

Subversion

Configuration, Access, Basic
Operations, Troubleshooting,
Additional Reading

What is SVN?

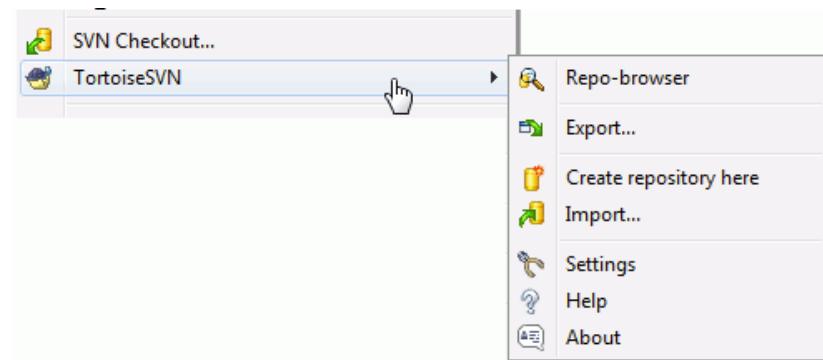
- Subversion(SVN) is a version control system, which allows you to keep old versions of files and directories (usually source code), keep a log of who, when, and why changes occurred, etc.
- Facilitates source code development by multiple software developers across time and location
- Helps resolve common issues faced by multi-developer projects
 - Restoring previous working versions of source code
 - Resolving conflicting versions of the same source code
 - Easily maintain backups of source code, keep copies of every single version, prevents developers from overwriting each other's work

How is SVN deployed?

- SVN typically served via webservers
- SVN Clients/Command line tools available on a variety of platforms
 - Command Line (Unix/Linux/Mac)
 - GUI Driven (Most IDE's, Standalone clients Ex – Windows TortoiseSVN)
- Common Tasks
 1. Checkouts
 2. Commits
 3. Renaming
 4. Deleting
 5. Adding
 6. Updating
 7. Resolving Conflicts
- There can be multiple ways of accomplishing these. It is important to understand what happens conceptually on one platform. All other platforms/Clients are going to behave similarly.
- The following examples are based on: a sample Google Code SVN Server and Tortoise SVN. Instructions are also provided to use: CSE SVN Servers instead (needs VPN Access).

SVN Repository Layout

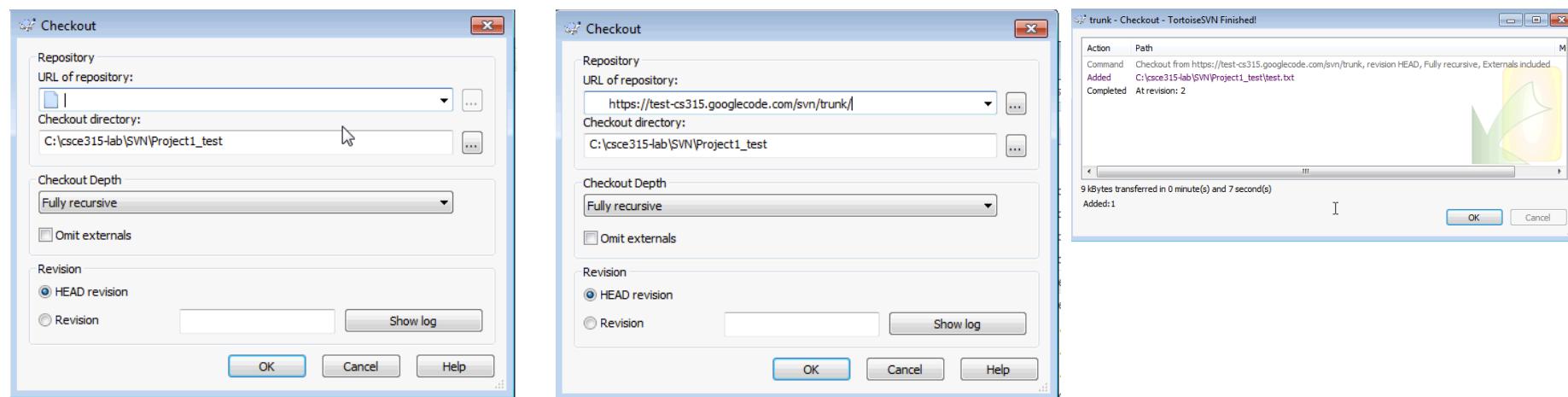
- Standard Layout
 - /trunk – holds the main line of development
 - /branches – contains branch copies
 - /tags – contains tag copies
- These layouts are so common that TortoiseSVN offers to create this for you by default.
- A user is free to ignore these common layouts and create any other variation which works. A switch from one layout to another is a series of server-side moves.
- It is preferred to use separate repositories for unrelated projects. In case, 2 unrelated projects share the same repository, their revision numbers may have large gaps (a revision number is produced for the entire repository every time its contents change)
- In case, the standard layout is not available in the default repository, it is possible to import a desired hierarchy into the SVN.
 - Create a new Empty Folder on your Hard Drive (say C:\SVN\Temp)
 - Create \trunk, \branches, \tags within this empty folder (or any other combination)
 - Right-Click on C:\SVN\Temp and select TortoiseSVN → Import → Enter repository name
 - This will create a default folder structure in the repository



