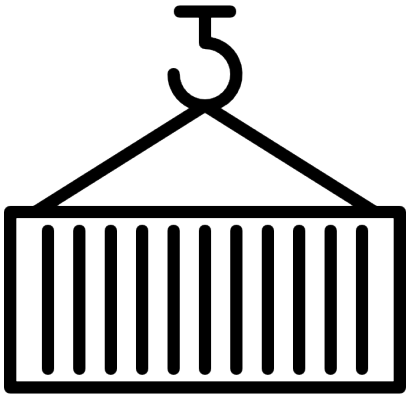
C1 - Devops

C-DEV-110

Git

Mini-Project 01



1.3

Instructions: Before You Go Further

- The exercises have to be done alone.
- Sources must be turned in to GitHub using GIT.

Exercise 1: Intranet

Log in to the Epitech's Intranet using the login and password given to you, then enroll to the unit's activities.

Now connect to your mailbox and read the confirmation e-mails you received.

intra.epitech.eu
Office 365

If you are not registered on the intranet you will **NOT** be marked

Exercise 2: Console

All exercises of this mini-project will take place in the console (or "terminal").

A terminal does not have any graphical interface. Everything is done from command line.

The 'man' command, for example, can provide more information about other commands.

Use the commands 'ls' and 'cd' to show the files in the current directory and to move from one directory to another.

man ls; man cd

Exercise 3: Create directory

To help organize yourself you are going to create a directory named “Rendu”. The idea is to put all your projects that you will be doing in that repository so that it is easier for us and for you to find your work.

Exercise 4: Git

To turn in your exercises, you have to use the version control software ‘git’.
To do this, you must first install it on your computer.

Note that to install software on your computer you must run the software installation command as administrator.

A guide is available to students, which will explain briefly the steps to do a delivery:
how_to_turn_in.pdf

man sudo; man yum for fedora or man apt for ubuntu ;
Documentation Git

Exercise 5: IDE

Create a file named “test” in your home directory and write “Les Entrechats” inside. You can display the content of the file with the command cat.

There are many ways to write in a file, but this time, you have to use the Emacs text editor.

~ represent the path to your home directory. man touch;

Exercise 6: SSH Key

This folder exists on GitHub server and you must now clone a copy on your computer.

In order to communicate with the GitHub server and for this server to accept your connection, you must have an ssh key.
First you need to generate a key.
Then in your home, go to the **.ssh** directory

man mkdir; man ssh-keygen;

Exercise 7: SSH Key Upload

To communicate with the server, you have to send your public ssh key to GitHub.

Once you have achieved this step, it is still necessary to tell your machine which key to use with which server. You have to create a file ~/.ssh/config and write the right configuration. (This file must have the rights 600).

man chmod;

EXERCICE 7 3/4: CREATE A REPOSITORY

This exercise should only be done if (and only if) you don't have access to the official repository (it can happen on a school day) of the current project on the Github organization.

The goal of this exercise is to use a personal alternative repository while the official repository is being created.
Create a repository on Github (https://github.com/new) on your personal space.

This repository must be private.

EXERCISE 8: CLONING

If the previous exercises have been properly done, you can now retrieve the folder you created on the GitHub server using git.

The repository address is: git@github.com:EpitechIT2020/*module-city-session-subject-firstname.lastname*.git

In case of success git should give you the following message :

Terminal

```
~/Rendu>
[...]
warning: You appear to have cloned an empty repository.
[...]
```

man git-clone;

If git tells you that you don't have the rights, wait 5 minutes for the key to be uploaded to the server. If it still doesn't work, redo exercises 7 and 8.

Exercise 9: Copy

Copy the file “test” you created in exercise 5 inside the turn in directory that you just cloned.

man cp;

Exercise 10: Add

For the following exercises, you have to use git to send the file “test” on the server.

First, you have to add this file to the local git repository to enable it to be tracked.
This step must be done only one time for each new file you wish to add on the git repository.

man git-add;

Exercise 11: Commit

Once your file has been added, you must save the changes locally using the command “commit” of git.
Remember to always put a suitable commit message to be able to easily find an older version of your work.
In our case, you can write “Added test file”.

man git-commit;

Exercise 12: Push (12pts)

Now that your changes have been saved locally, you must go through one last step. You have to synchronize your local repository with the one on the server for your modification to be accessible from others who have the correct permissions.
This step is done with the command “push” of git.

man git-push;

Exercise 13: Z (6pts)

Turn in: ex_01/my_z

In your git repository, make another folder “ex_01”. Then create a file called “my_z” in which you will write the letter ‘z’ followed by a newline.

Terminal

```
~/ex_01> cat -e my_z
z$
```

Use Emacs.

Do not forget to turn in! This will not be reminded again.

Exercise 13 3/4: submit your work to the right repository

This exercise should only be done if (and only if) you didn't had access to the official repository (it can happen on a school day) of the current project on the Github organization.

