**This document is accessible at:**

**http://bit.ly/Q70nUv**

PREFACE TEXT

This tutorial is built for those using the command line interface. You can follow along in parallel in a GUI (graphical user interface as well), but you might have to do a little detective work. In general, if you can, get a feeling for the command line version- chances are you'll have access to that more often than your favorite GUI version (for example when using a remote client, or logging on to other servers). The command line version also gives you an (arguably) better understanding of what's going on under the hood- which can be important for source control repositories.

\*'ed items are actions you should take by entering commands at the prompt.

indented items should be typed literally

**== Getting started ==**

\* How do you know svn is installed? Open a terminal/command prompt and type:

svn --help

\* If you want to create your own repositories you'll also need the admin tool. Try

svnadmin --help

If either command doesn't work you've either not installed the software at all, or there is a problem with your configuration.

\* You can use the help flag in front of any svn command for more information, e.g.

svn help commit

svn help revert

.. etc.

**== Creating a local repository ==**

SVN can be used anywhere, you need not use a server to take advantage of it (and other source repositories).

\* Navigate to a location (folder) that you'd like to create a new repository in. You'll also be downloading this courses tutorials and related documentation into this folder. If you haven't used a terminal before you'll need to use the 'cd' command to move around (e.g. 'cd myfolder' or 'cd ..'), and 'ls' or 'dir' to list the files/folders in your present location. Ask for help if you need it at this stage, we can go through navigation commands as well.

\* Create a new folder to hold your repositories, let's call it 'src' for source, from the command line it's:

mkdir src

\* Now navigate into that folder:

cd src

\* Create a new repository:

svnadmin create myrepo

\* Do 'ls -al' or 'dir', or examine the folder using your gui- what happened? What's did you create, and what's inside that which you created?

- You won't directly edit any files in this folder, but rather "checkout" a copy of this repository in another location.

- You'll need to determine the full path to your repository (e.g. something like c:/my\_folder/src/myrepo or /users/matt/src/myrepo) at this point. On mac or linux type "pwd" to get the path, or determine it with your file browser. Ask for help if you have trouble here.

\* In the root of your source folder make another folder called "working":

mkdir working

\* Now move into that folder:

cd working

\* And checkout your repository (you'll need the full path that you found with 'pwd' here plus "myrepo"):

svn checkout file:///Users/matt/tmp/src/myrepo

- If you see 'Checkout out revision 0' you've succeed!

\* What happened? What was created in your 'working' folder?

\* Navigate into the myrepo folder. What was created here? hint - do a "ls -al". How does your operating system (e.g. finder or explorer) handle files or folders prefaced with a "."?

- If you see files there, these are the hidden svn management files. If you see them, don't delete them! They also have consequences if you copy and paste folders (don't!).

**== Adding files to your repository ==**

\* Navigate inside your new folder

cd myrepo

\* Create a new text file inside this folder, use whatever mechanism you wish, if on linux or mac your can use "touch":

touch "first.txt"

\* Edit the file with your text editor, add a line that says 'revision 1'. Save the file

\* To see the status of your repository type:

svn status

- You'll see '? first.txt'

- The "?" indicates that the file is not yet under subversion control.

\* Let's add it to the repository:

svn add first.txt

- You should see 'A first.txt', which is also what you'd see if you do 'svn status'

\* Let's commit the change:

svn commit -m "my first commit"

- The -m flag designates a message to be added on commit. It's very important to include a message with each commit. If you omit the -m svn will likely open a text editor (as defined in its configuration), type your message there, save and close the editor, and the commit will automatically continue.

**== Diffing ==**

\* Edit your first.txt file, change the text to "revision 2"

\* Do

svn status

- You'll see a "M" beside first.txt, this means that file has been modified since the last commit.

\* Commit your file

svn ci -m 'a second commit'

- Notice we used the short form of the commit command 'ci' nearly all svn commands have short forms.

- If successful you'll see several messages, the last being: 'Committed revision 2.'

- SVN records every change ever made, let's look at the difference between commits 1 and 2:

svn diff -r 1

- This command does a "diff", it compares the contents of the current working copy with the revision indicated by the -r flag, you should see something like this:

Index: first.txt

===================================================================

--- first.txt (revision 1)

+++ first.txt (working copy)

@@ -1 +1 @@

-revision 1

+revision 2

- The --- and +++ lines indicate what is being compared.

- The - line refers to an actual line in the file, here we see that one line was deleted, and another was added.

- The remaining format is use by "patching" tools, i.e. you can automatically apply a diffs to other files.

- Edit your text file and add some random text. Don't commit the change. Use 'svn diff' as previously to examine the results. Try adding lines before and after the 'revision 2' line before running the "diff" to get a feel for what is happening.

