Introduction to Git –- Fall 2024

Lecture 0: Setup

Slides: https://hackmd.io/@git-fall-2024/L0-setup

Installing and setting up Git

We will use Git from the command line in this course. This is normally how you will use it on the various HPC centers, and this way it will also be easier to understand what is going on while you are learning to use Git. On Windows, this means you will be using Git Bash.

Graphical tools exists for Git, see below list for a few. All entries on the list are free and unless otherwise mentioned, available for Windows, macOS, and Linux:

- git-scm (https://git-scm.com) comes with a basic GUI
- Git Kraken (https://www.gitkraken.com/)
- Github Desktop (https://desktop.github.com/) Windows and macOS only
- Sourcetree (https://www.sourcetreeapp.com/) Windows and macOS only
- TortoiseGit (https://tortoisegit.org/) Windows only

Installing and setting up Git - continued

- Install Git, if you have not already
- Create a repository with `git init`
- Set your name and email with `git config` (local, global, system). More info in a moment.
- Test by creating a file
- Then adding the file with `git add`
- Then commiting the file with `git commit`
- Check with `git log` that all looks well.

When this is done, you will clone the course materials.

Installing and setting up Git - continued

NOTE: if you have a problem getting this to work on your own computer, and you have an account at Tetralith or any other HPC system, then you can use that instead.

We have some documentation for you for Tetralith: https://hackmd.io/@git-fall-2024/tetralith.

As mentioned, you may also use any other HPC system you have an account at, of course. The above documentation would need only minor adjustements.

Git install - Windows

- Go to the Git-scm website (https://git-scm.com/downloads) and click "Windows" to download the Windows version. It should automatically start download of the .exe file.
- The downloaded file can be installed by double-clicking and choosing "Run".
- Click "Yes" to let it be installed and then "Next" to accept the GNU GPL.
- The default options you are presented with should work, and we recommend using those.
- You will be using Git Bash for this course
- NOTE: when it comes to choosing the default editor, we recommend using either notepad or vim, unless you have a preferred editor. See the section on "Configure git" as well as the section on editors at the end of this document for some help.

Git install - macOS

If you have installed XCode (or its Command Line Tools), Git may already be installed. To find out, open a terminal and enter `git --version`.

If Git is not installed, you have several installation options. Apple maintains their own fork of Git, but it is usually a few versions behind, so we do not recommend installing that.

- SourceForge: https://sourceforge.net/projects/git-osx-installer/files/
- Git-scm.com: https://git-scm.com/downloads
- If you have Homebrew: `brew install git`

Git install - Linux

Git is usually already installed on Linux, but if not, this is how you install it.

Installing Git on Linux depends on which distro you are running.

- `sudo apt-get install git` (Ubuntu, Debian)
- `sudo dnf install git` (RHEL, CentOS)
- https://git-scm.com/download/linux (other)

Git install - primary branch

- The primary branch will probably be named "master" when installing Git.
- You can choose if you want to instead name it "main" (which is what GitHub uses as default).
- Regardless of which you pick, stick to one to avoid problems when pushing a repo
- You can change the naming of the primary branch in GitHub for a repo
    - Go to repo
    - Pick "Settings" -> "General"
    - Change the name in "Default branch"
- Instructions how to rename the primary branch in a repo from "master" to "main" on the command line: https://gist.github.com/danieldogeanu/739f88ea5312aaa23180e162e3ae89ab

Configure git (all OS)

First check that you have git installed (in a terminal or in Git Bash):

```
$ git --version
```

Now configure git with

- `git config (local, global, system)`

You should at least set your global name and email (just once):

```
$ git config --global user.name "Your Name"
$ git config --global user.email "yourname@example.com"
```

Configure git (all OS) - continued

Setting the editor (once) is also a good idea:

```
$ git config --global core.editor <editor>
```

Choices for editor could be (on Linux, though can be installed together with Git for other OS):

- nano
- vim
- emacs

You should be able to use notepad on Windows by setting:

```
git config --global core.editor "<path-to>/notepad++.exe"
```

Another option could be to install VS Code and do this config instead:

```
git config --global core.editor "code --wait"
```

GitHub has some documentation on choosing and setting editors for various OS:
https://docs.github.com/en/get-started/getting-started-with-git/associating-text-editors-with-git

See more about configuring and using editors with Git at the end of this document.

