**ANA 500 – Python for Data Science**

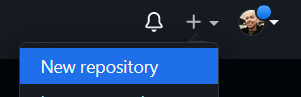
# **Git / Jupyter nb Workflow- Cheat Sheet**

April 2022 term

5/6/2022 10:04

Example: I am creating a new Jupyter notebook, and I want to use version control to track it.

Workflow:

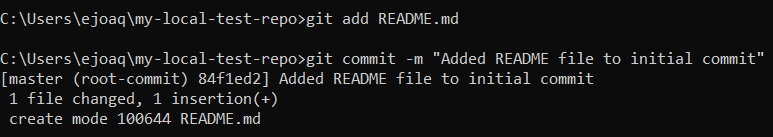
* 1. Create a New repository in github.com:
     1. 

**Do not** initialize the repository with a README, .gitignore or License file. This empty repository will await your code.

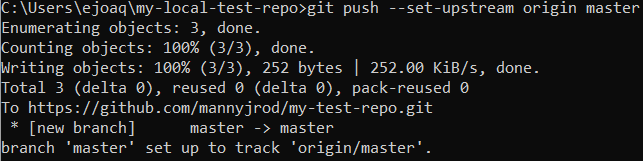
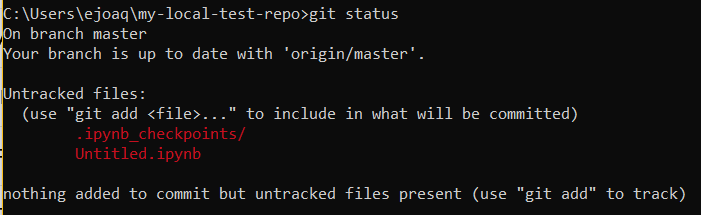
From <[*https://docs.github.com/en/get-started/using-git/about-git*](https://docs.github.com/en/get-started/using-git/about-git)>

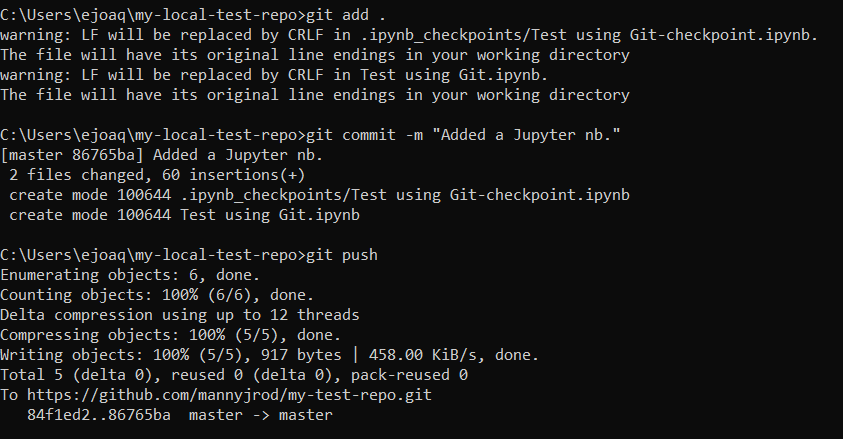
* 1. In a terminal, create a new directory, and **initialize** it with git specific functions
     1. 
     2. 'init' creates an empty git repository or reinitializes an existing one.
     3. Note: This only creates a directory in your local machine (a folder is created); 'git init' initializes a brand new Git repository and begins tracking an existing directory. It adds a **hidden subfolder (referred to as an 'index' by git)**  within the existing directory that houses the internal data structure required for version control.

