Introduction to Fermilab computing

Miquel Nebot-Guinot (mnebot@fnal.gov)

6th UK LArTPC Software Analysis Workshop

1st November 2021

Outline

1. Introduction.

2. Where to get help.

3. Accessing Fermilab computing.

4. Fermilab Computing philosophy:
   UPS, cmake, MRB, ART, LArSoft

5. Running LArSoft:
   Storage, Grid, Containers

Groundwork

- This talk aims to serve as an introduction to the LArSoft environment, the FNAL computing context, the underlying machinery and some tools you'll use for running/developing work.
    - More on LArSoft in the tutorials.
- Steep learning curve, but we're here to help. Don't let the feeling of "*Can't ask such trivial thing*" to stop you from learning. We've all been there.
- Lots of acronyms, Sorry!
  Lots of info/material in the slides as useful future reference.

DISCLAIMER:
This lecture is heavily inspired by previous workshops/ tutorials/schools by Andrzej, Pierre, Iker, Erika, Tom Junk, Keneth Herner…

Don't be scared Halloween just passed

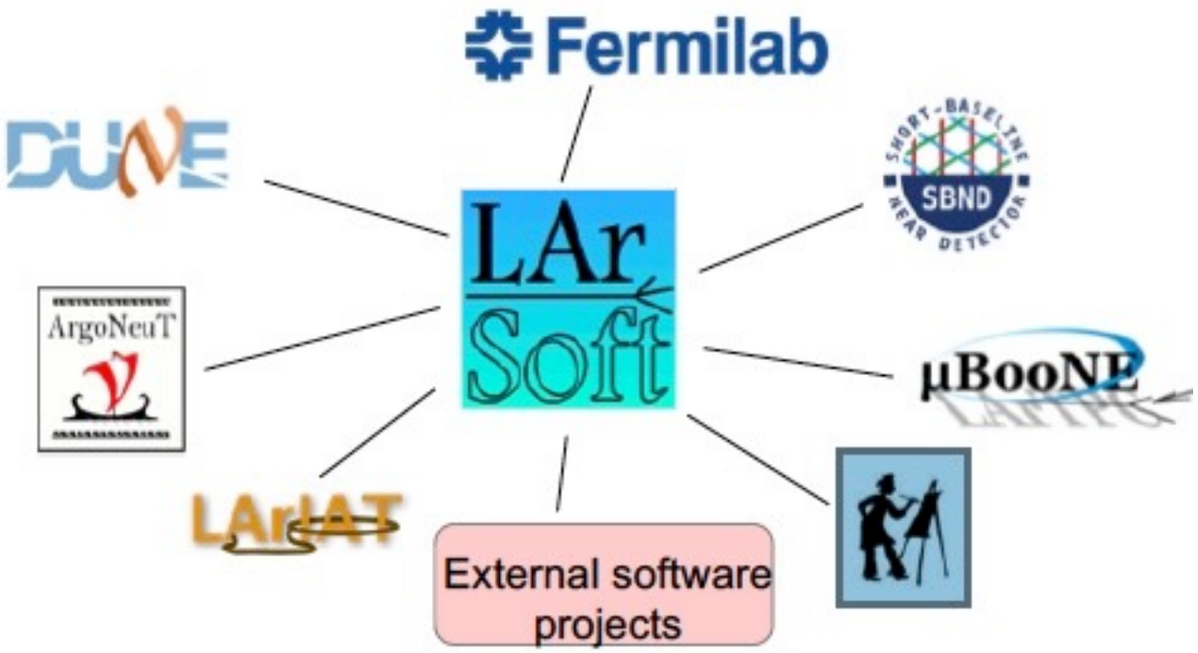
Command line, file …

Basic, not recommended.
Good.
Best 😉

link Links for more info

LArSoft (preamble)

- Liquid Argon Software framework used by several experiments. https://larsoft.org



- Where to run LArSoft?
- at Fermilab → get a FNAL account:
  https://microboone-exp.fnal.gov/at_work/start.html#Software
  https://sbnsoftware.github.io/sbndcode_wiki/Newbie_Material.html
  https://wiki.dunescience.org/wiki/DUNE_Computing/Getting_Started_with_DUNE_Computing
- At CERN → https://indico.fnal.gov/event/16218/contribution/2/material/slides/0.pdf
- On your installation →
  https://indico.hep.manchester.ac.uk/getFile.py/access?sessionId=26&resId=0&materialId=0&confId=5346
- Here → https://phcomputeppe01.ph.ed.ac.uk/guacamole

Wikis and general info

- First and foremost:
  - http://larsoft.org/training/
- You can also have a look here for LArSoft information:
  - https://cdcvs.fnal.gov/redmine/projects/larsoft/wiki
- And here for the experiment-specific information: (login)
  - DUNE: https://cdcvs.fnal.gov/redmine/projects/dunetpc/wiki
    https://wiki.dunescience.org/
    SBND: https://sbnsoftware.github.io/sbndcode_wiki/Wiki (public)
    microBooNE: https://cdcvs.fnal.gov/redmine/projects/uboonecode/wiki(login)

- Another link that I find very useful:
  - https://wiki.dunescience.org/wiki/DUNE_Computing/List_of_DUNE_Tutorials,_LArSoft_Workshops,_etc,_etc,_etc
- You can edit these pages (especially the experiment ones), once you have signed in Redmine/GitHub, and have been added to the user group.
  - PLEASE: If you find something isn't clear/wrong, make sure you change it for the next ones (or even for your future references)!!

List and slack

- You can also get help with through mailing list:
  - larsoft@fnal.gov -larsoft@fnal.gov SBND-SOFTWARE@fnal.gov microboone_analysis_tools@fnal.gov lariatsoft@fnal.gov dune-reco@fnal.gov dune-proto-sp-dra@fnal.gov
  - To subscribe:
    - Email to listserv@fnal.gov
    - No subject
    - And add in the body:
      - subscribe <list> name lastname
  - https://listserv.fnal.gov manage your list, subscriber's corner

Be wise choosing your subscription email

Slack: http://slack.com/signin
dunescience.slack.com
sbnd.slack.com
lariat-t1034.slack.com
shortbaseline.slack.com (SBN programme)
microboone.slack.com

- Interesting channels:

  #larsoft, #larsoft_beginners #<exp>_young, #newbie_questions …

Good practice:
—Check for the answer, someone may have already solved it.
–Make it easy for people to help you.
e.g. create "Minimal, Complete, and Verifiable" example: http://stackoverflow.com/help/mcve .
–This allows experts to reproduce your problem and find the fix quickly, as contrary to: "my code does not compile". *What code? Where? What version?*
–Often, you will find the solution yourself in the process. ;-)
–If you haven't spent the time to understand your problem – why should the experts?

