Software Engineering Tools Lab

Assignment No-6

Module 4 & 5- Understanding version control using VSS and Managing code using SVN

PRN: 2020BTECS00069

Name: Yashraj Patil

---

1) What is Microsoft's VSS? Provide the information of VSS tool with respect to below points.

Microsoft's Volume Shadow Copy Service (VSS) is a feature in Microsoft Windows that enables users to create snapshots of computer files or volumes, even when they are in use. The snapshots are known as shadow copies, and they can be used to restore files to an earlier state or to recover data that has been lost due to accidental deletion, hardware failure, or other reasons. VSS operates by creating a snapshot of the data on a volume, which can be used to restore the data to a previous state.

a. Owner/developer

Microsoft's Volume Shadow Copy Service (VSS) is a proprietary technology developed by Microsoft Corporation.

b. Brief information/introduction

Volume Shadow Copy Service (VSS) is a feature in Microsoft Windows that allows users to create snapshots of computer files or volumes, even when they are in use. The snapshots are known as shadow copies, and they can be used to restore files to an earlier state or to recover data that has been lost due to accidental deletion, hardware failure, or other reasons.

c. Basic operations involved

VSS operates by creating a snapshot of the data on a volume, which can be used to restore the data to a previous state. The VSS API is used by backup software to request the creation of a snapshot, and VSS coordinates with the file system and other applications to ensure that the snapshot is consistent and reflects the state of the volume at a specific point in time.

d. Advantages

VSS provides several advantages, such as:

Ability to create backups of files that are in use, without having to shut down applications or services.

Ability to restore files to a previous state or recover data that has been lost due to accidental deletion, hardware failure, or other reasons.

Ability to perform backups and restores quickly and efficiently, without consuming excessive system resources.

Compatibility with a wide range of backup software and hardware, allowing users to choose the solution that best fits their needs.

e. Disadvantages

Some potential disadvantages of VSS include:

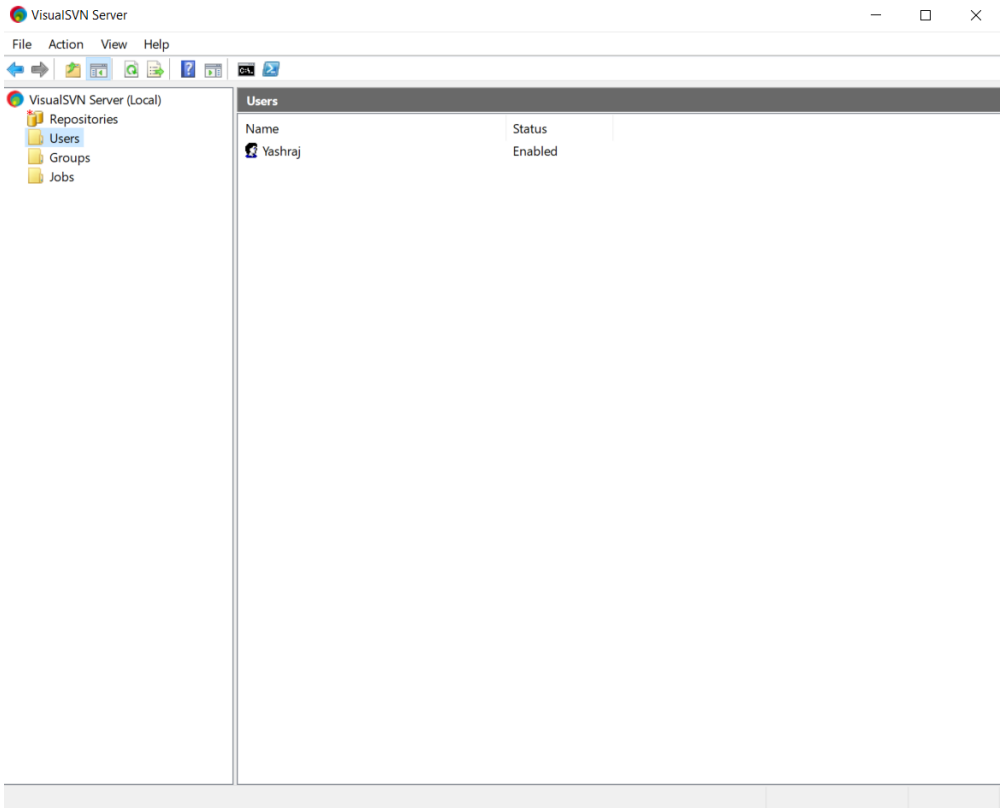
The amount of disk space required to store shadow copies can be significant, particularly if multiple snapshots are taken.

The creation of a snapshot can cause a temporary performance impact, particularly if the system is already under heavy load.

