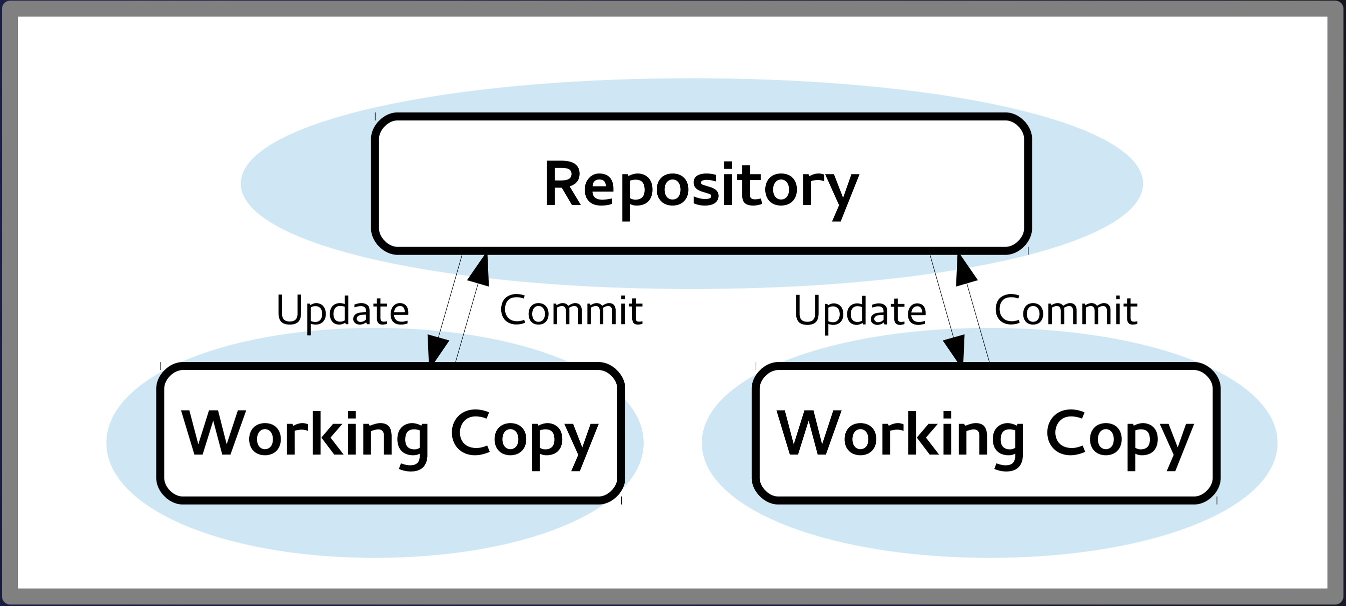
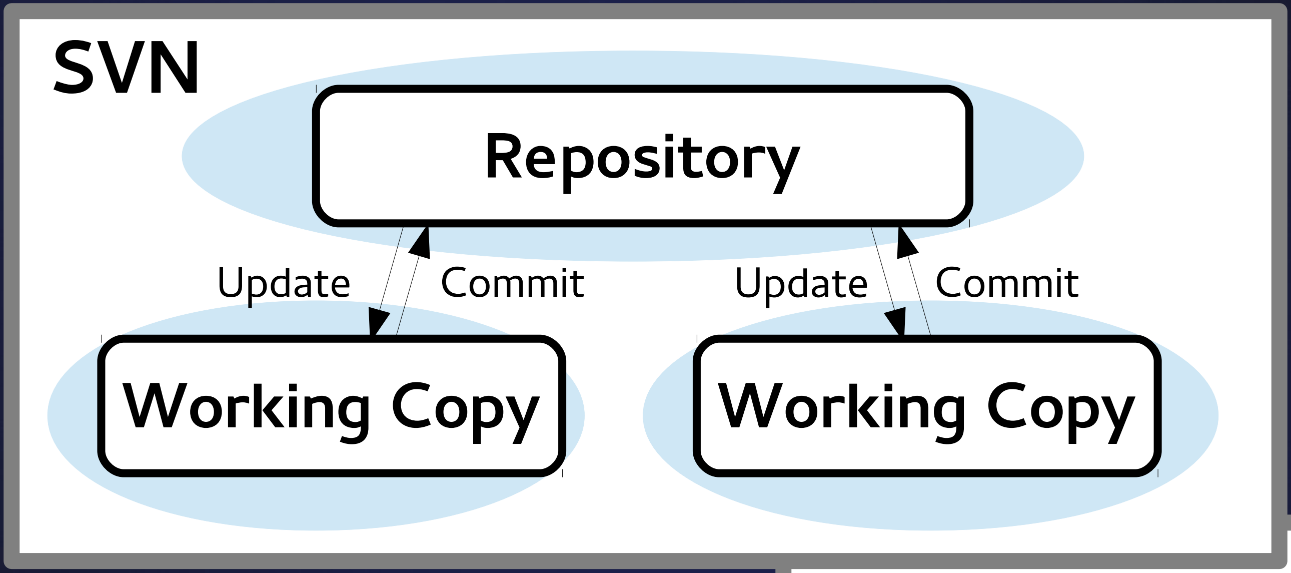
Git vs SVN

