VScode and GitHub Lesson Plan

VScode - SSH Turing/Hopper Guide

What is an “SSH”? - An SSH, or “Secure Shell”, is a cryptographic network protocol for operating network services securely over an unsecured network. SSH’s are important as you can use them to securely connect to servers remotely, securely transfer files, and more.

To grab your SSH to connect to Turing/Hopper in VScode, here's what you have to do.

SSH Folder:

1. Open up the terminal on your computer/laptop
2. Check if SSH is installed by typing ssh
3. log onto turing/hopper as so: ssh zXXXXXXX@turing.cs.niu.edu, yes, and then type your password
4. Once logged in, type ls -a in the terminal to see if you have an SSH folder, if not, type mkdir .ssh to create it
5. Log out of turing/hopper

Grabbing SSH:

1. Type ssh-keygen in the terminal (it is not necessary to add a passphrase)
2. Hit enter until your public and private keys are created
3. Type cd .ssh to change directories and type dir to find the id_ed25519.pub file
4. Type scp id_ed25519.pub zXXXXXXX@turing.cs.niu.edu:~/.ssh/authorized_keys

5. Once uploaded, whenever you SSH into the server from your computer it will just look for the key instead of asking for your password.

6. Re-log into turing to make sure you instantly log in

Connecting in VScode:

Once you have completed the earlier steps, here's what you have to do

1. Open VScode and download the extension SSH FS

2. Once in the SSH FS section, add a configuration and name it turing (or hopper) and hit save

3. Once in the edit page, scroll down to host and type turing.cs.niu.edu

4. For root, use /home/turing/zXXXXXXX/

5. For username, type your zID

6. Set private key by clicking prompt, and grab the id_ed25519 (no .pub) file from the .ssh folder (example: c:\Users\Dante\.ssh\id_ed25519)

7. If you added a passphrase, type it in the passphrase section, otherwise leave blank

8. Finally, click save

9. To connect to the terminal, click open remote SSH terminal

10. To view files, click the add to workspace button

And there you go! If you are using both a computer and laptop for classwork, you will need to do it again for both device's respective VScodes as they will not carry over.

VScode - Recommended Extensions

Doxygen - Extension that allows for a quick and easy way to make a documentation block

Git Graph - Allows you to view your repository on Github as a graph, which can help you keep track of things.

SSH Setup Video

Video: https://youtu.be/pPFAQ33YOVo

Git & Github Basics

Installing Git:

On Windows:

1. Head to the git webpage and find the installation option for Windows.
2. Once downloaded, open the executable where you will be prompted through the installation process.

On Mac:

1. Install homebrew.
2. From there, you can simply run **"brew install git"**
    a. For GUI, run **"brew install git-gui"**

On Linux:

1. Use package manager to install git (if not already pre-installed).
    a. For example, on Debian based distros run: **"(sudo) apt install git"**.
    b. To double check that it is installed, run: **"git --version"**.

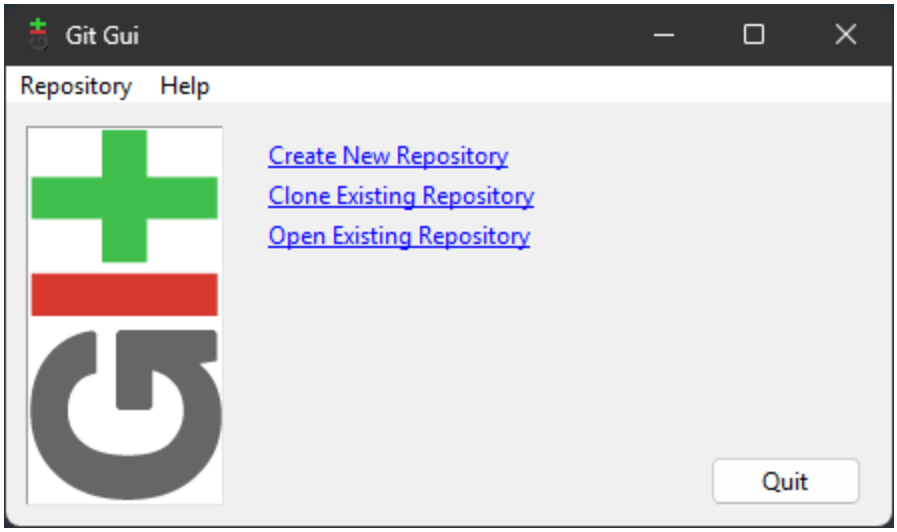
Using Git:

This is a very simple git introduction, for a more detailed guide I recommend checking the git documentation.