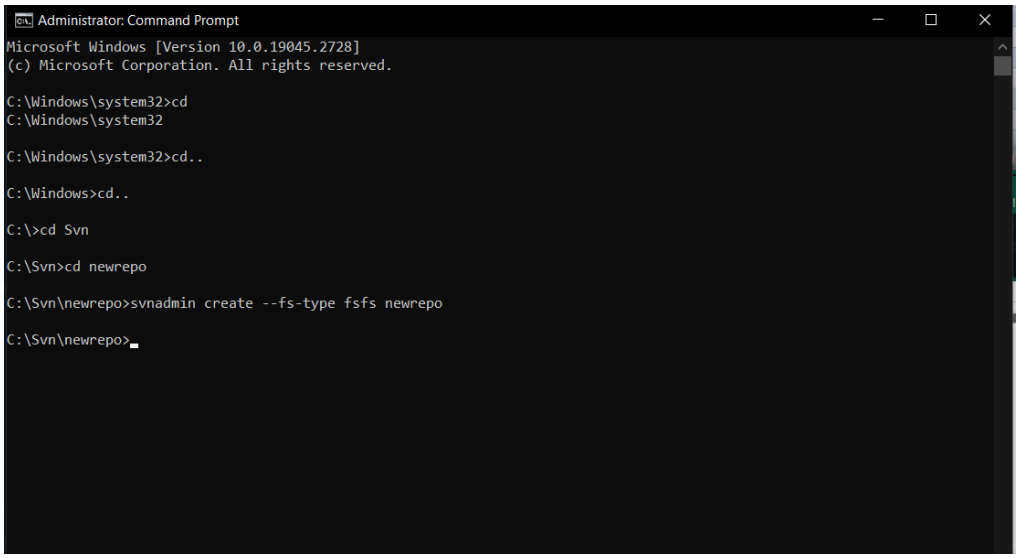
VSS may not be available on all versions of Windows or in all environments, limiting its usefulness in certain situations.

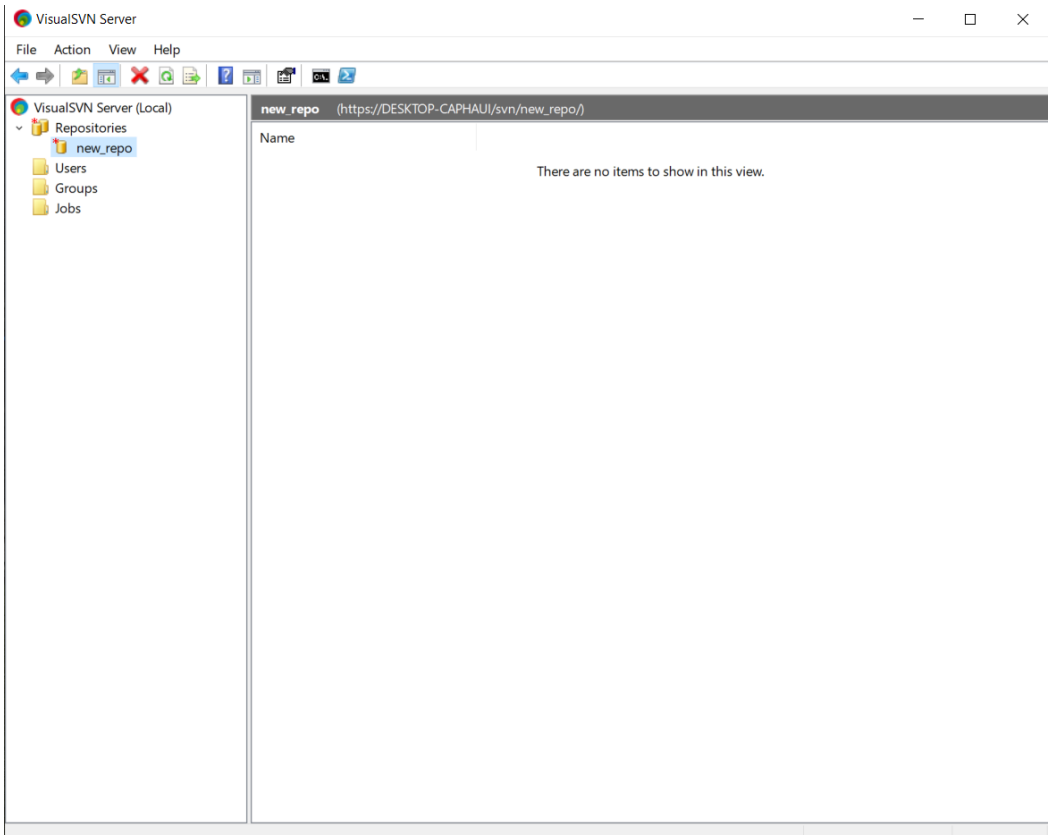
2) Create a SVN repository and perform below operations on that repository using SVN. Also explain below operations.

