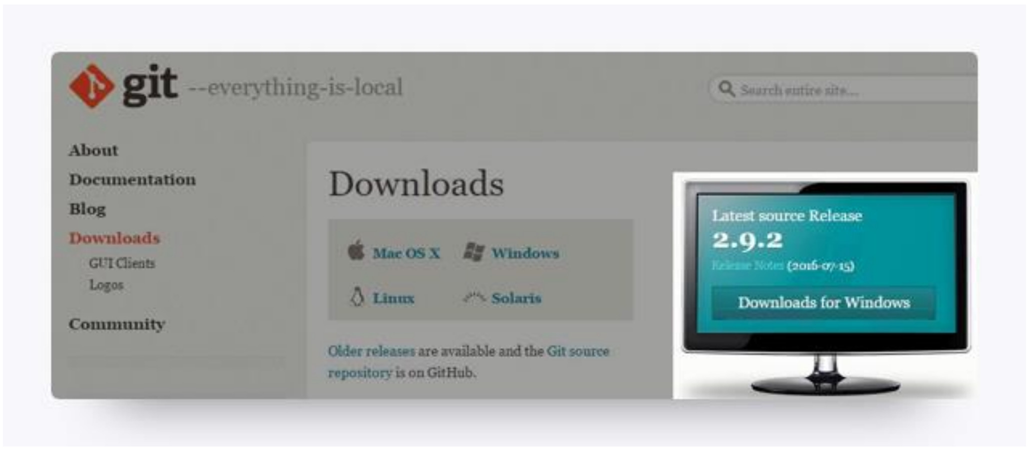
Git & Git Bash (You can use Cygwin if you like)

Coders depend on this tool for version control; the process of logging the development of programs and applications. This comes in handy during collaborative programming, when teams of programmers change, add, and remove code throughout a project's directory; this process would be chaotic without Git.

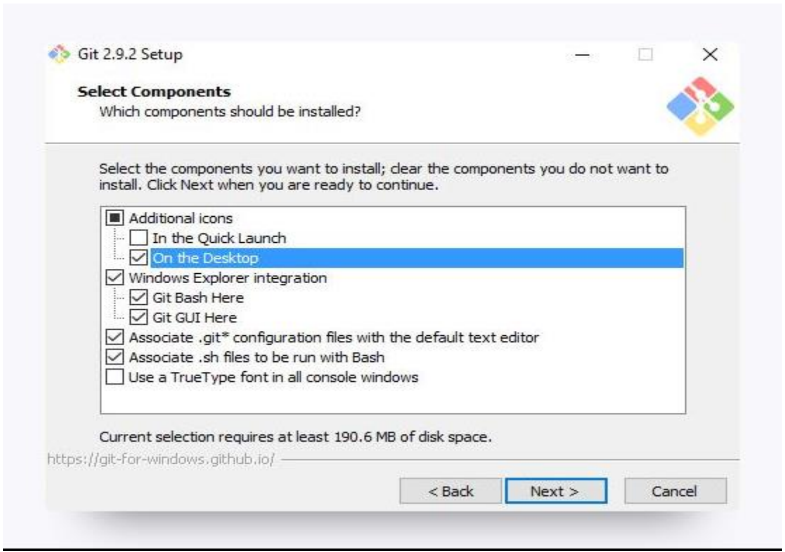
The installation also includes Git Bash, or Bash for short. You'll be using this command line terminal throughout the course and during the rest of these instructions.

1. Go to Git download page: https://git-scm.com/downloads. Click on the download for your computer.



2. Run through the installation file. Make sure you check off the right boxes as shown in these four images.

- Save Git to the desktop (this should save Git Bash to your desktop too).


