SVN Guide

This paper will help you to properly use the Subversion (SVN) system and the associated well-known client TortoiseSVN, which should already be installed on your computer. It also provides some checklists that you should ALWAYS follow before doing any action such as committing, merging, creating branch, etc. This will avoid some problems we’ve already had in the past ^-^

Content

[SVN Commit 2](#_Toc419293074)

[SVN Update 2](#_Toc419293075)

[SVN Checkout 3](#_Toc419293076)

[SVN Branches 3](#_Toc419293077)

[SVN Switch 4](#_Toc419293078)

[SVN Merge 5](#_Toc419293079)

[Deleting, moving, renaming and adding files 7](#_Toc419293080)

[About Linux 7](#_Toc419293081)

# SVN Commit

## What does “commit” mean?

What is commonly called a “commit” is the action of sending your local work copy to the SVN server, which is installed in the lab (probably). By committing your files, you “save” your work on the distant server and tell to everyone (especially the Project Manager) that you did some new work. SVN is smart and will only send the changed data in order to save bandwidth, so a commit is generally pretty fast. It is also a safety against local losses which could be caused by a problem on your computer or your local drive.

## When should I commit my work?

Remember this simple rule: for the Project Manager, a person who does not commit is a person who does not work. So commit as often as possible. However, in order not to overload the SVN history with too many commits, avoid committing when you did not do some relevant modifications to your local work copy. But you should always commit when you change your active computer (for example between home and the lab) in order to always work on the same version of your work (see the [Update](#_SVN_Update) section).

## Checklist for committing:

1. Check the log of your branch (Right click -> TortoiseSVN -> Show log). In the log, the version on which you are working is in bold type. The first line should therefore always be in bold type.
2. If this is not the case, do an [Update](#_SVN_Update) on your working copy.
3. Open the SVN commit window (Right click -> SVN Commit…)
4. Enter a message to describe what changes you brought to your work. ALWAYS write a message.
5. In the lower window, check all the files you want to commit and uncheck the ones you don’t want to commit (such as temporary files for example).
6. Now you can commit. Tortoise will display what it sends to the server in a new window. Close this window and you’re done.

# SVN Update

## What does “update” mean?

Updating means that you will synchronize your local working copy with the distant one (the version on the SVN Server). SVN will merge the modifications you brought locally with the distant copy. Actually, if you work with SVN properly, it should not have to merge anything, and will just retrieve the distant copy to your computer.

## When should I update my working copy?

Every time you (or someone else, but that should not happen in your branch) committed modifications from another computer. That way, you will always work on the last available version of your work.

## Checklist for updating:

1. Check the log of your branch (Right click -> TortoiseSVN -> Show log). In the log, the version on which you are working is in bold type. Therefore, if the first line is in bold type, you don’t need to update.
2. You can now update (Right click -> SVN Update). Tortoise will open a new window, showing you what changes are brought to your working copy.
3. Check and possibly solve all the merging problems (see [SVN Merge](#_Merge_errors)), and close the window.

# SVN Checkout

## What is a “checkout”?

A checkout simply consists in copying a distant repository to your computer, and thus creating a new local working copy of this repository. In opposition to the [Update](#_SVN_Update), it does not merge anything and simply retrieve everything from the repository. Actually, Tortoise will even turn down the checkout request if you try to do it inside an existing working copy.

## When should I do a checkout?

Every time you need to create a new local working copy of your work, or of the [Trunk](#_What_is_the) (see [SVN Merge](#_SVN_Merge)), or of someone else’s branch (don’t commit anything in this case…). It is strongly recommended to do a checkout inside an empty directory.

## Checklist for making a checkout:

1. Create a new empty directory in which to store the local copy.
2. Open the checkout window (Right click -> SVN Checkout…).
3. Enter the distant address of the branch you want to do a checkout on.
4. Click OK, and Tortoise will download everything. The results will be shown in a new window. Close it and you’re done.

# SVN Branches

## What is a “branch”?

A branch is basically a copy of some SVN repository. The way we use it is very simple: you copy some repository and you start working on it, leaving the original repository unchanged. We say that you “create a new branch”. The big advantage of this method is that you can work on the original repository as well, and then [merge](#_SVN_Merge) modifications brought in both repositories together.

It is very useful when a group of people work on the same project, because everyone can work in his own branch, and there is no interference with the work of other people. But it is also very handy even if you work alone on a project. For example, assume that your code works, but you need to implement a big brand new functionality, which will take one month to develop. So you start working, but after one week, the Chief Developer tells you that there is a bug in your original code and you need to correct it for tomorrow. The problem is, you’re in the middle of developing the new functionality, your code does not even compile anymore, so you cannot release a working version for tomorrow… The good way to do it is to use a branch for developing the new functionality: like this, you’re working on a *copy*, and then when your CD comes and tells you that something is wrong, you correct the bug on the *original* branch, you release the corrected version, and at the end of the month you merge the bug correction and the newly developed functionality, by just [merging](#_SVN_Merge) your two branches.

