IMT SCHOOL FOR ADVANCED STUDIES LUCCA

Python & networks: making a Git messing up with an Anaconda and a Python

Fabio Saracco, IMT Lucca

IMT SCHOOL FOR ADVANCED STUDIES LUCCA

Anaconda, but friends call me Conda

*Conda is an **open source package management system** and **environment management system** that runs on Windows, macOS and Linux. […] It was created for Python programs, but it can package and distribute software for any language.*

—Conda presentation

Installation Official Page

Conda **environment**:

Unix OS works on python. What may happen is that, while playing with python, you mess up with some package or module or some dependency among module or whatever. A conda environment is a place with its own python installation that does not comunicate with the outside, thus preventing any problem with everything outside.

In most of the cases we will not use **conda** package manager,
but standard **pip**

IMT SCHOOL FOR ADVANCED STUDIES LUCCA

Define your own environment

Conda allows to create separate environments containing files, packages and their dependencies that will not interact with other environments.
When you begin using conda, you already have a default environment named `base`.
Create separate environments to keep your programs isolated from each other.

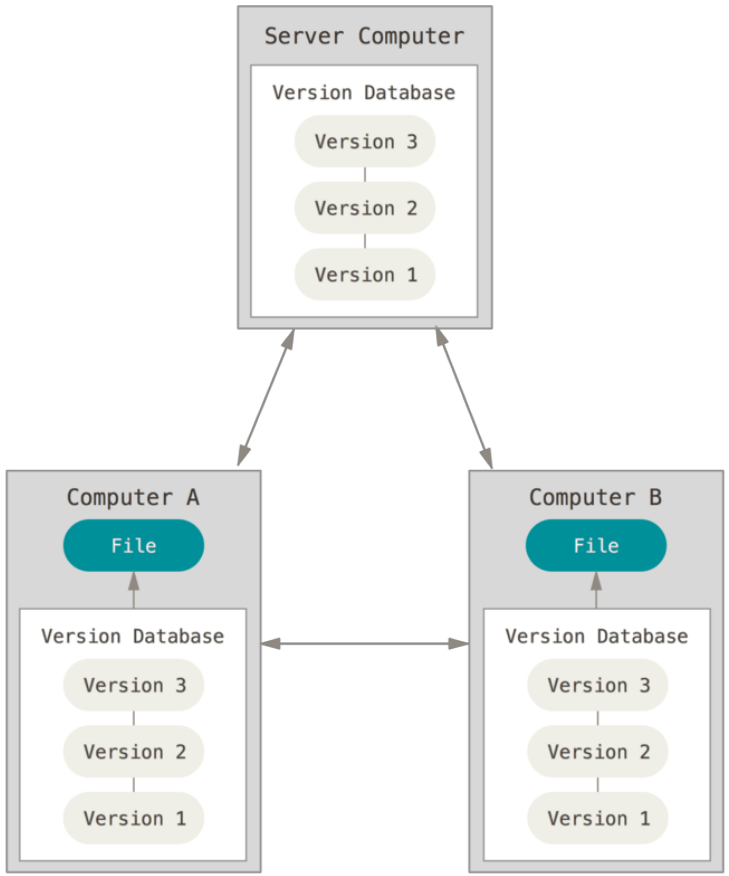
The official guide

1. On Windows: look for Anaconda Prompt from the menu Start (under Anaconda3)
   On Linux/Mac: open a terminal
2. Go to the folder where Anaconda is installed
3. Type `conda create -n pynne python=2.7`

   It forces the environment to install **python version 2.7**
4. Type `conda list` to check if the environment was created.
5. Activate the environment via `source activate pynne` on Linux/Mac or `activate pynne` on Windows
6. Deactivate the environment via `source deactivate` on Linux/Mac or `deactivate` on Windows

IMT SCHOOL FOR ADVANCED STUDIES LUCCA

Git: the freedom of making mistakes



Version control software (VCS) allows you to take **snapshots** of a project whenever it's in a working state. When you make changes to a project—for example, when you implement a new feature—you have the option of reverting back to a previous working state if the project's current state isn't functioning well. Using a VCS also generally means that if you screw things up or lose files, you can easily recover.

The main idea is to save changes to the file in the repository
(just the **changes**!)
and distribute it among the synchronised machines, i.e. the server and your local machine(s).

