CIS*1300 Lab 4: Git

Introduction to Programming

Git

- Git is a widely used version control system for software development.

Git allows teams of software developers to work on a project or even a single file together.

Git makes it (relatively) easy to track changes across a project or file, and to revert back to a previous version if a change is undesirable (or buggy!).

Great Resource: Pro Git book, written by Scott Chacon and Ben Straub

Git

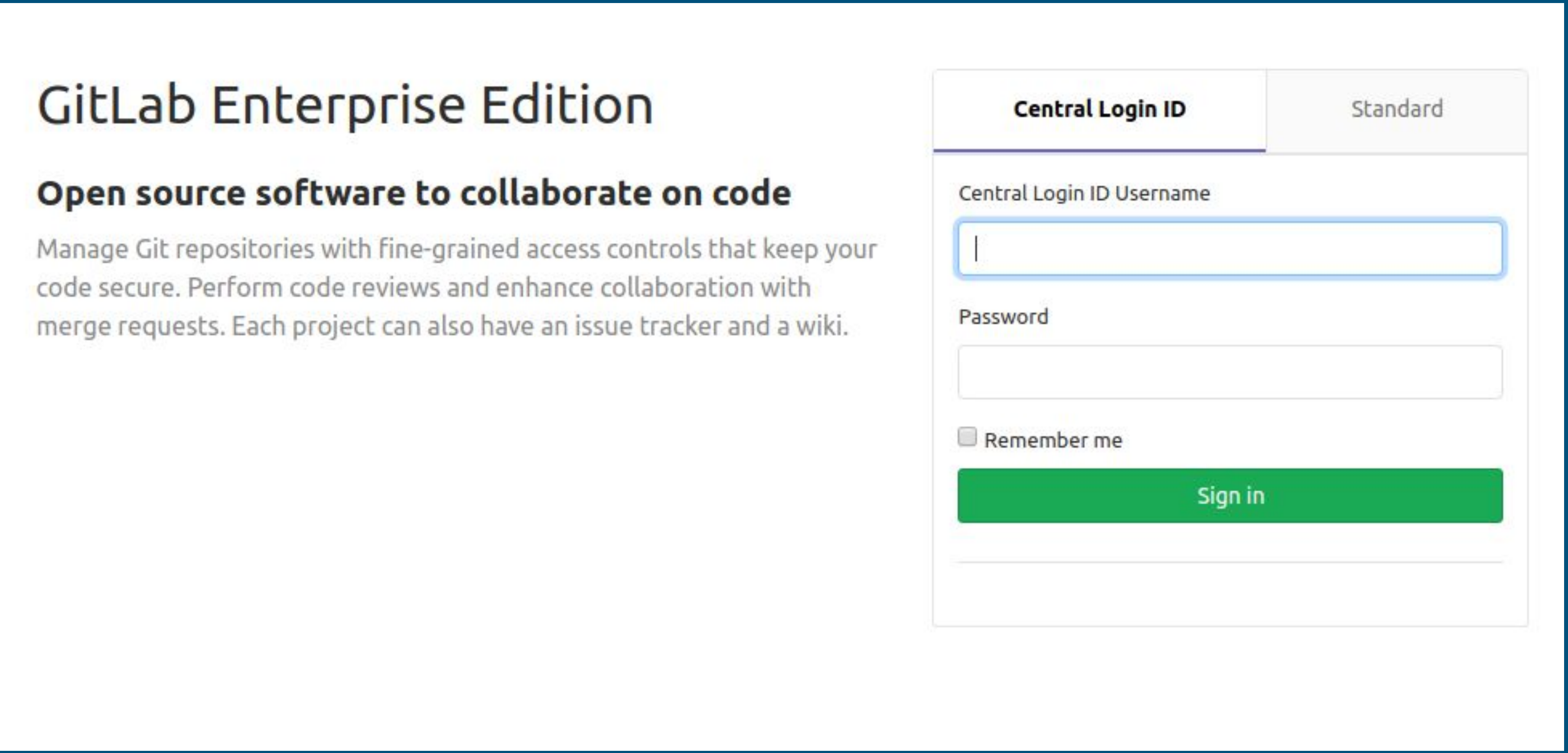
- The University of Guelph hosts a Git server for you!
    - https://gitlab.socs.uoguelph.ca
- You can take advantage of this to help you with your course work.
- Use Git to have a remote backup of your project, and to make updates across multiple devices.
- Use Git to make incremental changes to your projects and to recover from hard-to-fix bugs.

