Sparkleshare Installation Instructions (Currently available for Linux, Mac, and Android. Windows & iOS versions under development).

Download sparkleshare-0.4.0.tar.gz from Sparkleshare.org (I chose to download as tar.gz rather than zip which allowed me to place or copy file in /usr/local/src, where I extract it)

Since the file is a tar.gz file, the program will be compiled from source code.

It is considered a best practice to copy downloaded files to the /usr/local/src directory and extract them in the same directory.

$ cd /Downloads

$ sudo cp sparkleshare-0.4.0.tar.gz /usr/local/src

$ cd /usr/local/src

$ tar –xvzf sparkleshare-0.4.0.tar.gz

$ cd sparkleshare-0.4.0

$ sudo apt-get install gtk-sharp2 mono-runtime mono-devel monodevelop \

libndesk-dbus1.0-cil-dev nant libnotify-cil-dev libgtk2.0-cil-dev mono-gmcs \

libwebkit-cil-dev intltool libtool python-nautilus libndesk-dbus-glib1.0-cil-dev (install required dependencies)

$ sudo apt-get install libappindicator0.1-cil-dev (install Ubuntu libappindicator support)

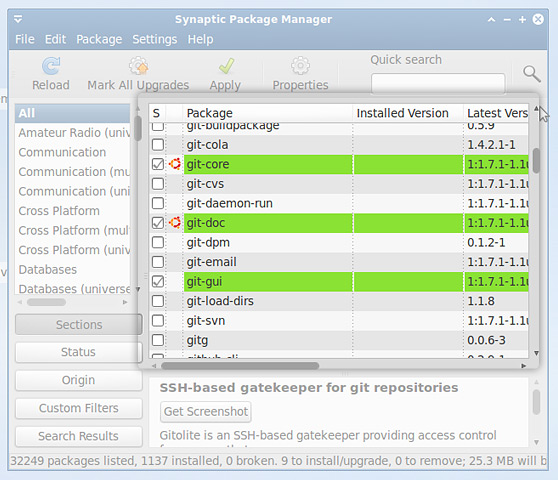
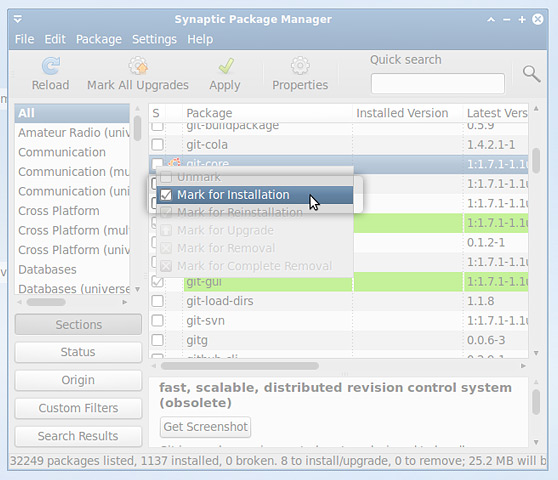
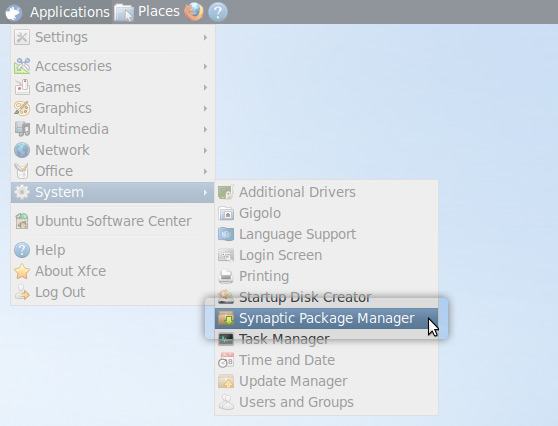
$ ./configure --prefix=/usr (or ./autogen.sh if you build from the repository)

$ make

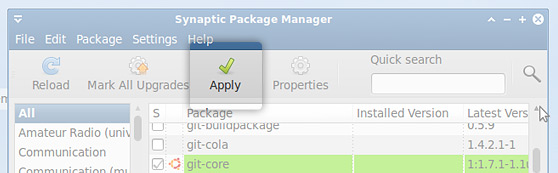
$ sudo make install

$ sparkleshare start ( before starting the program, setup Github)

**Download and install the latest version of Git with Synaptic Package Manager.**

Install git-core, git-gui, & git-doc. (Newer Debian/Ubuntu installations will have git instead of git-core.)