SVN Checkout

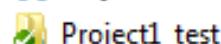
- Checkout

- Client connects to the SVN Server, downloads a version of the code from a repository
- Once downloaded, it can be worked with like un-versioned code
- Create a Local Folder (say C:\SVN)
- Make a Project Folder (say C:\SVN\Project1_test)
- Right Click on Folder and select SVN Checkout. Enter Repository Name. C:\SVN\Project1_test is ready

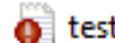


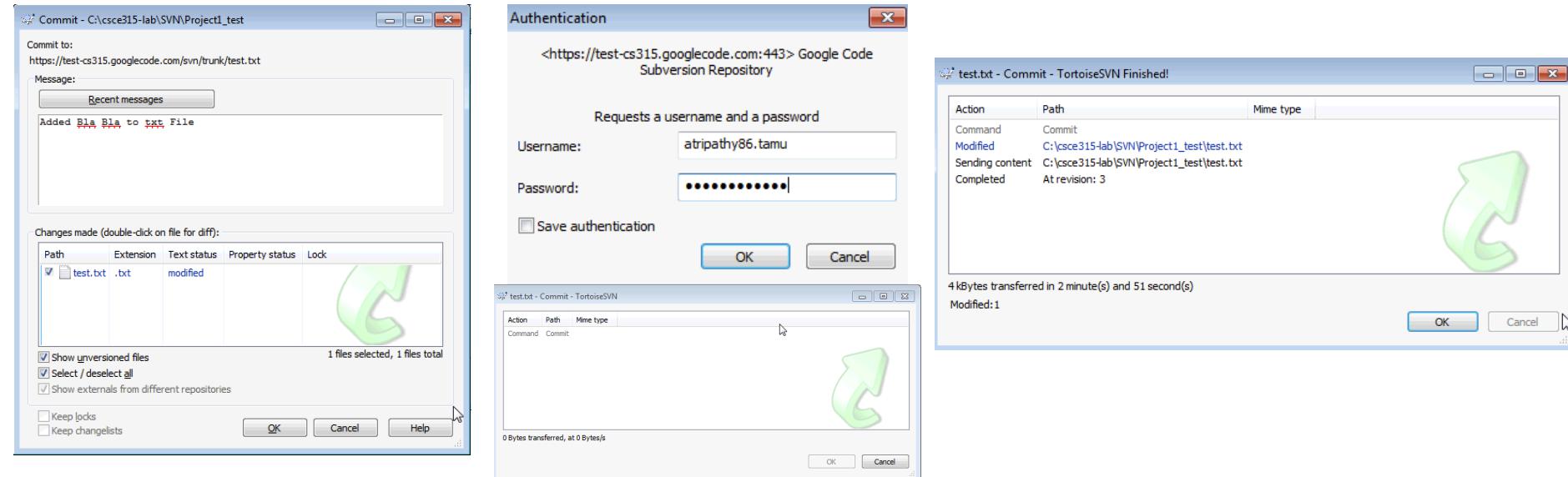
- Checkout – Additional Notes

- It is possible to just check out a sub folder of the repository or even a single file. Modify the URL of the repository as desired
- The Folder icon should change. The green check means that the folder contains SVN files and is in sync.



