COMP310/ECSE427

C Lab #1 - Intro to Git and workflow

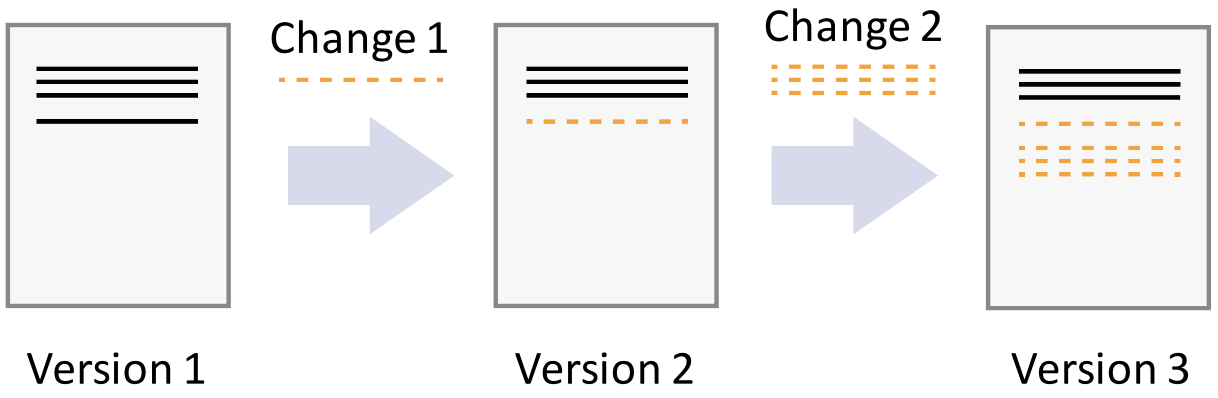
Sebastian Rolon

Overview

- Version Control Systems (VCS)
  - What are they and why use them
- Git
- GitLab
- COMP310 Autograder overview
- Lab exercises
  - CS Accounts
  - Mimi and VS Code
  - Coding and working with GitLab
  - Autograder in action with Hello World

Version Control Systems (VCS)

- Tools to collaborate and keep track of changes in code
- *The* solution to stop naming files "final", "final-2", "final-really"
- They record who made changes when



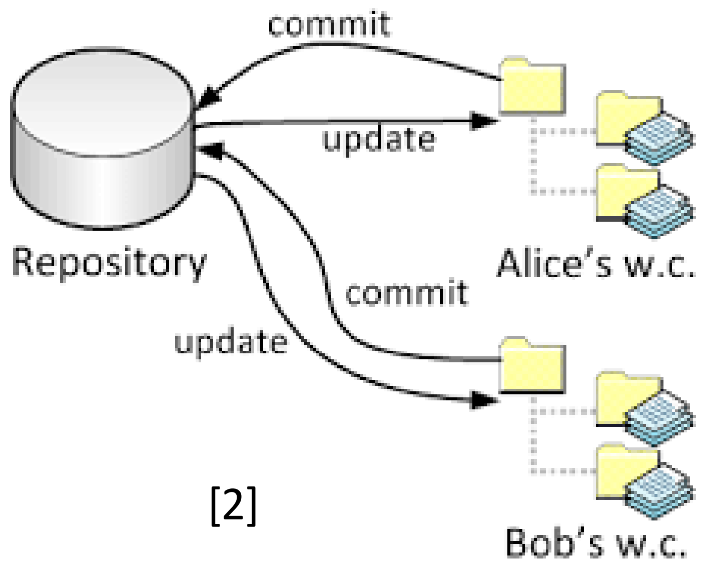
[1]

- Developers introduce changes and "save" versions
- Versions are stored in history forever and can be visited at will

Version Control Systems (VCS)