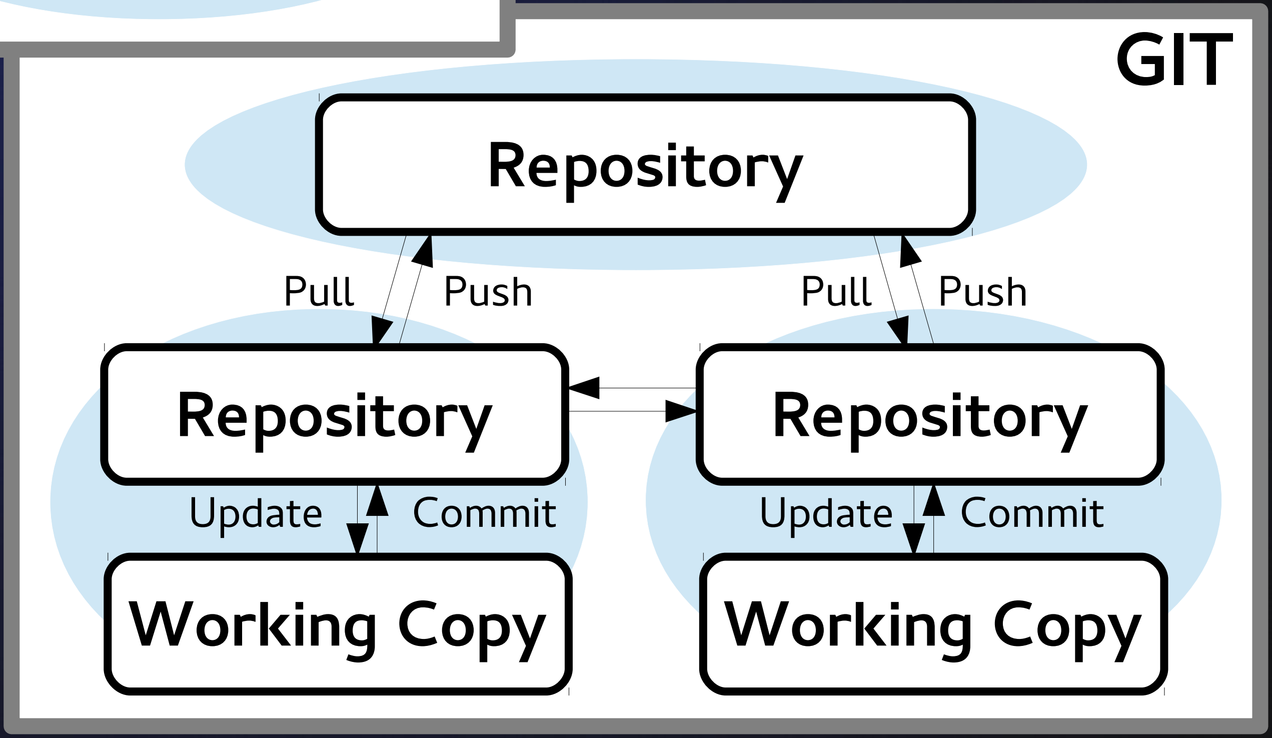
What is SVN?



