Using Git and GitHub to Simplify Development on the DCC

Summary

Last week we learned how to log in to the DCC to take advantage of the shared computing resources available at Duke. While access to large amounts to processing power and memory will be advantageous in your case studies moving forward, development on the cluster can be awkward and uncomfortable for those unfamiliar with a Linux/Unix command line. In part of the lab, we will introduce the command line interface of "git" as a straighforward way to link development on the cluster with local work on your personal computer.

The Four Essential Commands You Will Use All the Time

1. `git pull`: Update your local branch of a repository with any commits that have been pushed to the remote branch of your repository

   ---

   When you have made important changes to your local branch of your repository and want to update a remote "master" branch of your repository, you will need to *stage* **only** the files with important changes, save a snapshot of those staged changes, and then update the remote branch to include your latest snapshot. Those three steps are accomplished with `git add`, `git commit`, and `git push` respectively

2. `git add <filename1> <filename2> ...`
3. `git commit -m "<description of commit>"`
4. `git push`

Other git Commands you will sometimes use

- `git init`: Create a git repository in your current directory on your current computer
- `git clone <url to a git repository>`: Make a local branch of some repository such that the repository you are cloning from will be the master branch.
- `git status` shows color coded list of files in your local directory that are *tracked* in your repository or *untracked*.

        - If any tracked files are unchanged, they are not listed. Tracked files that have been changed but not staged are listed as
        - Untracked files are any files in your directory that are not members of your repository - i.e. if someone else clones your repository, untracked files will not come with it, and if you change an untracked file, git does not care.
- `git log`
- `git rm <filename>` works like `git add` but stages a file to be **removed** from your repository

Git versus GitHub

There is an important difference! Git is a version control system, not a website. It is a piece of software that helps you better track changes in your code. Think of it like a scrapbooking system for a CS or data science project. Git has nothing inherently to do with the internet.

GitHub is one (of many) websites that will host a copy of your scrapbook remotely. This is useful because you and your friends, who all have internet access, can work together on the scrapbook collaboratively. The scrapbook (git repository) typically becomes especially important for group CS or DS projects.

Lab Exercise

Let's go through the process together of creating a git repository and seeing how it can be used to simplify development for the DCC.

1. Create a Repository

- On GitHub:
    - Find the "new" button in the repositories section and click it.
    - On the next page, give your repository a name and click "create repository."
    - Set up the repository by
        - running the sequence of commands under "create a new repository"
        - clicking "creating a file" in the quick setup, making a "README.md" file, commiting the new file (button at bottom of screen) and then cloning the repository ("download code" button, but the download symbol instead of the word)
        - Create a new R Project (specific to R Studio) and specify

`git` as your version control system and provide the link to your remote repository.

- Locally: On the command line, navigate to the directory which you would like to track with a git repository (may already have files in it) and run `git init`. Then make a commit, allocate space for a repository on GitHub (first and second steps above) and follow the instructions to "push an existing repository from the command line"

2. Add files/write code locally

- Do everything as usual. Add a bash script with slurm options and `module load R`.

3. Log on to the DCC and clone the repository in your personal folder for STA440

- clone the repository with `git clone <github url>`
- note: you will still need to install any packages you need on the DCC via an interactive session of R.