Working collaboratively with Git

Justin W. Flory, ISTE-121.01 (HW09)

Introduction

Git? What? What are we getting?

- **Git**: free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency
- English?
  - Save points in a video game
  - With Git, you can revert back to a previous time in your code
  - Also simplifies working collaboratively with others asynchronously
  - Imagine a ledger of all revisions to your code, like a list of save points

Application: Homework assignment

- Imagine you're working on a homework assignment
  - Each time you add a major feature or hit a part of the rubric, you "commit" your code with git
- You decide to be ambitious and go for the bonus points
  - Uh-oh, you accidentally the entire assignment
  - It's 1am and you want to go back to where you were three hours ago because that met the rubric requirements and was technically done
- Revert back to a specific commit with Git
  - Like magic, your code is restored to that point in time
- Hooray! You can go to sleep with a completed homework assignment and secret disappointment over not getting the bonus

“Tag” milestones of your code

- Another handy feature of Git is tagging
  - You can “tag” a commit with as a specific point in history, like a version number
- Create a “tag” as an easily referenceable milestone in your code
  - For example, a personal project you are working on
  - You get it working and minimal functional, hooray! v1.0
  - Later, you hope to expand the features and add some other things
  - Skip ahead in the future, yay! It’s ready. v1.1
  - Both tags are easily reference for later

Working collaboratively

Git is great at code parties

- Git makes collaboration with others easier
  - Git can handle commits from multiple “authors”, automatically merge them if there are conflicts (or make it easier to merge them manually if it can’t automatically)
  - More convenient than copy+pasting code among your team members
- Introducing pull requests *
  - Great way to handle **code review**
  - Partner 1 works on project and has code ready
    - Opens pull request against Git repo (i.e. your code homebase)
    - Pull request has the code viewable by anyone but it is not actually “in” the main code
    - Once teammates review and look over it, they can accept and merge, or reject the pull request

What did the * mean?

- A note about pull requests
  - Not really an “actual” feature of Git
  - More of a front-end tool
- More about this later

In the real world

Why bother with this git thing?

- Outside of using Git to help make your projects and homeworks easier, it is widely in use
- Git is an **industry standard** for version control
  - Likely to experience it eventually while you're at RIT
  - After graduating, if you write code in an organization or company, you are almost guaranteed to find this
  - It's literally everywhere
- Learning this now and applying it to "small" things like homeworks or projects is making your future self's life easier

“I can just learn this later.”

- You can learn it later, but employers are already looking to see if you know it
- Not uncommon for employers / interviewers to ask for your **GitHub profile**
  - More about GitHub in a moment
- Learning Git and open sourcing your code is a great way to show off your experience and knowledge for that co-op you really want
  - Also has an ethical aspect about **open source** and what exactly that means for you and your code

Git is going to get
you sooner or later.