Simplest solution: Centralized Version Control

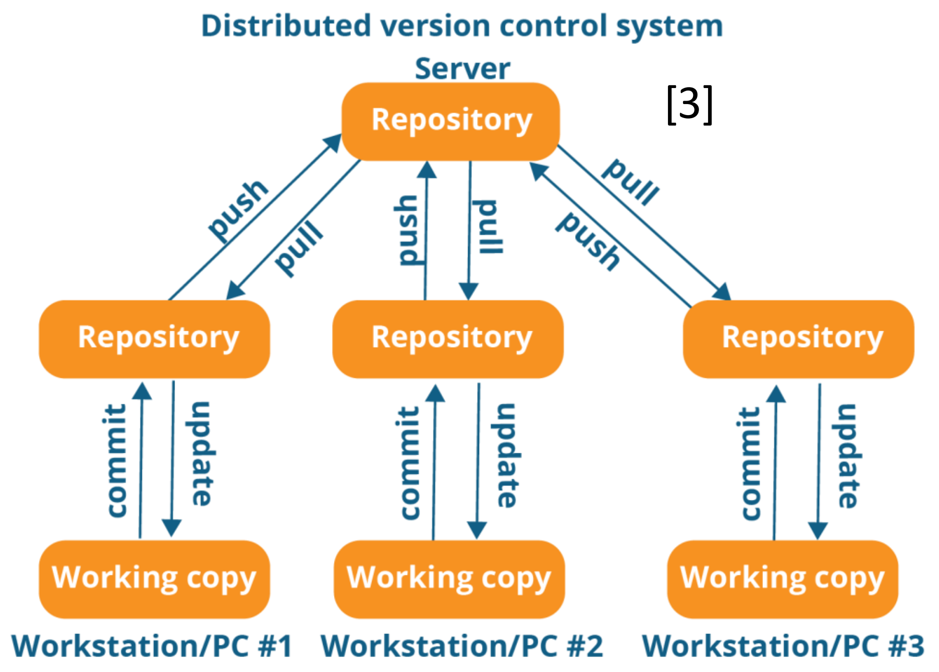
- Introduced around the 70s
  - Subversion (SVN) is one of the few remaining ones
- Version history is in a single server
- Developers need access to the repository to "save" their work
- What if the server dies? The history is completely lost



[2]

Git

- Distributed Version Control system
- Created in a month in 2005 by Linus Torvalds of Linux fame
- Each developer has an independent copy of the whole history
- If server dies, there are many backups of the history



[3]

