Making the most of version control

SVN for CESM Users, Scientists, and Developers

Steve Goldhaber

National Center for Atmospheric Research
CGD: Atmospheric Modeling & Predictability

May 9, 2014

Outline

SVN for CESM Users, Scientists, and Developers

- Why source control?
- What is source control?
- Basic Operations
- Sleuthing with history
- Branching and Merging

Why Use Source Control?

- **Collaboration**: A Source Control System allows you to work safely with multiple groups of multiple people
- **Safety**: A Source Control System stores previously committed changes so they are not lost
- **History**: A Source Control System allows you to see or use an old version, see what happened when, and see who did it

Note: This talk will use svn as the source control system but the concepts apply to any (modern) SC system.

What is a Source Control System?

- Files/code/data exist in the repo(sitory), and you have your own local working copy.
  Therefore: What you do to your local copy does not affect anyone else (until you commit).
- If something was in the repo at revision r, even if it's been removed since, then you can ALWAYS go back to revision r to get it.
  Therefore: Get rid of code/files/branches once you are done with them.
- The Source Control System provides an interface in the form of a set of commands to manage your files.
  Therefore: Always use svn commands to make changes to a repo. **Do not try to outsmart svn!** . . . Learn from our mistakes.

What do I need to know about a repository?

- A repository (repo) is like a directory which stores many versions (snapshots) of its contents.
- Along with each snapshot, a repo stores logging information describing significant information about that snapshot.
- The files you work on are called a working copy. They are not in the repo.
- The Source Control System (e.g., subversion or svn) refers not just to the repository, but to the command set (interface) that allows you to interact with it.

What can I do with a repo and svn?

- **checkout** (get a copy of part or all of a repository)
- **list** (find out what is in the repository)
- **update** (put newer versions into your working copy)
- **commit** (put changes in your working copy back into the repo)
- **add** new files (directories) to the repo or **remove** existing files (directories)
- **status** (find out how your working copy compares to the repo)
- **log** (learn about the history of any part of the repo)
- **diff**, **blame**, **merge**, . . .
- **revert** (recover from [almost] any mistake[†])

[†]If you have been using the repo and svn as described in this talk

Checkout

- Use **svn checkout** (or **svn co**) to make a working copy of a (portion of a) repository version:

```
% svn co https://pio.gcode.com/svn/trunk
```

will create a directory called *trunk* in your current directory.

- If you don't like that directory name, you can choose your own:

```
% svn co https://pio.gcode.com/svn/trunk pio
```

will create a directory called **pio** in your current directory.

- You can also checkout an older revision:

```
% svn co -r944
  https://pio.gcode.com/svn/trunk pio944
```

will create a directory called **pio944** in your current directory that contains a copy of the trunk directory as it was when checkin 944 was made.

What is there besides trunk?

- Use **svn list** (or **svn ls**) will show you what is in the repository at that location (very similar to the Unix ls command).

```
% svn ls https://pio.gcode.com/svn
    branch_tags/
    branches/
    genf90/
    libpiovdc/
    ncReshaper/
    sa_trunk/
    trunk/
    trunk_tags/
    wiki/
```

Let's check out a trunk tag

```
% svn ls https://pio.gcode.com/svn/trunk_tags
    pio1_0_0/
    pio1_0_1/
    pio1_0_10/
    ⋮
    pio1_8_7/
    pio1_8_8/
    pio1_8_9/
% svn co
    https://pio.gcode.com/svn/trunk_tags/pio1_8_8

    Where is my code?
% cd ??
```

How do I update my code?

- Bob tells me he made a fix to the code. How can I get the fix?
- The easiest way is to **update** your working copy from the repo (first cd into the directory you checked out):

```
% svn up
```

  This will update every file which has been changed in that repo directory since your last checkout or update

- Files which have been changed in the repo are updated in your working copy
- Files which were deleted are removed from your working copy
- Files which were added are added to your working copy
- Properties are also updated (more on that later)

Adding a new file

- First, add the file in your working copy
- Next, tell svn that this file should be added to the repo:

% svn add <filename>

- **svn add** also works for adding a whole directory
- To create an empty directory and add it to the repo:

% svn mkdir <dirname>

Moving, copying or removing a new file

-
- To remove a file from your working copy (and schedule it to be removed from the repo): **% svn rm <filename>**
- To move a file: **% svn mv <oldname> <newname>**
- Same or different?

% svn cp <oldname> <newname>
% svn rm <oldname>

Same!

- **To copy a file:**
  % svn cp <oldname> <newname>
- **Same or different?**

% cp <oldname> <newname>
% svn add <newname>

Different (no history)!

Updating the repo

- The **update**, **add**, **mv**, **rm**, and **cp** commands make changes to your working copy but do not update the repository.
- The simplest way to save them in the repo is to **cd** into the top level of your working copy and **commit**:

```
% svn ci
```

- Opens an editor (which you can customize with SVN_EDITOR). Write a meaningful message, save, and exit the editor..
- Sends changes from your working copy to the repository.
- You can commit a subset of your working copy by specifying files and directories.
- Always put thoughtful messages; You will find them useful

What gets committed?