Create an user in SVN server:



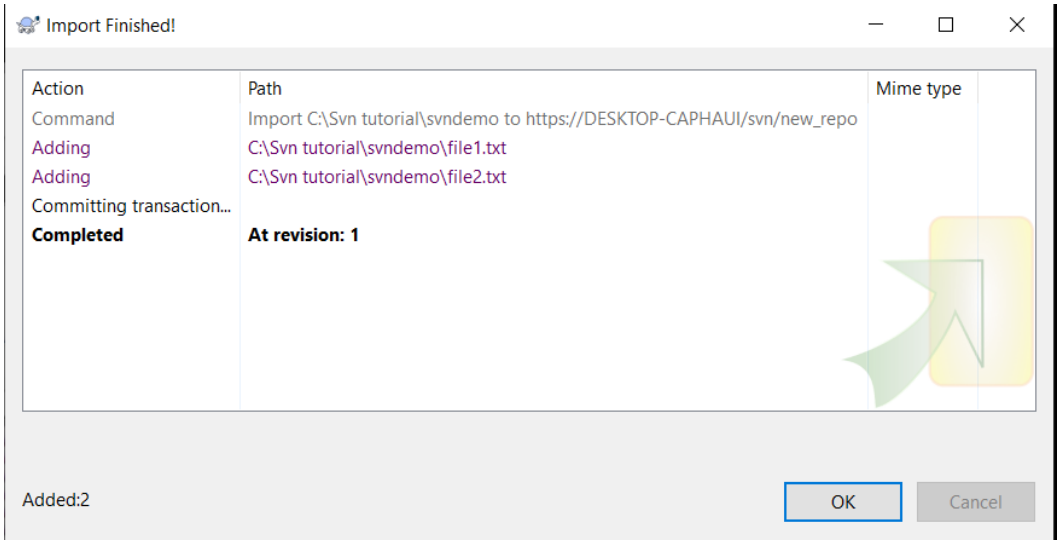
Create a repo in the SVN server:





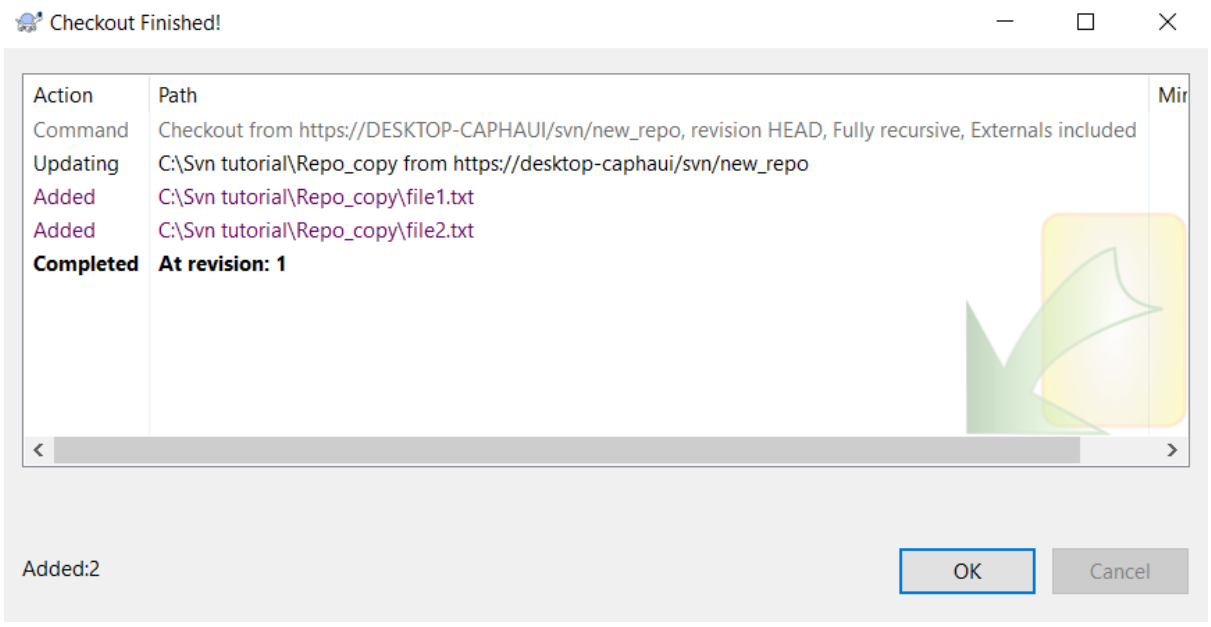
1. Import

The files are imported from the SVN tutorial folder to the new_repo in the server. It shows the revision as 1 because it is the first import.



