Configure SSH

For GitHub

Rick Miller, MS Computer Science
IT-566: Computer Scripting Techniques
Marymount University
Baslton Center, Arlington, VA

Contents

1 Introduction

Secure Shell (SSH) enables you to connect to and interact with your GitHub repository from the command line without the need to enter your GitHub username and password.

For inexperienced developers, SSH key generation and configuration can seem intimidating due to the need to enter terminal commands and explore the contents of hidden directories.

If you want to attempt the setup process using the GitHub instructions here's the link: https://docs.github.com/en/authentication/connecting-to-github-with-ssh/about-ssh

Fear not! While the instructions on GitHub are clear and well-written, this guide will walk you through SSH key generation and help you avoid common problems. 🙂

2 Assumptions

- You have a GitHub account
- Using Git Bash on Windows

3 Process Overview

- Verify the existence of or create ~/.ssh directory
- Create a ~/tmp directory to practice key generation
- Generate public and private SSH keys with a passphrase
- Copy the public and private keys to the ~/.ssh directory
- Add the public key to your GitHub account
- Add the private key to your local machine's SSH Agent
- Test your SSH key

4 All Operating Systems

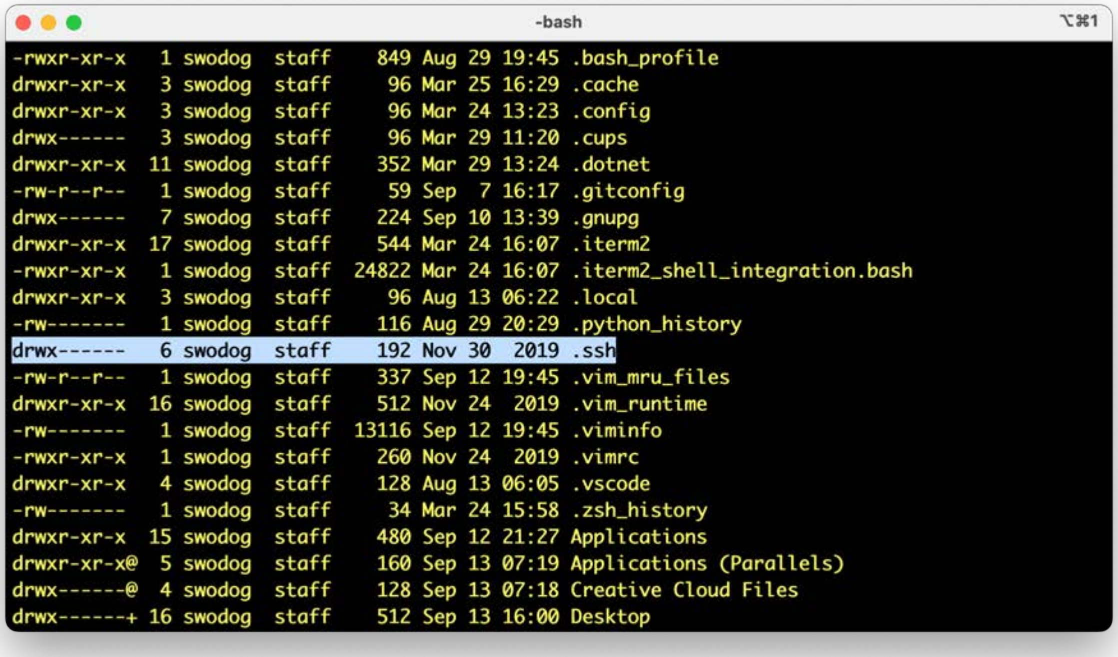
These steps are the same on all operating systems

4.1 Verify Existence of or Create ~/.ssh Directory

Check for the existence of the ~/.ssh directory. In your home directory (~) type:

```
ls -al
```

You should see the .ssh directory as shown in figure 1.



```
-rwxr-xr-x   1 swodog  staff    849 Aug 29 19:45 .bash_profile
drwxr-xr-x   3 swodog  staff     96 Mar 25 16:29 .cache
drwxr-xr-x   3 swodog  staff     96 Mar 24 13:23 .config
drwx------   3 swodog  staff     96 Mar 29 11:20 .cups
drwxr-xr-x  11 swodog  staff    352 Mar 29 13:24 .dotnet
-rw-r--r--   1 swodog  staff     59 Sep  7 16:17 .gitconfig
drwx------   7 swodog  staff    224 Sep 10 13:39 .gnupg
drwxr-xr-x  17 swodog  staff    544 Mar 24 16:07 .iterm2
-rwxr-xr-x   1 swodog  staff  24822 Mar 24 16:07 .iterm2_shell_integration.bash
drwxr-xr-x   3 swodog  staff     96 Aug 13 06:22 .local
-rw-------   1 swodog  staff    116 Aug 29 20:29 .python_history
drwx------   6 swodog  staff    192 Nov 30  2019 .ssh
-rw-r--r--   1 swodog  staff    337 Sep 12 19:45 .vim_mru_files
drwxr-xr-x  16 swodog  staff    512 Nov 24  2019 .vim_runtime
-rw-------   1 swodog  staff  13116 Sep 12 19:45 .viminfo
-rwxr-xr-x   1 swodog  staff    260 Nov 24  2019 .vimrc
drwxr-xr-x   4 swodog  staff    128 Aug 13 06:05 .vscode
-rw-------   1 swodog  staff     34 Mar 24 15:58 .zsh_history
drwxr-xr-x  15 swodog  staff    480 Sep 12 21:27 Applications
drwxr-xr-x@  5 swodog  staff    160 Sep 13 07:19 Applications (Parallels)
drwx------@  4 swodog  staff    128 Sep 13 07:18 Creative Cloud Files
drwx------+ 16 swodog  staff    512 Sep 13 16:00 Desktop
```

Figure 1 — Home Directory Showing .ssh Directory

If you don't see the .ssh directory, create it using the following command:

```
mkdir .ssh
```

Don't forget the dot '.' in front of ssh! Very important as it's a hidden directory. Verify once again the .ssh directory exists and when you're satisfied create a ~/tmp directory.

4.2 Create ~/tmp Directory

The purpose of the ~/tmp directory is to provide a space for you to practice SSH key generation without overwriting existing SSH keys you may have on your system. Of all steps in the SSH configuration process, it's key generation you may have to do a few times to get exactly right. I know it took me a few times when I first did it.

OK, create the ~/tmp directory using the following command. This is NOT a hidden directory so leave out the dot.

```
mkdir tmp
```

Navigate to the ~/tmp directory for the next step.