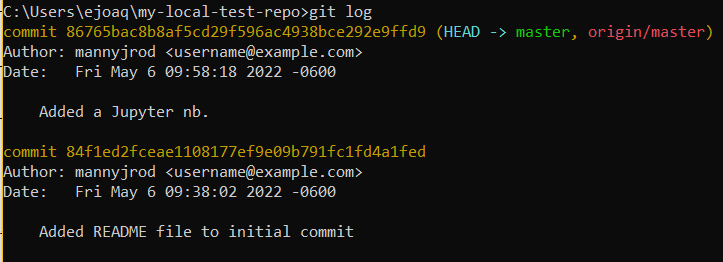
From <[*https://docs.github.com/en/get-started/using-git/about-git*](https://docs.github.com/en/get-started/using-git/about-git)>

* 1. Change to 'my-local-test-repo' directory
     1. 
  2. Create a README file and include text with the '**echo**' function:
     1. 
  3. Git isn't aware of the file, it must be **staged** with 'add', then taken a "**snapshot**" of with 'commit' and a message is included to document what was done in this moment in time:
     1. 
  4. Next, provide the path for the repository you created on github
     1. 'git remote add origin <https://github.com/YOUR-USERNAME/YOUR-REPOSITORY-NAME.git'>

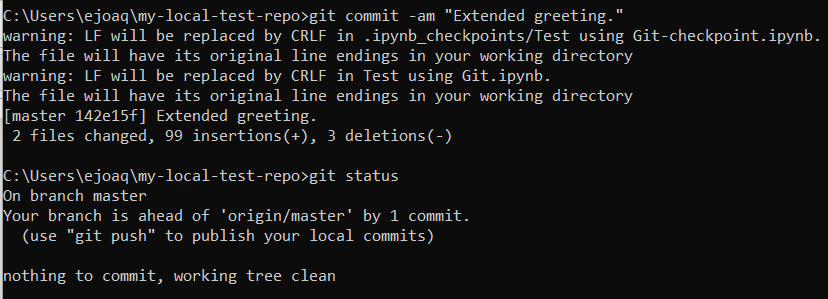
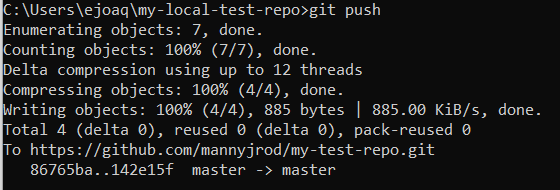
From <[*https://docs.github.com/en/get-started/using-git/about-git*](https://docs.github.com/en/get-started/using-git/about-git)>

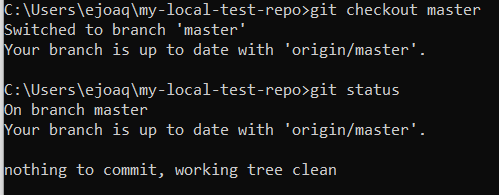
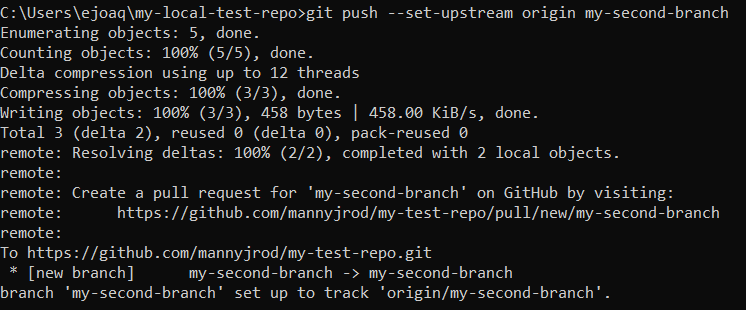
* + 1. 
    2. Note: Once this path is provided, this Git repository will now have this path stored for any follow-on add/commit/push.
    3. Note: '**origin**' is an *alias* on my machine for a remote repository -- this is common naming convention, and the word origin is not required (see: <https://stackoverflow.com/questions/9529497/what-is-origin-in-git>).
  1. Then, push the files to github and set the upstream (remote) branch:
     1. 
  2. Launch Jupyter:
     1. 
     2. Note: This will 'lock' this terminal, so you'll have to open a new terminal and change into the above directory to continue version tracking. It's then good practice to use '**git status'** to show the working tree status of the project (which branch I'm on, and which changes need to be committed).
        1. 
  3. Start a new Jupyter notebook using the Python 3 kernal; a **kernal** is the server that enables Python to run cells within the notebook.
     1. Quick script:
     2. Graphical user interface, text, application, email

