HSF Training :

Making “that thing my postdoc taught me” available for everyone

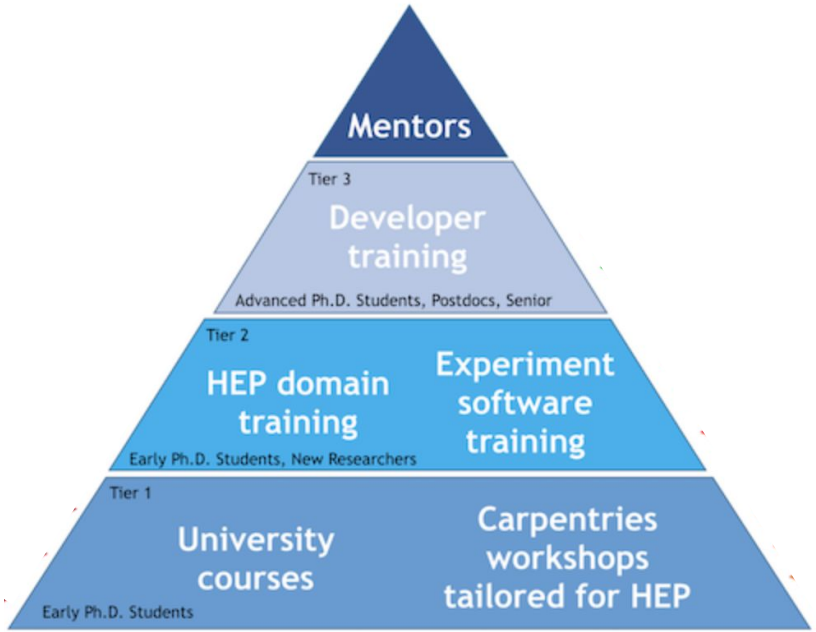
Link to the main training portal : hsf-training

Sam Meehan on behalf of Sudhir Malik & Kilian Lieret (HSF co-conveners) / The Growing Community

