Faculty of Computer Science, Dalhousie University *23-Feb-2026*

DGIN 5201 — Digital Transformation

Lab 4: A2.2 and Team Meetings (CSLab 2)

Location: Mona Campbell building 1201
Time: 11:35–12:25
Notes author: Vlado Keselj

Lab 4 (CS Lab 2): A2.2 and Team Meetings

Slide notes:

Lab Overview

- Completing A2.2 — Assignment 2 part 2
  - Complete e3 and e4 on timberlea
  - Submit e1, e2, e3, e4 to course GitLab
- Team meetings

Example e3: Available Material (Lab Review)

Exercise e3 Overview (Lab)

- We have a simple registration form on web in e2
- It can be printed and filled manually
- There is a link to available "material"
  - but no "material" yet
  - Let us provide material
- Login to your timberlea account as usual

Example e3: Next Iteration of Our Site: Available Material

- Let us make a copy of our `e2` site
- First, go back to the directory above `e2`:

```
cd ~/public_html/dgin5201
```

- Use command `pwd` to check your directory
- Copy `e2` to `e3` as an exact copy:

```
rsync -av e2/ e3/
```

- Check the new site `e3` in the browser
- `rsync` is a very useful utility for copying directory structures
  - it works locally as well as over ssh
  - it copies incrementally differences, which is important if two sites are large and mostly equal
  - it may preserve permissions if we use option `-a`

Example 3: Make material available

- Create readable and accessible ('executable') directory `material` (permissions: `rwxr-xr-x`)

- Copy PDF from: `~vlado/public/dt-mini-conf.pdf` into directory `material`
- Setup permissions for the directory `material` to be all readable and accessible (`rwxr-xr-x`), and for the file `dt-mini-conf.pdf` to be all readable (`rw-r--r--`)
- Try to access material link on the page. Does it work? Why?

Example 3: Prepare `.htaccess` in `material` directory

- Prepare file `.htaccess` and make it all readable (`rw-r--r--`):
```
Options Indexes
```
- Check material access now
- Add the following line to `.htaccess` and try accessing again:
```
Options Indexes
AddDescription "DT Conference Poster (PDF)" dt-mini-conf.pdf
```
- Add "and Information" to "DT Conference Poster" and access
- Add the following line and try again:
```
Options Indexes
IndexOptions DescriptionWidth=*
AddDescription "DT Conference Poster..." dt-mini-conf.pdf
```
- `.htaccess` file is used to configure Apache web server behaviour
  - can be used to provide a simple password-protected access

Summary of e3

- Files and permissions copied from `e2`
- File `material/dt-mini-conf.pdf` copied from a source as instructed, and permissions properly set
- File `material/.htaccess` prepared as specified

Hands-on e4: Password Protection

Example e4: Next Iteration of Our Site: Password Protection

- User `rsync` again to make a copy of `e3` to `e4`
- Example 4 (e4) will be used to demonstrate password protection

Example 4: Simple Password Protection

- `cd` to `e4` directory and let us prepare a password
- In a locally-only readable file `pw` (`rw-------`) we can save a password for our reference: `dt dt5201`
- Prepare the password for the site using the command:
```
htpasswd -bc .htpasswd dt dt5201
```
- Make the file `.htpasswd` all-readable and check its contents
- Prepare the file `.htaccess` and make it all readable:

```
AuthType Basic
AuthName dgin5201
AuthUserFile /users/webhome/<your_csid>/dgin5201/e4/.htpasswd
AuthGroupFile /dev/null
<Limit GET POST>
require user dt
</Limit>
```

- Check that site is password-protected

Summary of e4

- Files and permissions copied from `e3`
- `pw` file with permissions `rw-------`
- `.htpasswd` file with permissions `rw-r--r--` and appropriate content set up with the `htpasswd` command
- `.htpasswd` file with permissions `rw-r--r--` and content set up for password protection as given in class

Concepts Review: Example 4

- `htpasswd` command, password saved as hash
- Using `.htaccess` for password-controlled access

GitLab Submission: e1, e2, e3, e4

We will start this hands-on lab of working with the Dal FCS GitLab site, and also using your `timberlea` account to submit your assignment files using Git.

Step 1. Logging into DalFCS GitLab Website

The Faculty of Computer Science (FCS) at Dalhousie provides an open-source version of GitLab, which we will use in this tutorial.

Open your Web browser and go to the Dalhousie FCS GitLab web site: `https://git.cs.dal.ca` You should be able to see the Login screen, as shown in Figure 1. Login with your CSID login and password.

