GitHub tutorial

Eric Gordon

October 28th, 2019

Outline

- Introduction
  - Utility
  - Jargon
  - Basic web usage (markdown language)
- GitHub pages
- Command line usage

Git is a “version control” technique

- Created by the same person who made Linux
- Extremely useful for collaborating on code
  - Multiple people can work on components at once
  - Keep track of versions, changes, contributions

- Also useful for organization of personal custom scripts, downloading others scripts and website creation.

Jargon

- Repository – basically a folder for an individual project. Can be private or public. 1 Gb of storage per account only.
- Forking a repository – creating a copy for yourself to modify independently or save. This new creation is call a “branch”. Anyone can fork a repository.
- Pull request – submitting your branch to the administrator of a repository for approval to include any new changes you made as part of the master or main branch.
- Commit – any change made to any document, includes pull requests but also any change made to the “master branch” by the administrator.

Web usage

Lets go through how to create a pull request following:

- https://guides.github.com/activities/hello-world/

- https://www.markdownguide.org/basic-syntax/ and https://www.markdownguide.org/extended-syntax

- Add keywords if you want

You can also use GitHub to make a website for yourself or a class

Use your username to create a website easily without GitHub page elements. One per account but accounts unlimited (?).

- See: https://pages.github.com/

- Examples:
- https://biodataprog.github.io/2018_programming-intro/
- https://alexknyshov.github.io/R/page1.html

Git on the command line

Download from an existing repository

- If you have git installed **“git clone** https://github.com/erg55/Simon-lab-workshop.git**”**
- Or can get individual scripts with wget and raw file
- “**wget** https://raw.githubusercontent.com/erg55/Simon-lab-workshop/master/rename/renameoneliner.sh”

Changing your own repository

- **Create a new repository on the command line**
- echo "# name" >> README.md
  - git init
  - git add README.md
  - git commit -m "first commit"
- **…or push an existing repository from the command line**
- git remote add origin https://github.com/erg55/metagenomes.git git push -u origin master
- $ git push

git push will request your github ID and github password

https://help.github.com/en/github/authenticating-to-github/adding-a-new-ssh-key-to-your-github-account

Various commands

- Git fork

- Readme.md file is recommended. Can include LICENSE file and .gitignore files (for files you don’t want to update every time).

Choose an open source license

An open source license protects contributors and users. Businesses and savvy developers won't touch a project without this protection.

{ Which of the following best describes your situation? }

I need to work in a community.

Use the **license preferred by the community** you're contributing to or depending on. Your project will fit right in.

If you have a dependency that doesn't have a license, ask its maintainers to **add a license**.

I want it simple and permissive.

The **MIT License** is short and to the point. It lets people do almost anything they want with your project, like making and distributing closed source versions.

Babel, **.NET Core**, and **Rails** use the MIT License.

I care about sharing improvements.

The **GNU GPLv3** also lets people do almost anything they want with your project, *except* distributing closed source versions.

Ansible, **Bash**, and **GIMP** use the GNU GPLv3.

{ What if none of these work for me? }

My project isn't software.

There are licenses for that.

I want more choices.

More licenses are available.

I don't want to choose a license.

Here's what happens if you don't.