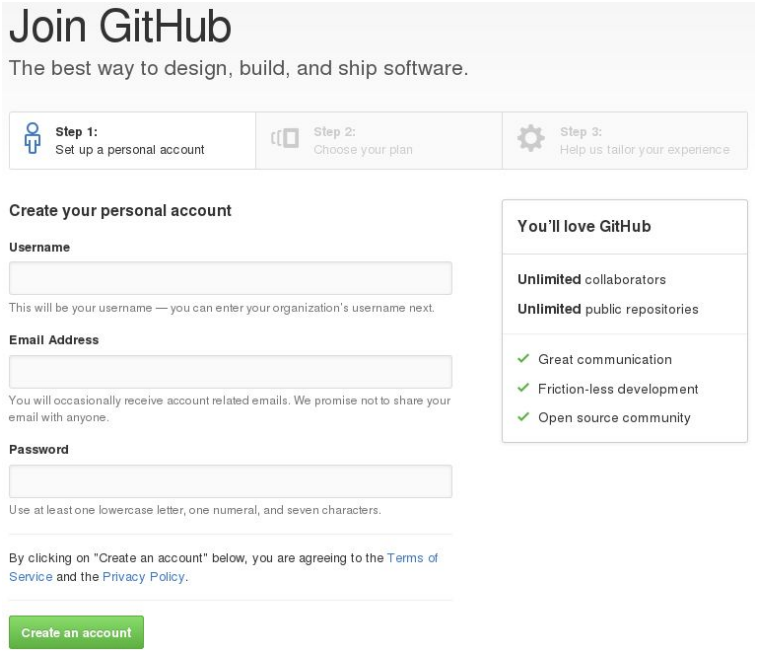
Introducing GitHub

Where does GitHub fit in?

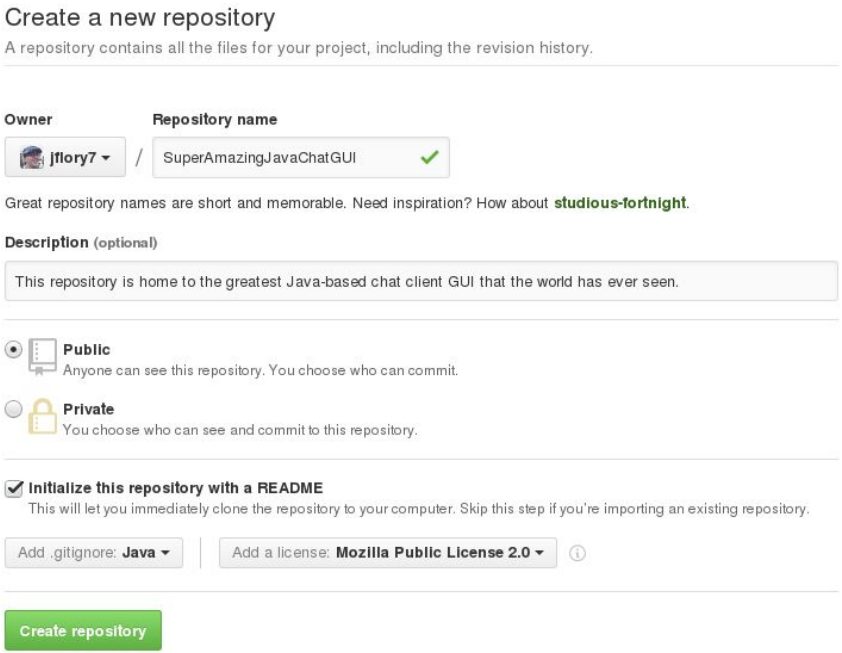
- First, know this: Git is just software
  - Anyone can download and install Git on their computer
  - It's the actual powerhouse behind this whole thing
- GitHub is a public place for you to host your Git repositories online
  - GitHub also adds a fancy web presence for your project on their website
  - Easy to browse code online and sometimes even make small edits
  - Also comes with unique tools like issue trackers, wikis, and project website hosting (for free!)
- GitHub is only a **frontend** for Git

Getting on GitHub

- Registering is super easy and quick
  - Drop a username, email, and password, and you're off
- Worth noting there are plenty of alternatives to GitHub
  - GitLab
  - Bitbucket
  - Pagure
  - And many others…
- GitHub is just the largest and most used site of its kind