- Use the **status** command to see how your working copy is different from the repo:

% svn st

```
...
M  models/atm/cam/src/dynamics/se/dp_coupling.F90
X  models/atm/cam/src/dynamics/se/share
M  models/atm/cam/src/dynamics/se/stepon.F90
...
```

- For a full explanation of all the status codes, see: http://gotofritz.net/blog/howto/svn-status-codes/
- **svn status -u** will show you which files have been updated in the repo.
- **svn status -v** will show you status information on every file in your working copy

I forgot what I changed

- Say that **svn st** shows a modified file, e.g.,

```
M  models/atm/cam/src/dynamics/se/stepon.F90
```

- Use **svn diff** to see the modifications in the working copy:

% svn diff models/atm/cam/src/dynamics/se/stepon.F90

- Output works just like the Unix diff command
- By default, the differences are between the working copy and the version you last checked out or committed.
- To see the difference between your copy and a different version:

% svn diff -r44444 models/atm/cam/src/dynamics/se/stepon.F90

- To see the difference between two repo versions:

% svn diff -r44444:44445 models/atm/cam/src/dynamics/se/stepon.F90

What happens to all those commit messages?

- **svn log** will show you all the commit messages
- To control long output, try **svn log | less**
- To see the log for just one file:
  svn log <filename> | less
- To see the changed files as well as the messages:
  svn log -v | less
- To see the verbose log for just one revision:
  svn log -v -r555
- You can use the **log** command on the repo as well:

svn log https://pio.gcode.com/svn/trunk

- There is a shortcut to the repo if you are in a working copy: **svn log ^/trunk**

What are all these revisions?

- **svn** maintains information on each commit and numbers them.

```
r856 | edwards.jim@gmail.com | 2013-11-19 14:48:54 -0700
     (Tue, 19 Nov 2013) | 1 line

Fixes for problems found in the build of cesm1_3_alpha06c
------------------------------------
r854 | edwards.jim@gmail.com | 2013-11-14 11:21:35 -0700
     (Thu, 14 Nov 2013) | 1 line
add support for PIO_64BIT_DATA
```

- What happened to r855?
- That commit happened in a different directory (branch).

Trunks & Branches & Tags, Oh My!

- branches, tags, trunk_tags, trunk, etc. are not special svn entities.
- By convention, we set up a repo with directories called trunk, branches, etc. but it is just a convention.
- We'll cover conventional use of these repo directories so don't panic.

Where am I?

% svn info .

```
Path:  .
URL: https://pio.gcode.com/svn/trunk ← What you checked out
Repository Root:  https://pio.gcode.com/svn
Repository UUID: 144a4905-da4a-0410-ac61-cbb8a8090720
Revision:  762 ← The latest revision when you checked out
Node Kind:  directory
Schedule:  normal
Last Changed Author:  edwards.jim@gmail.com
Last Changed Rev:  759 ← The last rev. in your branch (trunk)
Last Changed Date:  2013-04-02 14:52:15 -0600 (Tue, 02 Apr
    2013)
```

Conflict

- During an update or a merge, you run into a conflict:
- If you use vi, you will find sections like this:

```
Conflict discovered in 'pio/CMakeLists.txt'.
Select:  (p) postpone, (df) diff-full, (e) edit,
(mc) mine-conflict, (tc) theirs-conflict,
(s) show all options: e
<<<<<<< .mine
your version
=======
version from the repo in revision 1054
>>>>>>> .r1054
Select:  (p) postpone, (df) diff-full, (e) edit, (r) resolved,
(mc) mine-conflict, (tc) theirs-conflict,
(s) show all options: r
```

Who did that?

- You see an odd line of code and wonder, who did that?
- $dv = 8.794E\text{–}5\_r8 * t * *1.81\_r8/\infty$
- OK, now you wonder, why did he do that?
- Take note of the offending revision number, 48765

% svn blame micro_mg1_5.F90 | less

...

38788 santos@uca $rho = p/(r * t)$

48765 goldy@ucar $dv = 8.794E\text{–}5\_r8 * t * *1.81\_r8/\infty$

38429 santos@uca $mu = 1.496E\text{–}6\_r8 * t * *1.5\_r8/(t + 120.\_r8)$

...

Who did that?

- You see an odd line of code and wonder, who did that?
- $dv = 8.794E{-}5\_r8 * t * *1.81\_r8/\infty$
- OK, now you wonder, why did he do that?
- Take note of the offending revision number, 48765

% svn log -v -r48765 micro_mg1_5.F90

```
------------------------------------
r48765 | goldy@ucar.edu | 2012-10-12 11:28:41 -0600 (Fri, 12
    Oct 2012) | 1 line Changed paths:  M
    /cam1/branches/MG2_cam5_1_xx/models/atm/cam/src/physics/cam/micr

Fix floating point overflow - On to 5-week vacation with no
    email
------------------------------------
```

Mulligan

- When you decide that the changes you made in your local copy of <file> need to go away:

```
% svn revert <file>
```

- To undo all changes in a directory (<dirname>) and its sub-directories:

```
% svn revert –recursive <dirname>
```

- To undo changes to a directory's properties:

```
% svn revert <dirname>
```

- revert undoes changes to your working copy, even pending changes:

```
% svn rm important.F90
% svn revert important.F90
```

- 'Changes to your working copy' are relative to the version you checked out or last committed.