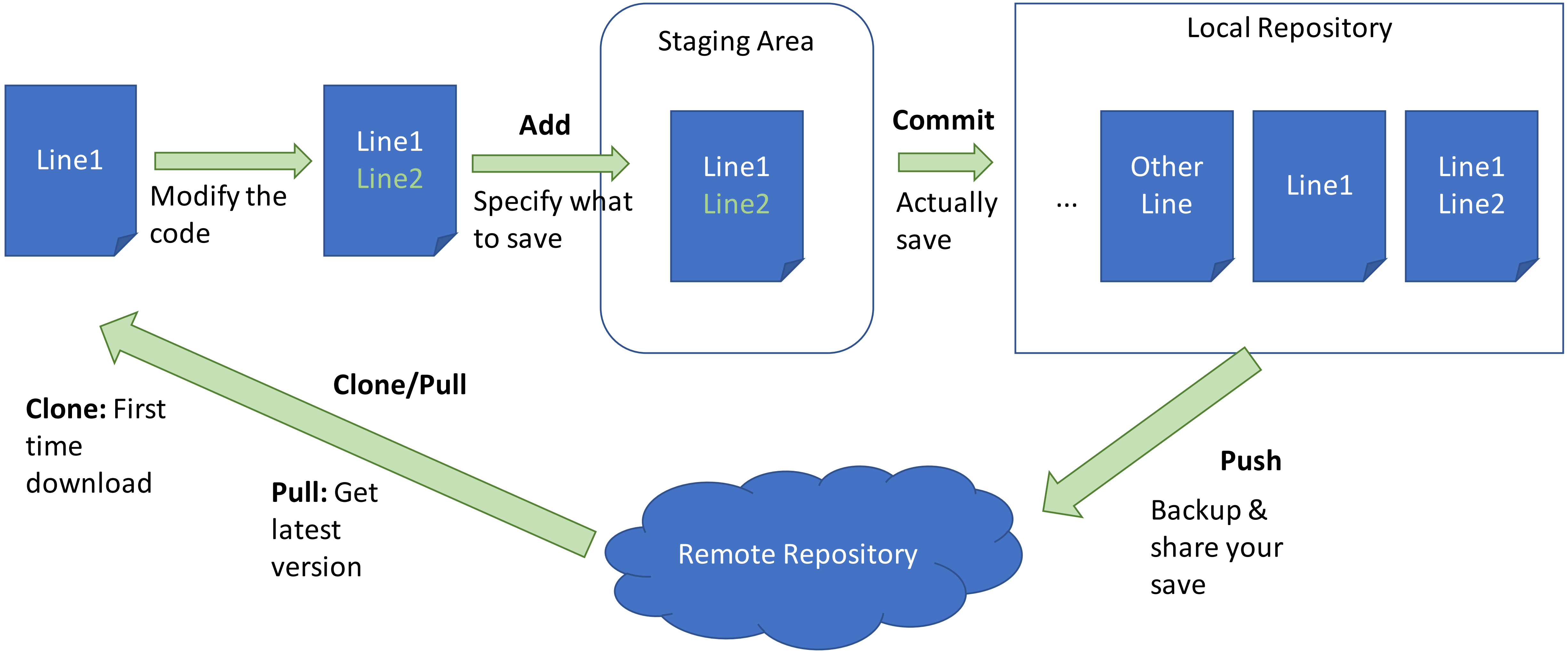
Git

- Git is very different than autosave on e.g. Google Docs
- Saving in Git involves numerous steps:
  1. Specify which files you want to save
  2. Name the save with a message describing the changes
  3. Upload ("backup") the save to the remote repository
- Why do this?
  1. Saves are never accidental, working history is clear
  2. The message helps you find the moment in time that you're looking for
  3. Multiple copies of the code are kept in different places

Git: Terminology

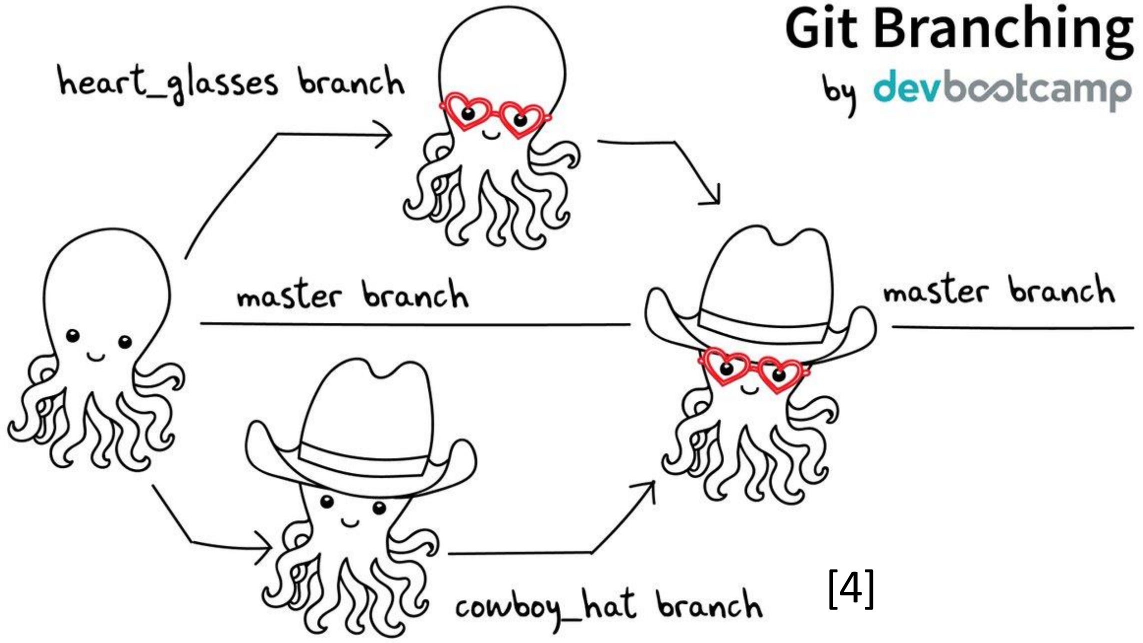
- **Repository:** A "folder" containing all the code files and the entire history of the code
- **Remote repository:** Your repository, but stored in a server where it is always accessible, safe, and your teammates can access it too
- **Staging area:** After you modify the code, the area in your computer where Git keeps track of which changes you want to save
- **Commit:** "Save" or register your changes into the history of the repository
- **Push:** Upload the saves to the remote repository
- **Pull:** Get the latest saves that people have uploaded to the remote repo
- **Clone:** Download the repository for the first time