Using git:

IMT SCHOOL FOR ADVANCED STUDIES LUCCA

Git: the structure

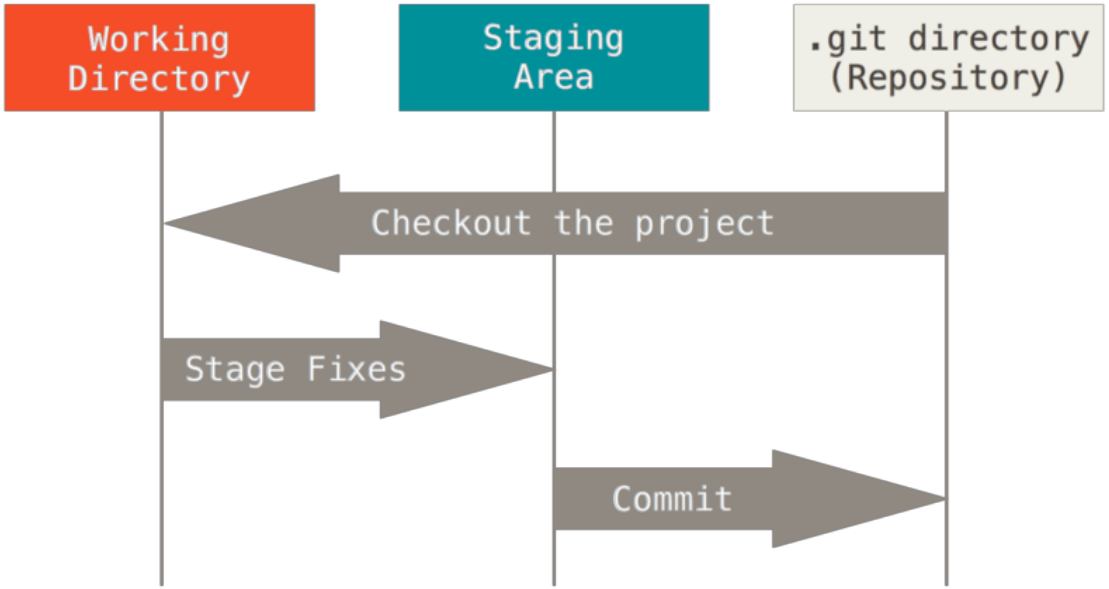
It is a single checkout of one version of the project

It stores information about what will go into your next commit

It's where Git stores the metadata and object database for your project

3 states of file:

1. committed
2. staged
3. modified



Standard usage:

1. Modify some files in your **working directory**;
2. **Stage** the ones you are interested in;
3. Commit them to the **repository**

IMT SCHOOL FOR ADVANCED STUDIES LUCCA

Git: installation

on Mac:

Do nothing, everything is already installed. Check it via writing on a terminal

```
git - - version
```

It should return the git version installed. Otherwise install it via the following link

on Linux:

Probably the same. Check it as for Mac. If it is not installed

```
sudo apt-get install git-all
```

on Windows:

If you want to install it only in the virtual environment (**suggested!**), once the environment is activated, digit

