KaBook
Configuration and Change Management Report

1 Introduction

Configuration Management is a process to systematically manage, organize, and control the changes in the documents, codes, and other entities during the Software Development Life Cycle. It is abbreviated as the SCM process in software engineering. The primary goal is to increase productivity with minimal mistakes.

Any change in the software configuration Items will affect the final product. Therefore, changes to configuration items need to be controlled and managed.

2 Purpose

- There are multiple people working on software which is continually updating
- It may be a case where multiple version, branches, authors are involved in a software project, and the team is geographically distributed and works concurrently
- Changes in user requirement, policy, budget, schedule need to be accommodated.
- Software should able to run on various machines and Operating Systems
- Helps to develop coordination among stakeholders
- Configuration Management process is also beneficial to control the costs involved in making changes to a system.

3 Configuration and Change Management Specifications

- The main reason behind configuration management process is that there are multiple people working on software which is continually updating. Configuration Management helps establish concurrency, synchronization, and version control.
- A baseline is a formally accepted version of a software configuration item
- Change control is a procedural method which ensures quality and consistency when changes are made in the configuration object.
- Configuration status accounting tracks each release during the Configuration Management process
- Software Configuration audits verify that all the software product satisfies the baseline needs
- Project manager, Configuration manager, Developer, Auditor, and user are participants in Configuration Management process
- The Configuration Management process planning begins at the early phases of a project.
- Git, Team foundation Sever and Ansible are few popular Configuration Management tools

Change management software should have the following three important points:

3.1- Concurrency Management:

When more tasks are working at the same time, it is known as concurrent operation. Concurrency in context to Configuration Management means that the same file being updated by multiple useres at the same time.

If concurrency is not managed correctly with this tools, then it may create many pressing issues.

3.2- Version Control:

Configuration and Change Management uses archiving method or saves all changed files. With the help of archiving or save feature, it is possible to roll back to the previous version in case of issues.

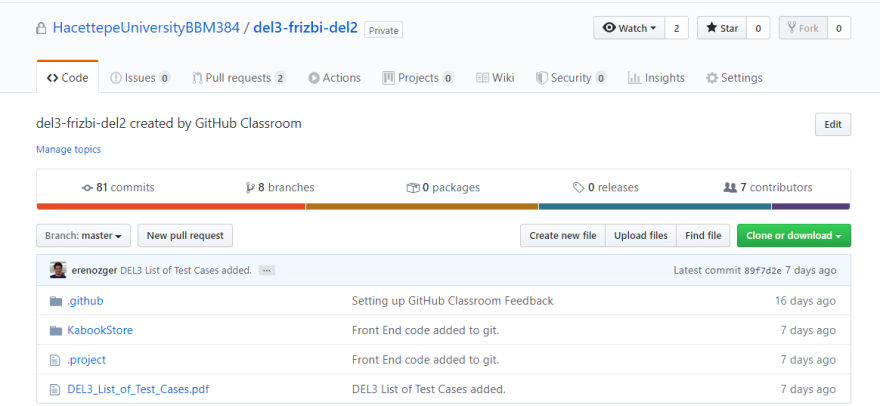
3.3- Synchronization:

Users can checkout more than one files or an entire copy of the repository. The user then works on the needed file and checks in the changes back to the repository. They can synchronize their local copy to stay updated with the changes made by other team members.

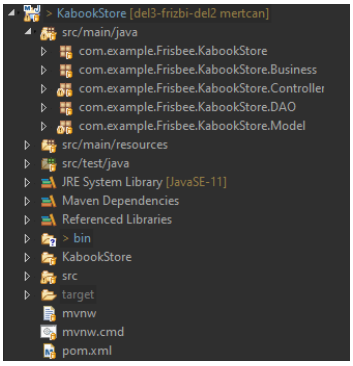
As we develop the project, we use Github to control changes and configurations. Github which is a hosting site for Git projects. So at first, we need to say about what Git is. Git is a distributed version control tool that was invented by Linus Torvalds to support the development of Linux. In April 2005 Git was used for the first time and Linus made the first commit with code for Linux. The purpose of Git is to provide a reliable and versatile version control and configuration management, it does this with a little different approach than some other version control tools. Git also enables people to work together and empowers teamwork.

