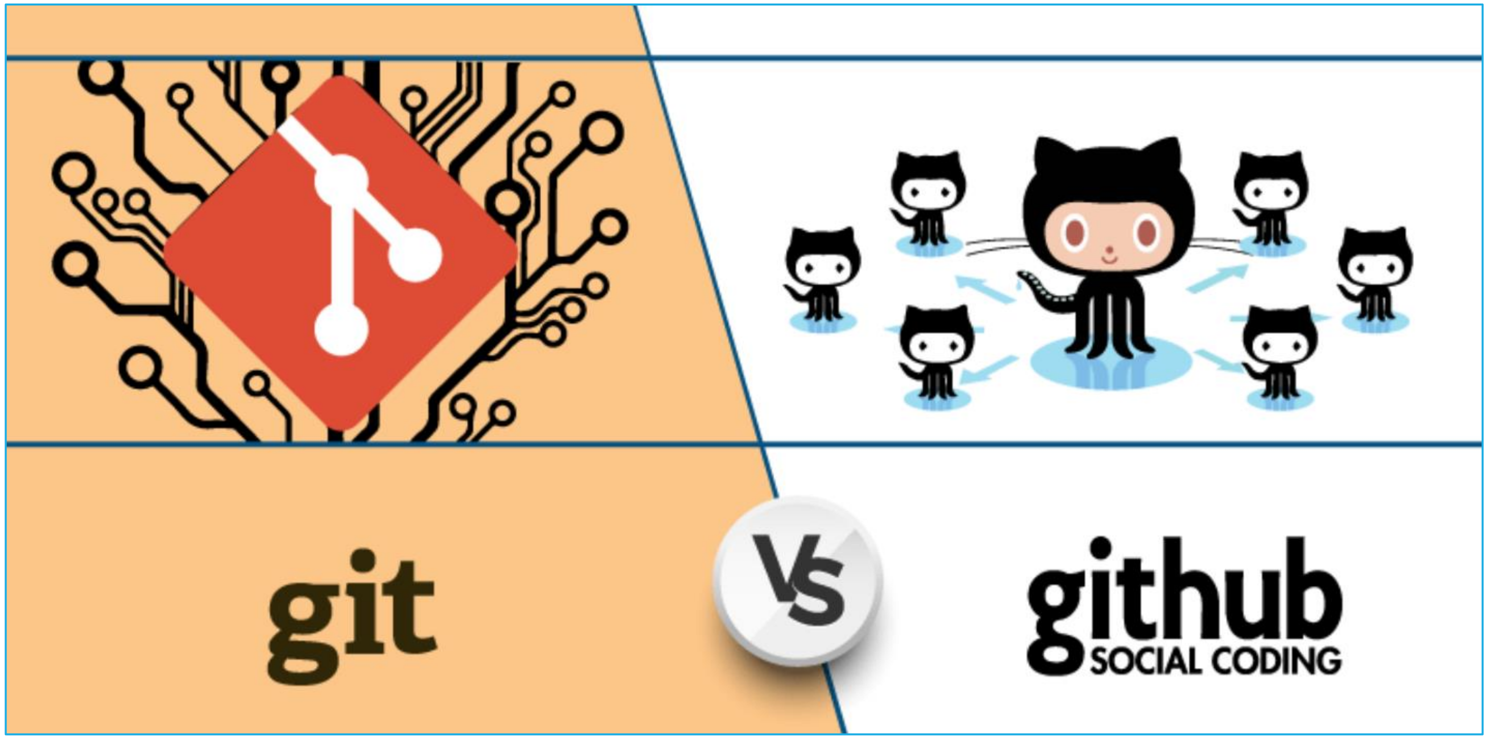
PROJECT VERSIONING

Antonio Luca Alfeo

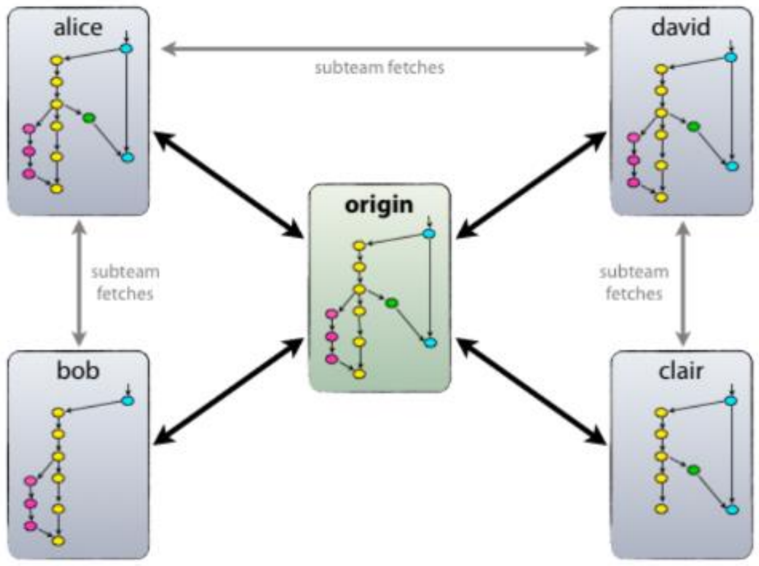
GIT AND GITHUB



Partially based on the lecture of F. Galatolo

GIT

Git is a **distributed Version Control System** in which each partecipant can have different view of the project.



REPOSITORY

A Repository (or “**repo**”) is a data structure containing all project’s **files and history**. You can **clone** a remote repo with the clone command:

```
git clone <repo url>
```

In a repo some files starting with “.git”, those are special files and folders used to store the project history and to instruct git.

.git folder containing the project history (*do not touch*!)