Git 2.9.2 Setup

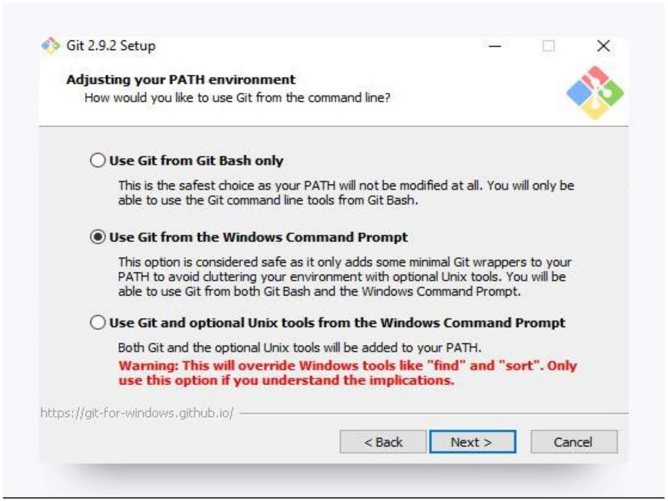
Select Components
Which components should be installed?

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
  - [✓] Git Bash Here
  - [✓] Git GUI Here
- [✓] Associate .git* configuration files with the default text editor
- [✓] Associate .sh files to be run with Bash
- [ ] Use a TrueType font in all console windows

Current selection requires at least 190.6 MB of disk space.

- Use Git from the Windows Command Prompt.



Git 2.9.2 Setup

Adjusting your PATH environment

How would you like to use Git from the command line?

( ) **Use Git from Git Bash only**

This is the safest choice as your PATH will not be modified at all. You will only be able to use the Git command line tools from Git Bash.

(●) **Use Git from the Windows Command Prompt**

This option is considered safe as it only adds some minimal Git wrappers to your PATH to avoid cluttering your environment with optional Unix tools. You will be able to use Git from both Git Bash and the Windows Command Prompt.

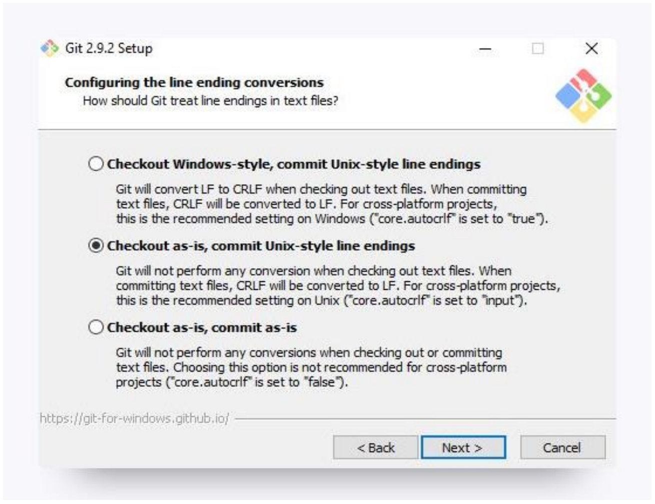
( ) **Use Git and optional Unix tools from the Windows Command Prompt**

Both Git and the optional Unix tools will be added to your PATH.

Warning: This will override Windows tools like "find" and "sort". Only use this option if you understand the implications.

https://git-for-windows.github.io/

- Checkout as-is.



Git 2.9.2 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**

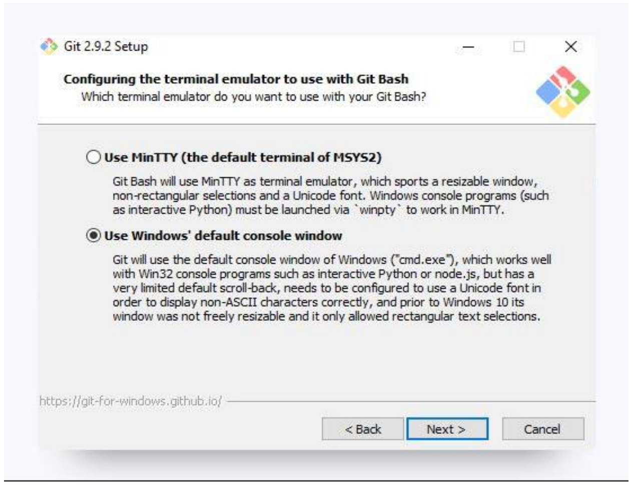
Git will not perform any conversion when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Unix ("core.autocrlf" is set to "input").

