Revision control
Introduction to **git**

Waterford Institute of Technology

June 19, 2015

John Fitzgerald

Revision control

What is it?

Also known as

- version control
- source control

Application to keep track of changes to file system

- your BlueJ projects
  - single developer
  - code on single machine
- managing app development
  - large developer team
  - same code base on multiple machines

Revision control

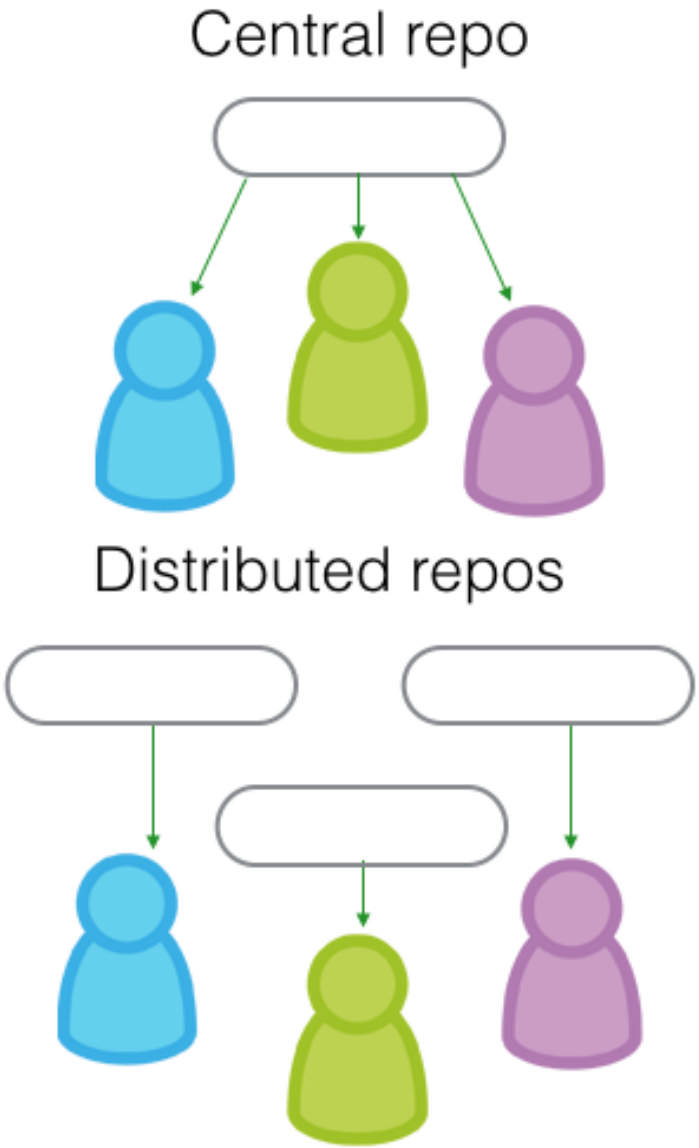
Centralized v Distributed

Centralized

- single data store, the repository
- files checked in and out
- checked-out files locked
- merging files non-trivial

Distributed

- no central repo
- each developer has copy
- merging files easier



Distributed Versioning Application

Git

Initially designed & developed by Linus Torvalds

- Became available 2005
- Supported by large developer community
- Free open source
- Extremely popular
- April 2014: 37% repos use *git*

