Lecture 1:
Course Overview & Introducing C

Sarah Nadi
nadi@ualberta.ca
Department of Computing Science
University of Alberta

CMPUT 201 - Practical Programming Methodology
Winter 2021

Agenda

- Course Overview
- Introducing Unix & C
- Introducing version control using git and GitHub

Readings

- Look at eClass course page
- Watch git tutorials
- Textbook: Ch 1 & 2

Course Overview

Course Objectives

- Understand how programs work in the background through teaching the C programming language
  - memory, debugging, execution, C syntax (not the main goal, but is a side effect),
- Teach the basics of UNIX programming
  - using shell commands, standard in-out-error, pipes etc.
- Teach some practical aspects of programming methodology
  - common modularization techniques
  - testing
  - source code version control - using git

Official Class Information

- Course outline: https://eclass.srv.ualberta.ca/pluginfile.php/6762210/mod_resource/content/2/CMPUT201_Outline.pdf

- Overview of course schedule:https://eclass.srv.ualberta.ca/mod/url/view.php?id=4875999

- Textbook: K.N. King. C Programming: A Modern Approach, 2nd Edition, W.W. Norton & Company, 2008 .

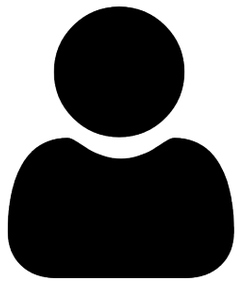
Student Evaluation

Component	**Details**	**Weight**
Quizzes (individual)	Best 8 out of 9 quizzes	8% total
Assignments (in pairs)	3 pair assignments	15% total
Lab Exercises (individual)	Best 10 out of 11 exercises	10% total
Lab Demos + oral exams (individual)	Best 5 out of 6 demos/exams	15% total
Midterms (individual)	2 midterm exams	30% total
Final (individual)	1 final exam	21%
Plagiarism quiz (individual)	Plagiarism quiz done at beginning of course	1%
Bonus participation	Bonus participation throughout the course	3%

But .. We are living through a pandemic after all …

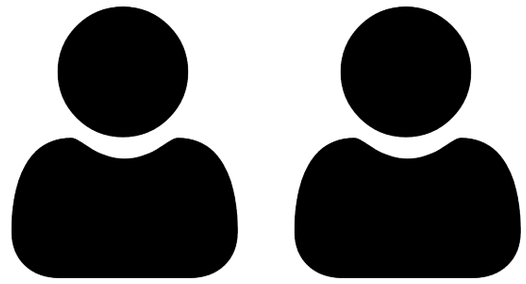
New changes to cater to new realities

Pre-pandemic



Individual Assignments

New in Winter 2021



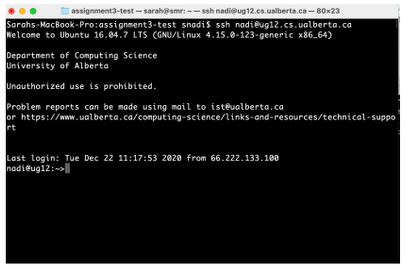
Assignments will be done in pairs

- More social interaction
- Learn together and from each other

All other course components (quizzes, labs, demos, and exams) are strictly individual though!

New changes to cater to new realities

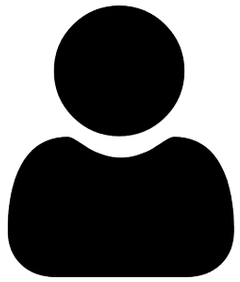
Pre-pandemic



Attend labs in order to demo

No demo, no grade

New in Winter 2021



Submit lab exercise through GitHub

Have to demo only half of labs

Demos have a separate grade

Demos include an oral exam

- More flexibility for students
- Factors in call-switching time etc. for TAs
- Can use demos for individual assessment

New changes to cater to new realities

Pre-pandemic

Written in-class midterm exams

New in Winter 2021

UNIVERSITY OF ALBERTA eClass

Midterms will be in the form of timed eClass quizzes available for ~24hrs

Lab demos will include an oral exam component

- Avoid time zone differences
- Accommodates any personal circumstances
- Additional multiple short oral exams through the term diversifies assessment

Icons made by Freepik www.flaticon.com

New changes to cater to new realities

Pre-pandemic

8 quizzes, all count, but for 0.5% each

New in Winter 2021

UNIVERSITY OF ALBERTA eClass

9 quizzes w/ best 8 counting towards your final grade

Each quiz counts for 1%

- Quizzes may have harder questions to be used as sample questions for midterm exams
- Give students a breather; one more free excused absence

Icons made by Freepik www.flaticon.com

Quizzes

- 9 quizzes w/ best 8 used for final grade— check schedule on eClass

- 5-10 simple questions per quiz

- Quizzes become available on Monday 8am and close on Tuesday 8am

- Quizzes are timed (15-25min depending on quiz)

- **Do not post questions about quizzes during quiz time**

- Quizzes are meant to help you make sure you are on track

Assignments

- Three assignments almost 4 weeks apart

- Each assignment has explicit instructions on expectations. When in doubt, ask on the discussion forum!

- Each assignment counts for 5% of your grade

- Assignments are done in randomly assigned pairs (different partner for each assignment). We will announce assignment 1 partners on Friday & also how your GitHub repos will work.

- Assignments will be automatically graded so **you must make sure to adhere to the requirements. If we cannot automatically grade your submission, you will get a 0.** We will provide scripts and test cases that will help you with that.

Lab Exercises

- Labs will start on Monday January 18th, 2021
- There are 11 lab exercises, each worth 1% of your final grade. Best 10 out of 11 lab exercises will be used towards your final grade (which means you have 1 free excused absence). **Labs due midnight on the day of your lab**
- **Read/watch tutorials & finish exercise before the lab**
- Before next week, make sure that:
  - You can ssh into the lab machines
  - Have a GitHub account (https://github.com/)
  - Have a CMPUT201 GitHub labs repository (more on that later)
  - Contact HelpDesk if you cannot ssh into the lab machines

Lab Demos/Oral Exams

- You are required to demo only 6 of the lab exercises (with best 5 counting towards your final grade)

- You will be pre-assigned to the labs you must demo. Check which labs you are assigned to and sign up for a demo slot. Exact process and details under “Labs & Lab Demo Guidelines” on eClass (https://eclass.srv.ualberta.ca/mod/page/view.php?id=4720430)

- Demos/oral exams may include:

  - Demoing parts of your lab exercise

  - Answering questions about your solution to the lab exercise

  - Answering questions related to the material covered in that week’s lab exercise

How Lab Demos Work

CMPUT 201 Lab Demos: H03

File Edit View Insert Format Data Tools Add-ons Help Last edit was made yesterday at 4:40 PM by Ildar Akhmetov

Share

100% $ % .0 .00 123 Calibri 11

	A	B	C	D	E	F	G	H	I	J	K
1		First name	CCID	Lab 1	Lab 2	Lab 3	Lab 4	Lab 5	Lab 6	Lab 7	Lab 8
2	0			0	0	1	0	1	0	1	1
3	1			1	0	0	0	0	1	0	1
4	2			0	0	1	0	1	1	0	1
5	3			0	0	0	0	1	0	1	0
6	4			1	1	1	0	1	1	0	0
7	5			0	1	0	1	1	0	1	0
8	6			1	0	0	1	0	0	1	0
9	7			0	1	1	0	0	1	0	0
10	8			0	1	0	0	1	0	1	0
11	9			0	0	0	1	1	1	0	1
12	10			1	0	0	1	0	1	0	0
13	11			0	1	1	0	0	0	0	1

Which labs do you need to demo? Book your time slot here! Explore

How Lab Demos Work

CMPUT 201 Lab Demos: H03 — Saved to Drive

File Edit View Insert Format Data Tools Add-ons Help — Last edit was seconds ago

Share

Available

	From	To	Demos (Puyan Liu) (insert Name + a Google Meet link)	Demos (Kushol Rafsanjany) (insert Name + a Google Meet link)	Demos (Habibur Rahman) (insert Name + a Google Meet link)	Demos (Ryan Hang) (insert Name + a Google Meet link)	Q&A Session (Zoom)
2	5:00 PM	5:10 PM	Available	Available	[redacted]	[redacted]	Habibur Rahman Ryan Hang Nahin Chowdury Eric Brisson
3	5:10 PM	5:20 PM	Available	Available			
4	5:20 PM	5:30 PM	Available	Available			
5	5:30 PM	5:40 PM	Available	Available			
6	5:40 PM	5:50 PM	Available	Available			
7	5:50 PM	6:00 PM	Available	Available			
8	6:00 PM	6:10 PM	Available	Available	Available	Available	
9	6:10 PM	6:20 PM	[redacted]	[redacted]	Available	Available	Puyan Liu Nahin Chowdury Eric Brisson
10	6:20 PM	6:30 PM			Available	Available	
11	6:30 PM	6:40 PM		Available	Available	Available	
12	6:40 PM	6:50 PM		Available	Available	Available	
13	6:50 PM	7:00 PM		Available	Available	Available	
14	7:00 PM	7:10 PM		Available	Available	Available	
15	7:10 PM	7:20 PM	Available	[redacted]	Available	Available	
16	7:20 PM	7:30 PM	Available		Available	Available	Nahin Chowdury
17	7:30 PM	7:40 PM	Available		Available	Available	
18	7:40 PM	7:50 PM	Available		Available	Available	
19							
20			Slots available:	44			
21							
22			**Q&A Session Link:**	https://ualberta-ca.zoom.us/j/96686093041?pwd=Wk5MbUVMNnlBQWN0bGJTNkZrTld4Zz09			
23							