When you’ve selected git-core, git-gui, and git-doc, hit “Apply” to install them.



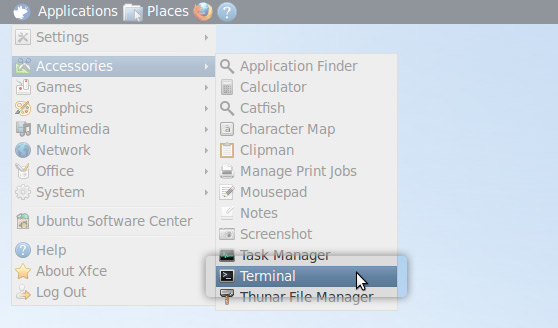
If you don’t have a package manager like Synaptic, or you’d rather install the necessary git components from the command line, you can alternatively run the script below:

**$ sudo apt-get install git-core git-gui git-doc**

**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 an app called Terminal.



Need a quick lesson about Terminal?

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**.

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:

$ lsLists all the subdirectories in the current directoryconfig 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

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*"Creates a new ssh key using the provided emailGenerating 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?

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      ||                 ||                 |+-----------------+

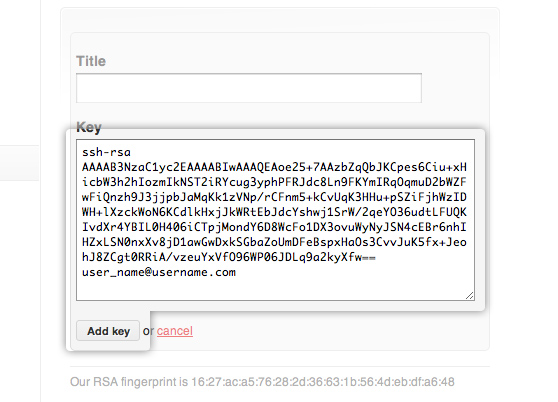
1. **Add your SSH key to GitHub.**

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:

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.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?

**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.

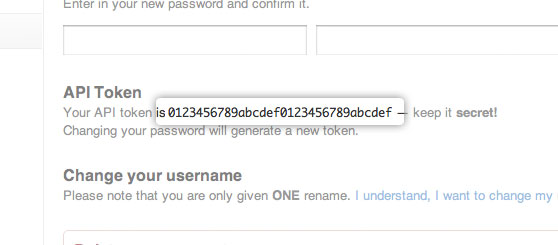
$ git config --global user.name "*Firstname Lastname*"Sets the name of the user for all git instances on the system$ git config --global user.email "*your\_email@youremail.com*"Sets the email of the user for all git instances on the system

More about user info

1. **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 *username*Sets the GitHub username for all git instances on the system$ git config --global github.token *0123456789yourf0123456789token*Sets the GitHub token for all git instances on the system

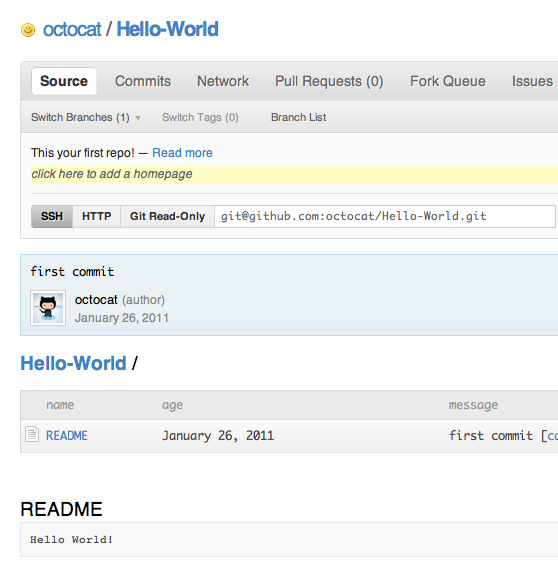
**\*Note\*** If you ever change your GitHub password, a new token will be created and will need to be updated.