( ) **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").

https://git-for-windows.github.io/

- Use Windows' default console window.



Git 2.9.2 Setup

Configuring the terminal emulator to use with Git Bash

Which terminal emulator do you want to use with your Git Bash?

( ) **Use MinTTY (the default terminal of MSYS2)**

Git Bash will use MinTTY as terminal emulator, which sports a resizable window, non-rectangular selections and a Unicode font. Windows console programs (such as interactive Python) must be launched via `winpty` to work in MinTTY.

(●) **Use Windows' default console window**

Git will use the default console window of Windows ("cmd.exe"), which works well with Win32 console programs such as interactive Python or node.js, but has a very limited default scroll-back, needs to be configured to use a Unicode font in order to display non-ASCII characters correctly, and prior to Windows 10 its window was not freely resizable and it only allowed rectangular text selections.

https://git-for-windows.github.io/

SSH Key (Completely Optional)

Generating SSH keys allows developers to interface with certain remote services without having to constantly type out login information. You're going to set up an SSH key for GitHub.

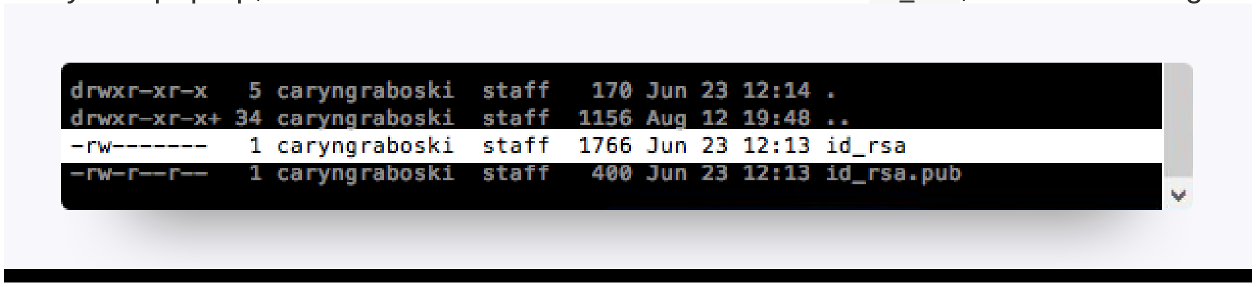
Without a key, you won’t be able to push your code to GitHub without entering a password each time; trust us, that would be as irritating as needing a key to open every door in your home.

1. If you haven't signed up for a GitHub account yet, you'll need to do so before moving on with these steps. Visit https://github.com.

2. Open up Bash.

3. We need to set up SSH keys. First, let’s make sure you don’t already have a set of keys on your computer. Type this into your Bash window **(copying and pasting will not work)**:

   - `ls -al ~/.ssh`
   - If no keys pop up, move onto step 4.
   - If keys do pop up, check that none of them are listed under `id_rsa`, like in this image:



```
drwxr-xr-x    5 caryngraboski  staff   170 Jun 23 12:14 .
drwxr-xr-x+ 34 caryngraboski  staff  1156 Aug 12 19:48 ..
-rw-------    1 caryngraboski  staff  1766 Jun 23 12:13 id_rsa
-rw-r--r--    1 caryngraboski  staff   400 Jun 23 12:13 id_rsa.pub
```

   - If you do find a key with a matching name, then you can either overwrite it by following steps 4 to 6, or you can use the same key in steps 10 and beyond. Be advised that you'll have to remember the password tied to your key if you decide not to overwrite it.

