2068-14

Advanced School in High Performance and GRID Computing - Concepts and Applications

30 November - 11 December, 2009

Introduction to Subversion

A. Messina
ICTP
Trieste
Italy

Introduction to Subversion

Antonio Messina <amessina@ictp.it>

The Abdus Salam International Centre for Theoretical Physics, Trieste

November 1st, 2009

Why do you need a (centralized) revision control system?

history
: all committed versions of every single file are maintained forever

Why do you need a (centralized) revision control system?

history	all committed versions of every single file are maintained forever
availability	project is securely accessible from everywhere

Why do you need a (centralized) revision control system?

history	all committed versions of every single file are maintained forever
availability	project is securely accessible from everywhere
sharing	several people can contribute to the project (easily)

history

- Changes to a file (or group of file, or directory. . .) are *committed* under a new **revision number** each time

glossary:

commit	when a change is uploaded on the repository
diff	program to show textual differences in files

history

- Changes to a file (or group of file, or directory. . .) are *committed* under a new **revision number** each time
- these incremental changes reflects the evolution of the project

glossary:

commit	when a change is uploaded on the repository
diff	program to show textual differences in files

history

- Changes to a file (or group of file, or directory. . .) are *committed* under a new **revision number** each time
- these incremental changes reflects the evolution of the project
- at every commit a log entry is added by the committer

glossary:

commit	when a change is uploaded on the repository
diff	program to show textual differences in files

history

- Changes to a file (or group of file, or directory. . .) are *committed* under a new **revision number** each time
- these incremental changes reflects the evolution of the project
- at every commit a log entry is added by the committer
- you can show *diff*erences between two different revisions or between your working copy and the current revision.

glossary:

commit	when a change is uploaded on the repository
diff	program to show textual differences in files

availability

- Versions of a document are tracked in a single place, the repository
- Documents are accessible over the internet, using secure protocols.

SVN URLs:

Url of a svn repository are in the form

```
http://hostname/path/to/repository/subtree
https://hostname/path/to/repository/subtree
svn+ssh://username@hostname/path/to/repository/subtree
file:///home/amessina/svn/foo/bar
```

sharing

- Date and time of a new revision is maintained along with the user who committed it
- It is possible to give different privileges to different users
- It is possible to create *branches* and *tags*
- Changes made to the same file by different users can be merged automatically or manually
- Changes made to a *branch* can be merged to the the *trunk* tree

Checking out the repository

You always work on a **working copy**, local to the machine.
To create a working you have to do a **checkout**:

```
$ svn checkout [--username username] [URL]
```

This command gets the latest version of the files contained in the repository associated with URL and create a working copy in the current directory

URL for the HPC school is

```
https://svn.gforge.escience-lab.org/svn/hpc-2008/
```

svn command syntax

svn [subcommand] [options] [url]

Glossary

repository	server in which **all** the revisions, log entries, copies of the project are stored.
revision	a number which refers to a particular *state* (snapshot) of the repository
working copy	copy of a *specific* revision of a project (usually **HEAD**, e.g. the latest revision)
checkout	creation of a working copy
commit	update of local changes to the repository
conflict	when local changes are in conflict with the current version of the file stored on the repository

updating

To update the working copy files to the latest revisions in the repository:

```
$ svn update
```

All files in the current directory are updated. To update a single file, simply use:

```
$ svn update [file]
```

This command can also fetch a revision different than the latest revision with the -r flag:

```
$ svn update -r n [file]
```

where n is the desired revision number.

status

File statuses from inside a working copy is shown with:

```
$ svn status
```

This command gives information files not yet updated, in conflict or unknown to the svn system.
The output is a list of files with a status code indicating the status of the file:

status codes:

?	file is not under version control
A	file is scheduled for addition
D	file is scheduled for deletion
M	the content in bar.c has local modifications
C	file has textual conflicts from an update

committing changes

After editing files, changes are committed from the working copy to the central repository using the **commit** subcommand:

```
$ svn commit [-m "a commit message"]
```

Without the `-m` option, Subversion starts an editor to ask for a commit message.

Writing meaningful commit messages is useful when you want to know what was actually changed

You can commit more files at once.

adding/removing files

New files are added or removed with:

```
$ svn add FILE_OR_DIRECTORY
$ svn delete FILE_OR_DIRECTORY
```

Note that files are not actually added or deleted to the repository until committing.

```
$ svn add test1 test2
A         test1
A         test2
$ svn ci -m 'two empty files added'
Adding         test1
Adding         test2
Transmitting file data ..
Committed revision 2.
```

copyng files (branching)

To copy a file you use the **copy** subcommand.

In subversion, branching and tagging are implicit: you *copy* the *trunk* tree into a new tree in `/branches` or `/tags`

```
$ svn copy trunk/ branches/myfirstbranch
A         branches/myfirstbranch
$ svn ci branches/myfirstbranch/ -m '* branch created'
Adding         branches/myfirstbranch

Committed revision 6.
```

If you want to *switch* from a branch to another or to the *trunk* tree you can use the **switch** subcommand

Viewing Commit Messages

Commit messages are available for all revisions with:

```
$ svn log [file]
```

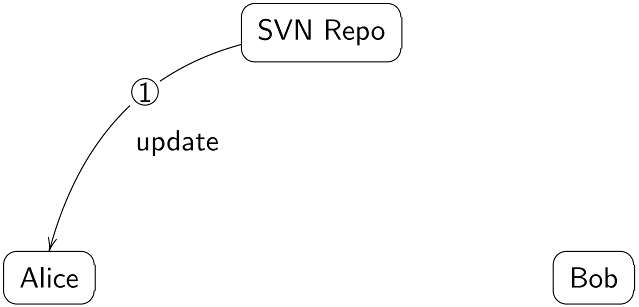
```
------------------------------------------------------------------------
r13 | alice | 2009-09-30 11:18:02 +0200(Mer, 30 Set 2009) | 1 line

fix typo
------------------------------------------------------------------------
r9 | bob | 2009-09-17 18:56:53 +0200(Gio, 17 Set 2009) | 1 line

added a check in do_some_stuff() function
```

Messages are printed in chronological order along with the associated revision number and author.

