The easiest way to do it, is through ssh, this will work for Linux/Mac

1. Open terminal to create ssh keys:
2. cd ~ #Your home directory
3. ssh-keygen -t rsa #Press enter for all values
4. Give the git administrator your ~/.ssh/id\_rsa.pub key and next time you push, no password required. If you're using GitHub, go to [**settings**](https://github.com/settings/ssh) and click 'add SSH key'

For Windows (Only works if the commit program is capable of using certificates/private & public ssh keys)

1. Use Putty Gen to generate a key
2. Export the key as an open SSH key
3. Give the git administrator your id\_rsa.pub

For a walkthrough on putty gen for the above steps, please see <http://ask-leo.com/how_do_i_create_and_use_public_keys_with_ssh.html>.

## Generating SSH keys

* [**MAC**](https://help.github.com/articles/generating-ssh-keys/#platform-mac)
* [**WINDOWS**](https://help.github.com/articles/generating-ssh-keys/#platform-windows)
* [**LINUX**](https://help.github.com/articles/generating-ssh-keys/#platform-linux)
* [**ALL**](https://help.github.com/articles/generating-ssh-keys/#platform-all)

SSH keys are a way to identify trusted computers, without involving passwords. The steps below will walk you through generating an SSH key and adding the public key to your GitHub account.

We recommend that you regularly [review your SSH keys list](https://help.github.com/articles/keeping-your-ssh-keys-and-application-access-tokens-safe) and revoke any that haven't been used in a while.

**Tip:** If you have [GitHub for Windows](https://windows.github.com/) installed, you can use it to clone repositories and not deal with SSH keys. It also comes with the Git Bash tool, which is the preferred way of running git commands on Windows.

### Step 1: Check for SSH keys

First, we need to check for existing SSH keys on your computer. Open Git Bash and enter:

ls -al ~/.ssh

# Lists the files in your .ssh directory, if they exist

Check the directory listing to see if you already have a public SSH key. By default, the filenames of the public keys are one of the following:

* id\_dsa.pub
* id\_ecdsa.pub
* id\_ed25519.pub
* id\_rsa.pub

### Step 2: Generate a new SSH key

1. With Git Bash still open, copy and paste the text below. Make sure you substitute in your GitHub email address.
2. ssh-keygen -t rsa -C "your\_email@example.com"
3. # Creates a new ssh key, using the provided email as a label
4. # Generating public/private rsa key pair.
5. We strongly suggest keeping the default settings as they are, so when you're prompted to "Enter a file in which to save the key", just press **Enter** to continue.
6. # Enter file in which to save the key (/Users/you/.ssh/id\_rsa): [Press enter]
7. You'll be asked to enter a passphrase.
8. # Enter passphrase (empty for no passphrase): [Type a passphrase]
9. # Enter same passphrase again: [Type passphrase again]

**Tip:** We strongly recommend a very good, secure passphrase. For more information, see "[Working with SSH key passphrases](https://help.github.com/articles/working-with-ssh-key-passphrases)".

1. After you enter a passphrase, you'll be given the fingerprint, or id, of your SSH key. It will look something like this:
2. # Your identification has been saved in /Users/you/.ssh/id\_rsa.
3. # Your public key has been saved in /Users/you/.ssh/id\_rsa.pub.
4. # The key fingerprint is:
5. # 01:0f:f4:3b:ca:85:d6:17:a1:7d:f0:68:9d:f0:a2:db your\_email@example.com

### Step 3: Add your key to the ssh-agent

To configure the [ssh-agent](https://en.wikipedia.org/wiki/Ssh-agent) program to use the SSH key you've generated:

If you have [GitHub for Windows](https://windows.github.com/) installed, you can use it to clone repositories and not deal with SSH keys. It also comes with the Git Bash tool, which is the preferred way of running git commands on Windows.

1. Ensure ssh-agent is enabled:
   * **If you are using Git Bash**, turn on ssh-agent:
   * # start the ssh-agent in the background
   * ssh-agent -s
   * # Agent pid 59566
   * **If you are using another terminal prompt**, such as [msysgit](https://msysgit.github.io/), turn on ssh-agent:
   * # start the ssh-agent in the background
   * eval $(ssh-agent -s)
   * # Agent pid 59566
2. Add your generated SSH key to the ssh-agent:
3. ssh-add ~/.ssh/id\_rsa

### Step 4: Add your SSH key to your account

To configure your GitHub account to use your SSH key:

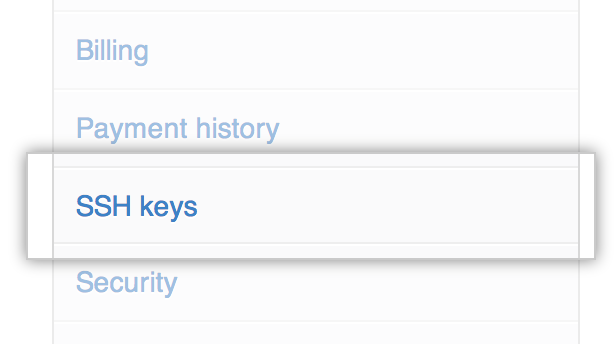
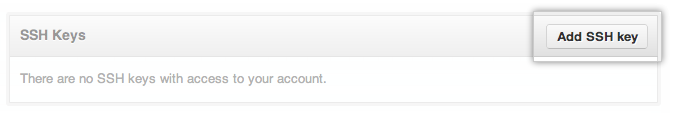
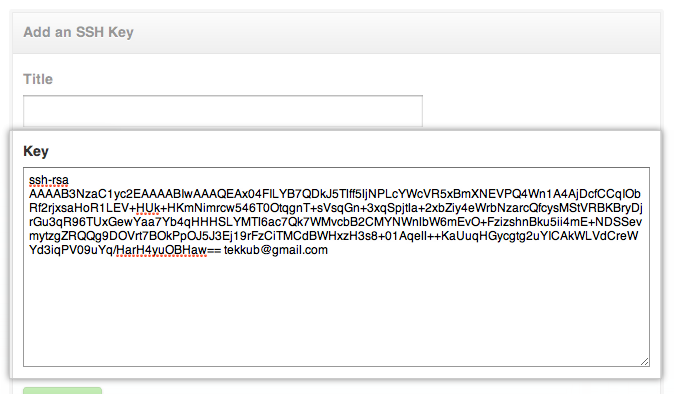
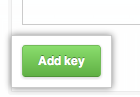
Copy the SSH key to your clipboard. If your key is named id\_dsa.pub, id\_ecdsa.pub orid\_ed25519.pub, then change the filename below from id\_rsa.pub to the one that matches your key:

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

# Copies the contents of the id\_rsa.pub file to your clipboard

**Warning**: It's important to copy the key exactly without adding newlines or whitespace.

Add the copied key to GitHub:

1. In the top right corner of any page, click .
2. In the user settings sidebar, click **SSH keys**.
3. Click **Add SSH key**.
4. In the Title field, add a descriptive label for the new key. For example, if you're using a personal Mac, you might call this key "Personal MacBook Air".
5. Paste your key into the "Key" field.
6. Click **Add key**.
7. Confirm the action by entering your GitHub password.

### Step 5: Test the connection

To make sure everything is working, you'll now try to SSH into . When you do this, you will be asked to authenticate this action using your password, which is the SSH key passphrase you created earlier.

1. Open Git Bash and enter:
2. ssh -T git@github.com
3. # Attempts to ssh to GitHub
4. You may see this warning:
5. # The authenticity of host 'github.com (207.97.227.239)' can't be established.
6. # RSA key fingerprint is 16:27:ac:a5:76:28:2d:36:63:1b:56:4d:eb:df:a6:48.
7. # Are you sure you want to continue connecting (yes/no)?

Verify the fingerprint in the message you see matches the following message, then type yes:

# Hi username! You've successfully authenticated, but GitHub does not

# provide shell access.

1. If the username in the message is yours, you've successfully set up your SSH key!

If you receive a message about "access denied," you can [read these instructions for diagnosing the issue](https://help.github.com/articles/error-permission-denied-publickey).

If you're switching from HTTPS to SSH, you'll now need to update your remote repository URLs. For more information, see [Changing a remote's URL](https://help.github.com/articles/changing-a-remote-s-url).

## Changing a remote's URL

The git remote set-url command changes an existing remote repository URL.

**Tip:** For information on the difference between HTTPS and SSH URLs, see "[Which remote URL should I use?](https://help.github.com/articles/which-remote-url-should-i-use)"

The git remote set-url command takes two arguments:

* An existing remote name. For example, origin or upstream are two common choices.
* A new URL for the remote. For example:
  + If you're updating to use HTTPS, your URL might look like:
  + https://github.com/USERNAME/OTHERREPOSITORY.git
  + If you're updating to use SSH, your URL might look like:
  + git@github.com:USERNAME/OTHERREPOSITORY.git

### Switching remote URLs from SSH to HTTPS

1. Open Terminal (for Mac users) or the command prompt (for Windows and Linux users).
2. Change the current working directory to your local project.
3. List your existing remotes in order to get the name of the remote you want to change.
4. git remote -v
5. # origin git@github.com:USERNAME/REPOSITORY.git (fetch)
6. # origin git@github.com:USERNAME/REPOSITORY.git (push)
7. Change your remote's URL from SSH to HTTPS with the remote set-url command.
8. git remote set-url origin https://github.com/USERNAME/OTHERREPOSITORY.git
9. Verify that the remote URL has changed.
10. git remote -v
11. # Verify new remote URL
12. # origin https://github.com/USERNAME/OTHERREPOSITORY.git (fetch)
13. # origin https://github.com/USERNAME/OTHERREPOSITORY.git (push)

The next time you git fetch, git pull, or git push to the remote repository, you'll be asked for your GitHub username and password.