A user of GitLab can participate in different *projects* and have different roles in them. All participants of a project are called *members.* A member's role in a project can be: *Owner*, *Maintainer*, *Developer*, *Reporter*, or *Guest.* We will refer to the Dalhousie FCS GitLab installation as a repository of these projects, but we will also refer sometimes to your project as the repository. Since we use the term *course project* as the part of your coursework, we hope that this will not be confused with referring to the concept of *GitLab project*, which we will call sometimes a *repository* as well. It should always be clear from the context to which concept we are referring to.

Step 2. Find your CSID Course Project in DGIN5201 Group

Since you can be a member of different projects in GitLab, you first need to find the project that is assigned to you within this course. This project has the same name as your CSID and it is within the course project group, within this term. In order to find it you can use the "Projects" or "Groups" menu option in the GitLab Web interface, or directly type in the URL of the project.

The URL of the project, either when you find it or type directly, is the following:
`https://git.cs.dal.ca/courses/2026-winter/dgin-5201/<your_csid>` where `<your_csid>`

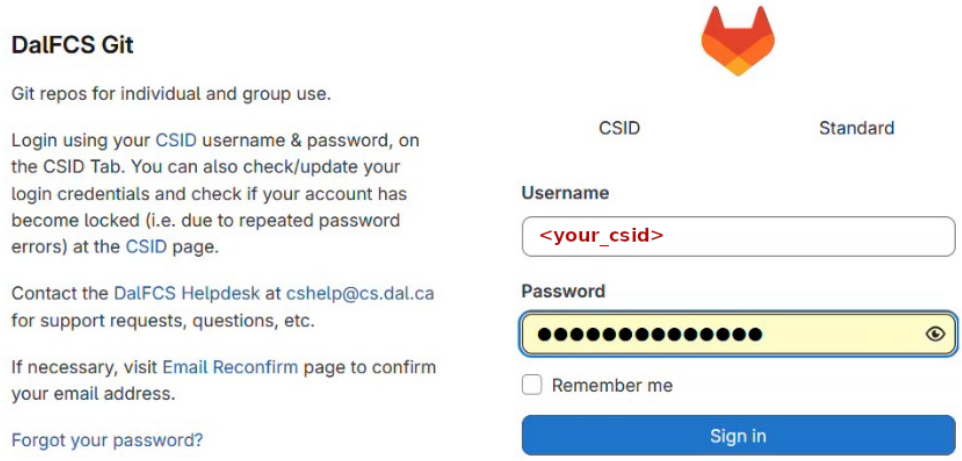


Figure 1: Dal GitLab login screen.

is your CSID. Figure 2 shows how the the top of front page of this project should look like, approximately, with some possible minor differences.

