Introduction to Subversion

Jamie Fairbrother

January 5, 2016
1. Introduction

2. Creating a repository and putting a project under VC

3. Basic Work Cycle
What is version control?

- Version control systems keep track of all changes made to files in a project
- Users can review and undo changes
- Allows several people to work on project:
  - Mutually exclusive changes are automatically merged
  - Changes which clash do not result in someone’s changes being automatically
How does Subversion work?

- Subversion stores all changes made to a project on in a database called a **repository**
- A user wanting to work downloads the latest version of the project to their computer - this is called the local **working copy**
- When the user is done making changes to the project, they **commit** their changes to the repository
- The computer which stores and manages this repository is called the server.
- The computer that downloads a project from a repository, and submits changes is called a client.
- The server and client may be the same computer.

When changes are committed to the repository, the global revision number for the repository is incremented by one.
Creating a repository

- Run the following commands in your home directory:

```
1. mkdir svn  # Make directory to store repository
2. svnadmin create svn  # Create repository in directory
```

- This creates an empty repository in the directory `svn`
Importing a project

- Go to your project directory and delete any files that shouldn’t be versioned

```bash
1. cd my_project
2. rm -f *.o *~  # Remove binary and backup files
```

- Run the following command to add this project to the repository:

```bash
1. svn import file:///home/jamie/svn/my_project/trunk /
   -m 'Initial import'
```
• The prefix `file://` specifies the path of a repository on the local machine.

• The `my_project/trunk` specifies the virtual location in the repository where the project resides.

• It is conventional to put the working files of a project in a directory called trunk but this is not required.

• The `-m` flag specifies a required log for the activity.
Checking-out a project

- The project in my_project is not under version-control
- To get a version-controlled copy you must check the project out of the repository:

```bash
mv my_project my_project_temp  # rename current copy
svn checkout file://home/jamie/svn/my_project/trunk / my_project  # Checkout project
```

- The second argument of `svn checkout` specifies where the project should be checked out to
- Once you have verified that the project has been successfully downloaded from the repository you can delete the old copy:

```bash
rm -rf my_project_temp
```

Jamie Fairbrother
Introduction to Subversion
January 5, 2016 8 / 17
Basic Work Cycle

1. Update working copy
   ▶ svn update

2. Make changes to files
   ▶ svn add
   ▶ svn rm
   ▶ svn cp
   ▶ svn mv

3. Examine your changes
   ▶ svn status
   ▶ svn diff

4. Possibly undo changes
   ▶ svn revert

5. Resolve conflicts
   ▶ svn update
   ▶ svn resolve

6. Commit your changes
   ▶ svn commit
If you keep working copies of your project on multiple computers, it is important that you update your working copy before you begin modifying it to avoid conflicts.

Use the following command to do this:

```
svn update
```
Adding, moving, copying etc.

- To add a new file to the repository use `svn add`:
  ```
  svn add wavelet.cpp  # add wavelet.cpp to repository
  ```

- To delete a file use `svn rm`
  ```
  svn rm gibbs_sampler.R  # delete gibbs_sampler.R
  # and remove from repository
  ```

- To move a file or change its name use `svn mv`
- To add a new directory to repository use `svn mkdir`
  ```
  svn mkdir include/  # create new directory include
  # and add to repository
  ```

- Note that the `svn rm`, `mv` and `mkdir` subcommands work just like the corresponding unix commands.
In the preceding examples, the `svn` commands act immediately on local working copy but are not propagated to the repository until the next commit.

Some `svn` subcommands (including the four above) can be used with URLs to the repository, rather than local paths.

```
1. `svn mkdir svn://193.113.58.233/wsd_calc/branches
   -m 'Created branches directory'
2. `svn cp svn://193.113.58.233/wsd_calc/trunk
   svn://193.113.58.233/wsd_calc/branches/my_branch
   -m 'Copied trunk to a new branch'
```

In this case, the changes are made directly on the repository.
Examining changes

- The exact differences between two versions of a file can be examined using the `svn diff` command:

```
1      svn diff Makefile  # Compare working copy of
2          # Makefile to base version
3      svn diff filter.py -r 205:225  # Compare version 225
4          # with version 205
5          # of plotter.py
```

- To see a list of files which have changed since the last commit use `svn status`

- The output uses the following code:

<table>
<thead>
<tr>
<th>M</th>
<th>File has been modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>File has been added</td>
</tr>
<tr>
<td>R</td>
<td>File has been removed</td>
</tr>
<tr>
<td>C</td>
<td>File in state of conflict</td>
</tr>
<tr>
<td>?</td>
<td>Unknown file</td>
</tr>
</tbody>
</table>
Logs from commits can be viewed using the `svn log` command:

```
1  svn log changepoints.M # View all logs for changepoints.M
2  svn log changepoints.M -r 13:PREV
3    # View log from revision 13 to previous commit
```
Undoing working changes

- To undo changes made to your local working copy use `svn revert`

```
svn revert bandits.h  # undo changes to bandits.h
```

- Note that this command only changes files in your working copy and not the repository
Commiting changes to repository

- When you are done making changes to your local copy, you should commit them to the repository:

  1. `svn commit`  # Commit all changed files in working copy
  2. `svn commit file1 file2`  # Commit only file1 and file2

- Any conflicts with the repository will be flagged up at this point
Getting help

- For a list of all available subcommands run `svn help`
- For usage information on a particular subcommand run `svn help subcommand`

- The book “Version Control with Subversion” published by O’reilly is recommended
- This is freely available online at [http://svnbook.red-bean.com/](http://svnbook.red-bean.com/)