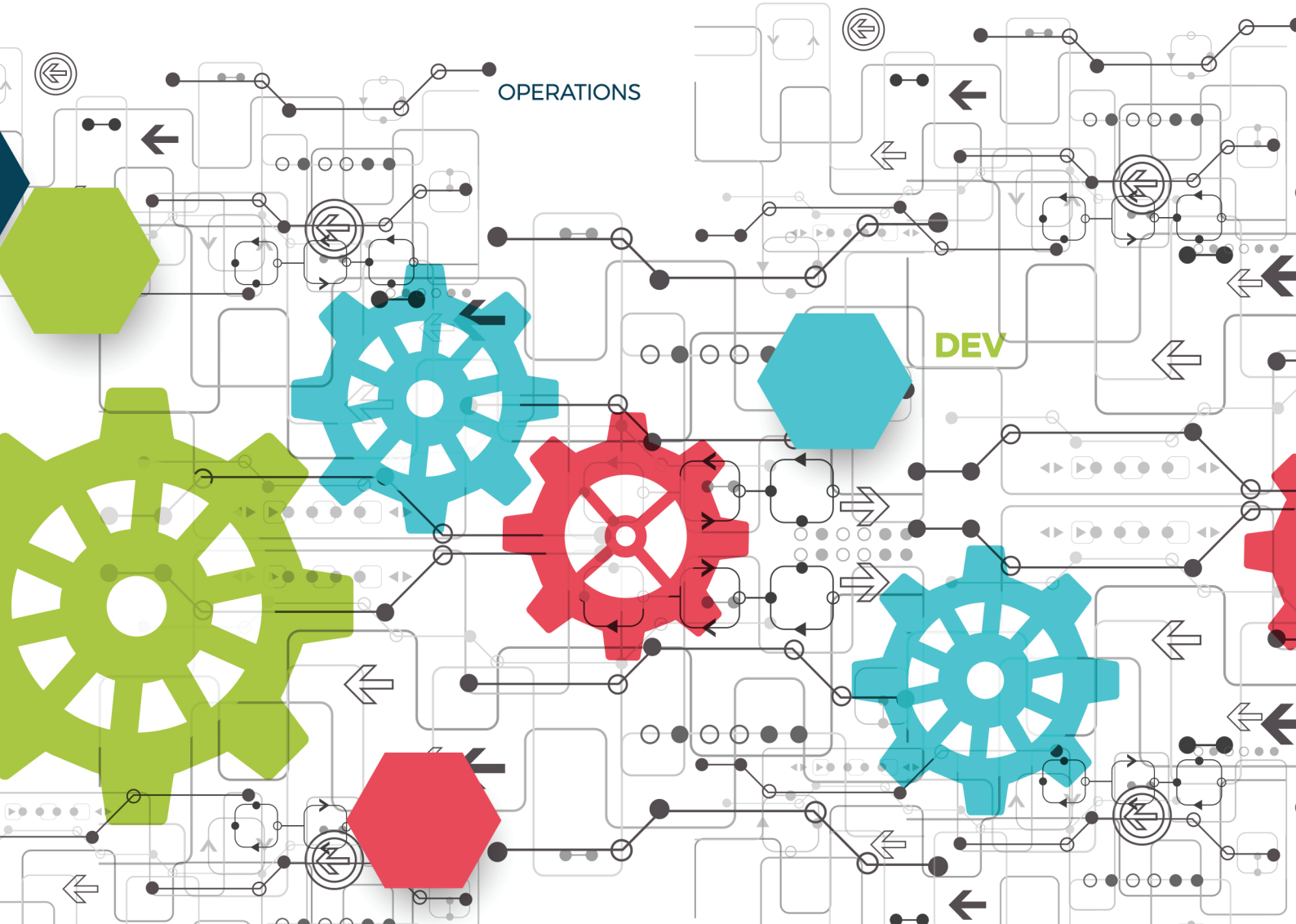
B.Tech Computer Science and Engineering in DevOps

Source Code Management

Additional Appendix
Version Control Systems



SVN – Apache Subversion

Introduction

Apache Subversion (SVN) is a free, full-featured, open-source version control system developed as a project by the Apache Software Foundation. It was created in 2000 by CollabNet Inc. The goal of Subversion was to replace CVS (Concurrent Versions System), but SVN has expanded beyond that. CVS was the go-to product for version control, but it had its own limitations and flaw. SVN creators wanted to replace CVS, hence used the same look and feel as that of CVS, in SVN, without the visible flaws that CVS possessed. SVN is thus a general system that can be used to manage any collection of files.