Advantages of using Git

- No more having to scp your files onto the server
- No more having email copies to yourself
- No more accidentally deleting your whole code base
- No more creating multiple copies of your file just in case!

- Collaboration!

Gitlab: https://gitlab.socs.uoguelph.ca



GitLab Enterprise Edition

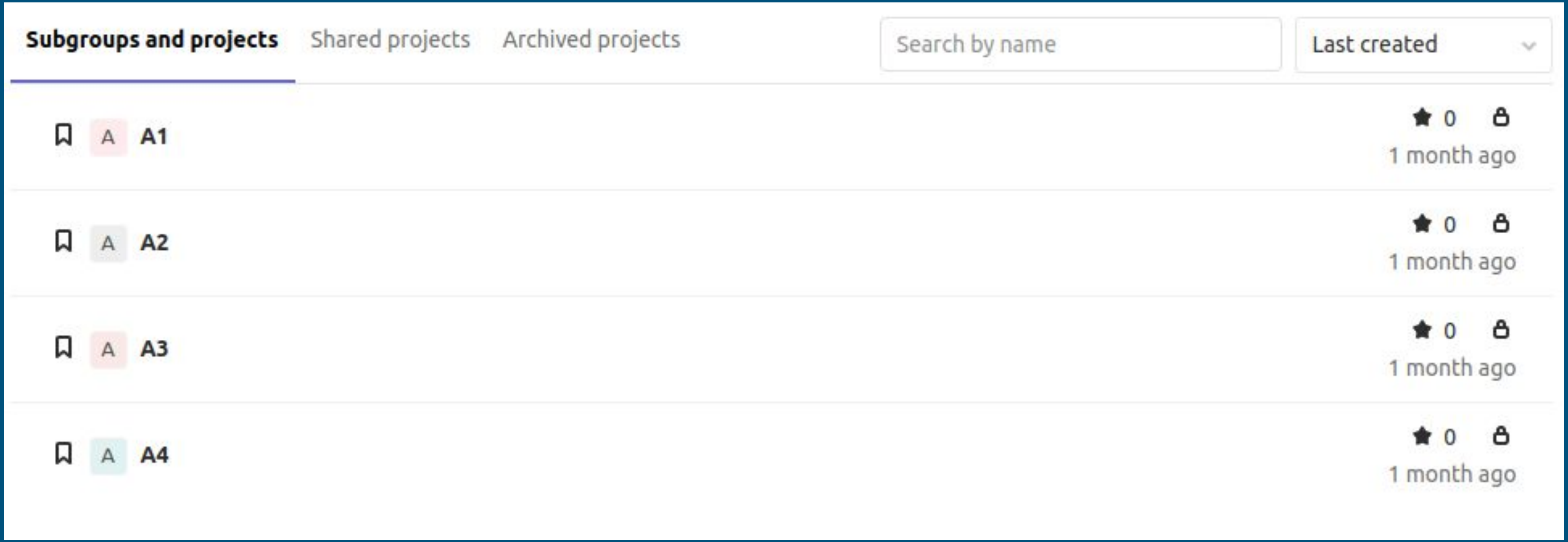
Open source software to collaborate on code

Manage Git repositories with fine-grained access controls that keep your code secure. Perform code reviews and enhance collaboration with merge requests. Each project can also have an issue tracker and a wiki.

Log in with your username and password

Your Repos

Once you login you should see 4 folders, feel free to explore gitlab



A	A1	0 · 1 month ago
A	A2	0 · 1 month ago
A	A3	0 · 1 month ago
A	A4	0 · 1 month ago

Gitlab

- Gitlab has a lot of cool features for collaborating and project managing
  - Issue tracker
  - Time Tracking
  - Milestones
  - CI/CD: built-in Continuous Integration, Delivery, and Deployment tool
  - GitLab Pages
  - Wiki
  - Snippets: Similar to gist.github.com
  - … a ton more explore it!

git config

- Before you can begin using Git, you must tell Git who you are.

