Software & Computing

& HPS

N. Baltzell - HPS Collaboration Meeting - June 3, 2025

Intro

Advertisements

- **GSPDA Computing Bootcamp (last month)**
  - JLab Graduate Student and Postdoc Association
  - https://indico.jlab.org/event/954/timetable/#20250522
  - Looks like a very good non-experiment-specific introduction to computing at JLab
- **HSF/IRIS-HEP Software Basics Training at CERN (Hybrid)**
  - https://indico.cern.ch/event/1516608/
  - https://iris-hep.org/
- **CLAS12 Working Groups**
  - AI/ML: https://clasweb.jlab.org/wiki/index.php/CLAS12_AI_Group
  - Software: https://clasweb.jlab.org/wiki/index.php/CLAS12_Software_Center

3 **Thursday 22 May**	
Welcome	3
Lecture: Computing Farm Basics and Best Practices	3
Break	3
Lecture: Accessing the Farm from Your Computer	3
Hands-On: Accessing the Farm from Your Computer	3
Networking Luncheon	3
Hands-On: Navigating the Farm and Moving Files	3
Break	3
Hands-On: Using Git and Executing a Basic Script	3
4 **Friday 23 May**	
Lecture: The Batch System and Using swif2	4
Break	4
Hands-On: The Batch System and Using swif2	4
Lunch	4
Lecture: Special Interest Topic - Analysis Persistence and Robustness	4
Break	4
Hands-On: Special Interest Topic - Analysis Persistence and Robustness	4

We are very excited to announce that the workshop on software basics in HEP organised through the HEP Software Foundation and IRIS-HEP will take place at CERN (30/7-018 Kjell Johnsen Auditorium) and also via Zoom. We are very grateful for financial support from **CERN EP-SFT** and **HSF**.

The times for the workshop are in CEST time zone (CERN time).

Over three days we will cover the fundamentals of:

Unix (e.g. shell, bash and scripting)
https://swcarpentry.github.io/shell-novice/

Git and Github – how to version control your code
https://mambelli.github.io/git-novice/ (extended version of https://swcarpentry.github.io/git-novice/)

Python – fundamentals of using the Python language
https://swcarpentry.github.io/python-novice-inflammation/
http://swcarpentry.github.io/python-novice-gapminder/

Python for Analysis – how to analyze data in Python either with PyROOT or with the tools from Scikit-HEP

This training is aimed at those who are new to HEP and want a fast-track to competency with software fundamentals, as well the non-expert self-taught who wish to ensure they do not have gaps in their knowledge.