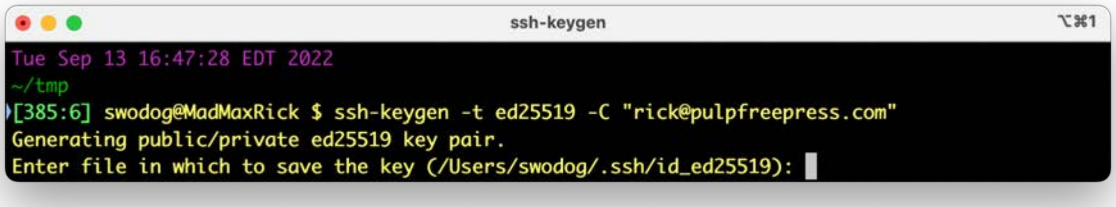
4.3 Generate SSH Keys

> ***NOTE:*** *Be careful. When you generate the key, the output location will be the ~/.ssh directory. That's OK if there are no keys in the directory, but you should specify the location where the keys are saved so they go into the ~/tmp directory for practice.*

Navigate to the ~/tmp directory and enter the following command, changing the email address to the one you used for your GitHub account.

```
ssh-keygen -t ed25519 -C "your_github_email@example.com"
```

Hit return. You will see the message: "Generating public/private id_ed25519 key pair." as shown in figure 2.



```
Tue Sep 13 16:47:28 EDT 2022
~/tmp
[385:6] swodog@MadMaxRick $ ssh-keygen -t ed25519 -C "rick@pulpfreepress.com"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/Users/swodog/.ssh/id_ed25519): 
```

Figure 2 — Generating SSH Public/Private Key Pair

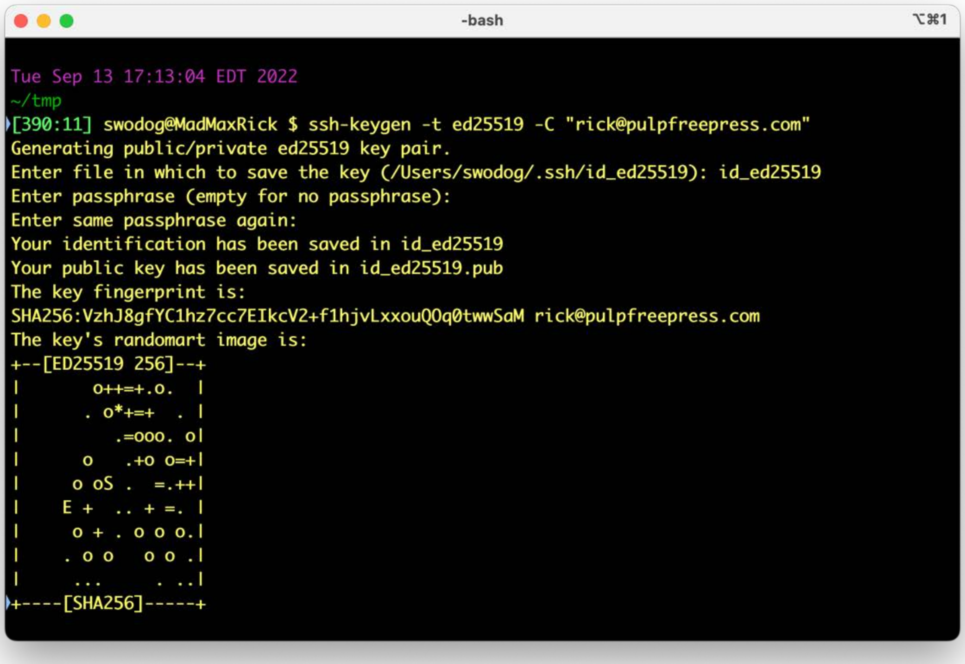
DON'T DO IT! But if you hit return now, the keys will be written automatically to the ~/.ssh directory.

- Since you are already in the ~/tmp directory simply enter the filename: id_ed25519. Hit return.
- Enter a passphrase. This will be the password you want to use for the SSH key. Hit return.
- Verify passphrase. Hit return.

If everything goes well, you'll see an output similar to figure 3 but with your email.

Let's decompose that command.

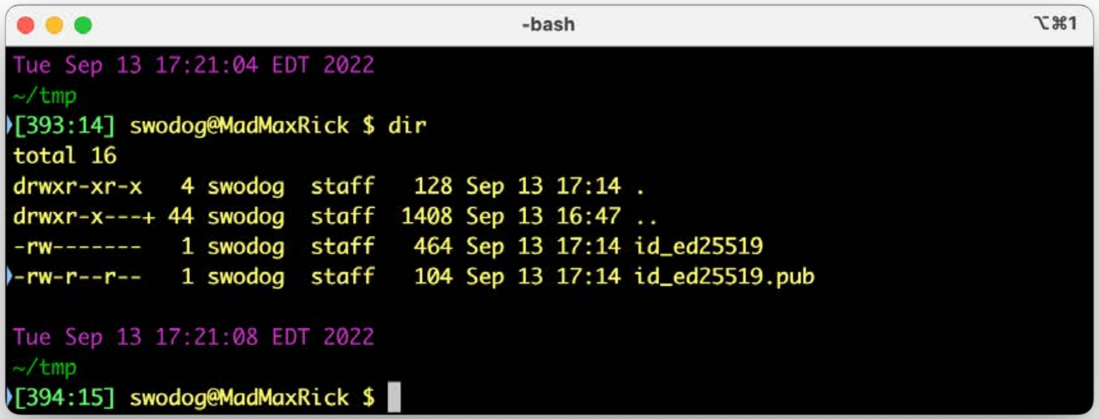
ssh-keygen	Command that generates SSH keys
-t ed25519	**-t** Specifies the type of key gen algorithm. In this case it's specifying the ed25519 algorithm.
-C "rick@pulpfreepress.com"	**-C** Add a comment. GitHub requires you to add your GitHub account email.



```
-bash                                                        ⌥⌘1

Tue Sep 13 17:13:04 EDT 2022
~/tmp
[390:11] swodog@MadMaxRick $ ssh-keygen -t ed25519 -C "rick@pulpfreepress.com"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/Users/swodog/.ssh/id_ed25519): id_ed25519
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in id_ed25519
Your public key has been saved in id_ed25519.pub
The key fingerprint is:
SHA256:VzhJ8gfYC1hz7cc7EIkcV2+f1hjvLxxouQ0q0twwSaM rick@pulpfreepress.com
The key's randomart image is:
+--[ED25519 256]--+
|       o++=+.o.  |
|      . o*+=+  . |
|         .=ooo. o|
|      o   .+o o=+|
|     o oS .  =.++|
|    E +  .. + =. |
|     o + . o o o.|
|    . o o   o o .|
|     ...     . ..|
+----[SHA256]-----+
```

Figure 3 — Key Gen Success!

