github:help

- Git Reference
- Support
- Back to GitHub

Set Up Git

If you've found yourself on this page, we're assuming you're brand new to Git and GitHub. This guide will walk you through the basics and explain a little bit about how everything works along the way.

This is the guide for setting up git in **Windows**. There are also guides for **OSX** and **Linux**.

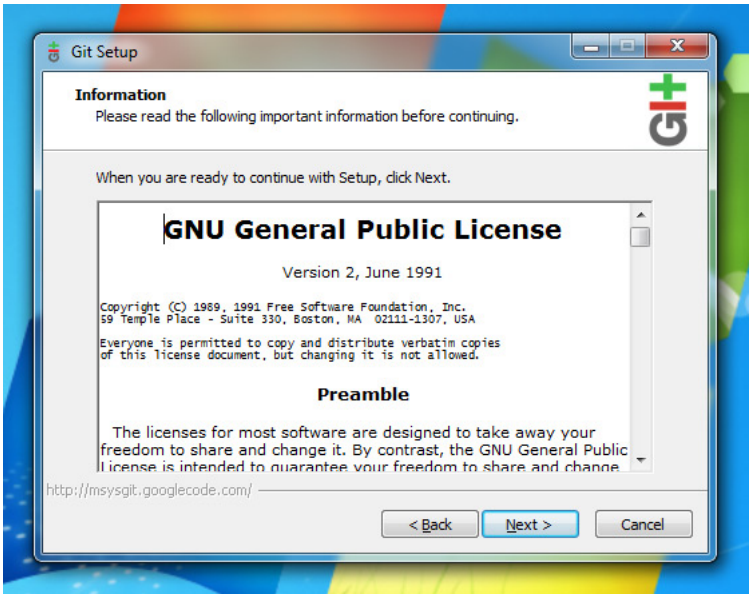
First: Download and Install Git

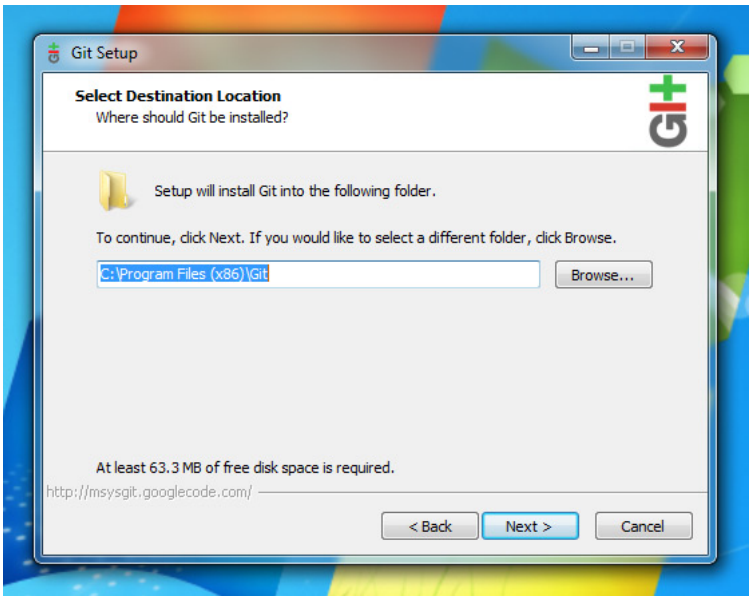
At the heart of GitHub is an open source version control system (VCS) called Git*. Created by the same dudes that created Linux, Git is responsible for everything GitHub related that happens locally on your computer.

