Getting Started with Git and GitHub

In this discussion we will set up our system to use Git and GitHub. This will allow us to track files and easily make changes to our projects. This will also allow us to share, deploy, and collaborate on our code.

Introduction to our command syntax:

- Commands will be written in bold, like this **echo "Welcome to CS391"**
- Bold terms/characters should be interpreted literally
- Italicized and purple terms/characters indicate you will likely need to replace that portion of the command with something else. For example **mkdir *directory-name***
    - This command indicates that instead of literally typing out "directory-name" you should replace it with something else, such as what you would like to name a new directory

Note:

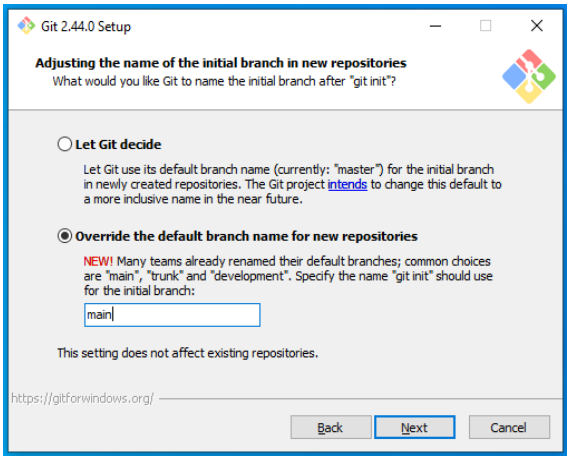
If you have already setup Git on your computer and/or created a GitHub account, you may skip to the appropriate task.

Task 1: Create a GitHub account

1. Go to https://github.com/ and sign up for an account
    a. Feel free to use your BU email or your personal email

Task 2: Download and set up Git

1. Go to this website and select your operating system: https://git-scm.com/downloads
2. Download and run the appropriate installer. If asked if you would like to add to PATH, do so. Additionally, override the default branch name to main, as shown below. Other than that, follow the default options.
    a. This installation will include Git Bash. If you are using Windows, we recommend using Git Bash as your shell.



3. Set your name by using the following command:
   git config --global user.name "*first-name last-name*"
4. Set your email by using the following command:
   git config --global user.email "*email@bu.edu*"

Task 3: Connect your computer to GitHub

1. In your terminal, run the following command: **ssh-keygen**
2. You do not need to edit the output file or enter a passphrase (just hit the enter key a couple times)
3. Your public key will be located in the file ending with **.pub** within the **~/.ssh** directory. Navigate to this directory (**~/.ssh**) and use the ls command to find the name of the file.
4. Use the cat command to output the contents of this file. You can pipe the output of this command into clip if you're on Windows or pbcopy if you're on macOS to copy the public key to your clipboard.
   a. Windows: **cat *filename*.pub | clip**
   b. macOS: **cat *filename*.pub | pbcopy**
5. Go to https://github.com, sign in, and click your avatar/icon in the top right corner. Then go to settings
6. Select **SSH and GPG keys** from the menu on the left and click **New SSH key** at the top
7. Enter an appropriate name for your computer and paste the SSH key from step 4
8. Click **Add SSH key**

Task 4: Create a local repository

1. In your terminal navigate to the location you would like to create a repository. (e.g. **~/projects**, **~/cs391/mini-projects**, **~/cs391/labs**)
2. Create a new directory by using the command **mkdir *directory-name***
   a. We will be pushing/uploading this directory and its contents to GitHub
3. Enter this directory with the command **cd *directory-name***
4. Create a text file with some content using the command **echo "*file contents*" > *filename***
   a. Here we use the echo command to send the specified string to standard out, which is then sent to the specified file

Task 5: Initialize your local Git repository

1. In the same location as the previous task, use the command **git init** to initialize the directory as a Git repository
   a. Some popular frameworks do this for your, so this may not always be a necessary step

2. Use the command **git status** to see the current status of the files within your directory. Right now, all files should be untracked
3. Use the command **git add *filename*** to stage your changes. This tells Git about any changes you've made to this file since its last commit. When using this command, you can use more than one filename. If the file you wish to add is not at the root level of this directory, include its relative path.
    a. You can use the command **git add .** to add all files with untracked changes in the current directory. Be wary of this command as you don't want to accidentally include sensitive files.
    b. If you wish to unstage/unadd files, use the command **git reset**. This command will unstage your files and Git will no longer have any information about those files' changes since their last commit. This command will not affect the contents of any files.
4. Once you have added all relevant files, use the command **git commit -m "*your commit message*"** to commit your changes. This confirms your staged changes with Git. Be sure to include a helpful/informative commit message when using this command.
    a. If you do not include the **-m** flag, you will be prompted to write a commit message in your default text editor. This may be vim. If it is, have fun exiting :)

Task 6: Push your repository to GitHub

1. Create a new GitHub repository by visiting https://github.com and clicking **New** on the left next to **Top Repositories**
2. Give your repository the same name as the directory you're working in and click **Create repository**
3. Follow the directions under **or push an existing repository from the command line** in your terminal
4. Your files should now be on GitHub! Refresh your GitHub tab and you should see your files.