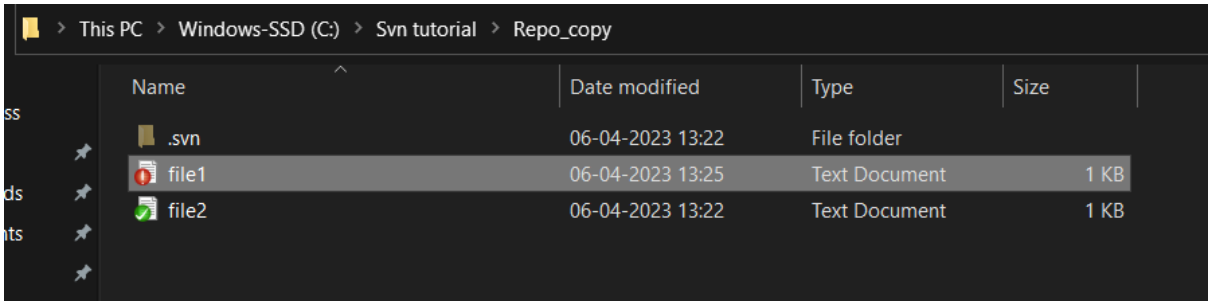
2. Checkout:

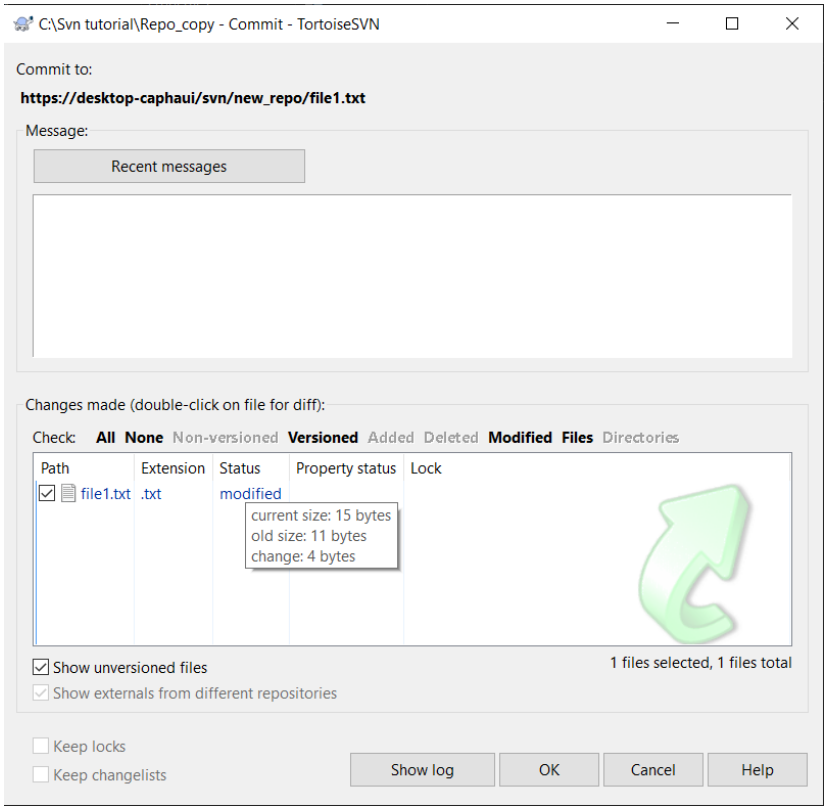
To checkout a copy of the repository to your local machine.

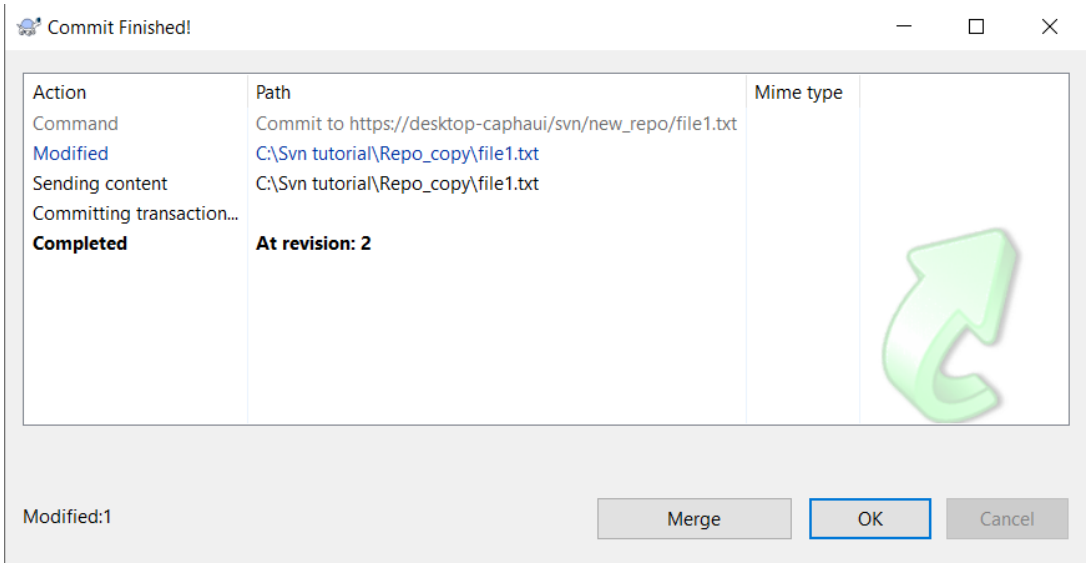


3. Commit:

After modifying the file 1:

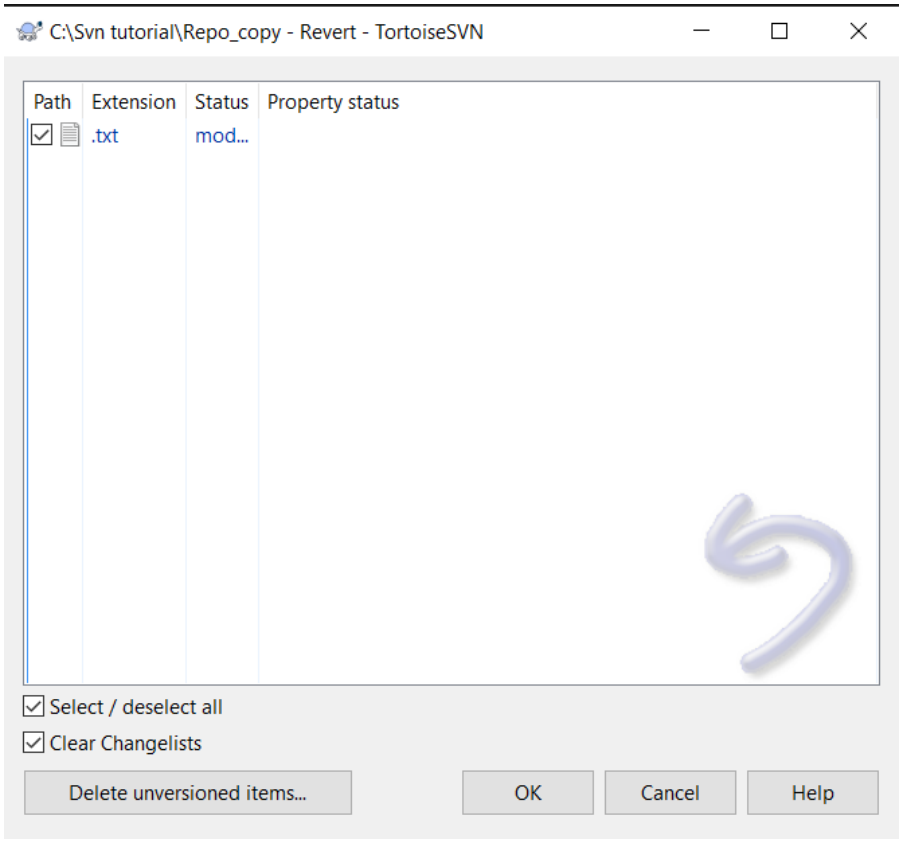

This PC
Windows-SSD (C:)
Svn tutorial
Repo_copy
Name
Date modified
Type
Size
.svn
06-04-2023 13:22
File folder
file1
06-04-2023 13:25
Text Document
1 KB
file2
06-04-2023 13:22
Text Document
1 KB

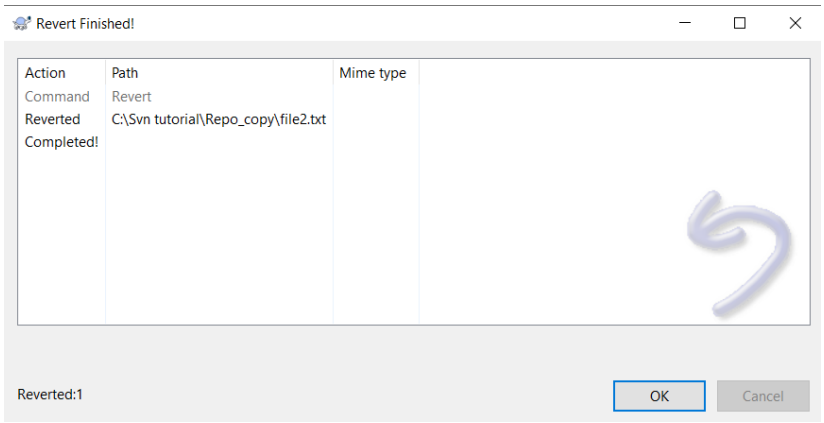

C:\Svn tutorial\Repo_copy - Commit - TortoiseSVN
Commit to:
https://desktop-caphaui/svn/new_repo/file1.txt
Message:
Recent messages
Changes made (double-click on file for diff):
Check: All None Non-versioned Versioned Added Deleted Modified Files Directories
Path
Extension
Status
Property status
Lock
file1.txt
.txt
modified
current size: 15 bytes
old size: 11 bytes
change: 4 bytes
Show unversioned files
Show externals from different repositories
1 files selected, 1 files total
Keep locks
Keep changelists
Show log
OK
Cancel
Help