Where to start: First steps

1. Fermilab computer account, kerberos password and services account:
   https://get-connected.fnal.gov/accessandbadging/access/

2. Request experiment/collaboration specific computing accounts:
   https://fermi.servicenowservices.com/wp/?id=evg_sc_cat_item&sys_id=d361073881218500bea3634b5c987c4c

Access to a remote server authenticating as yourself by providing a Kerberos 5 ticket

~

```
$ kinit username@FNAL.GOV
$ ssh username@sbndgpvm01.fnal.gov -XY
```

Make your ~/.ssh/config like this:

```
Host *.fnal.gov
  # User   username              # Fermilab user name (allows something to ssh like `ssh sbndgpvm01.fnal.gov`)
  ForwardAgent yes
  ForwardX11 yes                # Establish an X11 connection to get graphics on your laptop (via X servers like X11 or XQuartz)
  ForwardX11Trusted yes
  GSSAPIAuthentication yes.  # Enable authentication via Kerberos 5
  GSSAPIDelegateCredentials yes. # Forward Kerberos 5 credentials
  LocalForward 5901 localhost:59XX  # VNC setup (XX change for your VNC server number)
  ConnectTimeout    60          # bail out if no answer for one minute
```

VNC

login using your Kerberos password

Ssh into your selected server

```
$ kinit -Af -r 7d <your-kerberos-principal>@FNAL.GOV
$ ssh -K -XY <your-username>@sbndgpvm0N.fnal.gov
```

!! case matters for FNAL.GOV

kinit

-f (forwardable)
-p (proxiable)
-A (not restricted by address)
-r (renewable within [life])
-l with lifetime [lifetime])

-K tells SSH to forward the Kerberos credentials
-X Enables X11 forwarding
-Y Enables trusted X11 forwarding

ssh

```
kinit -R username@FNAL.GOV || kinit -Af -l 26h -r 7d username@FNAL.GOV
```

tmux

Fermilab Computing philosophy

- Complex, multi-level systems, all relying on each other:
  A. **UPS** → setting the right dependancies.
  B. **CMake** → compiling
  C. **MRB** → version control (= GIT)
  D. **ART** → the underlying structure for LArSoft (process events)
  E. **LArSoft / <experiment>code** → what is actually interesting you (where physics, simulation and analysis happen)
- You need to know a bit of all these to be able to develop efficiently.

Fermilab Computing philosophy

A: UPS

- To run program you need libraries.
- This is code you can reuse (like the underlying code for an `std::vector`)
- These are already compiled, gets linked at runtime
- Very sensitive against: the machine, the version of the code

```
$ ls -lh /usr/lib
total 97712
-rwxr-xr-x   1 root  wheel   568K 21 Sep 05:16 libATCommandStudioDynamic.dylib*
-rwxr-xr-x   1 root  wheel   218K 21 Sep 05:17 libAccessibility.dylib*
-rwxr-xr-x   1 root  wheel    23K 21 Sep 05:16 libAccountPolicyTranslation.dylib*
-rwxr-xr-x   1 root  wheel    44K 21 Sep 05:17 libAppleSSEExt.dylib*
-rwxr-xr-x   1 root  wheel   128K 21 Sep 05:16 libAppletTranslationLibrary.dylib*
-rwxr-xr-x   1 root  wheel   935K 21 Sep 05:16 libAudioIssueDetector.dylib*
-rwxr-xr-x   1 root  wheel   168K 21 Sep 05:17 libAudioStatistics.dylib*
-rwxr-xr-x   1 root  wheel    79K 21 Sep 05:16 libBSDPClient.A.dylib*
```

- UPS (Unix Product Support) is a system developed at Fermilab in late 1990s, that allows you to run code that depends on different versions of libraries, now called PRODUCTS, on the same machine.

UPS

- UPS is taking care of linking the correct libraries together with the correct version of the code you are trying to run.
- UPS changes automatically some environment variables (the ones you get when you do `$ env`).
  - Everything that gets setup is in the environment variable `$PRODUCTS`
  - You can try to do `echo $PRODUCTS`
- Where products live:
  - `/cvmfs/`fermilab.opensciencegrid.org/products/larsoft/
  - `/cvmfs/`uboone.opensciencegrid.org/products/
  - `/cvmfs/`sbnd.opensciencegrid.org/products/
  - `/cvmfs/`dunetpc.opensciencegrid.org/products/
  - Usually `source <path to product directory>/setup` will make this set of products available to you.

Discontinued by Fermilab
Used almost exclusively by Fermilab-hosted experiments
Spack
(package manager for supercomputers)

cvmfs

A: UPS: crash course

- Listing what software is available:

```
$ ups list -aK+ <software> <version>
```
or
```
$ ups list -aK+ <software>
```

- Listing what you have already asked ups to use:

```
$ ups active
```

- Listing the dependencies:

```
$ ups depend <software> <version> -q <qualifiers>
```
<qualifier> (e.g. e20:prof) is a qualifier that specifies the compiler version

- To setup a software:

```
$ setup <software> <version> -q <qualifiers>
```

- To do the inverse of above:

```
$ unsetup <software>
```

```
# setup the dunetpc environment
source /cvmfs/dune.opensciencegrid.org/products/dune/setup_dune.sh

# cvmfs access can be slow to start as the cache needs to be filled
# see what versions of dunetpc there are
ups list -aK+ dunetpc

# setup a recent one on the list (pick a more recent version if there is one):

setup -B dunetpc v09_34_00 -q e20:prof
# From here you can use dunetpc!!!
```

Fermilab Computing philosophy

B: CMake



- Different ways to compile you code:
  - OPTION 1: Command line

```
$ gcc yourfile.C -o Executable.exe
```