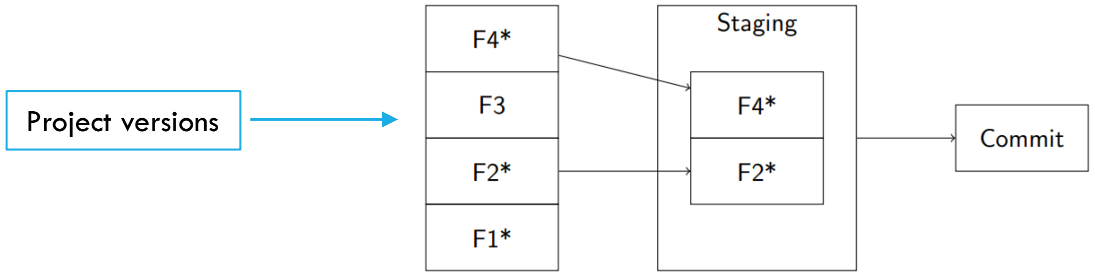
.gitignore file containing ignoring rules

.gitmodules file containing submodules information

COMMIT

A commit is immutable and atomic units of modification within a project, a snapshot of the project in a given time, it is uniquely identified by its hash and can be commented by a message. A git commit is a two-stages process.

- Add files to the staging environment `git add file/folder`
- Commit the changes `git commit -m "commit message"`



HEAD is a Git variable that points to the most recent commit. HEAD can be changed as it follows, the second option change also the files in your repo:

```
git reset <NEW_HEAD>
```

```
git reset --hard <NEW_HEAD>
```

Tags allows pointing to specific commits that represent milestones.

BRANCH... ...AND MERGE

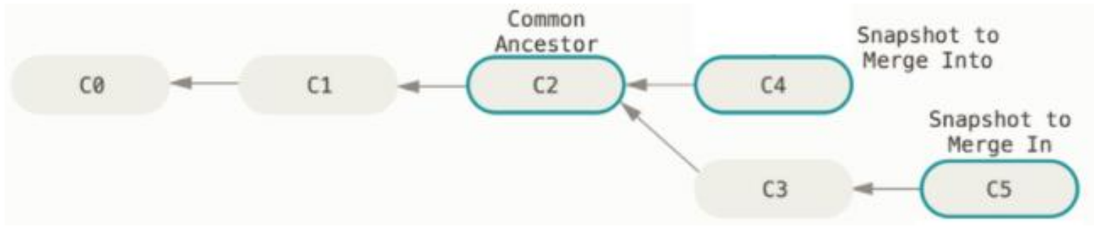
A **branch** is a semantically significant collection of commits. A commit is **always** in a branch. As a commit represent an atomic unit of modification, a branch represent a continuous flow of modifications.

- `git branch`
  - Show all the existing branches and the active one (denoted with an asterisk)
- `git branch <branch>`
  - Create a new branch called <branch>
- `git checkout <branch>`
  - Switch working on the branch <branch>

Different branches (or even single commits) can be **merged** in order to transfer all the changes developed in different branches.

All you have to do is **check out** the branch you wish to merge into and then run the git merge command:

- `git merge <branch>`
  - Merges <branch> in the active branch
- `git merge –abort`
  - Abort the merge and restore everything as it was before git merge



HANDLING MERGE CONFLICTS

If the commits changed different files or different sections of the same files the merge is **without conflicts**, *as it may happen in your project since each one of you should work on a different module!*

If the commits changed the same sections of the same files (at least once) the commit is **with conflict**.

Handling conflicts is pretty easy. When a conflict is detected git puts in the place of the conflict:

```
Some non-conflicting text
<<<<<<< HEAD
CONFLICTING PORTION COMING FROM HEAD
=======
CONFLICTING PORTION COMING FROM bugfix
>>>>>>> bugfix
Some other non-conflicting text
```

Is up to you keep the portions that you want and then git add and git commit the changes

REMOTES

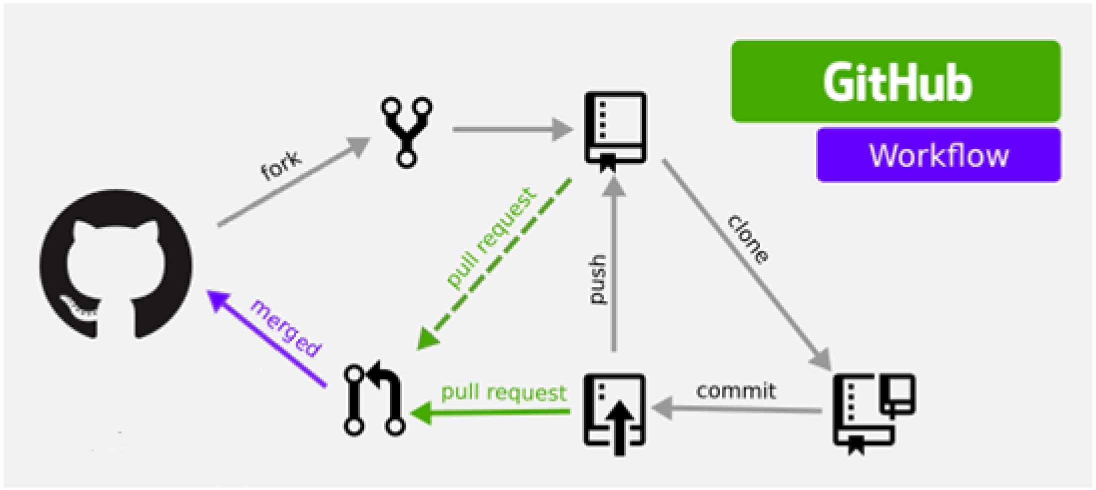
Git is distributed. A **remote** is a reference to a remote instance of the repository. The default remote is called **origin**: if you clone a repository then origin points to the cloned repo.