4 Key Considerations

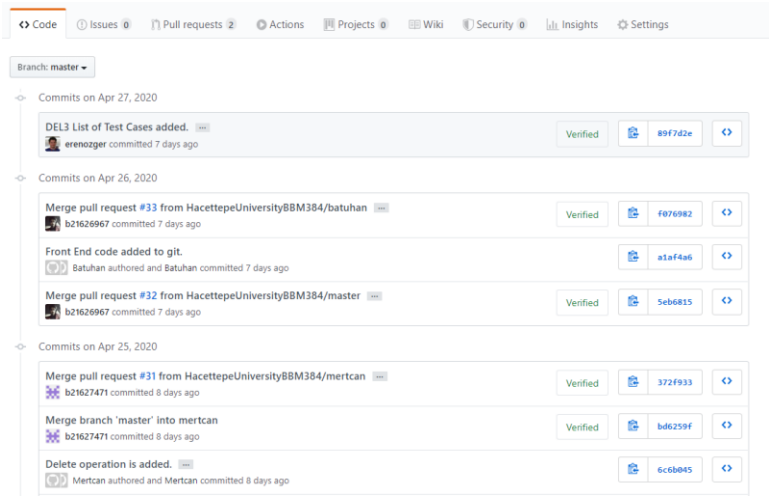
There are several key considerations for Github. To briefly explain them:



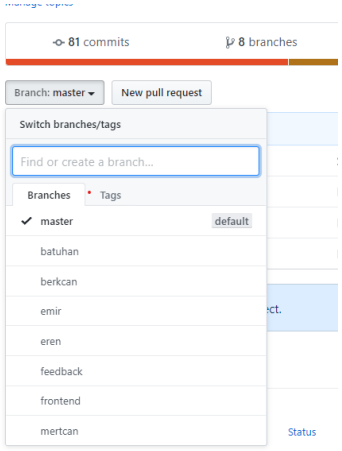
Repository: Git is built around two parts your workspace and repositories. A repository in Git is similar to repositories or depots in other tools. The repository is where the files and information in the decentralized system is stored. From a repository we can extract information about the current version and earlier versions. Our repository act as the interface to other friends we are working with as well as a powerful part in the configuration management process.



Workspace: A workspace is where you do all your work. A workspace is private to one developer and can be said to be his or her personal area. When you edit a file, you do it in the workspace and no one else is affected, since it is your workspace.

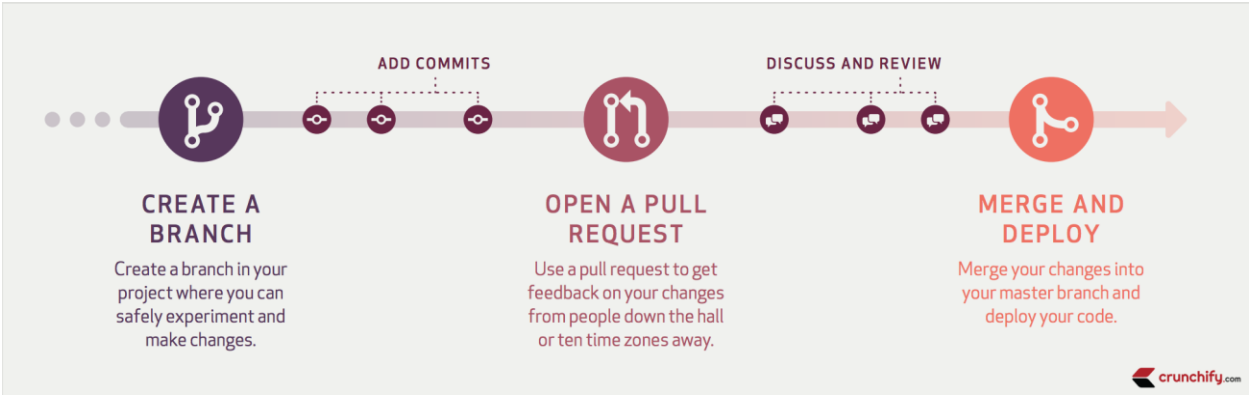


Commit/Pull/Push: When a piece of code is written the developer does a commit and the changes are saved in the private repository. By looking at two hash codes it is impossible to see which order the commits was made. An obstacle that needs to be solved by Software Configuration tools is when the user want to rename files. Git uses the content of the files to track and save changes, therefore it can easily discover if a file has been renamed rather than just tracking files by their file names.



Branches: An important concept in software configuration management is branches, and it is even more prominent when using Git as opposed to other centralized SCM tools. In Git every commit in the repository has the potential to become a branch. To be useful, branches need to have things in common, if they don't they could just as well be and entirely different software project. Branches serve the purpose of isolating tasks during development to provide a stable base to work on without distractions. Branches can then be merged together resulting in a branch with the features of both of the initial branches. Often there is a 'main-branch' in a project which is what all small branches is merged to. While we were developing the project, we all created our own branches and work in different branches.

Briefly, the logic of Github is as in the picture.



(image taken from crunchify.com)