Test your Git installation

Create an example folder and change to that, then create a file test.txt. On Linux you would do this:

```
$ mkdir <mydir>
$ cd <mydir>
$ touch test.txt
```

Now initialize a repository and *stage* the new file:

```
$ git init
Initialized empty Git repository in /home/bbrydsoe/test-git/.git/
$ git add test.txt
```

Now *commit* the change. The editor which you configured earlier should open. Add an example *commit message*:

```
$ git commit test.txt
[master (root-commit) ff8b6f6] Test of git
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 test.txt
```

Test your Git installation - continued

Now let us look at the log:

```
$ git log
commit ff8b6f699d98c72d5cffc64d65a1c618b976b45a (HEAD -> master)
Author: Birgitte Brydsö <bbrydsoe@cs.umu.se>
Date:   Thu Sep 17 13:53:59 2020 +0200

    Test of git
```

When you do `git log`, you should see something like the above, but with name, email, date, and commit message different. If that is the case, your Git should be configured correctly.

Download the course materials

For the individual hands-on part of the course, we have created some course materials which you will download from the course GitHub: https://github.com/hpc2n/course-intro-git (normally you click the green "Code" button to get the link to clone or download)

- Please go to the terminal window where you have downloaded and set up Git.
- Change the directory to wherever you wish to have the course material.
- *Do one of*:
    1. `git clone https://github.com/hpc2n/course-intro-git.git`
    2. Download the zipfile (directly with `wget https://github.com/hpc2n/course-intro-git/archive/refs/heads/main.zip` or elsewhere then transfer) and unzip. You can also get the link from the course GitHub: https://github.com/hpc2n/course-intro-git.

Web based Git repositories

There are several web based Git repositories. Some of the more popular ones are:

- GitHub (https://github.com/)
- GitLab (https://www.gitlab.com)
- Bitbucket (https://bitbucket.org)
- SourceForge (https://sourceforge.net/)

We are going to use GitHub for the part of the hands-on where you will be working together in groups.

Please go to

- https://github.com/

and sign up for an account. You will need to setup 2FA also.

Create a new SSH key for GitHub - Linux and macOS

This part will be done before the section "Working with remotes" on day 4, but you can create and add your SSH key to GitHub now if you want to.

1. Open a terminal. In the command below, "GitHub" is a label added to the key for clarity. You can add any you want:
   a. Do this

```
$ ssh-keygen -t ed25519 -C "GitHub"
```

   b. If you have an older system, this may work better

```
$ ssh-keygen -t rsa -b 4096 -C "GitHub"
```

2. You will be asked for a file to save the key. Unless you have an existing SSH key, accept the default.
3. Enter a passphrase and repeat it.

Create a new SSH key for GitHub - Linux and macOS

4. Add the key to the ssh-agent. Here we assume the default name for the new systems - change to what your key was called (`.ssh/id_rsa` for the legacy system):

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

$ ssh-add ~/.ssh/id_ed25519
```

5. Switch to the `.ssh` folder, open the file `id_ed25519.pub` with some editor and copy it (`id_rsa` for legacy systems). Do NOT add any newlines or whitespace!

Create a new SSH key for GitHub - Windows

This part will be done during the exercises on day 5, but you can create and add your SSH key to GitHub now if you want to.

1. Open Git Bash. In the command below, "GitHub" is a label added to the key for clarity. You can add any you want:
   a. Do this

```
$ ssh-keygen -t ed25519 -C "GitHub"
```

   b. If you have an older system, this may work better

```
$ ssh-keygen -t rsa -b 4096 -C "GitHub"
```

2. You will be asked for a file to save the key. Unless you have an existing SSH key, accept the default.
3. Enter a passphrase and repeat it.

Create a new SSH key for GitHub - Windows

4. Add the key to the ssh-agent. Here we assume the default name for the new systems - change to what your key was called (`.ssh/id_rsa` for the legacy system):

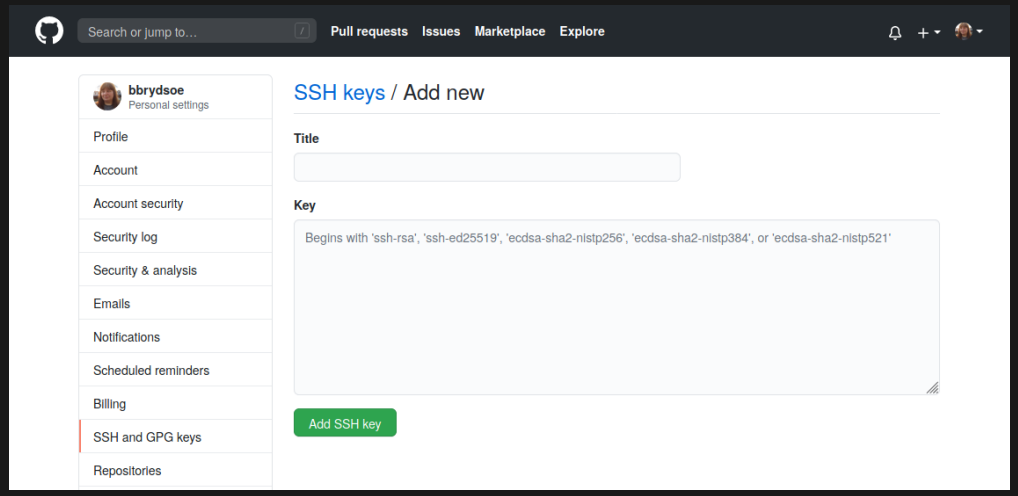
```
$ eval "$(ssh-agent -s)"