**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**](http://help.github.com/create-a-repo/)
3. [**Fork A Repository**](http://help.github.com/fork-a-repo/)
4. [**Be Social**](http://help.github.com/be-social/)

Creating vs Forking a Repository



# Fork A Repo

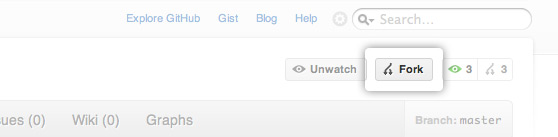
If you’ve found yourself on this page, we’re assuming you’re brand new to Git and GitHub. This guide will walk you through the basics and explain a little bit about how everything works along the way.

## First: Fork A Repo

At some point you may find yourself wanting to contribute to someone else’s project, or would like to use someone’s project as the starting point for your own. This is known as “forking.” For this tutorial, we’ll be using the [Spoon-Knife](https://github.com/octocat/Spoon-Knife) project.

1. **Fork the “Spoon-Knife ” repo**

To fork this project, click the “Fork” button.



## Next: Set Up Your Local Repo

You’ve successfully forked the Spoon-Knife repo, but so far it only exists on GitHub. To be able to work on the project, you will need to clone it to your local machine.

1. **Clone the “Spoon-Knife” project**

Run the following code:

$ git clone git@github.com:username/Spoon-Knife.gitClones your copy of the repo into the current directory in terminal

1. **Configure remotes**

When a repo is cloned, it has a default remote called origin that points to your fork on GitHub, not the original repo it was forked from. To keep track of the original repo, you need to add another remote named upstream:

#### More about remotes

$ cd Spoon-KnifeChanges the active directory in the prompt to the newly cloned "Spoon-Knife" directory$ git remote add upstream git://github.com/octocat/Spoon-Knife.gitAssigns the original repo to a remote called "upstream"$ git fetch upstreamPulls in changes not present in your local repository, without modifying your files

## Then: More Things You Can Do

You’ve successfully forked a repo, but get a load of these other cool things you can do:

* **Push commits**

Once you’ve made some commits to a forked repo and want to push it to your forked project, you do it the same way you would with a regular repo:

#### More about commits

$ git push origin masterPushes commits to your remote repo stored on GitHub

* **Pull in upstream changes**

If the original repo you forked your project from gets updated, you can add those updates to your fork by running the following code:

$ git fetch upstreamFetches any new changes from the original repo$ git merge upstream/masterMerges any changes fetched into your working files

#### What is the difference between fetch and pull?

* **Work with branches**

Branching allows you to build new features or test out ideas without putting your main project at risk. A Git branch is a small file that references the commit it was spawned from. This makes Git branches very small and easy to work with.

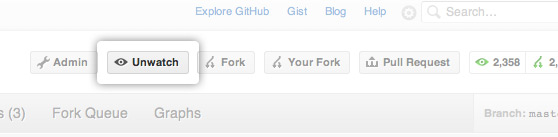
#### How do I use branches?

* **Pull requests**

If you are hoping to contribute back to the original fork, you can send the original author a [pull request](http://help.github.com/pull-requests/).

* **Unwatch the main repo**

When you fork a particularly popular repo, you may find yourself with a lot of unwanted updates about it. To unsubscribe from updates to the main repo, click the “Unwatch” button on the **main repo**.



* **Delete your fork**

At some point you may decide that you want to delete your fork. To delete a fork, just follow the same steps as you would to [delete a regular repo](http://help.github.com/delete-a-repo).