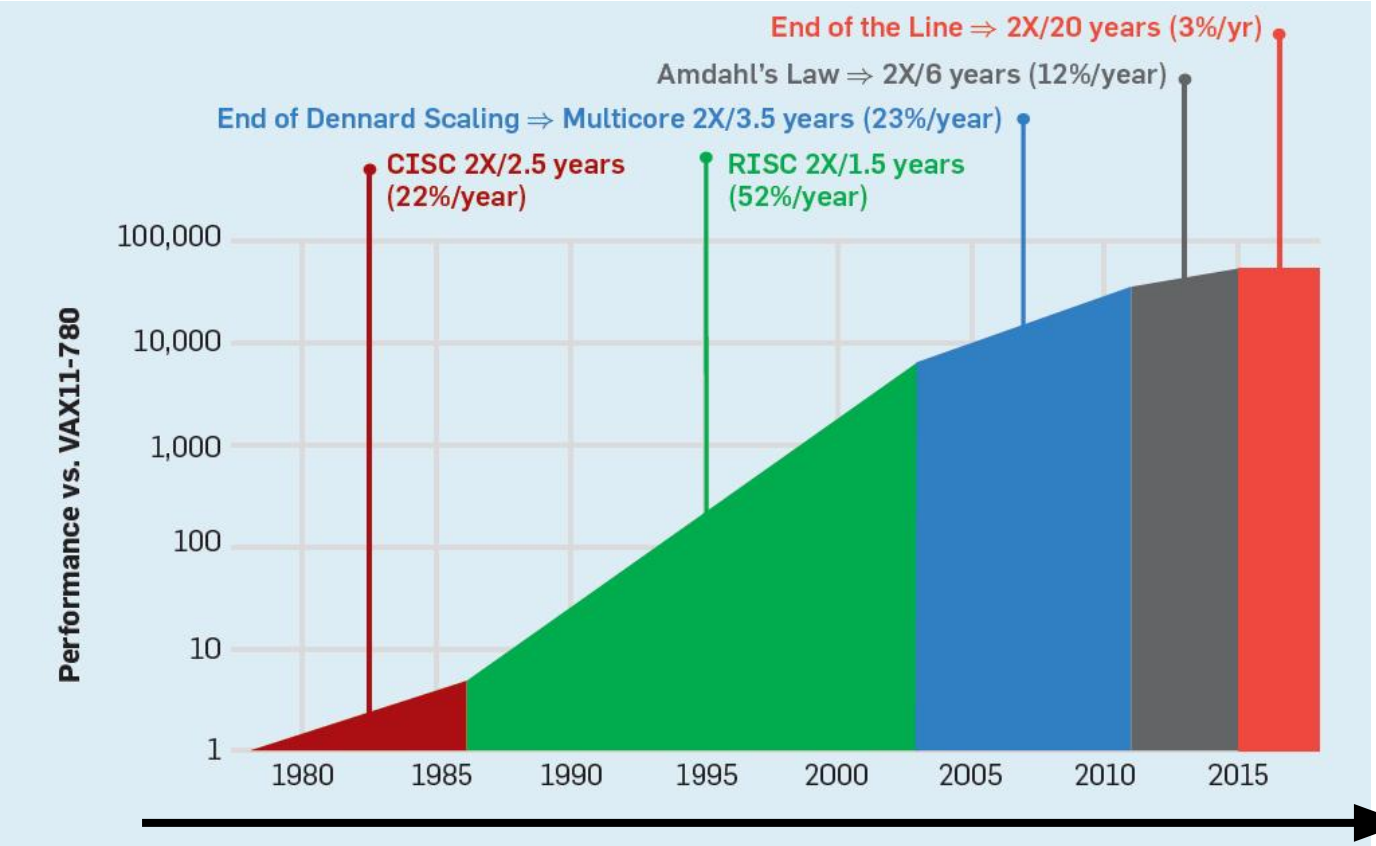
22 July 2020

Mission & Philosophy

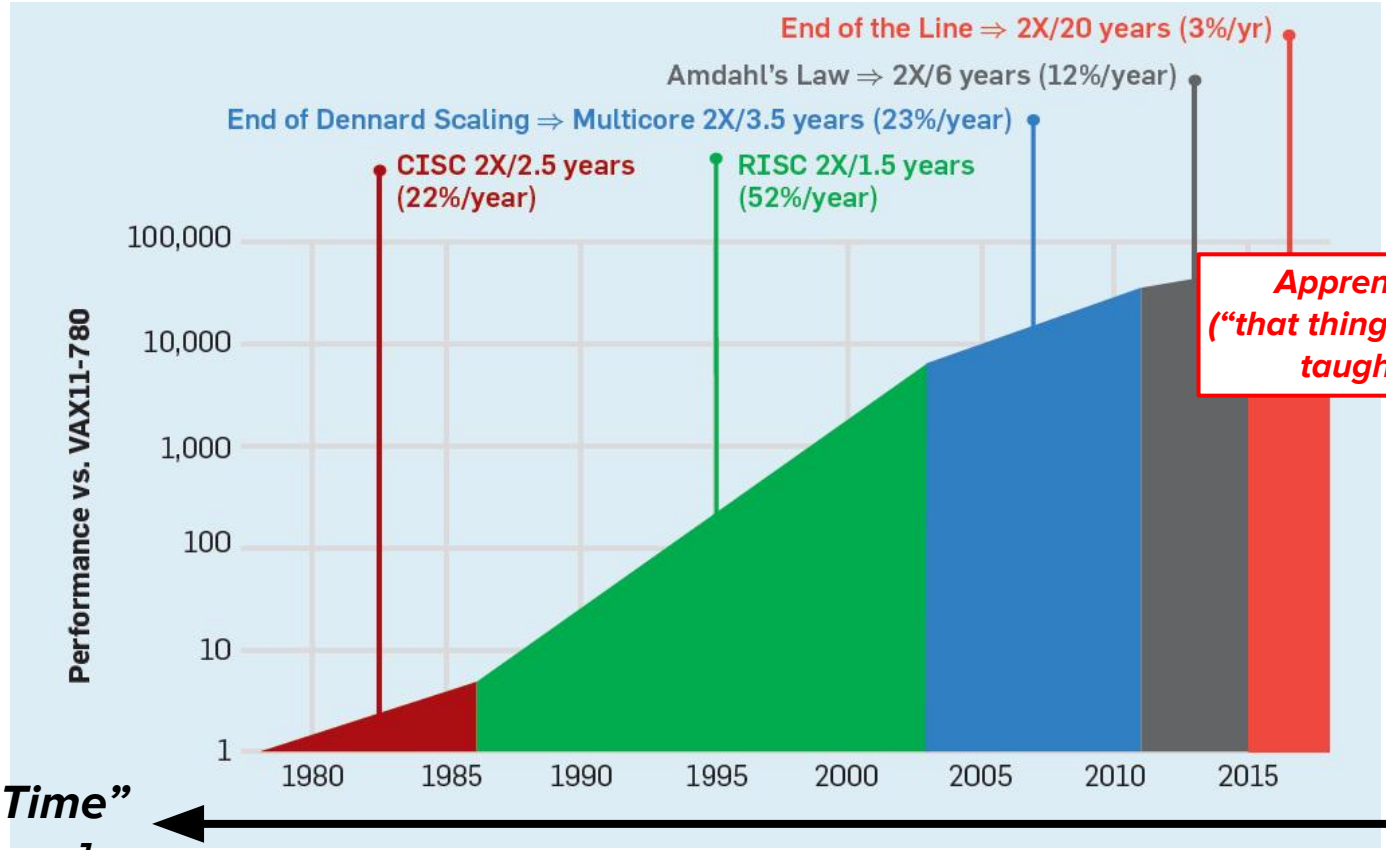
- Mission : *"to help the research community to provide training in the computing skills needed for researchers to produce high quality and sustainable software"*
- Philosophy : largely inspired by Software Carpentries
  - [1] Hands-on
  - [2] Student-centric
  - [3] Experiment Agnostic
  - [4] Re-useable
  - [5] Open and Accessible
- Goal : ***Sustainability ← → Scalability***