- `git fetch <remote> <branch>`
  - Fetch the commits from branch <branch> of <remote> in the local branch <remote>/<branch>
- `git pull <remote> <branch>`
  - Fetch the commits from branch <branch> of <remote> and merge them in the local <branch>
- `git push <remote> <branch>`
  - Push the state of the local branch to the origin branch <branch>

GITHUB

- GitHub is Git server.
- Used by over 65 Million Users
- Contains over 200 Million Repositories
- Largest source code host in the world

MORE ON GITHUB: FORK AND PULL



In GitHub any public repository can be **Forked**. After a fork you own an exact copy of that repository into yours (you are the administrator). A fork is not a Git clone.

If you want to ask for the integration of your changes in the forked repository you can create a **Pull Request**. In the pull request you have to specify the destination branch (original repository) and the source branch (your repository).

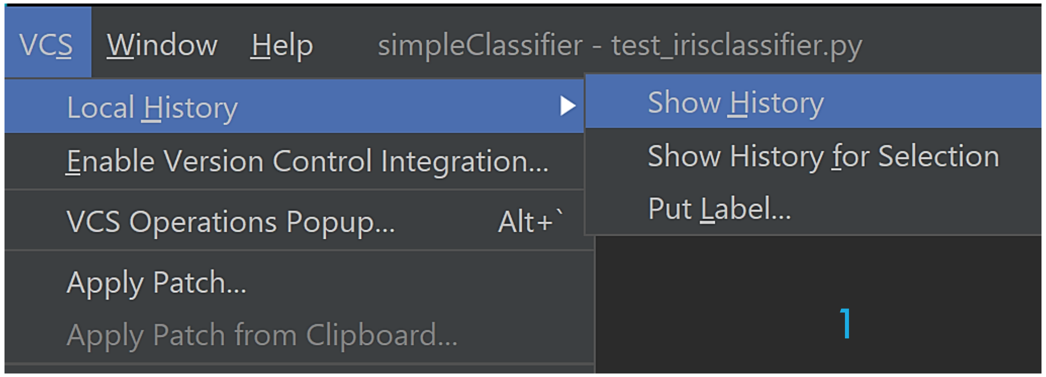
PYCHARM VCS

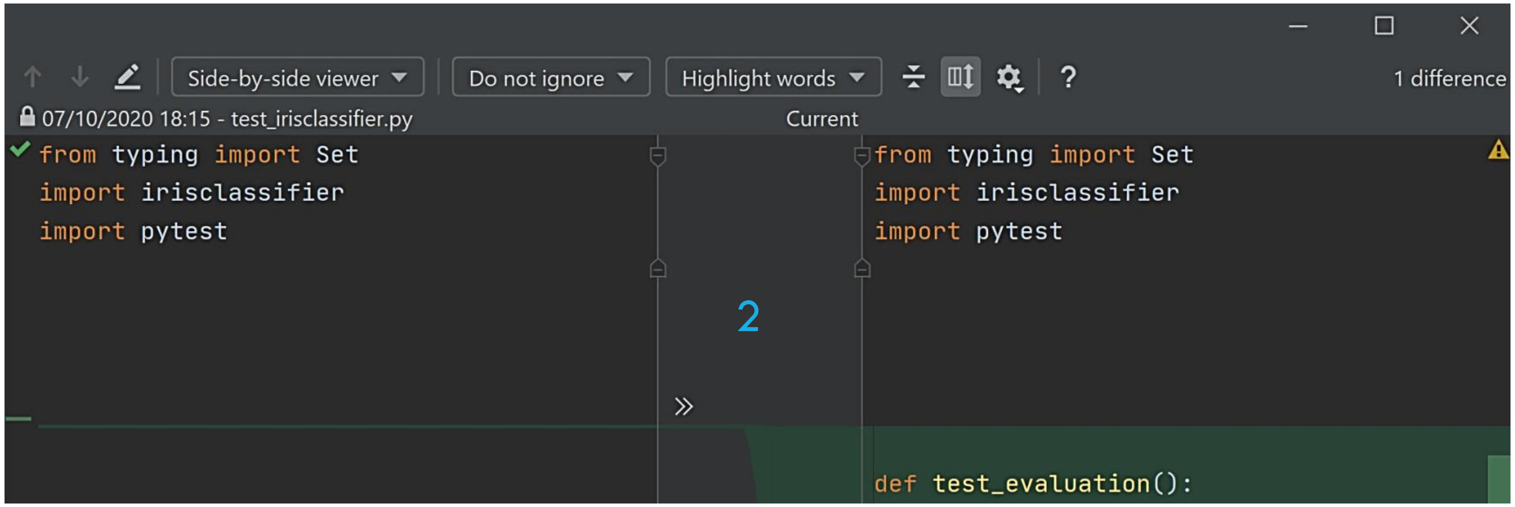
Based on previous lecture by A. L. Alfeo

BASIC VERSION CONTROL

In Pycharm the Version Control can be accessed locally:

1. via VCS > Local History
2. Stores only local and recent change in the project highlighting them


VCS
Window
Help
simpleClassifier - test_irisclassifier.py
Local History
Enable Version Control Integration...
VCS Operations Popup... Alt+`
Apply Patch...
Apply Patch from Clipboard...
Show History
Show History for Selection
Put Label...
1



```
from typing import Set
import irisclassifier
import pytest
```

