2. Reproducible Science with Git and GitHub

Z620: Quantitative Biodiversity, Indiana University

OVERVIEW

Eventually, you may choose to set up your personal computer with Git. But to help ensure that things go smoothly, you will have access to a dedicated Mac desktop computer that is located in Lindley Hall, which you can access using a Remote Desktop connection. This computer has all the software required for the course. You will use Git and GitHub on that computer to access and record changes to QB **worksheets** that contain in-class **exercises** and **assignments** that will eventually be submitted via Github. The instructions below will show you how to configure github on a computer. After that, you will learn how to **fork** and **clone** a GitHub repository. Finally, you will learn how to **commit** and **push** changes so that you can access them later on your personal computer using **fetch** and **merge** commands.

---

1) Remote Desktop

Owing to the pandemic, we are unlikely to meet face-to-face during our regularly scheduled class time. To avoid any initial complications arising from software incompatibility, at least initially, you will all remotely log into one of the cloned Mac desktop computer in Lindley Hall that has all of the correct versions of software needed for QB. Please follow these steps:

1. Visit https://uits.iu.edu/iuanyware and click on "Apps" along the top banner
2. Select "Remote Desktop Connection" from the list of software
3. Enter the IP address that was assigned to you in the text box following "Computer" and hit "connect"
4. Supply your IU username and password
5. You should now see the desktop of the computer in Lindley Hall

2) Git Test

Before we get started with Git, we first need to test our current installation to make sure there aren't any issues. The easiest way to do this is to determine what Git version is currently installed. We will use **Terminal**

On the lab computers, you can find terminal in one of two ways. First, you can search for terminal with spotlight [Cmd+Space]. Second, you can navigate to **Utilities Folder** in Finder. You may want to drag Terminal to the **Dock** at the bottom of your screen, as you will frequently be using this program.

1. Find Terminal and open a new window
2. After you've opened a new Terminal window (Shell > New Window), type the following commands, which will print your current working directory (pwd), list folders in your working directory (ls), and supply the version of installed git.

```
pwd
ls
git --version
```

3) Git User Configuration

1. **Organization:** We recommend that you create a folder in your user directory called '*GitHub*' to make this and future assignments easier to manage. In the following code, you will move to your user directory using the `cd` (change directory) function. The *slashdot* (./) specifies your current directory (../ would specify the directory above your current directory) and you will make a new directory with `mkdir` function.

```
cd ~
mkdir ./GitHub
cd ./GitHub
pwd
```

2. **User Configuration:** You will need to configure your local Git installation. You will do this by entering your name and email. You will also set two parameters: push.default and credential.helper

```
git config --global user.name "John Doe"
git config --global user.email johndoe@example.com
git config --global push.default simple
git config --global credential.helper store
git config --global core.editor "nano"
```

3. The last thing you need to do is configure how Git handles line endings. Line endings are invisible characters that your operating system places at the end of each line in a document. On Unix machines (e.g., Mac), this is the linefeed character (LF). On Windows machines, this is the carriage-return (CR) and linefeed (LF) characters. This difference in line endings between Mac and Windows causes incompatibilities between the two systems. However, Git is equipped to handle the differences by silently converting line endings when repos are pushed to remote servers. We *strongly* urge that you configure this behavior in order to prevent any future issues when collaborating across computer platforms.

   On Mac

```
git config --global core.autocrlf input
```

   On Windows

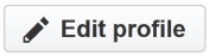
```
git config --global core.autocrlf true
```

You are now ready to *Git !!!*

4) Create User

1. Navigate to https://github.com
2. Sign in with your Username and Passphrase
3. On the top right of your screen, click on your Username

4. Click on the Edit Profile Icon to edit your profile



5) Fork and Clone a Repo

1. Navigate to the QB student organization: https://github.com/QBstudents
2. Click on the repository (repo) with your name
3. Fork your repo by clicking on the Fork icon in the top right of your screen

   You should now see the repo on your GitHub page.
4. Clone the repo onto your local machine using the command line (Terminal). Replace "username" with your GitHub username and "repo" with the name of the repo in the QBstudent organization (e.g., QB2021_Fishman)

```
cd ~/GitHub
git clone https://github.com/username/repo
cd ./repo
git status
```

The repo should have downloaded onto your local computer and the status should stay "all up to date".

