

# SubVersion Cheat Sheet

This Subversion cheat sheet was created during the initial setup of Subversion on Apache 2.0 on Windows and Mac OS X. A detailed tutorial covering most of the features of Subversion can be found in the online Subversion book. However, to make Subversion more useful for me, I created this Readers' Digest version.

## Create a Repository

To store projects in Subversion, first you must create a repository. This must be done to a local drive on a local machine. Creating a repository on a network drive is not supported. To create a repository type:

**UNIX**

```
svnadmin create /path/to/repository
```

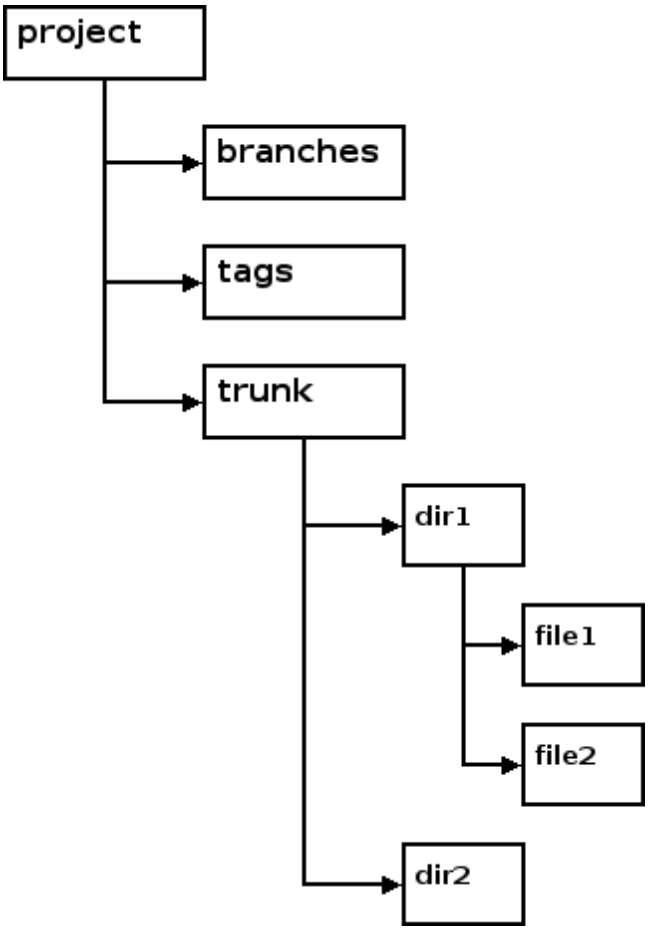
**Windows**

```
svnadmin create d:/path_to_repository
```

By default this sets up a Berkeley database to store the repository. Individual projects should be created as subdirectories of the repository directory (see the next section). Notice that the Windows version includes a drive letter, but also uses forward slashes instead of back slashes. The forward slashes are required even on Windows.

## Add a New Project - `svn import`

To add a project, the Subversion documentation suggests that you create a directory structure like the following:




A root project directory contains three subdirectories, branches, tags, and trunk. Your files and directories are stored under the trunk directory.

Create the directories as described. Assuming the project directory is a subdirectory of the current directory, you would enter the following command

**UNIX**

```
svn import project file:///repository_name/project -m "First Import"
```

**Windows**

```
svn import project file:///d:/repository_name/project -m "First Import"
```

**Network**

```
svn import project http://host_name/svn_dir/repository_name/project -m "First
Import"
```

Notice the Network example includes an svn_dir. This assumes you are using Apache 2.0 and the Subversion modules. When setting up Subversion on Apache, a virtual directory is created on the server that points to your repository directory. More information on Apache 2 setup is described later in this document.

This creates the initial project which you can work from. To get the files under version control, you must checkout a project to begin working on it.

## Checking Out a Project - `svn checkout`

To start using the version control features check out a project into your local working directory. This is done with the following command:

**UNIX**

```
svn checkout file:///repository_name/project/trunk project
```

**Windows**

```
svn checkout file:///d:/repository_name/project/trunk project
```

**Network**

```
svn checkout http://host_name/svn_dir/repository_name/project/trunk project
```

In these examples, project is the name of the directory where you want to store the checked out project on your local file system.

## Getting a List of Projects - `svn list`

To get a list of the current projects stored in a repository, you can use the following command.

**UNIX**

```
svn list --verbose file:///repository_name/project
```

**Network**

```
svn list --verbose http://host_name/svn_dir/repository_name/project
```

This will show you a list of each project directory in that repository.

## Reviewing Changes - `svn status`

To see what files you have changed or added to your checked out work, use the update command:

**UNIX**

```
svn status
```

This command will give you a listing of new files, files that have been changed, and files that have been deleted. New files or deleted files must be added or removed using the add and delete commands (see more below.)

