CSC-450, Github Assignment

Introduction

Github (https://github.com) is a development platform and is probably the most popular web-based source code management system. Github uses *git* for revision control and source code management. In addition, Github implements collaborative features such as bug tracking and feature requests. Continuous integration (CI), which includes automated testing, is possible using Github workflows or third party services. Github is free to use (additional features are available for an additional charge). Other source code management systems such as BitBucket (https://bitbucket.org) are available as well. For Github, both command line *git* tools and a GUI tool called Github Desktop are available. Github is also integrated into IDEs such as Visual Studio Code.

Why use Github?

- Github is a version control system:
  - you can easily revert to a previous version of code, if needed
  - you can easily see how files change from one version to another
  - branches can be used to separate major versions of code (e.g., release and development versions; see figure below)
- Github makes it easy to work with existing code and to collaborate:
  - a user can (a) *fork* (copy) code from anyone's repository, (b) clone the code (i.e. make a local copy for development), (c) make changes to the code, (d) commit (record) the changes and (e) push (save) the changes to the remote repository (i.e., the repository on github.com).
  - if appropriate, the user can submit a *pull request*, which is a request to merge his/her code with another branch or version, possibly from another user
  - pull requests can be automatically merged if there are no conflicts; otherwise conflicts are noted and must be manually resolved.
- Github makes it easy to automate workflows, such as testing and deployment, through Github actions (https://docs.github.com/en/actions)

Figure 1. Example Github branches, source: http://nvie.com/posts/a-successful-git-branching-model/

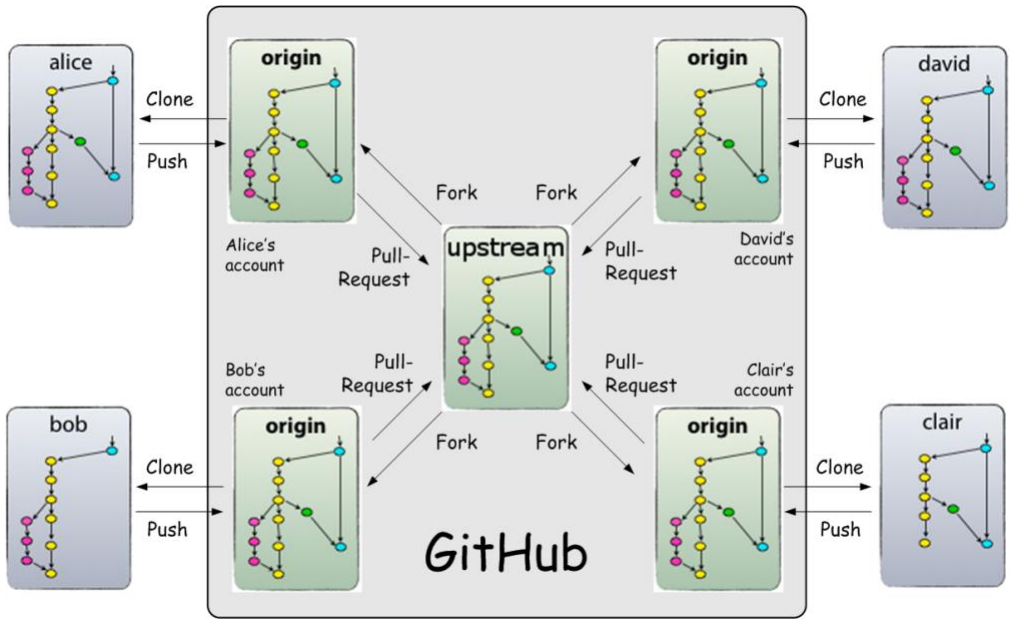


Figure 2. Collaboration on Github, source: http://www.dalescott.net/2012/09/14/using-gitflow-with-githubs-fork-pull-model/

Reproducible research

Reproducible research is fundamental to scientific research. For research involving code or data, access to source code and data is necessary to reproduce and verify published results. By making your software and data available, others will be able to reproduce (i.e., validate) your findings, and can expand on what you have done, either by asking different questions about your data and/or modifying the code you have developed. Failure to reproduce research is a large problem in fields such as cancer biology[1] and psychology[2].

As part of the project requirement, you must make your source code and/or data available. We will use Github for this purpose (other resources for hosting your code/data can be used with permission). In addition to supporting your research project, having a Github account is a great way to promote the work you have done and will look great on your resume!

For ease of use, I will go over using Github Desktop, which is available here: https://desktop.github.com/. However, I strongly encourage you to look at the command line version. You do not need to use Github Desktop in this class, but you will need to be able to fork and modify one of my repositories, and submit a Pull Request.

