

# Hall C Software Development

From the perspective of a user



# Outline

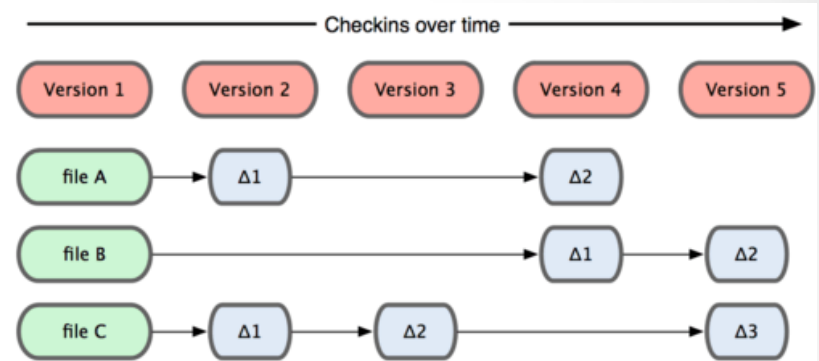
- Version management with Git
  - Overview of git in practice
  - Examples of pushing/pulling updates
  - Issue tracking
- Development of a new build system – SCons
  - Overview of SCons
  - Current status of build/configure system

# Git

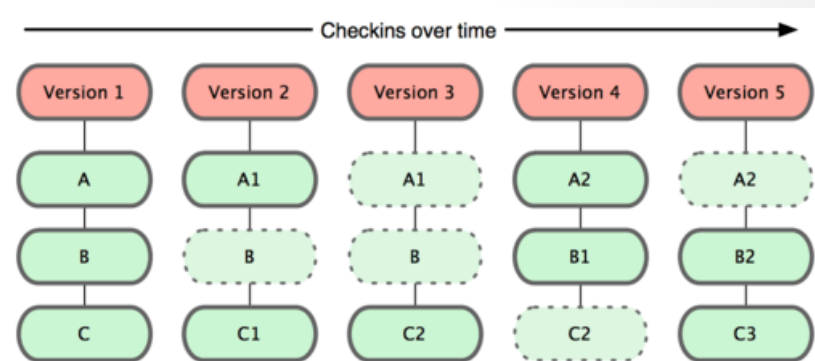
- Free and open source distributed version control system
- Created by the Linux development community (primarily for version control of the Linux kernel), with a focus on:
  - Speed
  - Simple design
  - Strong support for non-linear development (1000's of branches)
  - Fully distributed
  - Able to handle large projects efficiently

# Git Philosophy

- CVS, Subversion, etc.
  - Based on “changes” to files
  - Difficult to “back out” once changes are committed



- Git
  - Based on “snapshots” of filesystems
  - Trivial to back out changes
  - This is hugely important for large parallel development, as well as for newbie users!



# Github.com

- Online project hosting using git
- Developers use their own account on github.com to create “forks” of JeffersonLab repositories
- Changes are committed to a developer’s forked repository, and at some point may be merged with the main JeffersonLab repository, via a pull request.
- The “gatekeeper” for the Jefferson Lab repository governs this process.



