

CSci 3081W: Program Design and Development

Week 1

Lecture 02 – Version Control and Git

Discord Demo and Commands - <https://discord.gg/P2w9cQPn>

- !help – shows all commands
- !queue join – adds you to the Queue
- !queue show – displays the Queue
- !queue leave – removes you from the Queue

TAs can use the following:

- !queue next – pops the first off of the Queue, now it's their turn
- !queue next <int> - removes that indexed student from the Queue and then it's their turn
- !queue clear – clears the queue
- !queue add @username – adds the user to the Queue (corrects mistakes)

What is version control?

Also known as source control

Why is it important for development?

Talk to your neighbors



THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.

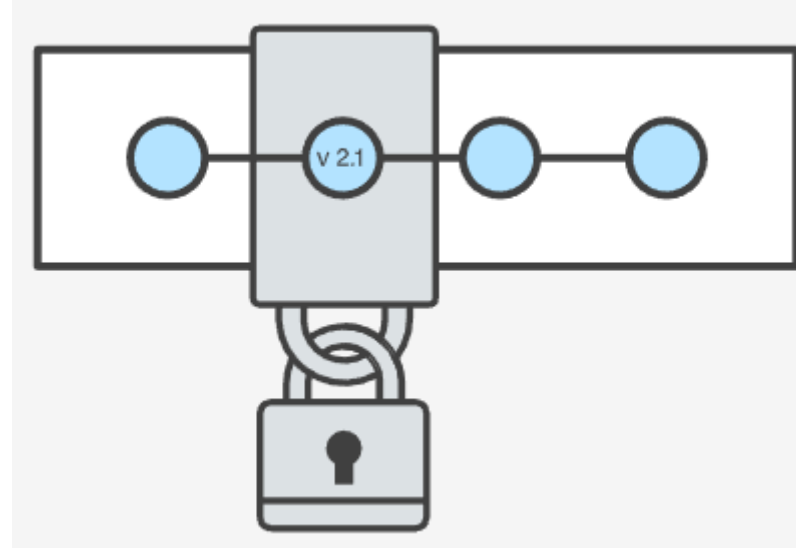
COOL. HOW DO WE USE IT?

NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.



Definition

- Practice of tracking and managing changes to software code
- Version control systems
 - Software tools like Git that manage changes to source code over time
- What does it do?
 - Tracks every modification to the code in a database – allows rollbacks and comparisons



Source Code

- The main code (main branch)
 - The crown jewel!
 - Protect it at all costs
- The source code should never have bugs in it

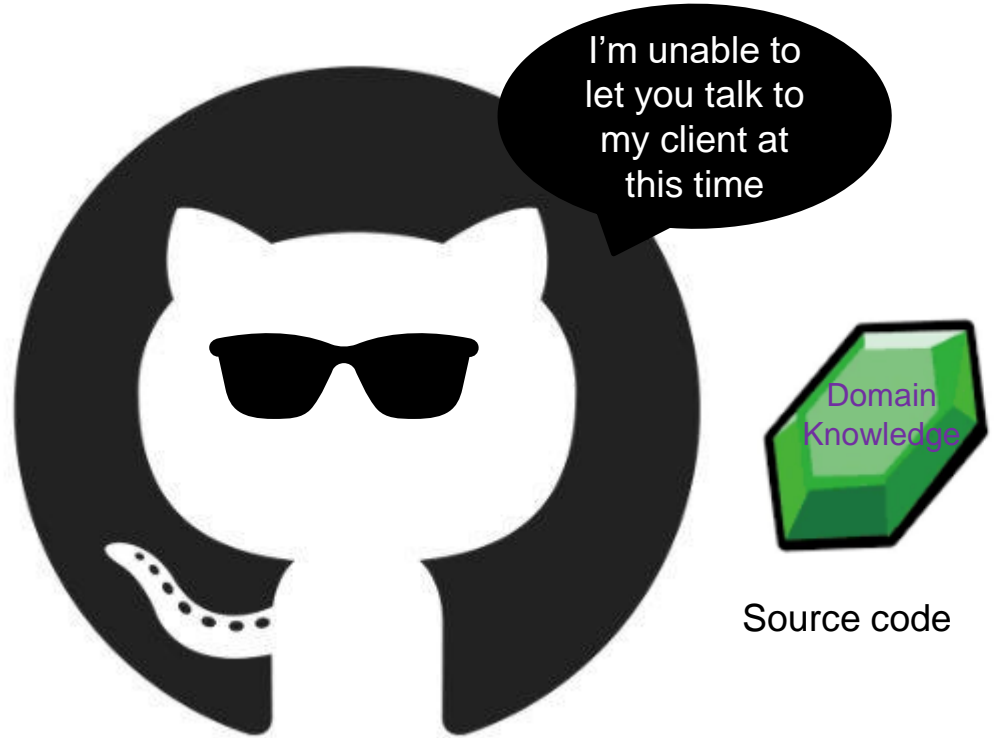
The source code of Windows' troubleshooting program has leaked

```
1  #include <windows.h>
2  #include <stdio.h>
3
4  int main() {
5      printf("Searching for problems...\n");
6      Sleep(60000);
7      printf("We didn't find any problems\n");
8  }
9
```