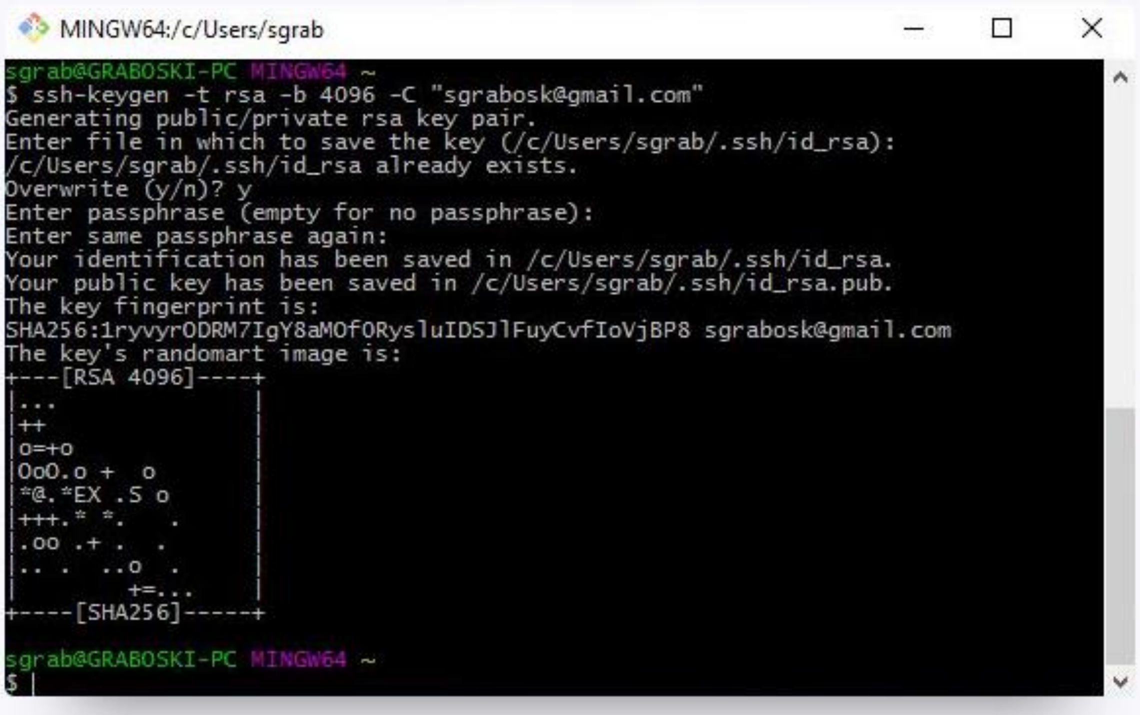
4. Type in this command along with your email to generate your keys

   - `ssh-keygen -t rsa -b 4096 -C "YOURGITHUBEMAIL@PLACEHOLDER.NET"`

5. When asked to enter a file to save the key, just hit `enter`.

   - Also enter a passphrase for your key.
   - Note: You shouldn’t see any characters appear in the window while typing the password.

6. When you're finished, your window should look like this:



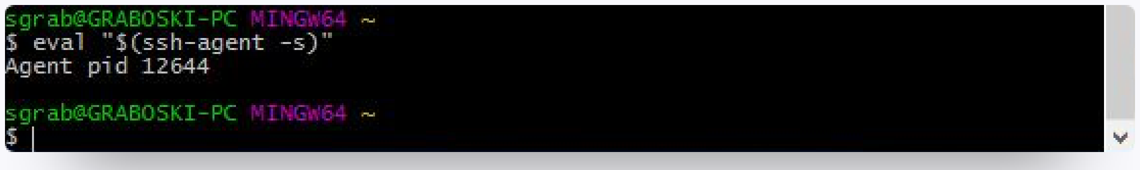
MINGW64:/c/Users/sgrab

```
sgrab@GRABOSKI-PC MINGW64 ~
$ ssh-keygen -t rsa -b 4096 -C "sgrabosk@gmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/sgrab/.ssh/id_rsa):
/c/Users/sgrab/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/sgrab/.ssh/id_rsa.
Your public key has been saved in /c/Users/sgrab/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:1ryvyrODRM7IgY8aMOfORysluIDSJlFuyCvfIoVjBP8 sgrabosk@gmail.com
The key's randomart image is:
+---[RSA 4096]----+
|...              |
|++               |
|o=+o             |
|OoO.o +  o       |
|*@.*EX .S o      |
|+++.* *.    .    |
|.oo .+ .   .     |
|.. .   ..o  .    |
|        +=...    |
+----[SHA256]-----+

sgrab@GRABOSKI-PC MINGW64 ~
$ |
```

