Computer Science 507
Software Engineering
The College of Saint Rose
Spring 2014

Lab 5: Source Code Control

Due: 6:00 PM, Monday, March 10, 2014

In this week's lab exercises, we will be experimenting with some of today's more popular source code control systems: Subversion and Git.

You should work in groups of size 2 to 4 for these exercises. Ideal groups would combine members who have some experience with version control who can guide the team through the basics, and others for whom this is new. If you don't know the basics, don't let your team get ahead of you before you understand what is going on!

Version Control Basics

Read about version control in Section 25.5 of Sommerville, and the Version Control Basics section of the free online book, Version Control with Subversion.

Lab Question 1:
Even if working independently on a software project, describe briefly why a source code control system would be helpful to manage development. Consider both multiple development environments (*e.g.*, a portable computer as well as an office desktop computer, both used in development) and the need for revision history. (3 points)

Subversion

Read the sections Version Control the Subversion Way and Chapter 2, Basic Usage (except the "Dealing with Structural Conflicts" section) in Version Control with Subversion.

I have created a shared subversion repository for all of us to try out on `mogul.strose.edu`. It was created with the command:

```
svnadmin create /home/cs507/s14/svn-lab/svn
```

I then added the typical directory structure to include "`branches`", "`tags`", and "`trunk`" directories in the root of the repository, using these commands:

```
mkdir -p /tmp/svn/{branches,tags,trunk}
svn import /tmp/svn file:///home/cs507/svn-lab/svn -m "initial import"
```

Once these commands were executed, the repository was ready to be used. So let's use it.

First, you (all group members) will need to set your permissions mask, so that when you commit to our group repository, the permissions will remain "group writeable".

Please edit the file `~/.bashrc` on `mogul.strose.edu` so that it contains the command

```
umask 002
```

This will mean files that you create, will, by default, include group write permissions. Combined with the fact that I have set the `svn-lab` directory to have "set group-id" permissions, this should allow all of us to share access to the repository.

Once you have edited the file, log out and log back in to make sure it has taken effect.

You can then "check out" a copy of our repository.

To avoid potential conflicts (which we will reintroduce intentionally later on), we will take turns working through this section of the lab. Please do not proceed until your group is cleared to do

Create an empty directory where you would like to do this, and issue one of the following commands. If on `mogul.strose.edu`, you can check out with the `file` protocol.

```
svn checkout file:///home/cs507/s14/svn-lab/svn/trunk svn
```

If on a different machine, you will need to use the `svn+ssh` protocol.

```
svn checkout svn+ssh://you@mogul.strose.edu/home/cs507/s14/svn-lab/svn/trun
```

In either case, you should now have a directory called `svn` which contains the contents of the "trunk" of the repository.

? Lab Question 2:
What is in that directory when you first checked it out? (1 point)

You should see at least a file called `CLASSFILE` in your working copy directory.

? Lab Question 3:
Add a paragraph for yourself or your group to `CLASSFILE` and commit your change with an appropriate commit message. Paste the output you get when you run your command. (2 points)

For example, the following will commit your version of `CLASSFILE` with a good commit message.

```
svn commit -m "CLASSFILE addition for Alice, Dilbert, and Carol" CLASSFILE
```

This is the end of the part of the lab where we need to take turns. If your group is working on this part, please announce when you get to this point so the next group can get started.

Now, create another file in our class repository. It should be a program in any programming language you wish that prints out the names of your group members, and anything else you wish.

? Lab Question 4:
What does the `svn status` command tell you about your file after you created it? (1 point)

After you create your file, you still need to tell `svn` about it. You will want to `add` it to the repository with `svn add`.

? Lab Question 5:
What does the `svn status` command tell you about your file after you add it? (1 point)

It is still not committed. `svn commit` will be needed for that.

? Lab Question 6:
What does the `svn status` command tell you about your file after you commit it? (1 point)

Next, using a different group member's account, check out a new working copy of the repository (or update it if you had already checked it out previously). We'll call this "copy B", and the one you were working with previously "copy A".

? Lab Question 7:
List the commands you used to do the above. (2 points)

In each copy of the repository, make a change to the program you added. To avoid conflicts, make sure each copy is edited in a different place in the code.

Commit the file from copy A to the repository, then attempt to commit the file from copy B.

? Lab Question 8:
What output do you get when you attempt the second commit? (1 point)

Assuming the intent is to get both changes into the repository, we need to update copy B after committing from copy A other, then commit from copy B, and update copy A.

? Lab Question 9:
List the commands you used to achieve this. (2 points)

Git

If you are not familiar with Git, read Learn GitHub's Introduction to Git. Don't worry so much about the pages beyond the "Normal Workflow" page.

We will do a similar exercise as above, but with a repository I have created at GitHub.

If you do not already have a GitHub account, create one and log yourself in.

Next, clone a copy of my repository from `https://github.com/terescoj/cs507s13lab.git` onto mogul or onto your own computer (you'll need the "`git clone`" command).

Note: if you get an error about certificates missing, you can issue the command

```
git config --global http.sslVerify false
```

and the error should go away.

? Lab Question 10:
What output do you get when you issue the appropriate "`git clone`" command? (1 point)

You should also at this time set your name and email address to use for git to use to identify you:

```
git config --global user.name "Your Name"
git config --global user.email you@example.com
```

In your cloned working copy, edit the file `CLASSFILE` much like you did for the subversion part.

? Lab Question 11:
What output do you get now from a `git status` command? (1 point)

Commit your new version of `CLASSFILE` (`git commit`).

? Lab Question 12:
What output do you get from the commit command, and from a subsequent `git status` command? (1 point)

Note that your update file is in your clone of the repository, but it is not back in the original repository on GitHub. You can verify this by looking at it via the `https://github.com/terescoj/cs507s13lab.git` URL.

An additional command is needed to do this: "`git push`"

? Lab Question 13:
What output do you get from the push command, and from a subsequent `git status` command? (1 point)

Note: this will not work with our current setup. Just give the error message you see.

Verify that your changes now show up on the original repository on GitHub.

Next, add a new file to your working directory, add it and commit it to your local repository. You may use the same program you used for the subversion part or write another.

Lab Question 14:
Show the commands you used to do the above and the output they produced. (2 points)

Note: you should be able to add and commit to your local repository clone but will not be able to push back to the original GitHub repository.

Submission

Before 6:00 PM, Monday, March 10, 2014, submit your lab for grading. Package up all required files into an appropriate archive format (`.tar.gz`, `.zip`, and `.7z` are acceptable) and upload a copy of the using Submission Box at `http://sb.teresco.org` under assignment “SCC”.

Grading

This lab will be graded out of 20 points.