* I like the Windows troubleshooting program, this is just a joke

Protect the source code with Version Control

- Prevents code from “not working anymore”
 - Due to integration errors or human errors



Git – Version Control

Two Developers Example

- There exists a directory structure (file tree)
- One developer is writing new code with new functionality that relies on old code
- The other is fixing an unrelated bug

Version control's role is to track every change by each developer such that concurrent work doesn't conflict.

What if we don't use version control?

What if we don't use version control?

- Don't know which developer made which change
- Don't know if we created any integration failures
 - Leads to reworks and refactors



ParseToTable.ipynb



ParseToTable2.ipynb



ParseToTableD1.ipynb



ParserPipeLine_finalv5.ipynb

Version control is mandatory for all group work

- Version control is also helpful for solo projects
- You'll do some version control work for lab 1

Version Control Systems (VCS)

- AKA Source Code Management (SCM) or Revision Control System (RCS)
- Most popular: Git
 - Git is technically a Distributed VCS (DVCS)
- Benefits:
 - Long term history for all changes for all files
 - Branching and Merging
 - Traceability



History of changes

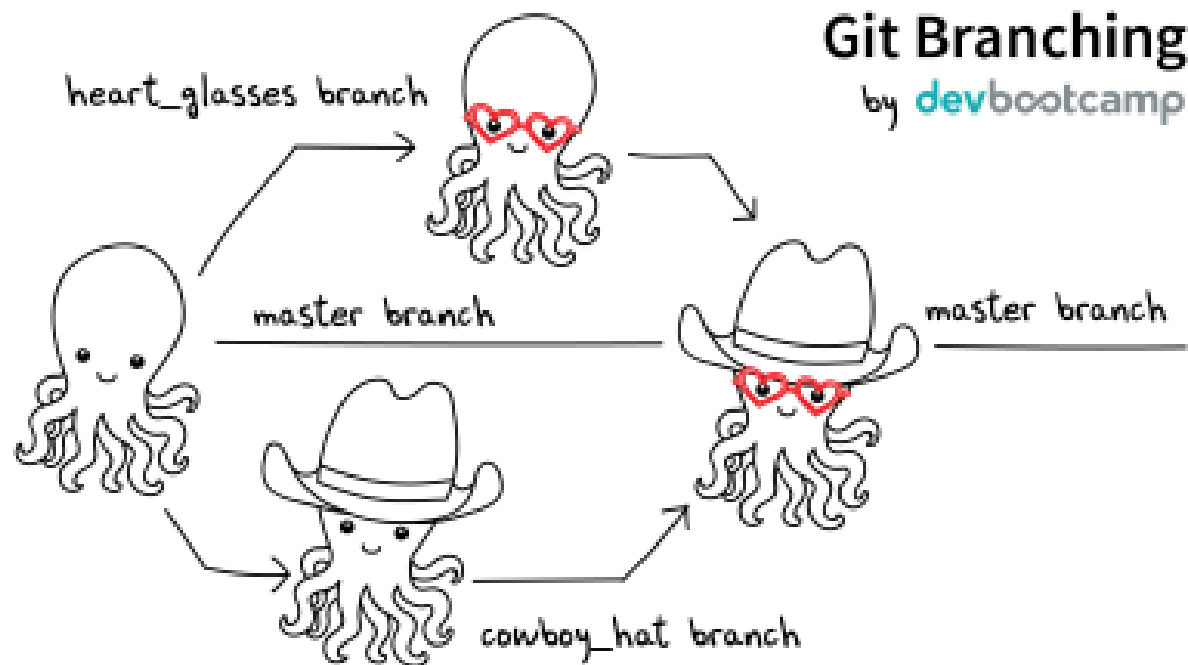
- Includes edits as well as creation/deletion of files
- Includes author, date, and any notes that the author included
 - `git commit -m "notes go here, can reach a higher character count through Git's website"`
- Enables us to go back to previous versions to find insight or causes for bugs