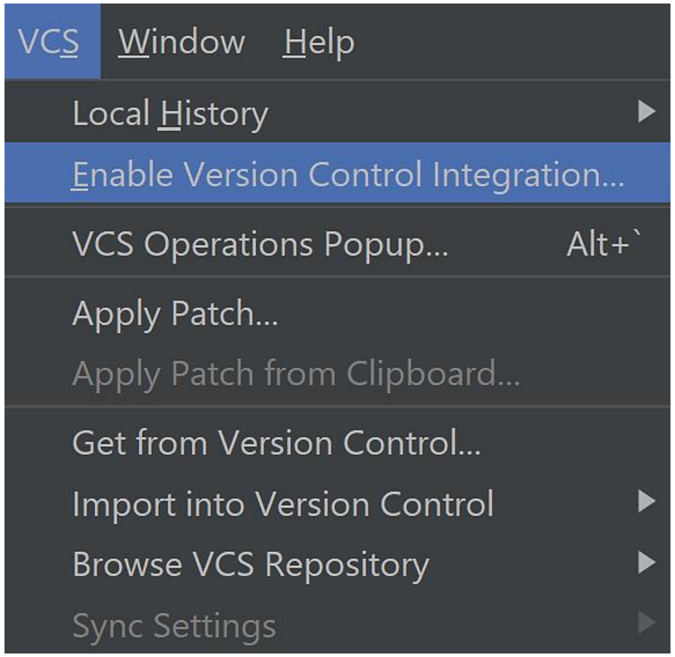
```
from typing import Set
import irisclassifier
import pytest


def test_evaluation():
```

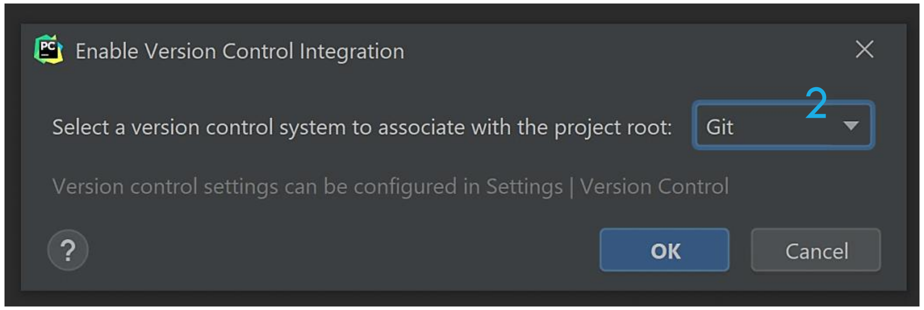
MANAGE PROJECTS WITH GITHUB

Github is way more convenient especially when a project is developed in group. Now you need to:

1. Install **Git** using only the default settings in the installation process
2. After VC integration is enabled (1 and 2), PyCharm will ask you whether you want to share project settings files via VCS.
3. In the dialog you can choose "**Always Add**" to synchronize project settings with other repository users who work with PyCharm.

1


VCS
Window
Help
Local History
Enable Version Control Integration...
VCS Operations Popup... Alt+`
Apply Patch...
Apply Patch from Clipboard...
Get from Version Control...
Import into Version Control
Browse VCS Repository
Sync Settings

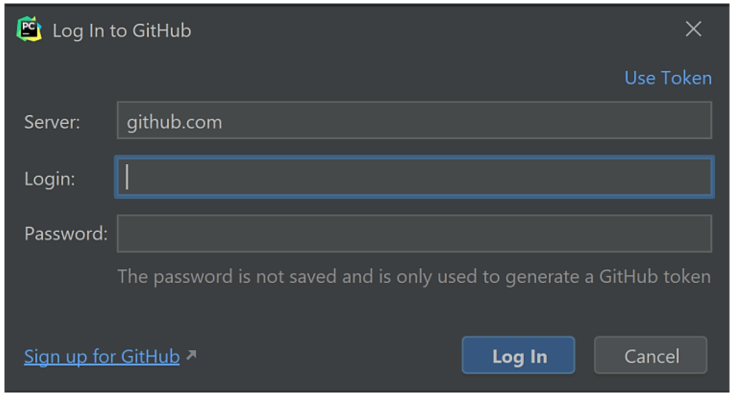


Select a version control system to associate with the project root: Git

Version control settings can be configured in Settings | Version Control

ENABLE VCI TO USE GITHUB

Login into GitHub, or request a new access token with your login and password (1). Then, authorize JetBrain IDE integration (2 and 3).


Log In to GitHub

Use Token

Server: github.com

Login:

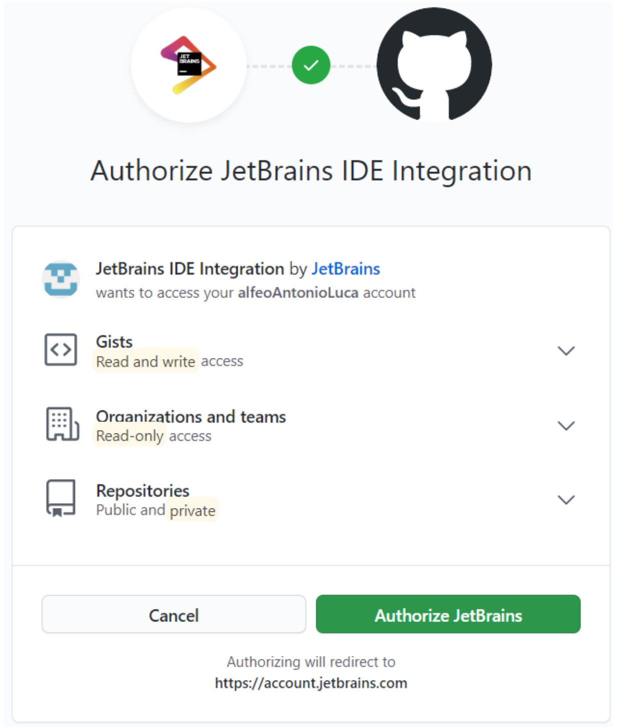
Password:

The password is not saved and is only used to generate a GitHub token

Sign up for GitHub ↗

Log In | Cancel

1


Authorize JetBrains IDE Integration

JetBrains IDE Integration by JetBrains
wants to access your alfeoAntonioLuca account

Gists
Read and write access

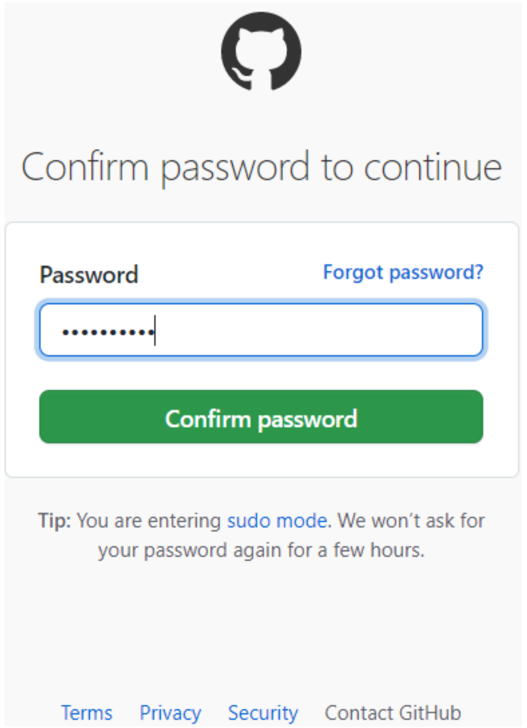
Organizations and teams
Read-only access

Repositories
Public and private

Cancel | Authorize JetBrains

Authorizing will redirect to
https://account.jetbrains.com

2


Confirm password to continue

Password — Forgot password?

••••••••••

Confirm password

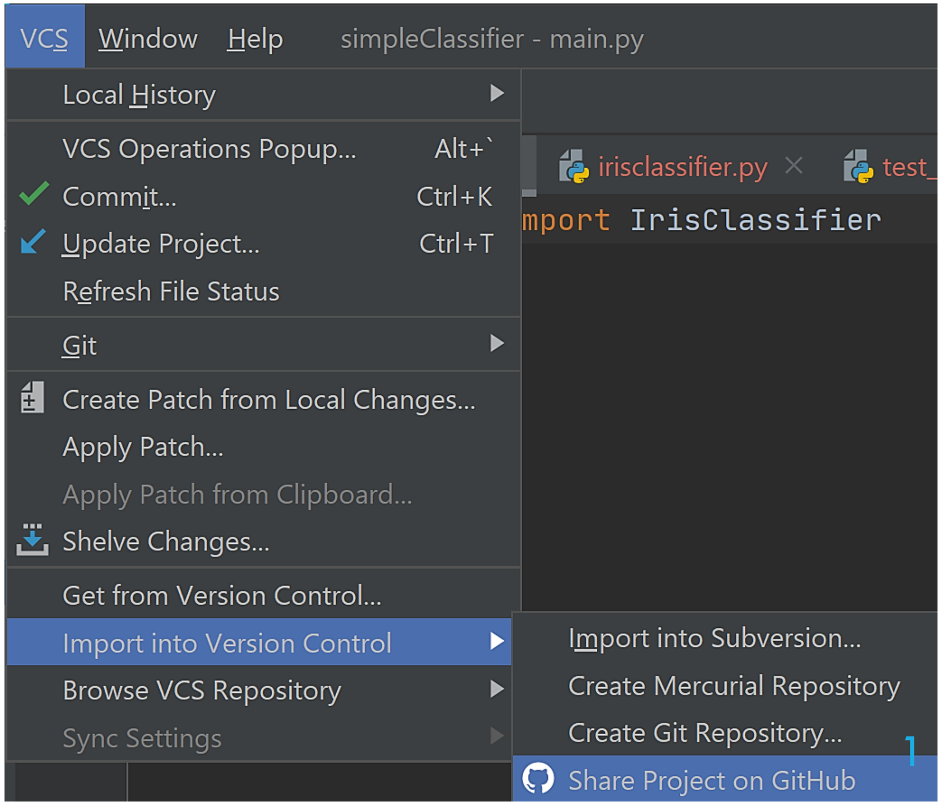
Tip: You are entering sudo mode. We won't ask for your password again for a few hours.

Terms Privacy Security Contact GitHub

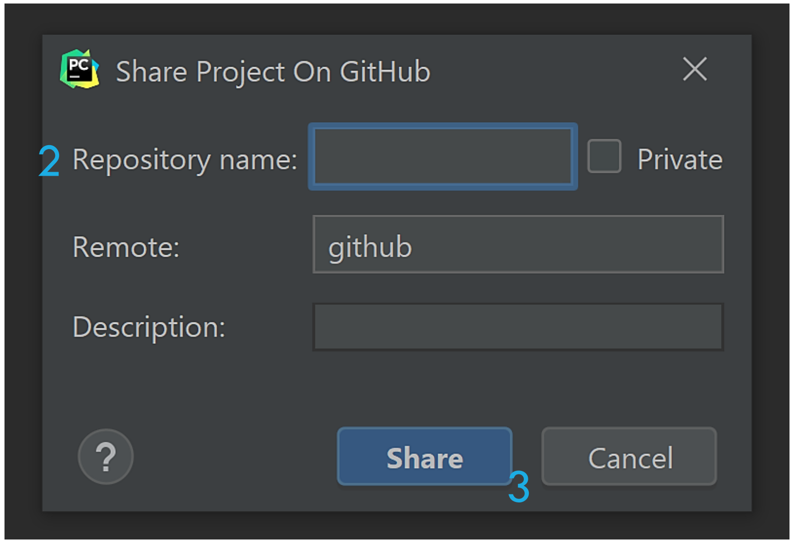
3

SHARE A PROJECT 1/2

1. When connection to GitHub has been established, the Share Project on GitHub dialog appears in VCS > Import Version Control > Share …(1).
2. Name the repository (2) and click share (3)