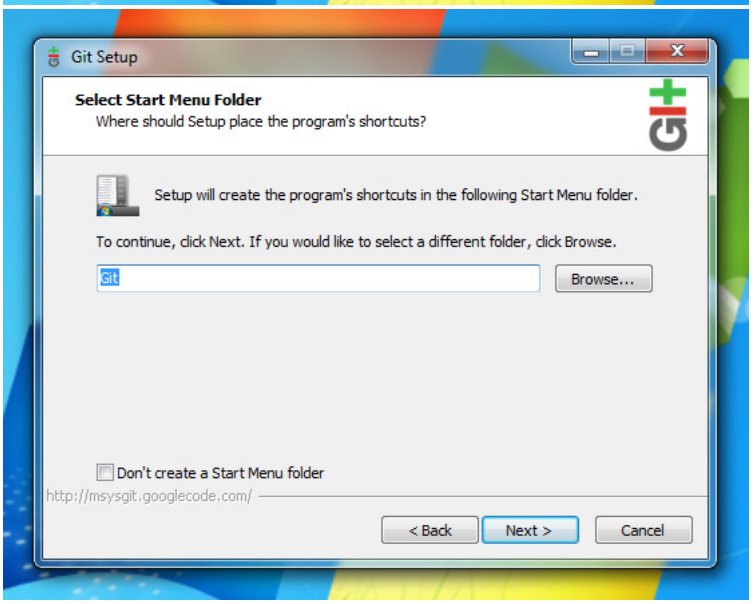
*If you don't already know what Git is, take a crash course.

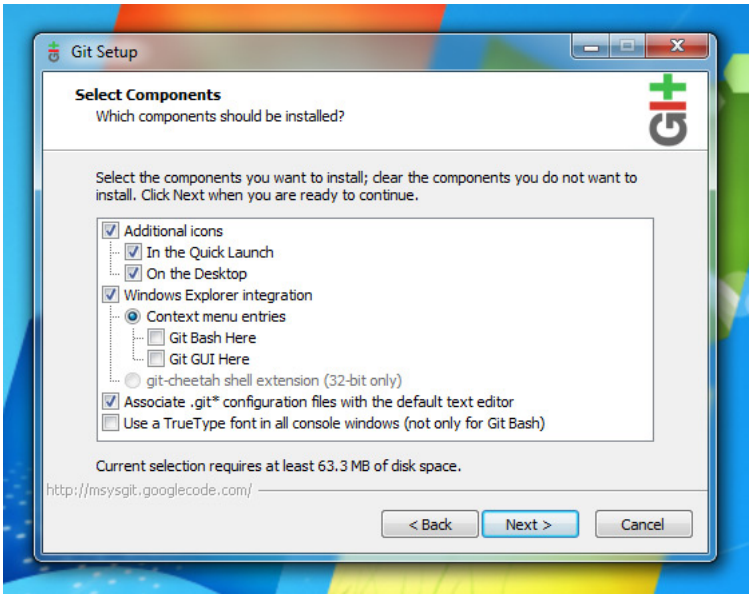
1. Download and install the latest version of Git for Windows.

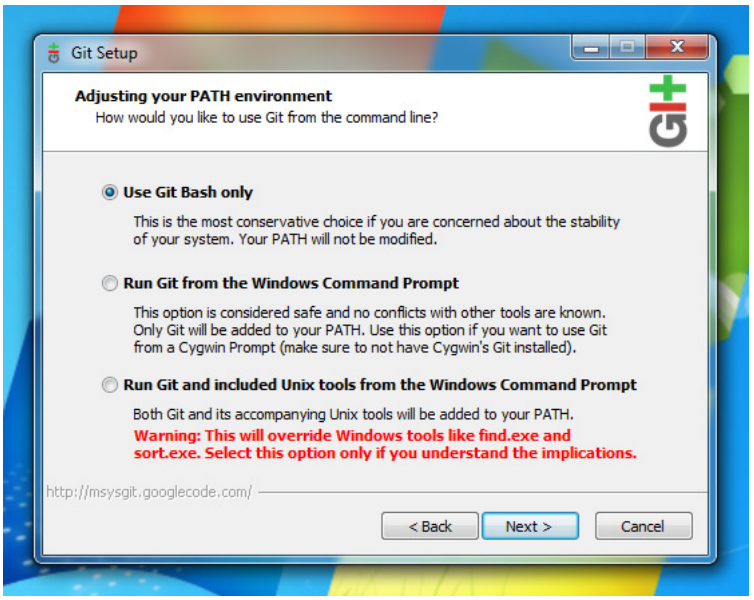
   Use the default options for each step.

