learn to use Git on the command line.

Words People Use When They Talk About Git

In this tutorial, there are a few words I'm going to use repeatedly, none of which I'd heard before I started learning. Here's the big ones:

Command Line: The computer program we use to input Git commands. On a Mac, it's called Terminal. On a PC, it's a non-native program that you download when you download Git for the first time (we'll do that in the next section). In both cases, you type text-based commands, known as prompts, into the screen, instead of using a mouse.

Repository: A directory or storage space where your projects can live. Sometimes GitHub users shorten this to "repo." It can be local to a folder on your computer, or it can be a storage space on GitHub or another online host. You can keep code files, text files, image files, you name it, inside a repository.

Version Control: Basically, the purpose Git was designed to serve. When you have a Microsoft Word file, you either overwrite every saved file with a new save, or you save multiple versions. With Git, you don't have to. It keeps "snapshots" of every point in time in the project's history, so you can never lose or overwrite it.

Commit: This is the command that gives Git its power. When you commit, you are taking a "snapshot" of your repository at that point in time, giving you a checkpoint to which you can reevaluate or restore your project to any previous state.

Branch: How do multiple people work on a project at the same time without Git getting them confused? Usually, they "branch off" of the main project with their own versions full of changes they themselves have made. After they're done, it's time to "merge" that branch back with the "master," the main directory of the project.

Git-Specific Commands

Since Git was designed with a big project like Linux in mind, there are a lot of Git commands. However, to use the basics of Git, you'll only need to know a few terms. They all begin the same way, with the word "git."

`git init`: Initializes a new Git repository. Until you run this command inside a repository or directory, it's just a regular folder. Only after you input this does it accept further Git commands.

`git config`: Short for "configure," this is most useful when you're setting up Git for the first time.

`git help`: Forgot a command? Type this into the command line to bring up the 21 most common git commands. You can also be more specific and type "git help init" or another term to figure out how to use and configure a specific git command.

`git status`: Check the status of your repository. See which files are inside it, which changes still need to be committed, and which branch of the repository you're currently working on.

`git add`: This does *not* add new files to your repository. Instead, it brings new files to Git's attention. After you add files, they're included in Git's "snapshots" of the repository.

`git commit`: Git's most important command. After you make any sort of change, you input this in order to take a "snapshot" of the repository. Usually it goes `git commit -m "Message here."` The `-m` indicates that the following section of the command should be read as a message.

`git branch`: Working with multiple collaborators and want to make changes on your own? This command will let you build a new branch, or timeline of commits, of changes and file additions that are completely your own. Your title goes after the command. If you wanted a new branch called "cats," you'd type `git branch cats`.

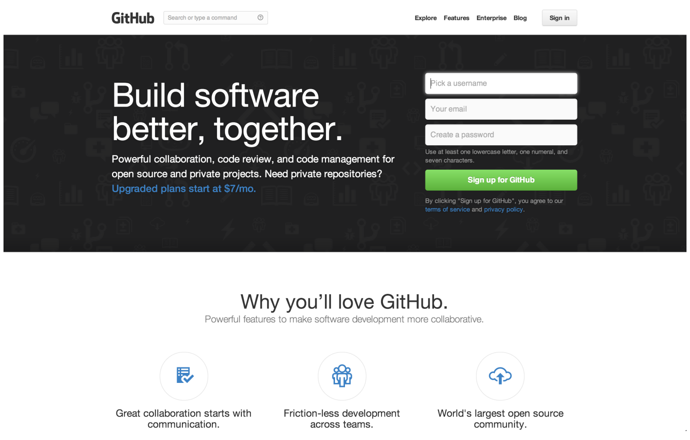
`git checkout`: Literally allows you to "check out" a repository that you are not currently inside. This is a navigational command that lets you move to the repository you want to check. You can use this command as `git checkout master` to look at the master branch, or `git checkout cats` to look at another branch.

`git merge`: When you're done working on a branch, you can merge your changes back to the master branch, which is visible to all collaborators. `git merge cats` would take all the changes you made to the "cats" branch and add them to the master.

`git push`: If you're working on your local computer, and want your commits to be visible online on GitHub as well, you "push" the changes up to GitHub with this command.