Creating a repo



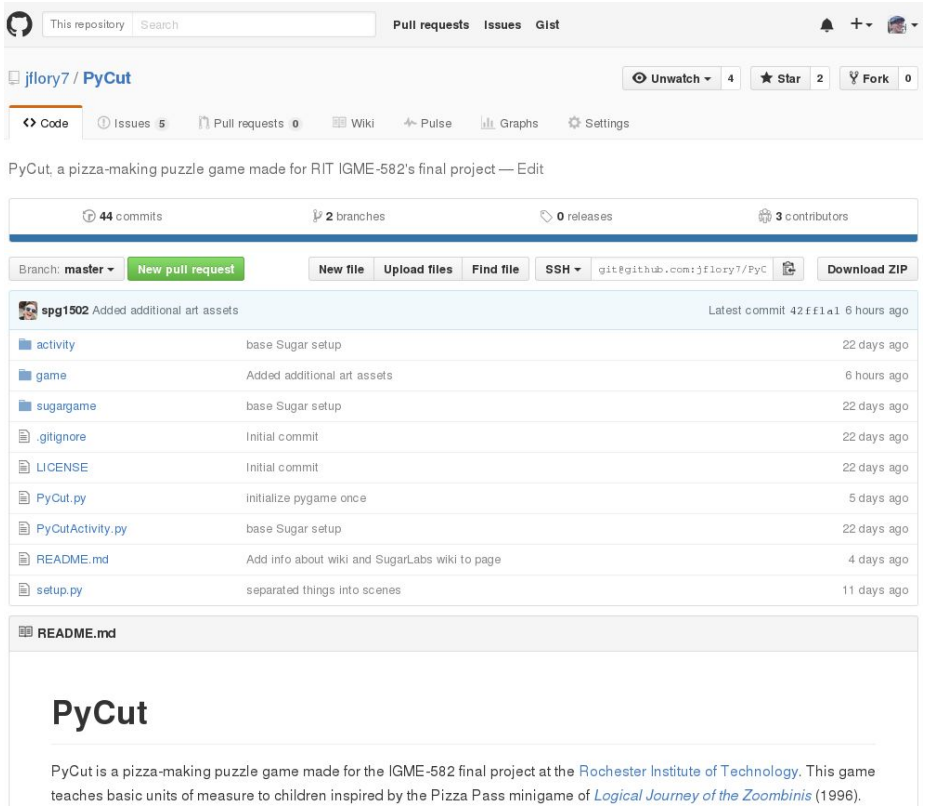
- To get started, you need a repo
- Create one when you get started or from the top of the page when logged in
- **Vocabulary**
  - Repository name: The name of your project
  - Description: A one-line description of your project
  - Public / Private: Whether your code is publicly visible (private repos cost money)
- Note the other bottom three items

Three small but important options

- *Initialize this repository with a README*
  - A README is a file that is displayed under your repo with information about your project - for both you and the future's sake, create one and put some basic info about your project in there later
- *Add .gitignore*
  - In short, you want this
  - Tells Git to ignore certain files, like .class files, which aren't readable by humans and should not be in your repository
- *Add a license*
  - This is a bigger topic and deserving of its own lecture / explanation
  - **Choosing a license** for your code is the **most important thing you can do** when open sourcing your code
  - The "rules" for how others can use your code

Your new repo

It may not look like this yet, but you too can aspire for a great and glorious GitHub repo too



Getting more help

Confused? Lost? Wondering how to actually do this?

- Worry not!
  - As mentioned earlier, Git really is used everywhere
  - And as a result, there are **countless websites, guides, and documentation** dedicated to helping teach how to use Git
- By default, Git is a command line utility
  - If you're just getting started, that can be an intimidating way to use Git
  - The GitHub Desktop app is a great place to get started (desktop.github.com)
    - Has an easy-to-use, understandable, and functional GUI for interacting with Git and GitHub

Learn by example

Working with a Git / GitHub repo

- For the purposes of this demonstration, we have a special repo that represents an ISTE-121 homework
  - See: https://git.io/vwE3O
- In this repo, let's…
  - Clone the repo
  - Add a new file
  - Modify an existing one
  - "Commit" the files
  - "Push" them from the computer to GitHub
  - Make a branch, make changes, submit pull request to master branch
  - Submit an issue for later

Live demo

Questions? Comments?

Justin W. Flory
ISTE-121.01 (HW09)

Credit

- Git: https://git-scm.com/
- GitHub: https://github.com/
- Learn Git in 15 mins: https://try.github.io/