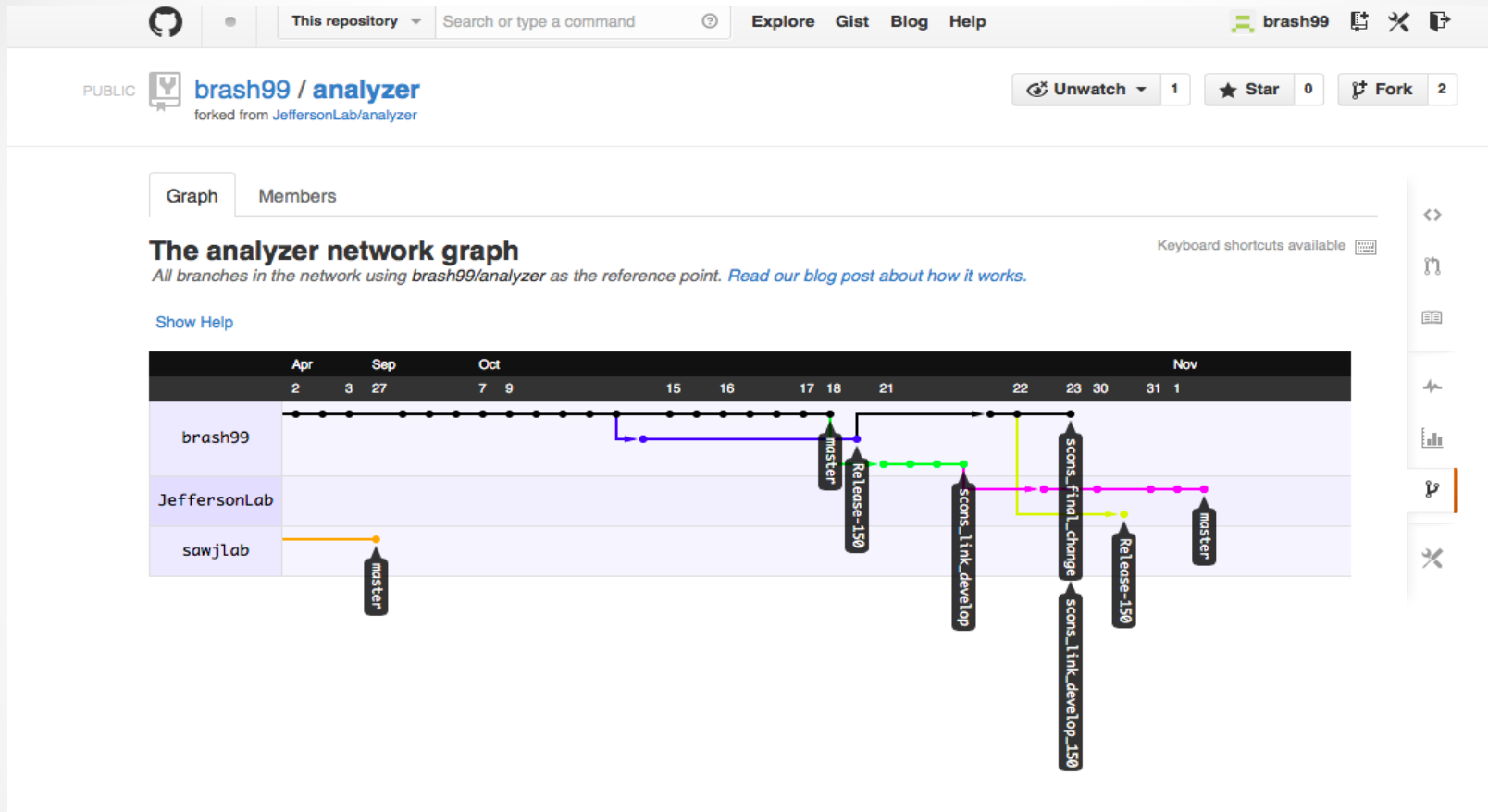
# Current JLab Git Projects

- Hall A C++ Analyzer
- Hall C C++ Analyzer
- Hall C Fortran engine
- Hall C Replay x 2 (for comparisons)
- Hall C Geant simulation of Compton Polarimeter
- SIMC (Monte Carlo for Halls A and C)
- MC\_SHMS\_SINGLE
- TreeSearch (Hall A TreeSearch track reconstruction)

# Development under Git

- Easy to install and update code on JLab systems as well as on local (Mac and Linux) machines
  - git clone [git@github.com:brash99/analyzer.git](https://github.com/brash99/analyzer.git)
  - git branch -a
    - Master
    - remotes/origin/HEAD -> origin/master
    - remotes/origin/Release-070
    - remotes/origin/Release-100
    - remotes/origin/Release-110
    - remotes/origin/Release-120
    - remotes/origin/Release-130
    - remotes/origin/Release-140
    - remotes/origin/Release-150
    - remotes/origin/master
    - remotes/origin/scons\_final\_change
    - remotes/origin/scons\_link\_develop
    - remotes/origin/scons\_link\_develop\_150
  - git pull origin <branch-name>