  - OPTION 2: Makefile which contains the above line.

```
$ make
```

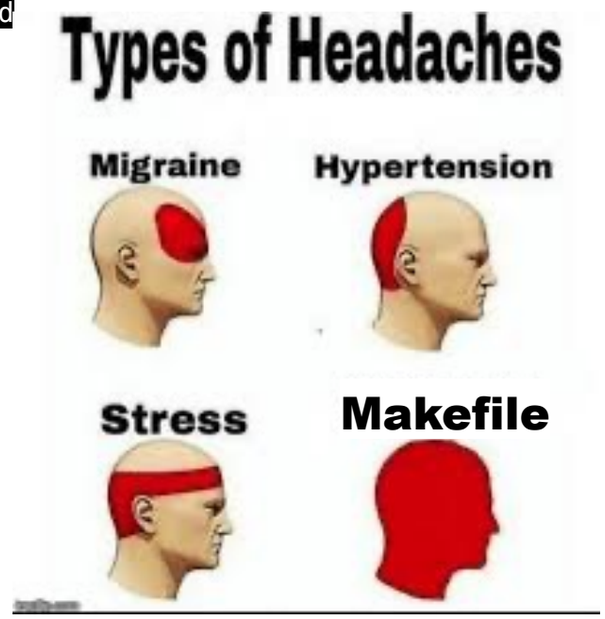
  - OPTION 3: CMakeList.txt which contains the information to generate the Makefile.

```
$ cmake <path where CMakeList.txt is>
$ make
```

- Why do we do this? Say you want to link ROOT to your executable
  - OPTION 1:

```
$ gcc yourfile.C -o Executable.exe -I/data/root/include -L$/data/root/lib -lCore -lCint -lGraf -lGraf3d -lHist -lHtml -lMatrix
-lMinuit -lPostscript -lProof -lTree -lGpad -lGui -lGX11 -lRint -L/usr/lib/X11R5 -lXpm  -lX11 -lm -ldld
```

  - OPTION 2: That line would be in the makefile
- Now you change the version of ROOT, or change its location. OPTION2 fails.
- You want to have something that generate the Makefile for you.
  - **CMake!!** It is a “meta-make”, i.e. it looks at your system configuration and creates its makefiles depending on your system configuration. Helps if you have multiple repositories (as we do).
    - Very simple way of writing very complicated Makefile, do learn how to use it, it will save you time.
      https://cmake.org/cmake-tutorial/
      https://root.cern.ch/how/integrate-root-my-project-cmake

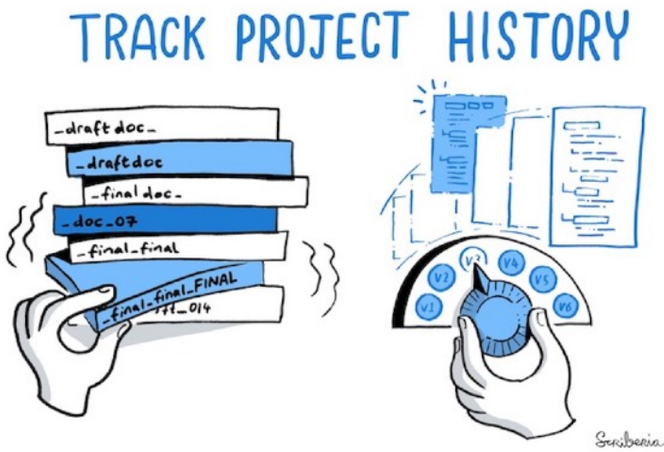


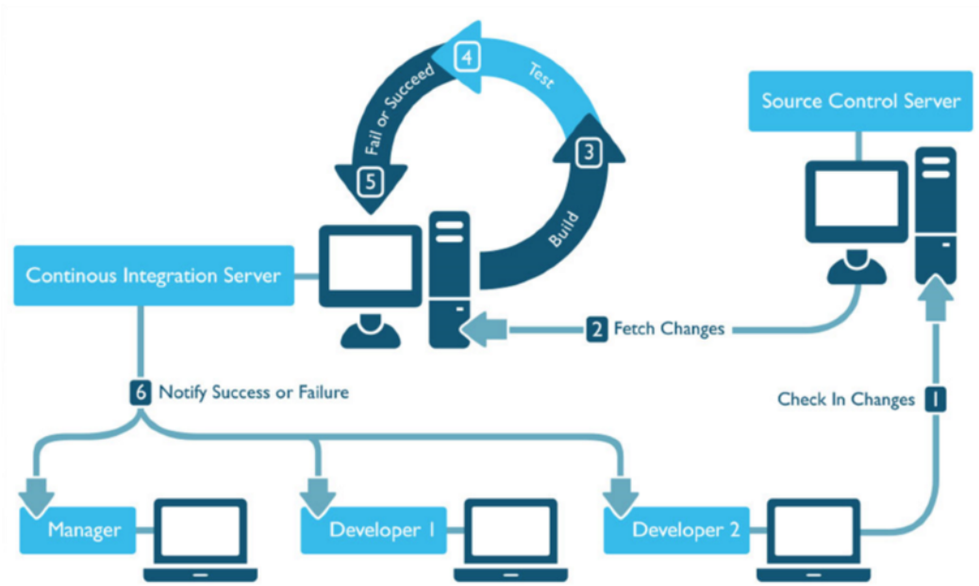
Fermilab Computing philosophy

C: MRB: Version control

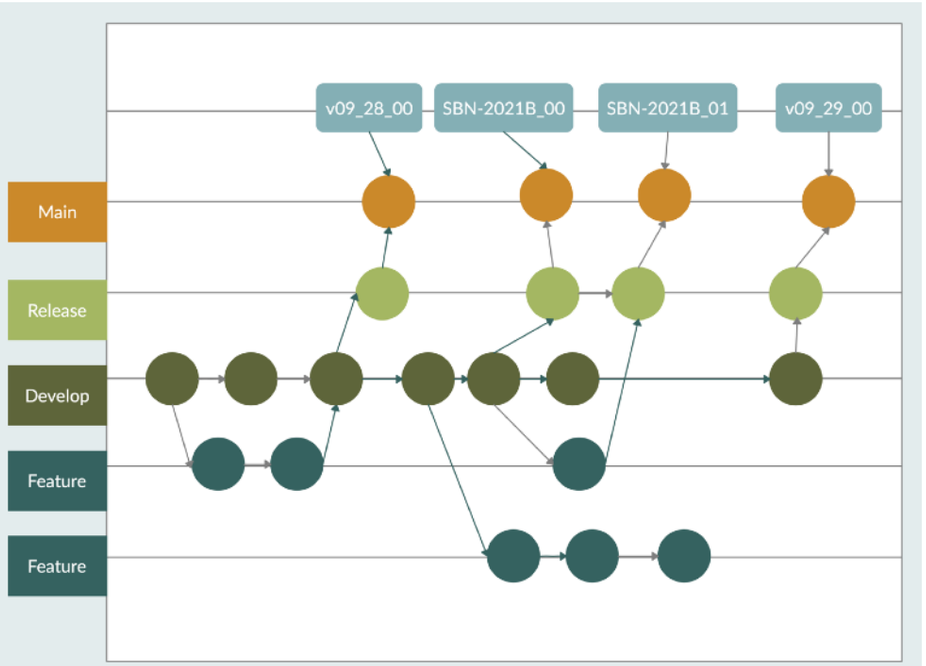
- You may be already familiar (or should be) with some kind of version control for your documents (reports, thesis …)
- Version control is a must for any sort of collaborative work which involves coding.
- Essentially, Git tracks the changes of the code (and allow you to revert).
- Big advantages: scales very well with code size and number of people.
- Hard to understand the logic. At the beginning, everyone makes mistakes
- Easy way to learn:

https://www.coursera.org/learn/version-control-with-git
https://lab.github.com/githubtraining/introduction-to-github