SVN Commit

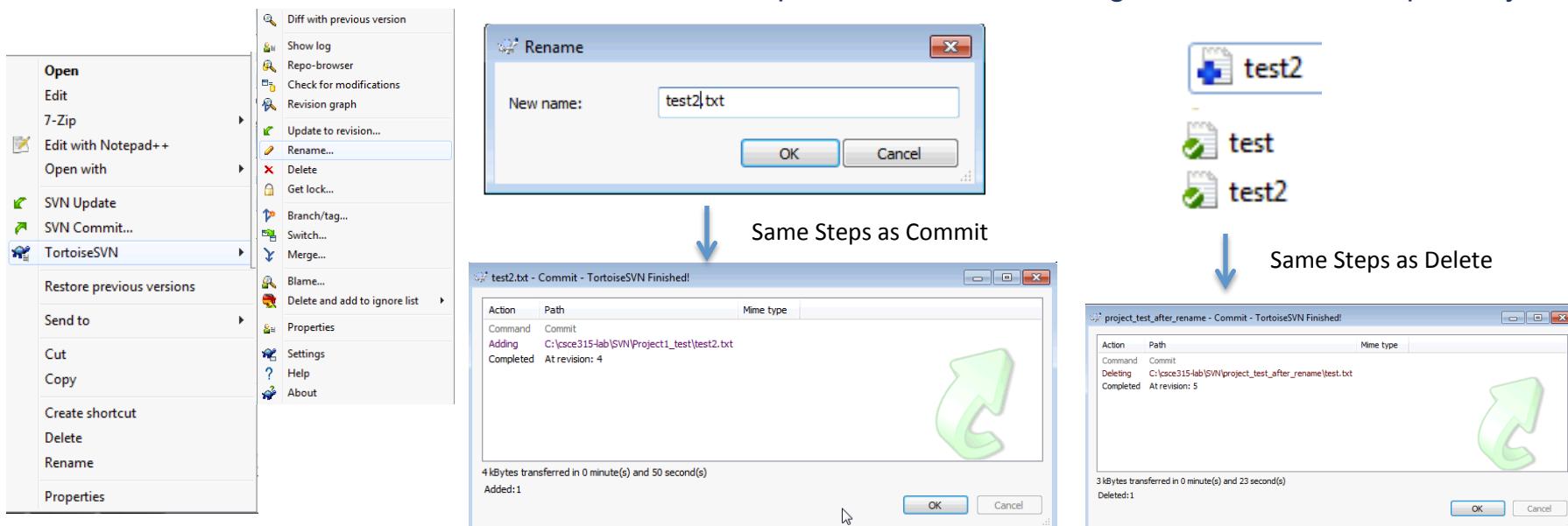
- Commit
 - Once a significant milestone is achieved, the updated “working” code is committed back to the SVN repository
 - Any subsequent check-outs will acquire this latest version of code
 - Let’s assume we have made changes to one existing file. Its icon changes (shows un-synced) 
 - Right Click on File (or Folder) and select SVN Commit. Update Log with comment that is concise, but accurately describes the important changes made to repository.



- Commit – Additional Notes
 - There will be a need to interact with the commit interface further when deleting, renaming, adding files or folders.
 - Now, you are free to delete your local folder (C:\SVN\Project1_test). Recreate it and checkout again. You should have the previous version since it was successfully synced with the repository.