JLab Filesystems

FAQ/Reminders

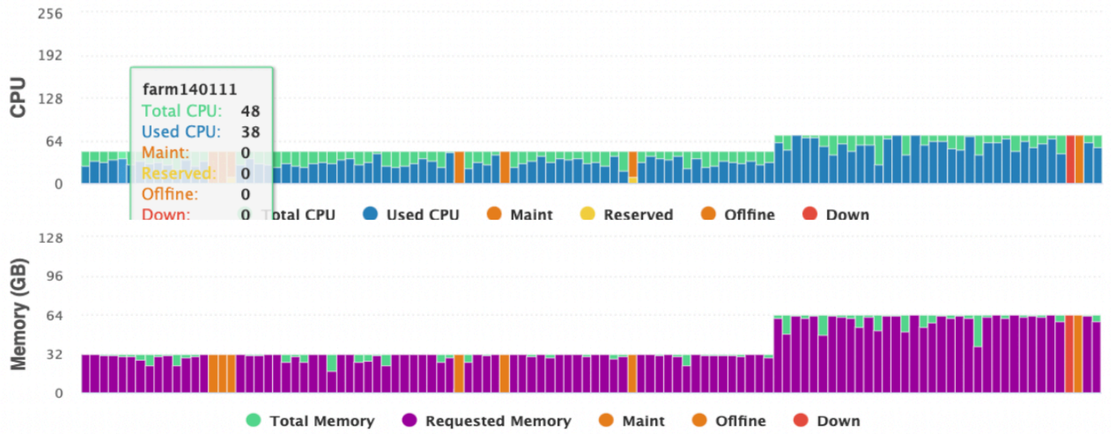
- scicomp.jlab.org !!!!
- `/work/hallb/hps` - ZFS - **25 TB**
    - one single, hard quota, manually managed, no automatic deletion
    - live snapshots at `/work/hallb/hps.zfs`
    - not ideal for large I/O from many batch jobs simultaneously
        - clas12 chefs do not use it for any data
        - the conventional place for collaborators to store their more "personal", somewhat permanent, physics data and large software
- `/volatile/hallb/hps` - Lustre - **30-190 TB**
    - one single, "soft" quota. with auto-deletion queue based on file modification times FIFO
        - generally a 2-month lifetime for CLAS12
        - if you copy something there and it gets deleted prematurely, you might have copied it with old timestamps, e.g., `rsync -p`
    - no backups of any kind
    - ideal for large I/O from many batch jobs
    - bad for lots of small IOPS, e.g., streaming log files, and tons of small files' metadata
- `/cache/hallb/hps` - Lustre - **70-140 TB**
    - Very similar to /volatile, but serves as a disk mirror of tape, and generally has a lifetime of many months
        - e.g. `jcache get /mss/clas12/...` writes data from tape to disk
    - Files written there are also guaranteed to be copied to tape automatically before automatic deletion, except files smaller than 1 MB!
    - The main filesystem used during large-scale processing for HPS data that should be stored permanently and used for publications
    - Data on tape should be associated to an experiment, not individual users, so we coordinate with the run group's analysis coordinator when we want to write more "personal" data to tape for permanent storage
- `/mss/hallb/hps` - pseudo-filesystem
    - just the metadata of all files on tape, cannot write to it directly
- `/farm_out/$USER` - SSD
    - few-GB per-user, hard quotas (can cause jobs to crash!)
    - good for small IOPS and the ***only*** place for log files from batch jobs
- `/home/$USER` - SSD
    - few-GB per-user, hard quotas, see cc.jlab.org for values
    - live snapshots at `.snapshot` at every level in the directory tree, plus regularly backed up to tape

There's a new, larger, faster /farm/home, available upon request

- `/group` - SSD
    - generally only written to by group coordinators, admins, for official software builds, databases, web interfaces, etc.
    - same snapshot/backup policy as `/home`