Sustainability ← → Scalability



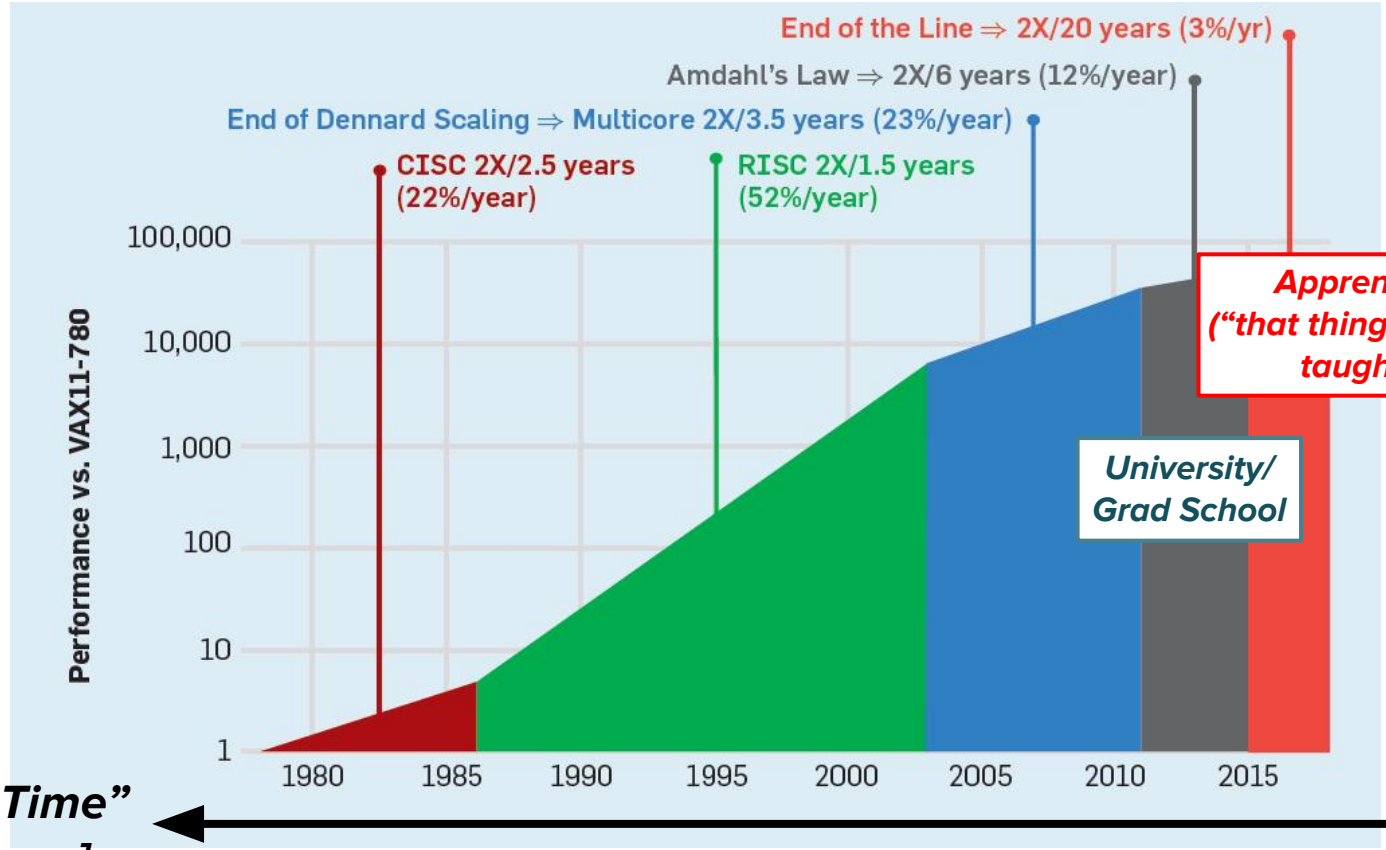
Sustainability ← → Scalability



