Revision Control and GIT

On UD HPC Community Clusters

William Totten
Network & Systems Services

Why use revision control

- You can go back in time
- It makes it easy to try things out which might not work
- Facilitates multiple people making changes at once
- Helps for distribution
- Great for programming source code
- Also great for configuration files
- Helps with peer review

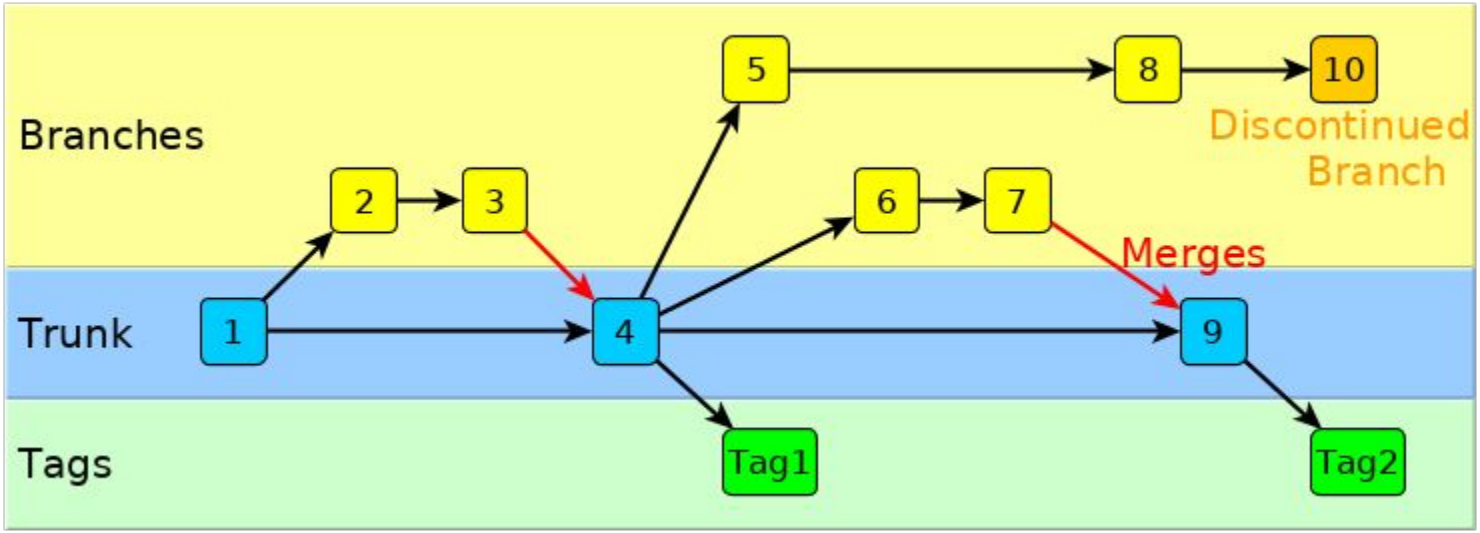
Revision control systems

- Git
  - Local repositories
  - Consider https://github.com/ (free for open source) or https://gitlab.com/ (free for all)
  - Allows for multiple local commits before uploading them centrally
- Subversion (svn)
  - Local repositories
  - Consider https://sourceforge.net/
  - You can check out any part of the tree you want
- Others
  - RCS
  - CVS
  - Mercurial (aka hg)
  - ...

