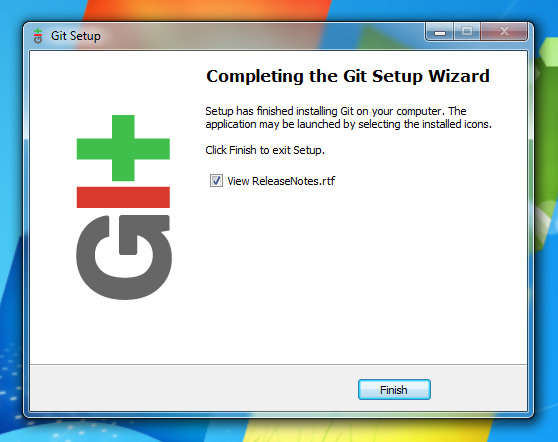
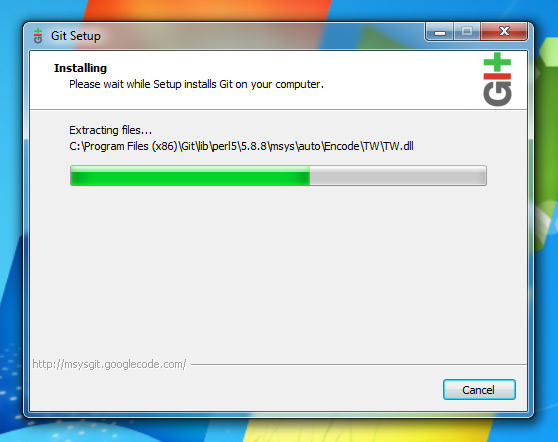
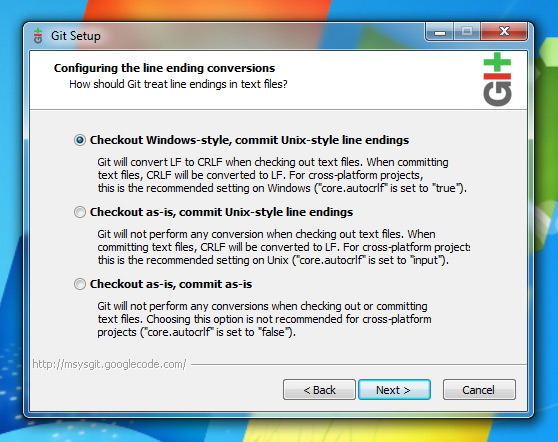
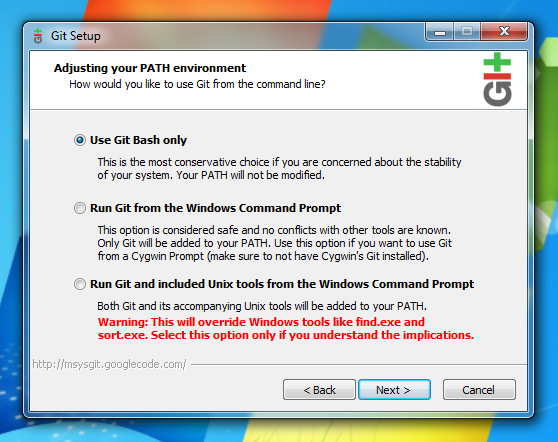
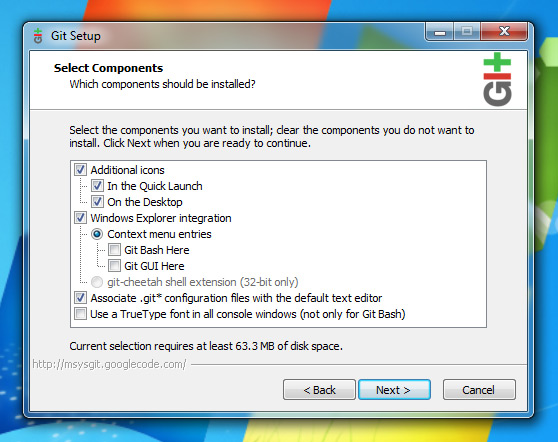
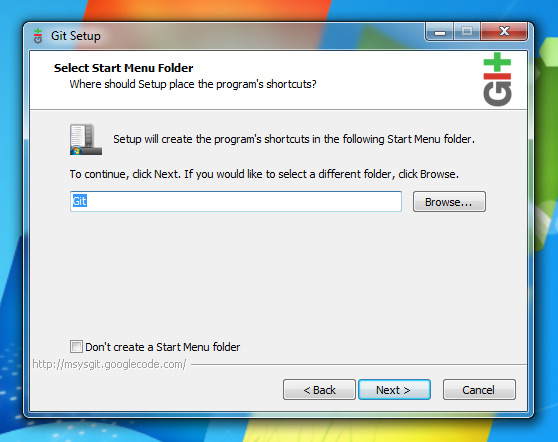
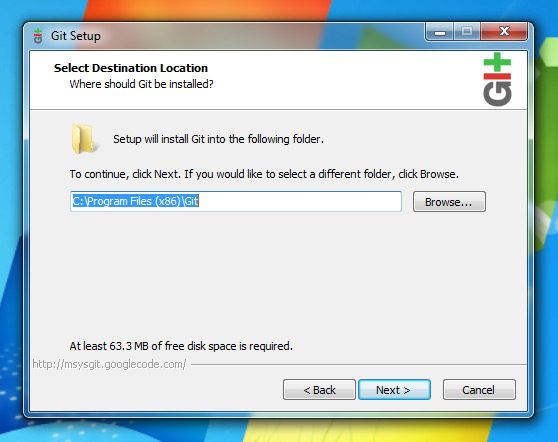
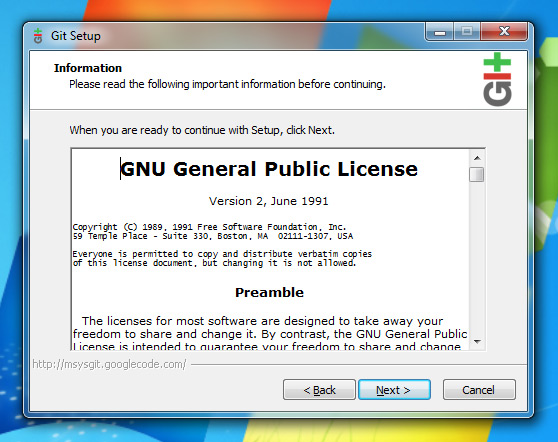
# Set Up Git