SVN Rename

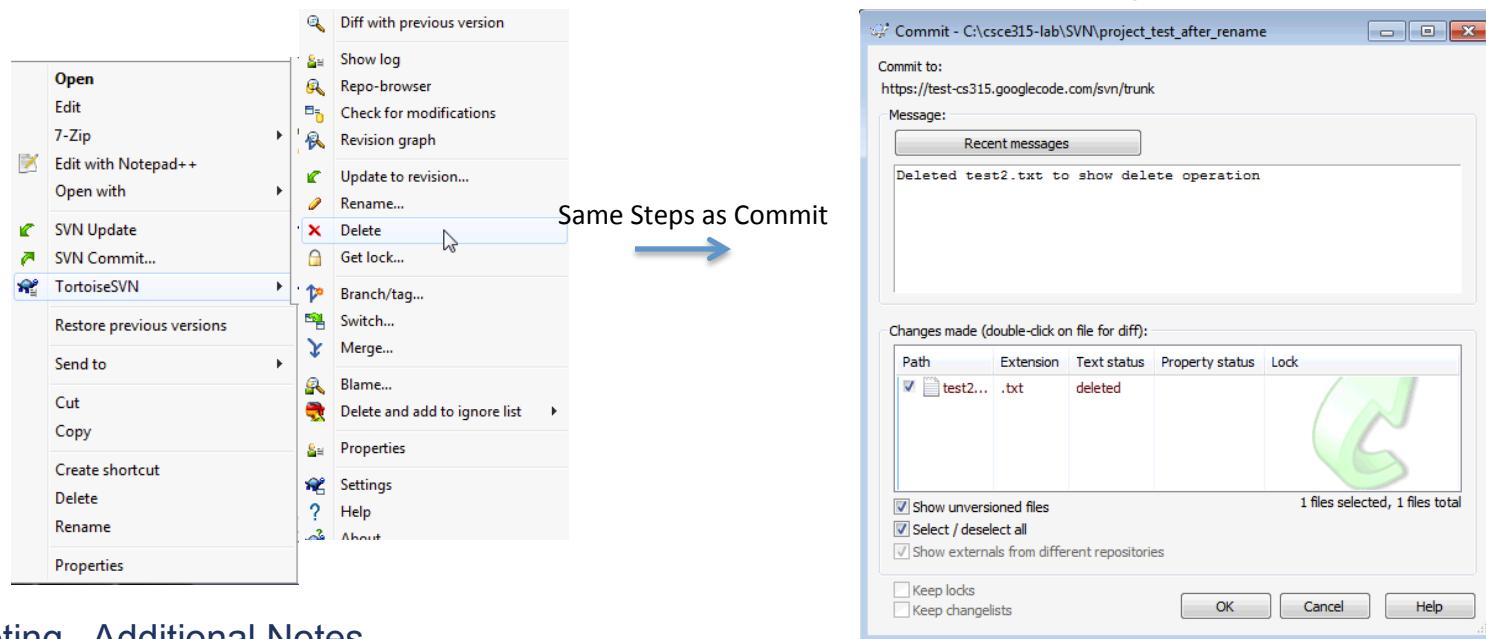
- Renaming
 - Make sure you have the latest checked out version on local machine. (Use SVN Update if necessary)
 - Right Click on File/Folder → select SVN Rename → Enter New Name
 - It's Not Done! An SVN Commit needs to be performed for the change to reflect in the repository



- Renaming – Additional Notes
 - SVN behaves strangely during a rename operation. The old file is still present in the repository. It needs to be explicitly deleted.