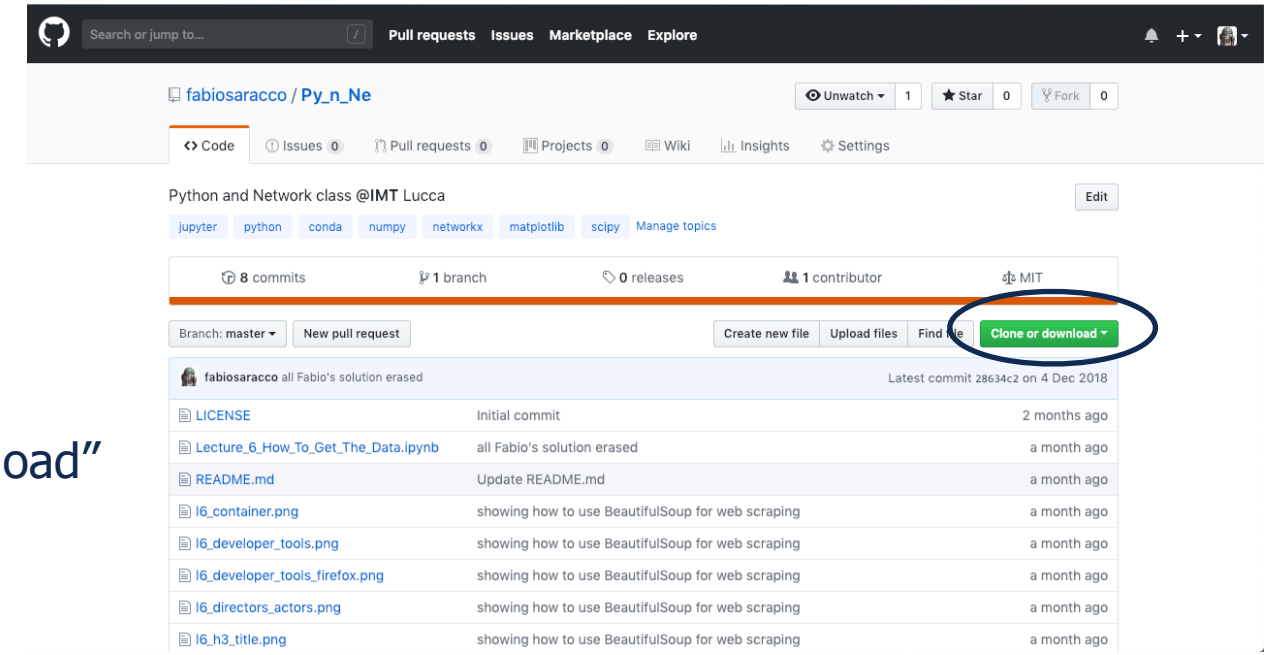
```
conda install -c anaconda git
```

Otherwise the following link takes you directly to the download.

IMT SCHOOL FOR ADVANCED STUDIES LUCCA

Forking a repository

1. (Once you have an account on github) Go to the repository
2. Click on "Fork"
3. Go to your "Fork"
4. Click on "Clone or download"
5. Copy the url



6. Go to the folder where you want to put everything **inside your conda environment** and create a new folder for the repo

7. Digit `git clone`, then paste the url. Then press Return.

8. List all the file, i.e. `ls` (on Unix machines) or `dir` (Windows)

IMT SCHOOL FOR ADVANCED STUDIES LUCCA

From Working to the Stage

1. Go to the local folder of the repository
2. Digit `echo “My cool new file" >> new_file.txt`
3. new_file is in the **Working directory**
4. Digit `git status`
5. Digit `git add new_file.txt`
6. Now the file is in the **Stage**. To commit i.e. to keep track of changes, digit `git commit -a -m ‘some message’`
7. Digit `git status`
8. You are a **commit ahead** And now?

A “branch” is the version of the projection you are working on

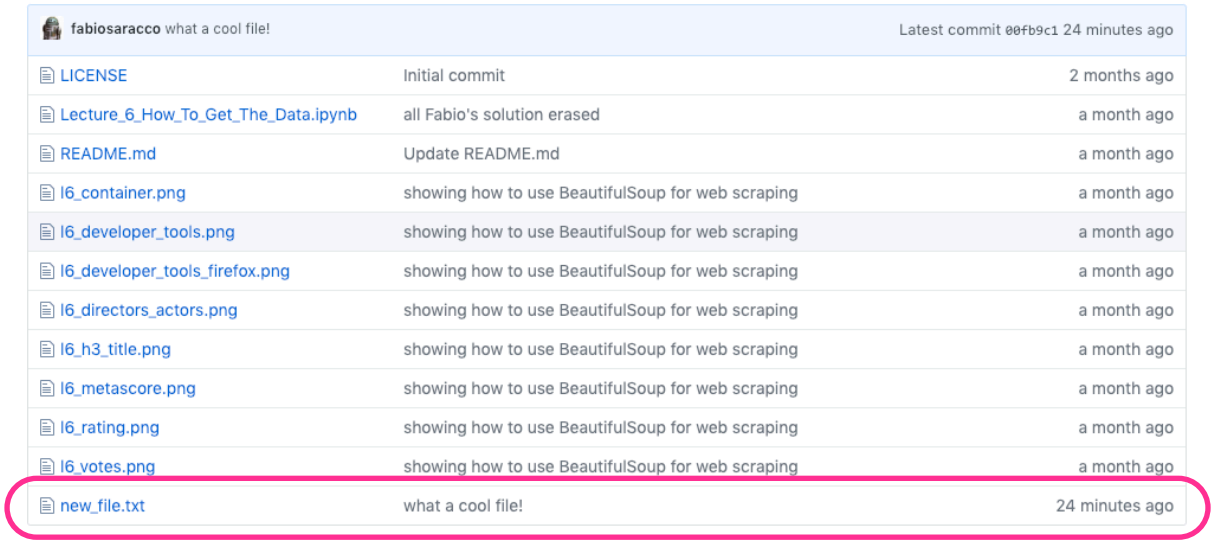
```
[Fabios-MacBook-Pro:Py_n_Ne Fabio$ echo "My cool new file" >>new_file.txt
[Fabios-MacBook-Pro:Py_n_Ne Fabio$ git status
On branch master
Your branch is up to date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .ipynb_checkpoints/
        new_file.txt

nothing added to commit but untracked files present (use "git add" to track)
[Fabios-MacBook-Pro:Py_n_Ne Fabio$ git add new_file.txt
[Fabios-MacBook-Pro:Py_n_Ne Fabio$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   new_file.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .ipynb_checkpoints/

[Fabios-MacBook-Pro:Py_n_Ne Fabio$ git commit -am 'what a cool file!'
[master 00fb9c1] what a cool file!
 1 file changed, 1 insertion(+)
 create mode 100644 new_file.txt
```