OK, list the ~/tmp directory. You should see something like this:



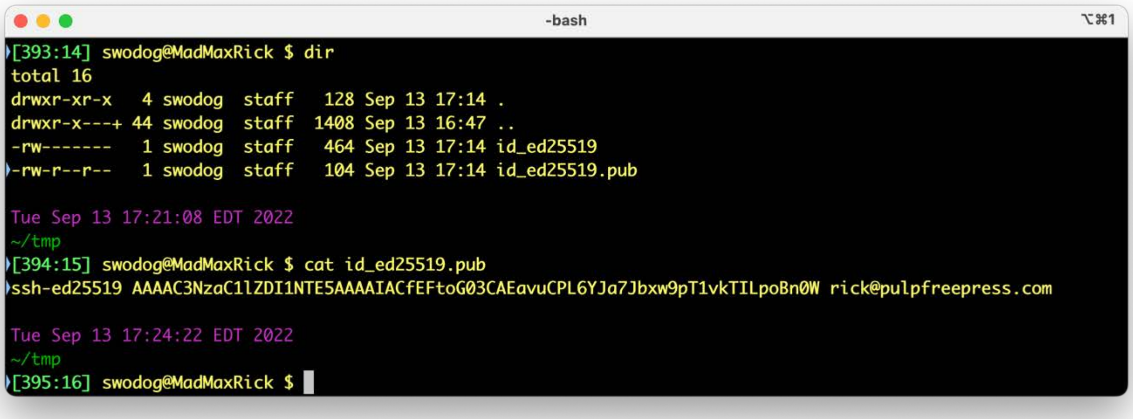
```
-bash                                                        ⌥⌘1
Tue Sep 13 17:21:04 EDT 2022
~/tmp
[393:14] swodog@MadMaxRick $ dir
total 16
drwxr-xr-x    4 swodog  staff   128 Sep 13 17:14 .
drwxr-x---+ 44 swodog  staff  1408 Sep 13 16:47 ..
-rw-------    1 swodog  staff   464 Sep 13 17:14 id_ed25519
-rw-r--r--    1 swodog  staff   104 Sep 13 17:14 id_ed25519.pub

Tue Sep 13 17:21:08 EDT 2022
~/tmp
[394:15] swodog@MadMaxRick $ 
```

Figure 4 — Public and Private Keys

You should see both a public and a private key. The public key ends in .pub. Now, list the contents of the public key by typing the following command:

```
cat id_ed25519.pub
```

You should see an output similar to figure 5.



```
-bash
[393:14] swodog@MadMaxRick $ dir
total 16
drwxr-xr-x   4 swodog  staff   128 Sep 13 17:14 .
drwxr-x---+ 44 swodog  staff  1408 Sep 13 16:47 ..
-rw-------   1 swodog  staff   464 Sep 13 17:14 id_ed25519
-rw-r--r--   1 swodog  staff   104 Sep 13 17:14 id_ed25519.pub

Tue Sep 13 17:21:08 EDT 2022
~/tmp
[394:15] swodog@MadMaxRick $ cat id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIACfEFtoG03CAEavuCPL6YJa7Jbxw9pT1vkTILpoBn0W rick@pulpfreepress.com

Tue Sep 13 17:24:22 EDT 2022
~/tmp
[395:16] swodog@MadMaxRick $
```

Figure 5 — Listing Contents of Public Key

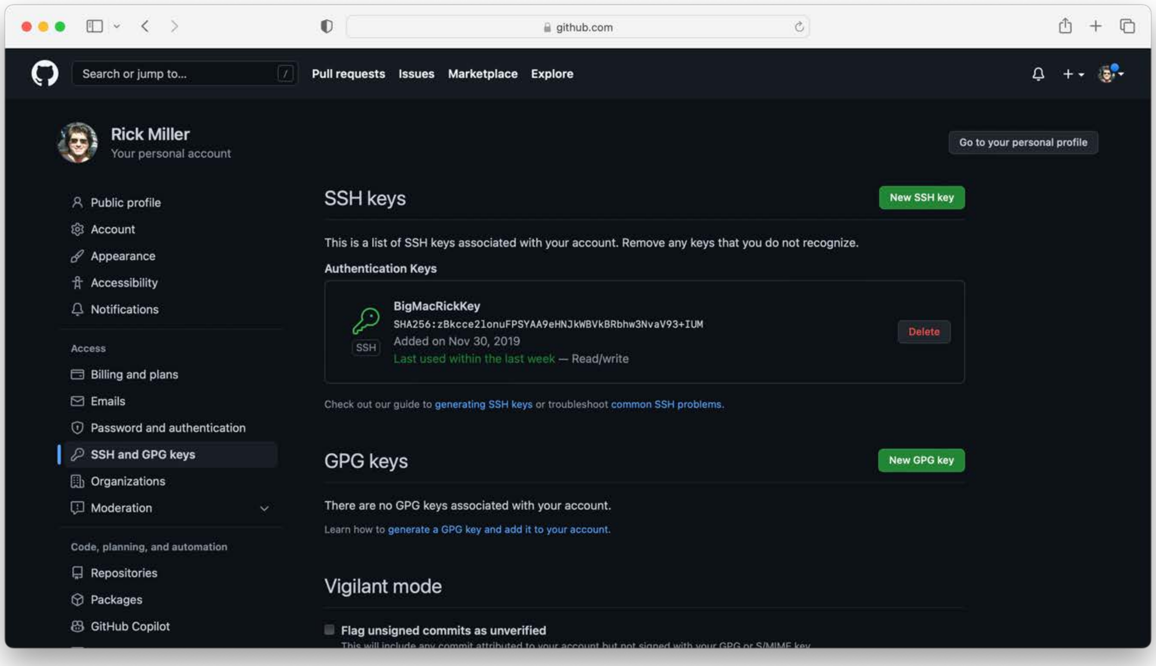
Your character string and email will be different.

OK, some things to consider. When you generate the keys you can name them anything you want. I sometimes use the term **devkey** or **githubkey** but using the default name is fine.

If you're happy with the keys you can copy them to the ~/.ssh directory.

