What are the differences between Subversion and Git?

In this article

Subversion (SVN) repositories are similar to Git repositories, but there are several differences when it comes to the architecture of your projects.

Directory structure

Each *reference*, or labeled snapshot of a commit, in a project is organized within specific subdirectories, such as `trunk`, `branches`, and `tags`. For example, an SVN project with two features under development might look like this:

```
sample_project/trunk/README.md
sample_project/trunk/lib/widget.rb
sample_project/branches/new_feature/README.md
sample_project/branches/new_feature/lib/widget.rb
sample_project/branches/another_new_feature/README.md
sample_project/branches/another_new_feature/lib/widget.rb
```

An SVN workflow looks like this:

- The `trunk` directory represents the latest stable release of a project.
- Active feature work is developed within subdirectories under `branches`.
- When a feature is finished, the feature directory is merged into `trunk` and removed.

Git projects are also stored within a single directory. However, Git obscures the details of its references by storing them in a special *.git* directory. For example, a Git project with two features under development might look like this:

```
sample_project/.git
sample_project/README.md
sample_project/lib/widget.rb
```

A Git workflow looks like this:

- A Git repository stores the full history of all of its branches and tags within the *.git* directory.
- The latest stable release is contained within the default branch.
- Active feature work is developed in separate branches.
- When a feature is finished, the feature branch is merged into the default branch and deleted.

Unlike SVN, with Git the directory structure remains the same, but the contents of the files change based on your branch.

Including subprojects

A *subproject* is a project that's developed and managed somewhere outside of your main project. You typically import a subproject to add some functionality to your project without needing to maintain the code yourself. Whenever the subproject is updated, you can synchronize it with your project to ensure that everything is up-to-date.

In SVN, a subproject is called an *SVN external*. In Git, it's called a *Git submodule*. Although conceptually similar, Git submodules are not kept up-to-date automatically; you must explicitly ask for a new version to be brought into your project.

For more information, see "Git Tools Submodules" in the Git documentation.

Preserving history

SVN is configured to assume that the history of a project never changes. Git allows you to modify previous commits and changes using tools like `git rebase`.

GitHub supports Subversion clients, which may produce some unexpected results if you're using both Git and SVN on the same project. If you've manipulated Git's commit history, those same commits will always remain within SVN's history. If you accidentally committed some sensitive data, we have an article that will help you remove it from Git's history.

Note: Subversion support will be removed from GitHub on January 8, 2024. A future release of GitHub Enterprise Server after January 8, 2024 will also remove Subversion support. To read more about this, see the GitHub blog.

Further reading

- "Branching and Merging" from the *Git SCM* book
- "Importing a Subversion repository"