History of SVN

The development of Subversion as a replacement to CVS began in 2000 by CollabNet Inc. CollabNet Enterprise Edition (CEE) was a collaboration software suite offered by CollabNet, one of the components of which was a version control system. CVS was used initially, but it had to be discontinued because of the limitations. In 2000, CollabNet approached Karl Fogel, who was developing a new version control system, along with his friend Jim Blandy. In May 2000, both of them started working on the new version control system, under CollabNet, along with Ben Collins – Sussman. Many developers soon started contributing to the project actively and Subversion became self-hosting in August 31, 2001.

In 2009, CollabNet worked with the developers of Subversion to integrate it with the Apache Software Foundation (ASF). In early 2010, Subversion became one of the top-level projects and it was renamed as Apache Subversion.

Features of Subversion

1. BASIC OPERATION

The Subversion interface is similar to CVS. The primary method of interfacing with Subversion is through the command line, command-line operations of SVN are also similar to CVS. Most of

the commands in CVS have been used in SVS with the same name and a user familiar with CVS will find using SVN very simple. For features that have an CVS counterpart, Subversion uses an identical command syntax. Subversion has a lot of features compared to CVS, so many of them do not have a CVS counterpart. Subversion also belongs to the client-server paradigm, and the central repository lives in a server and the clients check out local working copies and modify the code as required. Once modifications are done, the changes between the working copy and the repository are merged and the updated version is committed back to the repository.

2. REPOSITORY FLEXIBILITY

Subversion offers a great flexibility in terms of repositories, as it keeps the revision histories of both files and directories during moves, copies and renames. Version control systems like CVS do not allow one to move, copy or rename without splitting the file's history. With CVS, one cannot move the repositories without editing the repository directly. With Subversion, we can move, rename and copy files and directories in the way we want without worrying about loss of history of corrupting the older versions.

3. ATOMIC COMMITS

While modifying the database, Subversion uses transactions. At the beginning of the commit process, the current state of the database is marked before modifications are carried out. Even if there is a crash, database will not be corrupted and the database can be safely rolled back to the state before the commit. This feature is also not available in CVS, and during a network outage that causes the commit to fail, system will be in a corrupted unstable state.

4. BRANCHES AND TAGS

In many version control systems, the revision trees of individual files and directories can be branched and tagged, but Subversion doesn't support this explicitly. Subversion lacks the concept of branching or tagging. Instead, Subversion provides copies of files or directories. For example, the svn copy command will not make a copy of data contained in those files. It will mark the location of the new file and will link it back to the history of the original file, up to the point where the copy is made. From that point on, if changes are applied to the copy, a new

path of revision is created for the copy, independent of subsequent revisions applied to the original file. By this feature, a branch can be created simply by copying the file or directory that needs to be branched.

In the same way, tags are also created by making a copy, usually in the ‘tags’ directory. This offers flexibility in the way tags are used. One disadvantage is that there is no built-in rule that ensures that tags stay as tags, and don’t inadvertently become branches when someone makes a change to them. But it is possible to enforce tag policies.

5. BINARY FILES

Versioning binary files demands more effort than versioning text files. Text file data can be easily understood and it becomes easy to merge them or examine the differences. For binary files, external program is required to analyze the file and present them in a way that can be understood by humans. It is difficult for version control systems to perform merging or presenting diffs automatically. Diffs will also result in incomprehensible binary data and merging this will result in corrupted files, which are difficult to read by external program.

Subversion still doesn’t have any direct support for automatically merging binary files (which would be nearly impossible anyway, unless SVN could understand the binary files). It does, however, have much better support for resolution of merges that can’t be handled automatically. When a merge conflict occurs, Subversion provides complete copies of both versions of the file, which allows the user to easily use an external editor to manually merge the conflicted file.

6. SYMBOLIC LINKS

Starting from Release 1.1 of Subversion, symbolic links from UNIX systems - like GNU Arch - can be versioned. For example, a user working on a UNIX-like system can add symbolic links to the repository, like he/she would do with any other file, and the repository will retain the link information for other UNIX user who checks out the working copy of the repository. Windows users will not get symbolic links, though.

