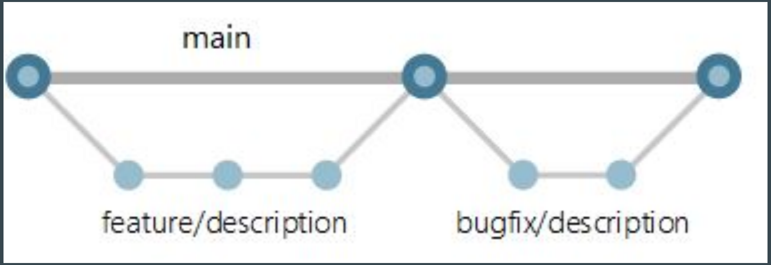
From SVN to git ...

Christian Rab

USM Code Coffee- 15.11.2022

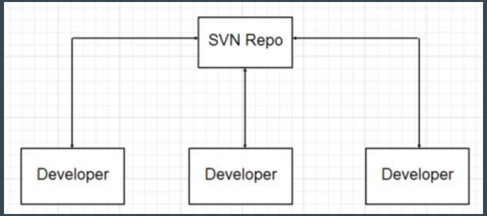
Version Control Systems - Source Code Management

- Simply keep track of changes to files and attach a version to each change
- Can get complex with many files/directories and especially if one is sharing data with others
- Version control systems (CVS, SVN, git) do the job for you
- Repository keeps track of changes; commit changes; update from repository (i.e. synchronize changes with other users)
- Provide also tools for branching/merging, release management, (Source Code Management, SCM)
- Version Control 101 (5 min read)



Subversion (svn)

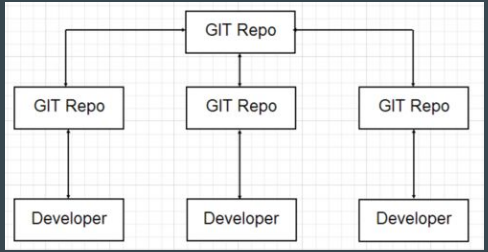
- **Centralized** (client - server), client does not have all the information/data, commits/changes to the repo are immediate
- Supposed to be better with handling large binary files
- Working with branches is not very convenient
- ….



Git

- **Decentralized**, client (local) workstation has all the information/data - need to synchronize different repositories.
- Very good and fast with branching
- Supposed to be bad with large files (but https://git-lfs.github.com/)
- …

https://hackbrightacademy.com/blog/svn-vs-git/



- Most differences for working with git or svn are a consequence of the underlying concept of decentralized vs. centralized(nice summary here)
- If you are new to version control go for git … mainly because the svn community is dying ….
- In case you don't know it, here is the opinion from Linus Torvalds (the creator of git) on svn/cvs (WARNING: contains some strong language!)

Migration from SVN to git ... example ProDiMo

- **13 year old Fortran code was managed by svn**/trac until last year; but precursor of the code even older
- 3 to 4 main developers, but also (Phd) students should do code development
- not open source, but everyone can have it; mostly used by (PhD) students of the main developers ... about 10 to 20 active users at a time
- **No full time code maintainer; user support/management**
- With svn it happened that **developments became "disconnected"** ... merging them back into the main code was very painful
- code was hosted at an university ... but none of the developers was affiliated with this university anymore
- -> decided to move to git on a gitlab server managed at university of a developer with a permanent position

How to migrate

- Git provides a tool for migration, one can even “connect” git and svn repositories (useful scenario: https://www.atlassian.com/git/tutorials/migrating-convert)
- Decided to do a sharp switch. Stop any development, migrate into git repo, deactivate the old one (just read access), continue on git (inform the users etc.)
- Migrating the code itself turned out to be relatively easy and fast (likely depends also on the code an your repository).
- Also other things to consider
  - How to migrate registered users
  - Used wiki pages for documentation (part of trac) - need to migrate that to github/gitlab
  - ...

Technical migration I

- *developers.txt* file (important mapping of svn author to git author)
  Can be partly generated, see here; users that committed something
- Clone the svn repo into a git repository

```
git svn clone --trunk=/trunk/version0.7
--authors-file=developers.txt https://forge.roe.ac.uk/svn/ProDiMo
prodimo
```

- Put it on your remote git repository (can be github, a gitlab server etc.)

```
cd REPODIRECTORY
# might not be required
git remote rename origin old-origin
git remote add origin
https://gitlab.astro.rug.nl/prodimo/prodimo.git
git push -u origin --all
git push -u origin --tags
```

Technical Migration II

- .svnignore (files excluded from version control)

```
cd REPODIRECTORY
git svn show-ignore > .gitignore
git add .gitignore
git commit -m 'Convert svn:ignore properties to .gitignore.'
```

- The above procedure puts the svn revision number into the git commit message -> connection between svn revision number and git hash. Can be deactivated, but is recommended.
- Done almost ....

Technical Migration III

- Adapted a **script from the web to migrate the wiki to markup**. Ask me if you interested. Likely not general. This can be an annoying issue!
- If you use the **svn revision number or version tags in your code** (access them, in a makefile etc.) ... you need to change the for git.
- Adapt scripts from **automatic test runs** (if there are any) ....
- **Learn to use git and agree on some “standard” ways to do things** (e.g. work with branches etc.) ...
- provide some guidelines for developers ... but well not everybody is reading them

How to keep your git repository in order ...

- everyone (developer) has it own **local copy(ies) of the git repo -> local git configurations** (e.g. commit user, default for pull or push etc.). No option to have the config centralized
- **Agree on a way to do things and stick to it** ... e.g. all developer should have same config... but again now way to force that ...
- **new developments (features) in branches** ... when finished merge into master (protected branch) ... inexperienced developers work in branches but cannot merge into the master ... use merge request
- Branch management in git is great ... but can be challenging in the beginning, especially if people are only used to subversion

Some examples ...

- Merging vs. rebasing:
  - https://www.atlassian.com/git/tutorials/merging-vs-rebasing
  - https://www.edureka.co/blog/git-rebase-vs-merge/