Using Git through c4science

March 10, 2017

This is a short tutorial of Git (through c4science), which can be a useful tool during your collaboration with your project partner for the semester project.

Why Git?

Git is a free and open source distributed version control system (VCS) designed to handle small or large projects with efficiency. It can help you work on the same project with other people, store your files on an online repository (while also preserving a clone of it into a local directory) and also keep a history of the changes you will make.

If you are not familiar with a VCS, just think about a system that records changes to a file or set of files over time, so that you can recall specific versions later. All files are stored in the repository and the VCS can record and manage all the changes you make to them. Changes are usually identified by a code, termed the "revision number", or simply "revision". For example, an initial set of files is "revision 1". When the first change is made, the resulting set is "revision 2", and so on. Each revision is associated with a timestamp and the username of the person making the change. Revisions can be compared, restored, and with some types of files, merged. You may have already encountered a VCS-like system that records the changes of files and maintains revisions. That is the well-known “Google Docs”!

We prefer Git, because it is easy to learn, has a tiny footprint with fast performance and offers a smart way to manage a repository with numerous possibilities. It is also supported by the c4science platform.

How do we create and share a Git Repository in c4science?

Being a student at EPFL, you can create a Git Repository through the c4science platform.

C4 Science is a co-creation platform, curation and code sharing, that includes version control systems, common authentication to all Swiss universities to local + external collaborators, a social dimension (wikis, bug tracking, ...) and a code test system. For the purposes of your project, we are interested in the VCS systems that are supported by c4science, and especially Git.

The steps for creating a Git repository inside c4science are the following (both students working on the same project need to follow steps 1-4 and 10, but only one of them needs to do the rest):

![Figure 1: Edit your account.](image)
1. Log into c4science (https://c4science.ch/auth/start/?next=%2F) using your Gaspar account.

2. You may be asked to create a username. For this, you can use either your Gaspar username or another username that you will need to remember. You do not need to create a password, because you will always log in through your Gaspar account.

3. Once inside c4science platform, edit your profile by clicking on the account icon at the upper right corner (Fig. 1). Click on “Manage” and then “Edit Profile”. Please add your real name, so that other people who want to share a project or repository with you can easily find you.

4. Set up an authentication method for your c4science account: You can do this by either uploading your SSH public keys or setting up a VCS password. Click again the account icon at the upper right corner (Fig. 1) and click on “Settings”. Then select one of the two methods under “Authentication” at the left of the page (Fig 2) and follow the instructions. You do not need to set both authentication methods, one is enough. If you do not know much about public keys, then it is easier to create a VCS password and remember it.

5. Once both students working on a project have set up their c4science accounts, one of them needs to create a project: Go to the main c4science page and click on “Create a Project” at the right part of the page. Name your project and add both your name and your collaborator’s name in the “Initial Members” field (Fig 3).

6. Once you have created your project, you are free to create a Git repository: Go back to the main c4science page and click on “Create a GIT Repository” at the right part of the page. Fill in the information that is asked from the page and create the repository. Do not forget to add your project’s name as a tag. Once done, you will have created an inactive private repository.

7. Activate your repository (as shown in Fig. 4) and go back to the home page. You should be able to see your newly created repository in your dashboard (Fig 5).
8. Click on the name of the repository, find its url and share it with your project partner (e.g. in Fig. 6 the url of the example repository is https://c4science.ch/source/proj.git).

9. Click on “Manage Repository” and then “Polices” (at the left of the page), and then edit polices to allow only the members of your project to “view”, “edit” and “can publish” the repository. Please do not make your project available to public access. So, make sure that the project access is restricted to your project partner and only. **Warning: if your project is plagiarized by anybody else, there will be a penalty on your grade, too.**

10. *(For both collaborators of the project:)* Clone the Git repository into your VM (under /myfiles). To do this, use the following command:

    ```bash
    $ git clone <project_url> ~/myfiles/<project_directory>
    ```

    E.g. for the above example repository the command would be:

    ```bash
    $ git clone https://c4science.ch/source/proj.git ~/myfiles/project
    ```

    You will be asked the username and the password (i.e. the VCS password) that you created in steps 2 and 4. If you want, you may also clone the repository to your local machine.

    **Note:** Git is already installed in the machines of the CO rooms that you are using for the course. However, if you plan to maintain locally a clone of your repository, you have to install it. Instructions can be found [here.](#)
Figure 4: Activate a repository.

Main Commands of a Git Repository

A Git repository is a collection of files, which you can manage though a set of commands from inside the directory of the cloned version. Change to that directory and use Git commands to perform specific actions, e.g.:

- add new files to your repository, e.g. the source file utilitaire.c:
  
  
  \$ git add utilitaire.c

- commit your changes to your cloned repository in order to save and record multiple versions of the files over time (which is very useful in case you want to revert back one day and check a previously working version of your code), e.g.:

  \$ git commit -m 'Added first file to repo' utilitaire.c

- push your changes to the online repository, so that your collaborator can review them:

  \$ git push

- pull the online directory and merge any possible changes that have been made by your collaborator with your changes:

  \$ git pull

The best basic intro to Git for beginners is this Git simulator, which shows how to use Git in less than 30 minutes. It provides an interactive pseudo-terminal, and step-by-step instructions on how to start using Git. It is a very fun and easy way to start playing with Git. You can ignore the `git init` command in our context, because this has already been done by c4science.

However if you feel that this is not enough, you can watch the following videos:

https://www.youtube.com/watch?v=0fKg7e37bQE&ab_channel=LearnCode.academy
https://www.youtube.com/watch?v=oFYyTzwMyAg&ab_channel=LearnCode.academy.

A complete documentation and tutorial about Git can be found here.
Figure 5: c4science Dashboard.

Figure 6: Repository.