- Most people start understanding and using diffs using graphical tools. Your instructors will show you a couple examples during this tutorial.

**== Reverting ==**

\* Let's say you just flat out messed up your current version, or you want to return to an older state. With your edited, but not committed file do:

svn revert first.txt

\* Now do

svn status

\* Notice the "M" line is no longer there- your file has reverted to its state immediately of its last commit.

! In general you should frequently use 'status' commands to keep track of your changes.

**== Using 'update' ==**

\* You can update any file(s) to a given revision with 'update':

svn up -r 1 first.txt

\* Check the contents of the file- what does it say?

\* If you use 'up' without a -r (revision) flag you get the latest, or HEAD version:

svn up

- You should see a message ending with "Updated to revision 2."

**== svn info ==**

\* the 'info' command is handy way to see where your repository is stored, and a little more about it

svn info

**== Best practices ==**

- Commit early and often, commit small changes, it makes it easier to revert

- Always do 'svn up' \*before\* you start to work on a file- in the examples we'll get to next changes will come from others working on files in the repo.

- Always provide verbose messages- they become very useful when looking at the history of a file.

**== Remotely managed files ==**

- In this exercise we're going to collaboratively edit a file to build a "pass it on" story.

\* To start move back to the your "src" folder

\* Check out a tutorial folder from the course SVN repository (later we'll check out the whole repo).

svn checkout https://nescent-anatomy-course.googlecode.com/svn/trunk/material\_for\_course/mon/svn\_tutorial --username your\_name@gmail.com

\* Enter the password for your Google Code account if provided. (See bottom of this document for how to find your password).

\* What happened?

\* Navigate into the folder that was created. You should see a 'story.txt' file.

- At this point we're going to take turns editing the documents, the instructors will number you off. When it's your turn you'll add a line in the story to the story.

\* To add your line first always update your repo before you begin working:

svn up

\* Now edit 'story.txt' and add a single line that continues the story.

\* Finally, commit your changes back to the repository and announce to others it's their turn to edit:

svn ci -m 'your\_first\_name changes'

\* You may be prompted for your Google code account password (this is not the same as your Google account password, see the bottom of the file for how to find this).

\* You can follow along on the progress of the story by doing svn up as other's commit their changes.

- Now in real life things aren't nicely ordered in steps.

\* With our finished story make another change (it can be minor), and try committing back to the repository, don't wait for anyone else. What happens?

- You likely find that there are "conflicts". An instructor will quickly go over how to resolve conflicts, but we'll not go in to depth here.

\* When you are finished delete this repository. Are the files really gone?

**== Working with ontologies ==**

- Even if you don't have write access to a repository it's useful to use SVN to keep up to date on changes (as opposed to finding the files on line and downloading, which is also straightforward, but not quite as efficient).

- Let's download an ontology from the OBO foundries repositories.

\* Navigate back to the root of your src folder and type:

svn co http://pato.googlecode.com/svn/trunk/ pato

\* What happened? Notice we downloaded files into the "pato" folder, as designated by the option folder name after the repository address.

\* Navigate into the pato folder and open quality.obo in a text editor. Make a change at random, save, and close your file.

\* Try:

svn status

- You should see 'M quality.obo'

\* Just for fun try committing your changes:

svn ci -m 'breaking pato ftw!'

\* What happens? Why?

**== Setting up your course repo ==**

- We'll now download the course repository. You'll have to access this repo regularly to retrieve assignments, upload answers, and download slides.\* Navigate to your "src" folder (or anywhere that you'd like to keep your source files, then checkout the files

svn checkout https://nescent-anatomy-course.googlecode.com/svn/trunk/ nescent-anatomy-course --username your\_username@gmail.com

\* !! Enter the password provided by Google (NOT your gmail password). You can find this password via information at the bottom of this file.

\* Voila! You have checkout out the course materials. Everyone has permission to commit to this repository, whoa- isn't that dangerous!? Why not?

- Your personal space is in the /users subfolder corresponding to your first name.

**== Advanced Topics ==**

- Given available time the instructors will go over these topics

- Gotchas - There are several:

- API vs. filesystem

! copy, paste, rm

- .svn files

- corruption

- And time permitting:

- Visualizing diffs using SmartSVN

- Resolving conflicts

- Branching/Tagging

- Merging

- Alternatives to SVN

**== Your Google Code account password ==**

Note that you need to be signed in to your Google account first.

Find your password through the link here:

http://code.google.com/p/nescent-anatomy-course/source/checkout

Click on the link that is in here: "When prompted, enter your generated googlecode.com password." You’ll see a big crytpic password.