git for the true beginner

For a single-developer team

This presentation and accompanying lab designed for beginner

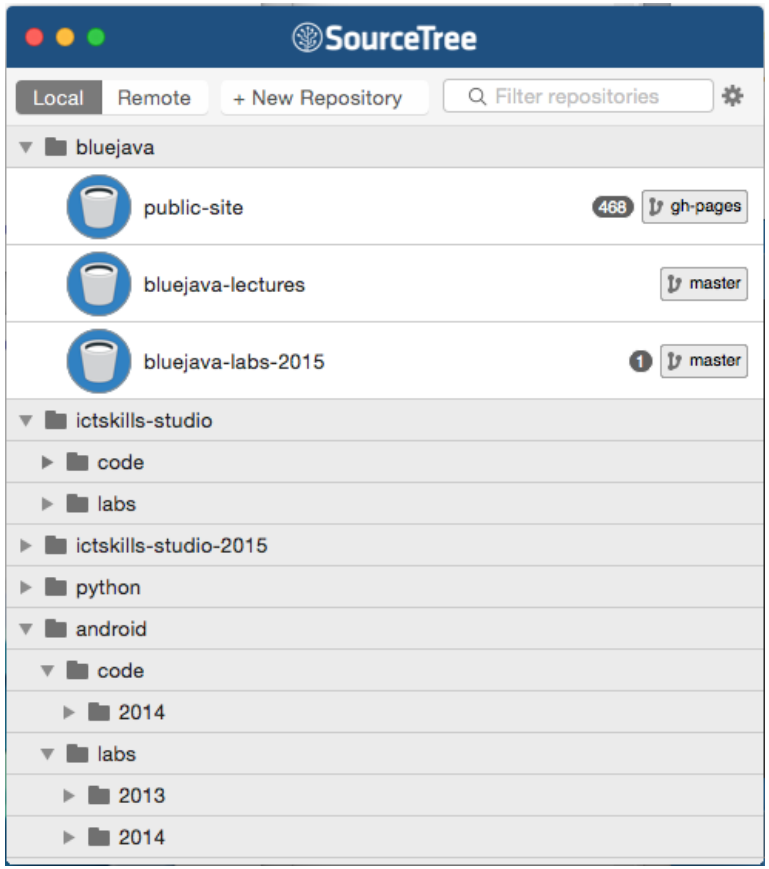
- Using command line git.
- Several GUIs exist (ignored).
  - Egit for Eclipse
  - SourceTree
  - GitHub

Developing with *git*

Graphical User Interfaces (gui)

Atlassian SourceTree

- Free Git client for Windows & Mac
- Helps organise your repos
  - Say goodbye to cli
  - Or use combination gui-cli



Developing with *git*

Graphical User Interfaces (gui)

GitHub

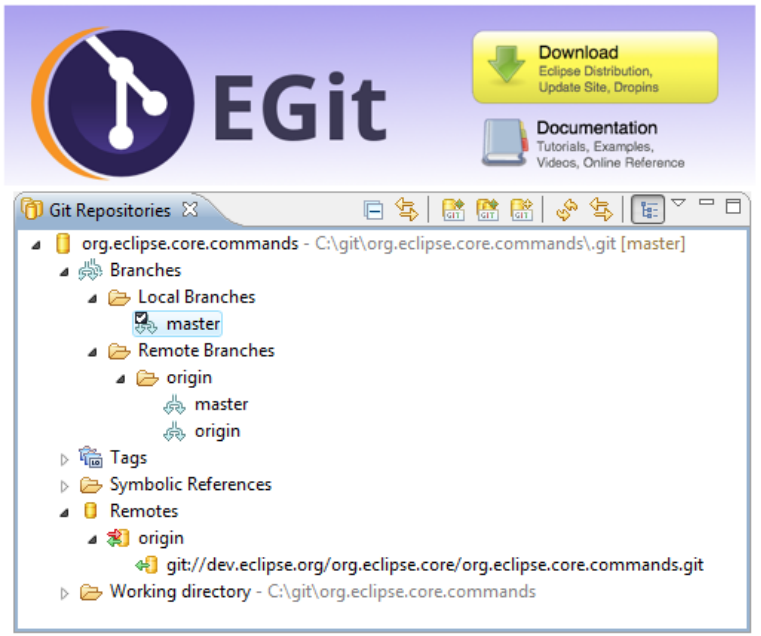
- Free Git client for Windows & Mac
- Manages repositories
- SourceTree competitor