Daily workflow

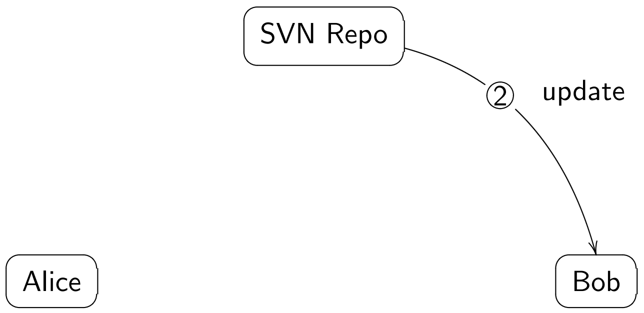
- Update your working copy
  - svn update
- Make changes
  - svn add
  - svn delete
  - svn copy
  - svn move
- Examine your changes
  - svn status
  - svn diff
- Possibly undo some changes
  - svn revert
- Resolve Conflicts (Merge Others' Changes)
  - svn update
  - svn resolved
- Commit your changes
  - svn commit

conflict management



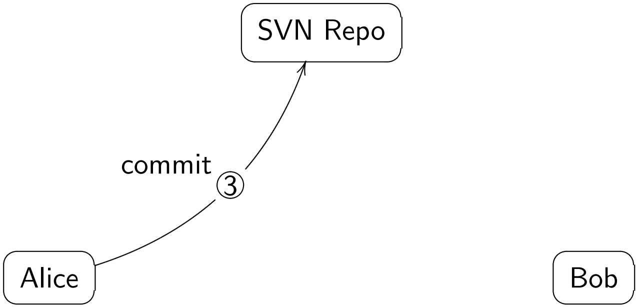
- Alice updates from repository

conflict management



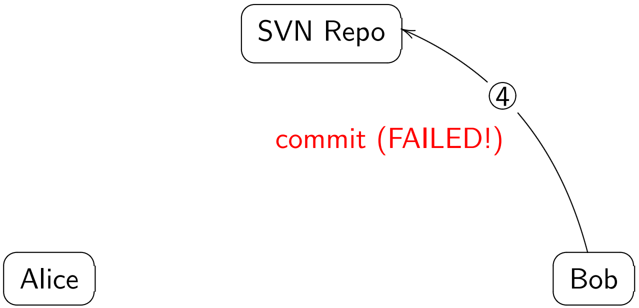
- Alice updates from repository
- Bob update from repository

conflict management



- Alice updates from repository
- Bob update from repository
- Alice commits her changes

conflict management



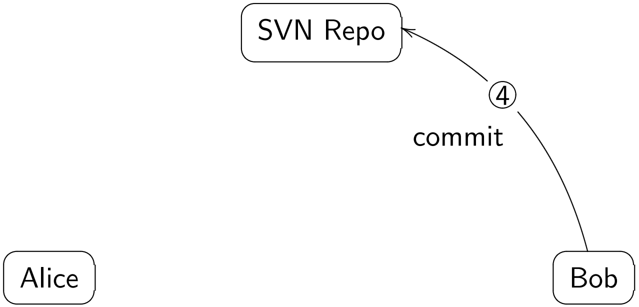
- Alice updates from repository
- Bob update from repository
- Alice commits her changes
- Bob's commit fails because of a conflict

conflict management

SVN Repo
fixing
Alice
Bob

- Alice updates from repository
- Bob update from repository
- Alice commits her changes
- Bob's commit fails because of a conflict
- Bob fixes the conflicts (maybe talking to Alice)

conflict management



- Alice updates from repository
- Bob update from repository
- Alice commits her changes
- Bob's commit fails because of a conflict
- Bob fixes the conflicts (maybe talking to Alice)
- Bob commit his changes

getting help

- `svn help command`
- http://svnbook.red-bean.com/
- http://subversion.tigris.org/
- http://www.google.com

Repository layout

```
$ svn list https://svn.gforge.escience-lab.org/svn/hpc-2008/
branches/
tags/
trunk/
```

trunk (current)	holds the *main line* (like HEAD in CVS)
branches	contains a directory for each branch
tags	contains a directory for each tag

SVN vs. CVS

- versioning is on a per-repository base instead of file. This means:
  - you can have transactions (i.e. you can commit more file at once)
  - revisions represent different *states* of the projects, not just of the file
- you can track directories too (and copied/renamed files)
- more offline operations (status, diff, revert)
- `svn status` is human readable :)
- tags and branches are treated as ordinary directories
- you can attach arbitrary metadata to files and directories
- you cannot commit a file if there is an unresolved conflict
- better support for binary files