Midterms and Final

- Two midterms (see Schedule on eClass)
- Midterm open for a 24hr period, and timed once started
- Final exam will be synchronous during a fixed slot, as determined by the Registrar's office
- Exams are through eClass in the form of an eClass quiz.
- All exams must be done on an individual basis and without using any additional resources. Any suspected plagiarism cases will be investigated
- Questions and choices are randomized. Not all questions are multiple choice. Some questions may require uploading a picture of your solution on a piece of paper.
- No online proctoring tools will be used

Important Information

- Read the course policies on eClass. Pay specific attention to the collaboration policy.

- **General rule: No late submissions accepted for quizzes, assignments, or lab exercises**

- **All labs and assignments will be marked on the lab computers.**

Collaboration Policy

- **Write your own solution. All labs, exams, and quizzes are individual!**
- **Only assignments will be done in (Randomly assigned) pairs.**
- You **must not** post your solutions on public code repositories — use the provided private repo & take a look at the CMPUT 201 License that applies to all solutions submitted in this course.
- What does collaborating (beyond your assigned partner for pair assignments) mean ?
  - You can verbally talk about how to solve a particular problem in the assignment
  - You can draw pictures (e.g., of linked lists)
  - You can share test cases on the forums.
  - **You cannot share code. You cannot look at someone else's code. You cannot sit together (even if virtually) side by side and code the solution step by step together. You cannot post any code snippets on the forum. You cannot have access to somebody else's code to "guide" you through the solution… you get the picture :-)**
- We will check for plagiarism both manually and through automated tools.

The “Ugly” Slide

- Each term, we report at least 10 plagiarism cases
- Sanctions included 0 on the assignment, letter grade reduction, suspension, and prevention from graduation. Some were in addition to an “8” note on your transcript.
- Getting an extra mark or two on an assignment is not worth jeopardizing your future!

Plagiarism Quiz

- https://eclass.srv.ualberta.ca/mod/quiz/view.php?id=4872008
- Plagiarism quiz is open all this week and closes Friday Jan 15th at midnight
- You have unlimited attempts
- You must get 100% on this quiz to get credit for doing the quiz
- If you do not get credit for this quiz, you will not receive a grade for this course
- The plagiarism quiz is worth 1% of your total grade

Bonus Participation

- 3% bonus participation (i.e., this is above the course components adding up to 100%)

- Instructors will monitor participation throughout the term and determine the appropriate bonus participation grade at the end of the term.

- Example activities that count towards participant:
  - Questions in class
  - Participation in forum (especially answering other students' questions)
  - Participation in any in-class exercises
  - Sharing well-formatted and well-thought through class notes on the course forum.

Class Setup

- Synchronous classes through Zoom with recorded videos available afterwards

- Slides are shared on eClass

- Depending on that week's content, the instructor may choose to use slides, demos, in-class exercises, or hand-written explanations

- For content usually covered in labs, pre-recorded videos will be available beforehand so you can finish the exercises before the lab

Getting Help From Course Staff

- Post on the discussion forum or respective assignment forums (best chances of getting help from 2 instructors, 10 TAs, & over 200 students!)

- Go to one of the 9+ office hours available each week (options with both instructors and TAs). See “Contact Information and Office Hours” on eClass https://eclass.srv.ualberta.ca/mod/page/view.php?id=4720420

- If all else fails, email TAs or your section instructor:

  - Clearly state the course number. Include your name, ccid, GitHub username

  - Clearly state your problem and what you have already tried (simply saying “I cannot do X” will not help us solve your problem)

Introducing C & Unix

Why Unix?

- Dominant server operating system
- Uses open standards (e.g., POSIX threads)
- Free open-source versions available (FreeBSD, OpenBSD, Linux)
- Many free software development tools (e.g., gcc, emacs, gdb etc.)
- We will be using Linux in the labs

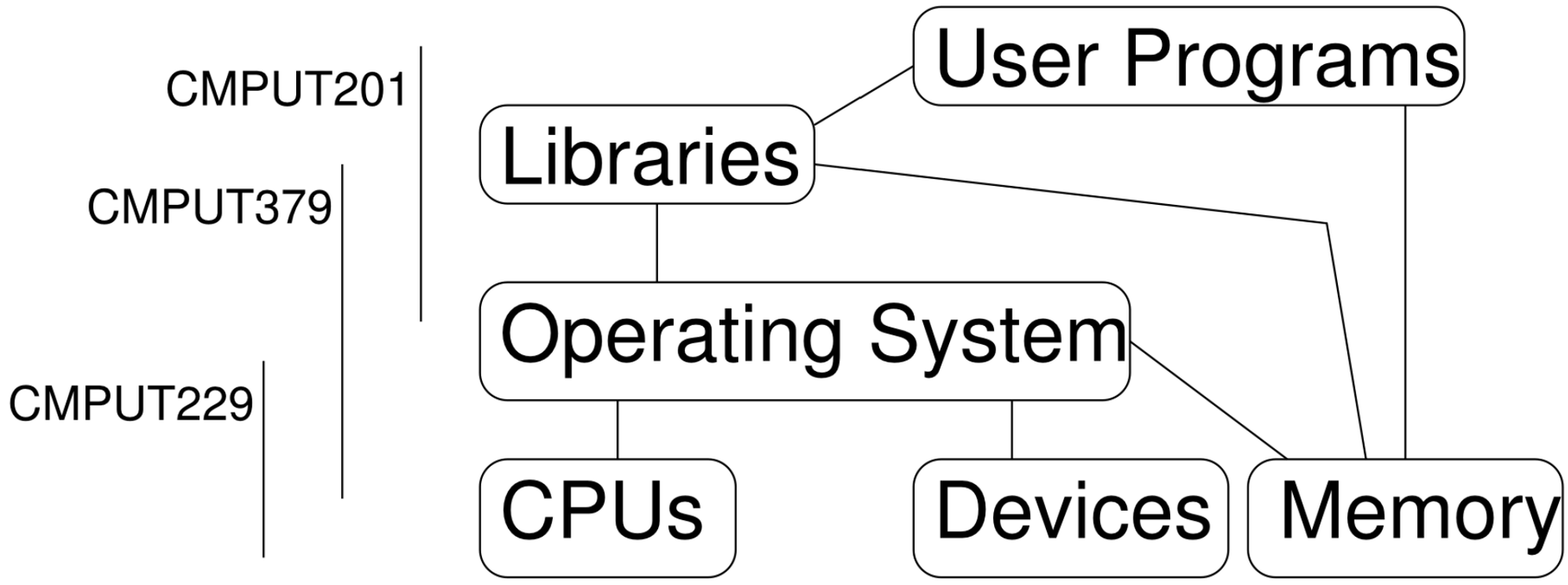
Why C & What is C?