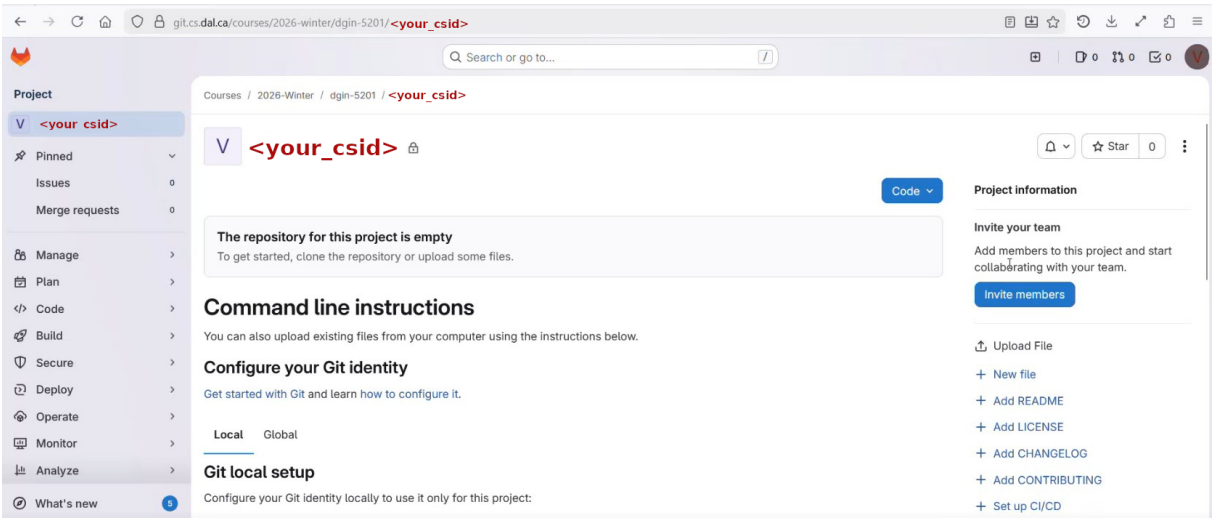


Figure 2: Top of the front page of your GitLab course repository

Step 3: Uploading your Files from `timberlea`

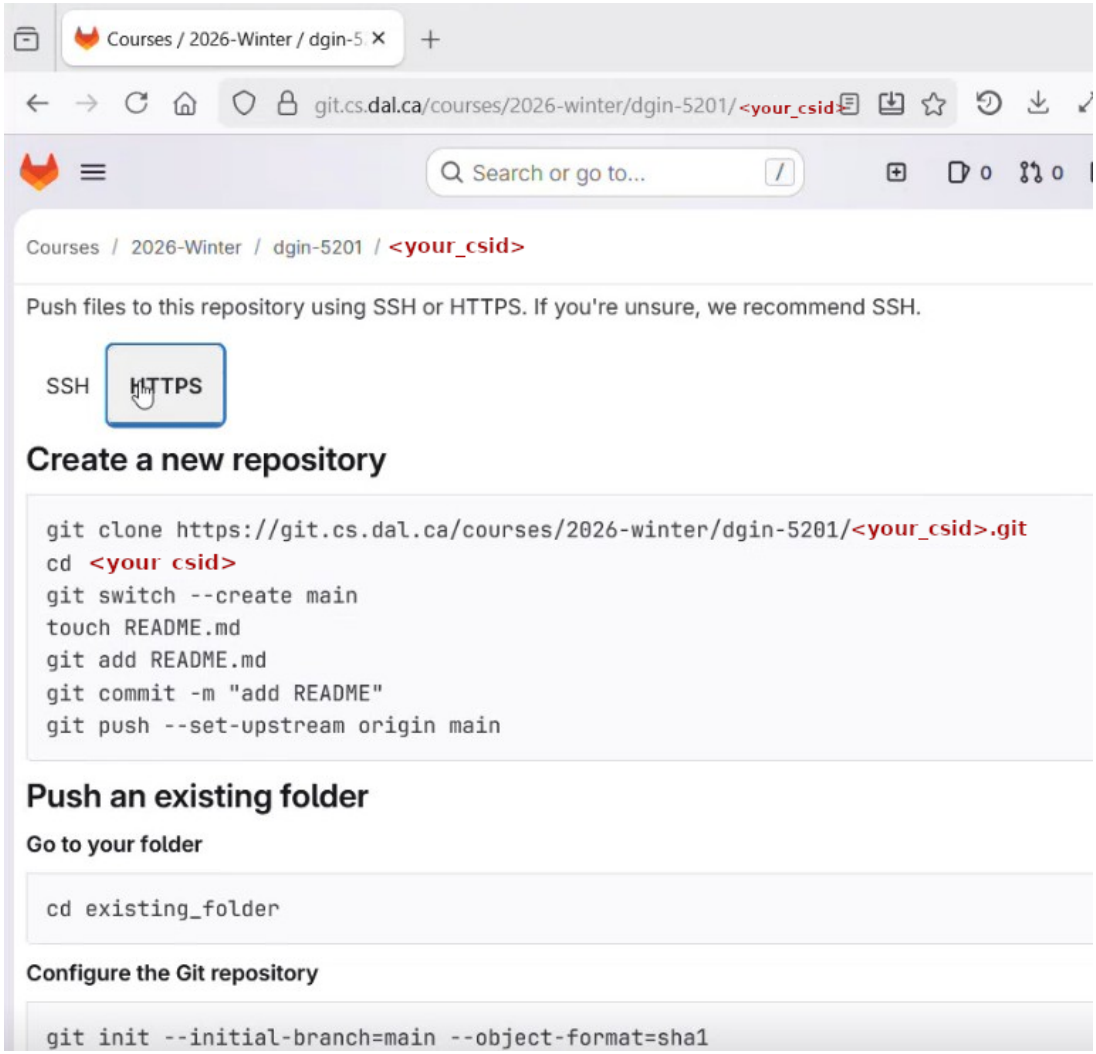
- In this step you should upload your lab files from `timberlea` into the GitLab server
- The instructions are shown in the GitLab page
- You should also open another command-line window for ssh login to `timberlea`

The instructions that we will use to upload the lab files (i.e., the directories `e1`, ... `e4`), are shown in the project page. There are two sets of instructions, one set based on HTTPS access, and another based on SSH. Make sure that you select HTTPS access now.

Note: During the class and recorded video of Lecture 10 on 19-Feb-2026, we followed first the istructions for SSH, which did not generally work, because students generally do not have their public SSH keys uploaded into the GitLab. We later switched to the HTTPS instructions, and they worked.