Apprenticeship ("that thing my postdoc taught me")

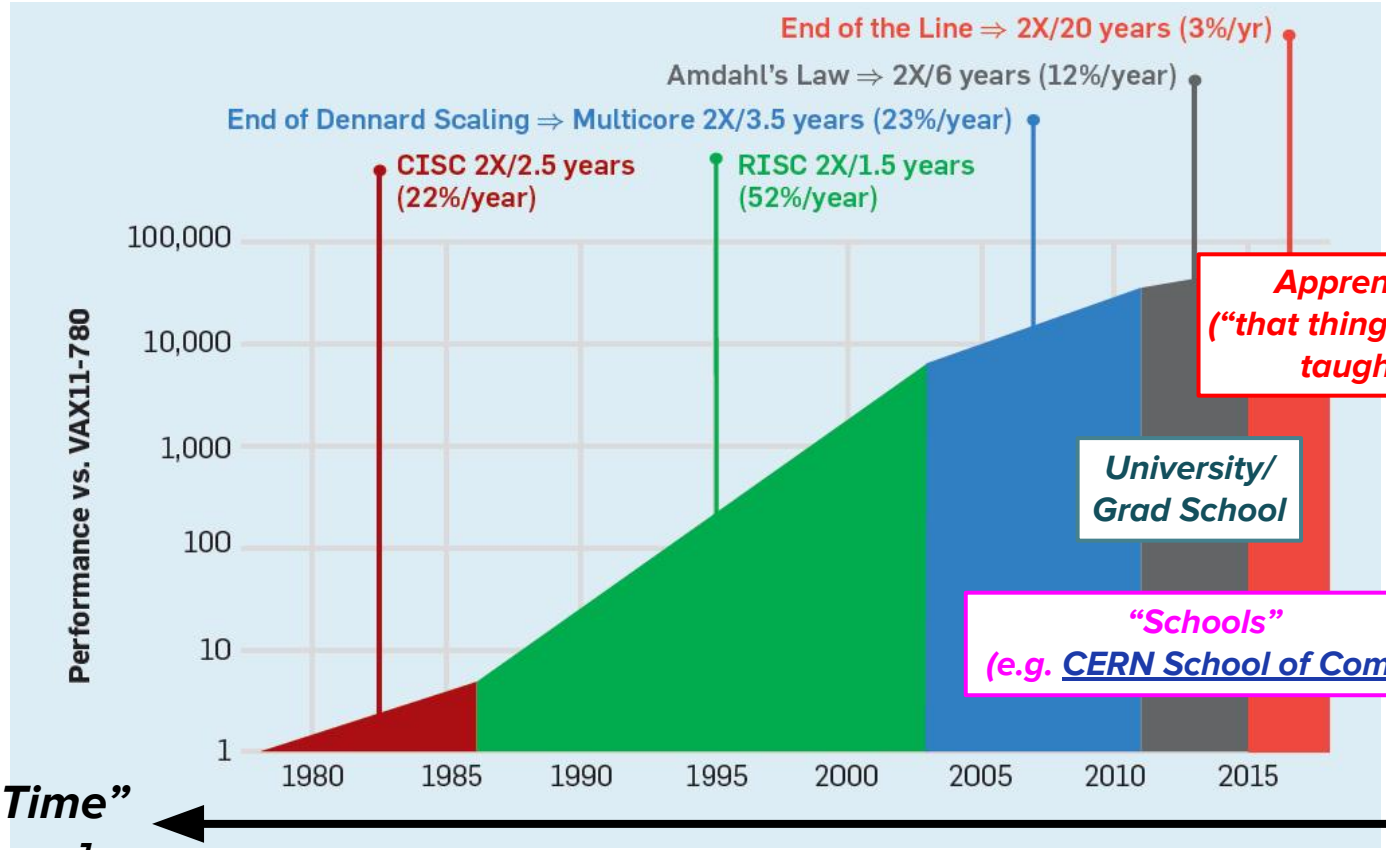
"Sam Time" [< years]