History of changes

```
41      42      """
42      43      try:
43      -      client = MongoClient(url)
44      +      client = MongoClient(url, tlsCAFile=certifi.where())
44      45      self.conn = client
45      46      except:
46      47      print("\033[1;31mconnect(): Failed to connect to ", url, "\033[0m")

      ↓
      ↑
      *****
@@ -375,4 +376,4 @@ def compare_dicts(self, dict1, dict2):
375      376      if (l == [] and v == []):
```

Branching and Merging



Branching and Merging

- Team members must work concurrently!
 - But even solo projects can benefit from working on independent streams of changes
- Creating a branch in a VCS keeps multiple streams of work independent from other
 - Also allows the merging of these branches together, enabling developers to make sure that the changes in each branch don't conflict

Branching and Merging

Default branch


`main` Updated 9 months ago by dtorban

Default

Your branches


`csv` Updated 11 months ago by gango010

167 | 22

 New pull request


`singleton` Updated 11 months ago by gango010

167 | 21

 New pull request

`shonal-devel` Updated 11 months ago by gango010

30 | 2

 New pull request

Stale branches


`repo-ying` Updated 12 months ago by LU000097

172 | 1

#1  Closed

`lab03` Updated 12 months ago by BACKM083

136 | 1

 New pull request

`lab03-edits` Updated 12 months ago by BHAND092

123 | 0

#2  Merged

`lab3-checkpoint2-sol` Updated 12 months ago by bikax003

126 | 0

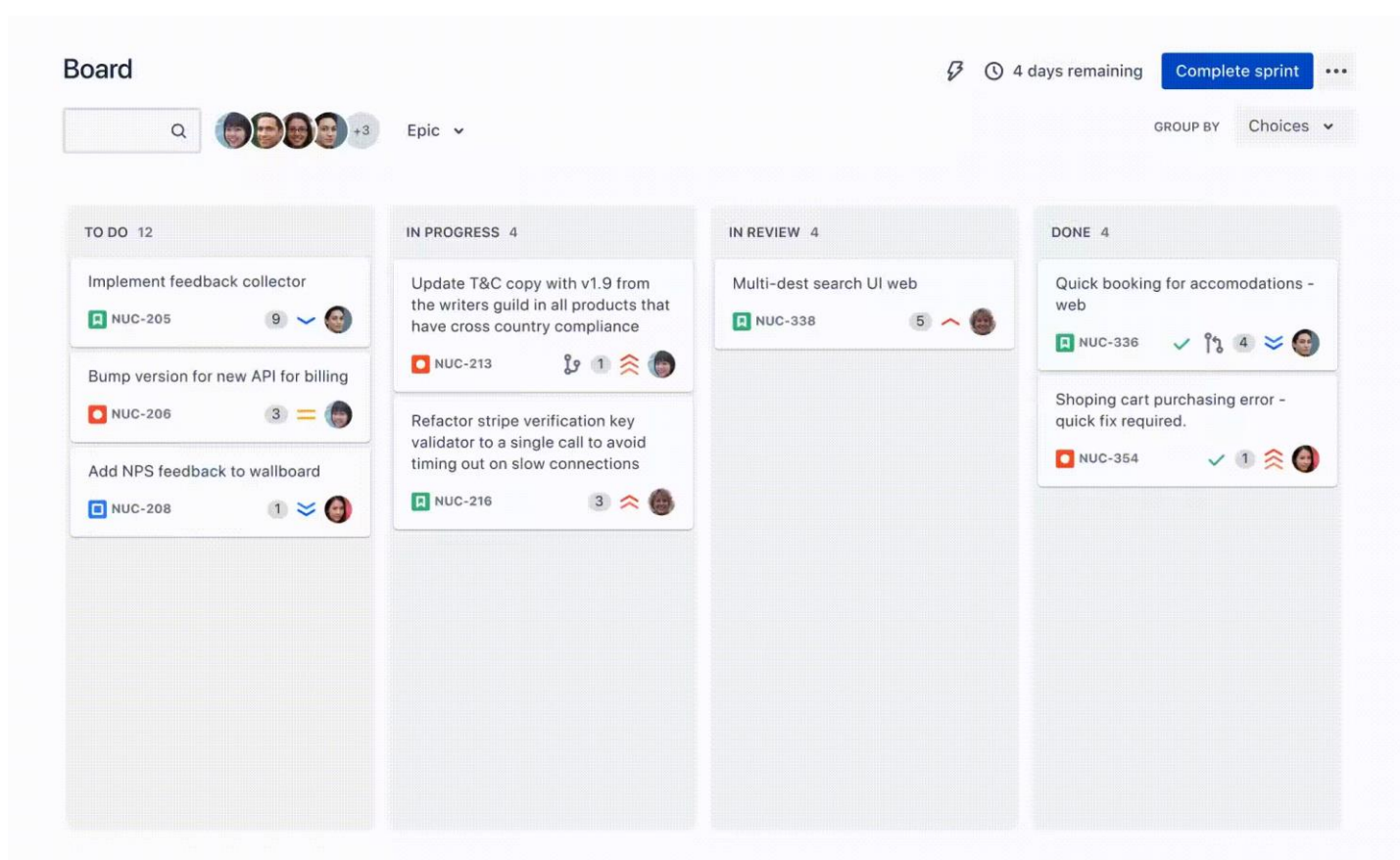
#3  Merged

`shivam-flask` Updated 11 months ago by BHAND092

35 | 1

#6  Open

Traceability







- We can trace changes and connect them to project management and software such as Atlassian's Jira.

Version Control