The HTTPS instructions are selected by clicking on the following HTTPS tab:

Selecting HTTPS Instructions for Initial Upload



We will follow instructions for pushing an existing folder as shown.

Instructions to Upload our Files

Push an existing folder

Go to your folder

```
cd existing_folder
```

Configure the Git repository

```
git init --initial-branch=main --object-format=sha1
git remote add origin https://git.cs.dal.ca/courses/2026-winter/dgin-5201/<your_csid>.git
git add .
git commit -m "Initial commit"
git push --set-upstream origin main
```

In order to execute these instructions we need to go to the server `timberlea` in our terminal window, where we are logged in. If the window is not there, we may need to login to `timberlea` as instructed.

Login to `timberlea` Server

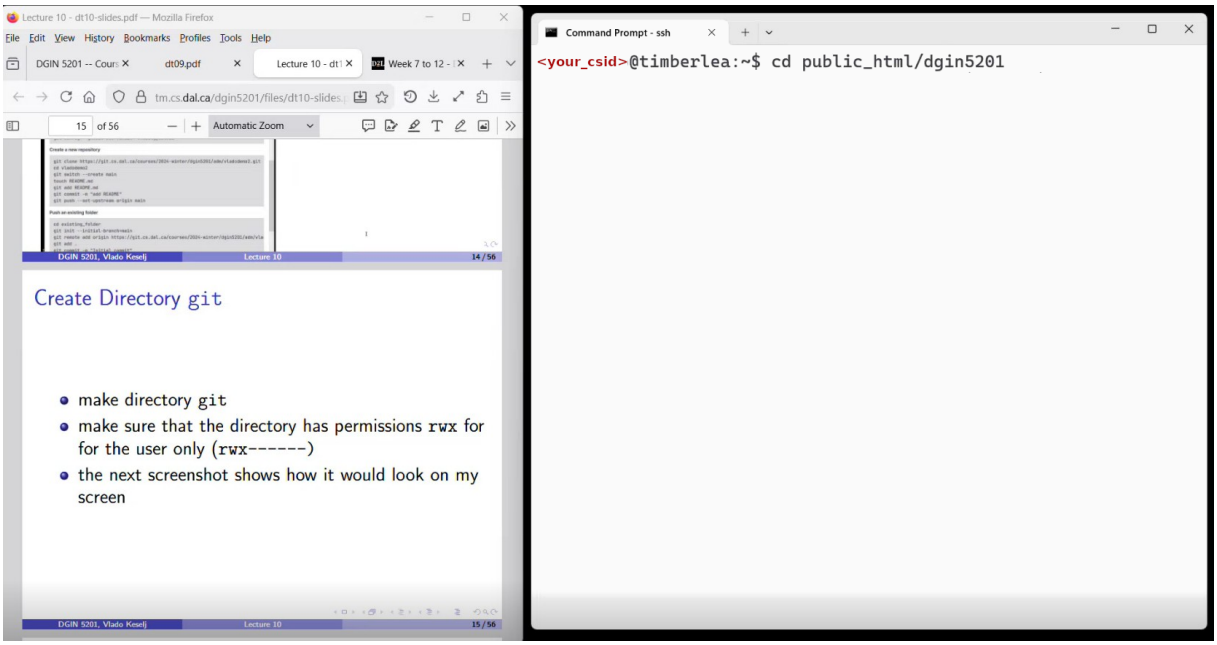
- If not logged in, we login to `timberlea` server
- We open a command-line window or terminal window in which we type the command:

  ```
  ssh <your_csid>@timberlea.cs.dal.ca
  ```

  where instead of `<your_csid>` you should use your own CSID
- or maybe you can use PuTTY, mobaxterm, or other SSH application
- You should try to have two windows: the web browser with GitLab, and the command-line window