Revision control terminology

branch	A set of files under version control may be *branched* or *forked* at a point in time so that, from that time forward, two copies of those files may develop at different speeds or in different ways independently of each other.
checkout	To *clone*, *check out* (or *co*) is to create a local working copy from the repository. A user may specify a specific revision or obtain the latest. The term 'checkout' can also be used as a noun to describe the working copy.
commit	To *commit* (*check in*, *ci* or, more rarely, *install*, *submit* or *record*) is to write or merge the changes made in the working copy back to the repository. The terms 'commit' and 'checkin' can also be used as nouns to describe the new revision that is created as a result of committing.
conflict	A conflict occurs when different parties make changes to the same document, and the system is unable to reconcile the changes. A user must *resolve* the conflict by combining the changes, or by selecting one change in favour of the other.
merge	A *merge* or *integration* is an operation in which two sets of changes are applied to a file or set of files.
pull/push	Copy revisions from one repository into another. Pull is initiated by the receiving repository, while push is initiated by the source. Fetch is sometimes used as a synonym for pull, or to mean a pull followed by an update.
revision	Also version: A version is any change in form.
tag	A *tag* or *label* refers to an important snapshot in time, consistent across many files. These files at that point may all be tagged with a user-friendly, meaningful name or revision number.
trunk	The unique line of development that is not a branch (sometimes also called Baseline, Mainline or Master)

Revision control in action



- In SVN, revisions are numbers like shown above and tags can be any text string
- If this were Git, the revisions would have seemingly random hexadecimal strings like f7fd3d4

Local SVN

- In SVN, the repository should go someplace special
- You may want to keep an "official" checkout in a workgroup directory
- You can start out with an empty repository, then add stuff
- Try to make the names match up between the repository & checkout
- You have to manually add the files you want in the repository
- Once you check-in the files, anyone with access to the repository can create their own working copy

```
$ mkdir repos
$ svnadmin create repos/project1
$ svn mkdir -m Structure file:///home/work/it_nss/repo/svn_exampl
e/repos/project1/{trunk,branches,tags}
Committed revision 1.
$ svn co file:///home/work/it_nss/repo/svn_example/repos/project1
/trunk project1
Checked out revision 1.
$ cd project1
$ cp /opt/templates/dev-projects/C_Executable/* .
$ svn add *
A         Makefile
A         helloworld.c
A         printmsg.c
A         printmsg.h
$ svn ci -m 'Initial code check-in'
Adding         Makefile
...
Transmitting file data ....
Committed revision 2.
```

Updating code in an SVN repository

- Always try to ensure your copy is up-to-date before making changes
- Use whatever process you are most comfortable with to change files
- Always use `svn add`, `svn rm`, and `svn mv` to add/remove/move files
- You should always provide a comment when making changes
- SVN is centralized, so version numbers are monotonically increasing
- SVN can tell you if any new files aren't revision controlled

```
$ svn up
At revision 2.
$ vi Makefile
$ svn ci -m 'Simplify Makefile, switch to gcc'
Sending        Makefile
Transmitting file data .
Committed revision 3.
$ make
gcc -g -O   -c -o helloworld.o helloworld.c
gcc -g -O   -c -o printmsg.o printmsg.c
gcc -g -O -o helloworld helloworld.o printmsg.o  -lm
$ svn status
?       helloworld
$
```

Initializing Git

- When you share your changes, they need to be linked to you
- You need to tell Git something unique about you to tie to your changes
- You can tell Git your email address, this is standard practice
- You can also tell Git your name
- You can set your text editor
- The easiest way to distribute your code is SSH, create a key if you don't have one already

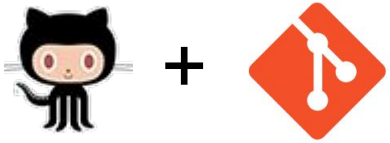
```
$ git config --global user.email totten@udel.edu
$ git config --global user.name "William Totten"
$ git config --global core.editor /usr/bin/vile
$ ls -la .ssh/id_rsa.pub
-rw-r--r-- 1 totten everyone 406 Aug 20  2014 .ssh/id_rsa.pub
```

Local Git

- Git is distributed, so every clone is a repository by definition
- You may want to keep an "official" clone in a workgroup directory
- You can start out with an empty repository, then add stuff
- You have to manually add the files you want in the repository
- Once you check-in the files, anyone else with access to the repository can clone their own copy

```
$ mkdir project1
$ cd project1
$ git init .
Initialized empty Git repository in
/home/work/it_nss/repo/git_example/project1/.git/
$ cp /opt/templates/dev-projects/C_Executable/* .
$ git add *
$ git commit -am 'Initial code check-in'
[master (root-commit) f7fd3d4] Initial code check-in
 4 files changed, 153 insertions(+), 0 deletions(-)
 create mode 100644 Makefile
 create mode 100644 helloworld.c
 create mode 100644 printmsg.c
 create mode 100644 printmsg.h
```

