EPFL Fall 2022
Master level
MATH-458
September 22, 2023

Programming Concepts in Scientific Computing

Using GIT

The goal of the present exercise is to use the revision control GIT.

Exercise 1: *Creating your own GitLab-EPFL repository*

1. To create your own repository go to https://gitlab.epfl.ch
2. Log in using your EPFL credentials
3. Click on the link *Create a project* and follow the instruction to create your first repository named `<name-of-your-choice>`.
4. Open a terminal.
5. In order to be able to retrieve repositories on GitLab, you will need a pair of `ssh` keys. Run in the console:

   ```
   ssh-keygen                    # Follow the steps
   cat ~/.ssh/id_rsa.pub
   ```

   Copy the output of the last command. Go to https://gitlab.epfl.ch/profile. On the left panel, click "SSH Keys". Paste the output of the `cat` command and give it a title. Additionally, you should add GitLab to the list of known hosts:

   ```
   touch ~/.ssh/known_hosts
   ssh-keyscan gitlab.epfl.ch >> ~/.ssh/known_hosts
   ```

6. GitLab should provide you with an URL for your repository (it starts with `git@gitlab.epfl.ch:` and accessible via "Clone" button on the repository page). You can clone your repository

   ```
   git clone git@gitlab.epfl.ch:<your-username>/<name-of-your-choice>.git
   ```

   You now have a local clone of your repository named `<name-of-your-choice>` (a new directory should have been created for it). You can go in the directory using:

   ```
   cd <name-of-your-choice>
   ```

7. Inside this directory, create a new file *test.cpp*.
8. Add a line of text in this file
9. Observe the actual status

   ```
   git status
   ```

   You should see that the file *test.cpp* is shown in the terminal.

10. Add this file to the repository

    ```
    git add test.cpp
    ```

11. Observe the actual status

```
git status
```

12. Commit your changes

```
git commit -m "Your commit message"
```

Note: if you use the `commit` command without the `-m` option, an editor will open. There's a chance that it is VIM-editor. If it is the case, you can exit by pressing the `<Escape>` key a bunch of times, then entering `:q!<Enter>`. Then you can use the `-m` option with a commit message. Note: If it's the first time you are using Git on the computer you will see the following message.

```
*** Please tell me who you are.

Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.
```

These two commands are used to configure git for the commits. If you add the `--global` flag, git will use the same email address and name for all of your local git repositories.

13. Push your changes to your repository

```
git push origin main
```

The *origin main* is mandatory only the first time. *main* (formerly *master*) is name GitLab automatically assigns to your default branch. In order to get rid of the buzzy message you should hit:

```
git config --global push.default simple
```

After you can simply hit

```
git push
```

14. Go to https://gitlab.epfl.ch and verify that the newly created *test.cpp* is in the repository `<name-of-your-choice>`

Exercise 2: *Conflict resolution*

1. Clone your repository once again in an other folder to simulate the fact that someone else share the repository

```
cd ..
git clone <gitlab-repository-url> <other-folder-name>
cd <other-folder-name>
```

2. Change a character in the line you wrote in this new copy of the file
3. Commit the change
4. Push the change on the server
5. In the fist clone change the same line differently
6. Commit this change
7. Before pushing the changes you have to pull to get the server changes

```
git pull
```

8. Since you made a change that has no unique merge solution you will get a conflict message.
9. Open the *test.cpp* file and choose one of the version that is in between brackets be removing everything else than the test you want

```
<<<<<<<<<<
One version
==========
Other version
>>>>>>>>>>
```

10. You can finish the conflict resolution by comiting the solution

```
git commit -am "resolution message"
```

11. And finally you can push your changes
12. What does the '-a' stands for ? You can find out by issuing the command:

```
git help commit
```

    You can thus realize that all commands are documented within the command line:

```
git help the-command-I-am-intrigued-about
```

Exercise 3: *Add a remote to the class repository*

In the repository `<name-of-your-choice>`, add a new remote that points to the official PCSC repository.

```
git remote add upstream https://gitlab.epfl.ch/anciaux/pcsc.git
```

You can see all the remotes in you repository with:

```
git remote -v
```

Pull the class material in your repository:

```
git pull upstream master
```

You will probably get a screen with a merge message. If it is VIM, press `:wq<Enter>` to save and exit. You should now be able to find this week's exercises. If you see an error about 'unrelated histories' you should use the following instead:

```
git pull --allow-unrelated-histories upstream master
```

You can push this new content to your personal repository:

```
git push
```

All material for the class, including class notes and exercises starters will be provided through this GIT repository. Avoid putting modifications directly in the class material, or it is very likely that you will get conflicts if you try to pull from `upstream`. Instead, copy the files you need to modify elsewhere in your repository.

Exercise 4: *Branches*

You can find an interactive exercise on http://learngitbranching.js.org/.