At the heart of GitHub is an open source version control system (VCS) called Git\*. Created by the same dudes that created Linux, Git is responsible for everything GitHub related that happens locally on your computer.

\*If you don’t already know what Git is, [take a crash course.](http://progit.org/book/ch1-3.html)

### Download and install the latest version of [Git for Windows](http://code.google.com/p/msysgit/downloads/list).

Use the default options for each step.

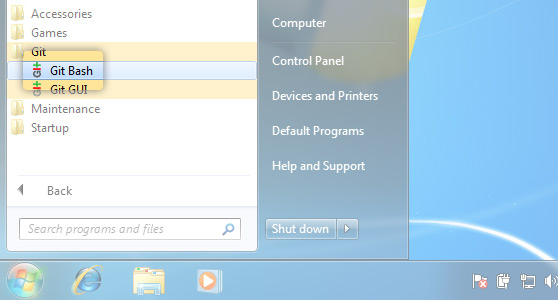


**Do not use PuTTY if you are given the option. GitHub only provides support for openssh.**

### Next: Set Up SSH Keys

We use SSH keys to establish a secure connection between your computer and GitHub. Setting them up is fairly easy, but does involve a number of steps.

To make sure you generate a brand new key, you need to check if one already exists. First, you need to open Git Bash (not the Windows command line), found in the Start Menu in the “git” folder.



1. **Check for SSH keys.**Have an existing key pair? You can skip to Step 4.

First, we need to check for existing ssh keys on your computer:

$ cd ~/.ssh

If it says “No such file or directory“ skip to **step 3**. Otherwise continue to **step 2**.

1. **Backup and remove existing SSH keys.**

Since there is already an SSH directory you’ll want to back the old one up and remove it:

$ ls

config id\_rsa id\_rsa.pub known\_hosts

$ mkdir key\_backup

$ cp id\_rsa\* key\_backup

$ rm id\_rsa\*

1. **Generate a new SSH key.**

To generate a new SSH key, enter the code below. We want the default settings so when asked to enter a file in which to save the key, just press enter.

**$ ssh-keygen -t rsa -C “your\_email@youremail.com”**

Generating public/private rsa key pair.Enter file in which to save the key (/Users/your\_user\_directory/.ssh/id\_rsa):<press enter>

Now you need to enter a passphrase.