- Open a new terminal window and type the following two commands:

Replace *FirstName* and *LastName* with your first and last name, and replace *username*@uoguelph.ca with your UofG email address:

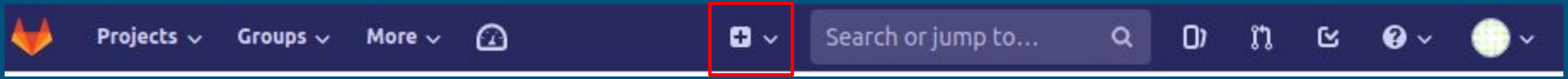
- $ git config --global user.name “*FirstName LastName*”
- $ git config –-global user.email “*username*@uoguelph.ca”

Git

- A **git repository** is a place for you to store your files.
  - It is a folder containing any number of files and other folders within it.

At any time, you can create a **personal Git repository** on the school's server by going to: https://gitlab.socs.uoguelph.ca/projects/new

or:



Setting up ssh keys (Optional)

- SSH keys allow you to establish a secure connection between your computer and GitLab.
- https://gitlab.socs.uoguelph.ca/profile/keys
- If you have one already:
  - `$ cat ~/.ssh/id_rsa.pub`
  - Paste the output into the text area
- Don’t have one:
  - `$ ssh-keygen -o -t rsa -b 4096 -C "USERNAME@uoguelph.ca"`
  - `$ cat ~/.ssh/id_rsa.pub`
- More info:
  - https://gitlab.socs.uoguelph.ca/help/ssh/README#generating-a-new-ssh-key-pair

Setting up your repo

- Go into your CIS1300 directory (Server or your machine where ever you have your most up to date code.)

```
$ cd {Assignment}
$ git init
$ git remote add origin https://gitlab.socs.uoguelph.ca/1300F19/{USERNAME}/{REPO}.git
$ git add .
$ git commit -m "Initial commit"
$ git push -u origin master
```

Assignment = {Assignment1, Assignment2, Assignment3, Assignment4}
REPO = {A1, A2, A3, A4}

You need to do this for each of the assignments

Setting up your repo...

Setting up A1 repository

```
$ cd Assignment1/
$ git init
$ git remote add origin https://gitlab.socs.uoguelph.ca/1300F19/{USERNAME}/{REPO}.git
$ git add .
$ git commit -m "Initial commit"
$ git push -u origin master
```

repeat this for A2-A4, replacing Assignment1 with the appropriate name, remember to also be in the correct folder.

- If you refresh gitlab.socs.uoguelph.ca, you can check out your changes on the web

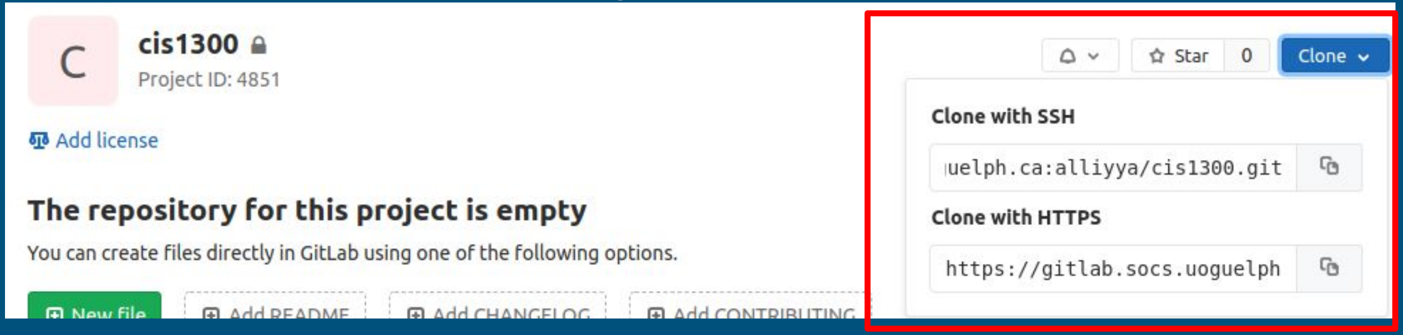
Git Clone

To create a local copy of one of these repositories, type

$ git clone **{REPO_URL}**

After doing this command, a **clone** of the remote repository is created locally on your machine as a folder in the current directory.