Commit Finished!
Action
Path
Mime type
Command
Commit to https://desktop-caphaui/svn/new_repo/file1.txt
Modified
C:\Svn tutorial\Repo_copy\file1.txt
Sending content
C:\Svn tutorial\Repo_copy\file1.txt
Committing transaction...
Completed
At revision: 2
Modified:1
Merge
OK
Cancel

4. Revert

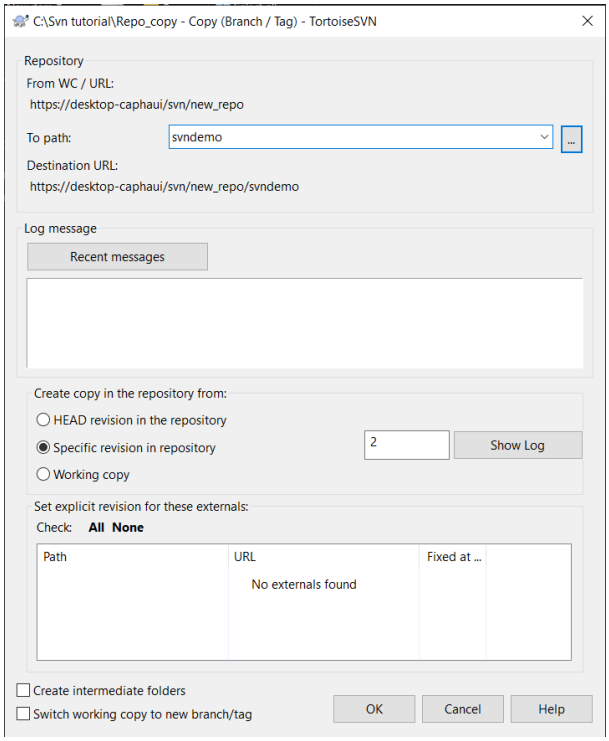
1. Navigate to the file or folder that you want to revert in your SVN client.
2. Right-click on the file or folder and select "TortoiseSVN" from the context menu.
3. From the TortoiseSVN submenu, select "Revert".
4. In the "Revert" dialog box, select the items you want to revert. You can choose to revert only the selected item, the item and its children, or the item and all its siblings.
5. Click "OK" to revert the selected items to their previous state.

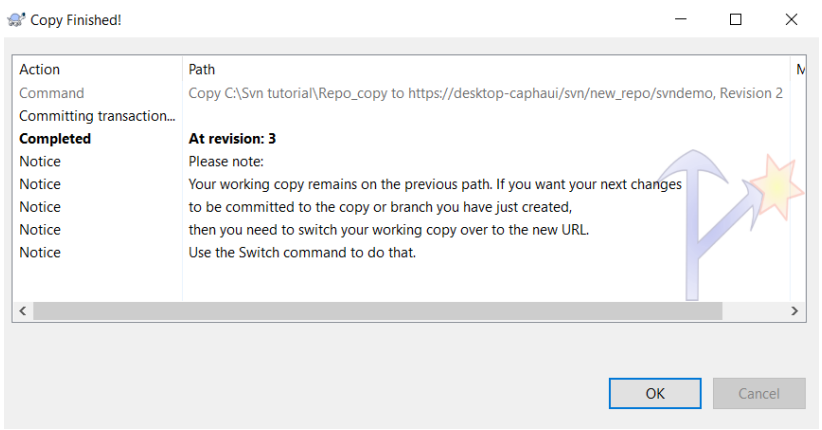




5. Copy

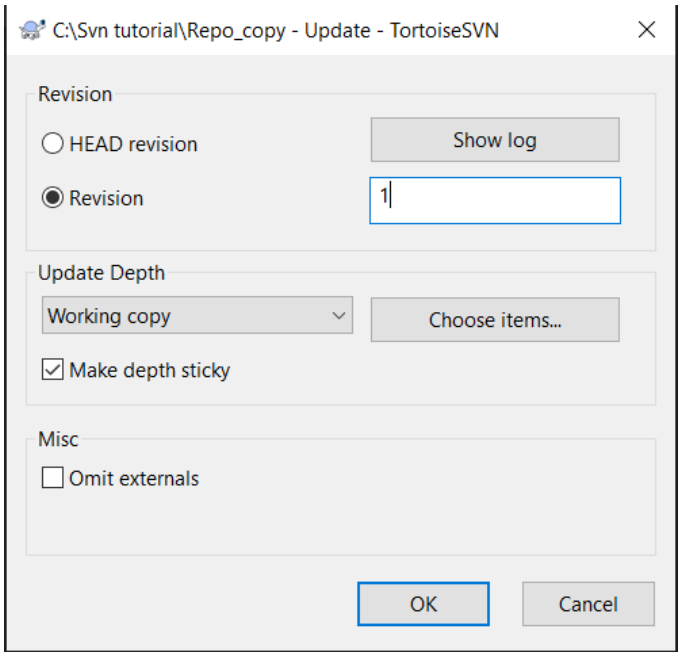
1. Navigate to the file or folder that you want to copy in your SVN client.
2. Right-click on the file or folder and select "TortoiseSVN" from the context menu.
3. From the TortoiseSVN submenu, select "Branch/Tag".
4. In the "Branch/Tag" dialog box, select the source repository URL and the destination URL for the copy. You can choose to create a new branch/tag or select an existing one.
5. Set any additional options you want to use, such as setting a revision number or specifying the copy depth.
6. Click "OK" to create the copy.