`git pull`: If you're working on your local computer and want the most up-to-date version of your repository to work with, you "pull" the changes down from GitHub with this command.

Setting Up GitHub And Git For The First Time


GitHub Search or type a command Explore Features Enterprise Blog Sign in

Build software better, together.

Powerful collaboration, code review, and code management for open source and private projects. Need private repositories? Upgraded plans start at $7/mo.

Pick a username
Your email
Create a password
Use at least one lowercase letter, one numeral, and seven characters.
Sign up for GitHub
By clicking "Sign up for GitHub", you agree to our terms of service and privacy policy.

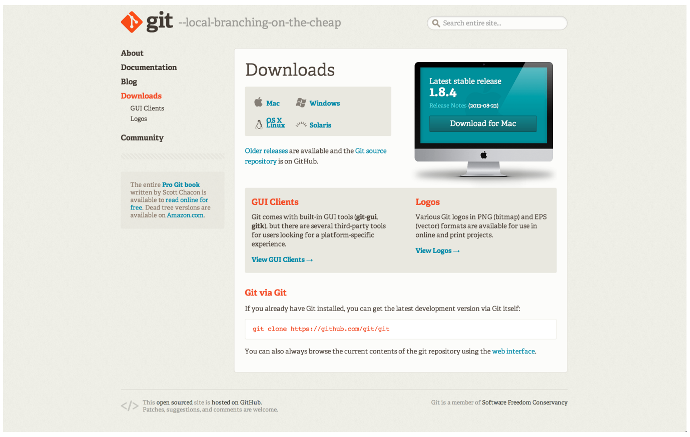
Why you'll love GitHub.
Powerful features to make software development more collaborative.

Great collaboration starts with communication.
Friction-less development across teams.
World's largest open source community.

GitHub's signup page.

First, you'll need to **sign up for an account** <https://github.com/> on GitHub.com. It's as simple as signing up for any other social network. Keep the email you picked handy; we'll be referencing it again soon.

You could stop there and GitHub would work fine. But if you want to work on your project on your local computer, you need to have Git installed. In fact, GitHub won't work on your local computer if you don't install Git. Install Git for Windows, Mac or Linux **as needed** <http://git-scm.com/downloads> .


git --local-branching-on-the-cheap Search entire site...

About
Documentation
Blog
Downloads
GUI Clients
Logos
Community

The entire Pro Git book written by Scott Chacon is available to read online for free. Dead tree versions are available on Amazon.com.

Downloads

Mac Windows Linux Solaris

Older releases are available and the Git source repository is on GitHub.

Latest stable release 1.8.4
Release Notes (2013-08-23)
Download for Mac

GUI Clients
Git comes with built-in GUI tools (git-gui, gitk), but there are several third-party tools for users looking for a platform-specific experience.
View GUI Clients →

Logos
Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.
View Logos →

Git via Git
If you already have Git installed, you can get the latest development version via Git itself:

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

You can also always browse the current contents of the git repository using the web interface.

This open sourced site is hosted on GitHub. Patches, suggestions, and comments are welcome.
Git is a member of Software Freedom Conservancy

http://git-scm.com/, where you download Git.

Now it's time to go over to the command line. On Windows, that means starting the Git Bash app you just installed, and on OS X, it's regular old Terminal. It's time to introduce yourself to Git. Type in the following code:

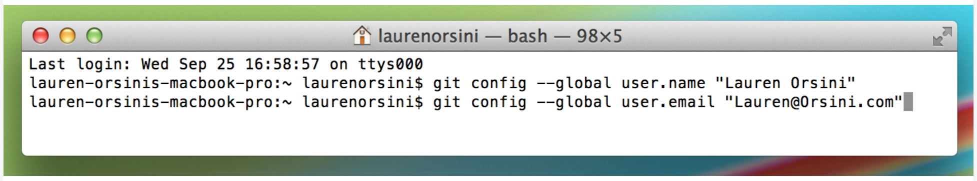
```
git config --global user.name "Your Name
Here"
```

Of course, you'll need to replace "Your Name Here" with your own name in quotations. It can be your legal name, your online handle, anything. Git doesn't care, it just needs to know to whom to credit commits and future projects.

