GIT VERSION CONTROL

GITHUB

Let us access a free public shared repository. To this end, link to `https://github.com/` , sign up to the free version and write down account and password. The account gives you the possibility to push commits to main repository `https://github.com/ADUCOLI/PyFinLib`.

GIT SOFTWARE

To exploit Git utilities, set up the machine to the latest version of this freeware software. First, download from `http://git-scm.com/downloads` the target version. During installation framework, we follow the basic configuration, that is flag the items:

1. Git Bash use
2. Open SSH
3. Checkout windows style

Once installation is successfully completed, we are able to use all git commands through Git Bash command line. For an optional user friendly GUI, install Tortoise Git `https://code.google.com/p/tortoisegit/` .

An Introduction to GIT

Let us briefly introduce Git for newbies. A versioning controller is useful for working group to parallelize code implementation, through utilities of automated merging, frequently releasing and versioning of the library. Moreover, it allows to works on multiple different branches in order to fix bugs, explore different solutions from master implementation and test start-up features. The main difference between Git and other freeware competitors, is the underlying versioning concept: each member of the team stores the whole repository, with the log story, the previous versions and branches, as a filesystem. Thence, it is able to modify, work and query the local repository with no reliance on internet connection. In fact, each version is a screen shot of the code, and lightness is achieved storing differences between files instead of a complete copy, employing a check-sum strategy. Once the working copy has a definitive version, we should commit the changes to backup and then start aligning to master repository. In a multi thread framework, where other members frequently push their versions, one should select the pull button to align the local repository to remote master, which performs a fast-forward merge from remote to local and put the user in charge of eventually resolve the conflicts. Once merging is performed, push it to remote. The same idea works for branches, were the alignment to local or remote master is performed through merge button. Let us now recap the main Git features:

- Clone: download the repository from remote
- Add: move files from unstaged area to target commit list
- Commit: backup point for local working copy
- Pull: align remote master to local one
- Push: update remote master using working copy
- Switch/Checkout: switch to a different branch or a previous release
- Branch: create a branch from the working copy
- Merge: update a target branch with the whole features of a different one
- Revert: restore an uncommitted file to the previous version

Setup PyFinLib Repository

Then, let us clone the remote repository to local filesystem. Select the target folder where Git should download the data, using Git Bash for a command line front end or Windows Explorer in case of Tortoise's GUI. Select Clone Repository and insert the over mentioned link. The PyFinLib folder contains the latest version of the code and above all a configuration hidden folder named .git and a configuration file named .gitignore to setup an automatic exclusion of trash files from release process.

Once a local repository is cloned, let us consider a case study. For example, the insert of a new file. First of all, right-click using Tortoise Git and select add button to move it from unstaged (not versioned) area to staged one, which would triggers its deployment once commit is done. To back up, commit you local version to the target branch using the proper command and insert a useful comment. Once commitment is completed, your log story is updated with a new field. To send the differences, pull from origin and then push it to remote repository, if administrative permissions are correctly set. Thus, insert the GitHub user account and the password to complete the process. To avoid a continuous insertion of credentials for each pushing, let us modify local PyFinLib settings, through Settings button, then Git/Remote/Origin and in the URL field insert the modified string `https://:@github.com/ADUCOLI/PyFinLib` . Enjoy!