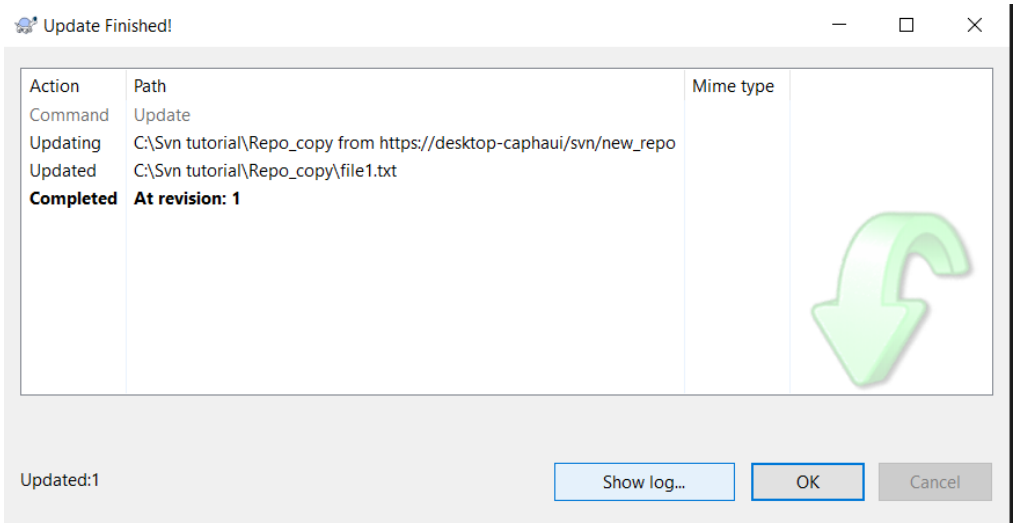




6. Update

1. Navigate to the file or folder that you want to update in your SVN client.
2. Right-click on the file or folder and select "TortoiseSVN" from the context menu.
3. From the TortoiseSVN submenu, select "Update".
4. In the "Update" dialog box, review the list of files that will be updated and select any additional options you want to use, such as updating to a specific revision number or limiting the update depth.
5. Click "OK" to start the update process.





Create a new repository using the svnadmin command:

```
PS F:\z.DOdo\WCE Assignment\Sem 6\SAT\6\SVN_repo> svnadmin create "F:\z.DOdo\WCE Assignment\Se
m 6\SAT\6\SVN_repo"
```

This will create a new SVN repository at the specified path.

a. Revert

```
PS F:\z.DOdo\WCE Assignment\Sem 6\SAT\6\SVN_repo> svn revert "F:\z.DOdo\WCE Assignment\Sem 6\S
AT\6\ckout"
PS F:\z.DOdo\WCE Assignment\Sem 6\SAT\6\SVN_repo>
```

b. Import

```
PS F:\z.DOdo\WCE Assignment\Sem 6\SAT\6\SVN_repo> svn import "F:\z.DOdo\WCE Assignment\Sem 6\S
AT\6\test" file:///F:/z.DOdo/WCE%20Assignment/Sem%206/SAT/6/SVN_repo -m "Initial import"
Adding         F:\z.DOdo\WCE Assignment\Sem 6\SAT\6\test\test.txt
Committing transaction...
Committed revision 1.
```

c. Checkout

```
PS F:\z.DOdo\WCE Assignment\Sem 6\SAT\6\SVN_repo> svn checkout file:///F:/z.DOdo/WCE%20Assignm
ent/Sem%206/SAT/6/SVN_repo "F:\z.DOdo\WCE Assignment\Sem 6\SAT\6\ckout"
A    F:\z.DOdo\WCE Assignment\Sem 6\SAT\6\ckout\test.txt
Checked out revision 1.
```

d. Commit

```
PS F:\z.DOdo\WCE Assignment\Sem 6\SAT\6\SVN_repo> svn commit "F:\z.DOdo\WCE Assignment\Sem 6\S
AT\6\ckout" -m "Commit in ckout "
Sending        ckout\test.txt
Transmitting file data .done
Committing transaction...
Committed revision 2.
```

e. Update

```
PS F:\z.DOdo\WCE Assignment\Sem 6\SAT\6\SVN_repo> svn update "F:\z.DOdo\WCE Assignment\Sem 6\S
AT\6\ckout"
Updating 'F:\z.DOdo\WCE Assignment\Sem 6\SAT\6\ckout':
At revision 2.
```