Sustainability ← → Scalability

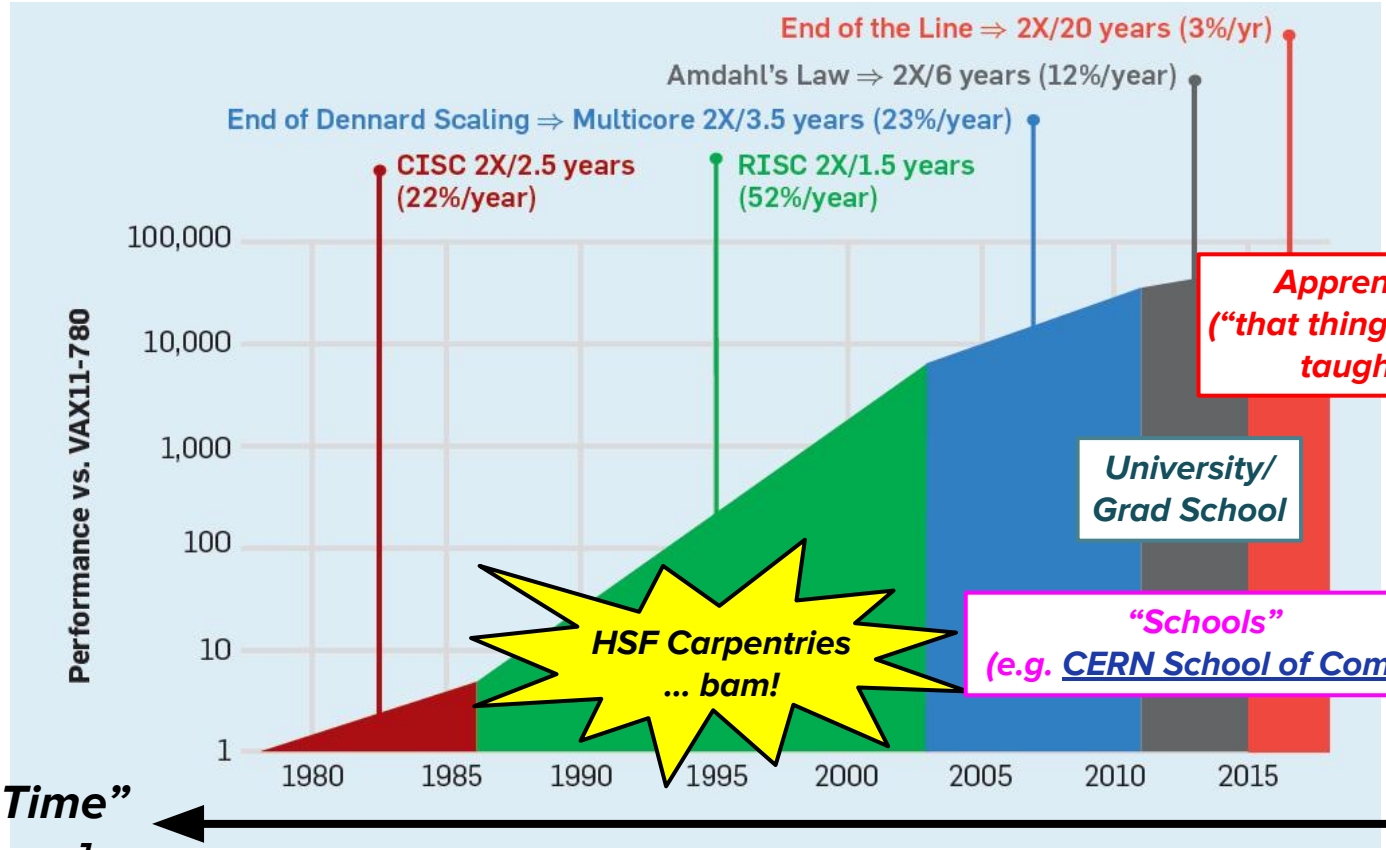


"Sam Time"
[< years]