# Example Network Graph



```
$ git checkout -b sconslink_final_change  
(make changes)  
$ git commit -a  
$ git push origin sconslink_final_change
```

[Analyzer Network](#)




# Example Network Graph

The screenshot shows a GitHub pull request interface for the repository `JeffersonLab/analyzer`. At the top, it indicates the repository is `PUBLIC` and has `7` Unwatch, `1` Star, and `2` Fork actions. The pull request is titled `brash99 wants to merge 3 commits into JeffersonLab:master from brash99:scons_link_de...` and is marked as `Closed`. Below the title, there are tabs for `Discussion`, `Commits 3`, and `Files Changed 5`. The main content area shows a message from `brash99` stating: `brash99 opened this pull request 17 days ago` with the title `Scons link develop 150`. It also shows `No one is assigned` and `No milestone`. To the right, a `Closed` badge is present, along with statistics: `+108 additions` and `-10 deletions`. Below this, a list of commits is shown, including `hansenjo` (Version is 1.5.25), `brash99` (Modifications to various SConscripts to fix bug in cleanup option), and `brash99` (Create softlink to libraries within SConscript files to allow use of ...). A comment from `hansenjo` is also visible, stating `hansenjo commented` and `Closed by 3bfc690`. At the bottom, a `Closed` badge is shown with the text `hansenjo closed the pull request 16 days ago`.

Users/developers who are “watching” development of the JeffersonLab/analyzer repository are notified by email of pull requests.

Typically, other developers, as well as the gatekeeper, can make comments on the proposed pull request prior to it being accepted (or rejected).

# Example Network Graph

PUBLIC  **brash99 / analyzer**  
forked from JeffersonLab/analyzer


Unwatch 1 Star 0 Fork 2

**Create softlink to libraries within SConscript files to allow use of system-wide SCons installation.** [Browse code](#)