- Centralised Version Control
- One big remote repository
- Checkout a branch from this central repository
- Commit connects to remote and sends changes
- Improved on CVS, similar concepts

Comparing Git to Svn

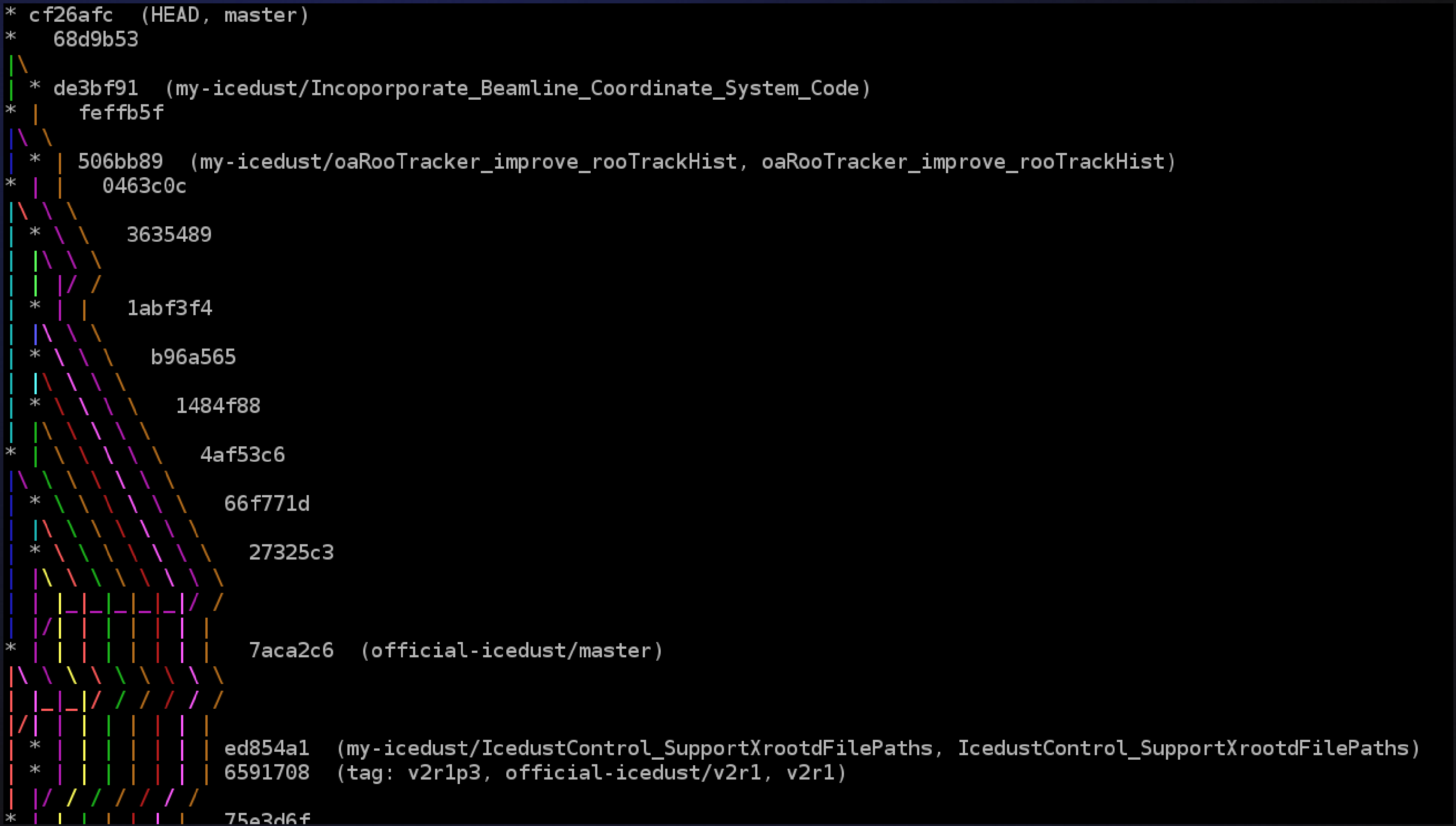




- Distributed Version Control
- "Clone" complete copies of the entire repository
- "Commit" stores local snapshot of working index
- Push and pull to any other "remote" git repository

Branches and Tags in SVN

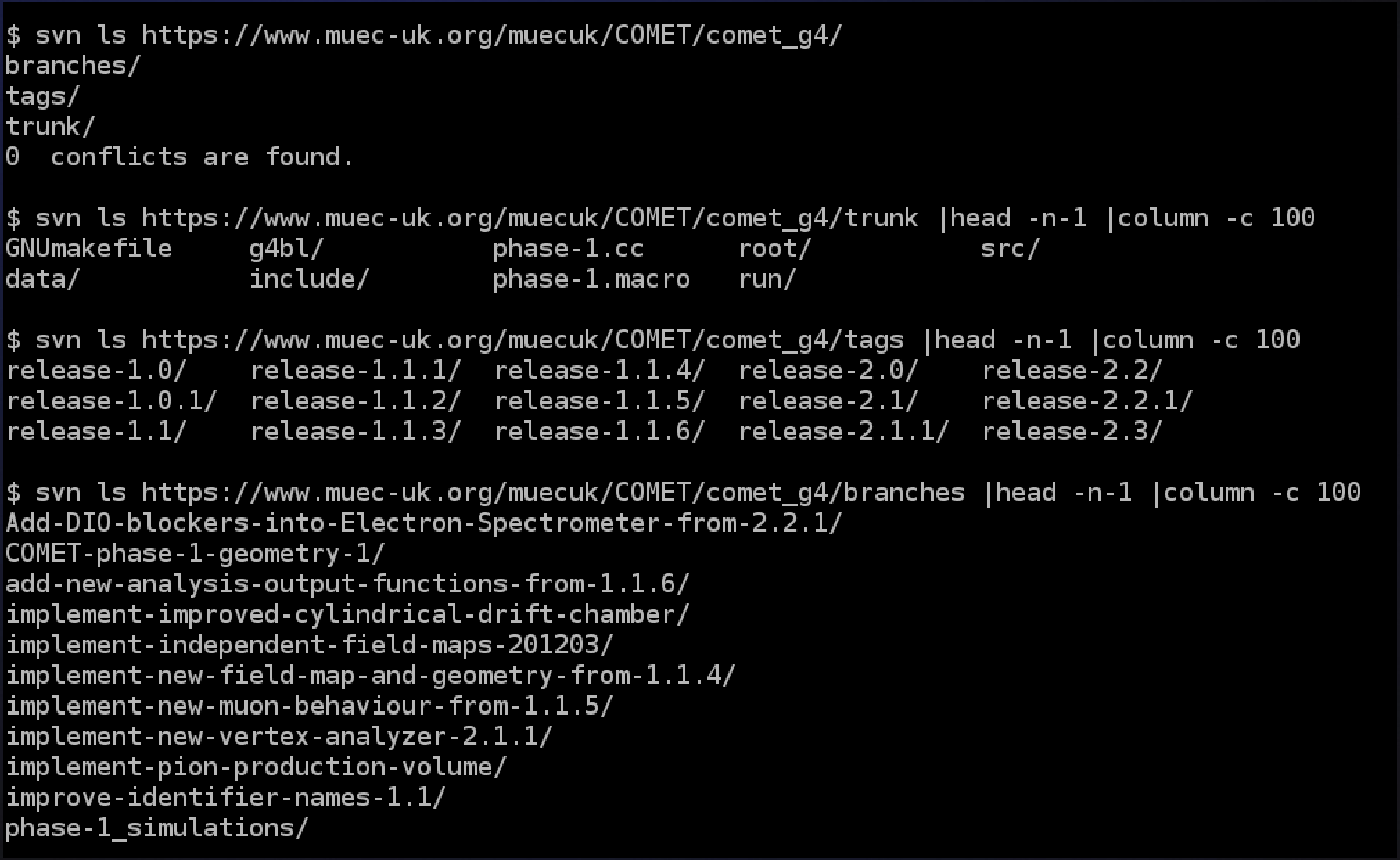
- In Git: tags and branches are just 'pointers' to a commit
- They have dedicated commands



```
* cf26afc  (HEAD, master)
*   68d9b53
|\
| * de3bf91  (my-icedust/Incoporporate_Beamline_Coordinate_System_Code)
* |   feffb5f
|\ \
| * | 506bb89  (my-icedust/oaRooTracker_improve_rooTrackHist, oaRooTracker_improve_rooTrackHist)
* | |   0463c0c
|\ \ \
| * \ \   3635489
| |\ \ \
| | |/ /
| * | |   1abf3f4
| |\ \ \
| * \ \ \   b96a565
| |\ \ \ \
| * \ \ \ \   1484f88
| |\ \ \ \ \
* | \ \ \ \ \   4af53c6
|\ \ \ \ \ \ \
| * \ \ \ \ \ \   66f771d
| |\ \ \ \ \ \ \
| * \ \ \ \ \ \ \   27325c3
| |\ \ \ \ \ \ \ \
| | |_|_|_|_|_|/ /
| |/| | | | | | |
* | | | | | | | |   7aca2c6  (official-icedust/master)
|\ \ \ \ \ \ \ \ \
| |_|_|/ / / / / /
|/| | | | | | | |
| * | | | | | | | ed854a1  (my-icedust/IcedustControl_SupportXrootdFilePaths, IcedustControl_SupportXrootdFilePaths)
| * | | | | | | | 6591708  (tag: v2r1p3, official-icedust/v2r1, v2r1)
| |/ / / / / / /
* | | | | | | |   75e3d6f
```