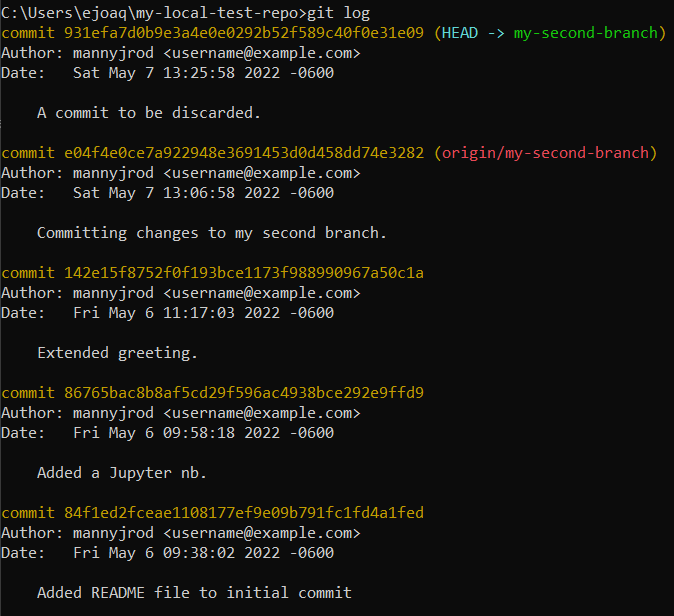
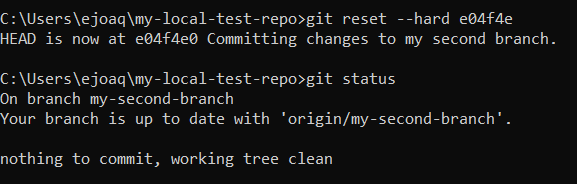
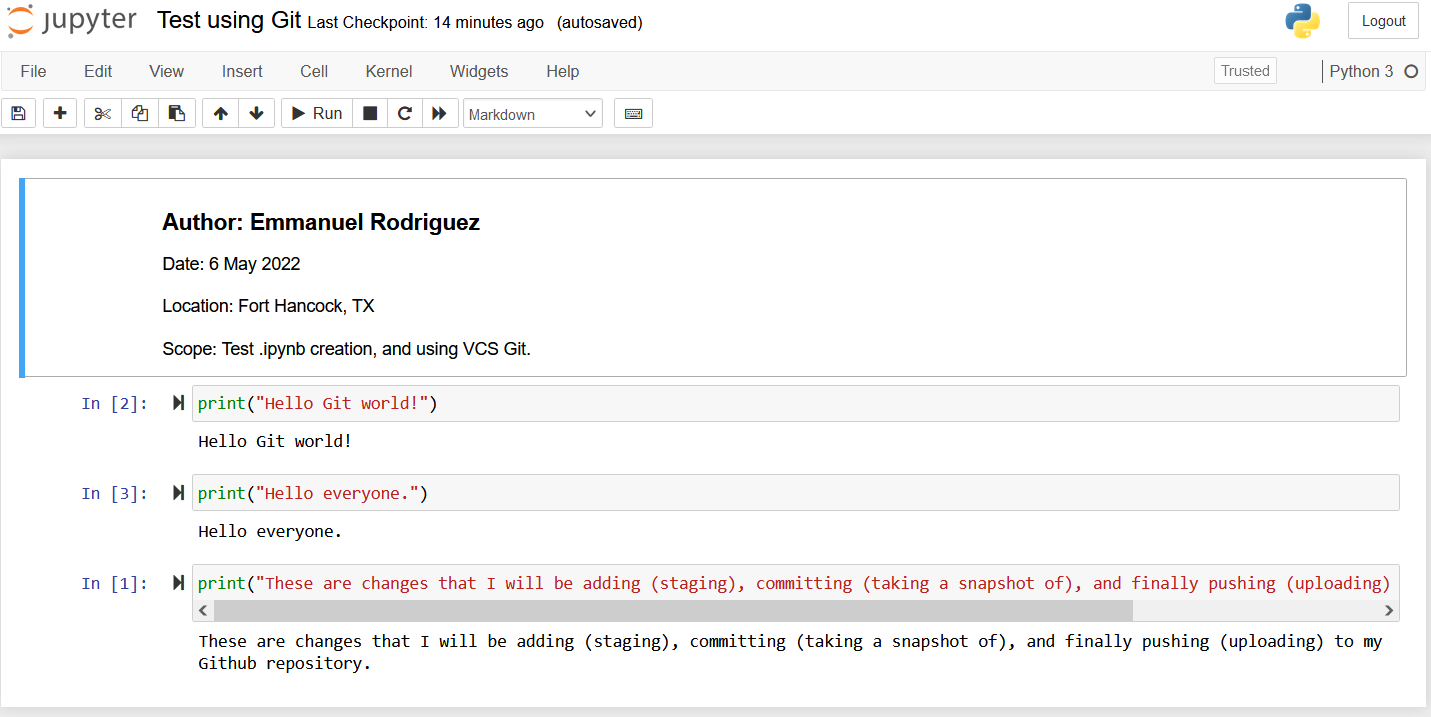
        Description automatically generated
  4. From the command line, use '**git add .**' to add *all* files within a project that aren't already being tracked to the repository, then '**commit'** with message, then '**push'** to github (remember that the remote path has already been set):
     1. 
     2. Note the above 'warning' -- this has to do with Unix systems using line feeds (LF) to represent the end of line, whereas windows uses line feed and carriage returns (CRLF) (see <https://stackoverflow.com/questions/5834014/lf-will-be-replaced-by-crlf-in-git-what-is-that-and-is-it-important>) -- this will not affect the readability of the code in github, and can be ignored.
  5. You can verify the nb has been pushed to github:
     1. Graphical user interface, application