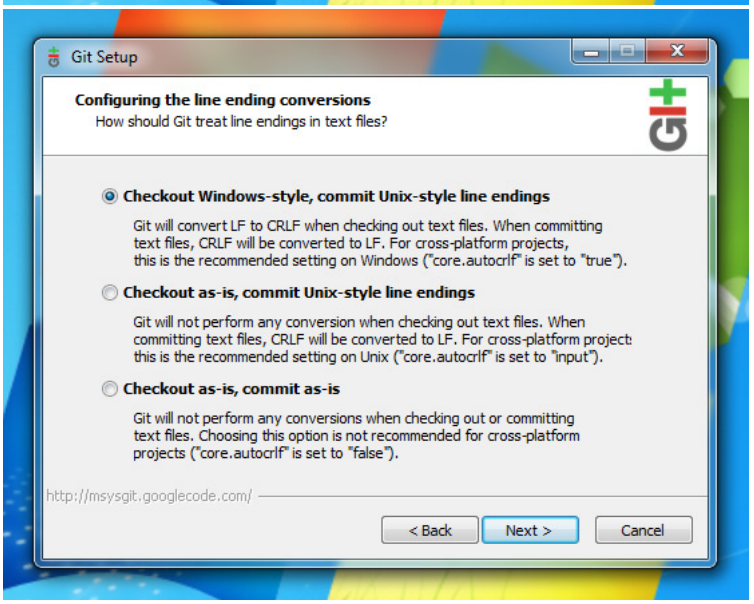


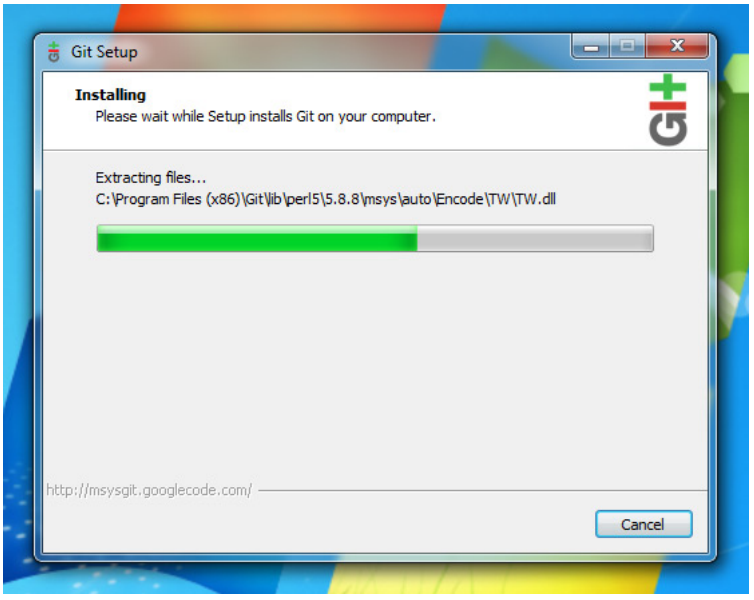

Git Setup
Select Destination Location
Where should Git be installed?
Setup will install Git into the following folder.
To continue, click Next. If you would like to select a different folder, click Browse.
C:\Program Files (x86)\Git
Browse...
At least 63.3 MB of free disk space is required.
http://msysgit.googlecode.com/
< Back
Next >
Cancel

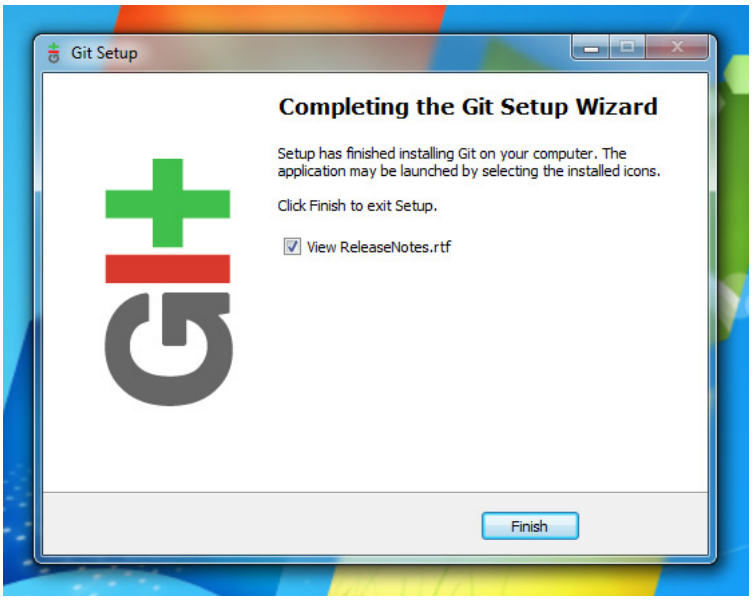

Git Setup
Select Start Menu Folder
Where should Setup place the program's shortcuts?
Setup will create the program's shortcuts in the following Start Menu folder.
To continue, click Next. If you would like to select a different folder, click Browse.
Git
Browse...
Don't create a Start Menu folder
http://msysgit.googlecode.com/
< Back
Next >
Cancel