## Adding New Files and Directories - `svn add`

When you add a new file or directory to a project that has been checked out, you must tell Subversion to include that file or directory in its version control.

**UNIX**

```
svn add file_or_dir_name
```

Adding a directory will add the directory and all the files and directories in it. However, this does not add the file or directory to the repository, you must still issue a commit to update the repository.

## Deleting Files and Directories - `svn delete`

If you can add, you can also delete. If you wish to remove a file your directory from be versioned, you use the delete command:

**UNIX**

```
svn delete file_or_dir_name
```

Like add, you must perform a commit before the file is actually deleted from the repository.

However, the delete command does have another option not found in add. With

the delete command you can remove files or directories from the repository. For example, the following command would remove a project and all the files under it.

**Network**

```
svn delete -m "Deleting project dir" http://localhost/svn_dir/repository/project_dir
```

This version of the command comes in particulary useful if someone has accidently imported files into the wrong place (I wouldn't know about that myself of course.)

## Committing Changes - `svn commit`

Once you have added, deleted, or changed files or directories, you can then commit those changes to the repository. This command is pretty straightforward:

**Network**

```
svn commit -m "Saving recent changes" http://localhost/svn_dir/repository
/project_dir
```

## Updating Your Local Files - `svn update`

If you have a set of files checked out and would like to update them to the most recent version of files in the repository, use the update command.

**Network**

```
svn update
```

If there are newer files in the repository, they will overwrite any files you have locally. Before using this command, you may want to use the svn diff command to find out what the differences are between your local files and the repository.

## Tagging Projects or Creating Project Specific Versions

Subversion does not track the version numbers for individual projects automatically. Instead, it tracks each update to the repository and tracks the versions of these updates. To create interim project releases, you must create "Tags" which identify a specify version of a project. This is done by making a

virtual copy of a project in the tags directory. For example:

```
svn copy http://host_name/repos/project/trunk http://host_name/repos/project
/tags/0.1.0 -m "Tagging the 0.1.0 release of the project"
```

This creates a sort of bookmark or snapshot which records the current state of the project. Then, you can checkout the project in this state at any time by simply referring to that release number.

To get a list of the releases for a project.

```
svn list http://192.168.0.4/svn/repos/prj1/tags
0.1.0/
```

Then to check out a release you would type:

```
svn list http://192.168.0.4/svn/repos/prj1/tags/0.1.0

A  0.1.0\dir1
A  0.1.0\dir1\file3
A  0.1.0\dir1\file4
A  0.1.0\file1
A  0.1.0\file2
A  0.1.0\textfile.txt
A  0.1.0\file3
                Checked out revision 13.
```

Since the project has been saved in the tags directory. Release 0.1.0 can be retrieved at any time in the future.

## Basic Apache Setup

You must use Apache 2.0 to install Subversion. Just compile and copy or copy the Subversion Apache module into the Apache modules directory. The following two files must be uncommented or added to the httpd.conf file:

```
LoadModule dav_module         modules/mod_dav.so
LoadModule dav_svn_module     modules/mod_dav_svn.so
```

Next, you must setup a location directive in the httpd.conf file to associate a directory with Subversion repositories. This example uses the SVNParentPath setting to point to a parent directory which contains repository subdirectories. This is convenient as it allows you to add as many repositories as you need without having to restart Apache or modify the httpd.conf file.

```
<Location /svn>
        DAV svn

        # All repos subdirs of d:/svn
        SVNParentPath D:/svn
</Location>
```

**Note:**When using Fink to install Subversion on Mac OS X, the Subversion Apache module is stored in the Fink package listing with the prefix: libapache2. The package full name is libapache2-mod-svn. If you are using Fink, it will automatically install the modules into the correct directory.

## General Notes

Below are a list of notes from initial setup and testing.

- Subversion versions the repository, not individual projects. For example, I have two projects, project 1 and project 2, and check out each project when the current repository version is 3. I make changes to each project and commit those changes back to the repository. For each change, the revision number is incremented in the repository and its current version is now 5. The current revision of each project will also be 5 as they have no separate revision number.
- To setup the Subversion module on Apache for Windows, I had to give the Apache Service access to the local file system. This is done on Windows by setting up a login account for the service. Setup an account in the Users application in Control Panel, make sure to set the password. Once this is done, go to the Services tool in Control Panel. Change the login for the Service to the account you created. XP will automatically give the Login as a Service privilege to the account (the OS must do this as the tools are not available XP Home, only in XP Pro). Once you do this and start and stop the Apache Service, you should be able to read and write to the repository directories. Note: Setting up a log in account for a Service can create a security hole. Consider your security requirements before doing this.
- Individual files and directories that did not exist during the initial import, must be added individually using the svn add command.

Content last updated 05/2012

---