Push/Pull

1. Once the files you are interested in are committed, you can push them to the server repository by `git push`

```
Fabios-MacBook-Pro:Py_n_Ne Fabio$ git push
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 292 bytes | 292.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/fabiosaracco/Py_n_Ne.git
   28634c2..00fb9c1  master -> master
Fabios-MacBook-Pro:Py_n_Ne Fabio$
```

2. You can check that the file is there by checking the online repository



fabiosaracco what a cool file! — Latest commit 00fb9c1 24 minutes ago

LICENSE	Initial commit	2 months ago
Lecture_6_How_To_Get_The_Data.ipynb	all Fabio's solution erased	a month ago
README.md	Update README.md	a month ago
l6_container.png	showing how to use BeautifulSoup for web scraping	a month ago
l6_developer_tools.png	showing how to use BeautifulSoup for web scraping	a month ago
l6_developer_tools_firefox.png	showing how to use BeautifulSoup for web scraping	a month ago
l6_directors_actors.png	showing how to use BeautifulSoup for web scraping	a month ago
l6_h3_title.png	showing how to use BeautifulSoup for web scraping	a month ago
l6_metascore.png	showing how to use BeautifulSoup for web scraping	a month ago
l6_rating.png	showing how to use BeautifulSoup for web scraping	a month ago
l6_votes.png	showing how to use BeautifulSoup for web scraping	a month ago
new_file.txt	what a cool file!	24 minutes ago

The opposite operation of `git push` is `git pull`, i.e. it "pulls" changes from the server version of the repository to your **local folder.**

!!!REMEMBER!!! Before you start doing something, always `git pull`

IMT SCHOOL FOR ADVANCED STUDIES LUCCA

Brief guide to handguns and Tequila (1)

"A computer lets you make more mistakes faster than any other invention, with the possible exceptions of handguns and Tequila."
*—**Mitch Ratcliffe***

Remove files from the staging area:

- `git rm filename`
  removes the file from the working folder
- `git rm - - cache filename` removes the file from the stage, but not from the working directory

Discard local modifications:

- `git checkout - - filename` takes filename to the last committed version

IMT SCHOOL FOR ADVANCED STUDIES LUCCA

Brief guide to handguns and Tequila (2)

Go back to some **local** (i.e. before **pushing**) commits:

1. undo the last 2 commits, but keep changes
   `git reset HEAD~2`

2. undo the last 2 commits, and discard changes
   `git reset - - hard HEAD~2`


```
commit 00fb9c14c2ba1333af07721e7d4d04b664b27e7b (HEAD -> master, origin/master,
origin/HEAD)
Author: sarawalk <f.sarawalk@gmail.com>
Date:   Mon Jan 14 11:44:47 2019 +0100

    what a cool file!

commit 28634c289b1c7d8111b5e62be81d6986e2bf706a
Author: sarawalk <f.sarawalk@gmail.com>
Date:   Tue Dec 4 17:40:30 2018 +0100
```

3. Every commit is equipped with a log. You can access it by `git log`. You can reset to a specific commit even by its log, i.e
   `git reset - - hard 28634c`
   (the first 5 characters are enough)

`git merge origin master`

Go back to some **global** (i.e. **AFTER** pushing) commits:

undo the last commit, but don't create a revert commit

```
git revert -n HEAD
git merge
```

In doubt, check this

IMT SCHOOL FOR ADVANCED STUDIES LUCCA

Brief guide to handguns and Tequila (3)

