# Git for csci labs/projects

- Most of our csci courses use a custom git setup to distribute and collect labs, projects, and assignments
- Students enrolled in a course will automatically be given access to the appropriate git repositories (edit: if you register in a class after the first day of the semester it may take a few days for your access permissions to be updated)
- Here we'll walk through the steps you'll need to follow to obtain, work on, and submit your labs using our git process

## What is git?

- Git is version control software, allowing developers to keep track of changes to projects involving multiple versions of multiple files over an extended period of time
- The developer is thus able to examine what changed and when, “roll back” to specific versions, create different branches/versions of the project, etc
- e.g. suppose you have to work on a project with many files, multiple versions (1.0, 1.1, 2.0, 2.01, 2.1, etc etc) on a variety of platforms ... somehow you have to be able to easily identify the right version of the right file for the right platform
- Projects are organized into repositories – basically just a directory containing all the files and subdirectories for the project, but with some extra git control and tracking files hidden in there somewhere

## The csci git server

- The instructor will post work, and you will submit work, through a special csci git server, with the general idea as follows:
  - The instructor posts the initial version of the project as a git repository on the csci git server
  - The student makes their own copy of the repository also on the csci git server (see the “fork” step later)
  - The student now copies (clones) from their copy to their working space on otter, where they make changes (see edit/add/commit later)
  - The student then “pushes” their changes back to the csci git server, where the instructor can access it for marking later

## Making your server-side copy (fork)

- To initially create your own copy of the original, you need to know the course id (e.g. csci160, csci265, etc) and the name of the repository (e.g. lab1, project, assign7, etc)
- The instructor will let you know what it is called, but you can also get a list of the available repositories using:

```
ssh csci info
```

- If the course was csci265 and the repo was lab1, the fork command would then be

```
ssh -x csci fork csci265/lab1 csci265/$USER/lab1
```

  Note the command is saying copy from csci265/lab1 to csci265/YOURNAME/lab1, the $USER variable automatically expands with your userid

- Be careful with the command, it's easy to make typos!  If something goes wrong, see the following URL for troubleshooting:

  csci.viu.ca/~wesselsd/guides/gitstudent.html

## Cloning to otter to work on it

- You can’t work on the files directly on the git server, you need to clone them to otter, work on them their, and send (push) the changes back
- Use cd to go to whereever you want your local copy to exist on otter, then use the following (again here written for course csci265 and repository lab1):

```
git clone csci:csci265/$USER/lab1
```

- See the url from the previous slide for troubleshooting

## Updating your copy on otter

- Make whatever changes you want in your copy on otter
- When you’re happy with the state of a file, use the following command to “add” the updated version

  **git add *filename***

- When you’re happy with all of the added files, you can create a “commit point”, basically a saved state of the project:

  **git commit -m “*describe what has changed*”**

- If you forget to add or commit, those changes ***won’t*** get included in the next step

## Sending the changes to the server

- To copy your latest commited version back to the git server, where the instructor can access it, use

  **git push**

- If you forget to push then the instructor won’t see your changes
- Extra adds, commits, or pushes aren’t a problem: forgetting any one of them is!

## Checking your repo status

- You can check the status of your repository using

  **git status**

- “up to date with origin/master” is good news

**[Note: hopefully git will soon drop the “master” term for something like “primary”]**

- “ahead of origin/master by N commits” means you need to push
- “changes to be commited” means you need to commit and push
- “untracked files” means you need to add, commit, and push
- Additional links and info are available at csci.viu.ca/~wesselsd/guides/gitstudent.html