GITLAB TUTORIAL: CIS*2500

WHAT IS GIT AND GITLAB?

→ Git and GitLab are two different things that integrate well together. GitLab is just a website, you can think of it as an add-on to Git and it gives us a nice UI, additional features and lets us host remote repositories. Git is a software which is responsible for handling all the version control things such as commits, merging, pulling, pushing, etc.

USING GITLAB FOR THIS COURSE:

→ You are required to submit all your labs and assignments to GitLab for this course, and this document was developed to aid you in using GitLab correctly. The concepts that will be discussed in this document include ***git cloning, git committing, git adding*** and ***git pushing*** all of which will be required for you to submit things for this course.

- Cloning – it enables you to bring a working repository that is hosted on some platform such as GitLab/GitHub into a folder on your local machine. It essentially gives you a copy of what is in the repository.
- Git Adding – is a tool/command of git that enables you to add files/folders from your local machine into a *staging area* waiting to be committed. A staging area is basically an area where you are temporarily adding/removing files that you want to be in the next commit
- Commits – commits are simply changes you have made to your code. You may have made a bunch of changes to your project/code on your local machine and then you want to "commit" those changes, meaning that you want to save your changes (in a bundle or package) with a time stamp which you now can reference at any time. You can think of it as a checkpoint. Commits should be used for important changes in your code, this can include you fixed a bug issue, you developed a new function or something significant for your project.
- Git Pushing – is a tool/command of git that enables you to upload your git commits to a remote repository that is hosted on GitLab/GitHub

GITLAB SUBMISSION INSTRUCTIONS FOR LABS & ASSIGNMENTS

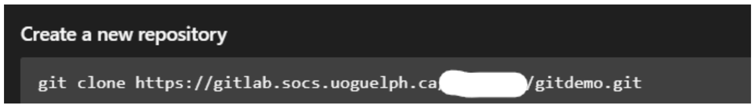
→ Below is this list of steps you need to follow to submit your labs and assignments for this course. Follow these instructions in order and your submission process should go smoothly.

Step 1:

- Ensure you have a GitLab account associated with your UofG email and that you have been added to the "CIS2500 W23" group. If you have any issue with this, please contact the course email (cis2500@socs.uoguelph.ca) as soon as possible.
- Depending on which lab/assignment you are working on, open the corresponding repository on GitLab and have it open on the side. Then login into the school server and navigate to folder you want your work to be stored or open a folder on your local computer and have a terminal open.
    - Note: if you are working on your own computer, ensure you have Git installed on it. If you require steps on how to install Git, follow this tutorial link.

Step 2:

- After you are in your desired folder, you will use the git clone command followed by your SSH or HTTPS key to bring down the existing repository into your local machine. On an empty repository you will find the git clone command listed in a dialogue box (refer to Figure 1) which you can copy into your terminal, or if there are existing files in the repo you can click on the "clone" button highlighted in blue (refer to Figure 2) and copy the key. In our case, we will use the HTTPS key to clone our repository.



Create a new repository

```
git clone https://gitlab.socs.uoguelph.ca[redacted]/gitdemo.git
```

Figure 1

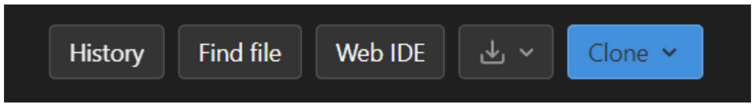


Figure 2

- In your terminal type

```
git clone https://gitlab.socs.uoguelph.ca/your username/demo.git
```

you will see that a new folder with the name of the repo has been created. Now you can modify existing files in the repo, add new files and pretty much do anything you need inside the copy you just made.

Step 3:

- After you have made changes in this folder such as modifying content in an existing repo file, and you want to "push" your changes back to the remote repository you will need to execute the following instructions (Note: if you added additional files, refer to step 3a). Firstly, you will commit the changes you made; this is done using:

```
git commit -m "some meaningful message"
```

- Note the use of the *-m* flag, this just means the “commit message”. When you are making your first commit you should always use the *-m* flag, but afterward you should use the *-am* flag which just means to append on to the message. So, for later commits it should look like:

```
git commit -am "some meaningful message"
```

Step 3a:

- If you add additional files into your folder, you will need to execute one additional command before you commit. This command is known as “git add” and what you are essentially doing is adding a new file which you want git to keep track off. So, to add a file you can use:

```
git add “filename” NOTE: DO NOT INCLUDE QUOTES
```

Step 4:

- Now your final step after committing your changes is to push to the remote repository, this is simply done through using:

```
git push -u origin main
```

- On your very first push it is highly recommended to use the *-u* flag followed by *origin main* because you are setting “main” as your default branch for your commits to be pushed to. So, after you set your default branch you would only need to enter “git push” to push your code and git will automatically push it into your main branch.
- Now your git push might have failed, and this might be because you did not set up a username and email yet. To set this up, execute the following two commands (you only have to do this your first time):

```
git config --global user.name "firstName lastName"
git config --global user.email "yourguelphid@uoguelph.ca"
```

Step 5:

- Now all you must do is repeat steps 3 and 4 to push your changes onto the remote repository on GitLab and you are good to go!