Git: Add->Commit->Push



Git: Branches

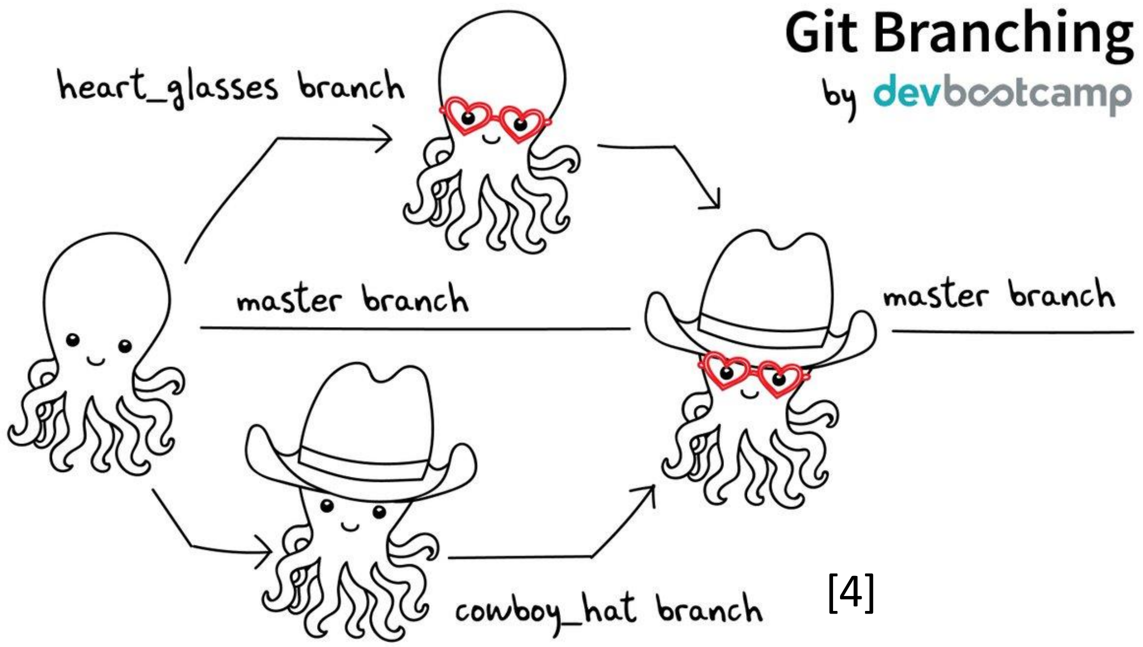
- Git allows you to have "parallel universe" versions of your code
- They are called **Branches**
- Branches are part of your repository
- Why have this?
    - Work on multiple ideas in parallel without risking the history of the code
    - Work on a feature until it's ready to be used without disturbing the main code



[4]

Git: Merging

- Merging two branches is combining their changes
- One of the branches ends up with all the changes
- What if two branches have different changes in the same place?
  - This is called a **merge conflict**
  - You will have to manually go through the conflicts and decide what to keep
- Merging is automatic if there are no conflicts



[4]