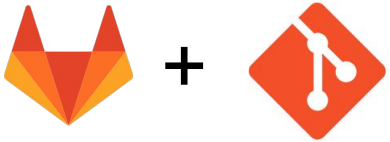
Using github.com



- Get a login for yourself on github.com, I recommend adding some SSH keys
- Create the new repository at https://github.com/new
- Now clone a working repository for yourself
- You have to manually add the files you want in the repository
- Once you check-in the files, anyone else with access to the repository can clone or pull their own copy
- The project must be Open Source on github, or they want money

```
$ git clone git@github.com:biell/project1.git
Initialized empty Git repository in
/home/work/it_nss/repo/git_example/project1/.git/
remote: Counting objects: 4, done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 4
Receiving objects: 100% (4/4), 850 B, done.
$ cd project1
$ vi README.md
$ cp /opt/templates/dev-projects/C_Executable/* .
$ git add *
$ git commit -am 'Initial code check-in'
[master (root-commit) 38ad2e3] Initial code check-in
 5 files changed, 153 insertions(+), 0 deletions(-)
 create mode 100644 Makefile
 create mode 100644 helloworld.c
 create mode 100644 printmsg.c
 create mode 100644 printmsg.h
```

Using gitlab.com



- Get a login for yourself on gitlab.com, I recommend adding some SSH keys
- Create the new repository at https://gitlab.com/projects/new
- Now clone a working repository for yourself
- You have to manually add the files you want in the repository
- Once you check-in the files, anyone else with access to the repository can clone or pull their own copy
- The project can be public or private on gitlab

```
$ git clone git@gitlab.com:biell/project1.git
Initialized empty Git repository in
/home/work/it_nss/repo/git_example/project1/.git/
Warning: You appear to have cloned an empty repository.
$ cd project1
$ vi README.md     # create this file b/c gitlab doesn't do it for you
$ cp /opt/templates/dev-projects/C_Executable/* .
$ git add *
$ git commit -am 'Initial code check-in'
[master (root-commit) 38ad2e3] Initial code check-in
 5 files changed, 153 insertions(+), 0 deletions(-)
 create mode 100644 Makefile
 create mode 100644 helloworld.c
 create mode 100644 printmsg.c
 create mode 100644 printmsg.h
```

Updating code in a Git repository

- Pull the latest trunk when using github
- Use whatever process you are most comfortable with to change files
- Always use `git add`, `git rm`, and `git mv` to add/remove/move files
- You can choose what to commit, add the "-a" to commit all changes.
- You commit to your local, distributed clone, not a central repository
- You can push any number of commits all at once

