Computer Science 507
Software Engineering
The College of Saint Rose
Spring 2013

Lab 5: Source Code Control

Due: 6:00 PM, Monday, March 18, 2013

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 may work alone or in groups of 2 or 3 for these exercises (though groups are better). The ideal groups would include at least one person who uses version control regularly and can guide the team through the basics. But 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.

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 up 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/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'll 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` 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. 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/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/svn-lab/svn/trunk sv
```

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

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

You should see a file called `CLASSFILE` in your `svn` directory.

Question 3: Add a paragraph for yourself or your group to `CLASSFILE` and commit your change with an appropriate commit message. (1 point)

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
```

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.

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`.

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.

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

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.

Question 7: 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.

Question 8: What output do you get now from a `git status` command? (1 point)

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

Question 9: 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`"

Question 10: 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, and push it back to the original repository on GitHub. You may use the same program you used for the subversion part or write another. (3 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.

Finally, create a private repository on GitHub, and share it only among your team and with your instructor. The details are left as an exercise. Include the URL of your repository in your submission. (5 points)

Note: this part of the lab is postponed until we can find out about possible free, temporary options to create private repositories.

Submission and Grading

To submit the assignment, send your responses to the questions above to *terescoj@strose.edu* by 6:00 PM, Monday, March 18, 2013. Please include a meaningful subject line (something like "CS507 Lab 5 Submission").

This lab will be graded out of 20 points.