Enter passphrase (empty for no passphrase):<enter a passphrase>

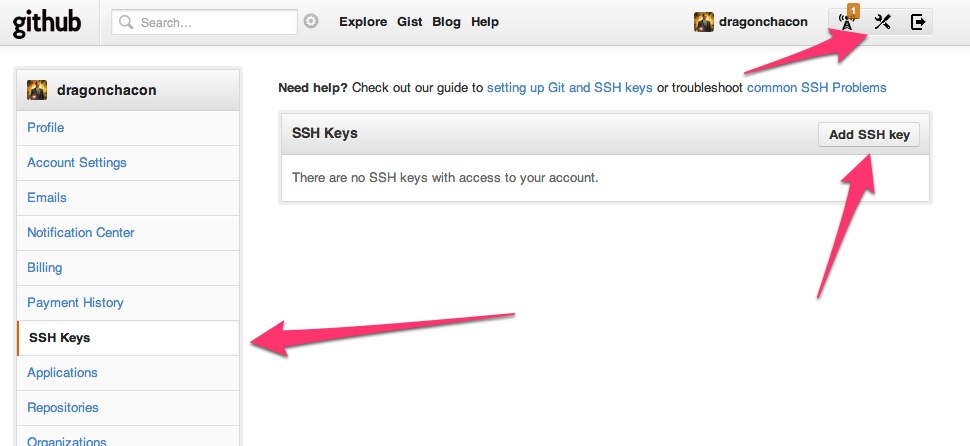
Enter same passphrase again:<enter passphrase again>

Which should give you something like this:

Your identification has been saved in /Users/your\_user\_directory/.ssh/id\_rsa.Your public key has been saved in /Users/your\_user\_directory/.ssh/id\_rsa.pub.The key fingerprint is:01:0f:f4:3b:ca:85:d6:17:a1:7d:f0:68:9d:f0:a2:db user\_name@username.com

1. **Add your SSH key to GitHub.**

On the GitHub site Click “Account Settings” > Click “SSH Keys” > Click “Add SSH key”

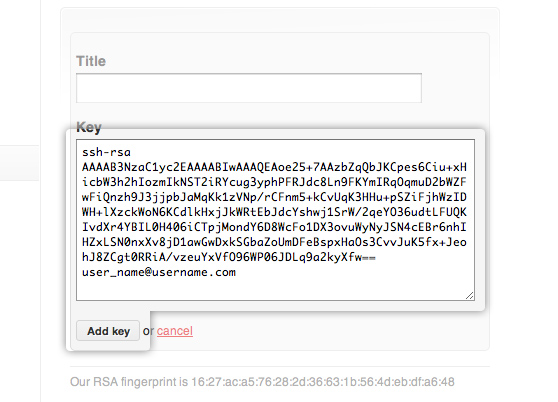


Open the id\_rsa.pub file with a text editor (Notepad, TextEdit, or gedit will do just fine). This is your public SSH key. You may need to turn on “view hidden files” to find it because the .ssh directory is hidden. It’s important you copy your SSH key exactly as it is written without adding any newlines or whitespace. Now paste it into the “Key” field.

#### Can’t view hidden files? Other ways to copy:

You can open Git Gui, go to Help > Show Key, and then press Copy To Clipboard to copy your public key to your clipboard

Now paste it into the “Key” field.



Hit “Add Key.”

1. **Test everything out.**

To make sure everything is working you’ll now SSH to GitHub. Don’t change the “git@github.com” part. That’s supposed to be there.

$ ssh -T git@github.com

Which should give you this:

The authenticity of host 'github.com (207.97.227.239)' can't be established.RSA key fingerprint is 16:27:ac:a5:76:28:2d:36:63:1b:56:4d:eb:df:a6:48.Are you sure you want to continue connecting (yes/no)?

Don’t worry, this is supposed to happen. Type “yes”.

Hi username! You've successfully authenticated, but GitHub does not provide shell access.

### Then: Set Up Your Info

Now that you have Git set up and your SSH keys entered into GitHub, it’s time to configure your personal info.

1. **Set your username and email.**

Git tracks who makes each commit by checking the user’s name and email. In addition, we use this info to associate your commits with your GitHub account. To set these, enter the code below, replacing the name and email with your own. The name should be your actual name, not your GitHub username.