## What is the “trunk”?

What is commonly called the Trunk is the original repository, parent of every branch. No one should work directly on the Trunk, or commit things directly to it. Every developer has its own branch, copy of the Trunk, and can [merge](#_SVN_Merge) his/her modifications to the Trunk from time to time.

## When should I create a branch?

As explained [above](#_What_is_a), when you need to implement a new functionality, which will take some time to develop and with no possible release before the end of the development. In our case, that should not happen. But if you really need it, ask the Project Manager first.

## Checklist for creating a branch:

1. First [commit](#_SVN_Commit) any local change. Uncommitted changes will not be brought in the new branch.
2. Open the repo-browser (Right click -> TortoiseSVN -> Repo-browser)
3. On the left tab, right click on the repository you want to copy (normally part of your home branch), and select “Copy To”. Do NOT create a directory without doing this “copy” thing.
4. Enter the distant address where to store your new branch. It should be inside your home branch.
5. Click OK and you’re done. You can now do a [checkout](#_SVN_Checkout) on your brand new branch to have a local working copy of it.

# SVN Switch

## What is a “switch”?

A switch simply consists in modifying the distant repository associated with a local working copy. Basically, it’s like deleting the local copy and doing a checkout on another distant repository. The only difference is that SVN will only download files which need to be downloaded, and consequently save bandwidth.

## When should I do a switch?

Every time you need to switch between branches but you don’t want to create a new working copy. Typically, it is useful when you [merge to the Trunk](#_Checklist_for_merging).

## Checklist for doing a switch:

1. First [commit](#_SVN_Commit) any local change. You will lose any uncommitted change.
2. Open the Switch window (Right click -> Tortoise SVN -> Switch…)
3. Enter the address of the distant repository you want to switch to.
4. Click OK and let SVN do its job. Tortoise will open a new window so you can see the changes brought to your working copy. Close it and you’re done.

# SVN Merge

## What is a “merge”?

As explained in the [Branches](#_SVN_Branches) section, a merge consists in merging modifications of two different branches (in general). SVN has a very efficient tool which is able to understand who did what and when; and automatically merge modifications according to this information. However, as explained [below](#_Merge_errors), sometimes it is not able to do it, and then you’ll have to do it manually. An important distinction must be made between the branch you merge *to* and the branch you merge *from*: the branch you merge to is modified by the merge, while the branch you merge from stays unchanged.

## When should I do a merge?

* When you finished the development you were doing on a separate branch -> Merge this branch to your home branch
* When you reached a milestone or any important key point in your development -> Merge your home branch *to* the Trunk
* When you’re notified by the Project Manager that the Trunk was modified (by someone else) -> Merge your home branch *from* the Trunk

## Merge messages

When you do a merge (or an [Update](#_SVN_Update)), SVN will try to merge the modifications brought in the two branches automatically. Here is the list of important messages that could be displayed during this process:

|  |  |  |
| --- | --- | --- |
| Message | Why? | What do I do? |
| Added | A new file was added by the branch you merge from. | No problem, the file is added in the branch you merge to. |
| Deleted | The file was deleted by the branch you merge from, and the other one did not modify it. | No problem, the file is deleted in the branch you merge to. |
| Updated | The file was modified by the branch you merge from. | No problem, the file is updated in the branch you merge to. |
| Merged | File was modified by both branches, but in different sections so an auto merge was possible. | Double click the file name to open the Tortoise Merge Tool, and check that everything’s fine. Note that sometimes it is not really easy to read, so you can just skip this verification and do it in your IDE later. But don’t forget to do it! |
| Merge conflict | File was modified by both branches in the same sections, so an auto merge was impossible. | Double click the file name and click on “Resolve conflict” to open the Tortoise Merge Tool. Then solve the conflicts (in red) by choosing what to put in the final file. Also check the parts which were auto merged (yellow / orange), as described above. Once you’re done, save the final file, close the software and mark the conflict as “Resolved”. |
| Tree conflict | The organization of the directory was changed by at least one of the branches, and SVN cannot know which file to compare to which. This can also happen when you have local uncommitted files with names used by files of the other branch. | This is chaos. Try to reorganize your directory and not to lose any information. Alternatively, call for help or cancel the merge. |