Developing with git

Graphical User Interfaces (gui)

EGit: an Eclipse git gui plugin

- Free Git client for Eclipse IDE

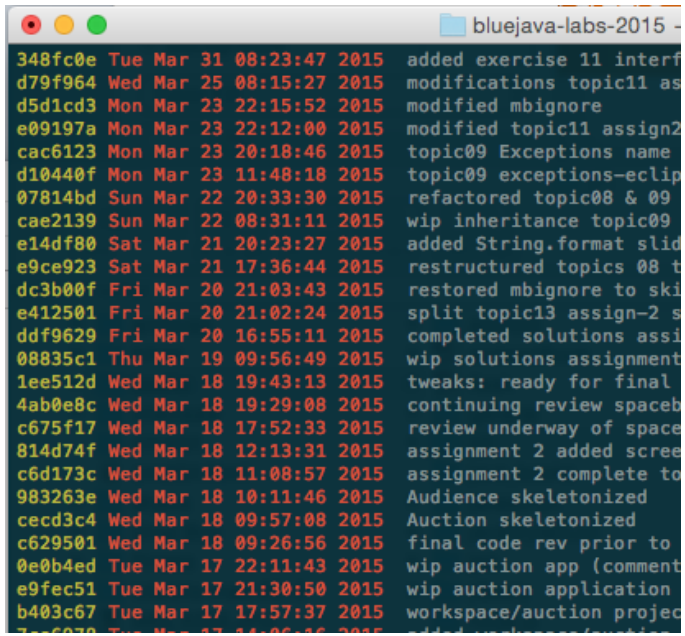


git for the true beginner

Using command line git

Why command line line *git*?
Why not begin with a git gui?

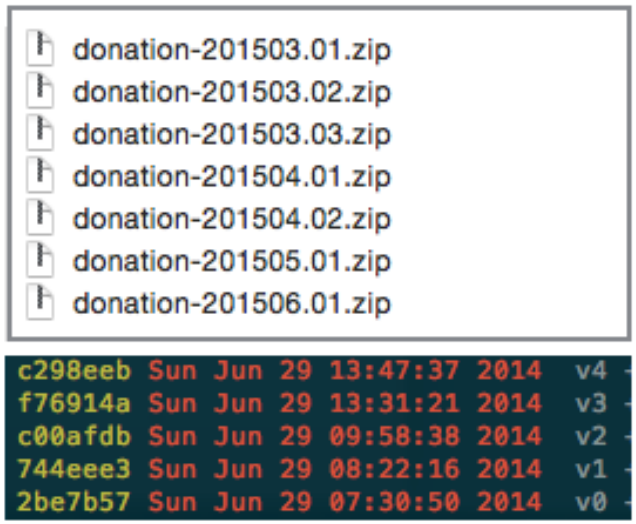
- Acquire deeper understanding
- Easier to learn
- Start with small set commands
- Follow simple procedure initially
- Gradually extend knowledge
- Beware: steep learning curve
- Fallback recommended



Keeping track of changes

Homespun approach

- Create archives during dev cycle
- Apply time stamp & version
- Cumbersome but effective
- Also provides fallback
- Disadvantage: File proliferation
- Contrast *git*: One file