Git: Forks

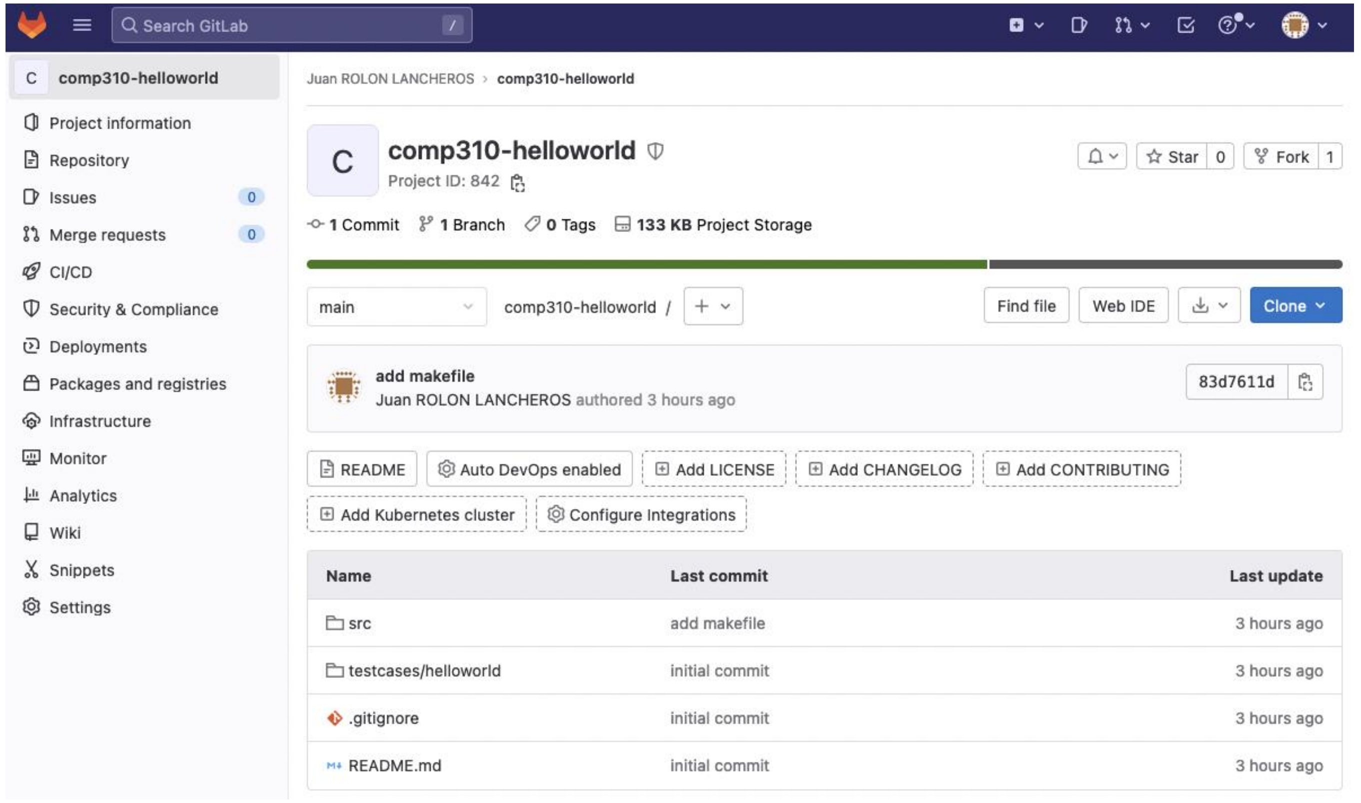
- Sometimes the changes that you want to do are too much for branches
- E.g. you found an open-source database, and you think you can modify it and sell it as a product
- The act of copying a repo wholesale is called **Forking**
- You have a complete copy where you can create new branches and modify the history without affecting the original
- We will use forks in the class to separate each team's code based on our starter code

Git: Remote repositories

- How do I set up a remote repository?
- Technically any repository can act as a main/remote server
- Practically this isn't done
  - Your computer isn't on all the time
  - You're on home internet and your ports are closed
  - You become vulnerable to attacks
- Repo providers are the answer
  - GitHub, GitLab, BitBucket, etc
- Bonus: you get a nice web interface to interact with your repo

GitLab (gitlab.cs.mcgill.ca)

- Widely used in industry
  - On-prem vs SaaS
- Includes other collaboration tools (CI, issue trackers)
- Has a REST API for automation
- Will be used throughout the course