`scons_final_change` + `scons_link_develop_150`

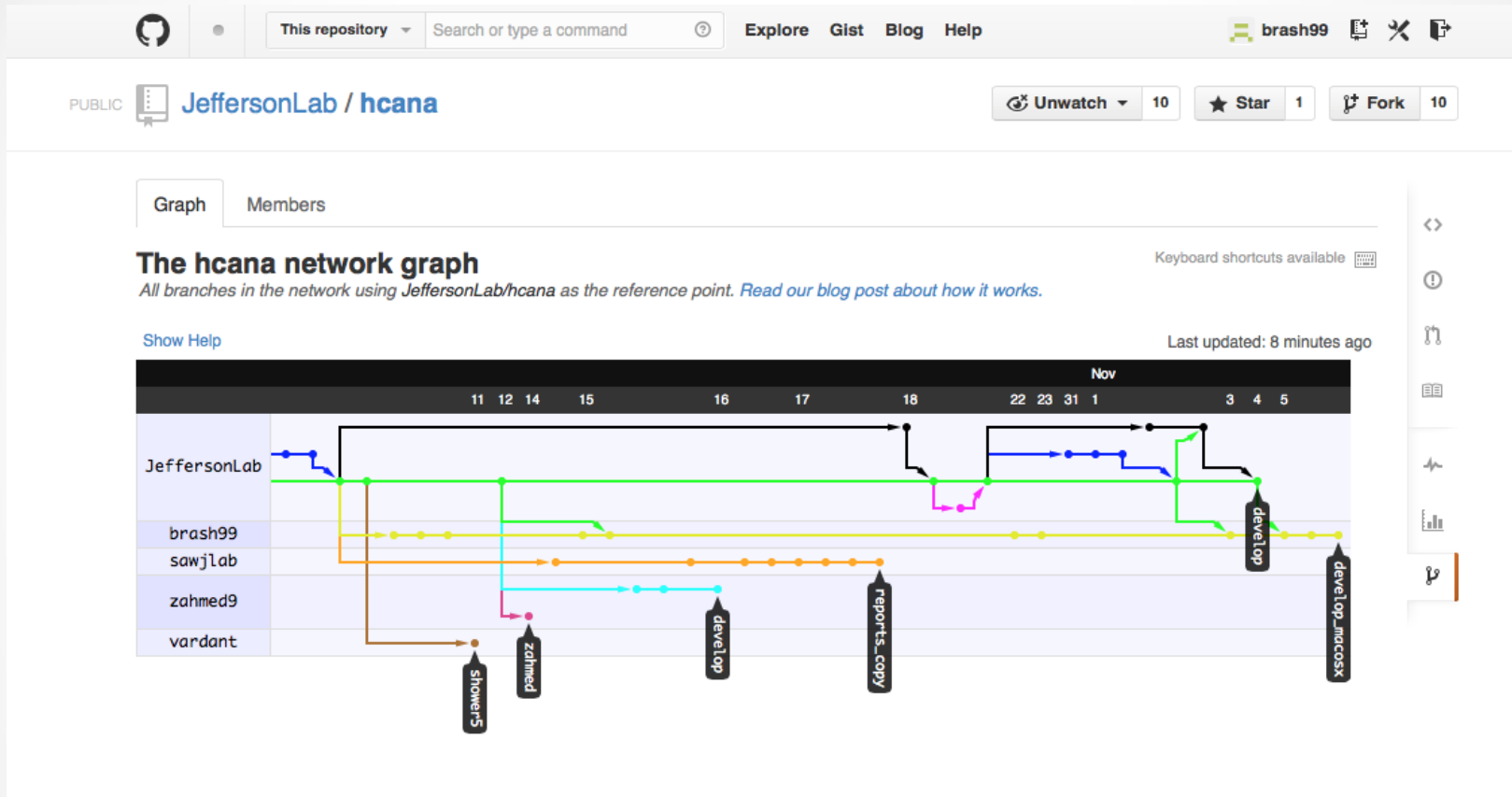
**brash99** authored 17 days ago 1 parent [ca74417](#) commit [3bfc690f7e7ead6b60ee1decb02e65db2ab3e465](#)

Showing 4 changed files with 103 additions and 38 deletions. [Show Diff Stats](#)

12  SConstruct.py [View file @ 3bfc690](#)

```
... .. @@ -30,9 +30,11 @@ def rootcint(target,source,env):
30 30 # print "Construction variable = '%s', value = '%s'" % (key, dict[key])
31 31
32 32 ##### Check SCons version #####
33 -print('!!! You should be using the local version of SCons, invoked with:')
34 -print('!!! ./scons/scons.py')
35 -EnsureSConsVersion(4,9,9)
36 +print('!!! You should be using the local version of SCons, invoked with:')
37 +print('!!! ./scons/scons.py')
38 +print('!!! Building the Hall A analyzer and libraries with SCons requires')
39 +print('!!! SCons version 2.1.0 or newer.')
40 +EnsureSConsVersion(2,1,0)
41
42 ##### Hall A Build Environment #####
43 #
44 .. @@ -41,8 +43,10 @@ def rootcint(target,source,env):
45 43 baseenv.Append(HA_SRC = baseenv.subst('$HA_DIR')+'/src ')
46 44 baseenv.Append(HA_DC = baseenv.subst('$HA_DIR')+'/hana_decode ')
47 45 baseenv.Append(HA_SCALER = baseenv.subst('$HA_DIR')+'/hana_scaler ')
48 -baseenv.Append(SOVERSION = '1.5')
49 +baseenv.Append(MAJORVERSION = '1')
50 +baseenv.Append(MINORVERSION = '5')
51 baseenv.Append(PATCH = '25')
52 +baseenv.Append(SOVERSION = baseenv.subst('$MAJORVERSION')+'.'+baseenv.subst('$MINORVERSION'))
53 46 baseenv.Append(VERSION = baseenv.subst('$SOVERSION')+'.'+baseenv.subst('$PATCH'))
54 47 baseenv.Append(EXTVERS = '')
55 48 baseenv.Append(HA_VERSION = baseenv.subst('$VERSION')+baseenv.subst('$EXTVERS'))
```