SVN Delete

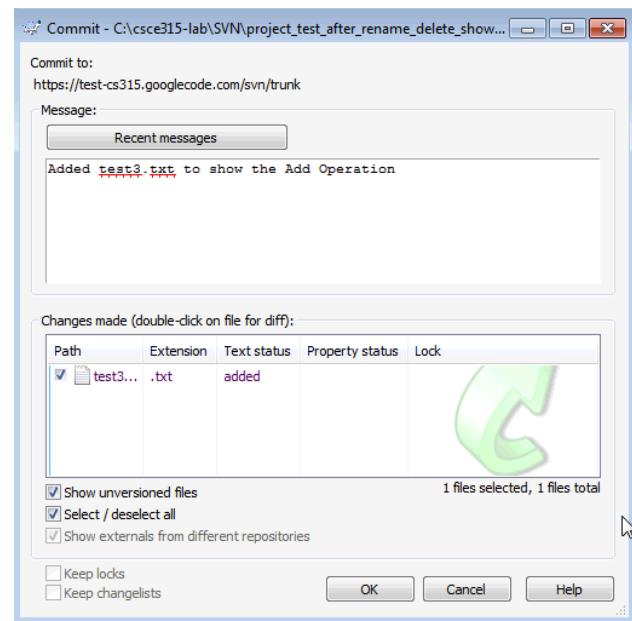
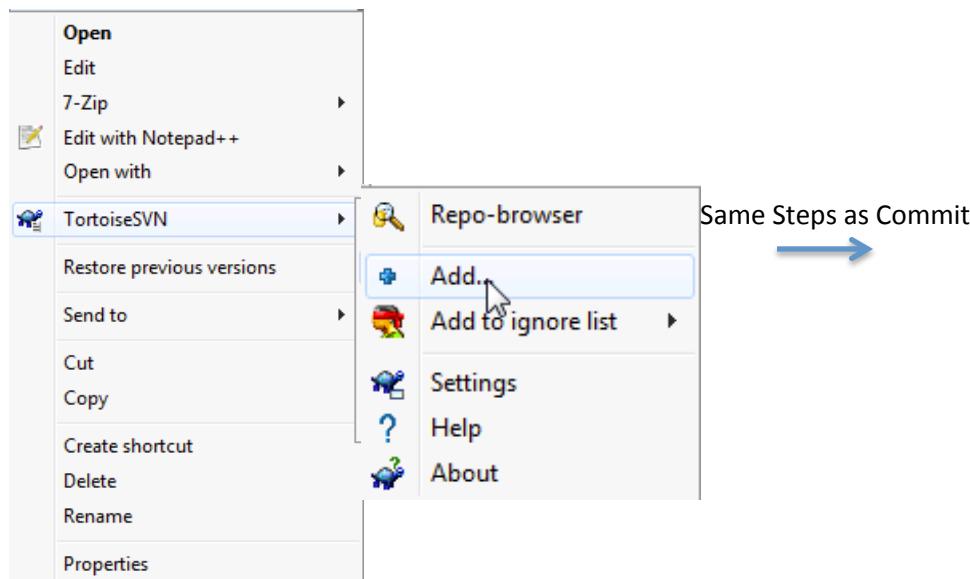
- **Deleting**
 - Make sure you have the latest checked out version on local machine. (Use SVN Update if necessary)
 - Right Click on File/Folder → TortoiseSVN-> Delete
 - It's Not Done! An SVN Commit needs to be performed for the change to reflect in the repository



- **Deleting— Additional Notes**
 - Make sure you Commit after a delete operation. It is easy to forget to do it since the file “apparently” disappears from the local folder after the delete operation.