Juan ROLON LANCHEROS › comp310-helloworld

comp310-helloworld

Project ID: 842

1 Commit · 1 Branch · 0 Tags · 133 KB Project Storage

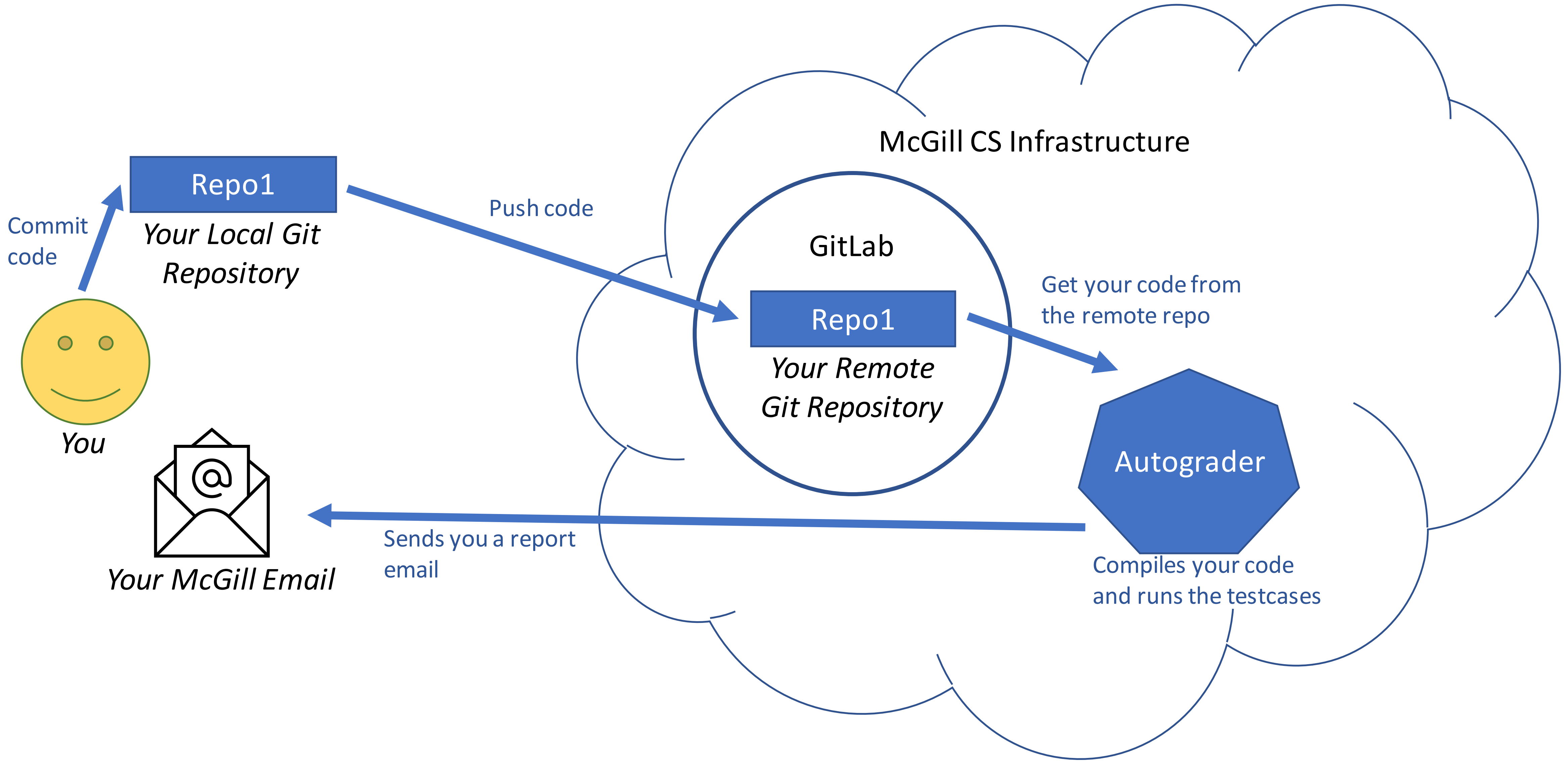
main · comp310-helloworld /

add makefile

Juan ROLON LANCHEROS authored 3 hours ago · 83d7611d

Name	Last commit	Last update
src	add makefile	3 hours ago
testcases/helloworld	initial commit	3 hours ago
.gitignore	initial commit	3 hours ago
README.md	initial commit	3 hours ago

GitLab and the Autograder


McGill CS Infrastructure
Commit code
Repo1
Your Local Git Repository
You
Push code
GitLab
Repo1
Your Remote Git Repository
Get your code from the remote repo
Autograder
Compiles your code and runs the testcases
Sends you a report email
Your McGill Email

Hands-on (1)

1. Create a McGill CS account: myaccount.cs.mcgill.ca/account
2. Log in to GitLab: gitlab.cs.mcgill.ca
3. Make sure your email is public in GitLab
4. Add your SSH key to GitLab
5. Create a fork of the helloworld repo: gitlab.cs.mcgill.ca/jrolon/comp310-helloworld
6. Give me Reporter access to the fork @jrolon
7. SSH into Mimi

- ssh <cs_username>@mimi.cs.mcgill.ca

8. Clone your fork

- git clone <https url of the repo>

9. Download Visual Studio and remote dev tools
    1. code.visualstudio.com
    2. marketplace.visualstudio.com -> Remote Development extension
    3. https://code.visualstudio.com/docs/remote/ssh#_connect-to-a-remote-host

Hands-on (2)

Other resources

- Git tutorial: https://swcarpentry.github.io/git-novice/
- Git explains what version control is: https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control
- Intro to Gitlab: https://youtu.be/_4SmIyQ5eis?t=90

References

1. Version Control with Git. The Carpentries. https://swcarpentry.github.io/git-novice/
2. Subversion – Source Code Control. Doug Harper. http://physics.wku.edu/phys316/software/svn/
3. What Is Git ? – Explore A Distributed Version Control Tool. Reshma Ahmed – Edureka. https://www.edureka.co/blog/what-is-git/
4. Git Intro – Branching and Merging. Code Refinery. https://coderefinery.github.io/git-intro/branches/