7. For the next step, we need to use a tool called ssh agent to link our key with our machine. Let's test whether ssh-agent is working. Run this command in Bash:

- `eval "$(ssh-agent -s)"`
- If your Bash window looks like the below image, move onto the next step.



```
sgrab@GRABOSKI-PC MINGW64 ~
$ eval "$(ssh-agent -s)"
Agent pid 12644

sgrab@GRABOSKI-PC MINGW64 ~
$ |
```

8. Now run this command:

- `ssh-add ~/.ssh/id_rsa`

9. When prompted for a passphrase, enter the one associated with the key.

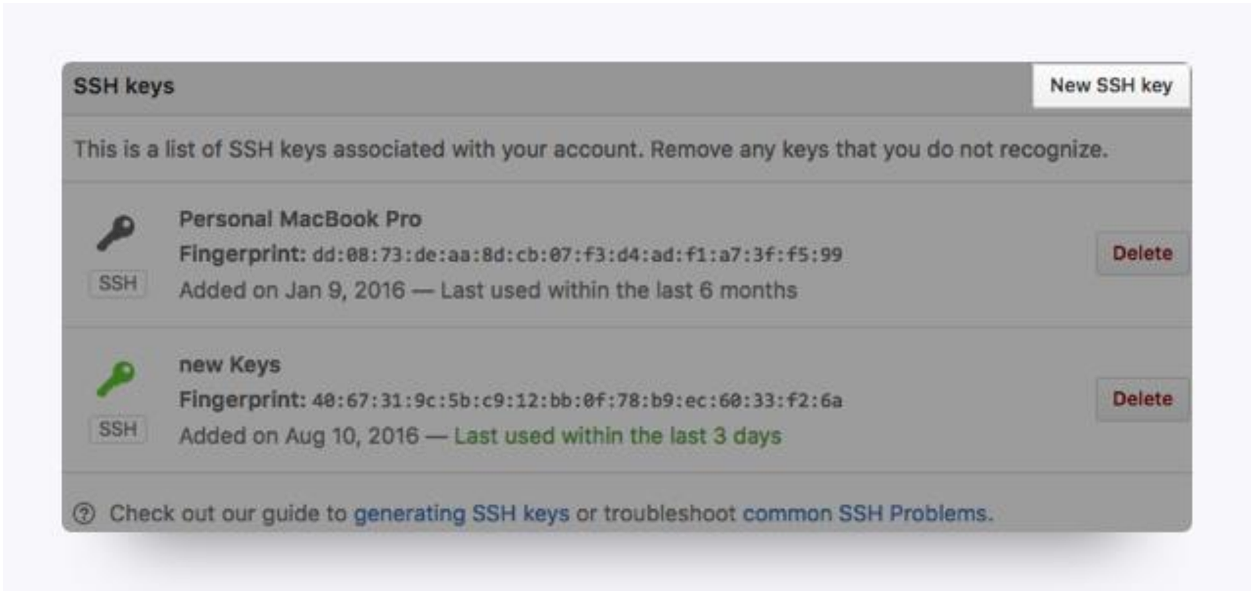
- If you've forgotten this password, just create a new one, starting with step 4.

10. We need to add the key to GitHub. Copy the key to your clipboard by entering this command:

- `clip < ~/.ssh/id_rsa.pub`

- You shouldn't see any kind of message when you run this command. If you do, make sure you entered it correctly.
- Do not copy anything else to your clipboard until you finish the next instructions. Otherwise, you'll have to repeat this step again.