# Example Network Graph



```
$ git remote add upstream git@github.com:JeffersonLab/hcana.git  
$ git fetch upstream  
$ git merge upstream/develop
```

# Issue Tracking in Git

PUBLIC JeffersonLab / ncana Unwatch 10 Star 1 Fork 10

Browse Issues Milestones [New Issue](#)

Everyone's Issues 9 9 Open 15 Closed Sort: Newest

Assigned to you 1  
Created by you 1  
Mentioning you 0

**Labels**

- enhancement 4
- bug 0
- duplicate 0
- invalid 0
- question 0
- wontfix 0

**Issues:**

- #24 **SCons build error: undefined reference to `typeinfo for THcFormula`**  
Opened by hansenjo 9 hours ago 1 comment
- #23 **Various SCons developments (consistency with Release-150 of PODD, among other things).**  
Opened by brash99 2 days ago
- #20 **Scaler analysis code**  
Opened by sawjlab 20 days ago
- #15 **Difference in calculation of hodo fptimes between ENGINE and HCANA**  
Opened by MarkKJones a month ago
- #13 **Time of flight calibration code** enhancement  
Opened by sawjlab a month ago
- #12 **EPICS events**  
Opened by sawjlab a month ago 1 comment
- #11 **Beam information** enhancement  
Opened by sawjlab a month ago
- #4 **Drift chamber debug flags** enhancement

- Issues to be solved can be created at any time, and assigned to a particular developer
- When pull requests are made, can be associated with one or more issues, and issue can be closed (if appropriate)

# A new build system - SCons

- Traditionally, Hall A/C software has been built with “make”
  - Platform/system/compiler dependent configuration handled within Makefiles (coupled with #ifdef statements within the code itself)
  - Dependency checking not included by default, and is based on timestamp.
  - Having an “autoconf”-like configuration is desirable, but GNU Autoconf is highly complex
  - Makefiles are platform-dependent, and incredibly cryptic – basically unreadable to non-experts – making changes and updates difficult
  - Libtool (management of libraries) not available for all platforms
- Is there something better out there?

# A new build system - SCons

- SCons is an open-source software construction tool
  - Written entirely in Python – power of a real programming language in configuration and build scripts ... plus, our students know and love Python!
  - Scripts are much more readable than Makefiles
  - Integrated functionality similar to Autoconf
  - Built-in support for C/C++, and easily extensible for other builders (ROOTCINT)
  - Built-in dependency-checking – based on MD5 signatures, and not timestamps – important for git.
  - Designed from the ground up for cross-platform builds
  - Currently used by the JLab DAQ group for EVIO

# Major Projects using SCons

- ASCEND - A system modeling package for engineering
- Cantera – A toolkit for chemical kinetics and thermodynamics
- CLAM – A framework to develop sophisticated audio analysis
- FreeNOS – A microkernel operating system written in C++
- IntensityEngine – A platform for 3D games and virtual worlds
- Lumiera – A professional video editor
- Madagascar – Geophysical data processing
- Nsound – C++ audio synthesis framework
- openEHR – Electronic Health Record standard
- V8 – Google's open source Javascript engine
- YafaRay – An open source raytracing engine

# SCons – Current status

- Build and configuration scripts have been written, tested, and committed for both PODD and HCANA
- Configuration checks for:
  - ROOT installation
  - gcc/g++ compiler installation and functionality
  - Platform-dependent compiler/linking flags (64/32 bit, Linux/MacOSX)
- Currently, we are maintaining the traditional Make system and SCons in parallel (for both PODD and HCANA)