Setting up GitHub Authentication in your VM

Note: If you're not familiar with working on a Linux system, these instructions might not make a lot of sense at first. It's OK to delay doing this part until after the second class in which we'll go over the basics.

GitHub is improving security related to accessing private repositories, and while this is a good thing for serious use, it does make it a little harder to set up for inexperienced users. The most immediate issue is that if you have two-factor authentication turned on with GitHub (a good idea!), then you will not be able to use the "HTTPS" method for cloning, pushing, or pulling repositories from GitHub. These instructions will guide you step-by-step through a setup procedure on your CSC362 virtual machine to make things compatible with the more stringent login/authentication settings on GitHub.

Read all the way through these instructions to understand the options you have, and then set up authentication in your VM.

SSH Authentication

In the first week's assignment, you are using the "slogin" program from ssh to log in and get a shell on a remote system. The "slogin" program is part of a larger collection of programs called "SSH" that use cryptography for a variety of helpful purposes, including authentication without the need to send a secret password to a remote system.

Keypairs and identity: The technology that allows this to happen is public key cryptography, where you create a "keypair" that consists of linked public and private keys. Unlike passwords, where the same secret value must be known on both sides of the connection, with public key authentication the remote system (the server - the one you authenticate to - GitHub in this case) only needs the public key half of your keypair, which is not sensitive. If someone were to steal the public key, that would not allow them to log in as you or to trick anyone into thinking they were you. To prove that you're you, you use the private key part, which will only be on your local system and should not be shared.

Step 1: Create a keypair

Your first step then is to create a keypair, which you do by typing this command at the shell in your virtual machine:

```
ssh-keygen
```

It will ask you for a "file in which to save the key" and you should just accept the default for that (just press enter). It will then ask you for a passphrase. You can leave this blank, which makes everything a little easier to use, but it is also less secure since your sensitive private key is stored in an unencrypted file in your VM. So for ease of use, leave the passphrase blank. For better security, use a passphrase. Below, I'll talk about a program named "ssh-agent" that allows you to keep some of the simplicity of a blank passphrase, while having better security.

Step 2: Add your public key to GitHub

Now you need to add the public key half of your keypair to GitHub, so it knows this key is authorized for you. Use Firefox in the virtual machine to log in to GitHub, and then click the top right icon that allows you to open up "Settings" for your account in GitHub.

On the settings page, click "SSH and GPG keys" on the left, and then click the green button labeled "New SSH key" on the following page. This prompts you for a "title" and a "key" - you can enter anything you want for the "title", since that's just a note to remind you want this key entry is for. To enter the key, go back to your shell and type the following command:

```
cat ~/.ssh/id_rsa.pub
```

This will type a bunch of gibberish and then output a new shell prompt. Use the mouse to select the gibberish (up to, but not including the prompt), and type ctrl-shift-c to copy that to the clipboard. Now go back to the browser, click in the "Key" input area, and then type ctrl-v to paste the key. Finally, click the green "Add SSH key" button at the bottom, and you're all set!

Step 3: Test it out

Back at the browser, navigate to your weekly-xxxxx repository in GitHub, click on the green "Code" button, select "SSH" and then click on the clipboard icon to copy the SSH/Git address for your repository. Back in the terminal window, type "git clone" followed by a space, and then ctrl-shift-v to paste the SSH/Git address, and then press enter.

If you entered a passphrase when you created your keypair, you'll need to enter it now. The repository should now be cloned on your local system (in your VM). This should work whether or not you have two-factor-authentication enabled in GitHub.

That's all there is for the basic setup!

Optional: Using ssh-agent

If you set a passphrase when you created your keypair, what happens is that your private key is stored on your local system in encrypted form, and so any time it is loaded from its file you will need to type the passphrase to decrypt it. To make this a little easier to use, there's a program called "ssh-agent" that can load your private key from the file and store it in memory in ready-to-use decrypted form. You will need to enter the passphrase the first time, but after that whenever a program like git needs the private key it can be simply loaded from memory. This way, the long-term key storage is still in an encrypted file, but you don't have to enter the passphrase every single time you use git (or anything else that uses SSH authentication).

To configure this, just type

```
ssh-add
```

in your shell. This will prompt you for the passphrase, so it can load your private key, and then every time you use git to access your private GitHub repositories after that, you won't need to enter the passphrase. Keep in mind that this doesn't persist, so if you log out or shut down your virtual machine, you'll need to do "ssh-add" again to load the key.