SVN Add

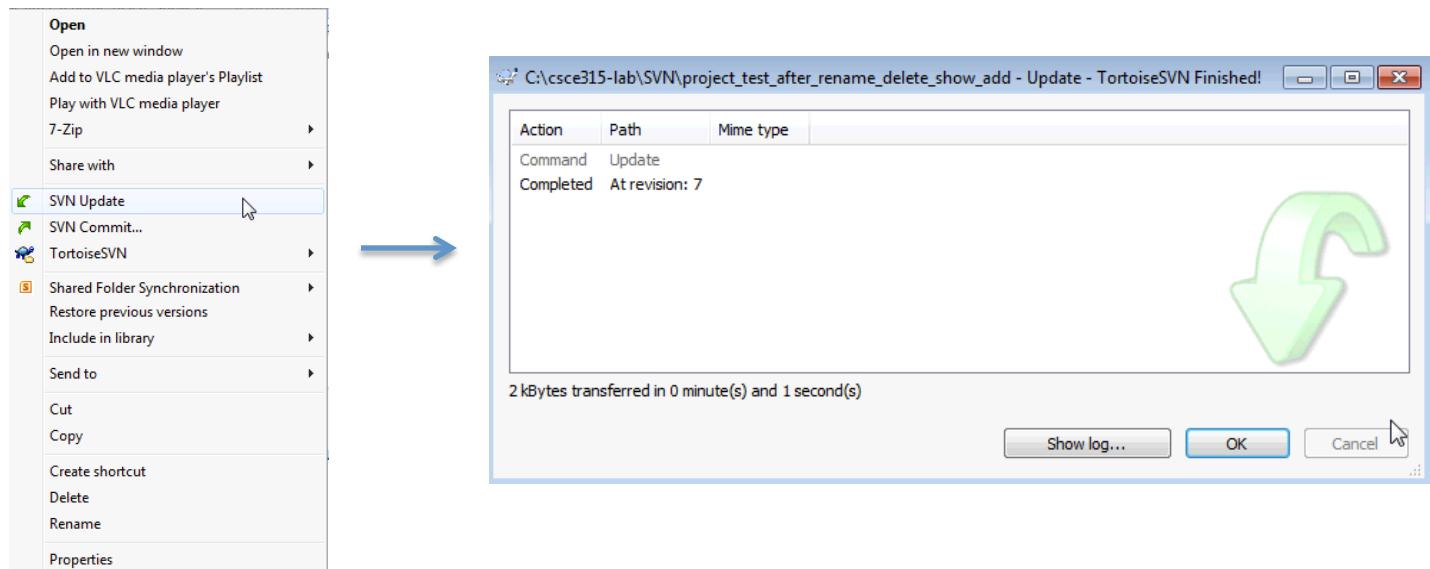
- Adding
 - Make sure you have the latest checked out version on local machine. (Use SVN Update if necessary)
 - Create the new File or folder. Right Click on File/Folder → TortoiseSVN-> Add
 - It's Not Done! An SVN Commit needs to be performed for the change to reflect in the repository



- Deleting—Additional Notes
 - Make sure you Commit after a delete operation. It is easy to forget to do it since the file “apparently” disappears from the local folder after the delete operation.

SVN Update

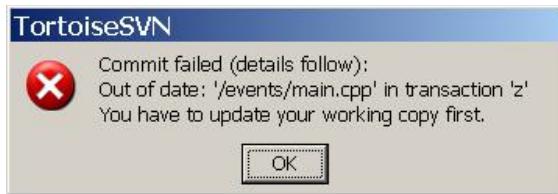
- **Update**
 - This is used to update the local snapshot of the repository to the latest available version.
 - Right Click → SVN Update



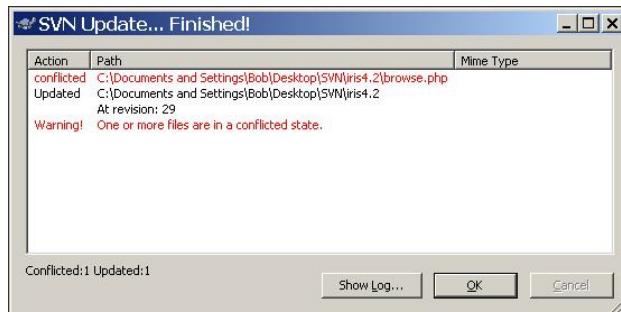
- **Updating – Additional Notes**
 - Update is possible only after a previous Checkout.
 - Files which were checked out are updated (or deleted). If new files have been added to the repository (or deleted/rename), they will not be reflected. That is, only the files already on the hard drive will be touched.
 - **WARNING** – Any local changes made to files will be wiped away. Make sure you commit before an update otherwise you may lose several hours of work. It is possible that entire folders may be deleted during an update (if the latest repository status is so)

SVN Resolve Conflicts

- Conflict Management
 - Suppose User1 is working at Revision 10 on main.cpp
 - In the meantime, User2 also working on main.cpp commits it. Latest Revision 11.
 - When User1 tries to commit, an out-of-date error will occur.