JLab Batch Jobs

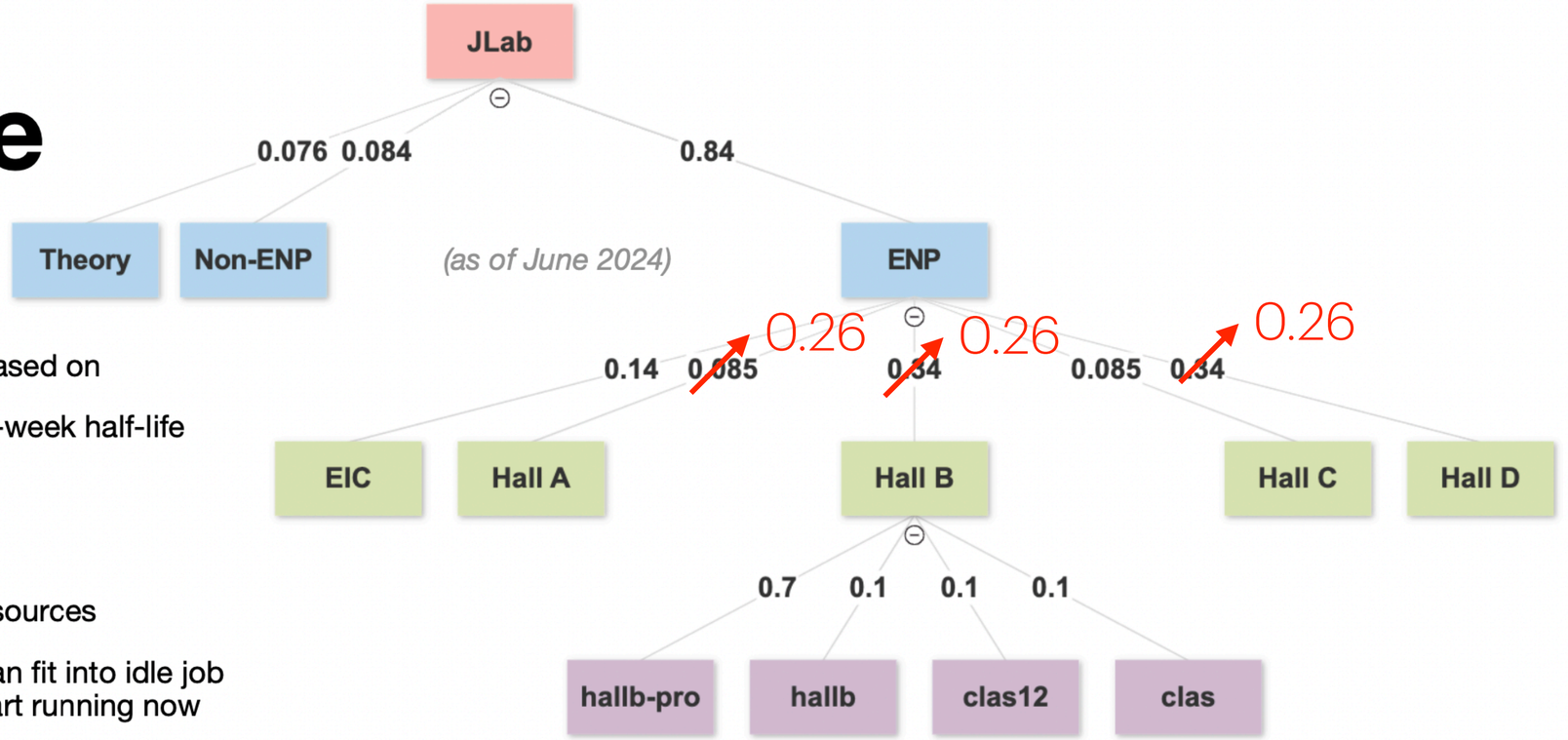
FAQ/Reminders



- scicomp.jlab.org !!!
- SLURM
    - The base job scheduler at JLab and an industry standard
    - Requires you being added to the `hallb` SLURM account in order to submit jobs
        - Normally happens when you get added to the corresponding UNIX groups, but sometimes it gets overlooked ...
- SWIF
    - A JLab frontend to SLURM, currently also supporting a couple remote sites
    - Also knows about the JLab tape library and can stage data to disk and automatically submit to SLURM when ready
        - If you need to read data off tape at JLab, you probably should use SWIF, especially if that data is large, many TB
        - Although we can "pin" optimized skims for physics analysis on `/cache`, so you don't have to read from tape
    - Also supports workflows of jobs with complex inter-dependencies, which we leverage in the workflows CLAS12 chefs use
- The main source of underutilization of the farm is dry queues and still sometimes memory requests/requirements
    - Make sure your requests are appropriate for your jobs' requirements!
    - Can run a few test jobs and see how much memory they used, or run them interactively and check with `htop`.

JLab Fairshare

Algorithm



- JLab uses SLURM's *Fair Tree Fairshare* algorithm
  - Decides whose jobs to run first, priority, based on
    - wall hours used in the past, with a ~1-week half-life
    - the fairshare targets in the tree
    - queued jobs' wall time requests
  - Aims for 100% utilization of computing resources
    - If there's any jobs in the queue that can fit into idle job slots, someone's jobs are going to start running now
    - If a group doesn't submit enough jobs to utilize their fairshare, it can get absorbed by the other groups at the same level in the tree.
  - No preemption, i.e., no jobs ever get killed due to priority, only due to misuse of resources (over memory, disk, time limit).