Quiz

- Same or different?

```
% svn revert foo.F90
```

vs.

```
% rm foo.F90
% svn up foo.F90
```

- Different! The svn up call might pull in a newer version of foo.F90 than the one you were working on.

Branching

- When you add new features to a project or begin any sizeable changes, you should create a branch.
- Branching is easy to do (once I show you how).
- There are lots of advantages to branching vs. working in trunk:
    - Your code need not pass tests at the end of the day . . . in fact, it doesn't even need to compile.
    - Your code is backed up if you commit it, which you may not want to do in the trunk.
    - You can collaborate with others.
    - Not messing up the trunk means never having to say you're sorry.
- Branch more often than you think you need to.

A bit about branches

- By convention, a branch goes in the branches directory at the top of the repo
- Typically, a branch will begin with a complete copy of the trunk
- As scary as that sounds, a new branch takes up almost no space (there goes another excuse not to branch)
- Your group will usually have some naming convention for branch names but svn really doesn't care
- So what's the difference between the trunk and the branch?
  The name!
- trunk is just another directory to svn. We treat it specially because we use it as our source for new releases.

Making a Branch

- Make a branch by using svn copy directly in the repo

```
% svn cp <URL>/trunk
      <URL>/branches/my_new_branch
```

- This creates the branch in the repo but doesn't check it out into a working copy:
- You will be asked for a commit message – explain the purpose of the new branch

```
% svn co <URL>/branches/my_new_branch
```

- Forgot the URL? Use svn info to find it
- If you are in a working copy of the repo, you can use the ^ shortcut:

```
% svn copy ^/trunk ^/branches/my_new_branch
% svn switch ^/branches/my_new_branch .
```

NB: switch will try to preserve your working-copy changes. Make sure this is what you want.

Tags

- A tag is a snapshot of the code at a certain revision.
- Why tag?
    - It's a release!
    - A bug-fix for a desperate user
    - Tests passed, let's tag that!
- What is a tag?
    - A tag is simply a copy of the trunk or a branch
    - A tag is no different than a branch, we just put them in a different directory by convention
    - Tags are typically not modified

Making a tag

Tags seem to be scary. There are three important things to remember about making a tag

1. Making a tag is no different from making a branch
2. Making a tag is no different from making a branch
3. **Making a tag is no different from making a branch**

```
% svn cp <URL>/trunk
    <URL>/trunk_tags/version1_2_297
```

Properties

- Directories and files in the repo can have metadata, which are called "properties"
- svn provides several functions for managing properties
- **proplist** will output a list of properties set for the current dir

```
% svn pl .
```

- **propget** will get the value of property, <propname>

```
% svn pg <propname>
```

- **propedit** allows you to edit the value of a property

```
% svn pe <propname> .
```

- **propset** will set the value of a property

```
% svn ps <propname> <value> <file or dir>
```

- The most common property is the svn:externals property (but I'm not going into details today).

Merging

- svn merge applies the differences between two sources to a working copy path.
- The two sources can be any two directories in your repo or a directory in your repo and your working copy
- The destination of the merge is always your working copy
- If your current directory is your working copy, **svn merge** will use that automatically
- Why merge?
    - Keep your branch up to date with changes to the trunk
    - Move your branch changes into the trunk

Keeping your branch up to date with the trunk

- It is wise to keep your branch close to the trunk with frequent updates. This makes merging your new code back into the trunk much easier in the end.
- First, cd into the top level of your working copy
- Make sure your local changes are checked in (**svn ci**).
- Merge in the trunk changes:

% **svn merge ^/trunk**

- This will merge changes into your local copy.
- After merging, you still need to commit your changes:

% **svn ci**.

- If conflicts arise during the merge, you resolve them just as with **svn update**.

Example: Merge branch into the trunk

- Check out a working copy of the trunk (or make sure your working copy is up to date with **svn up**).
- cd into the top level of the trunk working copy
- Merge in the branch

```
% svn merge ^/branches/my_new_branch
```

- This will merge changes into your local copy.
- Since no revisions were specified, svn will merge from the point the branch was first created (with svn cp) up to the current revision (HEAD) of the branch.
- The next time you merge this branch into the trunk, svn will pick up from where it left off
- What if you want to merge only changes between rev 200 and rev 222?

```
% svn merge -r200:222 ^/branches/my_new_branch
```

- Merge, test, and commit!

Thanks!

Questions?

- For some introductory svn material including local CGD information, go to: http://www.cgd.ucar.edu/systems/documentation/faqs/computing/subversion_info.html
- For more on svn, see the “Red Bean” svn book: http://svnbook.red-bean.com/

Contact: goldy@ucar.edu

Acknowledgments: Thank you DWS, I only steal from the best!