$ ssh-add ~/.ssh/id_ed25519
```

5. Switch to the `.ssh` folder, with some editor, open the file `id_ed25519.pub` (or `id_rsa.pub` for the legacy systems) and copy it. Do NOT add any newlines or whitespace!

Adding the SSH key to GitHub

1. On GitHub, click your avatar in the top right corner and pick "Settings".
2. Choose "SSH and GPG keys"
3. Click the green button labeled "New SSH key"
4. Add a descriptive label for the key in the "Title" field. In the key field you paste the content of the key (~/.ssh/id_rsa.pub)



5. Click "Add SSH key"
6. Confirm your GitHub password if you are prompted for it.

Testing the SSH keys

1. Open a terminal / the Git bash
2. `$ ssh -T git@github.com`
3. It will look similar to this:

```
$ ssh -T git@github.com
The authenticity of host 'github.com (140.82.121.3)' can't be established.
ECDSA key fingerprint is SHA256:p2QAMXNIC1TJYWeIOttrVc98/R1BUFWu3/LiyKgUfQM.
ECDSA key fingerprint is MD5:7b:99:81:1e:4c:91:a5:0d:5a:2e:2e:80:13:3f:24:ca.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'github.com,140.82.121.3' (ECDSA) to the list of known hosts.
Hi bbrydsoe! You've successfully authenticated, but GitHub does not provide shell access.
```

4. Verify that the resulting message contains your username.

More on editors, Linux

Vim

- You may need to install it first. (`sudo apt-get install vim`)
- Start with `vim <filename>` to open a file for editing. The file will be created if it does not exist before.
- Type `i` to enter 'insert' mode to be able to write in the editor.
- Use `ESC` to go to 'command' mode and then `:wq` to save and exit the editor.
- If you decide you do not want to save your changes, instead type `:q!` while in 'command mode'.
- When you are in 'command' mode, typing `dd` will delete the whole line your cursor is on.

Nano

- You may need to install it first. (`sudo apt-get install nano`)
- Start with `nano <filename>` to open a file for editing. The file will be created if it does not exist before.
- Ctrl-x will exit the editor, asking first if you want to save the file. If you started with just `nano` and did not give a filename, it will ask you for a name.

More on editors, Windows

- Using notepad: if you are using a newer version of Git, then you should be able to choose to install/use notepad during the Git install.
    - `git config --global core.editor notepad`
- Otherwise, you need to give the full path to notepad on your system
    - `git config --global core.editor "<path-to>\notepad++.exe"`
    - Example:
        - `git config --global core-editor "C:\Program Files (x86)\Notepad++\notepad++.exe"`
- Using vim: this is easy as it can be installed during the Git install and then setting it with `git config --global core.editor vim`

GitHub has a page for setting some editors for various OS'es: https://docs.github.com/en/get-started/getting-started-with-git/associating-text-editors-with-git

More on editors, various OS

GitHub has a page for setting some editors for various OS'es:

https://docs.github.com/en/get-started/getting-started-with-git/associating-text-editors-with-git

GitHub CLI

GitHub also has a command line interface that you can use if you want to.

It is available for Windows, macOS, and Linux.

You can use it if you prefer to do your workflow through a terminal, and you can call the GitHub API to script various actions as well as set a custom alias for any command.

More information and download here:

- https://cli.github.com/
- https://github.blog/2020-09-17-github-cli-1-0-is-now-available/