- History of C
  - a byproduct of the UNIX operating system
  - a higher-level language than assembly
  - mostly done in the 1960s and 1970s
  - different language standards that evolved over time
- C influences many modern programming languages
  - C++, Java, C#, Perl
- Understanding how memory works helps you understand other programming languages as well

Strengths/Weaknesses of C

- Strengths:
  - efficiency, portability, power, flexibility, standard libraries
  - low-level, access to machine-level concepts
  - small, limited features
  - permissive; you need to know what you are doing
- Weaknesses:
  - error-prone, difficult to understand, difficult to modify

Relationship to Other Courses



In 201, we take a look at **using** the operating system UNIX & the high-level programming language C

Relationship to the “Outside World”

Learning C also allows you to understand & appreciate higher-level programming languages, because you know what goes on “behind the scenes”

The UNIX/Linux Shell

- The machines in the CMPUT 201 lab are ugXX.cs.ualberta.ca, where XX ranges from 00 to 34

- If you are using a UNIX-based OS (e.g., Ubuntu or MacOS), just go to your terminal and use **`ssh`** to connect to the lab computer

- If you are using Windows, you will need an ssh client such as PuTTY

Example of Connecting to Lab Computer

```
Sarahs-MacBook-Pro:~ snadi$ ssh nadi@ug12.cs.ualberta.ca
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.10.0-28-generic x86_64)

Department of Computing Science
University of Alberta

Unauthorized use is prohibited.

Problem reports can be made using mail to ist@ualberta.ca
or https://www.ualberta.ca/computing-science/links-and-resources/technical-suppo
rt

nadi@ug12:~>
```

demo

Example of Connecting to Lab Computer

Shell/ terminal

```
Sarahs-MacBook-Pro:~ snadi$ ssh nadi@ug12.cs.ualberta.ca
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.10.0-28-generic x86_64)

Department of Computing Science
University of Alberta

Unauthorized use is prohibited.

Problem reports can be made using mail to ist@ualberta.ca
or https://www.ualberta.ca/computing-science/links-and-resources/technical-suppo
rt

nadi@ug12:~>
```

demo

Example of Connecting to Lab Computer

the ssh command

Shell/
terminal

```
Sarahs-MacBook-Pro:~ snadi$ ssh nadi@ug12.cs.ualberta.ca
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.10.0-28-generic x86_64)

Department of Computing Science
University of Alberta

Unauthorized use is prohibited.

Problem reports can be made using mail to ist@ualberta.ca
or https://www.ualberta.ca/computing-science/links-and-resources/technical-suppo
rt

nadi@ug12:~>
```

demo

Example of Connecting to Lab Computer

the ssh command

your ccid

Shell/ terminal

```
Sarahs-MacBook-Pro:~ snadi$ ssh nadi@ug12.cs.ualberta.ca
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.10.0-28-generic x86_64)

Department of Computing Science
University of Alberta

Unauthorized use is prohibited.

Problem reports can be made using mail to ist@ualberta.ca
or https://www.ualberta.ca/computing-science/links-and-resources/technical-suppo
rt

nadi@ug12:~>
```

demo

Example of Connecting to Lab Computer

the ssh command

your ccid

address of lab machine

Shell/ terminal

```
Sarahs-MacBook-Pro:~ snadi$ ssh nadi@ug12.cs.ualberta.ca
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.10.0-28-generic x86_64)

Department of Computing Science
University of Alberta

Unauthorized use is prohibited.

Problem reports can be made using mail to ist@ualberta.ca
or https://www.ualberta.ca/computing-science/links-and-resources/technical-suppo
rt

nadi@ug12:~>
```

demo

Shell Commands

- `ls, mkdir, cd, cat, vi, gcc, cp, rm, mv, …`
- `man` (manual pages are extremely useful)
- We will cover these commands throughout the course, especially in the Labs.
- You should also get used to searching for certain commands yourself.

demo

Sample C program:

```
#include <stdio.h>

int main(void) {
   int classNumber;
   printf("What is your class number? ");
   scanf("%d", &classNumber);


   printf("Hello CMPUT %d!\n”, classNumber);
   return 0;
}
```

Sample C program:

Only few things are included by default in C. You need to include the libraries you will use

```
#include <stdio.h>

int main(void) {
   int classNumber;
   printf("What is your class number? ");
   scanf("%d", &classNumber);


   printf("Hello CMPUT %d!\n”, classNumber);
   return 0;
}
```

Sample C program:

Only few things are included by default in C. You need to include the libraries you will use

the main function is the main entry point to your program

```
#include <stdio.h>

int main(void) {
   int classNumber;

   printf("What is your class number? ");
   scanf("%d", &classNumber);


   printf("Hello CMPUT %d!\n”, classNumber);
   return 0;
}
```

Sample C program:

Only few things are included by default in C. You need to include the libraries you will use

the main function is the main entry point to your program

this is a variable that can hold only integer values

```
#include <stdio.h>

int main(void) {
    int classNumber;

    printf("What is your class number? ");
    scanf("%d", &classNumber);


    printf("Hello CMPUT %d!\n”, classNumber);
    return 0;
}
```

Sample C program:

Only few things are included by default in C. You need to include the libraries you will use

```
#include <stdio.h>

int main(void) {
    int classNumber;

    printf("What is your class number? ");
    scanf("%d", &classNumber);


    printf("Hello CMPUT %d!\n”, classNumber);
    return 0;
}
```

this is a variable that can hold only integer values

the main function is the main entry point to your program

scanf is a library function that reads keyboard input

Sample C program:

Only few things are included by default in C. You need to include the libraries you will use

```
#include <stdio.h>

int main(void) {
    int classNumber;

    printf("What is your class number? ");
    scanf("%d", &classNumber);

    printf("Hello CMPUT %d!\n", classNumber);
    return 0;
}
```

this is a variable that can hold only integer values

the main function is the main entry point to your program

scanf is a library function that reads keyboard input

printf is a library function that prints output to the terminal

Sample C program:

Only few things are included by default in C. You need to include the libraries you will use

```
#include <stdio.h>

int main(void) {
    int classNumber;
    printf("What is your class number? ");
    scanf("%d", &classNumber);
    printf("Hello CMPUT %d!\n", classNumber);
    return 0;
}
```

the main function is the main entry point to your program

this is a variable that can hold only integer values

scanf is a library function that reads keyboard input

printf is a library function that prints output to the terminal

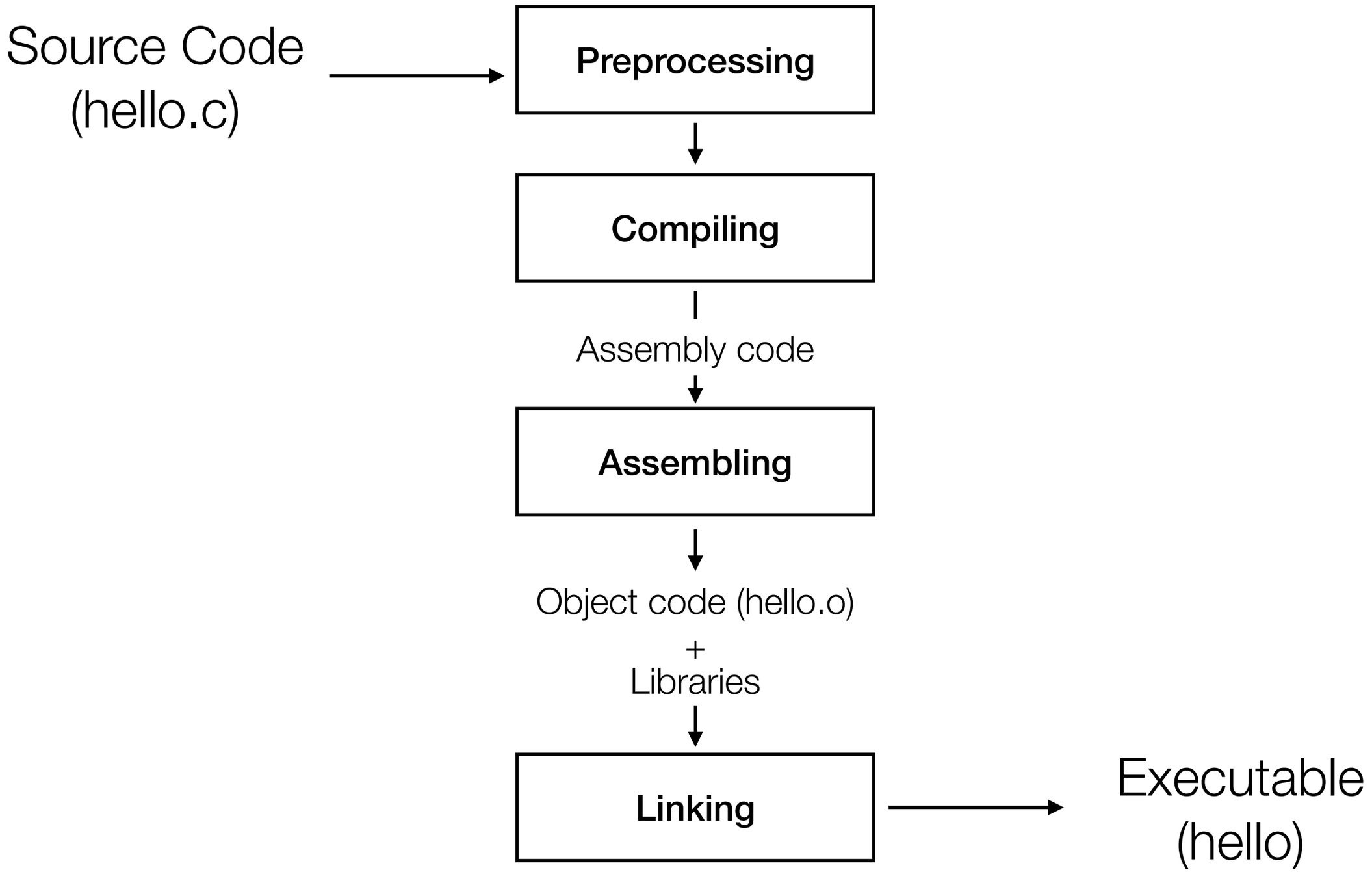
main must return 0 if it is successful

Compiling a C Program

- Wait, what does “compile” mean?
  - C is a *compiled* language, different from Python which is an *interpreted* language
  - With python, you used an interpreter that translates and executes one statement at a time. It would stop at the first error it meets.
  - With C, you use a compiler that translates the whole program into object code. After linking this object code, the program can be executed.
- We will use `gcc` with the following options to compile:
  - `gcc -Wall -std=c99 hello.c -o hello`
  - `-o` specifies the executable name, otherwise it’s a.out
  - `-Wall` enables all warnings and `-std=c99` specifies the C standard we will compile with

demo

Compiling a C Program (Behind the Scenes)