Sustainability ← → Scalability

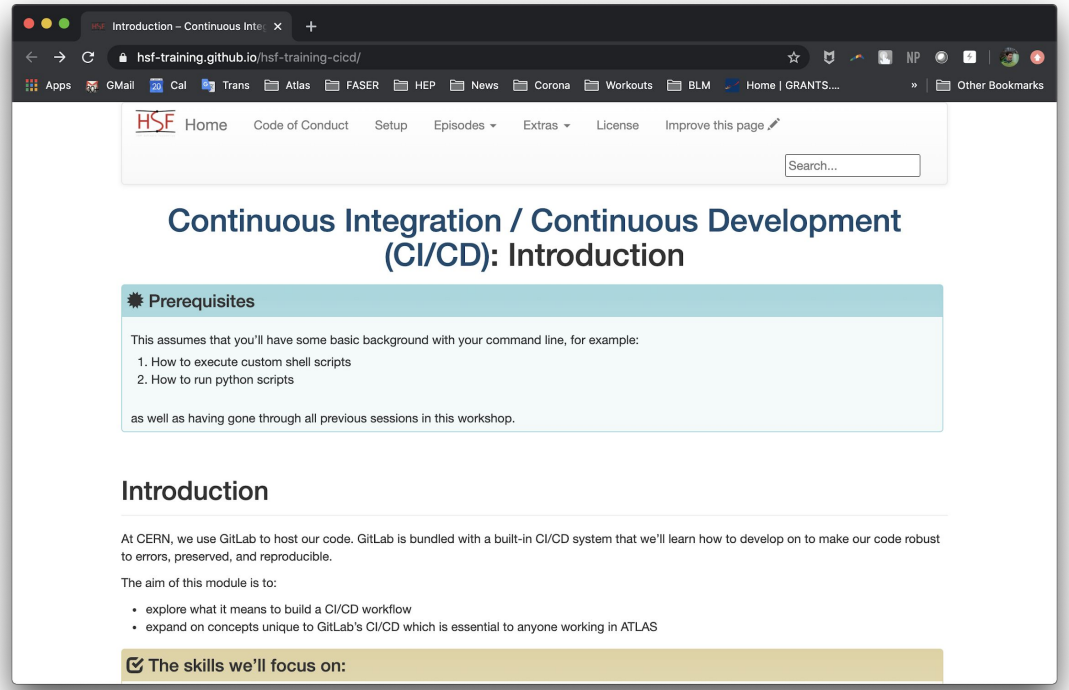


Sustainability ← → Scalability



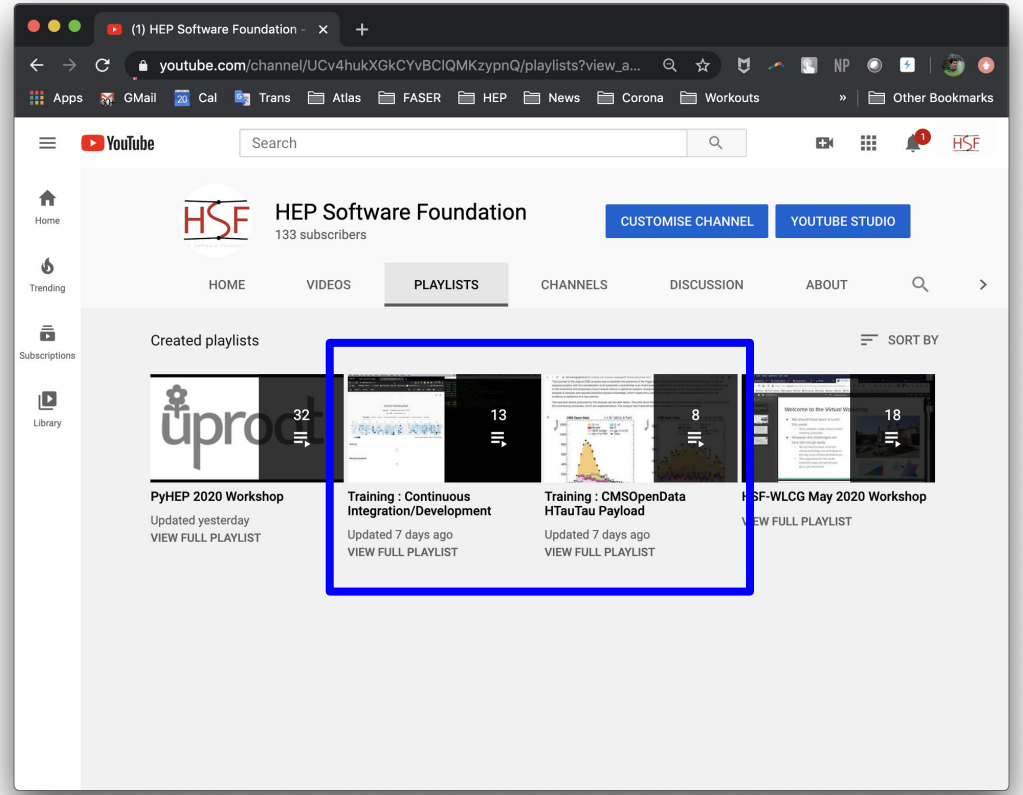
The Preserved Lessons

- ala Software Carpentry
  - Created our own “style”
  - Uniform contextualization and pedagogy of learning materials
- Housed in [hsf-training]
  - Encourage to *fork* and develop lessons ➡ push back any relevant improvements to main lesson
    - Different from
  - How-to page for potential developers
- Supplementing with videos
  - Housed on [HSF YouTube account]
    - 133 followers in one week!



The Fully Preserved Lessons

- ala Software Carpentry
  - Created our own “style”
  - Uniform contextualization and pedagogy of learning materials
- Housed in hsf-training
  - Encourage to *fork* and develop lessons ➡ push back any relevant improvements to main lesson
    - Different from
  - How-to page for potential developers
- Supplementing with videos
  - Housed on HSF YouTube account
    - 133 followers in one week!



The Lesson Wishlist

Taken from the IRIS-HEP Training
February Blueprint Meeting

1. Git/vcs essentials/github (“How to”)
2. Advanced module for git
3. Python foundations
4. Building programs with python
5. Data analysis: numpy, pandas
6. Advanced data analysis
7. Advanced python and pyroot, uproot
8. Build systems: from gcc to cmake
9. Continuous Integration/Development
10. Docker and Containerization
11. Unix (shell, bash, scripting, ...)
12. Advanced unix (shell, bash, scripting, ...)
13. Suggestion: Advanced Unix/terminal
14. Jupyter notebooks and Binder/SWAN
15. ROOT
16. C++
17. Package managers and RPMs
18. Distributed file systems (mounting, access protocols)
19. Batch systems (common scheduler concepts):
20. Distributed computing
21. Best practices and “software engineering”
22. Text editors (vim/emacs/...?) and IDEs
23. Authentication in general; SSH; keys; ssh config; tunneling
24. Machine Learning
25. Debuggers (gdb)
26. Parallel programming
27. Workflows (e.g. yadage) & Reproducibility (e.g REANA)
28. Monte Carlo (pythia, sherpa, madgraph, ...)
29. Simulations (e.g. GEANT)
30. Documentation (doxygen, sphinx ...)

