Version Control (PHY1610 lecture 6)

Ramses van Zon

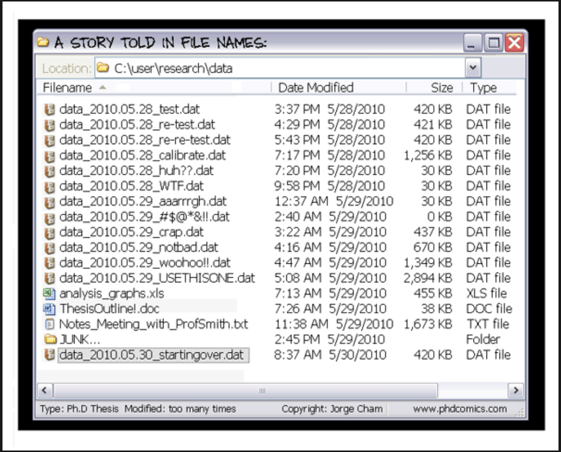
Winter 2022

What is version control?

- Version Control is a tool for managing changes in a set of files.
- Keeps historical versions for easy tracking.
- It essentially takes a snapshot of the files (code) at a given moment in time.

Why use it?

- Makes collaborating on code easier/possible/less violent.
- Helps you stay organized.
- Allows you to track changes in the code.
- Allows reproducibility in the code.



A STORY TOLD IN FILE NAMES:

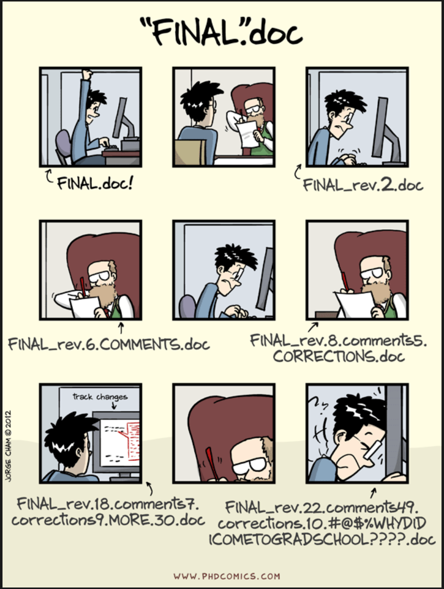
Location: C:\user\research\data

Filename	Date Modified	Size	Type
data_2010.05.28_test.dat	3:37 PM 5/28/2010	420 KB	DAT file
data_2010.05.28_re-test.dat	4:29 PM 5/28/2010	421 KB	DAT file
data_2010.05.28_re-re-test.dat	5:43 PM 5/28/2010	420 KB	DAT file
data_2010.05.28_calibrate.dat	7:17 PM 5/28/2010	1,256 KB	DAT file
data_2010.05.28_huh??.dat	7:20 PM 5/28/2010	30 KB	DAT file
data_2010.05.28_WTF.dat	9:58 PM 5/28/2010	30 KB	DAT file
data_2010.05.29_aaarrgh.dat	12:37 AM 5/29/2010	30 KB	DAT file
data_2010.05.29_#$@*&!!.dat	2:40 AM 5/29/2010	0 KB	DAT file
data_2010.05.29_crap.dat	3:22 AM 5/29/2010	437 KB	DAT file
data_2010.05.29_notbad.dat	4:16 AM 5/29/2010	670 KB	DAT file
data_2010.05.29_woohoo!!.dat	4:47 AM 5/29/2010	1,349 KB	DAT file
data_2010.05.29_USETHISONE.dat	5:08 AM 5/29/2010	2,894 KB	DAT file
analysis_graphs.xls	7:13 AM 5/29/2010	455 KB	XLS file
ThesisOutline!.doc	7:26 AM 5/29/2010	38 KB	DOC file
Notes_Meeting_with_ProfSmith.txt	11:38 AM 5/29/2010	1,673 KB	TXT file
JUNK...	2:45 PM 5/29/2010		Folder
data_2010.05.30_startingover.dat	8:37 AM 5/30/2010	420 KB	DAT file

Type: Ph.D Thesis Modified: too many times Copyright: Jorge Cham www.phdcomics.com

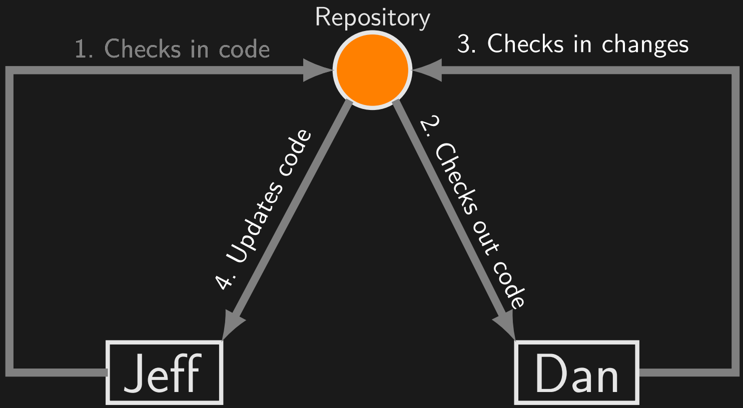
src: PhD Comics

And when something goes wrong, you can back up to the last working version.

