Development workflow

Document history

Date	Modification(s)
10/13/2019	Creation

Table of content

Document history 1

Table of content 1

Document overview 2

Quick definition of the workflow 3

In depth definition of the workflow 4

Files and folders organization 5

Document overview

The goal of this document is to standardize how the project's development will be handled. It is divided into two main parts :

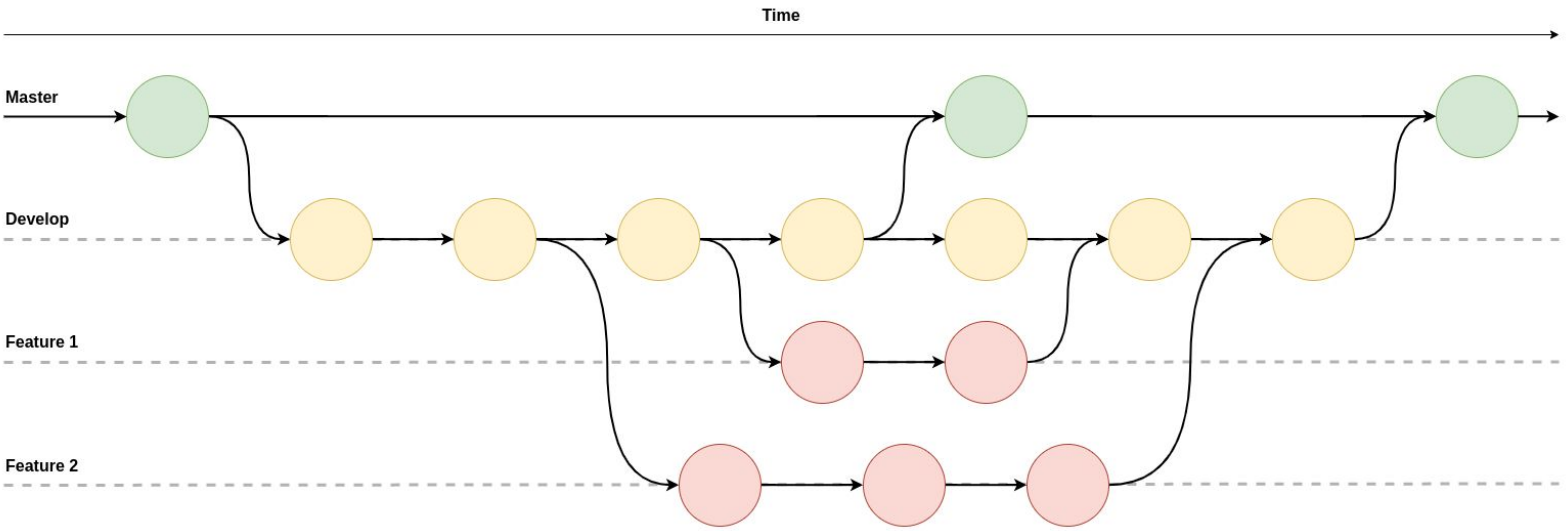
1. Quick definition of the workflow thanks to a visual support.
2. Textual and in depth definition of the workflow.

The used workflow is largely inspired by the GitLab Flow.

This document also describes how the different files are organized inside the various folders.

Disclaimer: every organization instructions described in this document can change regarding the usability of the latter. Hence, this document can change over time.

Quick definition of the workflow



While working on the project, there will be two mains branches :

- **Master branch:** at every moment, someone cloning this branch must have a working copy of the project.
- **Develop branch:** this will be the main working branch. Every feature branches will be derived from this one.
- **Feature branches:** each distinct feature must have its own branch.

In depth definition of the workflow

The most important rule here concerns the constant availability of a master branch containing a fully working program. At every moment, from the beginning of the development phase to the final delivery, everybody who wants to clone the project must obtain a clean and working copy of the program through the master branch. The first commit done to the master branch at the moment of the GitHub repository creation marks the beginning of the version 0. After that, each update of the master branch from the develop branch will increment the version number of the project. In order to merge the develop branch into the master one, all developers must agree on the cleanliness and the working aspect of the current develop branch.

The develop branch will contain every unstable change made on the project. This branch represents the main working branch of the project. Every small change made to the project are done here. Here is a non exhaustive list of change that can occur in the develop branch :

- adding a documentation file
- rearrange file organization
- fixing mistakes on functions interoperability

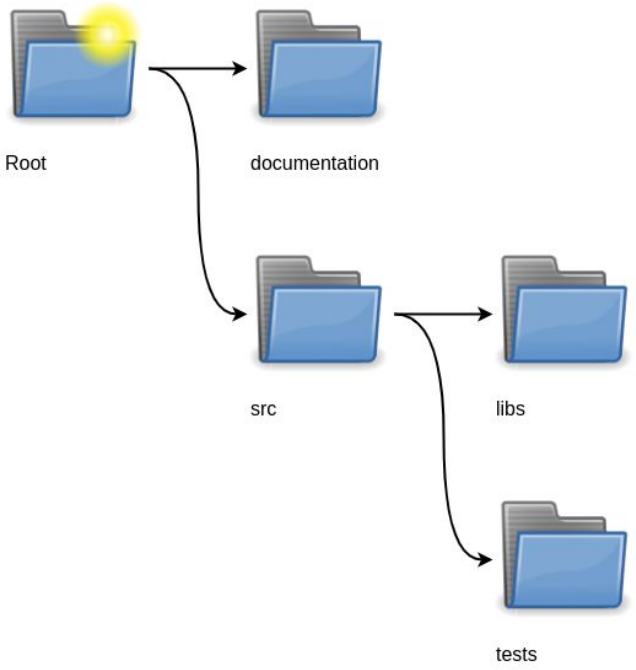
For every change related to a bigger functionality, use the feature branches.

The different feature branches permit to separate the development of distinct functionality that may be added to the program (e.g. adding a major function to write a result to a file). In some case, a functionality can be tested in a feature branch without merging it to the develop branch later. When adding one of these functionality to the main program, the feature branch must be merged to the latest develop branch and not the master one.

No matter the branch you are working on, try to often commit in order to reduce the change caused by a backtrack.

Files and folders organization

In order to clarify the project organization, the following folder structure will be used :



From the root folder, two folders can be found :

- documentation: contains all documentation files related to the development of the project or related to used frameworks, libraries…
- src: holds sub directories containing source files.
    - libs: the developed program is an API so most of the source files will be stored here.
    - tests: every source code used to test the developed API will be stored here.