11. Go to https://github.com/settings/ssh. Click the "New SSH Key" button.



SSH keys

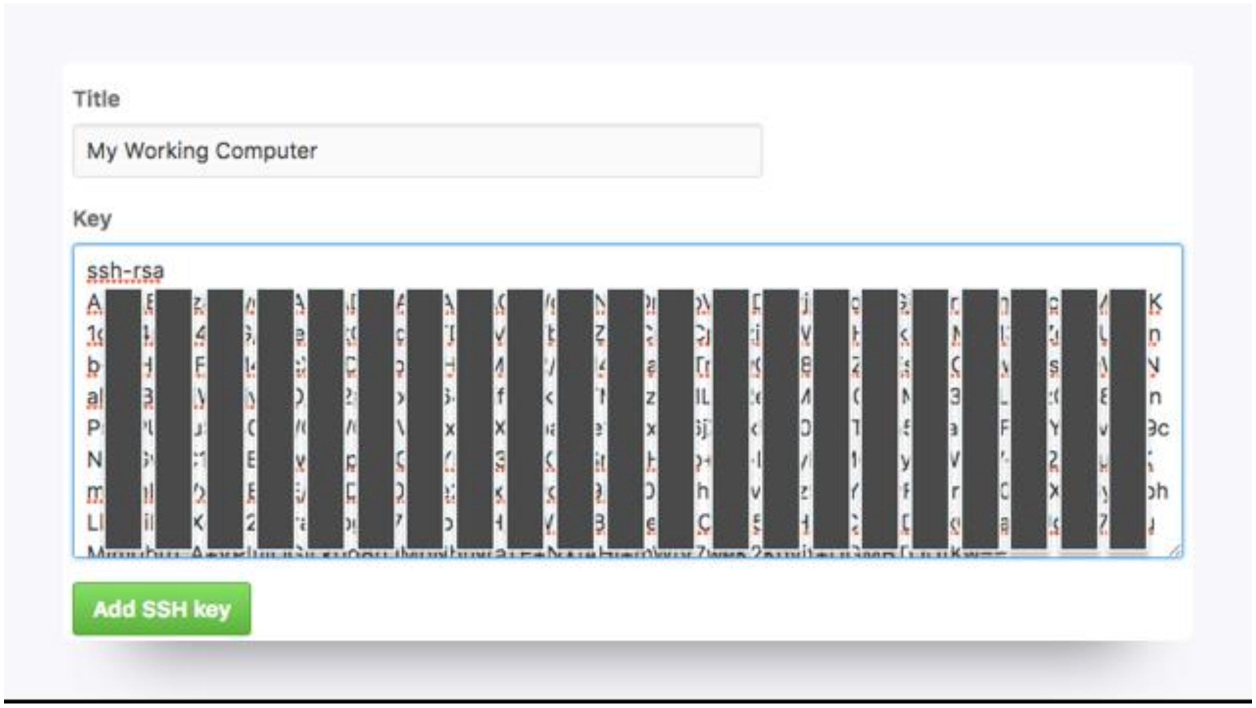
This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

Personal MacBook Pro
Fingerprint: `dd:08:73:de:aa:8d:cb:07:f3:d4:ad:f1:a7:3f:f5:99`
Added on Jan 9, 2016 — Last used within the last 6 months

new Keys
Fingerprint: `40:67:31:9c:5b:c9:12:bb:0f:78:b9:ec:60:33:f2:6a`
Added on Aug 10, 2016 — Last used within the last 3 days

Check out our guide to generating SSH keys or troubleshoot common SSH Problems.

12. When the form pops up, enter a name for your computer in the Title input. In the Key input, paste the SSH key you copied in step 10.



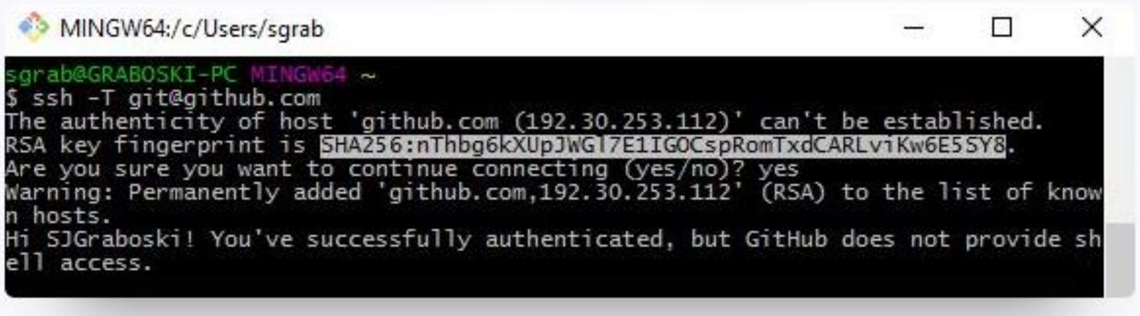
Title

My Working Computer

Key

ssh-rsa
[redacted]