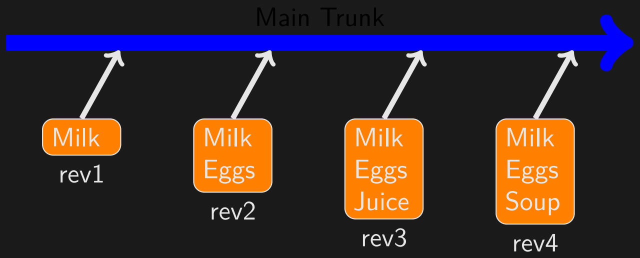

"FINAL".doc
FINAL.doc!
FINAL_rev.2.doc
FINAL_rev.6.COMMENTS.doc
FINAL_rev.8.comments5.CORRECTIONS.doc
track changes
FINAL_rev.18.comments7.corrections9.MORE.30.doc
FINAL_rev.22.comments49.corrections.10.#@$%WHYDID ICOMETOGRADSCHOOL????.doc
JORGE CHAM © 2012
WWW.PHDCOMICS.COM

https://git-scm.com

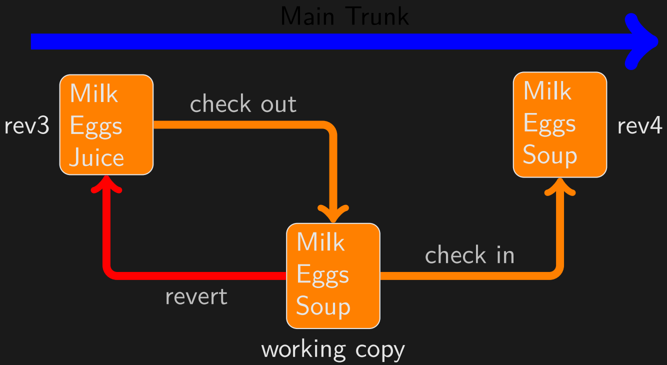
How does version control work?



Basic Checkins



Checkout and edit



Version Control: git

There are many types and approaches to version control.
Here we will introduce one implementation: git.

There are four main things you need to know how to do to get started with git:

1. Initialize a git repository.
2. Commit files to the repository.
3. Delete files from the repository.
4. Where to find more information.

- Linux
  - `yum/.../apt-get install git`
- MacOS
  - Xcode
  - fink/macports/homebrew
  - git OSX installer
- Windows (MobaXterm)
  - MobaXterm: `apt-get install git`
  - https://gitforwindows.org
- Teach cluster
  - `module load git`

Version control: setup a *local* repository

The first thing to do is set up a repository for your code.

```
$ module load git   # if on Teach
$ cd code
$ git init --initial-branch main # creates a repository for this directory, in the 'main' branch
Initialized empty Git repository in /home/s/scinet/rzon/code/.git/
$
```

This creates a `.git` directory, in the code directory, which contains the repository information.

```
$ ls -a
. .. .git
```

Version Control: setup your identity

The first time you might try to use git to commit, it might complain if it can't identify who are you. . .

```
*** Please tell me who you are.
Run
git config --global user.email "youremail@example.com"
git config --global user.name "FirstName LastName"
to set your account's default identity.
Omit --global to set the identity only in this repository.
```

Best to set these in advance instead of waiting for this error:

```
$ git config --global user.email "rzon@scinet.utoronto.ca"
$ git config --global user.name "Ramses van Zon"
```

Version control: adding repository files

Adding files to the repo - First you must *add* the files to the staging area, then you *commit*:

```
$ echo "some data" > temp.txt
$ cp temp.txt temp2.txt
$ ls
temp2.txt temp.txt
$ git add temp.txt temp2.txt
$ git commit -m "First commit for my repository"
[main (root-commit) dd9d139] First commit for my repository
 2 files changed, 2 insertions(+)
 create mode 100644 temp.txt
 create mode 100644 temp2.txt
$
```

Notice that you must always stage the files with `git add` before actually commiting them with `git commit`.

Version Control: comparing file versions

Let's update some data and see how can we compare it with the already commited files. . .

```
$ echo "some more data" >> temp.txt
$ git diff temp.txt
diff --git a/temp.txt b/temp.txt
index 4268632..fdd9353 100644
--- a/temp.txt
+++ b/temp.txt
@@ -1 +1,2 @@
 some data
+some more data
$ git add temp.txt
$ git commit -m "updating data due to ..."
```

Version Control: status report

Revisiting what it has been done in the repo: logs

```
$ git log
commit b0292f6e3a820856f1d29b5aee2acdc4fd9e73c9 (HEAD -> main)
Author: Ramses van Zon <rzon@scinet.utoronto.ca>
Date:   Thu Jan 27 09:50:01 2022 -0500

    updating data due to ...

commit dd9d13999ac5073089e6ea4282b0c78854256bc1
Author: Ramses van Zon <rzon@scinet.utoronto.ca>
Date:   Thu Jan 27 09:49:02 2022 -0500

    First commit for my repository
```

