SVN - BASIC CONCEPTS

What is Version Control System?

Version Control System *VCS* is a software that helps software developers to work together and maintain a complete history of their work.

Following are the goals of a Version Control System.

- Allow developers to work simultaneously.
- Do not overwrite each other's changes.
- Maintain history of every version of everything.

A VCS is divided into two categories.

- Centralized Version Control System *CVCS*, and
- Distributed/Decentralized Version Control System *DVCS*.

In this tutorial, we will concentrate only on the Centralized Version Control System and especially **Subversion**. Subversion falls under centralized version control system, meaning that it uses central server to store all files and enables team collaboration.

Version Control Terminologies

Let us start by discussing some of the terms that we will be using in this tutorial.

- **Repository:** A repository is the heart of any version control system. It is the central place where developers store all their work. Repository not only stores files but also the history. Repository is accessed over a network, acting as a server and version control tool acting as a client. Clients can connect to the repository, and then they can store/retrieve their changes to/from repository. By storing changes, a client makes these changes available to other people and by retrieving changes, a client takes other people's changes as a working copy.
- **Trunk:** The trunk is a directory where all the main development happens and is usually checked out by developers to work on the project.
- **Tags** : The tags directory is used to store named snapshots of the project. Tag operation allows to give descriptive and memorable names to specific version in the repository.

  For example, LAST_STABLE_CODE_BEFORE_EMAIL_SUPPORT is more memorable than

  Repository UUID: 7ceef8cb-3799-40dd-a067-c216ec2e5247 and

  Revision: 13
- **Branches:** Branch operation is used to create another line of development. It is useful when you want your development process to fork off into two different directions. For example, when you release version 5.0, you might want to create a branch so that development of 6.0 features can be kept separate from 5.0 bug-fixes.
- **Working copy:** Working copy is a snapshot of the repository. The repository is shared by all the teams, but people do not modify it directly. Instead each developer checks out the working copy. The working copy is a private workplace where developers can do their work remaining isolated from the rest of the team.
- **Commit changes:** Commit is a process of storing changes from private workplace to central server. After commit, changes are made available to all the team. Other developers can retrieve these changes by updating their working copy. Commit is an atomic operation. Either the whole commit succeeds or is rolled back. Users never see half finished commit.