You can do this on multiple machines, servers.



cis1300

Project ID: 4851

Add license

The repository for this project is empty

You can create files directly in GitLab using one of the following options.

Clone with SSH

`uelph.ca:alliyya/cis1300.git`

Clone with HTTPS

`https://gitlab.socs.uoguelph`

You can find the REPO URL here:
Clone with HTTPS

Ex.

$ git clone https://gitlab.socs.uoguelph.ca/1300F19/**{USERNAME}**/**A1**.git

Make a copy on the server (**git clone**)

```
$ ssh {USERNAME}@linux.socs.uoguelph.ca
$ cd CIS1300
$ git clone https://gitlab.socs.uoguelph.ca/1300F19/{USERNAME}/{REPO}.git
REPO = {A1, A2, A3, A4}
```

Important to do this so that you can test on the server!

git add, git rm and git commit

- In order to add files to the remote Git server, you must package them into something called a **commit**.

- A **commit** is a collection of changes to the contents of a repository.

This will mostly include adding, modifying and removing files and directories.

- If you have created a new file, you must use the **git add** command.

$ git add *fileNameOrDirectoryName*

Ex: $ git add helloWorld.c

- If you want to remove a file from your repository, you must use the **git rm** command.

git rm *fileNameOrDirectoryName*

Ex: git rm helloWorld.c

What files to add?

- Your source code
- Makefile
- Readme.md

- Don’t add your .o files or executables!

- General rule of thumb: Don’t add anything that can be generated
- Pro tip: use a **.gitignore** file!

.gitignore

- a hidden file that includes a set of rules that tells git not to track these files

General C .gitignore template

This file belongs in each of your repositories.

git commit

- Once you have made all of the changes you wanted to, you must finalize the commit. This is done using the **git commit** command, but there are a couple things to keep in mind.

Use the **-m** flag to include a message in the commit. This should be informative of the changes you have made. If you do not include this flag, nano will open up and you will create the message using it.

Use the **-a** flag to include any changes you have made to files you have previously added to a commit. This is a way of automatically adding changes to older files to the commit.

$ git commit -a -m "*your message*"

or: $ git commit -am "*your message*"

- Ex:$ git commit -a -m "modified calculator program"

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
AAAAAAAA	3 HOURS AGO
ADKFJSLKDFJSDKLFJ	3 HOURS AGO
MY HANDS ARE TYPING WORDS	2 HOURS AGO
HAAAAAAAANDS	2 HOURS AGO

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

Title text: Merge branch 'asdfasjkfdlas/alkdjf' into sdkjfls-final

git push

- After you have successfully run the **commit** command, you will have a complete commit on your local machine.

- You must now **push** your commit to the remote server with the **git push** command.

- Fortunately, you should not need any flags or arguments to this command. Just simply type:

```
$ git push
```

- You will be asked again for your **password**.

git pull

- this is used to update your repository to **pull** changes from your remote repository
- Use case: You work on your local machine and added, committed and pushed your changes to git. You want to test on the server. The repo is already cloned on the server. If you log onto the server and cd into your repo and git pull. It will update on the server.