HPS 2019 is ~4 weeks at 100% of hallb-pro's fairshare, assuming our 10 Hz tracking and some rounding up

- Entire tree not shown above, e.g., at the bottom of each branch are all individual users/accounts, all with equal fairshare
  - Current fairshares and priorities are available to all via the `sshare` command (used to be in graphical form at scicomp.jlab.org, presumably will return)
- hallb-pro is used for large-scale processing of real data, by special accounts for all Hall B experiments
  - It has a net fairshare of 0.84*0.34*0.7 → 20% of the entire farm (currently!)

0.26 → 15%

To be offset somewhat by farm25 node purchases, en route …

CLAS12@JLab Fairshare

Calculations

- For a few years we've maintained calculations of the farm hardware distribution and benchmarks of CLAS12 software on each, stored on JLab's O365 and publicly accessible:
    - CLAS12 @ JLab Batch.xlsx
- Used to project how long it'll take to process each large data set on Jlab's farm
    - This works for steady-state running, when our queues are kept primed for days.
    - Sometimes you might hear other, much slower numbers, but those are probably for things like time for a few runs submitted for calibration, which includes large startup overhead if the farm is occupied (no preemption, waiting for other's jobs to finish).
- Also used for HPS by scaling, ~10 Hz (?)
- The "priority" portion of Hall B fairshare is currently ~50 million CPU hours per year, shared by all Hall B run groups.
- HPS is still small CPU compared to CLAS12

	Nodes			Farm				CLAS12 Node		CLAS12 Farm		
flavor	memory (GB)	slots	memory per slot (GB)	nodes	slots	node fraction	slot fraction	node rate (Hz)	slot event time (ms)	rate (kHz)	rate fraction	events per day (M)
farm16	62	72	0.86	38	2736	0.15	0.10	76	944	2.9	0.08	250
farm18	92	80	1.15	78	6240	0.32	0.22	93	859	7.3	0.20	628
farm19	256	128	2.00	106	13568	0.43	0.47	172	746	18.2	0.49	1571
farm23	256	256	1.00	24	6144	0.10	0.21	363	705	8.7	0.24	753
Average			1.49					151	781			
Sum				246	28688					37.1		3202
Hall B Fairshare					8290					10.7		925
Hall B Pro Fairshare					5803					7.5		648

Playground		
Billions of Events:		*16.0*
flavor	days	days @ Fairshare
farm16	63.9	221.2
farm18	25.5	88.2
farm19	10.2	35.2
farm23	21.2	73.5
Hall B Fairshare		17.3
Hall B Pro Fairshare		24.7

User Input Fields	
Run Group	Rate Scale
RG-B	0.90

Run Group	Rate Scale
RG-A	1.00
RG-B	0.85
RG-K	1.43
RG-F	1.13
No Roads RG-A	1.13
RG-M	1.18
RG-C	0.9
RG-D	
RG-K	
RG-E	
Scales are relative to RG-A	

Tree Fairshares		Million Slot-Hours per Year
ENP	0.84	211.1
Hall B	0.34	72.6
Hall B Pro	0.70	50.8
Product	0.202	