Branches and Tags in SVN

- In SVN: a branch or tag is just a copy made into a new sub-directory of the repository



```
$ svn ls https://www.muec-uk.org/muecuk/COMET/comet_g4/
branches/
tags/
trunk/
0  conflicts are found.

$ svn ls https://www.muec-uk.org/muecuk/COMET/comet_g4/trunk |head -n-1 |column -c 100
GNUmakefile     g4bl/           phase-1.cc      root/           src/
data/           include/        phase-1.macro   run/

$ svn ls https://www.muec-uk.org/muecuk/COMET/comet_g4/tags |head -n-1 |column -c 100
release-1.0/    release-1.1.1/  release-1.1.4/  release-2.0/    release-2.2/
release-1.0.1/  release-1.1.2/  release-1.1.5/  release-2.1/    release-2.2.1/
release-1.1/    release-1.1.3/  release-1.1.6/  release-2.1.1/  release-2.3/

$ svn ls https://www.muec-uk.org/muecuk/COMET/comet_g4/branches |head -n-1 |column -c 100
Add-DIO-blockers-into-Electron-Spectrometer-from-2.2.1/
COMET-phase-1-geometry-1/
add-new-analysis-output-functions-from-1.1.6/
implement-improved-cylindrical-drift-chamber/
implement-independent-field-maps-201203/
implement-new-field-map-and-geometry-from-1.1.4/
implement-new-muon-behaviour-from-1.1.5/
implement-new-vertex-analyzer-2.1.1/
implement-pion-production-volume/
improve-identifier-names-1.1/
phase-1_simulations/
```