4.4 Add Public Key To GitHub

Log into your GitHub account and navigate to the SSH keys page by clicking the dropdown on your account icon in the upper right corner, click Settings, then in the left column click SSH and GPG keys. This will open the SSH and GPG keys page as shown in figure 6.



SSH keys

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

Authentication Keys

BigMacRickKey
`SHA256:zBkcce2lonuFPSYAA9eHNJkWBVkBRbhw3NvaV93+IUM`
SSH
Added on Nov 30, 2019
Last used within the last week — Read/write

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

GPG keys

There are no GPG keys associated with your account.

Learn how to generate a GPG key and add it to your account.

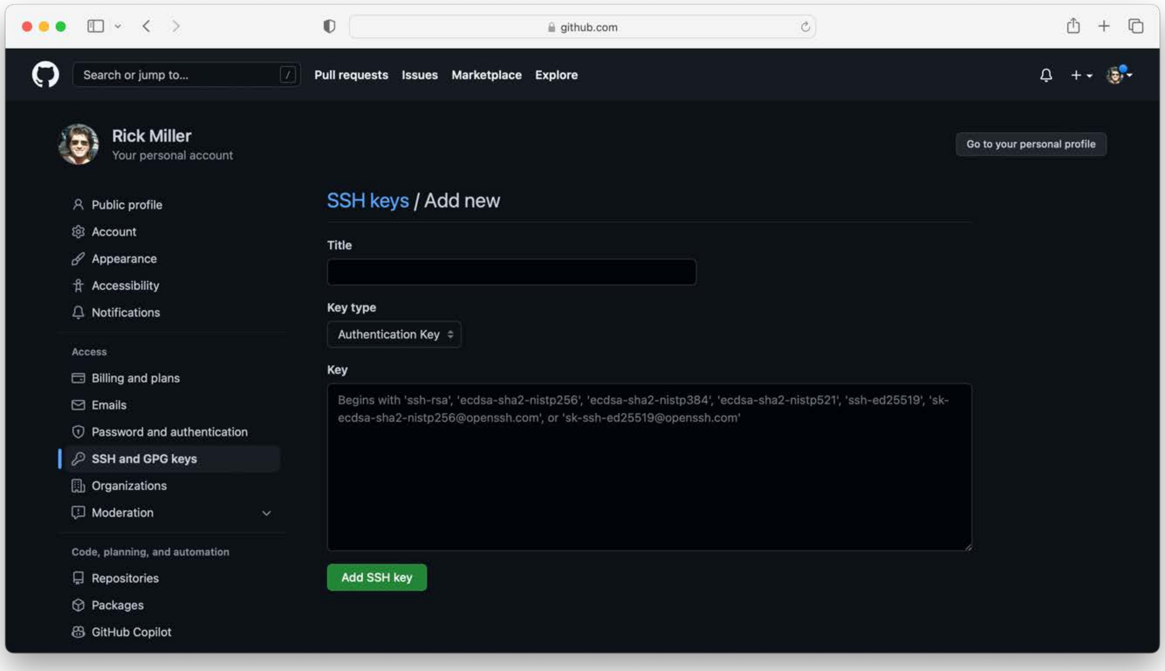
Vigilant mode

Flag unsigned commits as unverified

This will include any commit attributed to your account but not signed with your GPG or S/MIME key

Figure 6 — GitHub SSH and GPG keys Page

Referring to figure 6 — I have one SSH key listed for my account. To add a new key, click the green New SSH key button. This opens the New SSH key page as shown in figure 7.



SSH keys / Add new

Title

Key type

Authentication Key

Key

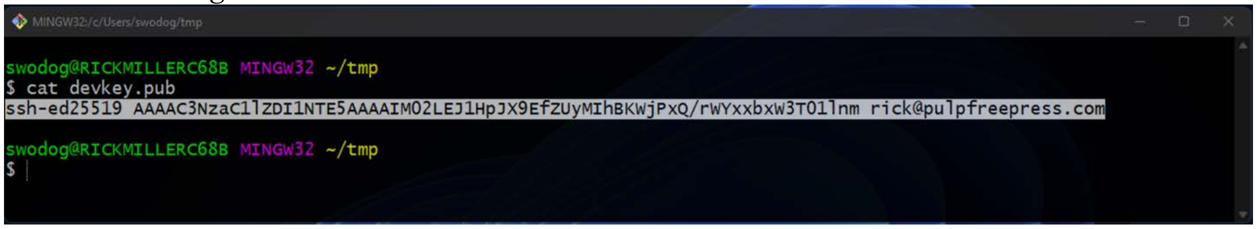
Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'

Figure 7 — New SSH Key Page

Enter a name for the key in the **Title** textbox. Leave the **Key type** dropdown set to Authentication Key, and list and copy the contents of your SSH public key into the **Key** textbox. You can list your public SSH key, which now should be located in the ~/.ssh directory, with the following command:

```
cat keyname.pub
```

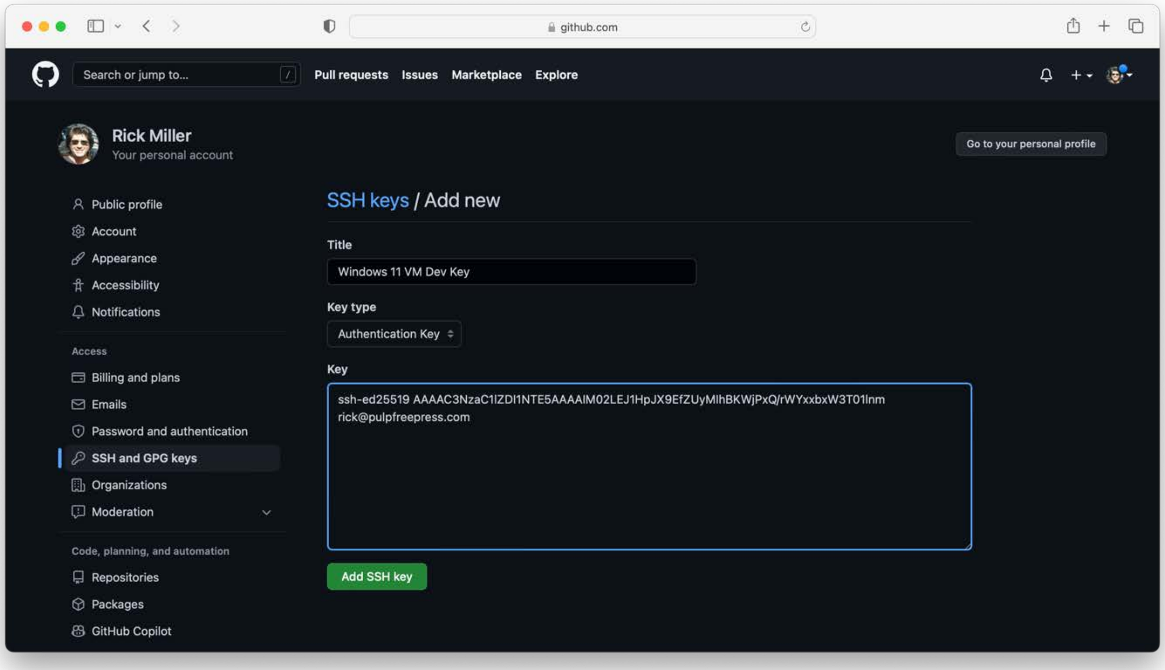
Where keyname is the name you used to generate the key. Copy the key text by selecting the text with your mouse, right-click, and select Copy from the dropdown as shown in figure 8.



```
swodog@RICKMILLERC68B MINGW32 ~/tmp
$ cat devkey.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIM02LEJ1HpJX9EfZUyMIhBKWjPxQ/rWYxxbxW3T01lnm rick@pulpfreepress.com

swodog@RICKMILLERC68B MINGW32 ~/tmp
$ 
```