13. Now we just need to add GitHub to your computer's list of acceptable SSH hosts. Go back to your Bash window. Type in this command: `ssh -T git@github.com`
   - You should see an RSA fingerprint in your window. Only enter "yes" if it matches the one highlighted in the image below.

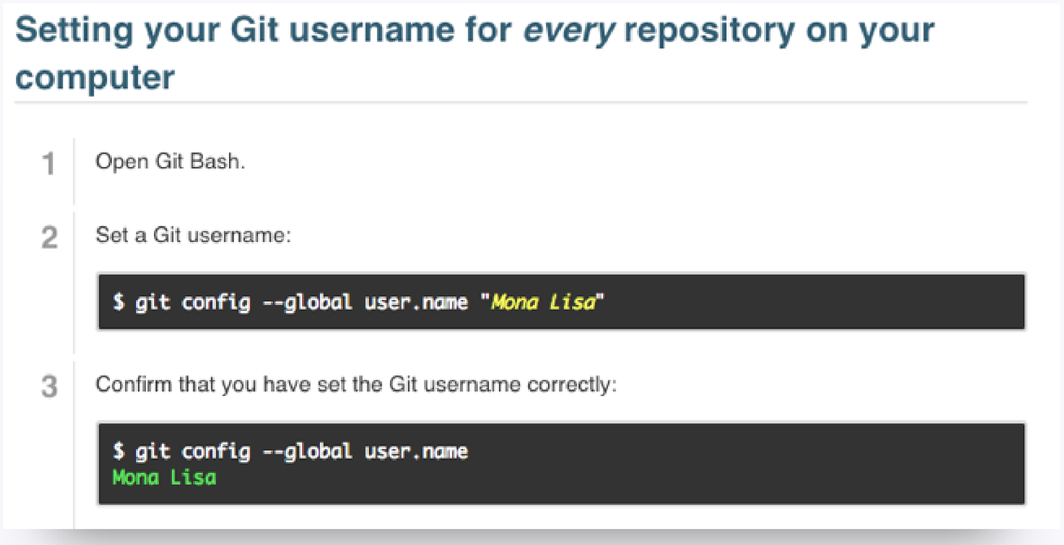


```
MINGW64:/c/Users/sgrab
sgrab@GRABOSKI-PC MINGW64 ~
$ ssh -T git@github.com
The authenticity of host 'github.com (192.30.253.112)' can't be established.
RSA key fingerprint is SHA256:nThbg6kXUpJWGl7E1IGOCspRomTxdCARLviKw6E5SY8.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'github.com,192.30.253.112' (RSA) to the list of know
n hosts.
Hi SJGraboski! You've successfully authenticated, but GitHub does not provide sh
ell access.
```

Setting your Git username for *every* repository on your computer

Git uses a username to associate commits with an identity. The Git username is not the same as your GitHub username.

You can change the name that is associated with your Git commits using the `git config` command. The new name you set will be visible in any future commits you push to GitHub from the command line. If you'd like to keep your real name private, you can use any text as your Git username. Changing the name associated with your Git commits using `git config` will only affect future commits and will not change the name used for past commits.