C: MRB - Git: Crash course



Git: an opensource, distributed version-control system

- GitHub: a platform for hosting and collaborating on Git repositories

- Branch structure, every code change that you make is assigned to a branch.
- **Master/Main:** The version that is tagged that you can get from UPS
- **Hotfix:** If ever there was a big problem in master that needed to be sorted quickly
- **Develop:** The most up-to-date version of the code. Develop becomes master at the time of a release.
- **Feature:** What you or somebody else are working on that may be integrated in develop at some point. Do use `feature/<username>_name`

Some “most used” git commands:
git branch -a –List available feature branches

git checkout feature/name –Check out a branch to work on.

git add <files> –Tell git to track selected files.

git status –See which files are tracked and which are not

git commit <files> -m “commit message” –Commit the tracked files to your local repository

git push origin –Push to the main repository for everyone to see.

[Git cheat sheet]

Fermilab Computing philosophy

C: MRB

- LArSoft itself currently resides in multiple repositories.
- MRB (multi repository build system) uses GIT and UPS to keep track of the dependencies and make sure you're good to go.
  - Often it will tell you that you're not.

```
mrbsetenv
----------- check this block for errors ------------------------
Error encountered when setting up product: larsoft
ERROR: Version conflict -- dependency tree requires versions conflicting with current
setup of product larsoft: version v09_35_00 vs v09_34_00
ERROR: setup -B larsoft v09_35_00 -q +e17:+prof failed
ERROR: For more information, type "ups depend larsoft v09_35_00 -q +e17:+prof"
                                or "ups list -aK+ larsoft v09_35_00 "
ERROR: setup of required products has failed
```

Most commonly used commands

command	short hand	arguments	description	(opinionated) comments
mrb newDev	mrb n	-v $Version -q $Qualifier	Start a new development area	Use this when updating version
mrb gitCheckout	mrb g	sbndcode	Clone a git repository	You could clone with git clone too, but don't
mrb install	mrb i	-j $numcores mrb i --generator=ninja	Run buildtool with install	Compile and move/update code
mrb zapDist	mrb zd		Delete everything in both your build and localProducts areas	zapBuild and zapInstall also exist. IMO better to remove everything
mrb updateDepsCM	mrb uc		Update the main CMakeLists.txt file	To be used after manually editing CMakeLists.txt
mrb test	mrb t		Run buildtool with tests	Run this when you want to modify base code… software manager will ask you about it
mrbsetenv			Setup a development environment: source $MRB_DIR/bin/mrbSetEnv	Always* use this before building code
mrbslp			Setup all products installed in the working localProducts_XXX directory: source $MRB_DIR/bin/setup_local_products	Always* use this before running custom code

mrb

Fermilab Computing philosophy

C: Day at work

Checkout the develop branch of the code:

```
$ cd $MRB_SRCS
$ mrb g dunetpc # this is where the git checkout happens
```

Create your branch

```
$ cd dunetpc/dune
$ git flow feature start username_myAwesomeFeature
# This will create a branch feature/username_myFeature locally
```

Edit / Modify

```
$ emacs -nw DAQSimAna/trigprim.cc
```

Track your changes

```
$ git add DAQSimAna/trigprim.cc
```

Commit

```
$ git commit -m "I modified trigprim and it now does that."
```

Finished your work?

```
$ git flow feature finish
# this will run the equivalent of:
# $ git checkout develop
# $ git merge feature/username_myFeature
```

Want to modify code from collaborator?

```
$ git fetch
$ git rebase feature/collaborator_ThatFeature
```

Want to share with collaborators?

```
$ git flow feature publish
# equivalent to $ git push origin
```

!! `$ git push origin develop` WARNING if you break someone else's code, get authorisation first!! (see next)

Fermilab Computing philosophy

C: Day at w

Checkout the develop branch of the cod

```
$ cd $MRB_SRCS
$ mrb g dunetpc # this is where the gi
```

Create your branch

```
$ cd dunetpc/dune
$ git flow feature start username_myAw
# This will create a branch feature/us
```