General Form of a C Program

```
/* directives */

/* global variables */

int main(void) {

  /* statements */

}
```

- Directives start with #
- Examples include:

```
// headers
#include <stdio.h>

// macros
#define PI 3.14
```

General Form of a C Program

```
/* directives */

/* global variables */

int main(void) {

 /* statements */

}
```

- Global variables are shared variables and can be accessed by any function in the file

- Example:

```
int counter = 1;
```

General Form of a C Program

```
/* directives */

/* global variables */

int main(void) {

 /* statements */

}
```

- Your program must have a main function
- This is the entry point to your program, i.e., the main function starts your program
- Upon successful completion, your main function should return a value of 0. This is called the status code.
- For now, our main function will not take any parameters (hence the void between parentheses)

General Form of a C Program

```
/* directives */

/* global variables */

int main(void) {

 /* statements */

}
```

- A function consists of a list of statements
- Examples:

```
/* declarations */
int classNumber, numberOfStudents;

/* assignments */
numberOfStudents = 30;

/* function calls */
scanf(“%d”, &classNumber)

/* function terminates, and
returns a value */
return 0;
```

General Form of a C Program

```
/* directives */

/* global variables */

int main(void) {

 /* statements */

}
```

In C, every variable must have a type that indicates the kind of data it will hold

- A function consists of a list of statements
- Examples:

```
/* declarations */
int classNumber, numberOfStudents;

/* assignments */
numberOfStudents = 30;

/* function calls */
scanf(“%d”, &classNumber)

/* function terminates, and
returns a value */
return 0;
```

Documenting & Formatting

- Use indentation to construct blocks
- Use blank lines to separate logical blocks of code
- Use /* … */ or // to add comments that explain the semantics of your code

```
/**********************************************/
/* Converts a Fahrenheit temperature to Celsius  */
/* Name: celsius.c                               */
/* Author: K. N. King                            */
/* January 7, 2016 ***************************/
```

Identifiers

- names for variables, functions, macros etc.
- must start with a letter or underscore
- case sensitive
- some keywords, such as `int` or `union` are reserved and cannot be used as identifiers (see Table 2.1 on page 26)
- try to use names that are self-explanatory and descriptive of the purpose of the variable, function, macro etc.

Version Control Systems

Version Control Systems

- record changes to a file over time
- allow you to keep track of the changes to your code and “go back” in time when you need to
- git is a widely used modern version control system
- Github (www.github.com) is an online project-hosting website that supports git
- a git cheat sheet is available at https://education.github.com/git-cheat-sheet-education.pdf
- there are LOTS of online resources on how to use git. Some of these are shared on eClass, including videos for this class (See CMPUT 201 Video Tutorials)

How will we use Git & Github in 201?

- You will have four **private** Github repositories: one for all lab exercises and then one for each assignment. All your lab and assignment solutions will be developed using Github.

- You will receive a GitHub classroom invitation link for each of these four repositories.

- When you go to the link, GitHub will create the corresponding repository for you. You **must** use these repositories for labs and assignments. (Note that the first time you click on any of the links we provide you, you will be asked to select your CCID, which will show up as an email address).

- When marking your assignments, we will simply pull the code on GitHub **at the time of deadline** and mark it. We will pull from the **master/main** branch.

- When marking lab exercises or during demos, the TA will pull the code from GitHub at the deadline

What Your Labs Repository Will Look Like

cmput201-w21 / **labs** Private

Watch 0 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Insights | Settings

No description, website, or topics provided. Edit

Manage topics

7 commits | 1 branch | 0 releases | 1 contributor | View license

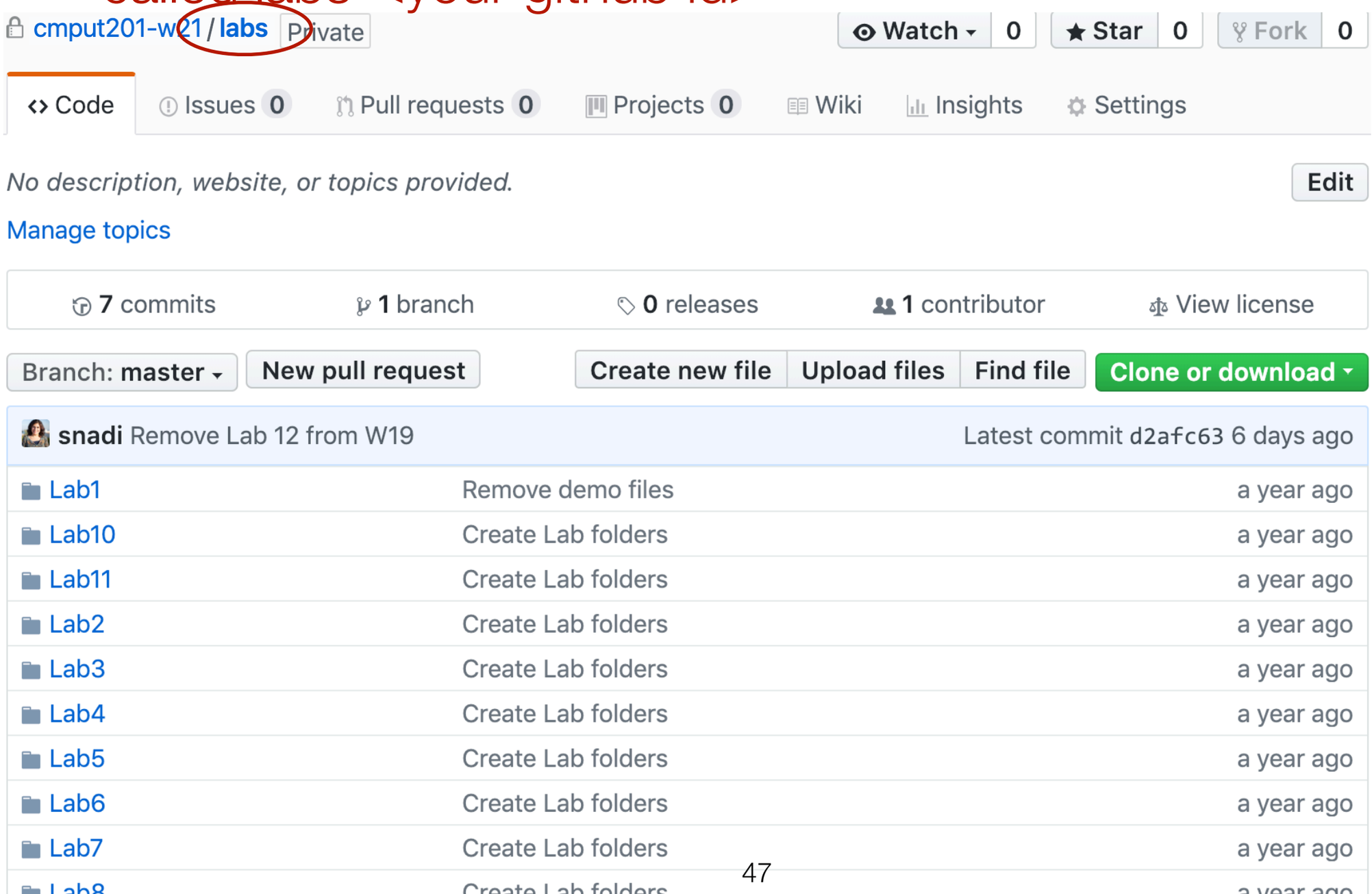
Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

snadi Remove Lab 12 from W19 — Latest commit `d2afc63` 6 days ago

Lab1	Remove demo files	a year ago
Lab10	Create Lab folders	a year ago
Lab11	Create Lab folders	a year ago
Lab2	Create Lab folders	a year ago
Lab3	Create Lab folders	a year ago
Lab4	Create Lab folders	a year ago
Lab5	Create Lab folders	a year ago
Lab6	Create Lab folders	a year ago
Lab7	Create Lab folders	a year ago
Lab8	Create Lab folders	a year ago