* If you have [two-factor authentication](https://help.github.com/articles/about-two-factor-authentication) enabled, you must [create a personal access token](https://help.github.com/articles/creating-an-access-token-for-command-line-use) to use instead of your GitHub password.
* You can [use a credential helper](https://help.github.com/articles/caching-your-github-password-in-git) so Git will remember your GitHub username and password every time it talks to GitHub.

### Switching remote URLs from HTTPS to SSH

1. Open Terminal (for Mac users) or the command prompt (for Windows and Linux users).
2. Change the current working directory to your local project.
3. List your existing remotes in order to get the name of the remote you want to change.
4. git remote -v
5. # origin https://github.com/USERNAME/REPOSITORY.git (fetch)
6. # origin https://github.com/USERNAME/REPOSITORY.git (push)
7. Change your remote's URL from HTTPS to SSH with the remote set-url command.
8. git remote set-url origin git@github.com:USERNAME/OTHERREPOSITORY.git
9. Verify that the remote URL has changed.
10. git remote -v
11. # Verify new remote URL
12. # origin git@github.com:USERNAME/OTHERREPOSITORY.git (fetch)
13. # origin git@github.com:USERNAME/OTHERREPOSITORY.git (push)

### Troubleshooting

You may encounter these errors when trying to changing a remote.

#### No such remote '[name]'

This error means that the remote you tried to change doesn't exist:

git remote set-url sofake https://github.com/octocat/Spoon-Knife

# fatal: No such remote 'sofake'

Check that you've correctly typed the remote name.

### Further reading

* "[Working with Remotes](http://git-scm.com/book/en/Git-Basics-Working-with-Remotes)" from the Pro Git book

## Working with SSH key passphrases

* [**MAC**](https://help.github.com/articles/working-with-ssh-key-passphrases/#platform-mac)
* [**WINDOWS**](https://help.github.com/articles/working-with-ssh-key-passphrases/#platform-windows)
* [**ALL**](https://help.github.com/articles/working-with-ssh-key-passphrases/#platform-all)

This article will walk you through the process of securing your SSH keys and configuring an authentication agent so that you won't have to re-enter your passphrase every time you use your keys.

### Why do I need a passphrase?

Passwords aren't very secure. If you use one that's easy to remember, it's also 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 it down. Both of these are Very Bad Things.

This is why you're using SSH keys. Of course, using a key without a passphrase is basically the same as writing down a random password: anyone who gains access to your computer has gained access to every system you use that key with. This is also a Very Bad Thing. The solution is to add a passphrase to the SSH key for an extra layer of security.

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

Neither do I! Thankfully, there's a nifty little tool called ssh-agent that can securely save your passphrase, so you don't have to re-enter it. If you're on OS X Leopard or later your keys can be saved in the system's keychain to make your life even easier. Most Linux installations will automatically start ssh-agent for you when you log in.

### Adding or changing a passphrase

You can change the passphrase for an existing private key without regenerating the keypair. Just type the following command:

ssh-keygen -p

# Start the SSH key creation process

# Enter file in which the key is (/Users/you/.ssh/id\_rsa): [Hit enter]

# Key has comment '/Users/you/.ssh/id\_rsa'

# Enter new passphrase (empty for no passphrase): [Type new passphrase]

# Enter same passphrase again: [One more time for luck]

# Your identification has been saved with the new passphrase.

If your key already has a passphrase, you will be prompted to enter it before you can change to a new passphrase.

## Caching your GitHub password in Git

* [**MAC**](https://help.github.com/articles/caching-your-github-password-in-git/#platform-mac)
* [**WINDOWS**](https://help.github.com/articles/caching-your-github-password-in-git/#platform-windows)
* [**LINUX**](https://help.github.com/articles/caching-your-github-password-in-git/#platform-linux)
* [**ALL**](https://help.github.com/articles/caching-your-github-password-in-git/#platform-all)

If you're [cloning GitHub repositories using HTTPS](https://help.github.com/articles/which-remote-url-should-i-use), you can use a credential helper to tell Git to remember your GitHub username and password every time it talks to GitHub.

If you clone GitHub repositories using SSH, then you authenticate using SSH keys instead of a username and password. For help setting up an SSH connection, see [Generating SSH Keys](https://help.github.com/articles/generating-ssh-keys).

**Tip:** You need Git **1.7.10** or newer to use the credential helper.

If you install [GitHub for Windows](https://windows.github.com/), the credential helper is included. The app also provides a Git shell so you won't ever need to install and configure Git manually.

If you prefer working with the command line, you can also install a native Git shell, such as [msysgit](https://msysgit.github.io/). With msysgit, running the following in the command line will store your credentials:

git config --global credential.helper wincred