Edit / Modify

```
$ emacs -nw DAQSimAna/trigprim.cc
```

Track your changes

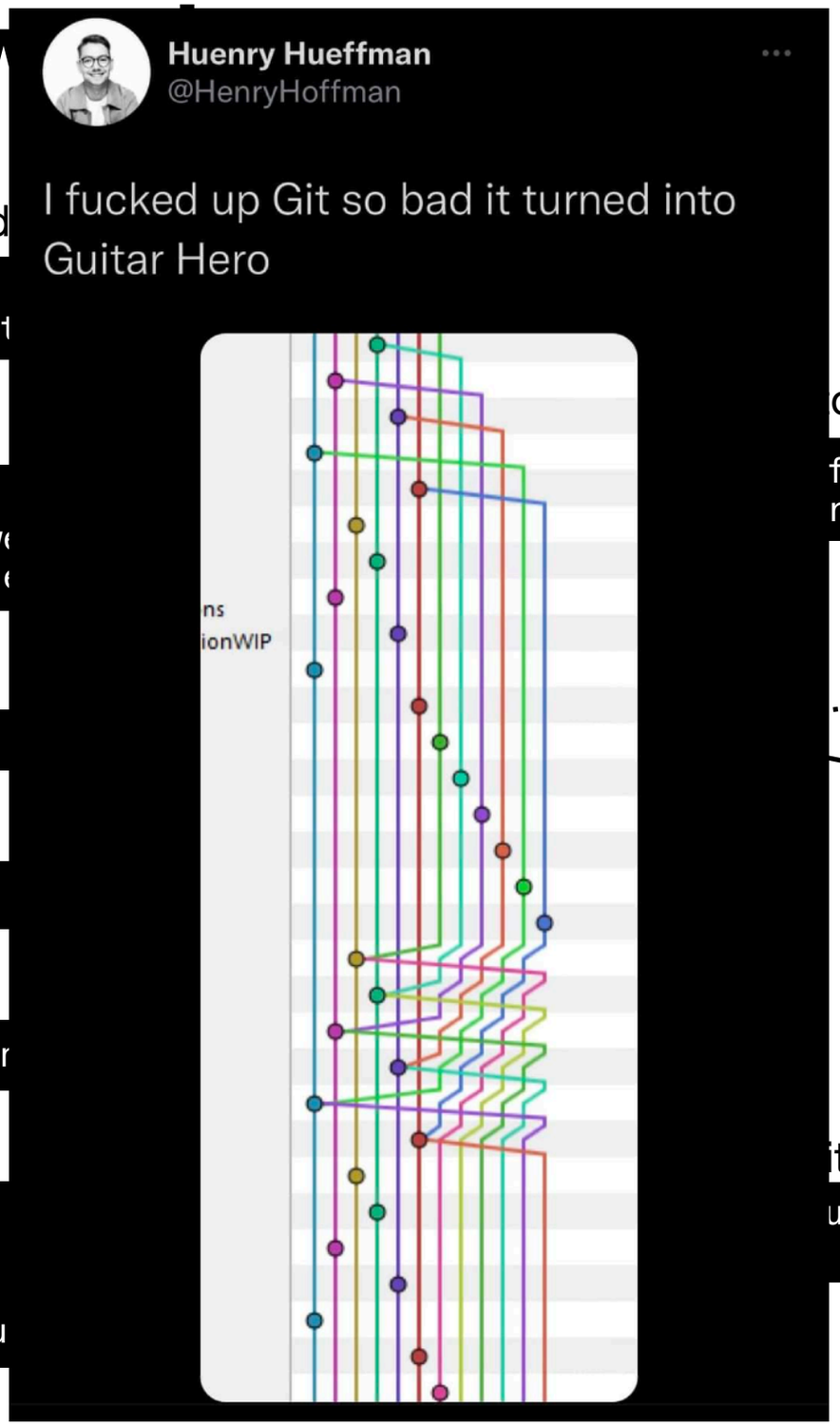
```
$ git add DAQSimAna/trigprim.cc
```

Commit

```
$ git commit -m “I modified trigprim ar
```

Finished your work?

```
$ git flow feature finish
# this will run the equivalent of:
# $ git checkout develop
# $ git merge feature/username_myFeatu
```



o modify code from collaborator?

```
fetch
rebase feature/collaborator_ThatFeature
```

ith collaborators?

```
ure publish
 $ git push origin
```

!! `$ git push origin develop` WARNING if you break someone else’s code, get authorisation first!! (see next)

C: GitHub

- LArSoft , SBND, ICARUS, some parts of DUNE have moved their repositories to GitHub.
  - Merging into develop happens after approval via a “pull request” (PR).
- You need to have a github.com account: https://github.com/join
- On the machine you’ll be working on, let git know what your repo is:
  - git config --global user.name "<First Name> <Last Name>"
  - git config --global user.email <Your-Email-Address>
  - git config --global user.github <Your-GitHub-Account-Username>

LArSoft

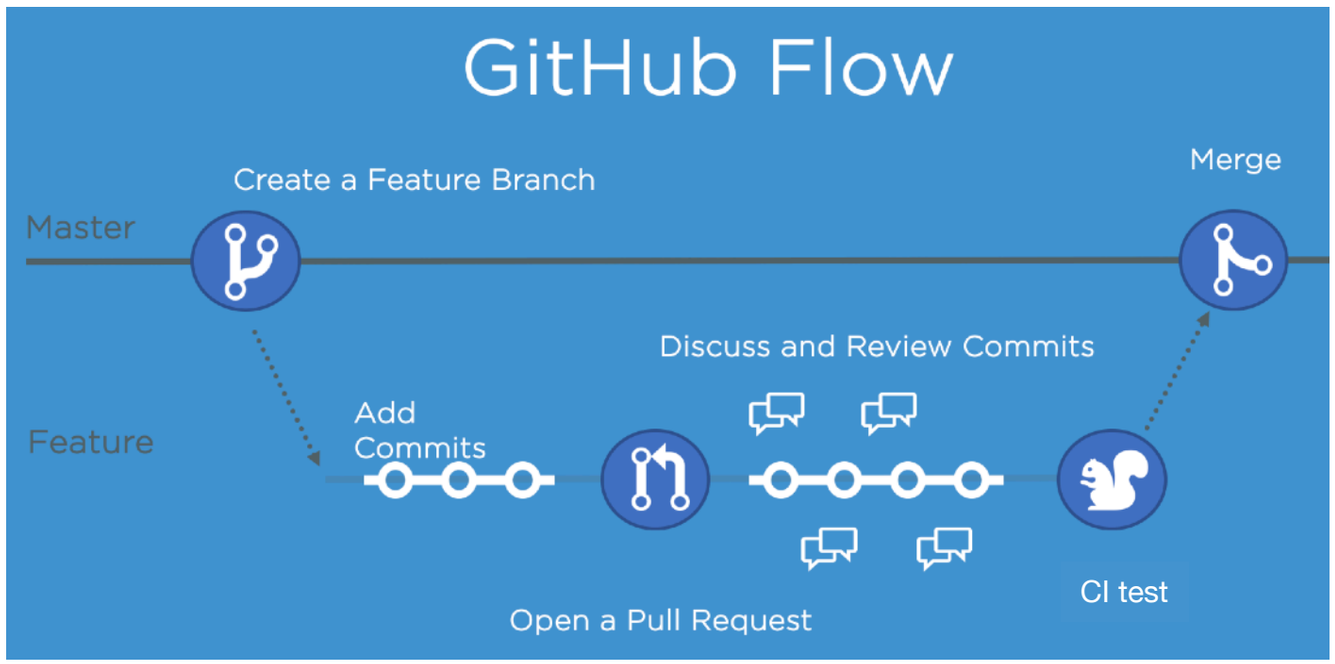
SBN

DUNE-DAQ

Still in redmine (no PR)