Going to your Labs Directory

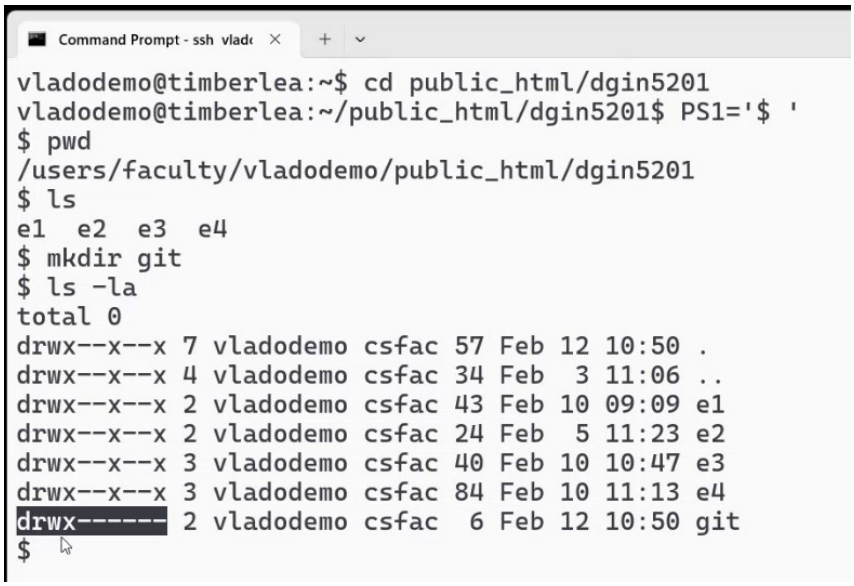
- Change directory to `~/public_html/dgin5201` and list contents
- your screen with two windows could look as follows:



We will now proceed to work in the terminal window, interacting with the `timberlea` server, while reading the instructions from the browser window, either from the provided notes, or from the GitLab web site.

Create Directory `git`

- make sure that you are in directory `~/public_html/dgin5201`
- create a directory `git`
- make sure that the directory has permissions `rwx` for the user only (`rwx------`)
- the next screenshot shows how it would look on my screen



Note: As explained earlier, the command `PS1='$ '` is used to simplify the command prompt, and it is not required that you use it.

We will now copy the directories `e1`, `e2`, `e3`, and `e4` into the directory `git` using the command `rsync`.

Copy directories `e1`...`e4` into directory `git`

- Copy the directories `e1`, `e2`, `e3`, and `e4` into the directory `git` using the `rsync` commands as follows:

```
rsync -av e1/ git/e1/
rsync -av e2/ git/e2/
rsync -av e3/ git/e3/
rsync -av e4/ git/e4/
```

- Go to the directory `git` and check its contents

You can use highlighted commands to check your git directory

```
$ ls
e1  e2  e3  e4  git
$ cd git
$ ls
e1  e2  e3  e4
$ pwd
/users/faculty/vladodemo/public_html/dgin5201/git
$ ls -a
.  ..  e1  e2  e3  e4
$ ls -la
total 0
drwx------ 6 vladodemo csfac 46 Feb 12 10:52 .
drwx--x--x 7 vladodemo csfac 57 Feb 12 10:50 ..
drwx--x--x 2 vladodemo csfac 43 Feb 10 09:09 e1
drwx--x--x 2 vladodemo csfac 24 Feb  5 11:23 e2
drwx--x--x 3 vladodemo csfac 40 Feb 10 10:47 e3
drwx--x--x 3 vladodemo csfac 84 Feb 10 11:13 e4
$
```

The output presented is the output obtained for the user `vladodemo`. The output that you get will be slightly different: it will include your CSID, and the first part of the path shown after the command `pwd` will be different prior to your CSID. The group shown after the `ls -la` command will not be `csfac` but some other group id.

Instructions to Upload our Files

- We use instructions in the GitLab page to upload exercises
- Use 5 git commands as shown, using your CSID

This is an explanation of the commands:

The first `git init` command creates a local git repository. The options specify that the default branch created is named `main`, and the secure hash function used is `sha1`, probably other than default to make operations faster. If your the command `ls -la` after this command you will notice a directory `.git` where local git repository is kept.

The second `git remote` command defines the remote repository for the local repository to be your course GitLab repository, and it is named `origin`.

The third `git add` command marks everything in the current directory; i.e., `e1`, `e2`, `e3`, and `e4` to be saved in the git repository. All these directories and files are not saved yet, but only "marked" for saving, so to speak, or *staged* in the Git language.

The fourth `git commit` command saves all files into the local directory as the first version of all files. The message after the option `-m` is the log message, which can be read after a `git log` command.

The fift, final `git push` command, pushes saved files into your GitLab course repository. It actually pushes the whole branch `main`, which means if we make saved more versions of the local files, all these versions would be pushed to the remote repository.

Note: In the class and video taked on 19-Feb-2026, we first defined the remote branch using the SSH address of the GitLab repository, which did not work since the SSH key was not prepared. For this reason, we defined renamed this repository using the "`git remote rename`" command, and created again the repository called `origin` with the HTTPS address, which worked.

GitLab Upload Completed

- Go to GitLab in the browser and refresh your repository page
- Directories e1, e2, e3, and e4 should be visible
- You can browse them to check the file contents
- With this, we finished uploading e1, e2, e3, e4 to GitLab