Cloning a Repository

This chapter discusses how to clone a repository. This is most likely the first thing one will do with a git repo. With modern git platforms like github, bitbucket, gitea and gitlab, initialization of repositories is not typically done.

Instead, the repo is initialized on the server managed by one of these git platforms and then the repository is cloned onto your local machine.

To clone a repository

There are two protocols used to clone a repository: ssh and https. Both are secure and encrypted in their transfer and both are rather quick. I prefer to use ssh as my git workflow is improved with the use of this protocol. This is because, particularly with the lengthy passphrases required by Indiana University, the IU Github instance is a pain to pull and push from. By using ssh, my key can have whatever password I want or no password at all while still maintaining a high level of security.

Here are some of the primary differences between the two protocols with regard to how one interracts with them in git. You can switch the protocol used after the initial clone. However, it is simpler to clone using the protocol that you prefer right from the start.

SSH

More initial setup is required to use ssh with git. The urls for cloning using ssh typically look something like this

`git@bitbucket.org/ksteimel/test_project.git`

Let's break this down really quick. The beginning `git@` part says that this ssh protocol is actually using the git user on the server. In order for this to work, the server needs to have a preshared key from our local machine.

To generate this key and put it on the server, follow the following instructions on a mac or linux machine:

1. Check to see if there is a file in `~/.ssh/` that ends in `.pub` . The default file is normally called `id_rsa.pub` on most modern systems.
2. If this file does exist, and you know the password associated with this key, simply open the file in your favorite text editor, copy the content to your clipboard and paste the contents into a new key in the web interface to the git server. Creating a new key is usually done by accessing your user settings (by clicking on your user icon and then clicking 'Settings') and then going to the submenu dealing with SSH & GPG keys.
3. If this file does not exist, create the file
   1. Run the command `ssh-keygen` on your local machine as the user you would like to use for this repo.
   2. This program will prompt you for the location where you would like to store the keys as well as the password for the keys. Typically, I leave these both at their defaults on OpenSuse which is `~/.ssh/id_rsa.pub` for the location and nothing for the password.
   3. If you have a different key location/name, you will need to add some content to your ssh config file. In `~/.ssh/config` put the following:

   ```
   Host github.com
   IdentityFile <path to identify file>
   User git
   ```

   4. Never upload your private key. This will not work and it will also leave your system exposed. Private keys are to be guarded closely, public keys are to share around.

4. After the public key file has been created, you should add this public key to the git server's web management system using the instructions in 2

HTTPS

To use https, simply select the https url option when cloning. There is no additional setup required. However, every time you push or pull, you will have to enter your username and password. If you use the credential helper, this can allow you to avoid this annoyance. To do this, run the following `git config credential.helper store` . You should only be prompted for your username and password one more time and then git should remember. If you need to change your credentials for any reason, just rerun that command and then you'll be prompted for credentials the next time you pull.

Revision #5

Created Mon, Sep 17, 2018 9:43 PM by kenneth

Updated Thu, Feb 14, 2019 3:50 PM by kenneth