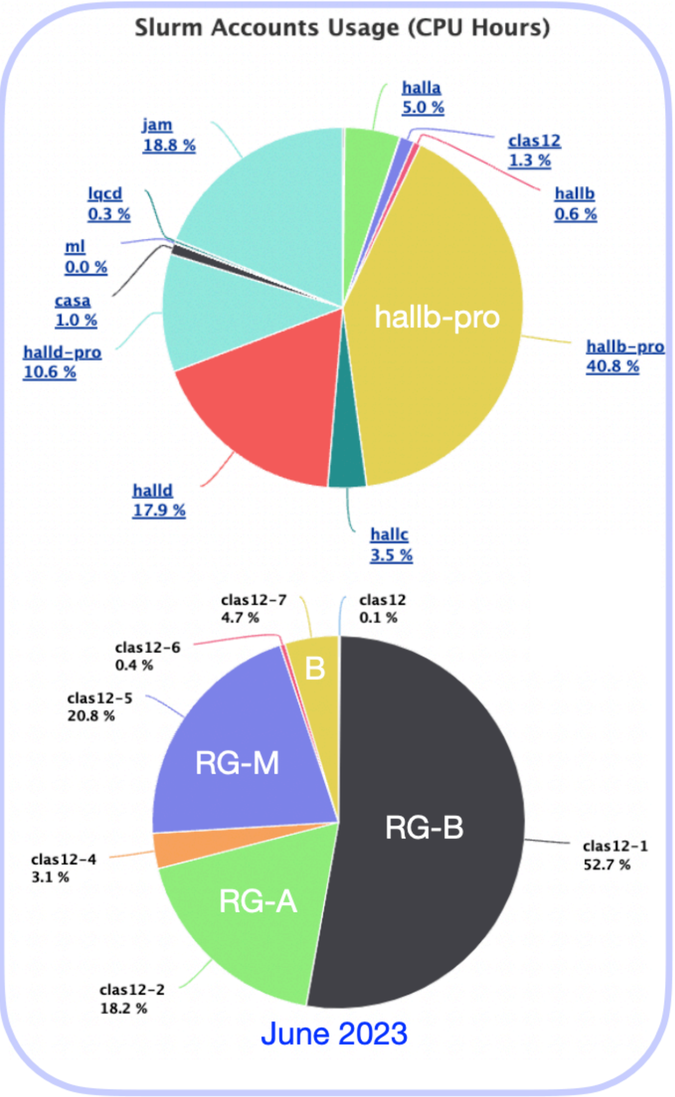
Global Scale Factor
0.85

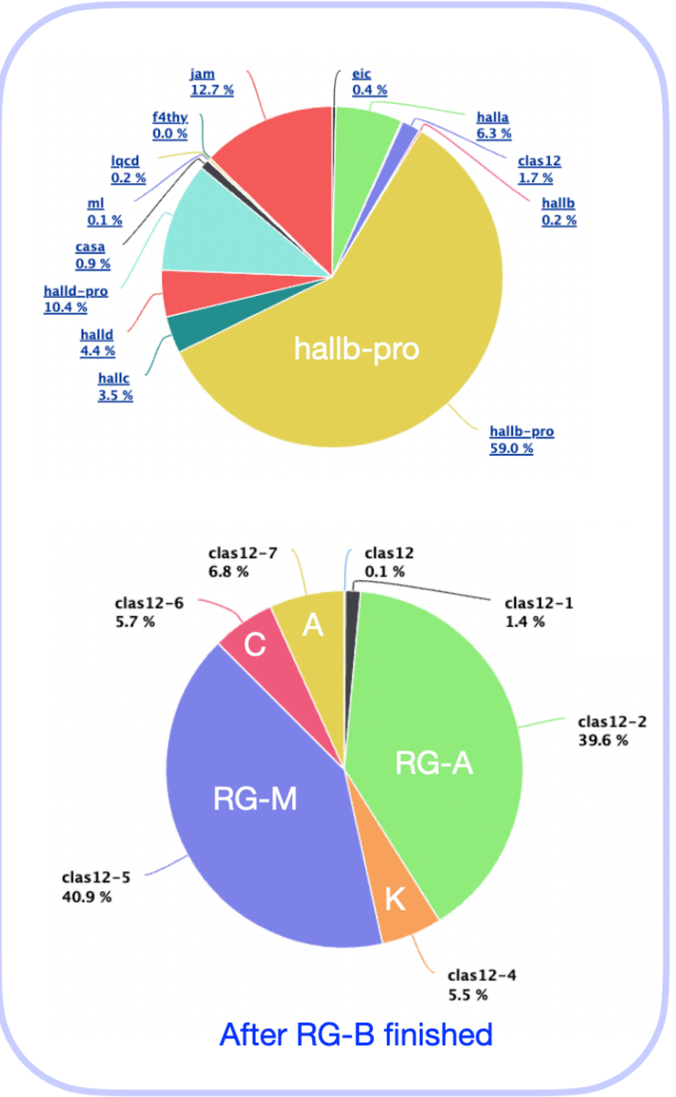
CLAS12@JLab

Reality

- RG-B's pass-2 ran for 30 days, start to finish
  - If we take the fraction of the farm clas12-1 really used during that time, and scale by the ratio of the expected hallb-pro fairshare of 20%, the 37-day projection from the previous slide turns into 30 days. 👍
  - The same assessment was done for RG-A/B's pass-1s, with similar agreement.
- You can also see RG-A/M's full passes coming online later in June, otherwise clas12-1's 53% would have been a much higher fraction of Hall-B's June usage, and then RG-A/M getting equal shares after RG-B finished.
- Meanwhile, calibration jobs are coming in from other run groups, and due to their much lower recent usage, immediately take priority over the full passes.
- So far, we have not (and do not want to!) manually managed any fairshares or job submissions to prioritize run groups or calibrations. All have equal fairshare.

Account Name	Current Purpose
clas12	admin
clas12-1	RG-B
clas12-2	RG-A
clas12-3	RG-F
clas12-4	RG-K
clas12-5	RG-M
clas12-6	RG-C
clas12-7	parallel calibrations
clas12-8	RG-D
clas12-9	RG-E
hps	HPS





Open Science Grid

- CLAS12 runs 99% of simulations on OSG, ~doubling available CPU resources relative to JLab's farm
    - Various machinery in place to automate for the collaboration's independent users, but HPS doesn't need all that
    - Timelines: https://clasweb.jlab.org/clas12offline/osg/
    - Submission portal: https://gemc.jlab.org/web_interface/index.php
- JLab has one submit node per ~hall, and submissions must go through HTCondor (not SWIF/SLURM) locally on that particular node
- Currently no mechanism other than HTCondor's built-in to get output data back from the job, and the submit nodes are DMZ'd, so it's staged locally on the submit node, and transferred manually to networked filesystems at JLab before manual deletion to ensure the local staging filesystem doesn't fill up.
- Large-scale stuff should be centrally managed by HPS, i.e., only one person submitting/managing jobs at a time.
- This writeup was distributed a couple years ago with some preparatory information and questions (data volume, job independence/atomicity, CPU hours) for HPS simulations on OSG
    - HPS @ OSG.docx