Version Control: detailed status report

Revisiting what it has been done in the repo: logs

```
$ git log --stat
commit b0292f6e3a820856f1d29b5aee2acdc4fd9e73c9 (HEAD -> main)
Author: Ramses van Zon <rzon@scinet.utoronto.ca>
Date:   Thu Jan 27 09:50:01 2022 -0500

    updating data due to ...

 temp.txt | 1 +
 1 file changed, 1 insertion(+)

commit dd9d13999ac5073089e6ea4282b0c78854256bc1
Author: Ramses van Zon <rzon@scinet.utoronto.ca>
Date:   Thu Jan 27 09:49:02 2022 -0500

    First commit for my repository

 temp.txt  | 1 +
 temp2.txt | 1 +
 2 files changed, 2 insertions(+)
```

Inspecting a specific previous version

To look at a previous commit temporarily, use the commit hash and do

```
$ git checkout dd9d13999ac5073089e6ea4282b0c78854256bc1 -b tempbranch
```

This creates a temporary branch, leaving the main branch intact.

Without the `-b tempbranch` option, we'd get the same but lose track of the real HEAD of the repo. Unless you jotted down the commit hash of commit you were at, it's hard to get back.

But because we made this a branch, there is an easy undo:

```
$ git checkout main
$ git branch --delete tempbranch
```

Rollback

Rolling back to a specific previous version and remove subsequent commits permanently from the `main` branch, you can do

```
$ git reset dd9d13999ac5073089e6ea4282b0c78854256bc1 --hard
$
```

Without the `--hard` option, only the repo in `.git` is update, but the files in the working directory would not have been restored.

Reverting changes – Reset, Checkout, and Revert

```
git reset <Commit>    # Discard commits in a private branch or throw away uncommited
git reset <file>      # Unstage a file
git checkout <Commit> # Inspect old snapshots (and lose your HEAD)
git checkout <Branch> # Switch between branches
git checkout <File>   # Discard changes in the working directory
git revert <Commit>   # Create a new commit that undoes the given commit
```

More details at https://www.atlassian.com/git/tutorials

Version control: removing repository files

Let's look at what we've done so far.

```
$ git log
Author: Ramses van Zon <rzon@scinet.utoronto.ca>
Date:   Thu Jan 27 09:49:02 2022 -0500

    First commit for my repository
$
```

Suppose you want to delete a file:

```
$ git rm temp2.txt
$ git commit -m "Remove temp2.txt"
[main f1af560] removed temp2.txt
 1 file changed, 1 deletion(-)
 delete mode 100644 temp2.txt
 $
```

Version Control: Git Branches

Version Control, Star Trek Style
revision 1
revision 2
branch 2a
branch 2b
Aieee! Roll back! Roll back!

- Shows current branch

```
$ git branch
```

- Show all branches

```
git branch -a
```

- Shows all remote branches (later more on those)

```
git branch -r
```

- Creates a branch

```
git branch MYNEWBRANCH
```

- Switch to the branch

```
git checkout branchname
```

Remote repositories

- Git is a distributed version control system.
- You can have clone a repo anywhere to copy it elsewhere.
- Each clone is a full-fledged repo.
- You can push and pull the state of one repo to another.
- You do not have to have one centralized, authoritive repo, but often, that is still convenient.
- Clones can live on remote computers or in the cloud (e.g. github, gitlab)
- Git can interact with remote repos using `ssh` (as well in other ways).
- Remote repos often don't need a working directory, they can be bare .git repos.

Remote repositories - Example

Setup a remote repo teach

```
local> ssh USER@teach.scinet.utoronto.ca
teach> module load git
teach> mkdir my_project.git
teach> cd my_project.git
teach> git init --initial-branch main
teach> git config receive.denyCurrentBranch ignore
teach> echo "hello" > world.txt
teach> git add world.txt
teach> git commit -m 'hello world'
teach> exit
```

Clones it on your local machine:

```
local> git clone USER@teach.scinet.utoronto.ca:my_project.git
local> cd my_project
local> touch temp.txt temp2.txt
local> git add temp.txt temp2.txt
local> git commit -m "Added files"
local> git push -u origin main
local> ssh  USER@teach.scinet.utoronto.ca
teach> git reset --hard
teach> ls
world.txt temp.txt temp2.txt
```

Version Control: a few tips

- Use it, will save you trouble.
- Commit often.
- Include sensible comment messages.
- Do not commit derivative stuff (e.g. log files, executables, compiled modules, . . .)
- can be used for several different kind of projects: code development, collaborations, papers, . . .
- There are other VC systems: hg, svn, cvs, . . .

For a very extensive tutorial, go here: https://www.vogella.com/tutorials/Git/article.html

You probably heard of web-based options for git as well:

- GitHub: https://github.com
- Bitbucket: https://bitbucket.org

These service can host your repos as a remote repo, and make them publically available.