3) Q 3. Perform below operations using CVS

a. CVS checkout

```
PS F:\z.DOdo\WCE Assignment\Sem 6\SAT\5> cvs -d "F:\z.DOdo\WCE Assignment\Sem 6\SAT\5\CVS" checkout -
d workspace workspace
cvs checkout: Updating workspace
```

b. CVS update

```
PS F:\z.DOdo\WCE Assignment\Sem 6\SAT\5> cd workspace
PS F:\z.DOdo\WCE Assignment\Sem 6\SAT\5\workspace> cvs add test.txt
cvs add: nothing known about test.txt
PS F:\z.DOdo\WCE Assignment\Sem 6\SAT\5\workspace> cvs add test.txt
cvs add: scheduling file `test.txt' for addition
cvs add: use 'cvs commit' to add this file permanently
```

c. CVS adds

```
PS F:\z.DOdo\WCE Assignment\Sem 6\SAT\5> cd workspace
PS F:\z.DOdo\WCE Assignment\Sem 6\SAT\5\workspace> cvs add test.txt
cvs add: nothing known about test.txt
PS F:\z.DOdo\WCE Assignment\Sem 6\SAT\5\workspace> cvs add test.txt
cvs add: scheduling file `test.txt' for addition
cvs add: use 'cvs commit' to add this file permanently
```

d. CVS removes

```
PS F:\z.DOdo\WCE Assignment\Sem 6\SAT\5\workspace> cvs remove .\test.txt
cvs remove: file `test.txt' still in working directory
cvs remove: 1 file exists; remove it first
PS F:\z.DOdo\WCE Assignment\Sem 6\SAT\5\workspace> cvs commit -m "Removed tests file "
cvs commit: Examining .
```

e. CVS commit

```
PS F:\z.DOdo\WCE Assignment\Sem 6\SAT\5\workspace> cvs commit -m "Added new file"
cvs commit: Examining .
RCS file: F:\z.DOdo\WCE Assignment\Sem 6\SAT\5\CVS/workspace/test.txt,v
done
Checking in test.txt;
F:\z.DOdo\WCE Assignment\Sem 6\SAT\5\CVS/workspace/test.txt,v  <--  test.txt
initial revision: 1.1
done
```

4) Differentiate Between The Git & SVN Repository?

Distributed vs. Centralized: Git is a distributed version control system, which means that every developer has their own local copy of the repository, and changes can be made offline. SVN, on the other hand, is a centralized version control system, where all developers access a single repository on a central server.

Branching and Merging: Git is known for its powerful branching and merging capabilities, which allow developers to create and merge branches quickly and easily. SVN also supports branching and merging, but it can be more complicated and time-consuming.

Speed: Git is generally considered faster than SVN, especially when it comes to operations like branching, merging, and committing changes.

Learning Curve: Git has a steeper learning curve than SVN, primarily due to its distributed nature and more advanced features. However, once users become familiar with Git, they often find it to be more efficient and flexible than SVN.

Integration: Git is widely used and has a large community of developers, which means that it has strong integration with other tools and services. SVN, on the other hand, is less widely used and may have limited integration options in some cases.

5) What is “branch”, “tag” And “trunk” In SVN?

In SVN, a branch is a copy of the repository that allows developers to work on a separate set of changes without affecting the main codebase. Branches are typically used to develop new features or fix bugs, and they can be merged back into the main codebase when the changes are complete.

A tag in SVN is a way to mark a specific version of the codebase, usually to indicate a release or milestone. Tags are read-only and cannot be modified, making them useful for creating snapshots of the codebase at specific points in time.

The trunk in SVN refers to the main line of development in the repository. This is where the latest version of the codebase is stored and where developers typically make changes before branching off to work on separate features or bug fixes.

6) How CVS is different from SVN?

Distributed vs. Centralized: SVN is a centralized version control system, where all developers access a single repository on a central server. CVS, on the other hand, is an older version control system that relies on a client-server architecture, but it is not a distributed system.

Atomic Commits: SVN supports atomic commits, which means that changes to multiple files can be committed together as a single transaction. CVS does not support atomic commits, so changes to multiple files must be committed separately, which can lead to problems if one of the changes fails.

Branching and Merging: SVN has more advanced branching and merging capabilities than CVS, making it easier to create and manage branches and to merge changes between branches.

Renaming and Moving: SVN supports renaming and moving files and directories, while CVS does not. This can be important when refactoring code or reorganizing the project structure.

File Properties: SVN supports file properties, which allow developers to attach metadata to files, such as author information, MIME type, and executable permissions. CVS does not support file properties.