VCS
Window
Help
simpleClassifier - main.py
Local History
VCS Operations Popup... Alt+`
Commit... Ctrl+K
Update Project... Ctrl+T
Refresh File Status
Git
Create Patch from Local Changes...
Apply Patch...
Apply Patch from Clipboard...
Shelve Changes...
Get from Version Control...
Import into Version Control
Browse VCS Repository
Sync Settings
Import into Subversion...
Create Mercurial Repository
Create Git Repository...
Share Project on GitHub
1
irisclassifier.py
test_

```
mport IrisClassifier
```

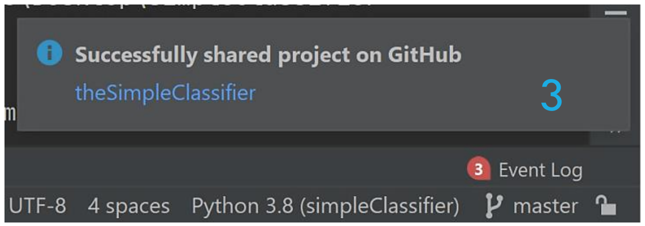


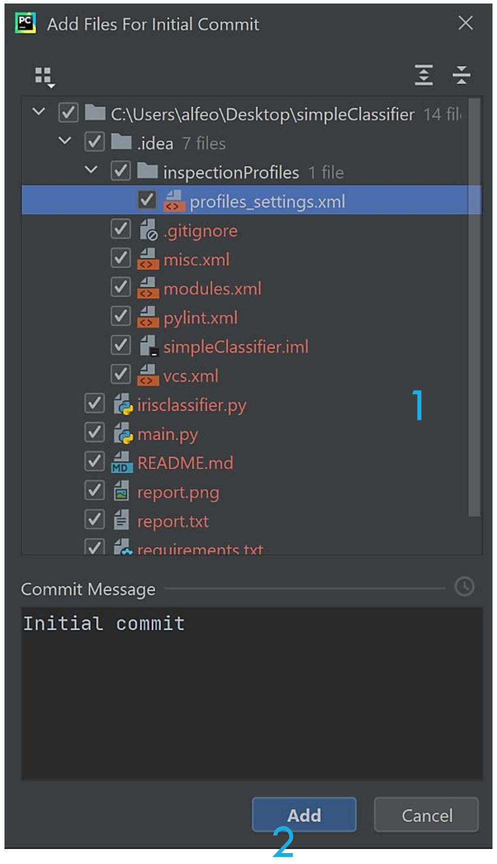
SHARE A PROJECT 2/2

1. Being the initial commit, of the project every file is highlighted in red (they are not in the VCS).

2. Click Add

3. PyCharm will notify once the process is finalized


Successfully shared project on GitHub
theSimpleClassifier

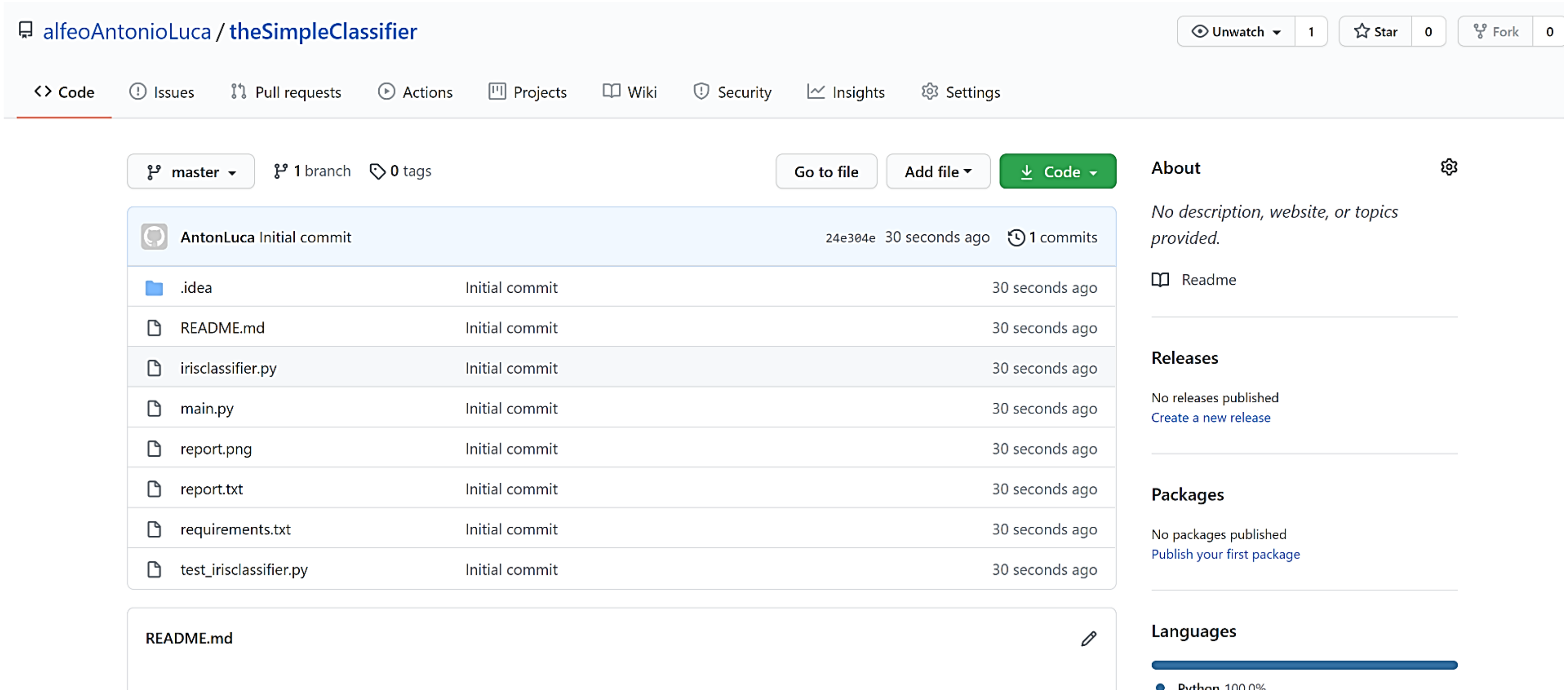

Add Files For Initial Commit

```
C:\Users\alfeo\Desktop\simpleClassifier 14 fil
  .idea 7 files
    inspectionProfiles 1 file
      profiles_settings.xml
    .gitignore
    misc.xml
    modules.xml
    pylint.xml
    simpleClassifier.iml
    vcs.xml
  irisclassifier.py
