An Introduction to Subversion

Flavio Stanchi

May 21, 2015

Table of Contents

Features of Subversion

- It's a version control system
- Uses a centralized model:
  - Server-client approach
  - Version merging
  - With wireless connections everywhere, it's rarely a limitation
- Easy to learn (but slower than Git)
- It's free

Getting Subversion

- Subversion can be found at `https://subversion.apache.org`

Getting Subversion

- Subversion can be found at `https://subversion.apache.org`
- Version 1.8 is the last release at this time

Getting Subversion

- Subversion can be found at `https://subversion.apache.org`
- Version 1.8 is the last release at this time
- Client for Windows:
  - TortoiseSVN (free): `http://tortoisesvn.net/`

Getting Subversion

- Subversion can be found at `https://subversion.apache.org`
- Version 1.8 is the last release at this time
- Client for Windows:
  - TortoiseSVN (free): `http://tortoisesvn.net/`
- Clients for Mac:
  - Xcode (free): `https://developer.apple.com/xcode/downloads/`
  - Versions ($): `http://versionsapp.com/`

Getting Subversion

- Subversion can be found at `https://subversion.apache.org`
- Version 1.8 is the last release at this time
- Client for Windows:
  - TortoiseSVN (free): `http://tortoisesvn.net/`
- Clients for Mac:
  - Xcode (free): `https://developer.apple.com/xcode/downloads/`
  - Versions ($): `http://versionsapp.com/`
- If you are using Linux ...

Getting Subversion

- Subversion can be found at `https://subversion.apache.org`
- Version 1.8 is the last release at this time
- Client for Windows:
  - TortoiseSVN (free): `http://tortoisesvn.net/`
- Clients for Mac:
  - Xcode (free): `https://developer.apple.com/xcode/downloads/`
  - Versions ($): `http://versionsapp.com/`
- If you are using Linux . . . use the terminal!

Subversion on Cornell servers

- CISER on RSCH101 and RSCH106 (tested)

Subversion on Cornell servers

- CISER on RSCH101 and RSCH106 (tested)
- ECCO

Subversion on Cornell servers

- CISER on RSCH101 and RSCH106 (tested)
- ECCO
- Quick reference guide at `http://www2.vrdc.cornell.edu/news/documentation/subversion/`

Table of Contents

Centralized version control

- Server-client approach
  - The repository is located in the server

Centralized version control

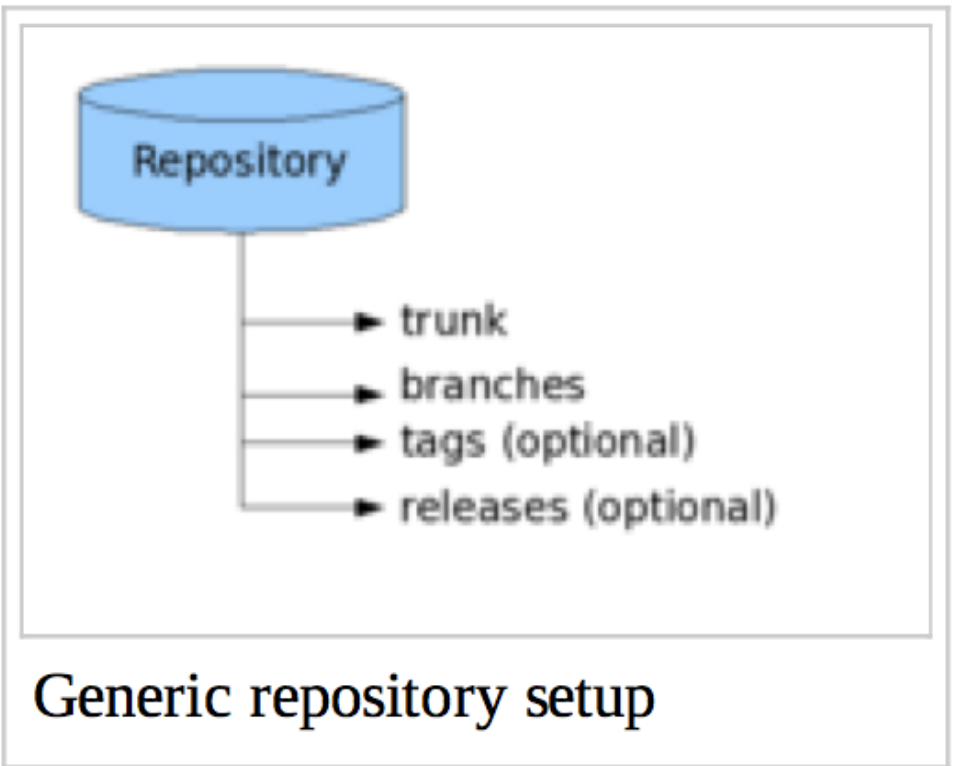
- Server-client approach
  - The repository is located in the server
  - No version control over local copies

Centralized version control

- Server-client approach
  - The repository is located in the server
  - No version control over local copies
- Version merging:
  - Multiple editors can check out any given file
  - Discrepancies are handled upon checkin

Generic setup

- Trunk: contains all the clean code
- Branches: where all initial work occurs
- Tags and releases (optional)



Generic repository setup

Local copy

- The repository may be remote or local . . .