What Your Labs Repository Will Look Like

Your lab repo will be a copy of this and will be called labs-<your github id>



What an Assignment Repository Looks Like

cmput201-w21 / **assignment1** Private

Watch 0 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Insights | Settings

CMPUT 201 W19 Assignment 1. Read RepoStructure.md to understand the structure of this repo. **Edit**

Manage topics

7 commits | **1** branch | **0** releases | **1** contributor | View license

Branch: **master** | **New pull request** | **Create new file** | **Upload files** | **Find file** | **Clone or download**

snadi Add checking scripts for assignment — Latest commit `7091c75` 16 minutes ago

LocalTestScripts	Add checking scripts for assignment	16 minutes ago
images	Add images for documentation	27 minutes ago
sampletests	Add checking scripts for assignment	16 minutes ago
.travis.yml	Add checking scripts for assignment	16 minutes ago
CMPUT201_W19_Assignment1.pdf	Add checking scripts for assignment	16 minutes ago
LICENSE.md	Fix license	21 hours ago
RepoStructure.md	Add note about failing test cases	21 minutes ago
check_general_requirements_ci.sh	Add checking scripts for assignment	16 minutes ago
test_program.sh	Add checking scripts for assignment	16 minutes ago

What an Assignment Repository Looks Like

Your assignment1 repo will be a copy of this and will be called assignment1-<your github id>

cmput201-w21 / **assignment1** Private

Watch 0 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Insights | Settings

CMPUT 201 W19 Assignment 1. Read RepoStructure.md to understand the structure of this repo. **Edit**

Manage topics

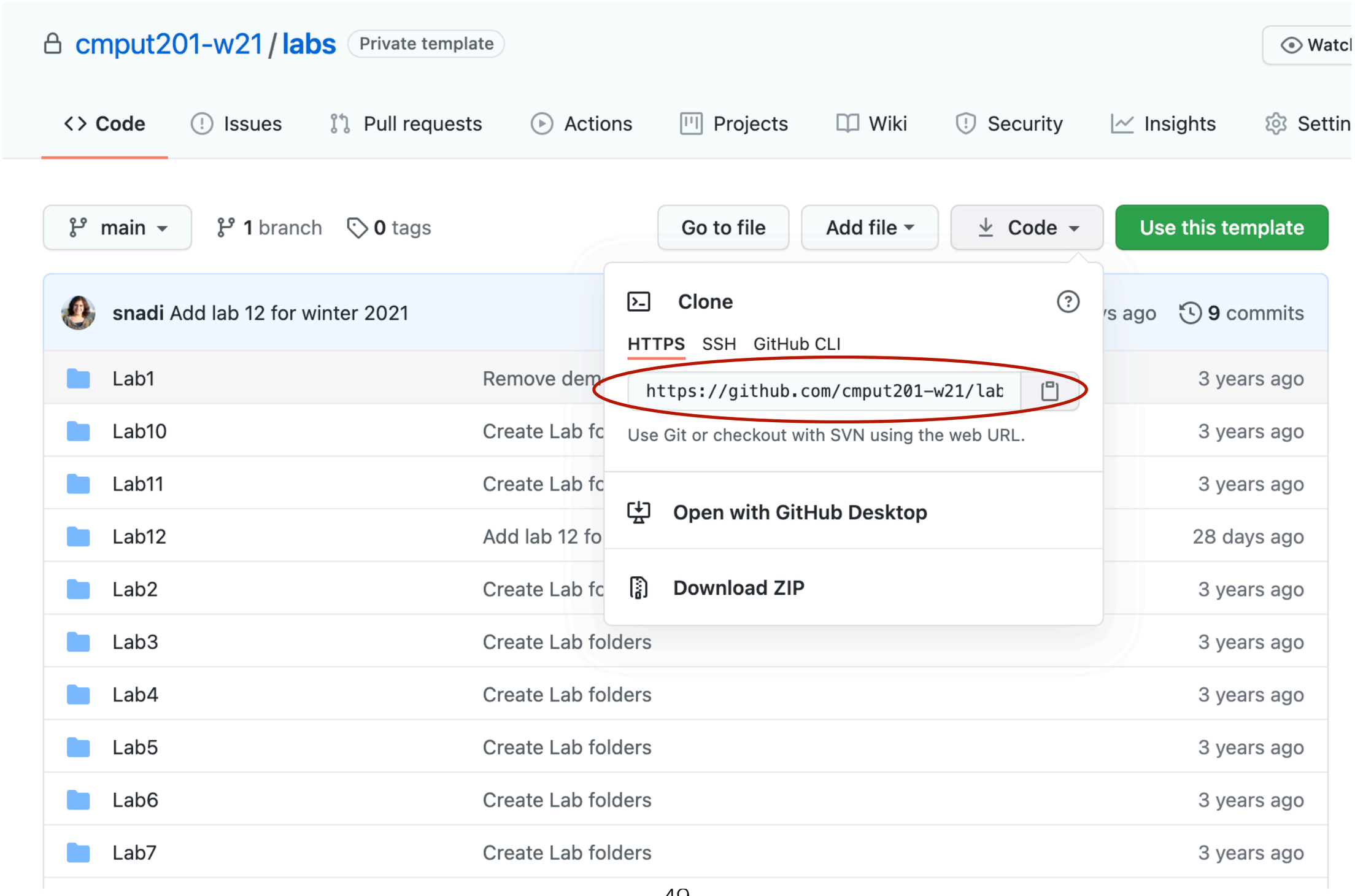
7 commits | **1** branch | **0** releases | **1** contributor | View license

Branch: **master** | **New pull request** | **Create new file** | **Upload files** | **Find file** | **Clone or download**

snadi Add checking scripts for assignment — Latest commit `7091c75` 16 minutes ago

LocalTestScripts	Add checking scripts for assignment	16 minutes ago
images	Add images for documentation	27 minutes ago
sampletests	Add checking scripts for assignment	16 minutes ago
.travis.yml	Add checking scripts for assignment	16 minutes ago
CMPUT201_W19_Assignment1.pdf	Add checking scripts for assignment	16 minutes ago
LICENSE.md	Fix license	21 hours ago
RepoStructure.md	Add note about failing test cases	21 minutes ago
check_general_requirements_ci.sh	Add checking scripts for assignment	16 minutes ago
test_program.sh	Add checking scripts for assignment	16 minutes ago

Your Repository's Address



Example of Using git

- Clone repository (need to do only the first time you use the repo on a given computer):

```
nadi@ug12:~>git clone git@github.com:cmput201-w21/labs.git
Cloning into 'labs'...
```

- Go to the directory that now contains your repository. You should see the same files you see on GitHub.

```
nadi@ug12:~>cd labs/
nadi@ug12:~/labs>ls
LICENSE.md  Lab10  Lab2  Lab4  Lab6  Lab8  README.md
Lab1        Lab11  Lab3  Lab5  Lab7  Lab9
```

- Add/Modify a file and then commit this file, including a commit message. You then need to push this file to GitHub.

```
nadi@ug12:~/labs>cd Lab1
nadi@ug12:~/labs/Lab1>echo "This is a test file" > TestFile
nadi@ug12:~/labs/Lab1>ls
README.md  TestFile
nadi@ug12:~/labs/Lab1>git add TestFile
nadi@ug12:~/labs/Lab1>git commit TestFile -m "Add Test file for 201 demo"
[master 6b8c10c] Add Test file for 201 demo
 1 file changed, 1 insertion(+)
 create mode 100644 Lab1/TestFile
nadi@ug12:~/labs/Lab1>git push
```

demo

Continuous Integration

Continuous Integration (CI)

- A practice where developers automatically build and test their code in response to every change pushed to the repository