Git Setup
Select Components
Which components should be installed?
Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.
Additional icons
In the Quick Launch
On the Desktop
Windows Explorer integration
Context menu entries
Git Bash Here
Git GUI Here
git-cheetah shell extension (32-bit only)
Associate .git* configuration files with the default text editor
Use a TrueType font in all console windows (not only for Git Bash)
Current selection requires at least 63.3 MB of disk space.
http://msysgit.googlecode.com/
< Back
Next >
Cancel


Git Setup
Adjusting your PATH environment
How would you like to use Git from the command line?
Use Git Bash only
This is the most conservative choice if you are concerned about the stability of your system. Your PATH will not be modified.
Run Git from the Windows Command Prompt
This option is considered safe and no conflicts with other tools are known. Only Git will be added to your PATH. Use this option if you want to use Git from a Cygwin Prompt (make sure to not have Cygwin's Git installed).
Run Git and included Unix tools from the Windows Command Prompt
Both Git and its accompanying Unix tools will be added to your PATH.
Warning: This will override Windows tools like find.exe and sort.exe. Select this option only if you understand the implications.
http://msysgit.googlecode.com/
< Back
Next >
Cancel


Git Setup
Configuring the line ending conversions
How should Git treat line endings in text files?
Checkout Windows-style, commit Unix-style line endings
Git will convert LF to CRLF when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Windows ("core.autocrlf" is set to "true").
Checkout as-is, commit Unix-style line endings
Git will not perform any conversion when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform project: this is the recommended setting on Unix ("core.autocrlf" is set to "input").
Checkout as-is, commit as-is
Git will not perform any conversions when checking out or committing text files. Choosing this option is not recommended for cross-platform projects ("core.autocrlf" is set to "false").
http://msysgit.googlecode.com/
< Back
Next >
Cancel


Git Setup
Installing
Please wait while Setup installs Git on your computer.
Extracting files...
C:\Program Files (x86)\Git\lib\perl5\5.8.8\msys\auto\Encode\TW\TW.dll
http://msysgit.googlecode.com/
Cancel

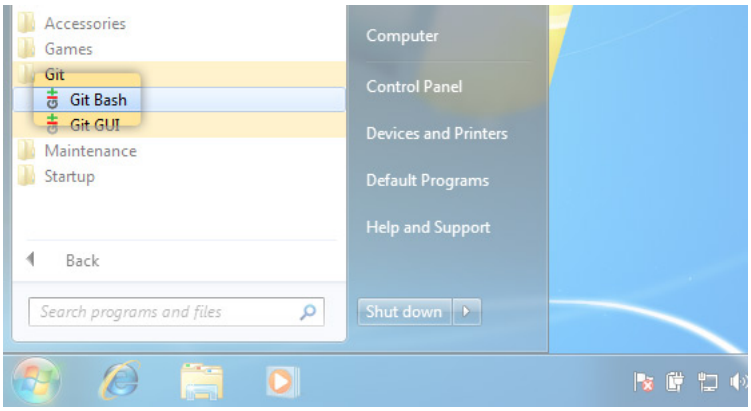


Do not use PuTTY if you are given the option. GitHub only provides support for openssh.

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.



Need a quick lesson about Git Bash?

Code blocks like those on this page are part of a scripting language called Bash. To use Bash scripts, we need to use an application that was installed with Git called Git Bash.

Input

```
$ echo 'This is input text'This tooltip tells you what's going on.
```

A line that begins with the dollar sign ($) indicates a line of Bash script you need to type. To enter it, type the text that follows the $, hitting the return key at the end of each line. You can hover your mouse over each line for an explanation of what the script is doing.