Using *git* from command line

Summary

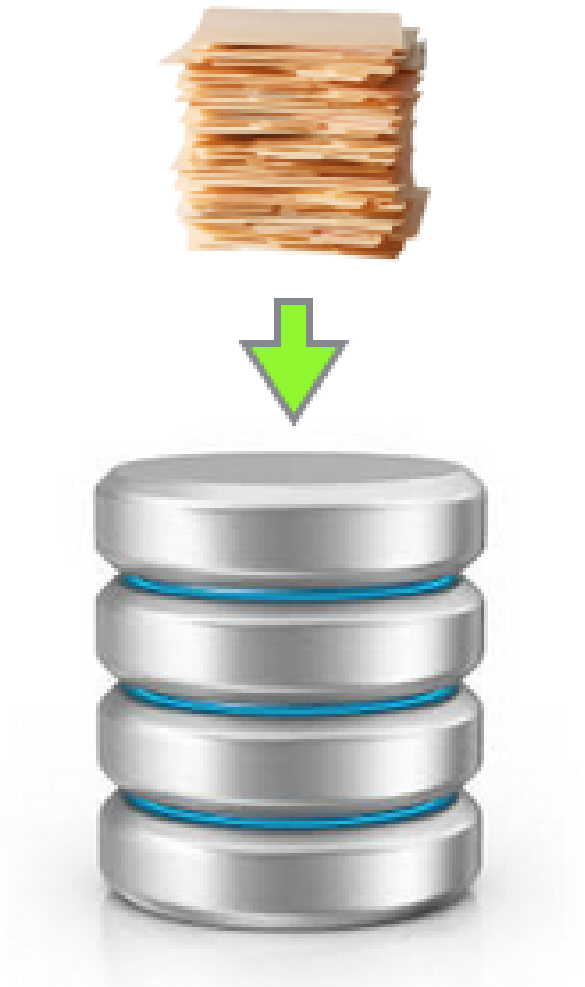
Associated lab provides detailed instructions on what follows.

- Download and install *git* app
- *cd* to project folder
- create local repo
  - *git init*
  - created in folder *.git*
- create a *.gitignore* file
  - exclude nominated files

Using *git* from command line

Add project to local repo

- stage project
  - *git add .*
  - Warning: note requirement for period (.) following *git add*
- commit the staged files to repo
  - *git commit -m 'baseline project'*
- copy project now in local repo



BitBucket.org

Push project to remote repo

Register an account & create a repo, example *donation*

In your project folder execute:

```
git remote add origin git@bitbucket.org:<yourdomain>/donation.git
git push -u origin -all
```

Response should be similar to this:

```
$ git push -u origin --all
Counting objects: 3, done.
Writing objects: 100% (3/3), 217 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To git@bitbucket.org:<yourdomain>/donation.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
```

BitBucket.org

Clone the remote repo

Exact copy of repo now exists on BitBucket
Verify this as follows:

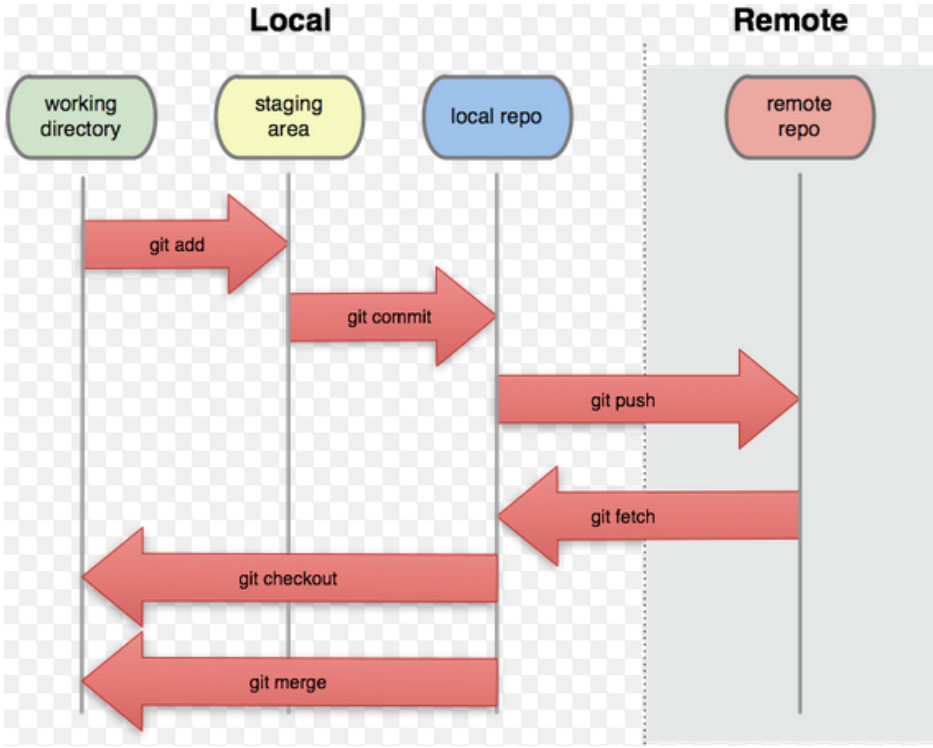
- cd to a new temp folder
- Clone the repo
- Check content of cloned folder
- Should match pushed *donation* project
- Try exercise on different computer

```
$ git clone git@bitbucket.org:<yourdomain>/donation.git
Cloning into 'donation'...
remote: Counting objects: 351, done.
remote: Compressing objects: 100% (324/324), done.
remote: Total 351 (delta 136), reused 0 (delta 0)
Receiving objects: 100% (351/351), 1000.35 KiB | 120.00 KiB/s, done.
Resolving deltas: 100% (136/136), done.
Checking connectivity... done.
```