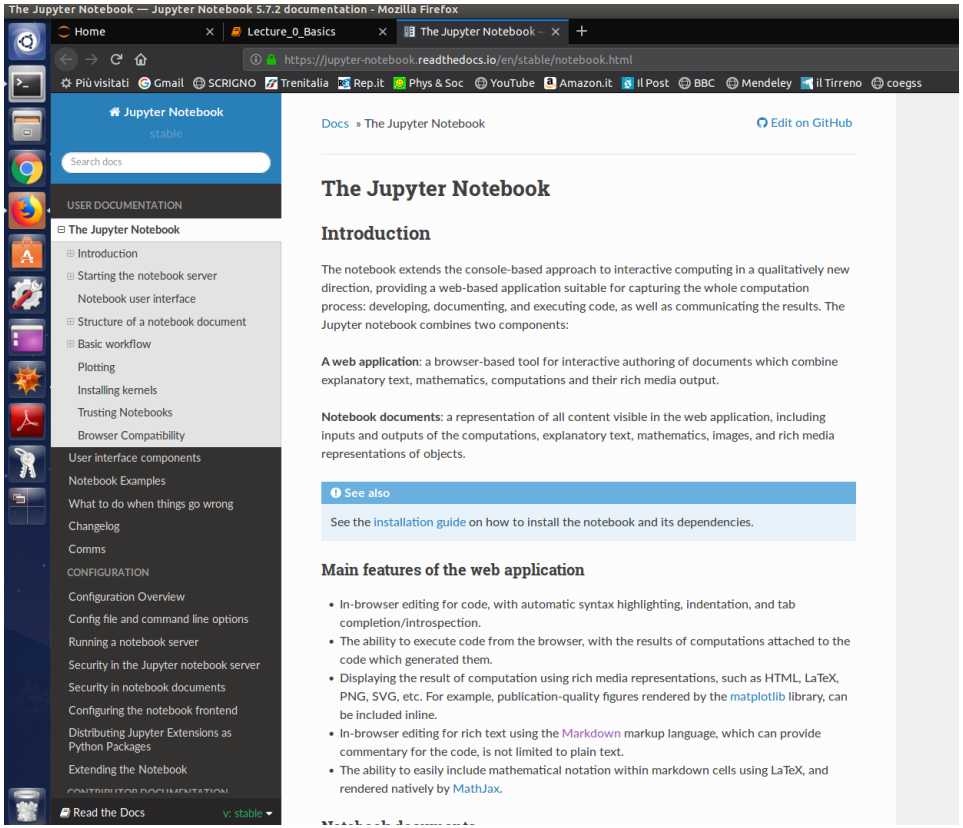
It may happen, for instance if the repository is **shared**, that the **version in the remote repository is different from the one in the local repository** (i.e. both committed in the same position) and changes cannot be merged automatically.

1. **Easy solution**: discard your local changes, `git reset - - hard 28634c`

2. **Less easy solution**: you need something in between the two versions.
   Use `git mergetool` and select element by element what you want from each of the two versions (look here). Otherwise you can do it manually (**discouraged for jupiter notebook**, in the following seconds), look here. On bitbucket you can do it online (look here).
   Nevertheless, if the project is shared, good manners imply asking collaborators in advance ;)

IMT SCHOOL FOR ADVANCED STUDIES LUCCA

Jupyter Notebook



From Jupyter notebook documentation:

For those familiar with Mathematica, jupyter notebooks are nothing but the notebook version of python.

Notebooks are pieces of interactive computations, where you can add comments, images and, more or less, whatever you can think it could be important for presenting your result.

When you launch a jupyter notebook, on your default browser a working tree representation is open.
Then you can access to your notebooks, i.e. .ipynb files.

IMT SCHOOL FOR ADVANCED STUDIES LUCCA

Installation

Installation instructions

1. Go to your conda environment and activate it.
2. Digit `python -m pip install --upgrade pip`
3. Then, digit `python -m pip install jupyter`
4. Now jupyter is installed! Let's add some superuseful extensions.
   Digit `pip install jupyter_contrib_nbextensions`
5. Now, digit
   `jupyter contrib nbextension install - -sys-prefix`
   The last part is needed for virtual environments as conda ones
6. We are not satisfied still. In order to easily activate different extensions
   `pip install jupyter_nbextensions_configurator`
7. Finally,
   `jupyter nbextensions_configurator enable - -sys-prefix`
8. Digit
   `jupyter notebook`
9. Close your eyes and press enter,
   you'll wake up in the magic kingdom of python!