The question isn't "should I use version control?". It's "which one do I use?".

- We will use Git
- Honorable mentions: Mercurial, SVN, CVS

Software	Network architecture	Conflict resolution	Development status
 Git	Distributed	Merge	Active
 Mercurial	Distributed	Merge	Active
 SVN	Client-server	Merge or lock	Active
 CVS	Client-server	Merge	Maintenance only

Git

- Linus Torvalds, 2005 (also creator of Linux OS kernel)
- What does the D in DVCS really mean?
 - Rather than one place for the full version history, every developer's working copy of the code is also a repository that can contain the full history of all changes
- In addition to being distributed, it's also been designed with performance, security, and flexibility in mind.

Git is the best choice for VCS

- Has the functionality, performance, security, and flexibility that most teams need
- Most broadly adopted tool of its kind
 - Atlassian, CLion, Visual Studio Code, etc.
- Quality open source project
 - Plenty of documentation too!

Git Cheat Sheets

- Linked in Canvas:
 - Sec 001:
<https://canvas.umn.edu/courses/333083/files/folder/GitCheatSheet?preview=30204593>
 - Sec 010:
<https://canvas.umn.edu/courses/333057/files/folder/GitCheatSheet?preview=30204614>

Git

	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT
MESSAGES GET LESS AND LESS INFORMATIVE.

Lab 1 - Git

- <https://github.umn.edu/umn-csci-3081-F22/public-lab01>
 - Lab 1 is on Friday, September 9th
 - Due Thursday, September 15th at midnight

Workshop 1 – C++ Basics

- Will be held in class on Tuesday, September 13th
 - Due Thursday, September 15th at midnight
 - **requires lab 1 completion for last step of workshop**

Homework 1 – C++ Basics

- Will be posted Friday or sometime this weekend