Git and a GitHub account

Version control is an essential tool for code development and management, particularly in collaborative environments. At present, git is the dominant version control system in use and **GitHub (https://github.com)** is one of the most common resources to use for hosting projects with git. There is nearly limitless information on git available online. Here we will only highlight a few important details for this course. More discussion and much practice will occur in class and the projects assigned throughout the semester.

Git

Git was originally created by Linus Torvalds to maintain the Linux kernel. It is a decentralized version control system that can be viewed as the swiss army chain saw, machine gun, and tactical nuclear weapon of version control systems. It is extremely powerful and flexible. This also allows for "slip of the finger" commands to wreak tremendous damage!

GitHub

The first thing to know is that **GitHub (https://github.com/)** **is not git**. Git is a version control tool. **GitHub (https://github.com)** is an online service for maintaining repositories, encouraging collaboration, and has some added services to make these tasks easier. When you create a **GitHub (https://github.com)** account, it will be your personal account and part of your personal and professional profile. It is an "advertisement" for yourself that is used by employers when screening job applicants. It is a good idea to ensure that your public repositories reflect well on yourself! No project is too small to be included.

Unfortunately, **GitHub (https://github.com)** was recently bought by a (former?) monopoly with a history of atrocious practices. What this will mean for the future of **GitHub (https://github.com/)** is still to be determined, however, for the time being it remains the main place where people look for public git repositories.

It should be stressed there really is nothing special about **GitHub (https://github.com)**. It does **not need to be used** in conjunction with git. It is quite possible (and common) to have git repositories independent of **GitHub (https://github.com)**. In fact, this is the whole point of git! It is decentralized. Every copy of a repository is as "good" or "official" as any other. A site such as **GitHub (https://github.com)** provides a common place where people can agree to go to share repositories. It is a "social" agreement, not a requirement.

GitHub Account

For this course you are required to have a **GitHub** **(https://github.com)** account. We will use it for most of the work in the course. If you already have a **GitHub** **(https://github.com)** account, use it for this course. If you do not, then create one (it is free). To do so, go to **GitHub** **(https://github.com)** and follow the instructions there to sign up for an account. As noted above, this is your personal account and one you should use for "everything". The work in this course will be a small part of your personal profile.

Account Information

Once you have an account (either old or new) it will be necessary to get me that information. This will be discussed in class. This will be used for giving you access to course materials and for sharing a repository with group members.

Accessing GitHub from your Computer

A copy of your repositories are stored on the **GitHub** **(https://github.com/)** servers, but they will be far more useful when you also have a copy on your own computer(s). Communication can be done using the secure web protocol (https) or over a secure shell (ssh). We will always use ssh! This is a far better way to do things, particularly when used with ssh keys. We will set up ssh keys.

Generating ssh keys

We will go over this in class, however, to help you out some basic resources are given below. We will make some minor changes to these directions.

The main resource comes from **GitHub** **(https://github.com/)** itself. See their **Generating a new ssh key** **(https://help.github.com/en/articles/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent)** page for basic information. If you are interested in even more details see the **Mozilla Recommendations** **(https://infosec.mozilla.org/guidelines/openssh)** page on OpenSSH. They have a section on key generation also.

Very crudely, ssh uses public key encryption and ssh keys provide a much more secure way of establishing a secure connection than using passwords. Further, when used with your OS key ring (or whatever they call it) you should not need to enter your passphrase every time you connect to the **GitHub** **(https://github.com/)** servers to perform operations on your repositories. This makes things far more convenient.

Again, the details will be covered in class. For now here are a few details we will be using.

1. **Do not use an existing ssh key.** Though the directions on **GitHub (https://github.com/)** suggest using an existing key, do not do this. It is far better to create new keys for each service and machine you use. The Mozilla site above does this. When we use ssh-keygen we will specify the file to create.
2. **Choose your algorithm.** The two most common choices these days are rsa or ed25519. The standard and best supported one is rsa. This is an old algorithm that used to be patented (despite being "trivial" and setting back the adoption of encryption by decades). It also means it is well studied but also has the most attacks against it. If you use rsa you **must use 4096 bits**. A more modern and new algorithm based on elliptic curves is ed25519. It is less well studied but expected to secure.

Examples

Generating the key. As an example, when I generated a key for my work computer I used something like the following to generate a 4096 bit rsa key.

```
ssh-keygen -t rsa -b 4096 -f ~/.ssh/github_work -C "GitHub Work"
```

Alternatively, if you want to use ed25519 only the type needs to be specified, there is a fixed bit size.

```
ssh-keygen -t ed25519 -f ~/.ssh/github_work -C "GitHub Work"
```

Using the Key

When the key is generated there are two files created, one called github_work and one called github_work.pub (for the name I chose above, if you choose a different name one will have .pub after it). The .pub file is public, the other is your private key and **should never be shared**. A remote system only needs to know about your public key. If you private key ever gets comprised then the key is worthless. You can freely share your public key, never your private key.

This should be sufficient to have ssh use your key, However, I prefer to directly configure it. If nothing else, this helps me keep track of what machines the keys are used to connect to. **This is not completely necessary.** But I recommend it.

In your local ssh directory (typically in a subdirectory of your home directory called .ssh (note the preceding dot)) there will be (or can be created) a configuration file. For example I have the following lines on my work computer in the file *~/.ssh/config*

```
host github.com
        IdentityFile    ~/.ssh/github_work
```

Here *User* is the **GitHub** (https://github.com/) username you used for your account and *IdentityFile* is the file containing your **private key** as created above.

Once this is set up you need to tell **GitHub** (https://github.com/) that this key is allowed to connect to your repositories. See **Adding a new ssh key to GitHub** (https://help.github.com/en/articles/adding-a-new-ssh-key-to-your-github-account) page for more information on how to do this.

Online References

As noted above, a quick search will find a nearly limitless list of pages discussing git and **GitHub** (https://github.com/). It is easy to get lost in the myriad of topics they will discuss. In general it is best to have a rough understanding of git and **GitHub** (https://github.com/) before delving too deep into the details. We will attempt to cover the main ideas in class. Even so, it is good to have some references. Here are a few that I have found to get you started.

- **A basic introduction from Real Python** (https://realpython.com/python-git-github-intro/) (which has many other good articles). Git is not just for working with Python, it can be used for "everything". It is more a coincidence that this one happens to be targeted towards using Python.
- A lengthier **book-like article from Launch School** (https://launchschool.com/books/git).
- The **Pro Git book** (https://git-scm.com/book/en/v2), which is freely available online and in ebook formats.
- The **main GitHub documentation site** (https://help.github.com/en).
- The **main GitHub guides site** (https://guides.github.com).
- A **git visualization tool** (https://github.com/git-school/visualizing-git). This is useful for more a more advanced understanding of the structure and manipulations of a repository. The provided link takes you to the GitHub page for the tool which shows you how to use it. The tool shows visually how various operations affect the state of a repository and your location in it. It contains a link to a site that allows you to play around yourself.