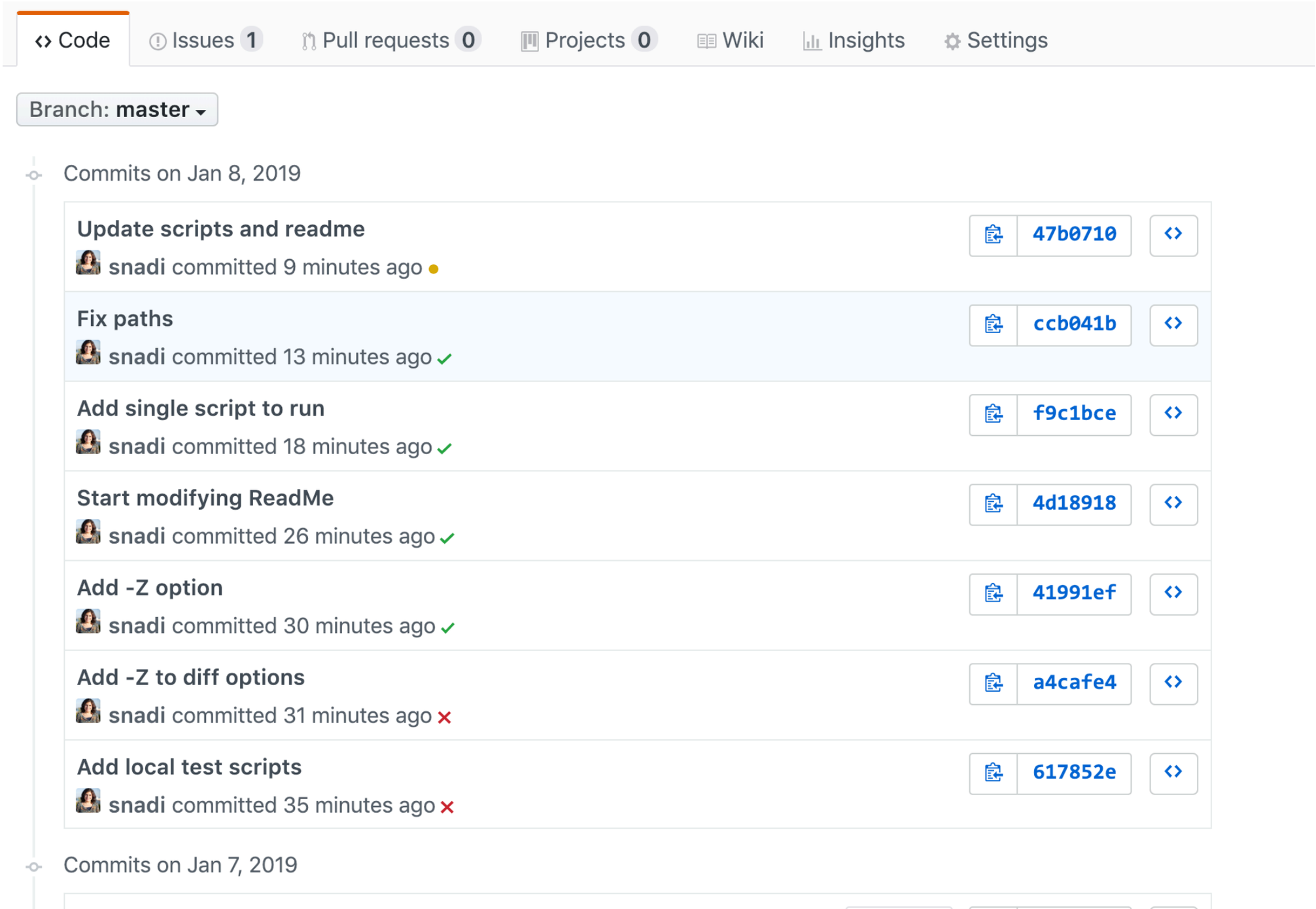
- Helps catch problems early

- *“Continuous integration doesn’t get rid of bugs, but it does make them dramatically easier to find and remove”* — Martin Fowler, Chief Scientist, ThoughtWorks

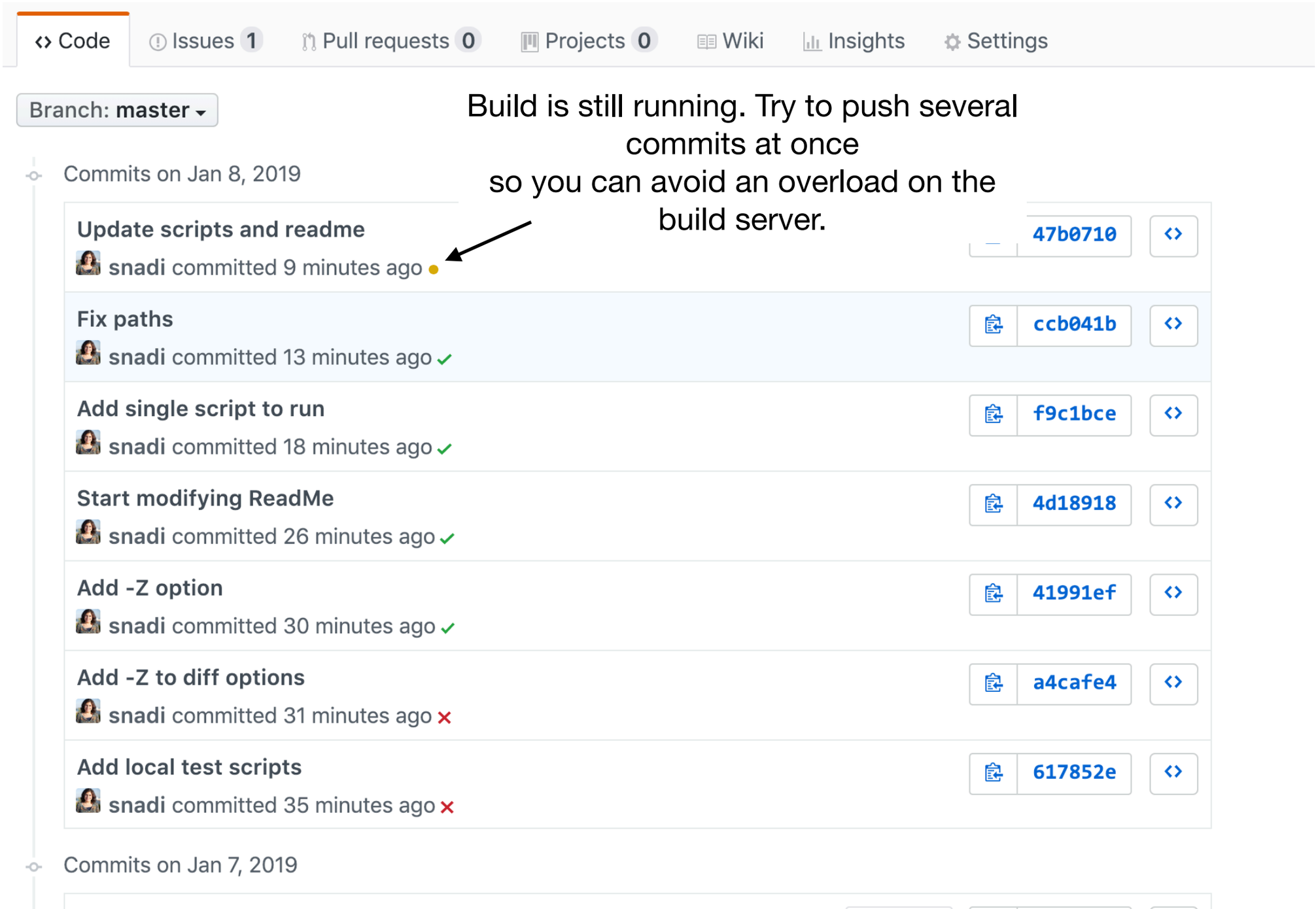
CI in CMPUT 201

- You don’t need to do anything yourself related to CI
- You don’t even need to understand what it is that much
- … BUT you need to be able to interpret its results
- We are integrating CI into your **assignment repos** to help you avoid silly mistakes that may result in a 0 in your assignment
- We will be using GitHub actions

