Code and data management with Git
Git and remote repositories

Some notes about using GitLab

Please take the following into account with respect to our GitLab server:

- You can login to GitLab using your LUMC account.
- If you don't have an LUMC account, ask us to create a GitLab account for you and select *Standard* instead of *LDAP* on the login page.

SSH Keys on GitLab

It is convenient to set up an ssh key on GitLab.

Open `https://git.lumc.nl` in your browser and log in.

Click on your avatar (top-right corner), choose "Settings", and then go the "SSH Keys" page.

Go to your terminal (or putty) and make an SSH key, as follows (you can leave all fields that require an input blank, i.e., press enter):

```
$ ssh-keygen
```

```
Enter file in which to save the key (/home/<username>/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

You can now retrieve your ssh key with the following `cat` command:

```
$ cat ~/.ssh/id_rsa.pub
```

Copy the key and paste it in the "Key" field on the GitLab page.

Give your key a title, e.g., "Course machine", and press the "Add key" button.

Add your repository to GitLab

> (If you don't have a repository on your local machine from the previous practical, create one now with at least one commit.)

Now you have a nice repository, of course you want to share it on GitLab.

Go to GitLab (`https://git.lumc.nl`) and create a new project (use the same name you used to store your repository locally).

- *Question:* What is the repository URL for your new project?

Add the GitLab repository as a remote to your local repository.

Push your commits to GitLab.

- *Question:* Can you see your repository content in the GitLab web interface?

Hint: Use `git push` once with the `-u` flag so you can use the `git push` / `git pull` shortcuts.

More synchronisation

Look for a way to edit a file directly from the GitLab web interface (in your browser) and do this at least once.

Update your local copy of the file with the change you just made (by fetching and merging).

Now make another commit locally and push it to the GitLab server. Verify that all these changes are now present both on your local machine and on the GitLab server.

Collaboration

In this practical, you'll work with both your repository and with the repository your neighbour created during the previous practical. We'll refer to this repository as **N**, and to your own repository from the previous practical as **Y**. Note that by now both repositories should have a corresponding GitLab project (a remote repository).

Make sure that your GitLab project **Y** is private and add your colleague as a member to it as **Guest**.

Try to **clone** his or her repository to your local machine. Make sure to do this in a separate directory.

- *Question:* Was the cloning successful? Try to identify why.

Change the role of your neighbour from **Guest** to **Developer** and try to **clone** his repository again once your neighbour made you a **Developer**.

Edit locally a file from your neighbour **N** directory, stage it, commit it, and try to **push**.

- *Question:* Were you able to push? Try to identify why.

Change your neighbour's role in your GitLab project from **Developer** to **Maintainer**.

After your neighbour upgraded your role to **Maintainer** try to push again your changes to his GitLab repository.

Inspect the commit graph on GitLab.