Set Up Git - OSX

This guide will walk you through the basics of installing and configuring. Don't worry too much if the terminology used is a bit mysterious – we'll cover everything you really need to know on the course!

This is the guide for setting up git in **OSX**. There are also guides for Windows and Linux (see the end of this guide for details).

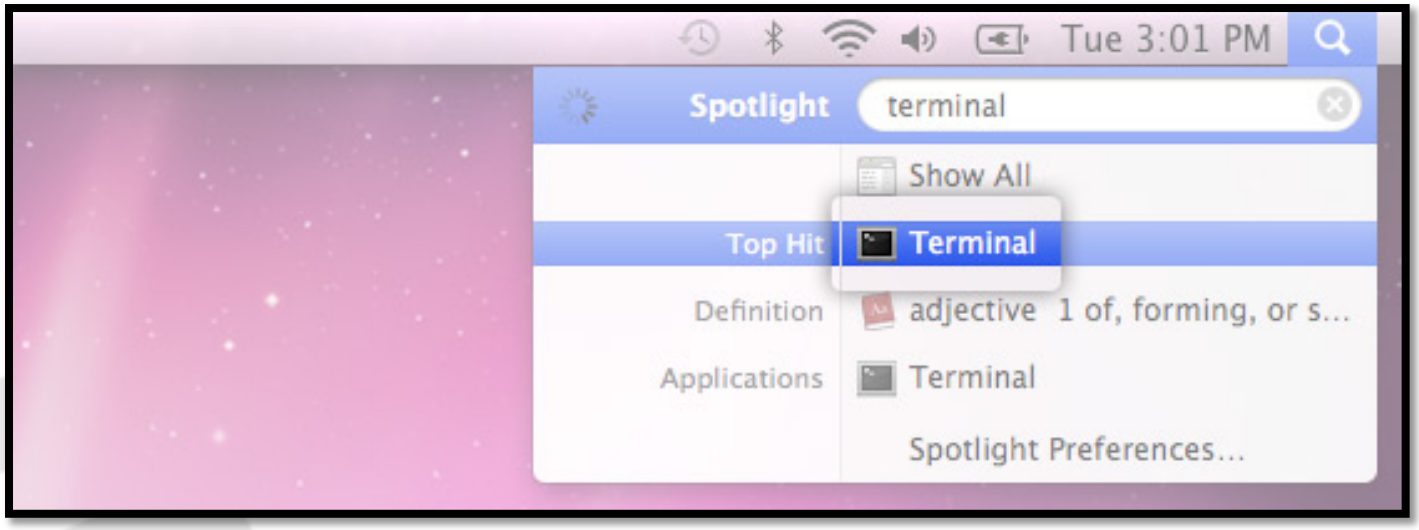
First: Download and Install Git

1. **Download and install the latest version of Git. (http://git-scm.com/)**

   Note Don't worry that you don't see an icon when it's done. It's not that kind of application.

Next: Open a new Terminal window

First, you need to open Terminal.app, usually found at /Applications/Utilities.



Then: Set Up Your Info

Now that you have Git installed, it's time to configure your personal info.

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

   Git tracks who makes each change (commit) by checking the user's name and email. To set these, enter the code below, replacing the name and email with your own…

```
$ git config --global user.name "Firstname Lastname"
$ git config --global user.email "your_email@youremail.com"
```

Then: Set Up SSH Keys

We use SSH (Secure SHell) keys to establish a secure connection between your computer and Git repositories hosted over the Internet. Setting them up is fairly easy, but does involve a number of steps.

1. **Check for SSH keys.** *Have an existing key pair? You can skip to adding the key to GitHub.*

   To make sure you generate a brand new key, you need to check if one already exists…

```
$ cd ~/.ssh
```

   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:

```
$ ls
config id_rsa id_rsa.pub known_hosts
$ mkdir key_backup
$ cp id_rsa* key_backup
$ rm id_rsa*
```

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"
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:

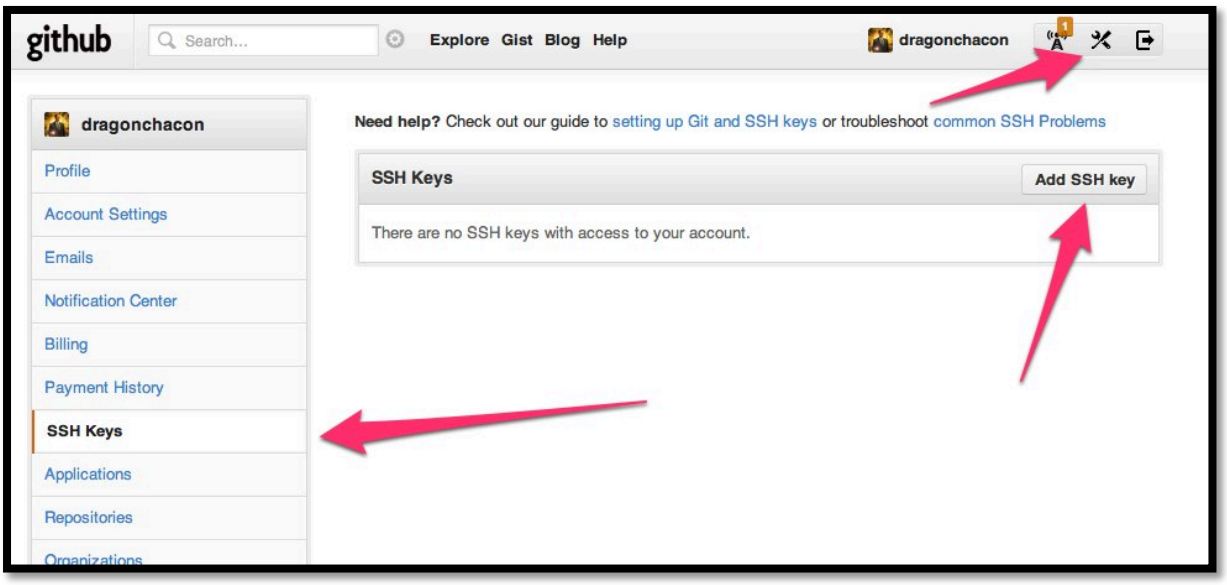
```
Your identification has been saved in
/Users/your_user_directory/.ssh/id_rsa.
Your public key has been saved in
/Users/your_user_directory/.ssh/id_rsa.pub.
The key fingerprint is:
01:0f:f4:3b:ca:85:d6:17:a1:7d:f0:68:9d:f0:a2:db user_name@username.com
```

Then: Sign up to GitHub and add your SSH key

GitHub is the most popular Internet site for hosting Git repositories. All public (i.e. open-source) repositories are free, private repositories you have to pay for.

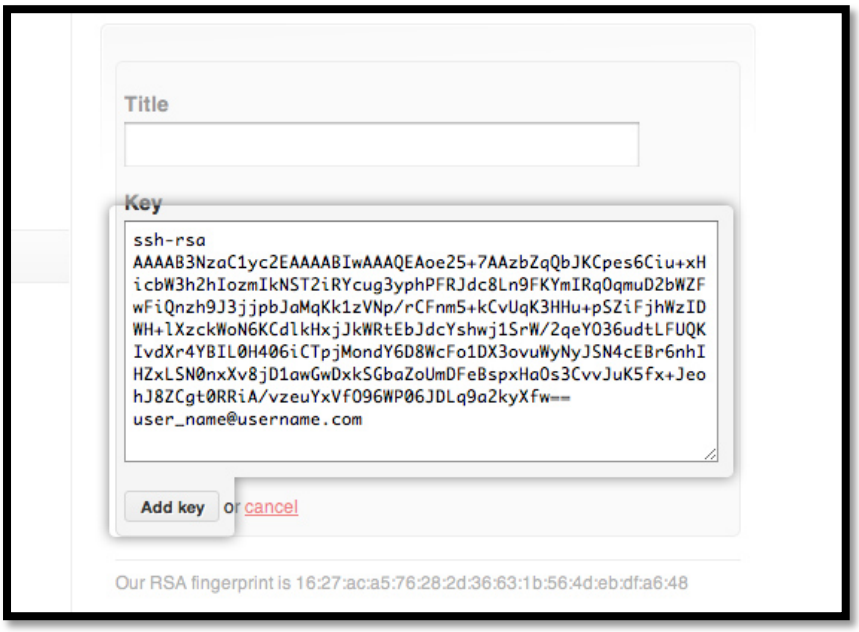
If you don’t already have a GitHub account, go to http://www.github.com and register for a free account.

Then, on the GitHub site *Click “Account Settings” > Click “SSH Keys” > Click “Add SSH key”*



Open the id_rsa.pub file with a text editor. 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.

Now paste it into the "Key" field.



Hit "Add Key."

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.
```

Finally: Set Up A Merge Tool (Nearly there!)

When you are trying to compare and merge two changes to the same file, it is helpful to have a good graphical tool. Git supports a number of merge tools, and you can use whichever one you are accustomed to. At Driven, we prefer P4Merge.

To download P4Merge go to: http://www.perforce.com/downloads/complete_list. Download and install the MACINTOSH version of the "The Perforce Visual Client (P4V)".

Now tell Git to use P4 when you have a merge conflict…

```
$ git config --global merge.tool p4merge
$ git config --global mergetool.p4merge.path "/Applications/p4merge.app/Contents/MacOS/p4merge"
```

Lastly: Celebrate

Congratulations, that's all there is to it. You now have Git all set up! We look forward to seeing you at the course.

driven software

These instructions are based on those found at GitHub.com, but include customisations that we at Driven Software believe will be helpful, and are used in our Git Training Course. (http://drivensoftware.com/Pages/Training/Git/)