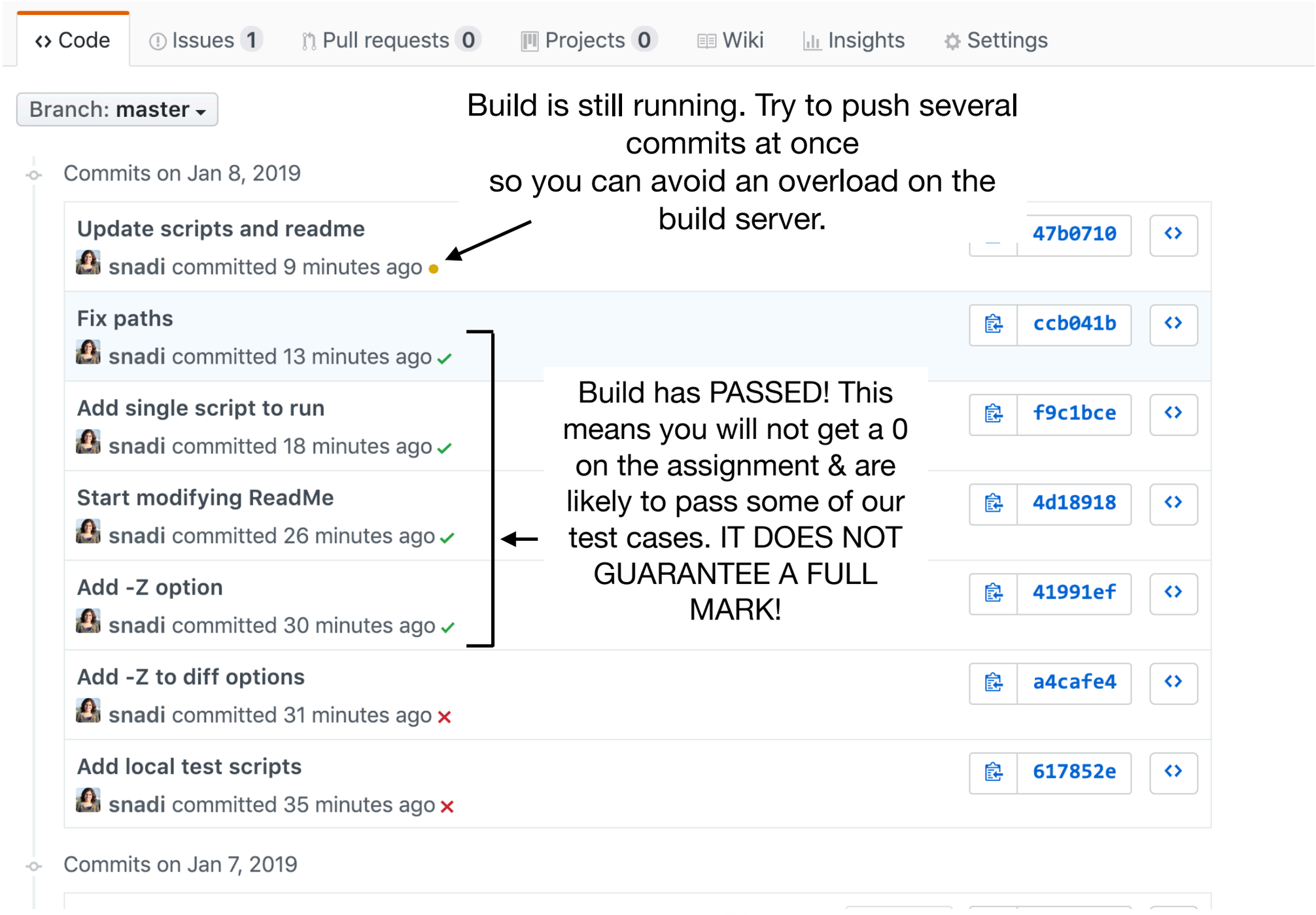
Checking CI Build Status



Checking CI Build Status



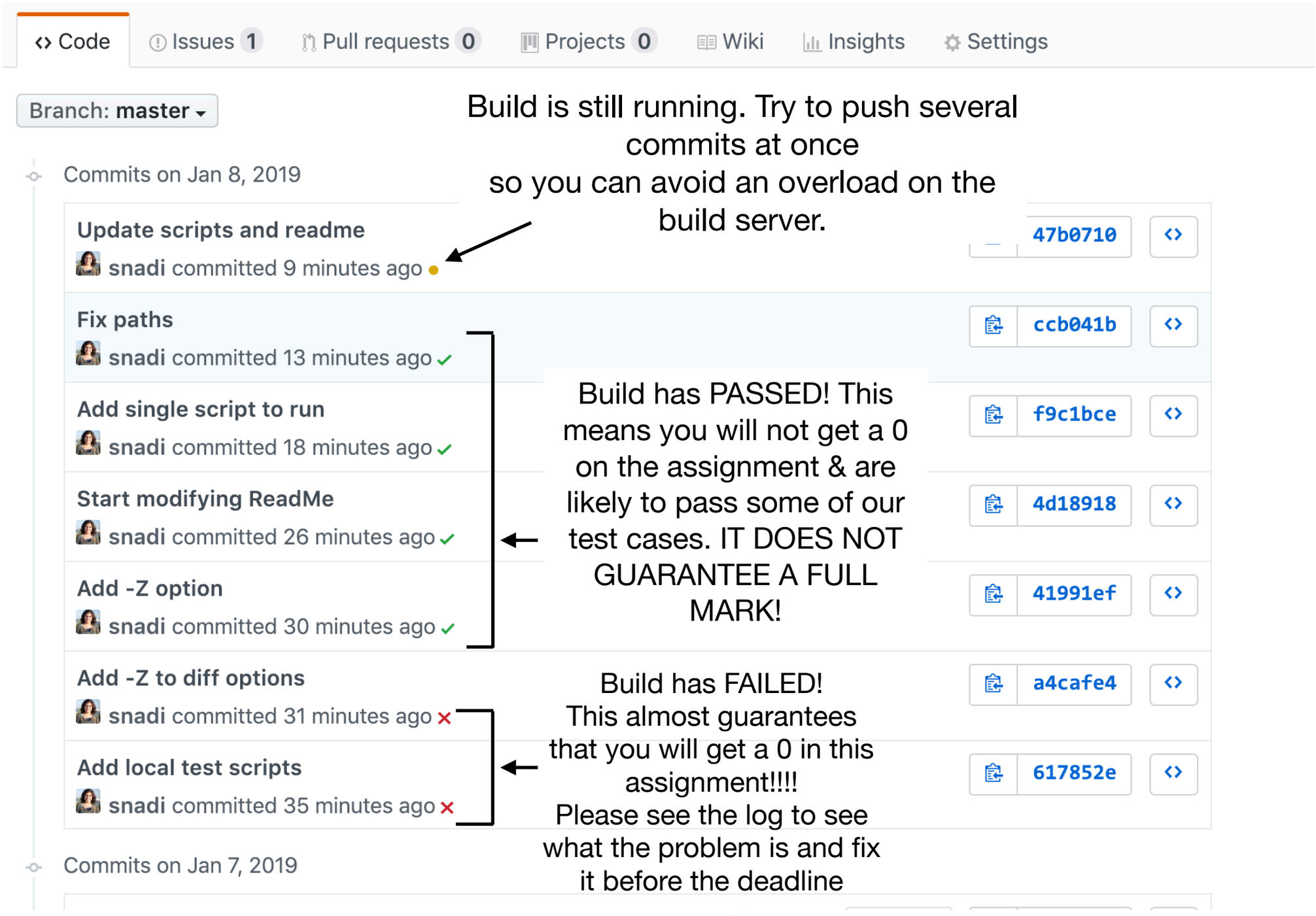
Checking CI Build Status



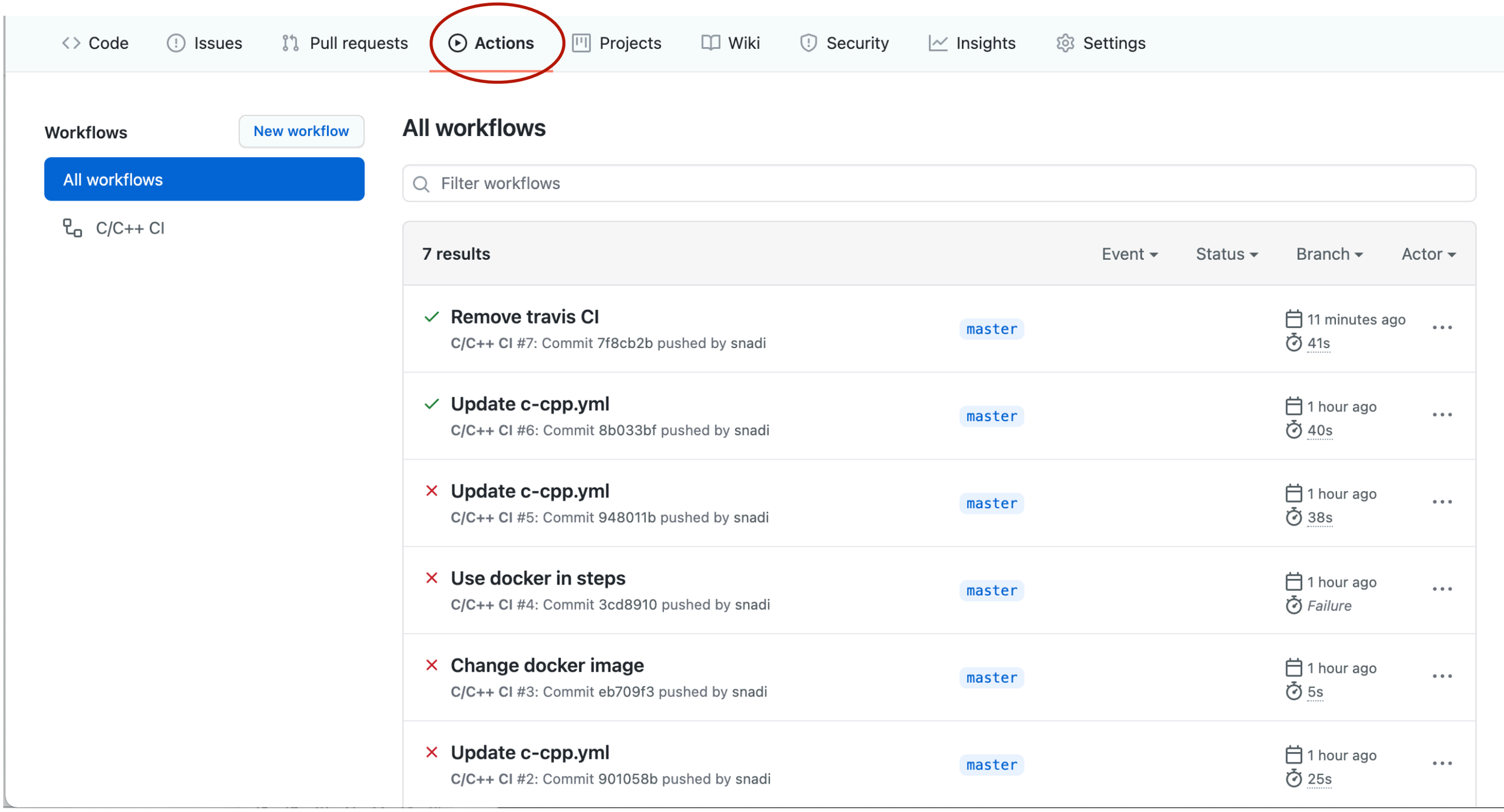
Build is still running. Try to push several commits at once so you can avoid an overload on the build server.

Build has PASSED! This means you will not get a 0 on the assignment & are likely to pass some of our test cases. IT DOES NOT GUARANTEE A FULL MARK!

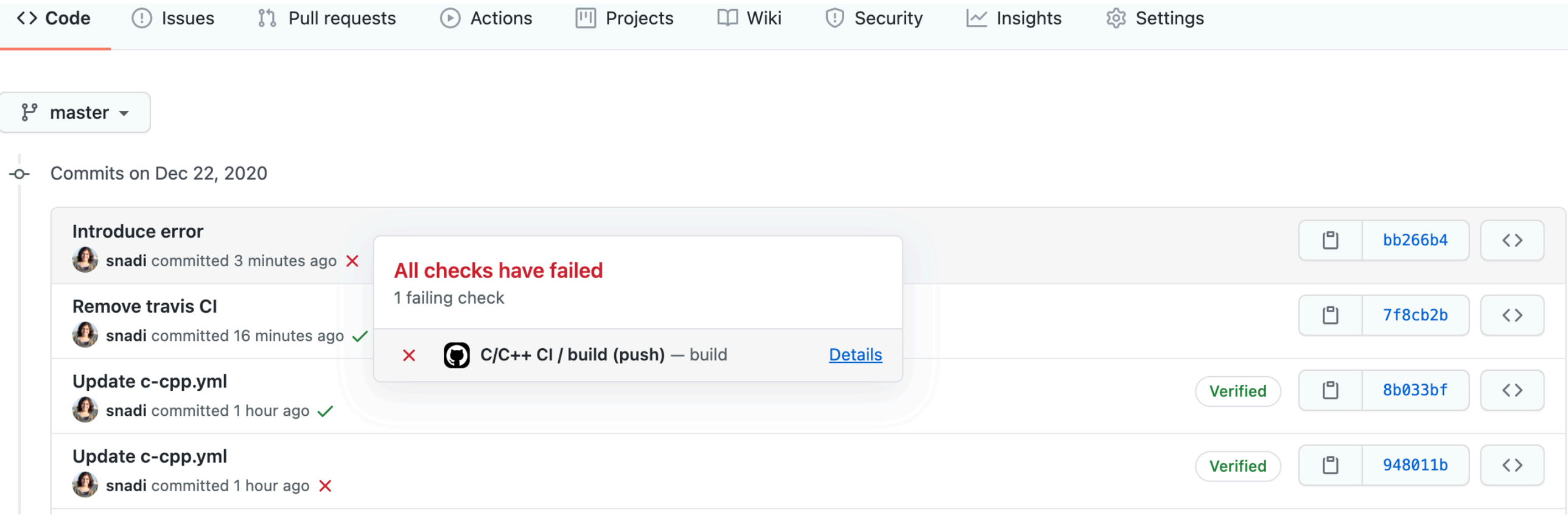
Checking CI Build Status



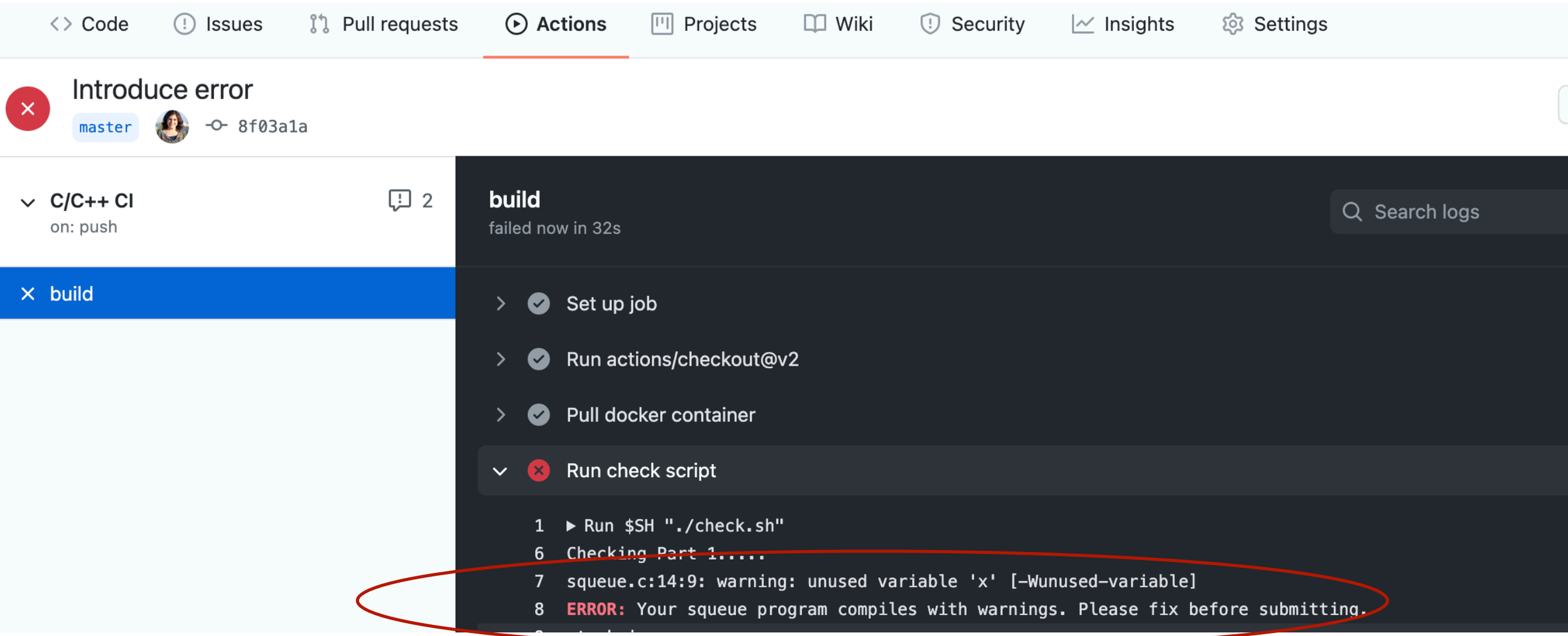
Checking CI Build Status



Understanding the “Red X”



Understanding The “Red x”



This explains what the problem is. Here, your code compiles with warnings because of an unused variable. You need to fix your code to remove this unused variable!

Your TODOs this week

- Do the plagiarism quiz by the end of the week
- Before the labs start, make sure you can ssh into the lab machines. Contact IST if you cannot.
- Create a Github account if you don't have one. Go to the GitHub labs invitation link: https://classroom.github.com/a/kt8tclrJ and link your GitHub account to your CCID/email
- Read Lab #1 tutorials, watch the videos, and try to finish the lab exercise
- Read all the course information posted on eClass