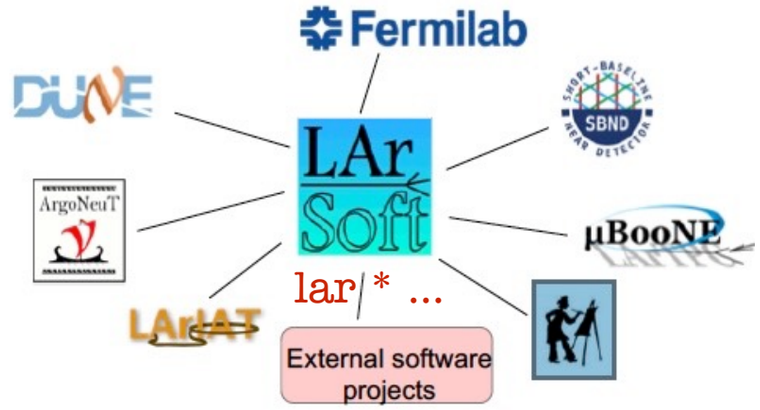
dunetpc

- You can also set up your ssh key on your machine to make check-ins easier: https://help.github.com/articles/generating-ssh-keys

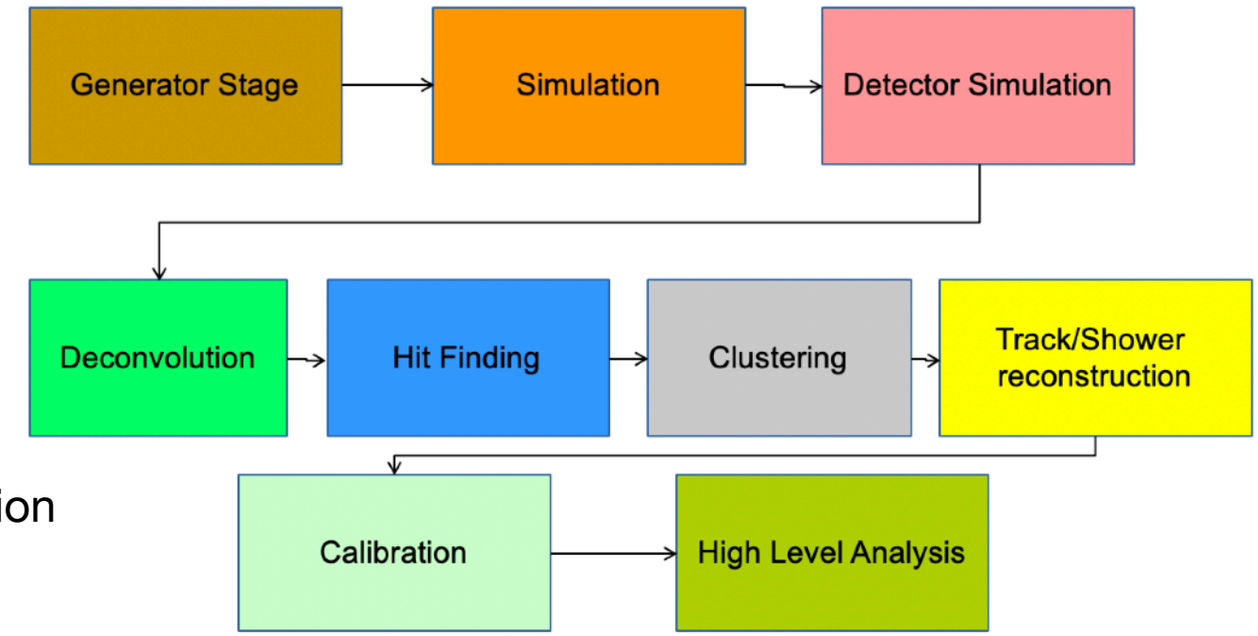


D: LArSoft (ART)

Liquid Argon Software framework used by several experiments.
https://larsoft.org



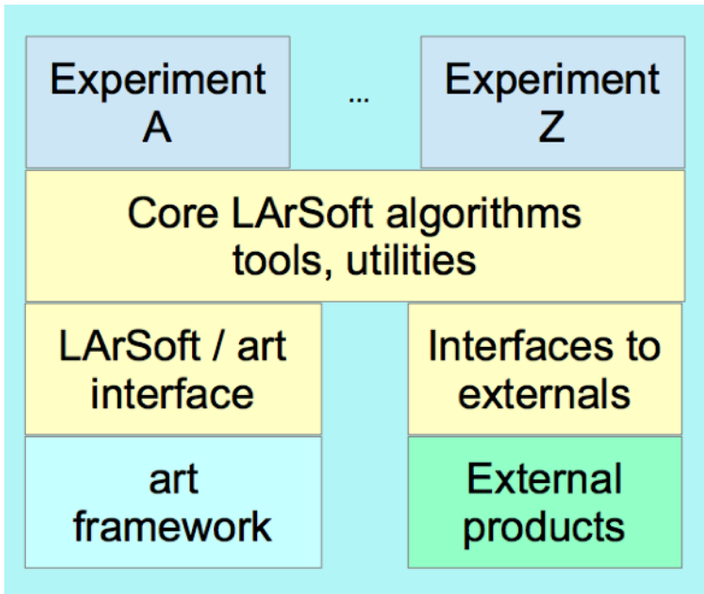
- LArSoft is a software suite that is very versatile:
  - can run multiple simulation and reconstruction algorithms
  - on different experiments and detectors.
  - Can run in stages.
  - Parameters can be changed through configuration files.
- It has its own data structure/related input output/ configuration file language.
- Code that it uses needs to be structured in pre-defined ways (modules) to work.



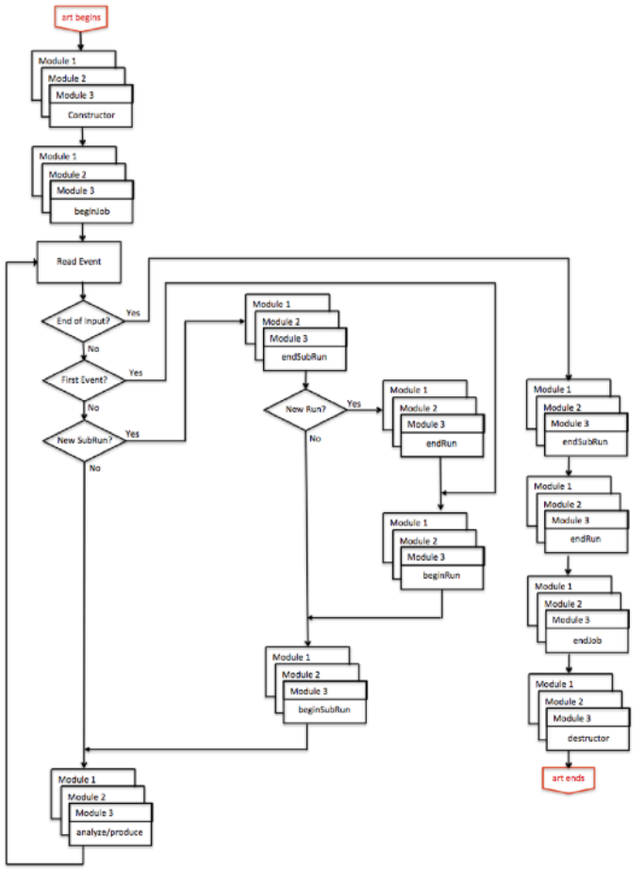
Each stage is a module or more.
Each stage passes data products, “objects”, to the next stage.