Developing with *git*

Frequently-used commands

- git status
- git commit
- git push
- git pull
- git log



Developing with *git*: a simplified approach

First session

- Create local repo
- Stage and commit file system
- Create remote repo
- Push local to remote
- Ensure working directory clean at end session

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean
```

Developing with *git*: a simplified approach

Resuming session

- cd to working folder
- git status to verify working directory clean
- git pull to update working tree
- Continue dev session
- At conclusion session add, commit and push to remote

```
$ git push origin master
Counting objects: 9, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (8/8), done.
Writing objects: 100% (9/9), 126.26 KiB | 0 bytes/s, done.
Total 9 (delta 4), reused 0 (delta 0)
To git@bitbucket.org:witpress/donation.git
   11f8391..b99ee02  master -> master
```

Developing with *git*: a simplified approach

Check logs

- Basic command *git log*
- More sophisticated formatting available:

```
git log --pretty=oneline --max-count=10
```

```
d5d1cd3771f59a01b7bff67b962f33efe97a41a0 modified mbignore
e09197ab28989686f4ecedff277eee2df7727993 modified topic11 assign2
d10440fa3da18ec536771a687939461c2ebeaaa1 topic09 exceptions-eclipse lab
07814bd0608c1ed8ee1d328535cdf9ff180f7cb5 refactored topic08 & 09 done
```