Figure 8 — Select and Copy Public Key Text

Next, copy the public key text into the GitHub SSH Key textbox as shown in figure 9.



SSH keys / Add new

Title: Windows 11 VM Dev Key

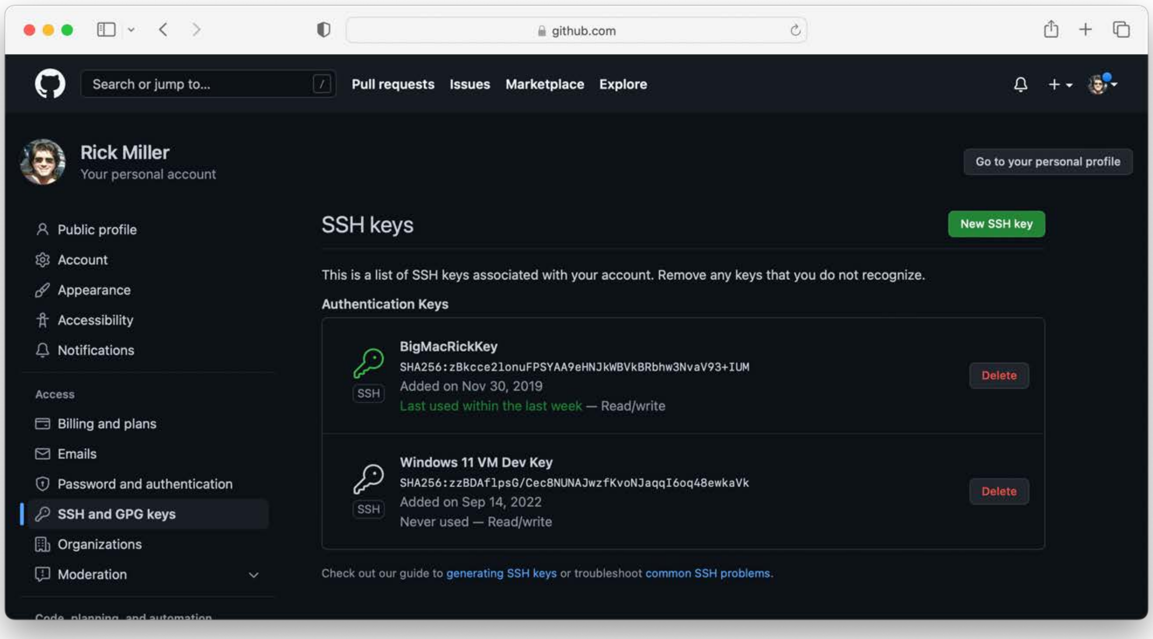
Key type: Authentication Key

Key:

```
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIM02LEJ1HpJX9EfZUyMIhBKWjPxQ/rWYxxbxW3T01lnm
rick@pulpfreepress.com
```

Figure 9 — Paste Public Key Text Into Key Textbox

Finally, name your new SSH key and click the green Add SSH key button. If you have two-factor authentication enabled you may get a pop-up asking you to authenticate. Figure 10 shows the new SSH key added to the list.



SSH keys

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

Authentication Keys

BigMacRickKey
SHA256:zBkcce2lonuFPSYAA9eHNJkWBVkBRbhw3NvaV93+IUM
Added on Nov 30, 2019
Last used within the last week — Read/write

Windows 11 VM Dev Key
SHA256:zzBDAflpsG/Cec8NUNAJwzfKvoNJaqqI6oq48ewkaVk
Added on Sep 14, 2022
Never used — Read/write

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

Figure 10 — New SSH Key Added to GitHub

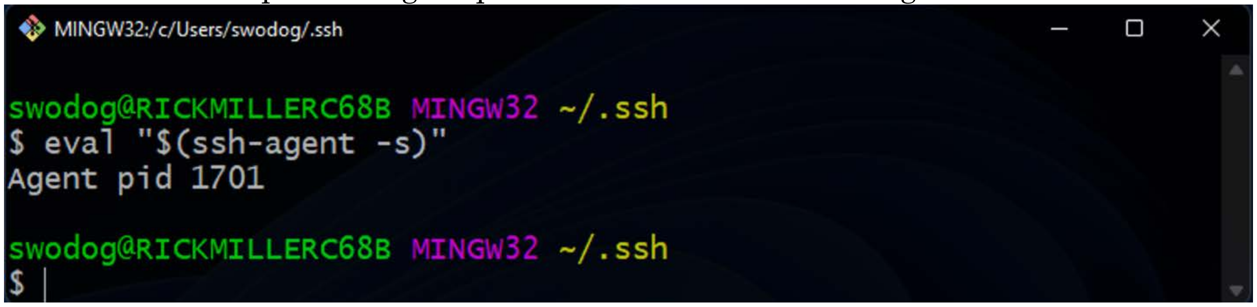
You will receive an email notifying you of the addition of a new SSH key to your GitHub account.

4.5 Add Private Key to ssh-agent

To use your SSH key to connect to GitHub, you need to add it to the ssh-agent. Navigate to the ~/.ssh directory and start the ssh-agent with the following command:

```
eval "$(ssh-agent -s)"
```

You should see a process Agent pid as a result as shown in figure 11.