        Description automatically generated
  6. Check log with **'log':**
     1. 
  7. Additional commits:
     1. Graphical user interface, text, application, chat or text message

        Description automatically generated
     2. Check status:
        1. Text

           Description automatically generated
     3. Commit the change and re-check status:
        1. 
        2. Note the '-**am'** flag. We make a new commit, passing the -am flags when we use the command git commit. The -a flag tells Git to add all modified files in the repository to the current commit. (If you create any new files between commits, simply reissue the git add . command to include the new files in the repository.) The -m flag tells Git to record a message in the log for this commit.
     4. **Push**:
        1. 
  8. Reverting a change:
     1. The command '**git checkout**' allows you to work with any previous commit. The command '**git checkout .**' abandons any changes made since the last commit and restores the project to the last committed state.
        1. Check log, then checkout by using the first **6 characters** of a commit code:
        2. Text

           Description automatically generated
        3. When you check out a previous commit, you leave the master branch and enter what Git refers to as a **detached HEAD state**. HEAD is the current committed state of the project; you’re detached because you’ve left a named branch (master, in this case).
     2. Refresh the notebook to update.
     3. To go back to the master branch:
        1. 
  9. To continue working off the detached head, and to retain any new commits created, create a new branch with **'git switch -c <new-branch-name>':**
     1. 
     2. Next, to push the current branch, an upstream branch is needed. Set the remote as upstream.
        1. 
        2. Noting that 'origin' was set earlier as the alias for the remote repository.
        3. Github now has the new branch:
           1. Graphical user interface, application

              Description automatically generated
  10. To check out a previous commit, and discard any recent commits (say go from commit #2 back to commit #1, discarding #2 altogether:
      1. Check the log:
         1. 
      2. Then, use '**reset --hard'** followed by the first 6 characters of the commit:
         1. 
         2. The commit "A commit to be discarded" is discarded, and the head is back to the specified commit - **permanently**.
  11. You now have the notebook file and its snapshots in history in the form of commits in Github (remote repo) and in the Git index (local hidden folder).
      1. 

* 1. 