5. Check and update remote repo location The following commands will set the location of your **upstream remote repository** on the QB Course GitHub site. (In the example below, you will need to replace "repo" with the name of your QB Repository)

```
git remote -v
git remote add upstream https://github.com/QBstudents/repo
git remote -v
```

   You can copy and paste the URL for your upstream repo from the GitHub website.
6. Open and edit the README.md file contained in your student repository:

   This file is a Markdown file. Markdown is markup language for writing and editing text that can be easily converted to other formats (e.g. HTML, PDF, Word). During this semester, we will edit Markdown files using RStudio. On the lab computers, you can find RStudio in the **Analysis & Modeling Folder**. From RStudio, navigate to and open your README.md file in the root of your student repo. After the text "Quantitative Biodiveristy course, Spring 2019, Indiana University", enter your name and email address. Write a short biography (~1 paragraph) and share something about yourself that we might not otherwise know. List three to five course expectations. Hint: View the Markdown guide to learn about formatting and making ordered lists https://guides.github.com/features/mastering-markdown/. When you are done, save the close the document.

6) Commit and Push

1. You just changed the README.md file. To record this, we are going to create a **commit** or "revision" with the following code. The `status` function identifies changes you made, the `add` command adds a file to the "staging area", and the `commit` command sends the changes to your repository along with a message (-m).

```
git status
git add ./README.md
git commit -m "Updated README.md with student information"
```

2. Now we will **push** the changes to GitHub. But before we do this, we always want to check for any changes that others have made; otherwise we can create **conflicts** that need to be fixed. We check for these changes using the git commands **fetch** and **merge** as follows:

```
git fetch upstream
git merge upstream/main
git push origin
git status
```

You should now see the repo, including your recent changes, on your GitHub webpage.

7) Pull Request

After pushing your commit, you need to follow up by making a **Pull Request**.

1. Navigate to your GitHub webpage to make sure that the file (e.g., README.md) was uploaded correctly. If so, submit a pull request to submit your file to the course instructors.

   The course instructors can now merge and see your changes.

8) Getting Up to Date

2. To get new worksheets, you will *pull* your upstream repository to your local computer using the **fetch** and **merge**. This will allow any updates your instructors have made to be merged with your local documents. In addition to pulling your upstream repo, you always want to push any updates to your origin.

```
git status
git fetch upstream
git merge upstream/main
git push origin
git status
```

During this course, you will receive **worksheets** and submit all **assignments** using these methods. In addition, you will use Git and GitHub to contribute to assignments in class and on your personal computers.

9) Basic Git and GitHub glossary:

The terms below will be used throughout this class. Some are commands that you will type into the terminal window. All are defined with respect to how we will be using Git and GitHub in this class.

Term	Meaning
repository	The collection of files and folders that compose a project.
upstream	Refers to the central repository, e.g., the version managed by your instructors.
origin	Refers to your on-line version of the class repository.
fork	Create a version of the class repository in your IU-GitHub account.
clone	Create a copy of 'origin' on your computer. This is the version you will edit.
fetch	Download changes that have been made to an online repository.
merge	Merge changes with your local version.
pull	'fetch' + 'merge'. 'pull' executes 'fetch' and 'merge' but give less freedom of control.
staging area	A file that Git uses to store information about your changes.
add	Having edited, removed, or added files, this command will add your changes to the staging area.
commit	Having added your changes, this command will save a picture of what your files looked like at that moment.
push	Having committed your changes, this command will update 'origin'.
pull request	Having updated 'origin', request that these changes be pulled into 'upstream'