# Git-Exercise

Environmental Data Analytics

John Fay and Luana Lima

Fall 2022

With our forked class repository now cloned to our local machine, let's give Git and GitHub a test drive. Unless otherwise noted, all the actions listed below occur in RStudio.

## Creating, adding, and tracking changes to a new file

**1. Pull any changes made to your remote GitHub repository to your local machine** It's unlikely you've actually made any changes yet to your remote repository (i.e. your fork of the class repository), but you need to build in the habit of pulling any remote changes before making any local changes to avoid creating divergent branches.

1. Click on the `Git` tab in the upper right quadrant of RStudio. This opens your Git command center.
2. Click the `Pull` button. This pulls any new changes on your remote GitHub repository to your local one. Likely, you'll just see an "Already Up to date" message appear that you can dismiss. You are all set!

**2. Pull any changes made to the upstream (i.e. the class) repository** There's no button to pull changes made to the class repository into your local one, so we'll have to do this by command. Type this in your RStudio **terminal**

```
$ git pull upstream main
```

If there are any changes, they'll be pulled into your local workspace. If not, you'll get a message "Already up to date."

> At this point, all our workspace are synchronized, we are ready to make local changes. Next, we'll simply create a new file and see how we work it into the Git system.

**3. Create a new file**

1. Create a new text file in RStudio (`File`>`New File`>`Text File`)
2. Save it as `GitDemo.txt` in your `Assignments` folder.

**4. Tell Git to track the file**

1. In the *Git command center* , you should see your new file with two yellow question marks next to it. This is Git telling you it sees the file you created, but it's not doing anything with it.
2. **Click the `Staged` button** next to the file. This tells Git that we want to start tracking changes this file.
3. **Click the Commit button**. This open a window to commit the change (add our empty file) to our local repository.
4. **Add a Commit message**: This is a label we associate with our change, to identify it later, if we want. It should be brief and specific, like 'Add new GitDemo file'.
5. Click **Commit**. This saves the change we just made to our Git repository. The change is registered in Git's history. You can close all windows other than RStudio.
6. Still, in the Git command center, click the clock :clock3: icon; this **opens the history** of changes to your Git repository. At the top you should see the commit you just made.

   Git has now added and tracked our first local change to the repository. The record of the change has only been recorded locally; we could push that one change up to our forked GitHub repo now or as a batch of changes later. Let's go ahead and push the change now.

**5. Push the change to our GitHub repository.**

1. Click the `Push` icon in the Git command center.
2. In a web browser, open up your GitHub repository.
3. Open up the Assignments folder: you should see your `GitDemo.txt` file. You should also see the message associated with committing that change.
4. You may also see a message "This branch is 1 commit ahead of ENV872:main". That's OK! It means that you have a change in your forked repository that is not in the class repository. Makes sense since that file you created is NOT in the class repository.
   *However, if you ever see a a message "This branch is X commits behind ENV872:main", that means there are changes to the class repository you'll need to pull to your local repository and then push to your forked one.*

---

## Modifying a tracked file

Now we'll look at the workflow for updating a file. If we were starting a new, you'd pull from your local, then from the upstream before the following. Since we are continuing, we'll start from here:

**1. Edit your `GitDemo.txt` file**

```
1. Open the file in RStudio or any text editor.
1. Add a few lines of text to your file and save it.
```

**2. Commit the changes in Git**

1. View the file in the Git command center. You should see a blue M next to it. This indicates the file has been **m**odified.
2. Check the `Staged` box. You see the blue M shifts to the left. This means that if you committed changes, this modification would be included in that commit.

3. Click the `Commit` button. Note that the window that appears show the edits you made to the file.
4. Add a message and commit your edit.
   :question:Would this file appear in your GitHub web site now?
5. Push the commit to your GitHub web site.

---

That's enough to get you started. Git can do a lot more, but we'll take it one step at a time. If you stick to these basic, and don't veer off the workflow presented here, Git will be a good friend. If problems do arise, don't panic, but please seek help of an instructor before you go online to seek a solution. More than likely, it's a simple issue we can resolve with you. But more often than not, digging into other "solutions" often gets things more tangled.