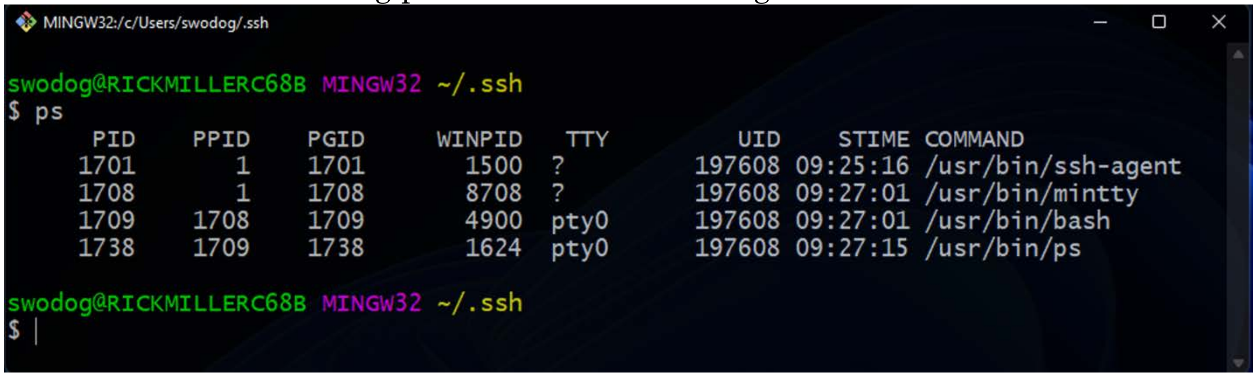
```
swodog@RICKMILLERC68B MINGW32 ~/.ssh
$ eval "$(ssh-agent -s)"
Agent pid 1701

swodog@RICKMILLERC68B MINGW32 ~/.ssh
$ |
```

Figure 11 — Starting ssh-agent in Git Bash

Note the pid number you see may be different. To verify the ssh-agent is running type the following command:

```
ps
```

You’ll see a list of running processes as shown in figure 12.



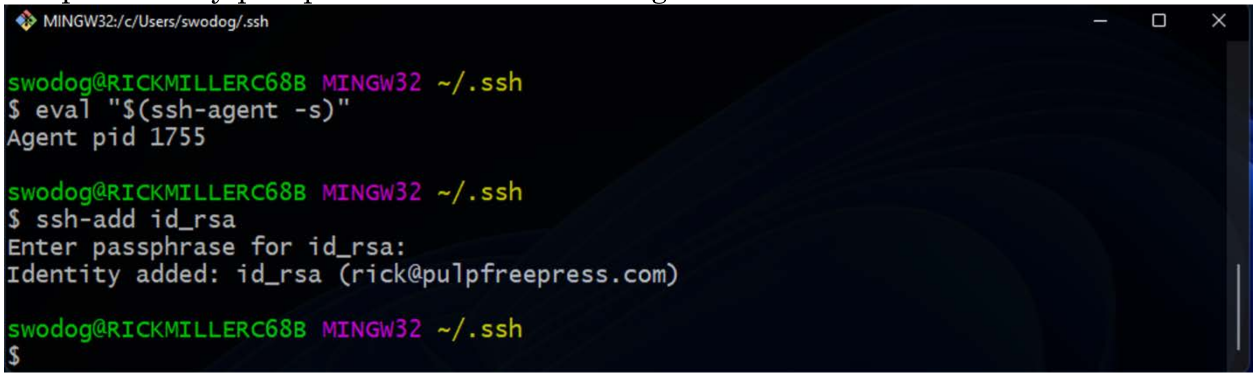
```
MINGW32:/c/Users/swodog/.ssh

swodog@RICKMILLERC68B MINGW32 ~/.ssh
$ ps
      PID    PPID    PGID     WINPID   TTY         UID    STIME COMMAND
     1701       1    1701       1500  ?         197608 09:25:16 /usr/bin/ssh-agent
     1708       1    1708       8708  ?         197608 09:27:01 /usr/bin/mintty
     1709    1708    1709       4900  pty0      197608 09:27:01 /usr/bin/bash
     1738    1709    1738       1624  pty0      197608 09:27:15 /usr/bin/ps

swodog@RICKMILLERC68B MINGW32 ~/.ssh
$ |
```

Figure 12 — List Running Processes with ps Command

With the ssh-agent running, add your private key to the ssh-agent using the following command:

```
ssh-add keyname
```

Where *keyname* is the name of your private SSH key. You will be prompted to enter the private key passphrase as is shown in figure 13.



```
MINGW32:/c/Users/swodog/.ssh

swodog@RICKMILLERC68B MINGW32 ~/.ssh
$ eval "$(ssh-agent -s)"
Agent pid 1755

swodog@RICKMILLERC68B MINGW32 ~/.ssh
$ ssh-add id_rsa
Enter passphrase for id_rsa:
Identity added: id_rsa (rick@pulpfreepress.com)

swodog@RICKMILLERC68B MINGW32 ~/.ssh
$
```

Figure 13 — SSH Private Key Added to ssh-agent

5 Testing Your SSH Key

OK, now you have generated public and private SSH keys. You added the public key to your GitHub account and added the private key to your ssh-agent on your machine. When you navigate to one of your GitHub repositories and click the green Code button you should see an SSH option as shown in figure 14.