Checking out a Repo

- Svn checkout:
  - Makes a **local copy of the tree** in a repository and matches each file to a remote one
  - Can **checkout a sub-directory** of a repository
  - Every directory has a ``.svn/`` directory
- Git clone:
  - Makes a **local copy of the repository** and makes your working index match the head of the master branch
  - Can only **check-out an entire repository** *(sort of)*
  - Top-level directory will contain a ``.git/`` directory

Commands are changed

SVN	GIT
checkout *repository*	clone *repository*
checkout *sub-directory*	*Sparse clones but not so simple*
commit	commit + push
revert *filename*	checkout *filename*
switch *branch*	checkout *branch*
update	pull
export	archive
add *filename*	add *filename*
Log / status / diff / blame	Log / status / diff / blame

Resetting the Working Copy

- Having made some changes, we want to roll them back
- In SVN:

```
$ svn revert -R directory/
Reverted 'directory/file1'
Reverted 'directory/file2'

$ svn revert filename
Reverted 'filename'
```

- In Git, it depends whether we have changed:
  - Working index:

```
$ git checkout filename
$ git checkout directory/
```

  - Staging area (after `` `git add` ``):

```
$ git reset filename
Unstaged changes after reset:
M       filename
```

Tagging a Release

- Repository IDs
  - SVN revision numbers: `r1401`
  - Git commit hashes `ff9e41983dd160cdc20d048a4153fa49c37a1b8f`
- Specific tags emphasize a release:
  - In SVN: Copy the trunk into the tags directory

```
$ svn copy http://svn.example.com/repos/calc/trunk \
           http://svn.example.com/repos/calc/tags/release-1.0 \
           -m "Tagging the 1.0 release of the 'calc' project."

Committed revision 902.
```

  - In Git: Use `git tag`

```
$ git tag release-1.0
$ git tag -a release-1.1 -m "This is a new release"
```

Merge Resolution

File conflicts:

- User A and B edit same file in the same place
  - Svn and git need to manually merge files
- Working with the merge interactively:
  - Svn gives you options immediately
  - Git will return control to you immediately
    - Use `git mergetool` which will give a more interactive (even GUI, if configured) tool

Merge Resolution

File conflicts:

- Finishing merges

```
$ svn update
Conflict discovered in 'file1'.
Select: (p) postpone, (df) diff-full, (e) edit,
        (mc) mine-conflict, (tc) theirs-conflict,
        (s) show all options: p

$ vi file1 # or emacs, sublime etc
.....
<<<<<<< .mine
changes by user1
=======
changes by user2
>>>>>>> .r2
.....
# Select desired hunk

$ svn resolve --accept working file1
$ svn commit -m "Fixed conflict"
```

```
$ git pull
Auto-merging file1
CONFLICT (content): Merge conflict in file1
Automatic merge failed; fix conflicts and then comm
it the result.

$ vi file1 # or emacs, sublime etc
.....
<<<<<<< HEAD
changes by user1
=======
changes by user2
>>>>>>> branch1
.....
# Select desired hunk

$ git add file1
$ git commit -m "Fixed conflict"
```

- Switch file versions:

```
$ git checkout --theirs filename
$ git checkout --ours filename
```