This exercise should only be done when the official repository is finally created.

The objective of this exercise is to transfer all the content (including the git history) of your local git folder from your personal repository to the official repository of the day.
Change the origin of your local repository.

Exercise 14 : branch

For this exercise you are going to create a branch in your repo. When you clone a repository you are set on a branch called *master*. Now create a second branch named *dev*. Once you are on this branch, create a file named **my_work.txt**. In this file write: “On the dev branch” followed by a newline.
Now commit and push your file on the *dev* branch.

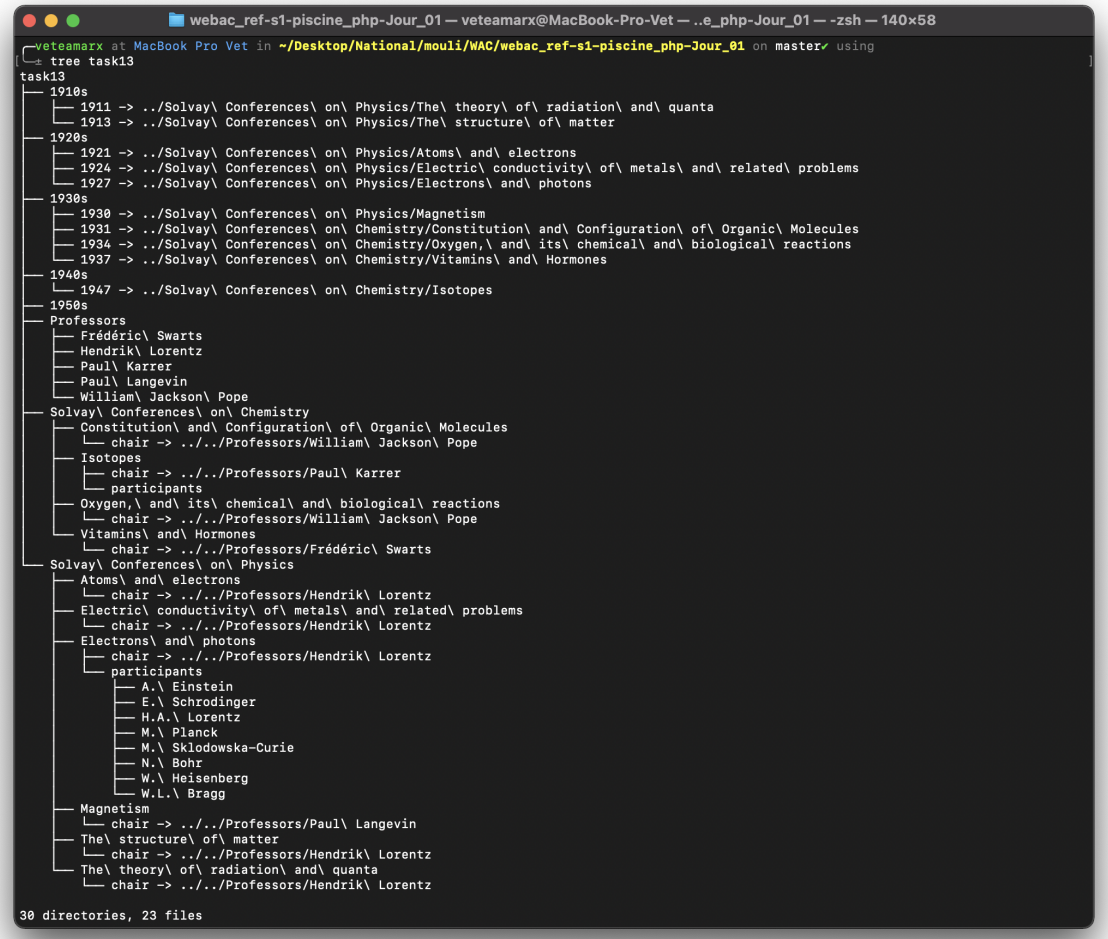
man git-branch

Exercise 15: merge

Now that you have work on your second branch named *dev*, you want all of your work on the *master* branch. To do so, you will need to merge the *dev* and *master* branches.
Once this is done you still need to commit and push your merge. This will lead to only have one branch left (*master*) as *dev* merged into. You will see all of your work on *master* now.

man git-merge

EXERCISE 16: TREE



Reproduce the folder structure as showned.

Git handles empty folders differently.

EXERCISE 17 - TAR

Create a compressed tarball of the previous step. Delivery : Piscine_PHP_Jour_01/task14/task13.tgz

man tar

Finished? Good! Did you really check everything properly? Yes? Do it again, just to be on the safer side =D. After that if you still have some free time take a bit of time to learn more about GNU/Linux and some of the basic commands, shell configuration and personalization.

man env; see .bashrc.