- Correct Approach is to immediately run SVN Update

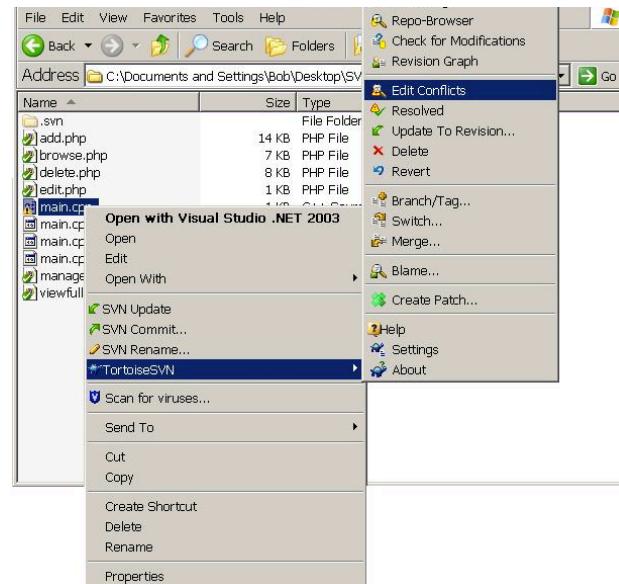


- This will create 4 new files

SVN Resolve Conflicts

- Conflict Management

- For every conflicting file, 4 new files are created
 - main.cpp.mine – What main.cpp v10 looked like after user1 changed it (no conflict markers)
 - main.cpp.r10 – What main.cpp v10 looked like (the file user1 checked out and modified)
 - main.cpp.r11 – What main.cpp v11 looks like (the file user2 committed. The latest version)
 - Main.cpp – A file with conflict markers inserted.
- Right Click on Main.cpp → TortoiseSVN → Edit Conflicts.
 - This opens a Merge Program – lets user1 look at the changes. User1 has to decide which changes to retain in the final file. Once this is done, user1 can mark the file as merged (from within the program).
 - It's Not Done! An SVN Commit needs to be performed for the change to reflect in the repository



SVN Further Reading

- Closing Notes
 - There are other options such as Branch/Tag, Switch, Check for Modifications, Revision Graph
 - Please make sure that you include a log file from your repository in all your project submissions. This should include the Revision Numbers, Usernames, Dates, Comments made.
- Additional Reading
 - [Version Control with Subversion](#)
 - [Command Line SVN \(for Unix\)](#)
 - [Source Control in 10 minutes](#)
 - [TortoiseSVN User's Manual \(v1.7\)](#)

CSE SVN

Team Name	SVN URL	Usernames
Team1	https://svn.cse.tamu.edu/ CSCE_315_100_PROJECT1_TEAM1/	gbrown, zacharh7, jratway
Team2	https://svn.cse.tamu.edu/ CSCE_315_100_PROJECT1_TEAM2/	chf9302, goodey, ckvooss
Team3	https://svn.cse.tamu.edu/ CSCE_315_100_PROJECT1_TEAM3/	steyck, dcm8174, ses8804
Team4	https://svn.cse.tamu.edu/ CSCE_315_100_PROJECT1_TEAM4/	dlasater, mcelroy, schwartz
Team5	https://svn.cse.tamu.edu/ CSCE_315_100_PROJECT1_TEAM5/	jchavez2, abhayo, lcpow1
Team6	https://svn.cse.tamu.edu/ CSCE_315_100_PROJECT1_TEAM6/	ells118, djfireman, ewok24
Team7	https://svn.cse.tamu.edu/ CSCE_315_100_PROJECT1_TEAM7/	ab1tar, jpineda, dreyna, zjw7965
Team8	https://svn.cse.tamu.edu/ CSCE_315_100_PROJECT1_TEAM8/	shuvo7, bishoyk, hhn0997, hjt13