7. CONFLICT RESOLUTION

With both Subversion and CVS first modifications can be made, and the modifications can be merged with the ones made by others. Most of the other VCSs do this by means of a file-locking mechanism. In CVS, conflict resolution during the merge process can be troublesome. When CVS fails to automatically merge changes between the working copy and the server, it replaces the conflicted file in the working copy with a version of the file containing diffs of the two different versions. If the conflict was large, the resulting diff can waste hours while the developer tries to sort through the changes; and because the local changes are not backed up, there is no way to revert to the pre-conflict state of the working copy without sorting through the diff. Subversion, of course, can’t prevent conflicts from happening any more than CVS can, but it does handle them better when they do happen. If a conflict occurs, SVN replaces the offending file with one containing diffs, just like CVS; however, it also adds temporary versions of the file with the local version, the server version, and the local version prior to any changes. These extra files make resolving conflicts significantly less painful, and once the conflict has been settled, a call to svn resolved removes the extra files.

8. STORAGE

Subversion has a flexible repository backend that allows different types of repository storage systems to be plugged in, transparently to the client. Originally, the only actual repository storage system that was available was the Berkeley DB database system. As of release 1.1 of Subversion, though, a filesystem-based backend (FSFS) is also available as part of the core Subversion system. Instead of storing the entire repository database in a single monolithic database, like the Berkeley DB backend does, the Subversion FSFS storage uses individual files for each revision in the repository. So, when you commit revision 3529, there will be a file named 3529 created, which holds all of the changes for that revision, regardless of how many versioned files were changed in that revision.

9. NETWORK PROTOCOLS

Subversion provides two servers for communicating with the repository via different protocols. The first server (known as svnserve) uses an SVN-specific network protocol that requires a

dedicated server and open port, which allows a Subversion server to be set up quickly and easily. svnserve also supports Inetd access, or tunneled-SSH style access. The other server is a module for the Apache Web server, and is based on the Web-based Distributed Authoring and Versioning (WebDAV) protocol, with a few extensions for version control-specific operations. By using this standard protocol, served over HTTP via Apache, there is no need to open a special port on the server. Because WebDAV support is built into a variety of file managers on different operating systems, it is also possible to get limited access to interact with a repository directly through the Gnome Nautilus file manager on Linux, the Microsoft Windows Explorer, or any other WebDAV client. If readonly access to the repository head is all that is required, you can even access the repository through a Web browser with no special clients or setup required.

10. DATA TRANSFER

Subversion reduces much of the overhead that is associated with communications between the client and server, through a couple of methods not used by many older VCSs, such as CVS. For starters, it only transfers file differences both from client to server and from server to client, whenever possible—unlike CVS, which only transfers differences when going from server to client. Subversion also caches a lot more information locally, which allows it to avoid network communications altogether in many instances. It even stores a full copy of the working directory as of the last update, to allow the user to make comparisons with local changes without contacting the server.

11. HOOK SCRIPTS

Subversion supports a broad array of hook scripts that are run in response to a variety of SVN actions, such as before or after a commit or property change. These scripts are given access to relevant information about the action that is taking place, as well as the capability to examine the repository. Hook scripts can be a powerful tool for automating tasks or enforcing policies. Although they are supported in one form or another by most version control systems, the Subversion support for hook scripts is much more flexible than that found in many others. CVS, for instance, provides commit scripts with little information about the commit being made, such as the target branch for the commit.

12. FULL-FEATURED API

Subversion features a very complete API, which developers can use to easily and elegantly create new client interfaces, to create new Subversion servers, or to integrate Subversion into other development tools. In fact, the standard Subversion client tools, as well as the SVN servers, use these same APIs to communicate with each other, the Subversion repository, and a local working copy. Additionally, the interfaces are available with language bindings for a number of different programming languages (such as C, C++, Java, Perl, and Python), which allows interfacing programs to be written in whatever language best suits the problem at hand (and the developer's expertise).

SVN Architecture

The basic architecture of SVN is given in the figure below. As given in the figure, the Subversion repository keeps the versioned data and Subversion client keeps the local copies of the versioned data. The Repository Access (RA) layer is present in between the repository and the client, which has multiple routes of communication. Some of them go through a network and then through the network servers which then access the repository, and some of them access the repository directly, bypassing the repository.