D: Art (concepts)

- LArSoft is built on top of the art event processing framework
- The art framework:
  - Reads events from user-specified input sources
  - Invokes user-specified modules to perform reconstruction, simulation, analysis, event-filtering tasks
  - May write results to one or more output files `Modules`
  - Configurable, dynamically loaded, user-written units with entry points called at specific times within the event loop → **You can reprocess an event and recreate data product.**



https://web.fnal.gov/project/ArtDoc/Pag es/workbook.aspx



D: ART (clases)

- Modules (Three types)
    - `Producer`: add data product to an event
    - `Filter`: filter events
    - `Analyzer`: read information from an event and retrieve data product (no addition of data product to an event)
    - Conclusion: **You cannot change a data product in an event: What is there stays there!!**
- Services:
    - Configurable global utilities registered with framework, with entry points to event loop transitions and whose methods may be accessed within modules
- Tools:
    - Configurable, local utilities callable inside modules
- .fcl files
    - The run-time configuration of art, modules, services and tools specified in FHiCL
    - Fermilab Hierarchical Configuration Language, allows configuring jobs and modules on the fly.
    - See art workbook and FHiCL quick-start guide for more information on using FHiCL to configure art jobs
    - See https://cdcvs.fnal.gov/redmine/projects/fhicl-cpp/wiki/Wiki for C++ bindings and using FHiCL parameters inside programs

Fermilab Computing philosophy

D: FHICL (.fcl)

- Fermilab Hierarchical Configuration Language, allows configuring jobs and modules on the fly.
- There are two types of files (mostly by convention): header/include files, and job files. Both end with .fcl

- How .fcls looks like

```
#include "fcl/minimalMessageService.fcl"

process_name : hello

source : {
    module_type : RootInput
    fileNames   : [ "inputFiles/input01.art" ]
  }

services : {
    message : @local::default_message
  }

physics :{
    analyzers: {
      hi : {
        module_type : HelloWorld
      }
  }

  e1         : [ hi ]
  end_paths : [ e1 ]
}
```

A series of definitions `name : value` form a FHiCL table or a para meter set (all the goes between {}).

The name of the module to be loaded/run and files.

The parameter set in analyzers (or filters or producers) defines the run-time configuration for all of the modules that are part of the job

D: ART Rules

- **Thou shalt not modify data products (objects) "on" the event**
  You may add things to the event, but once you have done so, they may not be changed
- **Thou shalt not have modules depend on other modules**
  Developed a really cool function in your track-finder module you want to use somewhere else? Too bad.
- **Thou shalt only have modules interact with each other via the event**
  Run through modules linearly, and always get previous results via the event

Fermilab Computing philosophy

E: LArSoft/<experiment>code

- **LArSoft** is a body of code with specific product for each experiment user.
  - **dunetpc, sbndcode, uboonecode …** are experiment software built using LArSoft/art.
    A release (and UPS product) is bound to a particular release of LArSoft.
- Once setup you experiment specific software you can use its modules and fcl to run LArSoft job tipping

```
lar -c config-file.fcl <other-options> [<source-file>]+
```

most common options

- -c The argument to -c is the *run-time configuration file* , a text file that tells one run of *art* what it should do.
- -s Source data file (multiple OK) or -S file containing a list of source files to read, one per line
- -n Number of events to process.
- -T File name for TFileService (name your histograms file)
- -o Event output stream file (different options for multiple files)
- --nthreads Number of threads to use for event processing
- --timing Activate monitoring of time spent per event/module.

Can be defined inside the fcl

Many more with `lar -h`

E: LArSoft day to day

- Most days you will be running larsoft jobs combining producer, analyzer and filter modules.
- You will configure what modules actually get run using a .fcl file.
- You will also configure these modules using the .fcl file (which detector, its conditions etc…)
- The modules may produce LarSoft objects/data products and pass them on to the next ones in the chain.
- At the end you'll need an analyzer module that either makes plots directly, or produces a TTree object (or analogous)

Fermilab Storage

- You'll run your code with different goals: develop (test), debug (fix), proof of concept, large analysis, …
- You'll need to understand where to run/ store depending on your necessities.

Overview of Storage Volumes at Fermilab

	Quota/Space	Retention Policy	Tape Backed?	Retention Lifetime on disk	Use for	path	Grid Accessible
Persistent dCache	No/~100 TB/exp	Managed by Experiment	No	Till manually deleted	immutable files w/ long lifetime	/pnfs/<experiment>/persistent	Yes
Scratch dCache	No/no limit	LRU eviction - least recently used file deleted	No	Varies, ~30 days (**NOT** guaranteed)	immutable files w/ short lifetime	/pnfs/<experiment>/scratch	Yes
(Soon deprecated) Resilient dCache	No/O(5) GB	Periodic eviction if file not accessed	No	Approx 30 days (your experiment may have an active clean up policy)	code library tarballs for grid jobs (do NOT use for grid job outputs)	/pnfs/<experiment>/resilient	Yes
Tape backed dCache	No/O(4) PB	LRU eviction (from disk)	Yes	Approx 30 days	Long-term archive	/pnfs/<experiment>/rest_of_path	Yes
BlueArc Data	Yes (~1 TB)/ ~100 TB total	Managed by Experiment	No	Till manually deleted	Storing final analysis samples	/<experiment>/data	No
BlueArc App	Yes (~100 GB)/ ~3 TB total	Managed by Experiment	No	Till manually deleted	Storing and compiling software	/<experiment>/app	No

https://cdcvs.fnal.gov/redmine/projects/fife/wiki/Understanding_storage_volumes

https://indico.fnal.gov/event/14943/contributions/28629/attachments/18059/22692/DUNE_Data_Management_tutorial_dingpf.pdf

Jobs on the Grid

- For large production you may need to use the grid.
  Check that what you want in not already available as official experiment data production.
  Depending on the scale (experiment) you may need to request it to the data management/production team.