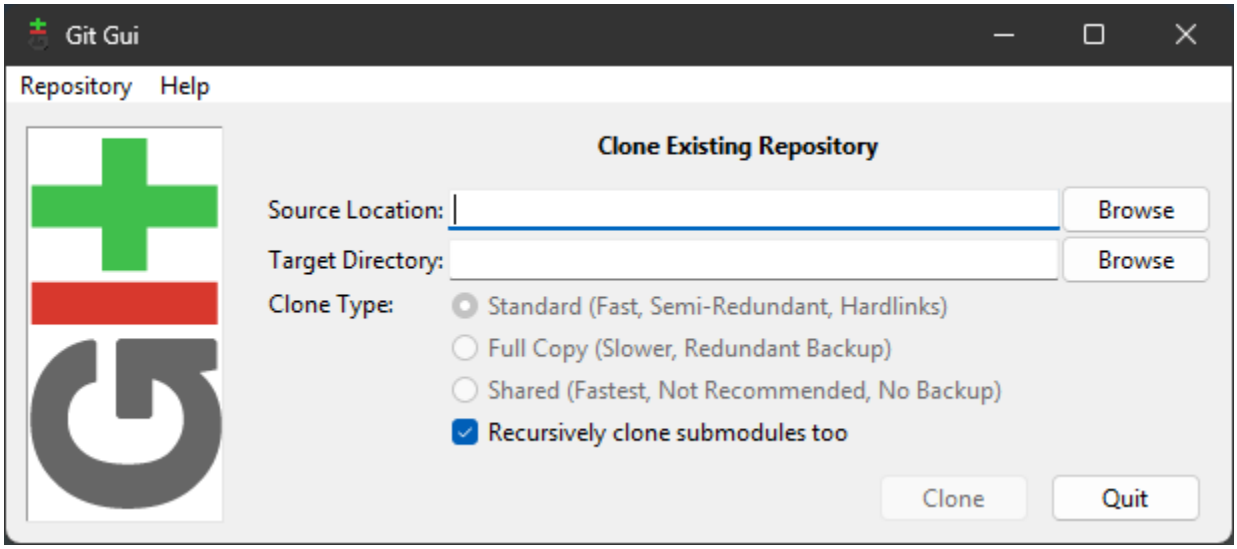
Basic Command Line Interface Usage:

- Initializing a repo: **git init** (from within a chosen directory)
    - I recommend consulting the documentation for specifics regarding initialization.
- Cloning a repo (e.g., off of Github): **git clone <repo url>**
- Checking status of repository: **git status**
    - This will display if there were modifications made to files, if files were created/added, and if files were deleted (among other things).
- Adding files/modifications to a commit: **git add <filename or wildcard * >**
    - If you want to add all changes, run: **git add ***
- Creating a commit with a message (best practice): **git commit -m "message"**
- Pushing a commit to a repo: **git push**
    - This will push to the default branch (main or master, which should be changed as github uses main).

Git GUI:

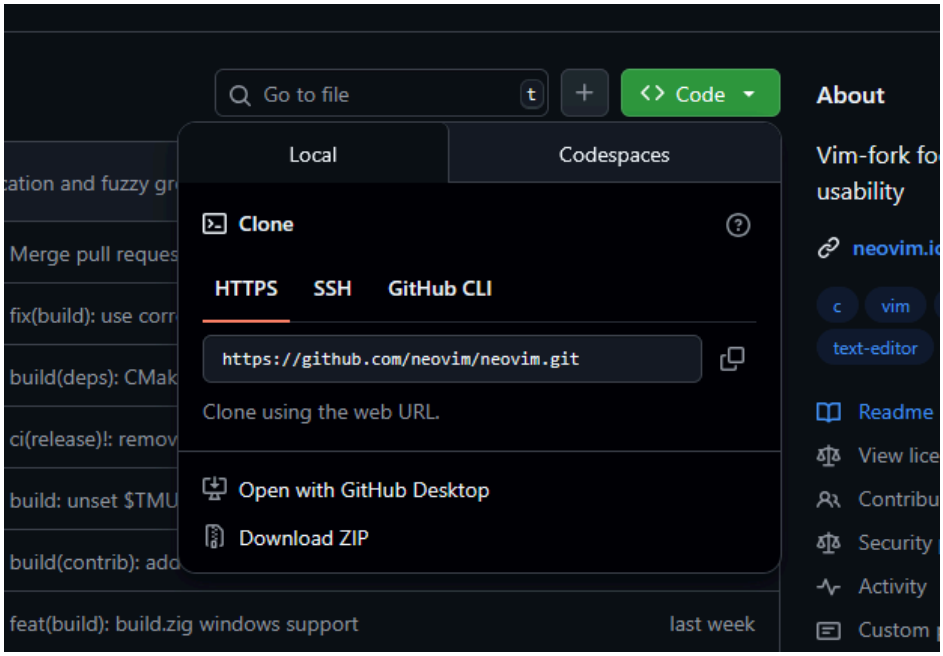


The functions of the GUI are pretty self explanatory, for our purposes you will most likely use the **“Clone existing repository”** option.

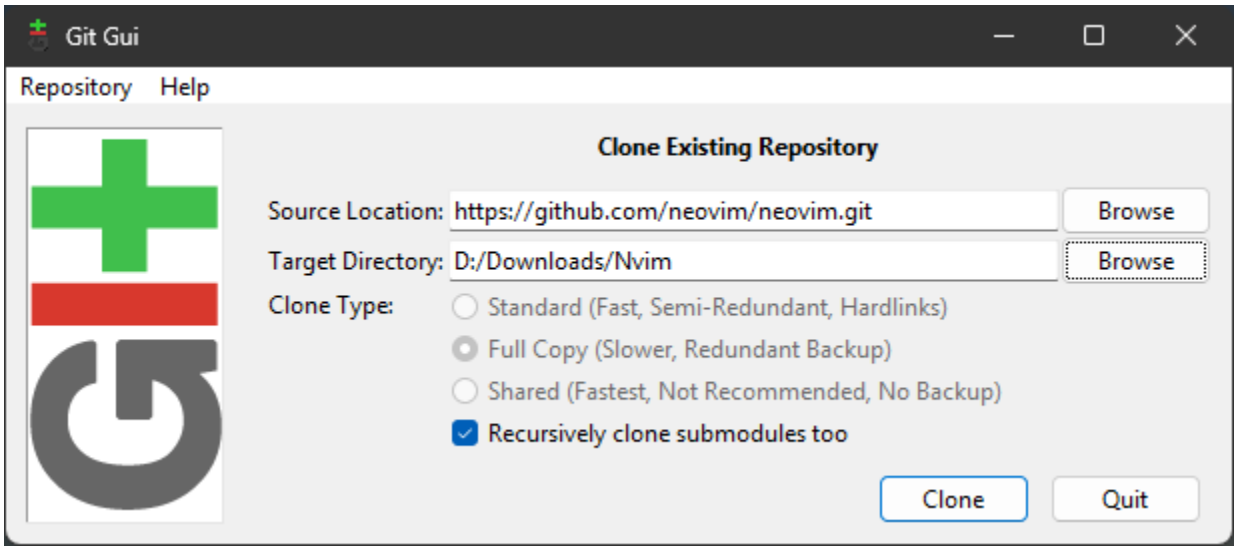


To clone a Github repository using the GUI:

- Copy the URL to the selected repo



- Enter the URL into the “Source Location” and choose a “Target Directory”



- This method can be used to clone other people’s repositories for personal use or to clone your own personal repositories.
    - If you initialize a repository locally, you would have to set a remote in order to ‘link’ it to the repository on your Github profile.
    - E.g., running: **“git remote add origin <repo URL>”**