Setting your Git username for *every* repository on your computer

1 Open Git Bash.

2 Set a Git username:

```
$ git config --global user.name "Mona Lisa"
```

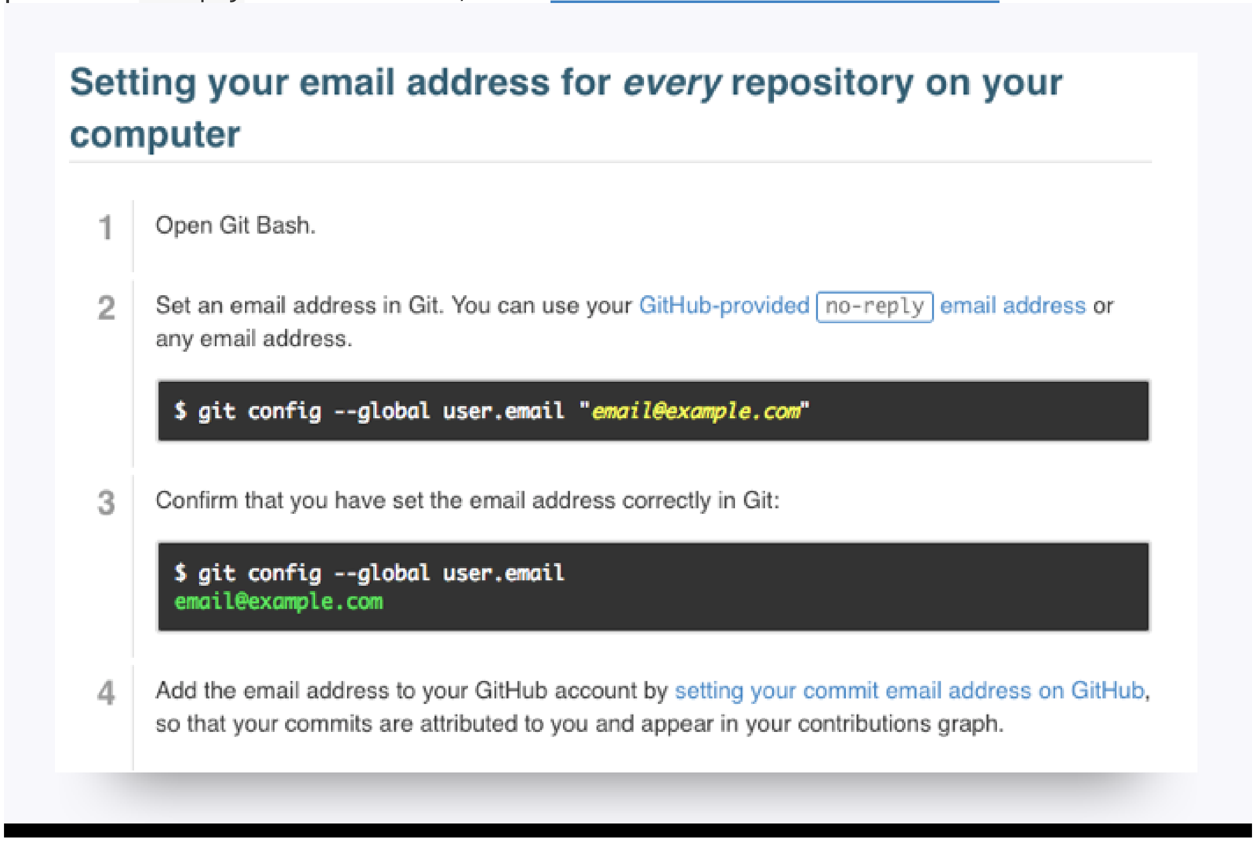
3 Confirm that you have set the Git username correctly:

```
$ git config --global user.name
Mona Lisa
```

Setting your email address for *every* repository on your computer

GitHub uses the email address set in your local Git configuration to associate commits pushed from the command line with your GitHub account.

You can use the `git config` command to change the email address you associate with your Git commits. The new email address you set will be visible in any future commits you push to GitHub from the command line. Any commits you made prior to changing your commit email address are still associated with your previous email address.
For more information on commit email addresses, including your GitHub-provided `noreply` email address, see "About commit email addresses."



Setting your email address for *every* repository on your computer

1. Open Git Bash.
2. Set an email address in Git. You can use your GitHub-provided `no-reply` email address or any email address.

```
$ git config --global user.email "email@example.com"
```

3. Confirm that you have set the email address correctly in Git:

```
$ git config --global user.email
email@example.com
```

4. Add the email address to your GitHub account by setting your commit email address on GitHub, so that your commits are attributed to you and appear in your contributions graph.