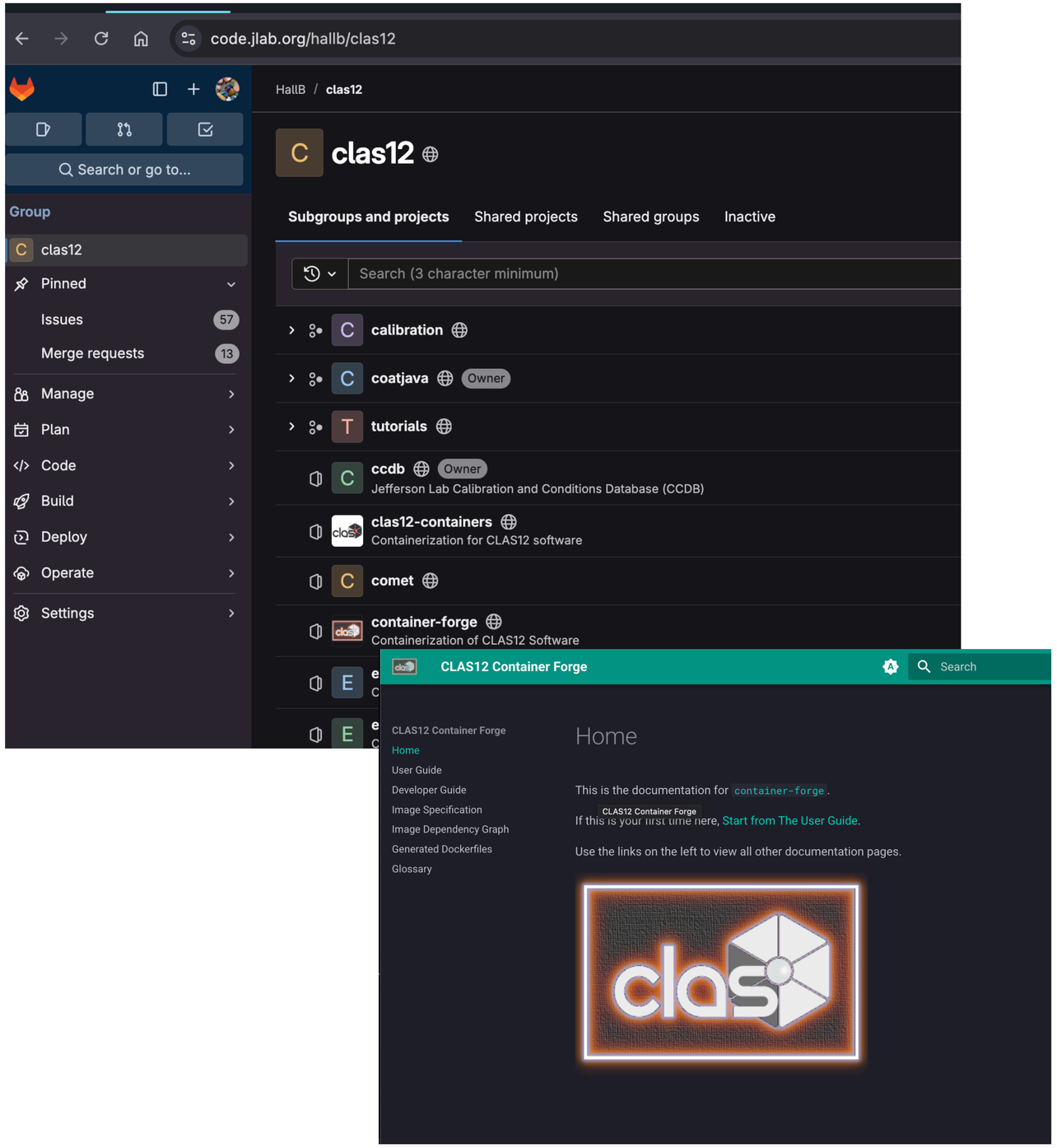
Pelican

JLab supports CVMFS and XRootD

With Pelican, no more manual data transfers ...

GitLab

- In heavy use for ~6 months
- Continuous Integration, Container Registry, "Pages"
    - lots of tools to streamline and save time on software workflows, building, deploying, releases, etc.
- JLab got it because they're grandfathered-in GitHub plan is too limiting, and upgrading is too expensive



Summary

- Checkout new JLab on-boarding material, other open software groups in the HEP/NP community
- JLab farm ready for HPS '21 data processing, see upcoming talks for details
- JLab is updating their Data Management Plan this year
    - The last time was a decade ago, and this one is probably going to be much more thorough …
- New at JLab
    - GitLab server code.jlab.org
    - RUCIO
        - Modern data catalog, discovery, transfer from a web browser
        - In use by EIC, next GlueX
    - Various system improvements, read-only cache, swif2, network/filesystem performance, regular maintenance days, see scicomp.jlab.org mailing list
    - Next routine purchase of farm25 nodes en route
    - Various changes / growing pains with modernizing webservers, databases, network