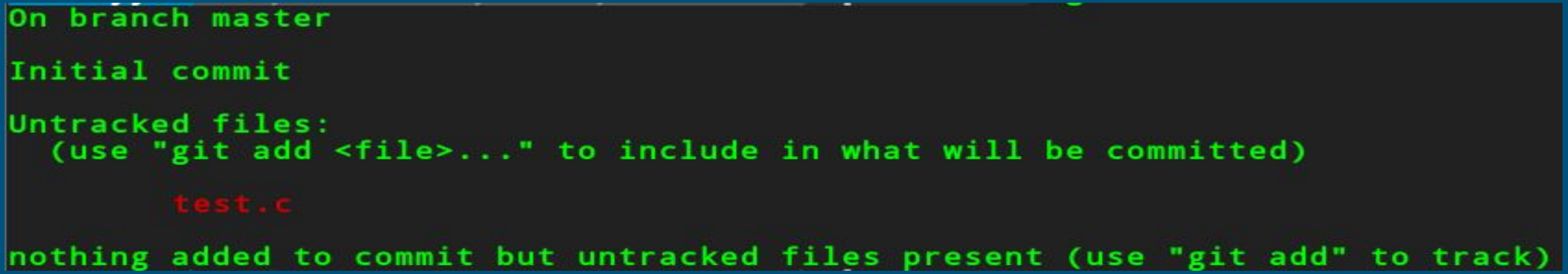
```
$ git pull
```

Remember to pull before you make modifications or test your code to ensure you're working with the right version of your code!

git status

- **$ git status**
- Will allow us to see the status of our repo, do we have new changes to commit?
- If you had a new file and didn’t add/commit it would look something like this

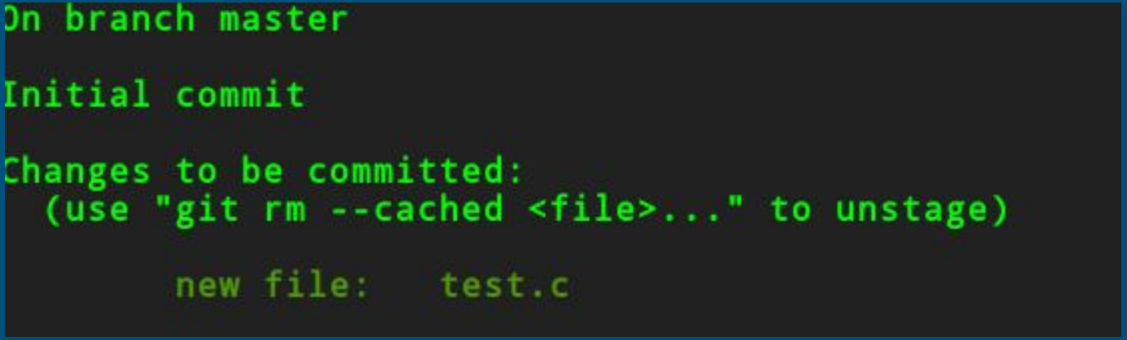
$git status



```
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        test.c

nothing added to commit but untracked files present (use "git add" to track)
```

$git add test.c

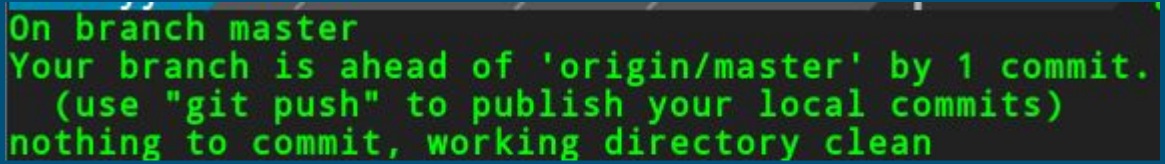
$git status



```
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   test.c
```

$git commit -m “initial commit”

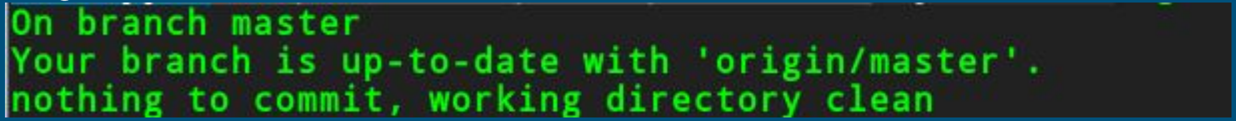
$git status



```
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)
nothing to commit, working directory clean
```