- Most tools rely on tarballs of your code.

- Few ways to submit jobs to the grid :

  - Project.py
    https://sbnsoftware.github.io/sbndcode_wiki/Using_projectpy_for_grid_jobs.html
    https://cdcvs.fnal.gov/redmine/projects/project-py/wiki/Project-py_guide
    https://cdcvs.fnal.gov/redmine/projects/larbatch/wiki/User_guide

  - Jobsub_client
    https://sbnsoftware.github.io/sbndcode_wiki/How_to_launch_grid_jobs.html
    https://wiki.dunescience.org/wiki/DUNE_Computing/Submitting_Jobs_at_Fermilab

  - POMS
    https://indico.fnal.gov/event/48790/contributions/213065/attachments/142274/179587/POMS_for_LArSoft.pdf
    https://indico.fnal.gov/event/49414/contributions/217601/attachments/144636/183861/FIFE_SummerSchool_POMS.pdf

FIFE

POMS

https://wiki.dunescience.org/wiki/DUNE_Computing/Submitting_jobs_on_the_grid_Jan2021

Containers

- Software in HEP is difficult to install/configure
  - Building an image captures everything required to install
  - Multiple users can run containers based on these images

  Why care? You should be familiar with this if using distributed computing (Grid) to avoid “dependency-hell”

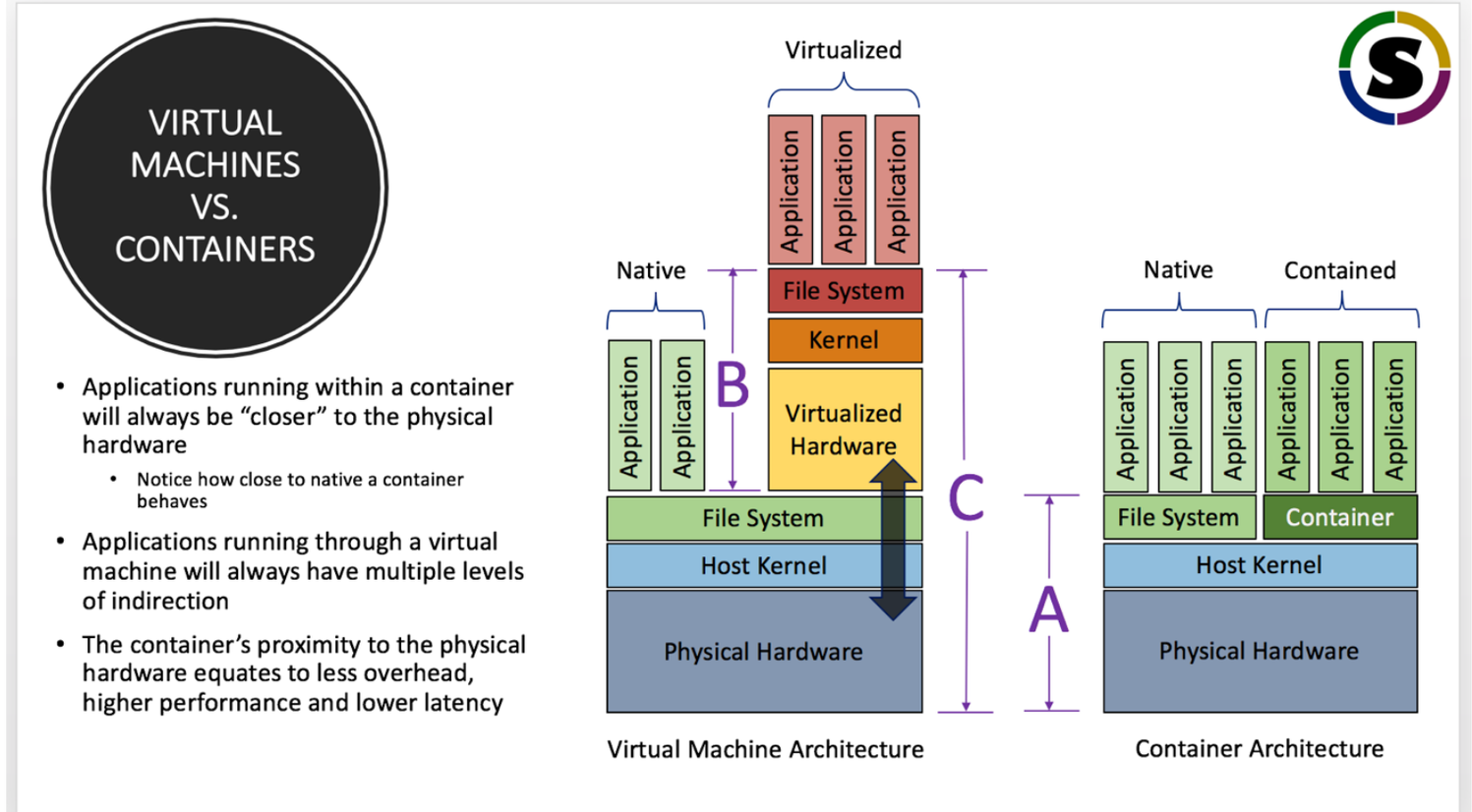
- So what is it? What does it do?
  1. Allows you to run jobs from old systems on the grid (reproducibility)
  2. Keeps jobs and users isolated (security)
  3. Can run same job/code on more machines/hardware (accessibility)

from Rob Currie

- Singularity is kind-of like Python virtualenv, but for the whole system. https://singularity.hpcng.org

- To ensure that your LArSoft jobs are executed in a complete environment, it is strongly recommended that the jobs are executed in the proper Singularity container.
  One such container featuring Scientific Linux Fermi 7 (SL7) is available in CVMFS.

- Enable it in jobsub_submit
  In project.py are already enabled in most configurations.



Summary

- “LArSoft” means/needs lots of things.
- Thankfully lots of developers have gone through and have documented their “odyssey”.
- Read, try, ask .. repeat.
- Enjoy your journey.

Backup

Tutorials/extra refs

- DUNE Software and computing tutorial https://indico.fnal.gov/event/14943/

- https://dune.github.io/computing-training-202105/index.html

- https://sbnsoftware.github.io/icaruscode_wiki/Computing_Resources.html#submitting-jobs-virtual-organisation