[1] https://www.sciencenews.org/article/cancer-biology-studies-research-replication-reproducibility
[2] https://www.nature.com/articles/nature.2015.18248

Exercise #1

1. Create a Github (https://github.com/) account if you do not already have one

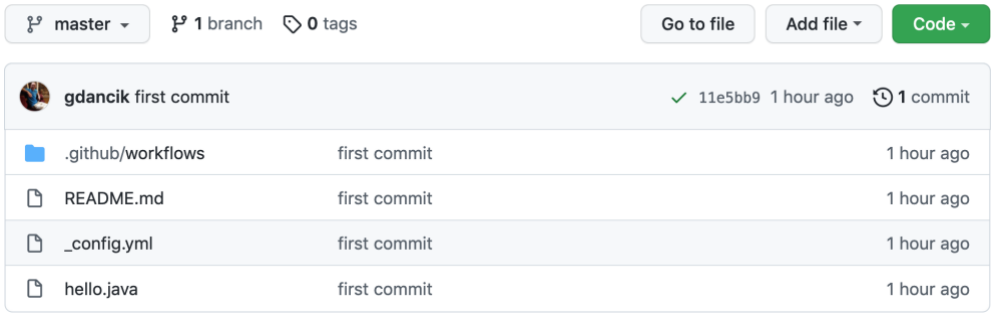
2. If you do not have Git, then download and set up Github Desktop: https://desktop.github.com/

3. Fork my *hello_world* program, available here, by clicking the *Fork* button on the top right of the page: https://github.com/gdancik/hello_world. This will create a new repository in your Github account with the name *hello_world*.

4. Open Github Desktop, select the *hello_world* repository, and click the ***Clone*** button, and specify the folder where *hello_world* should be cloned (copied) to.

5. Edit the main program (using a text editor of your choice) to change the value of *firstName* to your first name. Change the print statement so that the program outputs a greeting in the form, "Hello, I am Garrett ", where the print statement outputs the value of *firstName*. Note that you may test your code locally using an IDE such as Eclipse or you can run your code through an online compiler such as https://www.tutorialspoint.com/compile_java_online.php

6. When you are happy with your changes, *commit* them and then *push* them to your repository. On your repository's page, click on Actions and enable the *Run hello.java* workflow. This will allow you to launch a Github Action that will test your program by compiling and running it. If the program runs without any errors, and the output is formatted correctly, you will see a green checkmark by the commit, as shown below (Note: this doesn't check that the output follows all requirements, only the basic ones – make sure to follow the instructions in question #5).



7. Submit a pull request by clicking on the appropriate links from the *hello_world* repository on your Github page. Note: you can check whether you have successfully submitted a Pull Request by looking for your request at https://github.com/gdancik/hello_world/pulls. If you do not see your pull request here, you have not successfully completed the assignment!

8. For full credit on this assignment, you must (1) correctly make changes to the code, (2) run the Github Action to test the code on your repository, and (3) submit a Pull Request. You will not receive any credit if you do not submit the Pull Request.

Exercise #2

1. Create a repository for your Senior Research project by doing the following (you can skip this step if you already have one). Open Github Desktop, and from the "Let's get started page" click "Create a New Repository on Your Local Drive". Check "Initialize this repository with a README" and then Click "Create Repository".

2. Edit your README.md file so that it describes your project. Your description must include the objective of your project, but may include more details (such as background, etc). Note that the README file is a *markdown* file that is displayed in a formatted way. For example, you can use a single hashtag (e.g., *# Heading*) to display a large header. A markdown cheat sheet is available here: https://www.markdownguide.org/cheat-sheet/. To see how your README file will be displayed, you can copy and paste the text of your README file here: https://dillinger.io/

3. Commit and push your changes to update your project repository on Github.

4. Create a web page for your project repository by following the instructions from the **Publishing from a branch** section of https://docs.github.com/en/pages/getting-started-with-github-pages/configuring-a-publishing-source-for-your-github-pages-site.

5. Add a Jekyll theme by following the instructions here: https://docs.github.com/en/pages/setting-up-a-github-pages-site-with-jekyll/adding-a-theme-to-your-github-pages-site-using-jekyll. Note that you will need to create the file _config.yml. You can browse themes from https://pages.github.com/themes/. Add the theme by following the Usage instructions.

6. Your page will display the contents of your README file.

7. When completed, submit through Blackboard the **two** links: (1) a link for your project page and a link for your Github repository page. Your links will have the format below (these assume that the name of the repository is *senior_research*):

    a. your project web page (the URL will be similar to http://gdancik.github.io/senior_research/)

    b. your Github repository page (the URL will be similar to http://www.github.com/gdancik/senior_research)