$git push

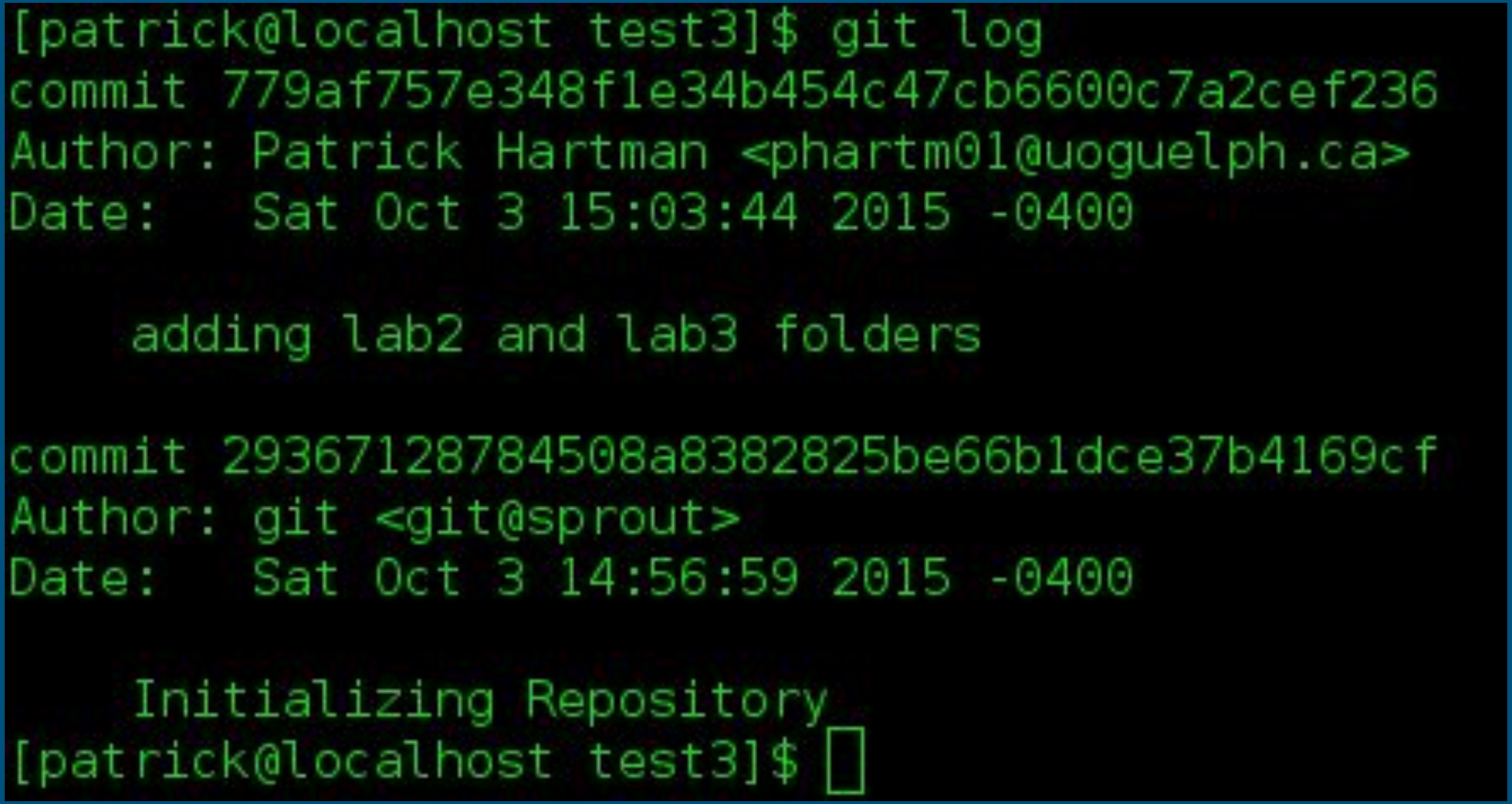
$git status



```
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean
```

git log

- In order to see a record of our commits, we can use **git log**.
- Type **git log** now.



```
[patrick@localhost test3]$ git log
commit 779af757e348f1e34b454c47cb6600c7a2cef236
Author: Patrick Hartman <phartm01@uoguelph.ca>
Date:   Sat Oct 3 15:03:44 2015 -0400

    adding lab2 and lab3 folders

commit 2936712878450 8a8382825be66b1dce37b4169cf
Author: git <git@sprout>
Date:   Sat Oct 3 14:56:59 2015 -0400

    Initializing Repository
[patrick@localhost test3]$ 
```

git add file

- Add your modified files

git commit -m "message"

- include a message that describes your changes.
- You typically want to commit when you have workable code or at a stopping point.