Output

```
This is output text.
```

A line that does not begin with a $ is output text that is intended to give you information or tell you what to do next. We've colored output text green in these bootcamp tutorials.

User Specific Input

```
$ echo 'username'Outputs the text in the quotation marks.
```

Areas of yellow text represent your own personal info, repos, etc. If it is part of an input ($) line, you should replace it with your own info when you type it. If it is part of output text, it is just for your reference. It will automatically show your own info in Git Bash.

Good to know: There will be times when you type code, hit return, and all you are given is another prompt. Some actions that you execute

in Git Bash don't have any output. Don't worry, if there is ever a problem with your code, Git Bash will let you know.

Good to know: For security reasons, Git Bash will not display what you type when entering passwords. Just type your password and hit the return key.

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 ~/.sshChecks to see if there is a directory named ".ssh" in your user directory
   ```

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

2. 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:

   ```
   $ lsLists all the subdirectories in the current directory
   config  id_rsa  id_rsa.pub      known_hosts
   $ mkdir key_backupmakes a subdirectory called "key_backup" in the current directory
   $ cp id_rsa* key_backupCopies the id_rsa and id_rsa.pub files into key_backup
   $ rm id_rsa*Deletes the id_rsa and id_rsa.pub files
   ```

3. 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"Creates a new ssh key using the provided email
   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.

   Why do passphrases matter?

   Passwords aren't very secure, you already know this. If you use one that's easy to remember, it's easier to guess or brute-force (try many options until one works). If you use one that's random it's hard to remember, and thus you're more inclined to write the password down. Both of these are Very Bad Things™. This is why you're using ssh keys.

   But using a key without a passphrase is basically the same as writing down that random password in a file on your computer. Anyone who gains access to your drive has gained access to every system you use that key with. This is also a Very Bad Thing™. The solution is obvious: add a passphrase.

   But I don't want to enter a long passphrase every time I use the key!

   Neither do we! Thankfully, there's a nifty little tool called `ssh-agent` that can save your passphrase securely so you don't have to re-enter it. If you're on OSX Leopard or later your keys can be saved in the system's keychain to make your life even easier.

   For more information about SSH key passphrases, check out our help guide.

   ```
   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
   The key's randomart image is:
   +--[ RSA 2048]----+
   |     .+   +      |
   |       = o O .   |
   |        = * *    |
   |       o = +     |
   |      o S .      |
   |     o o =       |
   |      o . E      |
   |                 |
   |                 |
   +-----------------+
   ```

4. Add your SSH key to GitHub.

   On the GitHub site Click "Account Settings" > Click "SSH Public Keys" > Click "Add another public 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 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:

   OSX

   ```
   $ pbcopy < ~/.ssh/id_rsa.pubCopies the contents of the id_rsa.pub file to your clipboard
   ```

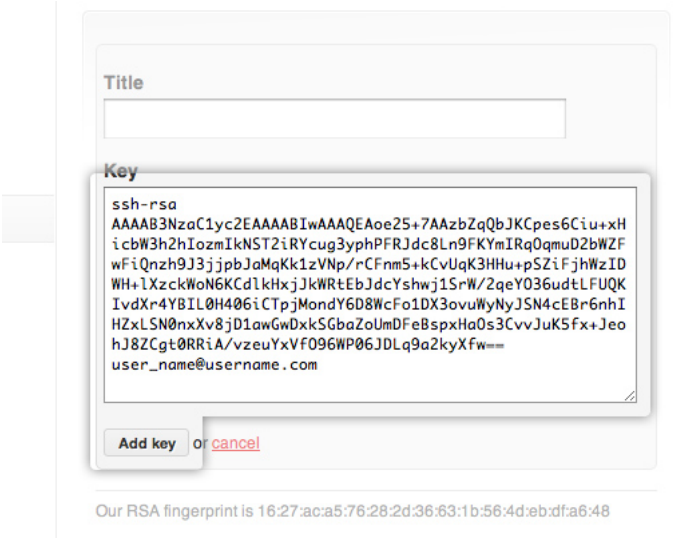
Windows

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

Linux

```
$ sudo apt-get install xclipDownloads and installs xclip
$ xclip -sel clip < ~/.ssh/id_rsa.pubCopies the contents of the id_rsa.pub file to your clipboard
```

Now paste it into the "Key" field.



Hit "Add Key."

5. 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.comAttempts to ssh to github
   ```