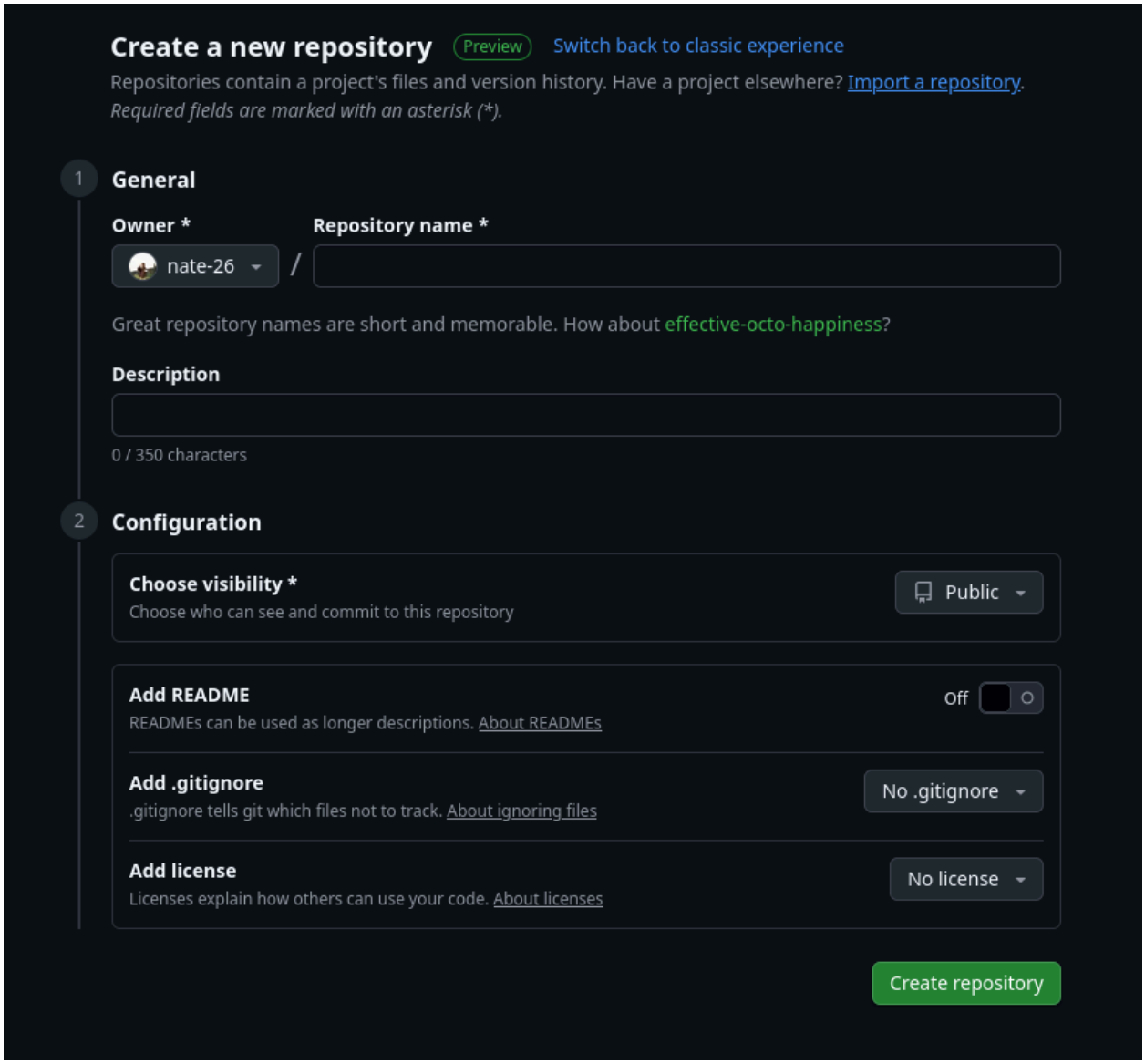
<u>Using Github</u>:

Github is a web based platform that uses Git for version control. With a free Github account you can host your own repositories, contribute to others, and make use of the various tools already hosted on there.