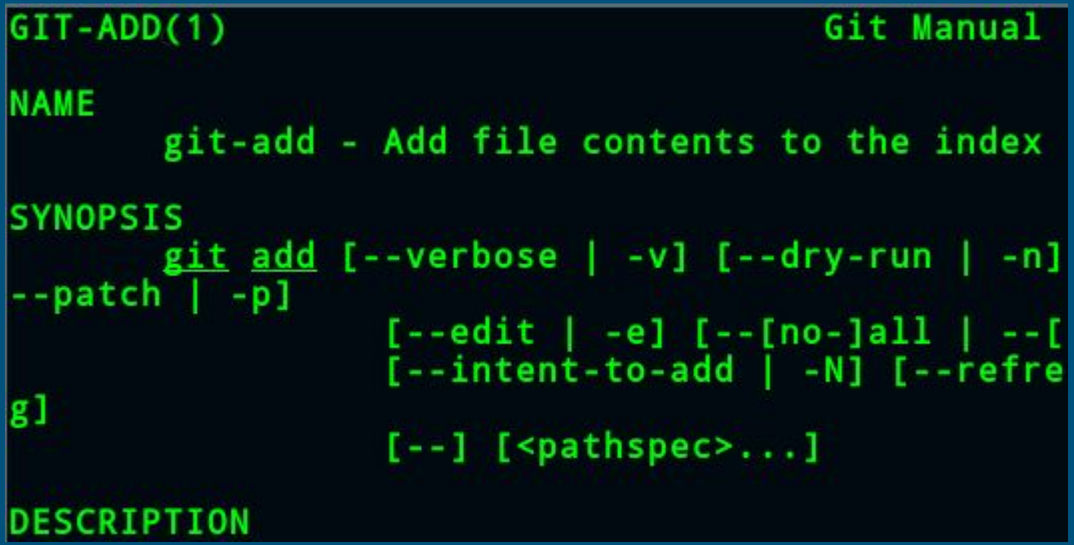
git push

- push your commits remotely

Remember you can use git status at anytime to see what's been changed and needs to be pushed.

Man pages are your friend!

- man git
  - see how many more git commands there are
- man git add



```
GIT-ADD(1)                                          Git Manual

NAME
       git-add - Add file contents to the index

SYNOPSIS
       git add [--verbose | -v] [--dry-run | -n]
--patch | -p]
                 [--edit | -e] [--[no-]all | --[
                 [--intent-to-add | -N] [--refre
g]
                 [--] [<pathspec>...]

DESCRIPTION
```

Short list of more git commands

- git config
- git init
- git clone
- git add
- git commit
- git diff
- git reset
- git status
- git rm
- git log

- git merge
- git remote
- git push
- git pull
- git stash
- git show
- git tag
- git branch
- git checkout

Check them out in the man pages or google

These commands all have a variety of arguments

Practice with git

- Create a new private repository named Labs to hold your labs
    - make sure to init with your files like you did previously
    - add a .gitignore file
    - clone it on the linux server
    - create a hello world program
    - add to git

Git challenges

- How would you undo a commit?
    - How would you undo 3 commits
- How do you use git to work with someone else?
- What’s a branching for?
- What are git aliases
- Find a project on gitlab/github and clone it and run it

Extra Resources

- https://try.github.io: Curated list by github of tools/reading
  - http://git-school.github.io/visualizing-git/: Visual tool to play with
  - https://github.github.com/training-kit/downloads/github-git-cheat-sheet.pdf: Cheatsheet
  - https://learngitbranching.js.org: Practice branching
- Pro Git book, written by Scott Chacon and Ben Straub

In case of fire

1. git add
2. git commit -m “Fire”
3. git push
4. exit building