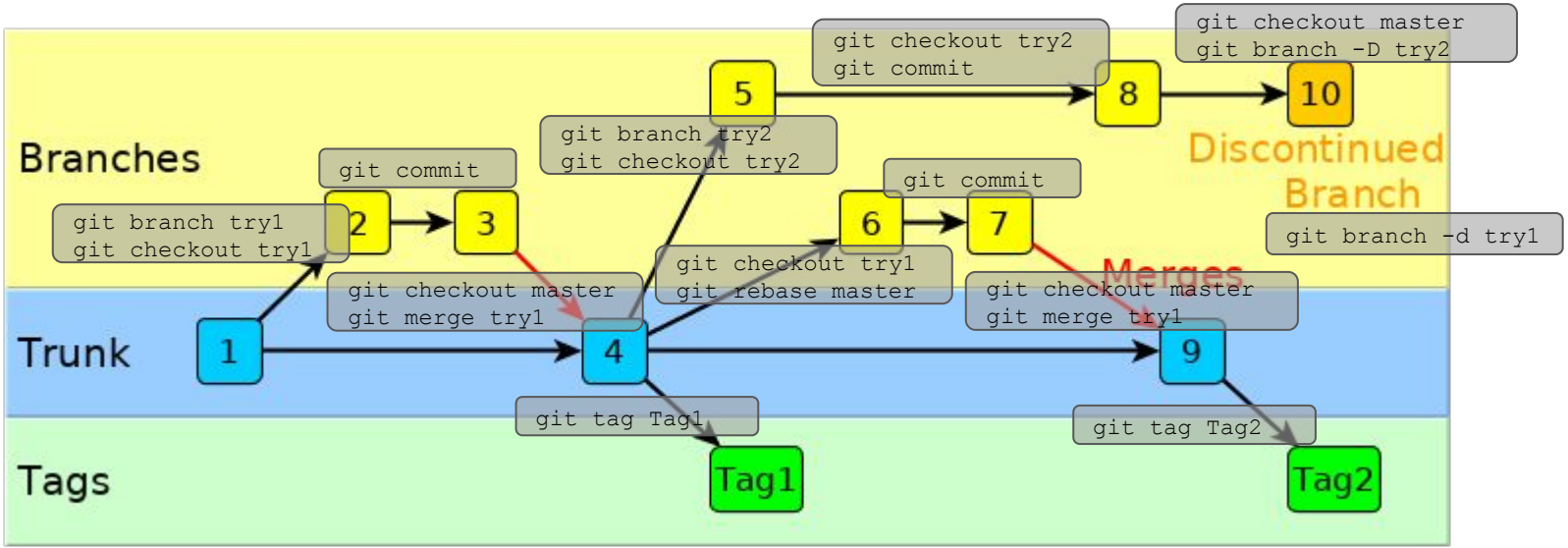
```
$ git pull
Already up-to-date.
$ vi Makefile
$ git commit -am 'Simplify Makefile, switch to gcc'
[master 5c0bfbd] update Simplify Makefile, switch to gcc
1 file changed, 3 insertions(+), 3 deletions(-)
$ git push
Counting objects: 4, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 416 bytes | 0 bytes/s, done.
Total 4 (delta 3), reused 0 (delta 0)
remote: Resolving deltas: 100% (3/3), completed with 3 local
objects.
To ssh://github.com/biell/project1.git
   38ad2e3..5c0bfbd  master -> master
$ make
...
$
```

Working with branches

- You can create a branch anytime you want to try something
- You can make as many commits and changes as you want to the branch
- You can switch back and forth between branches
- You can review the differences between two branches
- You can merge any branch with any other branch
- Merging a branch doesn't delete it

```
$ git branch
* master
$ git branch try1
$ git branch
* master
  try1
$ git checkout try1
Switched to branch 'try1'
$ git branch
  master
* try1
$ git diff master try1
diff --git a/file1 b/file2
...
$ git checkout master
Switched to branch 'master'
$ git merge try1
Updating 7acf9b8..dcda18a
...
$ git branch -d try1
Deleted branch try1 (was dcda18a).
$ git branch
* master
```

Revision control in action with Git



github.com is great for scientific research  + 

- https://github.com/gromacs/gromacs
- https://github.com/numpy/numpy
- https://github.com/FFTW/fftw3
- https://github.com/JuliaLang/julia
- https://github.com/opencollab/scilab

Useful reference

Revision Control	• https://betterexplained.com/articles/a-visual-guide-to-version-control/ • https://en.wikipedia.org/wiki/Version_control
SVN	• http://maverick.inria.fr/Members/Xavier.Decoret/resources/svn/index.html
Git	• https://try.github.io/ • https://git-scm.com/docs/gittutorial • http://gitreal.codeschool.com/
GitHub	• https://github.com/
Gitlab	• https://gitlab.com/
Bitbucket	• https://bitbucket.org/product

Questions and Open Forum

UNIVERSITY OF DELAWARE

Information Technology
Research Computing

Let's try git out

https://try.github.io/

After that, try some advanced topics

http://gitreal.codeschool.com/