- Abort merge:

```
$ git merge --abort
```

Merge Resolution

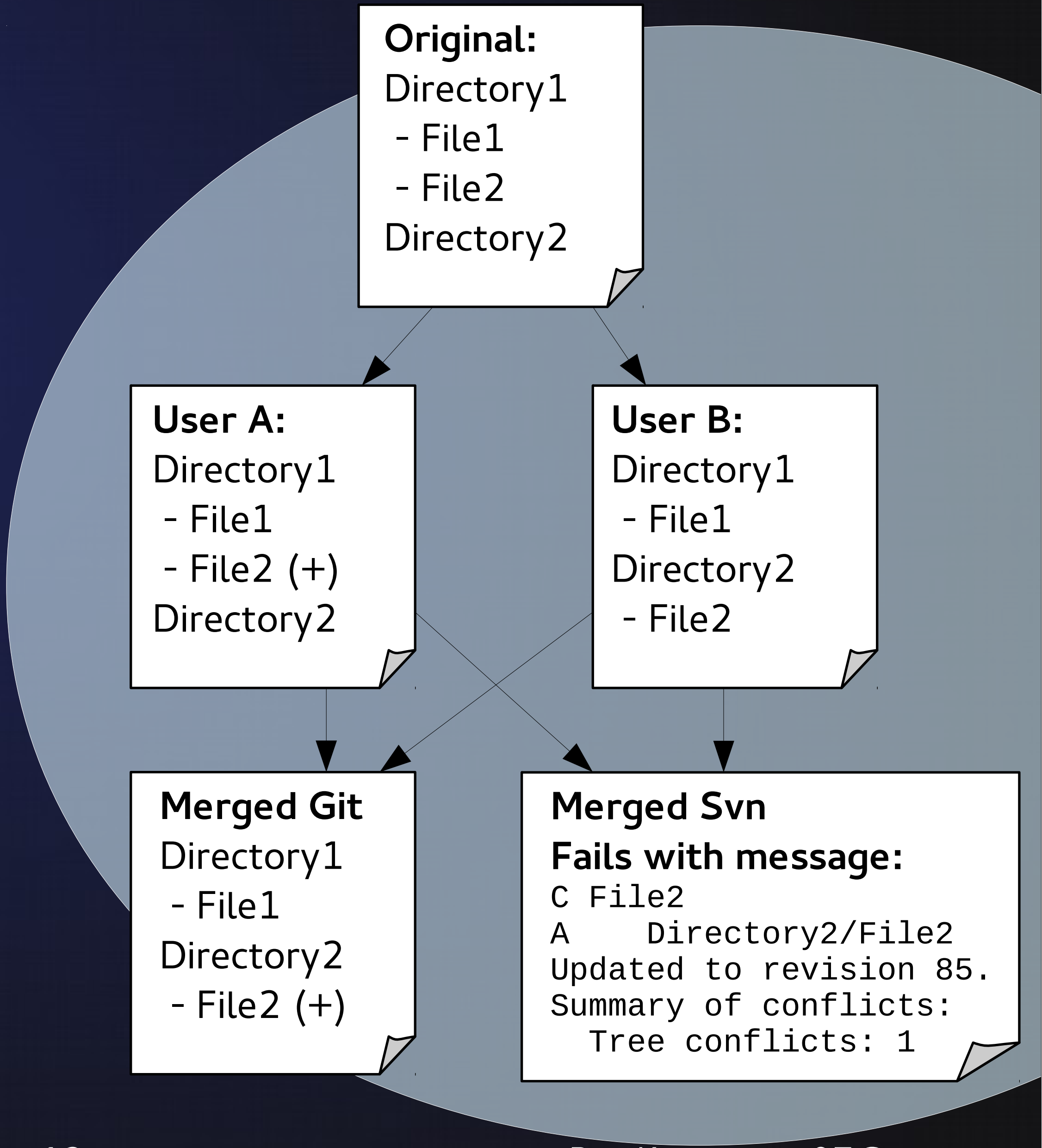
Merging Gotchas

- --theirs is the incoming file
- --ours is the current file
  - So when Merging, 'theirs' is the branch being merged in, 'ours' is the branch being merged into.
  - When rebasing, 'ours' is the commits being rebased onto (typically the remote, the other branch), 'theirs' is the branch being rebased (the branch being worked on).
- Use `git log --merge -p filename` to look at changes to a file that contribute to a conflict
- `Git merge branch2` will merge branch2 into your current branch

Merge Resolution

Tree Conflicts

- User A renames or moves a file (even to a sub-dir)
- User B changes its content
- Git can resolve automatically
- Svn will flag as a conflict
  - Need to solve manually



```
C File2
A     Directory2/File2
Updated to revision 85.
Summary of conflicts:
  Tree conflicts: 1
```