Need for Source Code Management System

* Integration of code is easy if the project is small and the developers are co-located. In larger projects where the number of developers is more, all of them might not be located in the same place. Manually integrating the code in such a scenario becomes a time-consuming process which leads to delay in delivery of software.
* So we need a central system which should keep track of the individual developer’s work in the form of a single project. This central system is called Source Code Management (SCM) system.

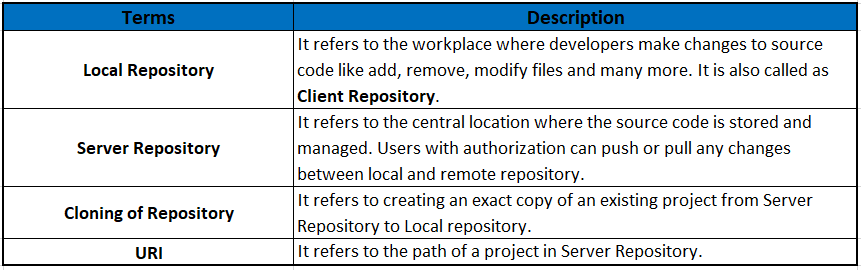
Introduction to Git - SCM System

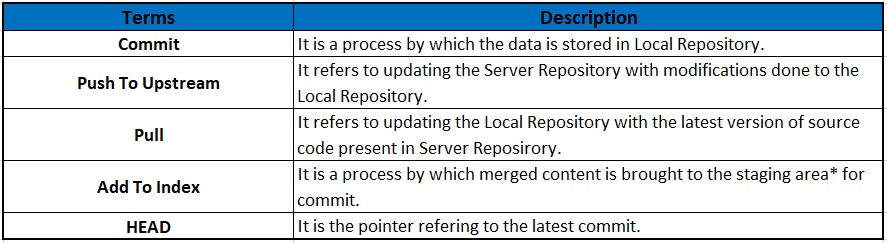
* We will learn about the most commonly used Source Code Management system Git and it has lot advantages than other SCM tools such as SVN - Subversion, TortoiseSVN, CVS and VSS.

1. Git is an open source distributed SCM tool.
2. Git provides fast collaboration as most operations are performed offline at client side.
3. Git has provision for implicit backup.
4. We do not need powerful central server.
5. It’s highly secured as it uses SSH\* keys to connect with server.

\* SSH(Secure Socket Shell) keys is a network protocol which provide a secure way to access remote server.

Terminologies in Git





File States in Git

Let us understand the Git workflow.

Git has three main states that the files can reside in : Modified, Staged, Committed.

In a Working Directory(workspace)

Modify a file, now it is in Modified state.

Add the modified files into staging area. Now file is in Staged State

In Staging Area (temporary location for modified files)

Do commit the changes to the local repository, now the file is in Committed State.

At GIT Repository (local repository)

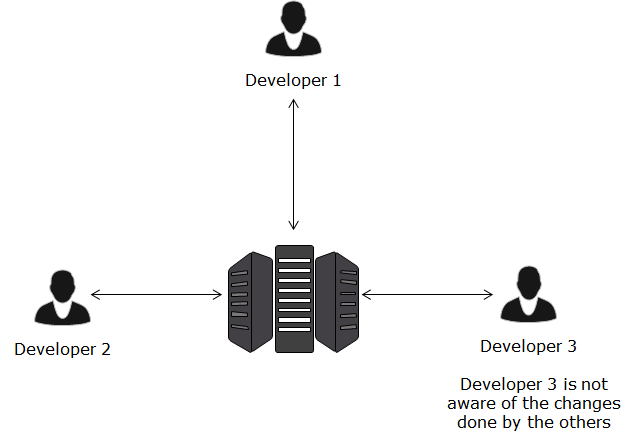
Perform push operations to update the server repository permanently.

Need for Synchronization

We have seen that the main advantage of using SCM tool is to allow multiple developers to work on the single common project in order to reduce the manual work involved while integrating the code.

But typically when the developer implements the code and tries to push changes to the server repository, he will not be knowing whether the server repository is updated by other developers or not.

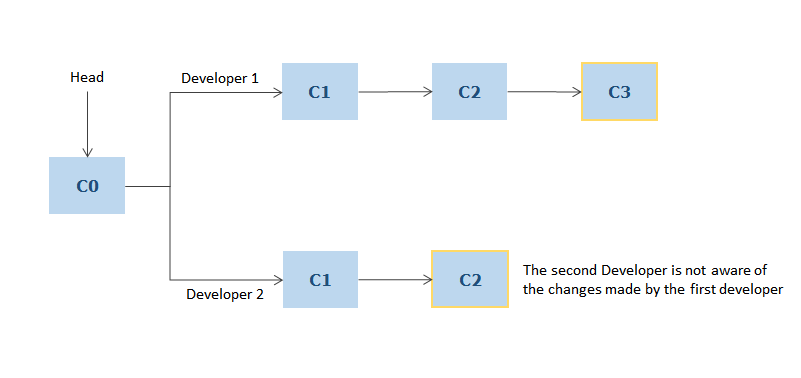
In order to check whether the server repository is updated by other developers or not, we need to synchronize the local repository with server.



Let us understand the need for Synchronization using the scenario.

Note: C1, C2, …… are indicating the Commit done by Developers.

C0 – represents initial commit given by admin.



**Scenario**:

Developer1 pushes the changes to Server Git repository. So, HEAD will be updated to C3 as shown.