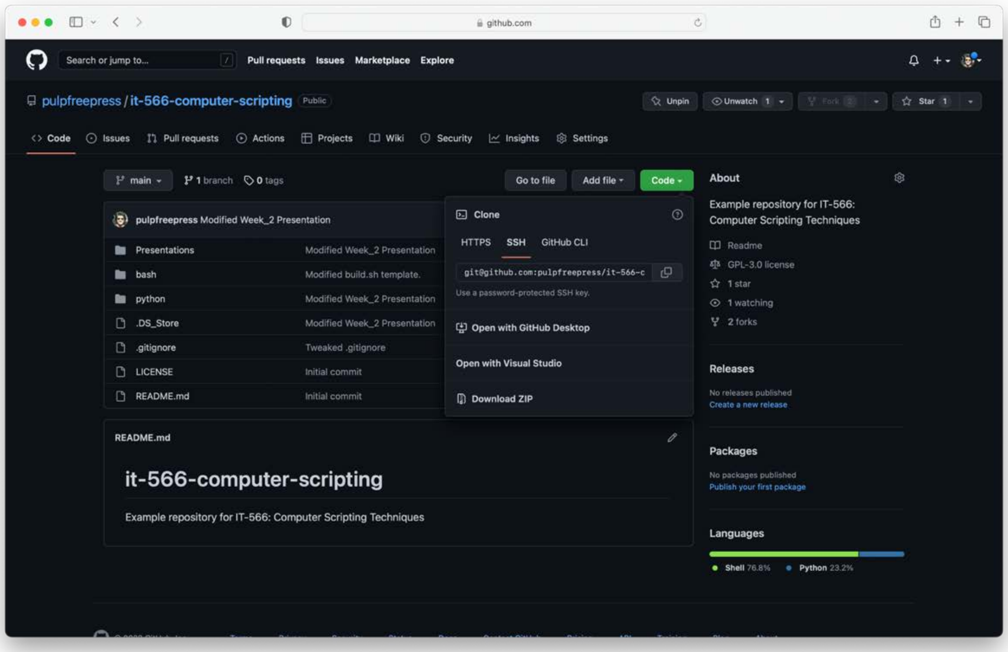
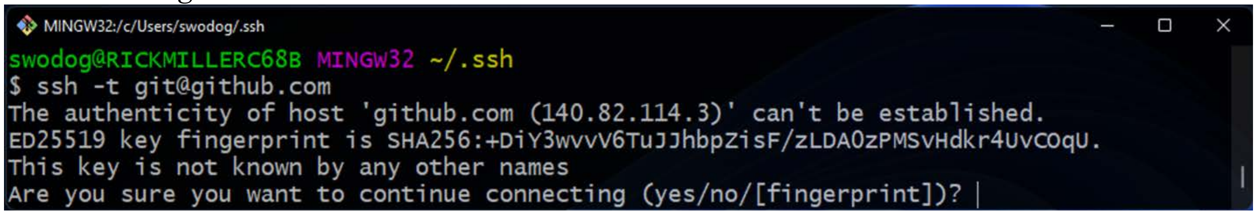

github.com
pulpfreepress / it-566-computer-scripting Public
Code Issues Pull requests Actions Projects Wiki Security Insights Settings
main 1 branch 0 tags
Go to file Add file Code
Clone
HTTPS SSH GitHub CLI
git@github.com:pulpfreepress/it-566-c
Use a password-protected SSH key.
Open with GitHub Desktop
Open with Visual Studio
Download ZIP
pulpfreepress Modified Week_2 Presentation
Presentations
bash
python
.DS_Store
.gitignore
LICENSE
README.md
README.md
it-566-computer-scripting
Example repository for IT-566: Computer Scripting Techniques
About
Releases
Packages
Languages
Shell 76.8%
Python 23.2%

Figure 14 — SSH Option

To test your SSH key, launch a terminal window on MacOS or the Git Bash terminal on Windows and type the following command:

```
ssh -T git@github.com
```

If this is your first time testing the key on your computer, you may see a message similar to figure 15.



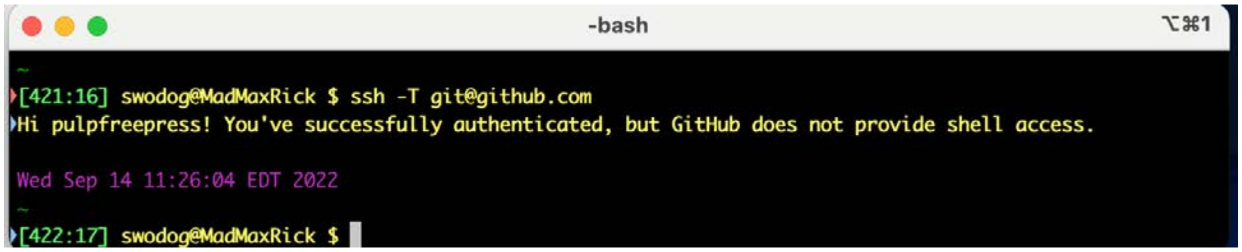
```
MINGW32:/c/Users/swodog/.ssh
swodog@RICKMILLERC68B MINGW32 ~/.ssh
$ ssh -t git@github.com
The authenticity of host 'github.com (140.82.114.3)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? |
```

Figure 15 — Fist Time SSH Key Test

Type yes and hit return. You will also be prompted for your passphrase. Enter it and hit return. If you get the following error in Get Bash…

```
PTY allocation request failed on channel 0
```

…close the Git Bash window, relaunch it, and try the test again. Second time's a charm. When the test succeeds, you will see something similar to the output shown in figure 16.



```
~
[421:16] swodog@MadMaxRick $ ssh -T git@github.com
Hi pulpfreepress! You've successfully authenticated, but GitHub does not provide shell access.

Wed Sep 14 11:26:04 EDT 2022
~
[422:17] swodog@MadMaxRick $
```

Figure 16 — Successful SSH Key Test

6 Test SSH Key On Repository

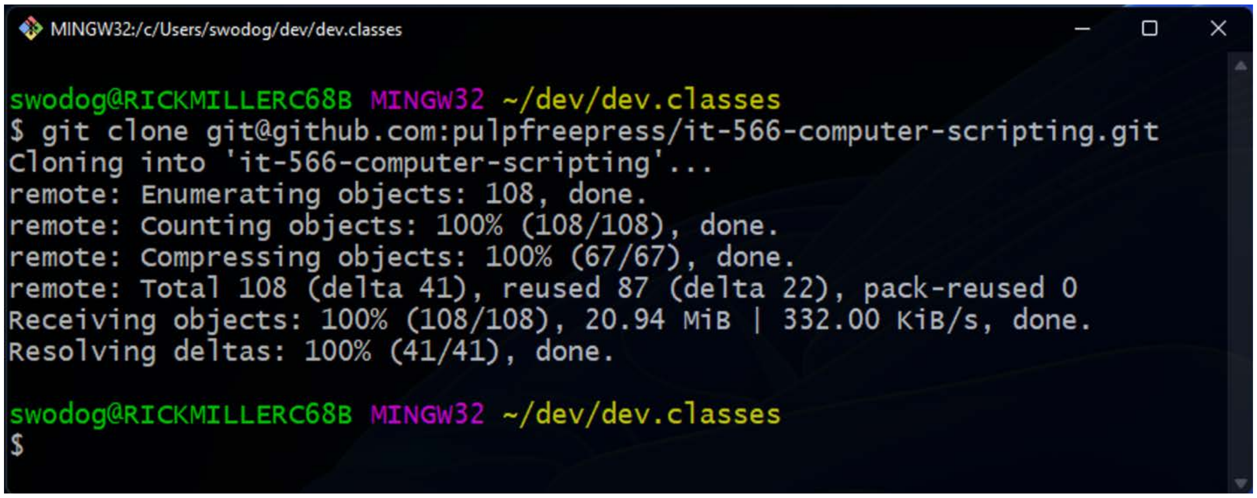
Clone one of your repositories using the SSH option. To do this, click the green Code button located in your GitHub repository window, select SSH, and click the copy icon. Navigate to a folder on your computer and type the following command…

```
git clone
```