  main.py
  README.md
  report.png
  report.txt
  requirements.txt
```

Commit Message

```
Initial commit
```

… ON GITHUB


alfeoAntonioLuca / theSimpleClassifier

AntonLuca Initial commit 24e304e 30 seconds ago 1 commits

.idea	Initial commit	30 seconds ago
README.md	Initial commit	30 seconds ago
irisclassifier.py	Initial commit	30 seconds ago
main.py	Initial commit	30 seconds ago
report.png	Initial commit	30 seconds ago
report.txt	Initial commit	30 seconds ago
requirements.txt	Initial commit	30 seconds ago
test_irisclassifier.py	Initial commit	30 seconds ago

README.md

About

No description, website, or topics provided.

Readme

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

Python 100.0%

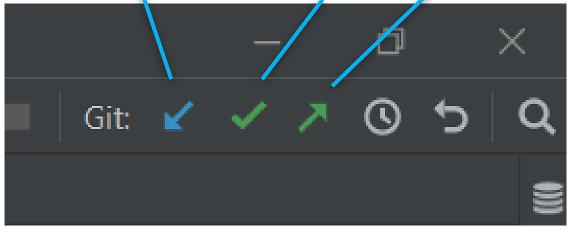
Let's say you have a **central repository** for your project, the contribution of each coder will be highlighted by the commits she/he made. In the next slide you will see how to "**sign**" your commit °

COMMIT AND PUSH

Update (pull)

Commit

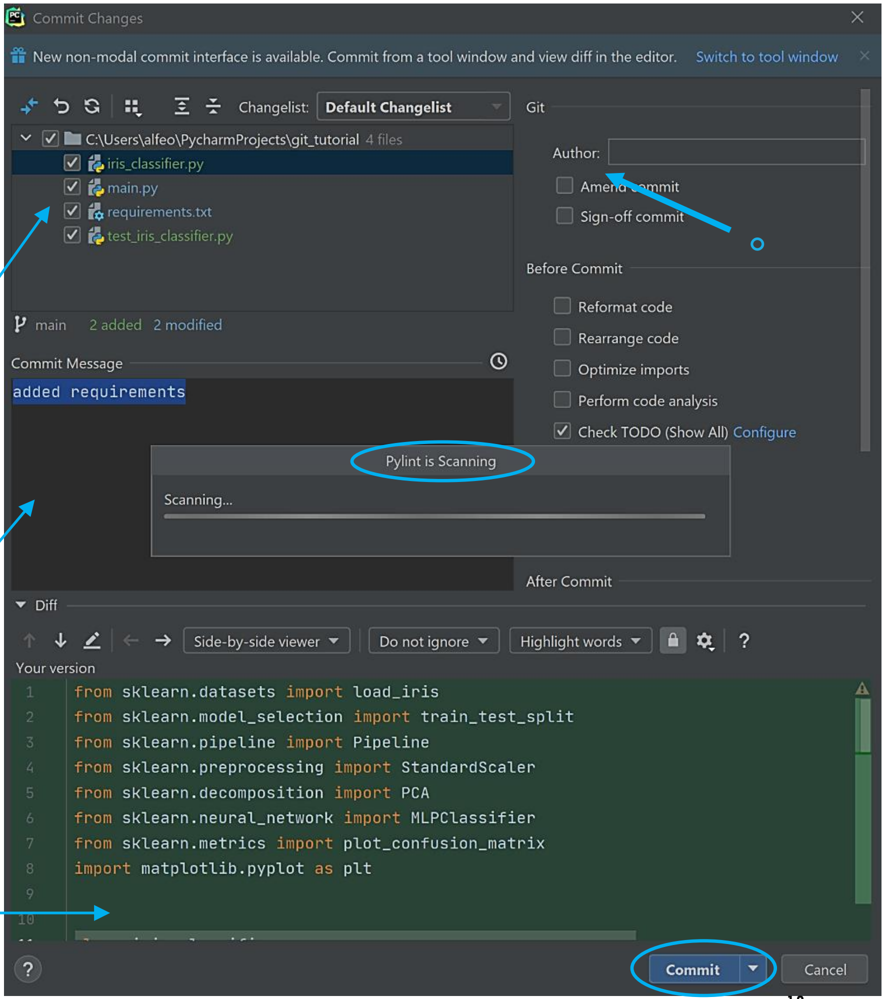
Push



Color code:
- Blue: modified
- Green: commit done
- Red: not in the VCS
- White: push done