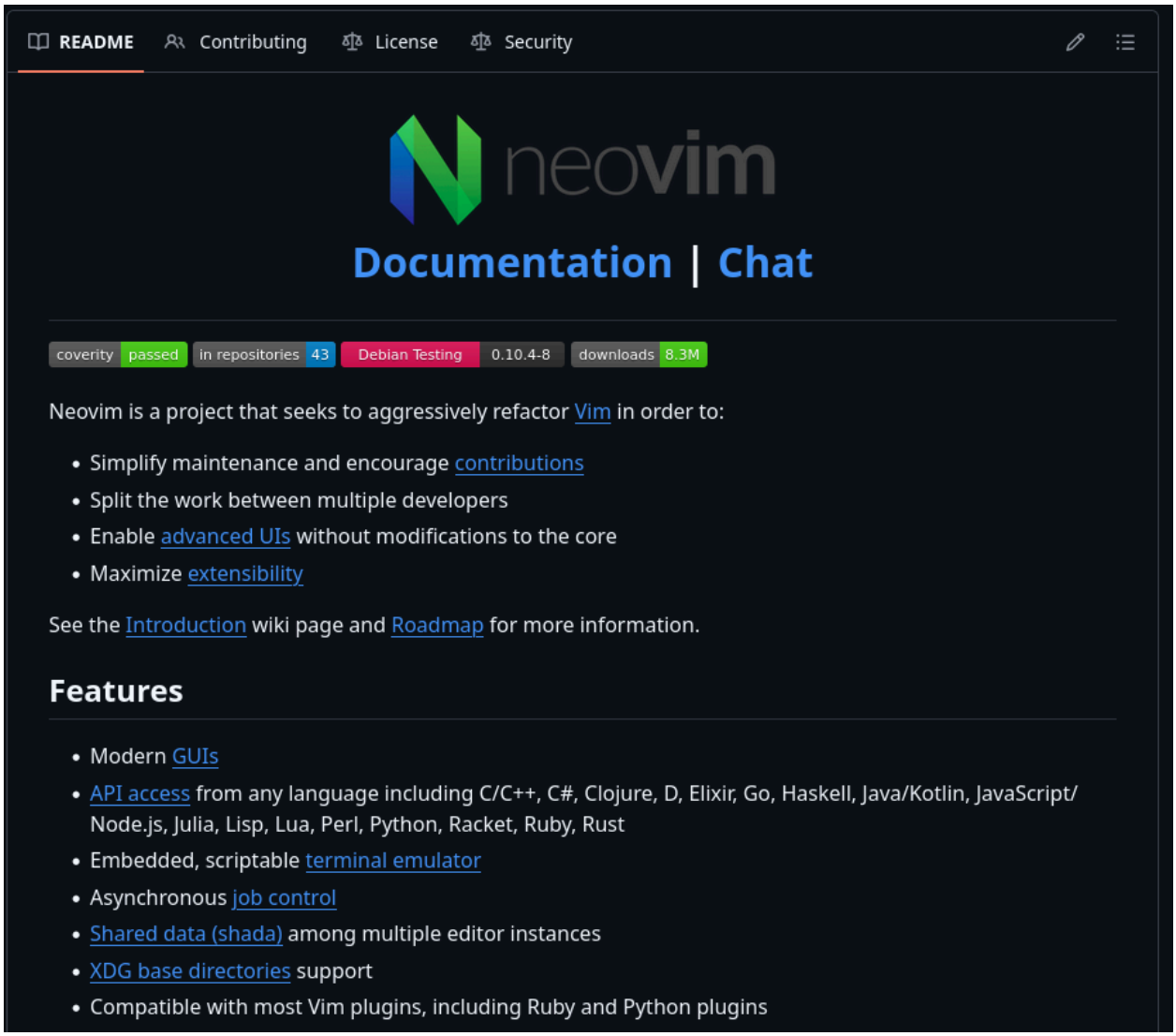
The first step to using Github is to **register an account** with them if you haven't already.

Hosting your own repositories:
Full disclosure, this is not the only method, nor is this the most efficient. The following method is the easiest for beginners to begin hosting their own repositories on Github.

- Begin by **creating a repository,** there should be a button on Github that says "New" or "New Repository".



- From there you would have to name the repository and choose whether or not you want it to be **Public or Private**.
    - Public repositories are visible to anyone who views your repository and they may clone whatever is within them.
- The **README** file is a markdown file that is used as a way to describe the contents/purpose of the repository. They may include installation guides, requirements, documentation, etc.



The image above is the README file for the neovim repository. README files are very useful for public repositories that have a lot of visitors/users.

- The **.gitignore** file is an important file that tells git what to ignore when “pushing” to the repository. Github supplies you with templates based on the type of project that repository is going to be used for.

    - Tip: do NOT push any passwords, authentication keys, etc., to your repositories. You should use a **.env file** instead, but that's beyond the scope of this lesson for now.

- Once created, you can **clone** the repository into your local device. Keep in mind you also have to be authenticated locally with your Github credentials (see below). When that's done you will be able to push your content to the repository on Github.

Local Authentication for Github:
In order to actually push your changes onto your remote repository, you have to login locally using your Github credentials.

- **VSCode:** This is the simplest method, VSCode can see when an open folder is a git repository and you can login from the version control tab.
- **Using Github CLI:** For this method you need to install the Github CLI. From there you just run "**gh auth login"** and follow along with the prompts. There is an option to authenticate with HTTPS, which will open a window on your browser. As long as you're logged into Github on your browser you will just have to input the code shown on the terminal.
    - For Linux users: there is a separate guide for installing on Linux. Once the repository is added you can simply run **apt install gh**.
    - You will also most likely be prompted to set a global username and email, set them to the same username and email used for your Github account.