Local copy

- The repository may be remote or local . . . but you don't usually work directly with it

Local copy

- The repository may be remote or local . . . but you don't usually work directly with it
- Instead, check out a local copy of the repository (or of its subelements)

Local copy

- The repository may be remote or local . . . but you don't usually work directly with it
- Instead, check out a local copy of the repository (or of its subelements)
- Make changes to the local copy
  - Important: use Subversion commands to do this, so that every change is registered

Local copy

- The repository may be remote or local . . . but you don't usually work directly with it
- Instead, check out a local copy of the repository (or of its subelements)
- Make changes to the local copy
  - Important: use Subversion commands to do this, so that every change is registered
- Commit the changes back into the repository
  - Add a commit (log) message
  - Every commit is registered with a revision number

Local copy

- Note: direct changes to the repository are immediately applied

. . .

Local copy

- Note: direct changes to the repository are immediately applied . . . while changes to the local copy are applied to the repository upon commit

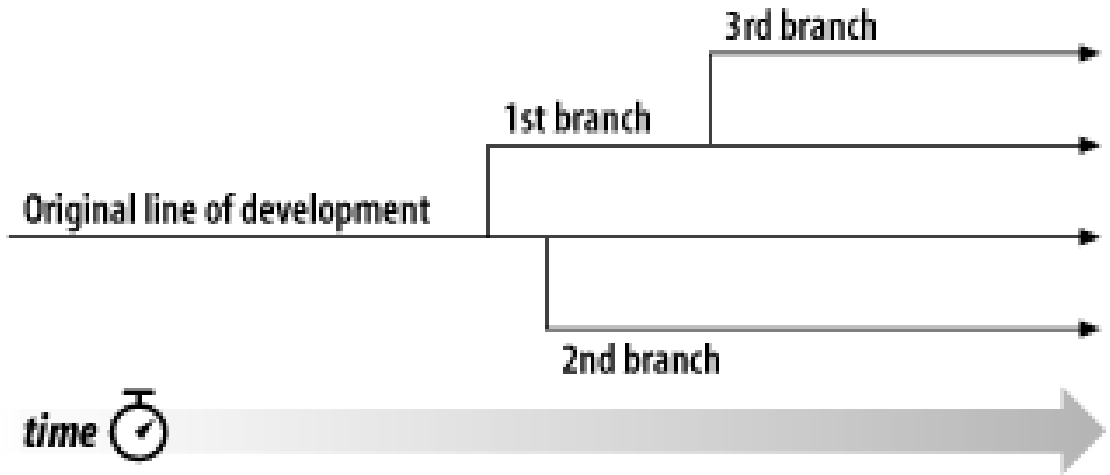
Local copy

- Note: direct changes to the repository are immediately applied . . . while changes to the local copy are applied to the repository upon commit
- Hence, commit frequently!

Branches

A branch is a line of development that exists independently of another line, yet still shares a common history if you look far enough back in time.

It begins life as a copy of something, and moves on from there, generating its own history.



Creating a branch

Make a copy of (a part of) your project tree in the repository using the svn copy command.

The copy may live wherever you wish. Usually, in a folder named **branches**.

Note: you can do a remote copy — a copy that immediately results in a newly committed repository revision — no working copy is required! Just copy one URL to another.

Cheap copies: when you copy a directory, you don't need to worry about duplicating the size, since SVN doesn't actually duplicate any data. Instead, it creates a new directory entry that points to an existing tree.