Explain clearly the change provided

After the first commit here there will be the "before and after" comparison

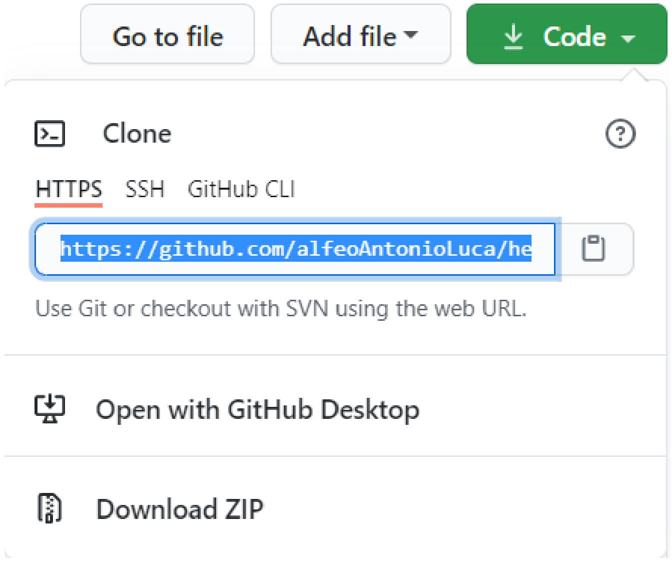


```
added requirements
```

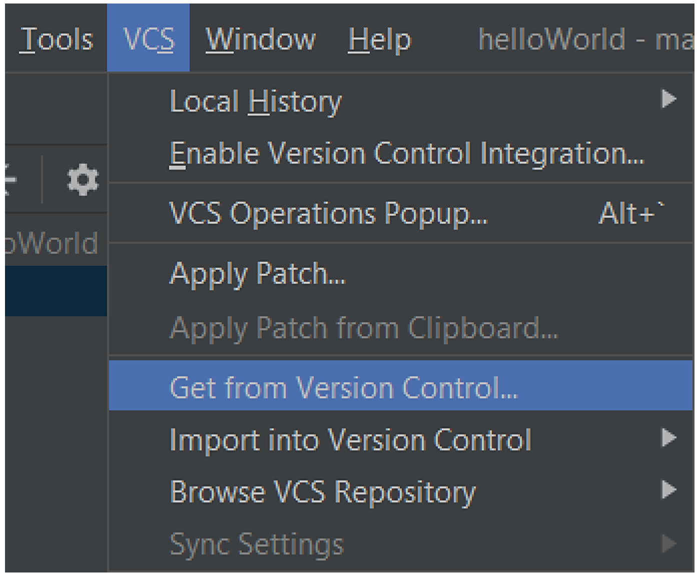
```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import plot_confusion_matrix
import matplotlib.pyplot as plt
```

CLONE FROM GIT 1/2

1. Go to GitHub and annotate the URL of the repository you are interested in (i.e. the project repo we will provide you with).
2. From the main menu, choose VCS > Get from Version Control, and choose GitHub on the left.

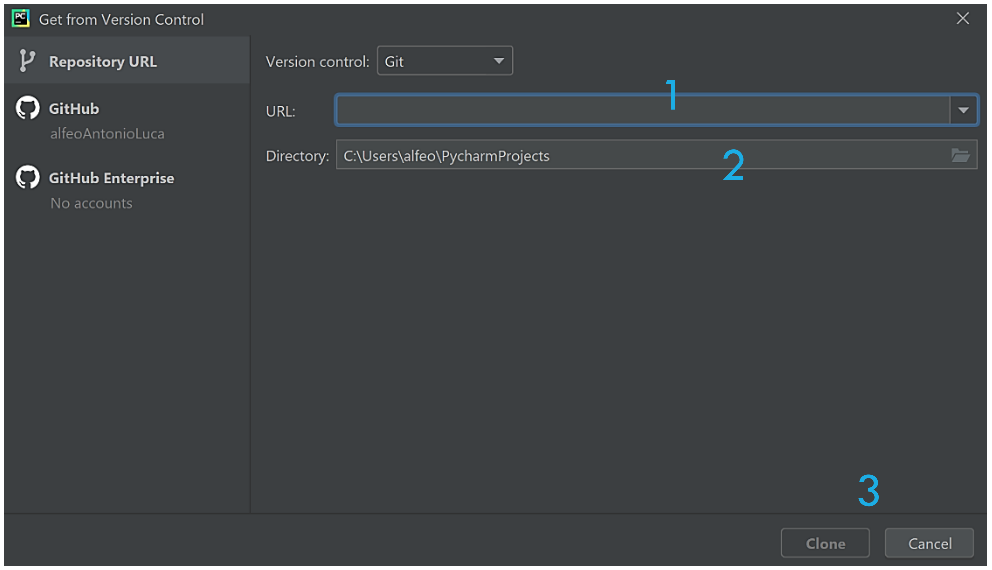


Go to file
Add file
Code
Clone
HTTPS
SSH
GitHub CLI
https://github.com/alfeoAntonioLuca/he
Use Git or checkout with SVN using the web URL.
Open with GitHub Desktop
Download ZIP



CLONE FROM GIT 2/2

1. Specify the URL of the repository that you want to clone (the one you annotated from GitHub).
2. In the Directory field, enter the path to the folder where your local Git repository will be created.
3. Click **Clone**. If you want to create a project based on these sources, click Yes in the confirmation dialog. PyCharm will automatically set Git root mapping to the project root directory.


Get from Version Control

Version control: Git

URL:

Directory: C:\Users\alfeo\PycharmProjects

MORE ON PYCHARM VCS

PyCharm provides **version control integration**, a unified user interface for many VCS such as **GithHub**.

It allows you to contribute to a GitHub project via commit, push, pull, change lists, cloning… and even branch, merge, and conflict via Git.

Here you have a couple of video-tutorials [1, 2, 3]

QUESTIONS?