…and paste in the SSH repo string you copied above. The full command will look something like this…

```
git clone git@github.com:pulpfreepress/it-566-computer-scripting.git
```

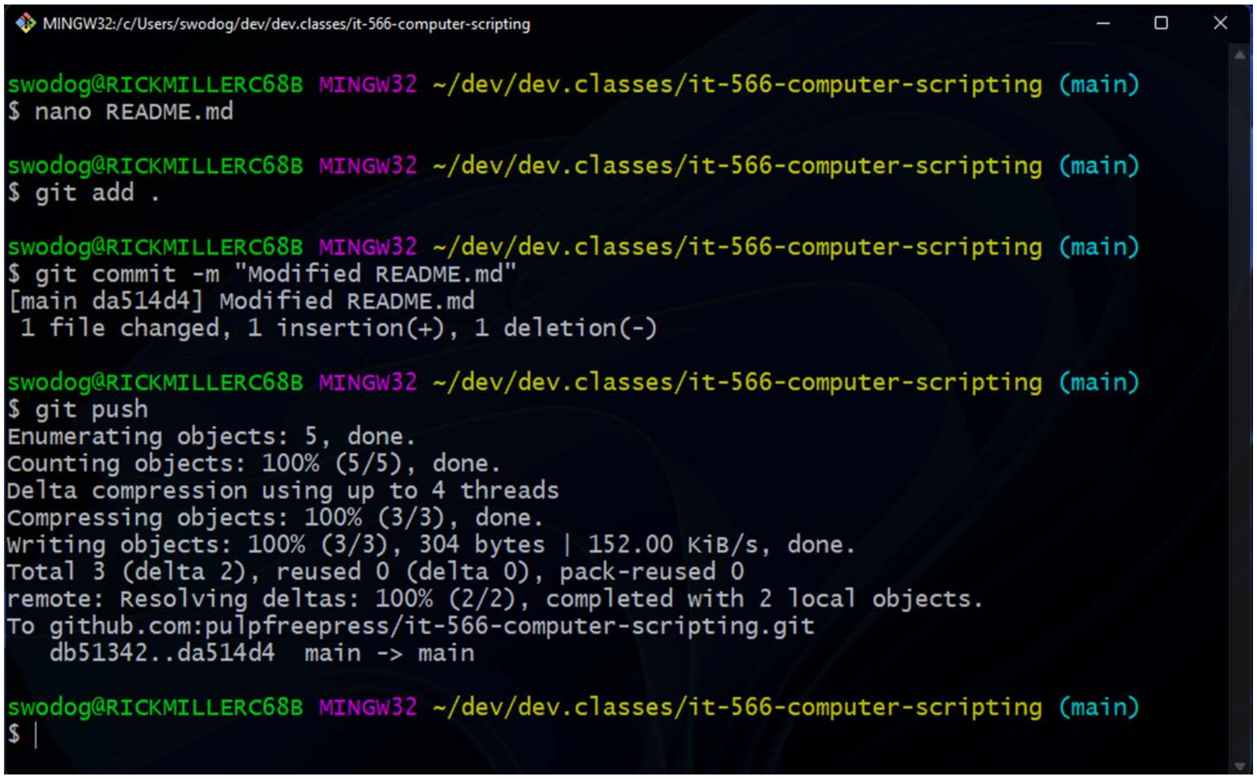
…but with the URL to your repository. You should see something similar to figure 17.



```
swodog@RICKMILLERC68B MINGW32 ~/dev/dev.classes
$ git clone git@github.com:pulpfreepress/it-566-computer-scripting.git
Cloning into 'it-566-computer-scripting'...
remote: Enumerating objects: 108, done.
remote: Counting objects: 100% (108/108), done.
remote: Compressing objects: 100% (67/67), done.
remote: Total 108 (delta 41), reused 87 (delta 22), pack-reused 0
Receiving objects: 100% (108/108), 20.94 MiB | 332.00 KiB/s, done.
Resolving deltas: 100% (41/41), done.

swodog@RICKMILLERC68B MINGW32 ~/dev/dev.classes
$
```

Figure 17 — Cloned Repo Using SSH

Next, edit and modify one of the files in your repository. Commit and push the changes. You will be prompted for your passphrase. Enter it and hit return. If all goes well (a phrase used a lot in this line of work) you should see something similar to figure 18.



```
MINGW32:/c/Users/swodog/dev/dev.classes/it-566-computer-scripting

swodog@RICKMILLERC68B MINGW32 ~/dev/dev.classes/it-566-computer-scripting (main)
$ nano README.md

swodog@RICKMILLERC68B MINGW32 ~/dev/dev.classes/it-566-computer-scripting (main)
$ git add .

swodog@RICKMILLERC68B MINGW32 ~/dev/dev.classes/it-566-computer-scripting (main)
$ git commit -m "Modified README.md"
[main da514d4] Modified README.md
 1 file changed, 1 insertion(+), 1 deletion(-)

swodog@RICKMILLERC68B MINGW32 ~/dev/dev.classes/it-566-computer-scripting (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 304 bytes | 152.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To github.com:pulpfreepress/it-566-computer-scripting.git
   db51342..da514d4  main -> main

swodog@RICKMILLERC68B MINGW32 ~/dev/dev.classes/it-566-computer-scripting (main)
$ |
```

Figure 18 — Successful push

7 Avoid Having To Type Passphrase All The Time

Depending on the operating system you have, you can add or modify configuration setting to avoid having to enter the SSH key passphrase for each push.

7.1 MacOS

Open a terminal and add a config file to the ~/.ssh directory. Edit the ~/.ssh/config file and add the following lines:

```
Host *
  AddKeysToAgent yes
  UseKeychain yes
  IdentityFile ~/.ssh/id_ed25519
```

Replace *id_ed25519* with the actual name of your private SSH key if necessary. Save the config file and relaunch the terminal.

7.2 Windows

Open Git Bash and edit the .bash_profile file and add the following lines:

```
eval "$(ssh-agent -s)"
ssh-add ~/.ssh/id_rsa
```

Save the file and relaunch the Git Bash window. You'll be prompted to enter the private key passphrase once when launching the window but not each time you need to push to a repository.

OK, that's about it.