**SSH Keys**

What are SSH Keys? SSH stands for Secure Shell. SSH Keys are used as a method of authentication and to create secure and confidential connections on the Internet.

Github has two methods of establishing secure connections whenever you push or pull, namely SSH keys and the HTTPS method. You’ll need to establish one of these secure connections before you can start pulling or pushing. If you’re using the GitHub application, the app automatically generates a keypair for you, so you don’t have to do it yourself.

SSH Keys are made in pairs, one public key and one private key. These two keys are different, so you cannot derive one from the other. Let’s say someone wants to send you a private message. He will use your public key to encrypt his message and only the person with the private key (which should ONLY be you) is able to decrypt it. However, the reverse is true as well. If you were to encrypt something using your private key, only the holders of the public key can decrypt it.

Of course, the public key can be known by more than one person, and thus decipherable by them, but this is useful for authentication. It proves that the message they received comes from you.

Now let’s apply this to GitHub. Whenever you push or pull, it’s a request that is sent to GitHub which is encrypted with your private key (assuming you’ve set up SSH keys already). GitHub knows that the request comes from you and not anyone else, because your message was encrypted with your own private key. That’s good, because GitHub now knows it can trust the connection, and fulfills your request. It sends data back to you, encrypted with your public key, and only you can decrypt it with your private key to receive the message.

**How to set up SSH Keys for GitHub**

1. First you’ll need to check if you have any ssh keys stored. Just open up GitBash and type:

cd ~/.ssh

If it says "No such file or directory", go to **step 2**. Otherwise, this means that you already have an existing keypair, and you can skip to **step 3**.

2. If you don’t have a keypair, you’ll have to make one. Type:

ssh-keygen -t rsa -C <your email>

Replace the <your email> part with your own email, and remember to put inverted commas, like this:

ssh-keygen -t rsa -C “jacetan93@gmail.com”

Next you’ll need to create a passphrase. When you type, note that nothing will appear on GitBash. Just type out your desired passphrase, hit Enter and type it again.

**Signing Up for a GitHub Account**

Go to github.com and click “Sign up for Free”.

Choose a username (this must be unique!), type in your email address, and choose a password.

Once you’ve retyped your password in the confirm password field, click “Create an account”.

That’s it! You’ve got an account with GitHub!

If you’d like to use GitBash, go to <https://code.google.com/p/msysgit/downloads/list?q=label:Featured> and download the full installer (probably the first one).

If you prefer to use the GUI, carry on to the next section to find out more.

3. Now that you have a keypair, you’ll need to copy it to your clipboard and give it to GitHub. Here’s how you do it. In your GitBash, type:

clip < ~/.ssh/id\_rsa.pub

Now go to github.com and log in. Go to your account settings and click SSH Keys on the sidebar on the left.

Next, click Add SSH Key and paste your key into the key field.

Click Add Key and confirm the action by entering your github password when prompted.

4. Now you need to make sure that everything is working properly. Go back to your GitBash and type:

ssh -T git@github.com

If you see a warning asking you to verify the connection, just type yes and press Enter.

Now GitBash should greet you with your GitHub username. If it does, then voilà, you’re successful. If it doesn’t, then you might want to use the HTTPS connection method instead.