   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.
   ```

Having problems?

Setting up SSH can be problematic some times. Here are some frequent issues people run into:

SSH to GitHub

When it came time to run `$ ssh -T git@github.com` did you type your own email in there? If you did, you will need to run it again, making sure it says "git@github.com". This is important because it is the address you are trying to SSH to (GitHub), not your email address.

Sudo

Are you using `sudo`? If you don't know what that means, then you probably aren't. If you are using it, stop! You shouldn't use `sudo` unless you have a very good reason.

Agent admitted failure to sign using the key

This one only seems to happen to Linux users. To solve it, try this:

```
$ ssh-add ~/.ssh/id_rsaAssociates your ssh key with ssh-agent
```

If that doesn't solve it, check out this thread.

My connection was refused

You may be blocked by a firewall. First, you should check your ssh debug info:

```
$ ssh -vT git@github.comPrints debug info for the git@github.com SSH connection
```

Make sure it is connecting to GitHub on port 22. If it is, check to make sure your firewall is not blocking port 22.

When all else fails

If none of these solved your problem, check out this guide or contact GitHub support

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
   ```

   More about user info

   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.

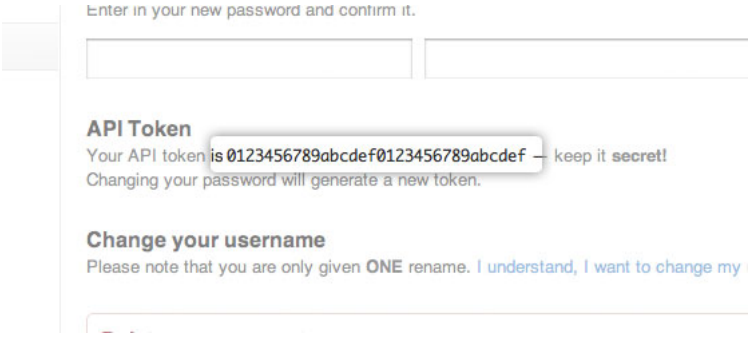
   ```
   $ cd my_other_repoChanges the working directory to the repo you need to switch info for
   $ git config user.name "Different Name"Sets the user's name for this specific repo
   $ git config user.email "differentemail@email.com"Sets the user's email for this specific repo
   ```

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

2. Set your GitHub token.

   Some tools connect to GitHub without SSH. To use these tools properly you need to find and configure your API Token.

   On the GitHub site Click "Account Settings" > Click "Account Admin."

   At the command line run the following code, using your GitHub username and token in place of the ones shown.

   ```
   $ git config --global github.user usernameSets the GitHub username for all git instances on the system
   	$ git config --global github.token 0123456789yourf0123456789tokenSets the GitHub token for all git instances on the syste
   ```

   Note If you ever change your GitHub password, a new token will be created and will need to be updated.

Lastly: Celebrate

Congratulations, you now have Git and GitHub all set up! What do you want to do next?

1. Set Up Git
2. Create A Repository
3. Fork A Repository
4. Be Social

- **Beginner**
  - Set Up Git
  - Create A Repo
  - Fork A Repo
  - Be Social
  - Delete a repo
  - Move a repo
  - Leave a collaborative repo
  - Send pull requests
  - Set your user name, email and GitHub token

- **Intermediate**
- **Advanced**
- **Troubleshooting**
- **Third party**
- **GitHub Resources**
- **Git resources**
- **Site policy**

This website is open source. Please help us by forking the project and adding to it.

Site Status:

All systems operational

GitHub Tips

Next Tip

Want to summon someone to an issue or pull request? Just @mention them.

GitHub Links

GitHub

- About
- Blog
- Contact & Support
- Training
- Site Status

Tools

- GitHub for Mac
- Gist
- Enterprise Install
- Job Board

Extras

- Shop
- The Octodex

Documentation

- GitHub Help
- Developer API
- GitHub Flavored Markdown
- GitHub Pages

Home

- Terms of Service
- Privacy
- Security

© 2011 GitHub Inc. All rights reserved.


Powered by the Dedicated Servers and
Cloud Computing of Rackspace Hosting®