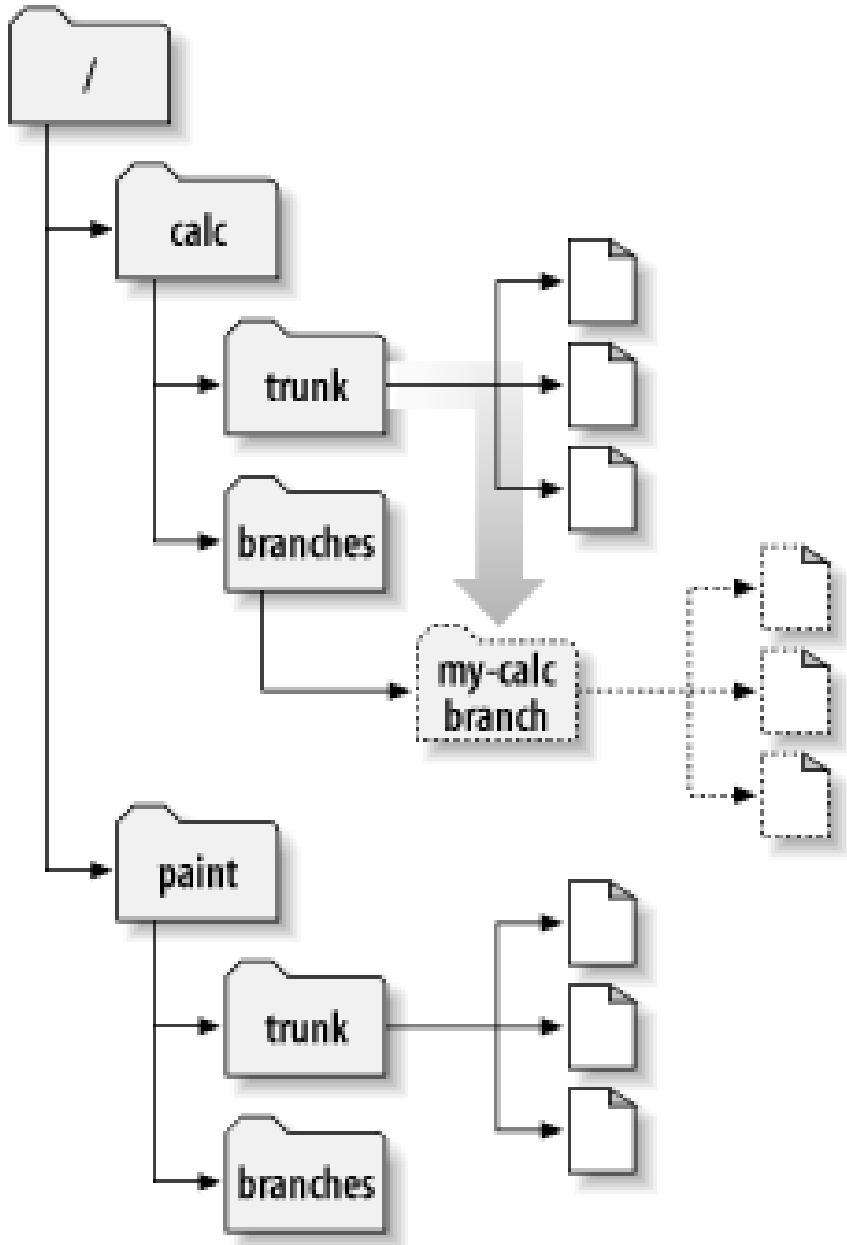

/
calc
trunk
branches
my-calc
branch
paint
trunk
branches

Table of Contents

The terminal

- On Linux and on OSX, use the terminal

The terminal

- On Linux and on OSX, use the terminal
- Advantages:
  - Flexibility
  - It's not so complicated

The terminal

- On Linux and on OSX, use the terminal
- Advantages:
  - Flexibility
  - It's not so complicated
- Every command must be preceded by *svn*

```
server> svn co repository:trunk /programs/production/prod/current
```

Your best friend

- The most important command is ...

Your best friend

- The most important command is . . . *help*

Your best friend

- The most important command is . . . *help*
- Calling *help* alone will print a summary of the commands and their usage

Your best friend

- The most important command is . . . *help*
- Calling *help* alone will print a summary of the commands and their usage
- Calling *help* followed by the name of a command will print a short description of the command and its options

Your best friend

- The most important command is . . . *help*
- Calling *help* alone will print a summary of the commands and their usage
- Calling *help* followed by the name of a command will print a short description of the command and its options
- Options are often useful (and sometimes necessary), but it's hard to remember them all: use *help*!

Basic workflow

1. Update your working copy or check out a new one
   - Commands: *co* (*check out*), *update*

Basic workflow

1. Update your working copy or check out a new one
   - Commands: *co* (*check out*), *update*
2. Make changes
   - Use your favorite editors
   - Commands: *add*, *delete*, *copy*, *move*

Basic workflow

1. Update your working copy or check out a new one
   - Commands: *co* (*check out*), *update*
2. Make changes
   - Use your favorite editors
   - Commands: *add*, *delete*, *copy*, *move*
3. Review the changes
   - Commands: *status*, *diff*, *log*

Basic workflow

1. Update your working copy or check out a new one
   - Commands: *co* (*check out*), *update*
2. Make changes
   - Use your favorite editors
   - Commands: *add*, *delete*, *copy*, *move*
3. Review the changes
   - Commands: *status*, *diff*, *log*
4. Fix any mistake
   - Command: *revert*

Basic workflow

1. Update your working copy or check out a new one
   - Commands: *co* (*check out*), *update*
2. Make changes
   - Use your favorite editors
   - Commands: *add*, *delete*, *copy*, *move*
3. Review the changes
   - Commands: *status*, *diff*, *log*
4. Fix any mistake
   - Command: *revert*
5. Resolve conflicts
   - Command: *resolve*

Basic workflow

1. Update your working copy or check out a new one
   - Commands: *co* (*check out*), *update*
2. Make changes
   - Use your favorite editors
   - Commands: *add*, *delete*, *copy*, *move*
3. Review the changes
   - Commands: *status*, *diff*, *log*
4. Fix any mistake
   - Command: *revert*
5. Resolve conflicts
   - Command: *resolve*
6. Publish changes
   - Command: *ci* (commit)