Next, tell it your email and make sure it's the same email you used when you signed up for a GitHub.com account just a moment ago. Do it like this:

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

That's all you need to do to get started using Git on your computer. However, since you did set up a GitHub.com account, it's likely you don't just want to manage your project locally, but also online. If you want you can also set up Git so it doesn't ask you to log in to your GitHub.com account every time you want to talk to it. For the purposes of this tutorial, it isn't a big deal since we'll only be talking to it once. The full tutorial to do this, however, is **located on GitHub** <https://help.github.com/articles/set-up-git> .


laurenorsini — bash — 98×5

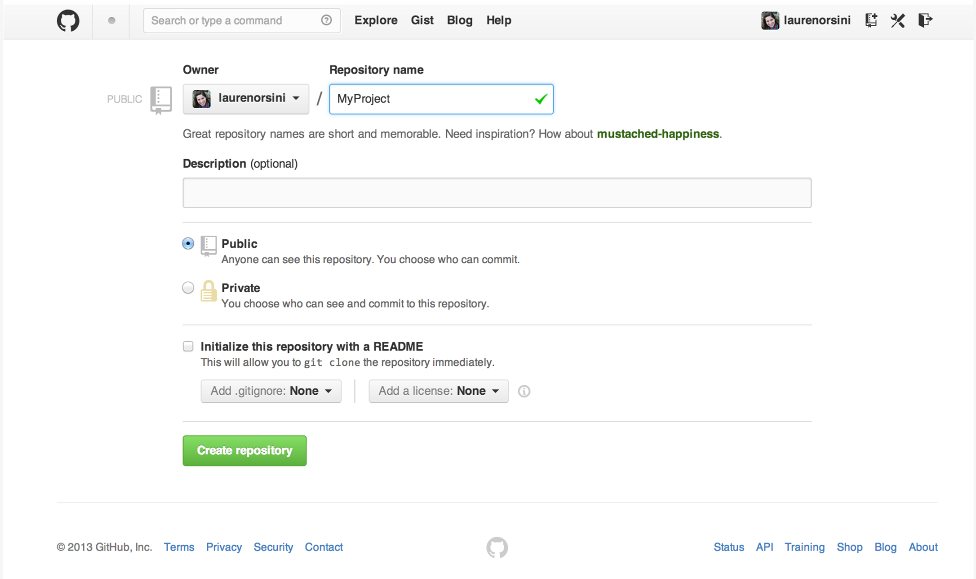
```
Last login: Wed Sep 25 16:58:57 on ttys000
lauren-orsinis-macbook-pro:~ laurenorsini$ git config --global user.name "Lauren Orsini"
lauren-orsinis-macbook-pro:~ laurenorsini$ git config --global user.email "Lauren@Orsini.com"
```

Baby's first Git commands.

Creating Your Online Repository

Now that you're all set up, it's time to create a place for your project to live. Both Git and GitHub refer to this as a repository, or "repo" for short, a digital directory or storage space where you can access your project, its files, and all the versions of its files that Git saves.

Go back to GitHub.com and click the tiny book icon next to your username. Or, go to the **new repository page** <https://github.com/new> if all the icons look the same. Give your repository a short, memorable name. Go ahead and make it public just for kicks; why hide your attempt to learn GitHub?



PUBLIC

Owner: laurenorsini / Repository name: MyProject

Great repository names are short and memorable. Need inspiration? How about **mustached-happiness**.

Description (optional)

● **Public**
Anyone can see this repository. You choose who can commit.

○ **Private**
You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**
This will allow you to `git clone` the repository immediately.

Add .gitignore: **None** | Add a license: **None**

Create repository

© 2013 GitHub, Inc. Terms Privacy Security Contact Status API Training Shop Blog About

Creating a new repository on GitHub.

Don't worry about clicking the checkbox next to "Initialize this repository with a README." A Readme file is usually a text file that explains a bit about the project. But we can make our own Readme file locally for practice.

Click the green "Create Repository" button and you're set. You now have an online space for your project to live in.