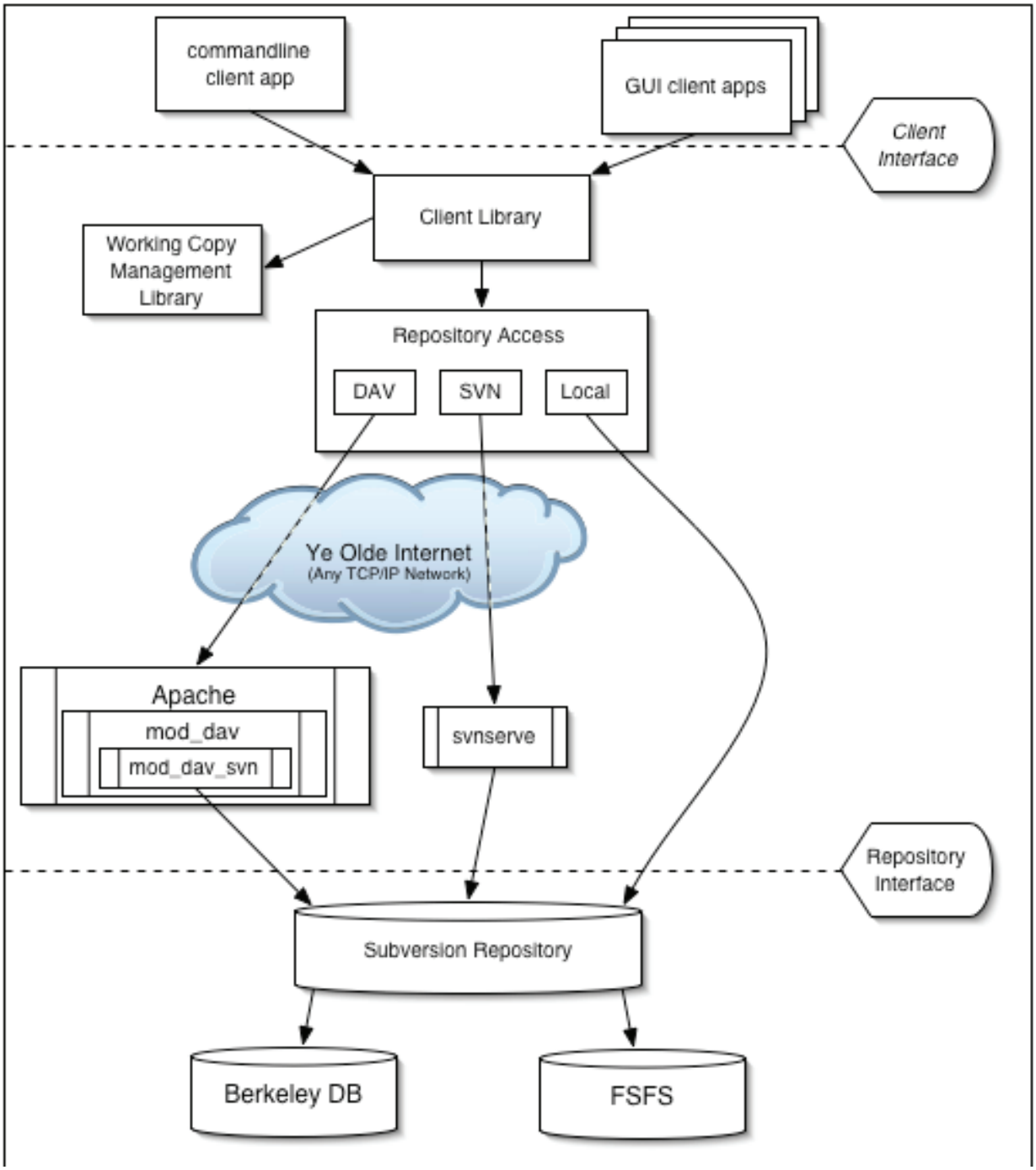


Image Source: Subversion book

Components of SVN

Component	Description
svn	The command-line client program
svnversion	A program for reporting the state (in terms of revisions of the items present) of a working copy
svnlook	A tool for directly inspecting a Subversion repository
svnadmin	A tool for creating, tweaking, or repairing a Subversion repository
mod_dav_svn	A plug-in module for the Apache HTTP Server, used to make your repository available to others over a network
svnserve	A custom standalone server program, runnable as a daemon process or invokable by SSH; another way to make your repository available to others over a network
svndumpfilter	A program for filtering Subversion repository dump streams
svnsync	A program for incrementally mirroring one repository to another over a network
svnrdump	A program for performing repository history dumps and loads over a network

Berkeley DB	Berkeley DB is one of the two types of repository storage systems in Subversion. This was used for the original development of Subversion. Due to its limitations, Berkeley DB is no longer used.