```
git log --pretty=format:'%C(yellow)%h %Cred%ad  %Creset%s' --date=local
```

```
d5d1cd3  Mon Mar 23 22:15:52 2015  modified mbignore
e09197a  Mon Mar 23 22:12:00 2015  modified topic11 assign2
cd10440f Mon Mar 23 11:48:18 2015  topic09 exceptions-eclipse lab
07814bd  Sun Mar 22 20:33:30 2015  refactored topic08 & 09
```

Developing with *git*: a simplified approach

Tags

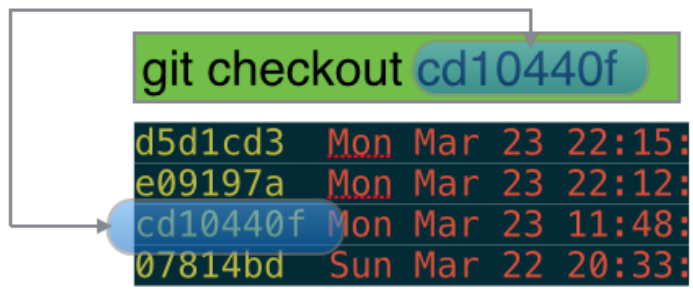
Tags may be used to denote specific points in the project history

- List tags: *git tag*
- List tags + messages: *git tag -n*
- Add local tag: *git tag -a tagName -m 'message'*
- Push tags to remote: *git push –tags*
- Add tag to specific commit: *git tag -a v0 ecc947 -m "message"*
- Delete tag locally: *git tag -d tagName*
- Then from remote repo: *git push origin: refs/tags/tagName*
- List remote tags: *git ls-remote*

Developing with *git*: a simplified approach

Checkout earlier commit

- Using tag
  - *git checkout tagName*
- Using hash
  - *git checkout cd10440f*
- Roll back to last commit
  - *git checkout -f*



Developing with *git*: a simplified approach

Checkout master following checkout of earlier commit

- Here we discuss only a single line of development.
- This is on a default branch referred to as the *master* branch.
- Commit changes only while at the tip of the master branch.
- Checkout previous commit: one no longer at tip master.
  - *git checkout tagName*
- To return to tip of master:
  - *git checkout master*
- But first any changes rolled back with:
  - *git checkout -f*

Developing with *git*: a simplified approach

Disaster recovery: scenario 1

Roll back to an archived version - the fallback.

- Unarchive the backup
- Delete all files and folders from working tree
  - Exceptions: .git folder & .gitignore
- Copy backup to working folder
- Add all, commit and push

Developing with *git*: a simplified approach

Disaster recovery: scenario 2

Roll back to an earlier commit

- Checkout the commit you wish to revert to