## Checklist for merging *to* the Trunk:

1. **Ask the authorization** of the Project Manager.
2. Be sure that you were working on the last version of the trunk. The best way to be sure is to [merge from the Trunk](#_Checklist_for_merging_1) first. If you were not working on the last version, then by merging you could possibly bring the trunk back to an older version – yours.
3. [Commit](#_SVN_Commit_1) any local change. Uncommitted changes will not be brought to the trunk.
4. [Switch](#_SVN_Switch) your working copy to the Trunk itself, or alternatively do a [checkout](#_SVN_Checkout) of the Trunk in another directory.
5. Now you have a working copy of the Trunk (don’t commit anything!). Open the merge utility (Right click -> TortoiseSVN -> Merge…)
6. Choose “Merge a range of revisions”, and Next.
7. In the URL field, enter the address of your branch, choose “all revisions”, and Next.
8. Don’t touch anything, and Merge.
9. Tortoise will open a window describing what it does. Solve all conflicts as described in the [previous section](#_Merge_messages), check everything. If something went wrong, give up now and destroy your local copy of the trunk (without committing!).
10. Close the window and [commit](#_SVN_Commit_1) (yes now you can) the changes. Write “Merged trunk from branch xxx” or something similar for the log message.
11. You’re done. If you did a [switch](#_SVN_Switch) at step 4, don’t forget to switch back to your branch!

## Checklist for merging *from* the Trunk:

1. First [commit](#_SVN_Commit_1) any local change. Uncommitted changes will be lost in case you have to cancel.
2. Open the merge utility (Right click -> TortoiseSVN -> Merge…)
3. Choose “Merge a range of revisions”, and Next.
4. In the URL field, enter the address of the trunk, choose “all revisions”, and Next.
5. Don’t touch anything, and Merge.
6. Tortoise will open a window describing what it does. Solve all conflicts as described in the [previous section](#_Merge_messages), check everything. If something went wrong, give up now and get back to the last version with a [checkout](#_SVN_Checkout).
7. Close the window and [commit](#_SVN_Commit_1) the changes to your branch. Write “Merged with trunk” or something similar for the log message.
8. You’re done, you can continue your work.

# Deleting, moving, renaming and adding files

Tortoise does his best, but it cannot watch all you do in the Windows Explorer. Consequently, sometimes you have to tell it what you did with your files. Here is a list of what you have to remember:

* When you want to **delete** a file / directory from your working copy, it’s better to use the Tortoise command (Right click -> TortoiseSVN -> Delete). Actually Tortoise is not stupid and if you delete the file the usual way it will work as well. But it is a good habit to use the SVN command.
* When you **move** a file / directory inside your working copy, you have to tell Tortoise that you moved the file. Otherwise, it will think that you deleted the file and that you created another one and you will lose the file history. The easiest way to do that is to cut the file (Ctrl + X) and then paste it to the new location using the Paste command of Tortoise (Right click -> TortoiseSVN -> Paste) instead of the standard Windows Paste.
* The same problem appears with renaming. So to **rename** a file, always use the Rename command of Tortoise (Right click -> TortoiseSVN -> Rename…).
* When you **add** a new file / directory in your working copy, by default it will not be committed with the other files the next time you’ll do a commit. You will have to manually select the file in the Commit window (see the [SVN Commit](#_Checklist_for_committing:) section), and you can easily forget to do it. The best practice is to add the file in Tortoise (Right click -> TortoiseSVN -> Add) immediately after adding it to the working copy.

# About Linux

If you are a Linux fan or if you are working in the lab (or in your virtual machine), you will have to use the SVN commands in a terminal instead of the nice Tortoise UI. It works approximatively the same way but you will not have any graphic interface. Some useful tips and commands (you have to be inside your working copy, except for the checkout command):

* svn checkout [url] will do a [checkout](#_SVN_Checkout) of the specified repository in a subdirectory of your current directory. You will be prompted for a username and a password the first time, of course.
* svn commit -m “message” will [commit](#_SVN_Commit_1) all changes to the server. If you don’t specify the -m option, you will be prompted for a log message. Don’t forget to add (see below) the new files / directories first, or they won’t be committed.
* svn update will [update](#_SVN_Update) your working copy.
* svn add [files] will [add](#_Deleting,_moving,_renaming) specified files / directory. You can use svn add \* to add every unversioned file, but it will skip any already versioned directory. In case you want to also add files which are located in already versioned directory, use svn add \* --force.
* svn move [src] [dest] will move or rename the specified file / directory. Do NOT rename or move files / directories of your working copy without using this command!
* svn delete [files] will delete the specified files / directories. Do NOT delete files / directories from your working copy without using this command!
* I would not recommend to do a merge or to create a new branch with the Linux commands, as the conflicts could be very difficult to handle without a graphic interface.

Actually I heard that there is a version of Tortoise for Linux but I’ve never tried it, so I don’t know if it works as well as the Windows version.