Creating Your Local Repository

So we just made a space for your project to live online, but that's not where you'll be working on it. The bulk of your work is going to be done on your computer. So we need to actually mirror that repository we just made as a local directory.

This—where we do some heavy command line typing—is the part of every Git tutorial that really trips me up, so I'm going to go tediously, intelligence-insultingly slow.

First type:

```
mkdir ~/MyProject
```

`mkdir` is short for make directory. It's not actually a Git command, but a general navigational command from the time before visual computer interfaces. The `~/` ensures that we're building the repository at the top level of your computer's file structure, instead of stuck inside some other directory that would be hard to find later. Actually, if you type `~/` into your browser window, it'll bring up your local computer's top level directory. For me, using Chrome on a Mac, it displays my Users folder.

Also, notice that I called it MyProject, the very same name I called my GitHub repository that we made earlier. Keep your name consistent, too.

Next, type:

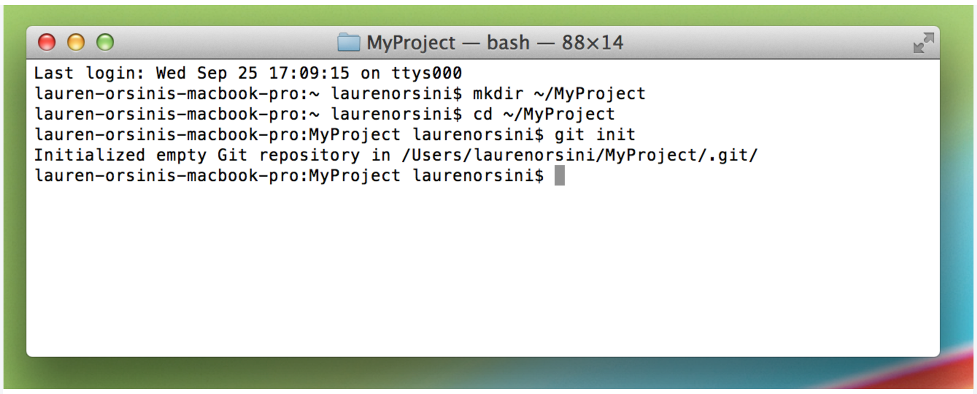
```
cd ~/MyProject
```

`cd` stands for change directory, and it's also a navigational command. We just made a directory, and now we want to switch over to that directory and go inside it. Once we type this command, we are transported inside MyProject.

Now we're finally using a Git command. For your next line, type:

```
git init
```

You know you're using a Git command because it always begins with `git`. `init` stands for "initialize." Remember how the previous two commands we typed were general command-line terms? When we type this code in, it tells the computer to recognize this directory as a local Git repository. If you open up the folder, it won't look any different, because this new Git directory is a hidden file inside the dedicated repository.



```
Last login: Wed Sep 25 17:09:15 on ttys000
lauren-orsinis-macbook-pro:~ laurenorsini$ mkdir ~/MyProject
lauren-orsinis-macbook-pro:~ laurenorsini$ cd ~/MyProject
lauren-orsinis-macbook-pro:MyProject laurenorsini$ git init
Initialized empty Git repository in /Users/laurenorsini/MyProject/.git/
lauren-orsinis-macbook-pro:MyProject laurenorsini$
```

Creating a local Git repository in three steps.

However, your computer now realizes this directory is Git-ready, and you can start inputting Git commands. Now you've got both an online and a local repo for your project to live inside. In **Part 2 of this series** <http://readwrite.com/2013/10/02/github-for-beginners-part-2> , you will learn how to make your first commit to local and GitHub repositories, and learn about more great GitHub resources.

(See also: GitHub For Beginners: Commit, Push And Go <http://readwrite.com/2013/10/02/github-for-beginners-part-2> **)**

#HACK <HTTP://WWW.READWRITE.COM/HACK>

#GITHUB <HTTP://WWW.READWRITE.COM/TAG/GITHUB>

#HOW-TO <HTTP://WWW.READWRITE.COM/TAG/HOW-TO>

#VERSION CONTROL <HTTP://WWW.READWRITE.COM/TAG/VERSION+CONTROL>