- Copy the working tree to a recovery folder
  - Do not copy .git folder
- *git checkout master*
- Delete contents working tree
  - Do not delete .git folder
- Copy backup to working folder
- Add all, commit and push

Developing with *git*: a simplified approach

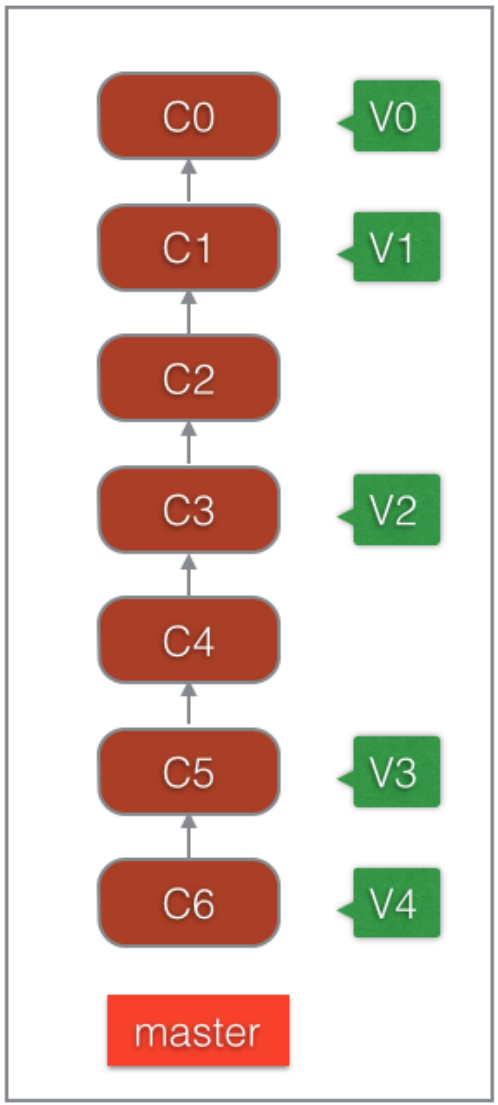
Repair error at earlier commit: a special case

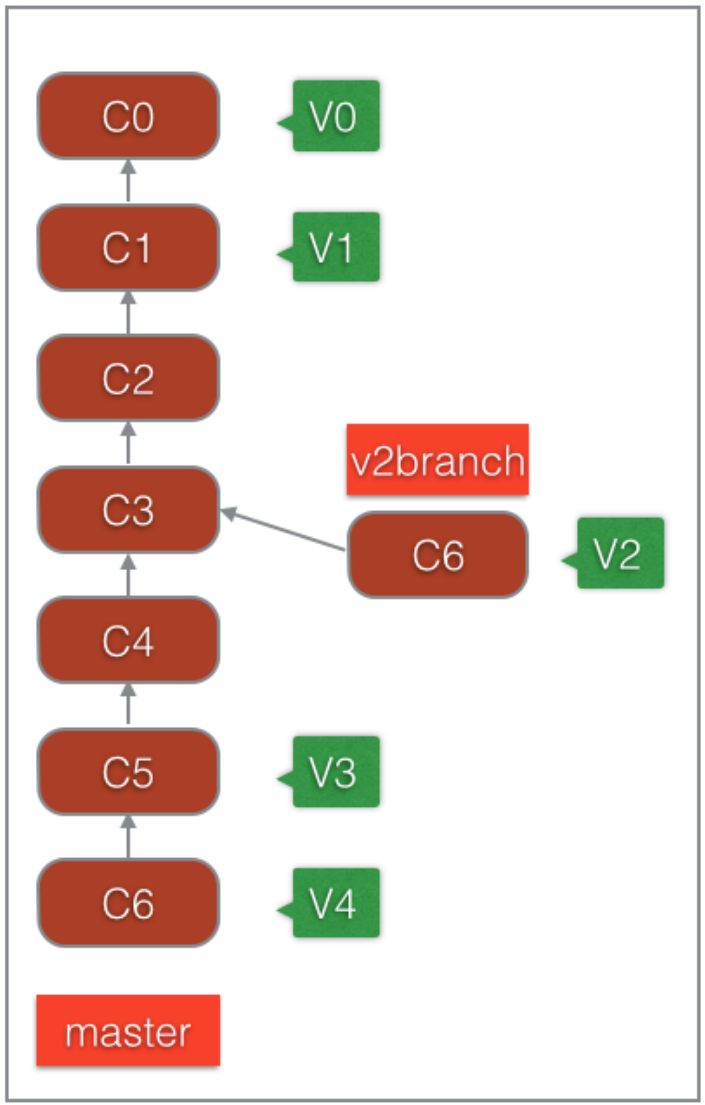
Assume we have an application comprising master branch, several commits and a series of tags, v0, v1 and so on and we wish to fix a bug at v2. Here's one approach:

- Checkout v2.
- Repair the error.
- Create and checkout new branch
  - git checkout -b v2branch
- Add and commit changes
  - git add .
  - git commit -m 'bug fix at original tag v2'
- Next, delete the original tag local & remote & recreate at this branch and push
  - git tag -d v2
  - git push origin: refs/tags/v2
  - git tag -a v2 -m 'v2 recreated following bug fix'
  - git push –tags

Developing with *git*: a simplified approach

Repair error at earlier commit: a special case





GitHub

Clone the programming course repo

GitHub a competitor to BitBucket
Private repos free on BitBucket
Public repos free on GitHub
Programming course hosted on GitHub in public repo

```
git clone https://github.com/usplitu/programming.git
Cloning into ‘programming'...
remote: Counting objects: 1814, done.
Receiving objects:  11% (211/1814), 4.52 MiB | 229.00 KiB/s
```

git

Parting advice

There may be trouble ahead...
So use zip files as fallback
Until competence acquired
Else you'll be obliged to face the music
And have teardrops to shed

- donation_20150331.01.zip
- time stamp + daily version

git

Summary

- This has been very brief intro
  - Sufficient for one-person development
- Steep learning curve
  - Use continuously
  - Be prepared for hiccups
  - Have file retrieval plan
  - Learn by doing

Referenced Material

1. Git usage statistics

`https://www.wikivs.com/wiki/Git_vs_Subversion` [Accessed 2015-03-03]