$ git config --global user.name "Firstname Lastname"Sets the name of the user for all git instances on the system$ git config --global user.email "your\_email@youremail.com"Sets the email of the user for all git instances on the system

The steps listed above show you how to set your user info globally. This means that no matter which repo you work in on your computer, you’ll be making commits as that user. If you find yourself needing to make commits with different user info for a specific repo (perhaps for work vs. personal projects), you will have to change the info in that repo itself.

$ cd my\_other\_repo

$ git config user.name "Different Name"

$ git config user.email "differentemail@email.com"

Now your commits will be “blamed” on (associated with) new user name and email whenever working in the specified repo.

1. **Set your GitHub token.**

Previous versions of our API used token authentication, but we’re removing support for that soon, see our [GitHub API Moving On](https://github.com/blog/1090-github-api-moving-on) blog post for details.

If you are using a 3rd party tool that is asking for your token, you should contact the maintainer and ask them to update to our [latest API](http://developer.github.com/v3/) as the old API will stop working soon.

Please note you do not need the API Token to work with git.

# Create A Repo and manage

Every time you make a commit with Git, it is stored in a repository (a.k.a. “repo”). To put your project up on GitHub, you’ll need to have a GitHub repository for it to live in.

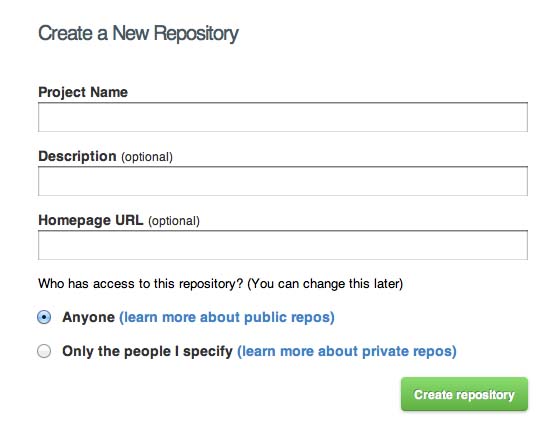
Git stores all of your project files in a repository. If you are able to view hidden files on your system, you’ll see a subdirectory called “.git” in the project directory where you ran git init. This is where Git stores all of your commits, as well as everything else it needs. In addition to your local repo, you can also have remote repositories (like GitHub repos). Remote repositories are the same as your local repo, but stored on a different server or computer for easy collaboration, backup, and general awesomeness.

### First: Create a new repo

Click [New Repository](https://github.com/repositories/new).



Fill out the information on this page. When you’re done, click “New Repository.”



Congratulations! You have successfully created your first repo!

### Next: Create a README for your repo.

While a README isn’t a required part of a GitHub repo, it is a good idea to have one. READMEs are a great place to describe your project or add some documentation such as how to install or use your project.

1. **Create the README file**

In the prompt, type the following code:

$ mkdir ~/Hello-World

$ cd ~/Hello-World

$ git init

Initialized empty Git repository in /Users/your\_user\_directory/Hello-World/.git/

$ touch README

Or you can init with another way.

$ mkdir ~/Hello-World

$ git clone git@github.com:username/Hello-World.git

$ cd ~/Hello-World

$ touch README

Open the new README file found in your Hello-World directory in a text editor and add the text “Hello World!” When you are finished, save and close the file.

1. **Commit your README**

Now that you have your README set up, it’s time to commit it. A commit is essentially a snapshot of all the files in your project at a particular point in time. In the prompt, type the following code:

Think of a commit as a snapshot of your project — code, files, everything — at a particular point in time. More accurately, after your first commit, each subsequent commit is only a snapshot of your changes. For code files, this means it only takes a snapshot of the lines of code that have changed. For everything else like music or image files, it saves a new copy of the file.

$ git add README

$ git commit -m 'first commit'

The code above executes actions locally, meaning you still haven’t done anything on GitHub yet. To connect your local repository to your GitHub account, you will need to set a remote for your repo and push your commits to it:

A remote is a repo stored on another computer, in this case on GitHub’s server. It is standard practice (and also the default in some cases) to give the name origin to the remote that points to your main offsite repo (for example, your GitHub repo).

Git supports multiple remotes. This is commonly used when forking a repo.

$ git remote add alias git@github.com:username/Hello-World.git

$ git push -u alias master

“alias” is a alias of git@github.com:username/Hello-World.git

“git push -u alias master” mean “git push -u git@github.com:username/Hello-World.git master”

Now if you look at your repository on GitHub, you will see your README has been added to it.