The Lesson Wishlist

From the SWC Curriculum
Production Ready
In (various stages of) Development

1. **Git/vcs essentials/github (“How to”)**
2. Advanced module for git
3. **Python foundations**
4. Building programs with python
5. Data analysis: numpy, pandas
6. Advanced data analysis
7. Advanced python and pyroot, uproot
8. **Build systems: from gcc to cmake**
9. **Continuous Integration/Development**
10. **Docker and Containerization**
11. **Unix (shell, bash, scripting, ...)**
12. Advanced unix (shell, bash, scripting, ...)
13. Suggestion: Advanced Unix/terminal
14. Jupyter notebooks and Binder/SWAN
15. ROOT
16. C++
17. Package managers and RPMs
18. Distributed file systems (mounting, access protocols)
19. Batch systems (common scheduler concepts):
20. Distributed computing
21. Best practices and “software engineering”
22. Text editors (vim/emacs/...?) and IDEs
23. Authentication in general; SSH; keys; ssh config; tunneling
24. **Machine Learning**
25. Debuggers (gdb)
26. **Parallel programming**
27. **Workflows (e.g. yadage) & Reproducibility (e.g REANA)**
28. Monte Carlo (pythia, sherpa, madgraph, ...)
29. Simulations (e.g. GEANT)
30. Documentation (doxygen, sphinx ...)

The Lesson Wishlist

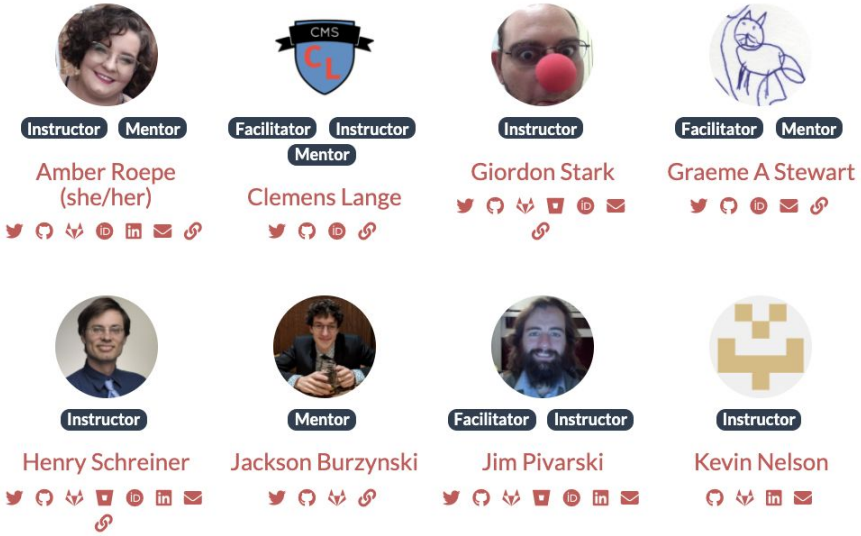
A link exists to some lesson, of varying quality, in various formats, that need access to "that postdoc that wrote it" to be useful

1. Git/vcs essentials/github ("How to")
2. **Advanced module for git**
3. Python foundations
4. **Building programs with python**
5. **Data analysis: numpy, pandas**
6. **Advanced data analysis**
7. **Advanced python and pyroot, uproot**
8. Build systems: from gcc to cmake
9. Continuous Integration/Development
10. Docker and Containerization
11. Unix (shell, bash, scripting, ...)
12. **Advanced unix (shell, bash, scripting, ...)**
13. **Suggestion: Advanced Unix/terminal**
14. **Jupyter notebooks and Binder/SWAN**
15. **ROOT**
16. **C++**
17. **Package managers and RPMs**
18. **Distributed file systems (mounting, access protocols)**
19. **Batch systems (common scheduler concepts):**
20. **Distributed computing**
21. **Best practices and "software engineering"**
22. **Text editors (vim/emacs/...?) and IDEs**
23. **Authentication in general; SSH; keys; ssh config; tunneli**
24. Machine Learning
25. **Debuggers (gdb)**
26. Parallel programming
27. Workflows (e.g. yadage) & Reproducibility (e.g REANA)
28. **Monte Carlo (pythia, sherpa, madgraph, ...)**
29. **Simulations (e.g. GEANT)**
30. **Documentation (doxygen, sphinx ...)**

The Educators

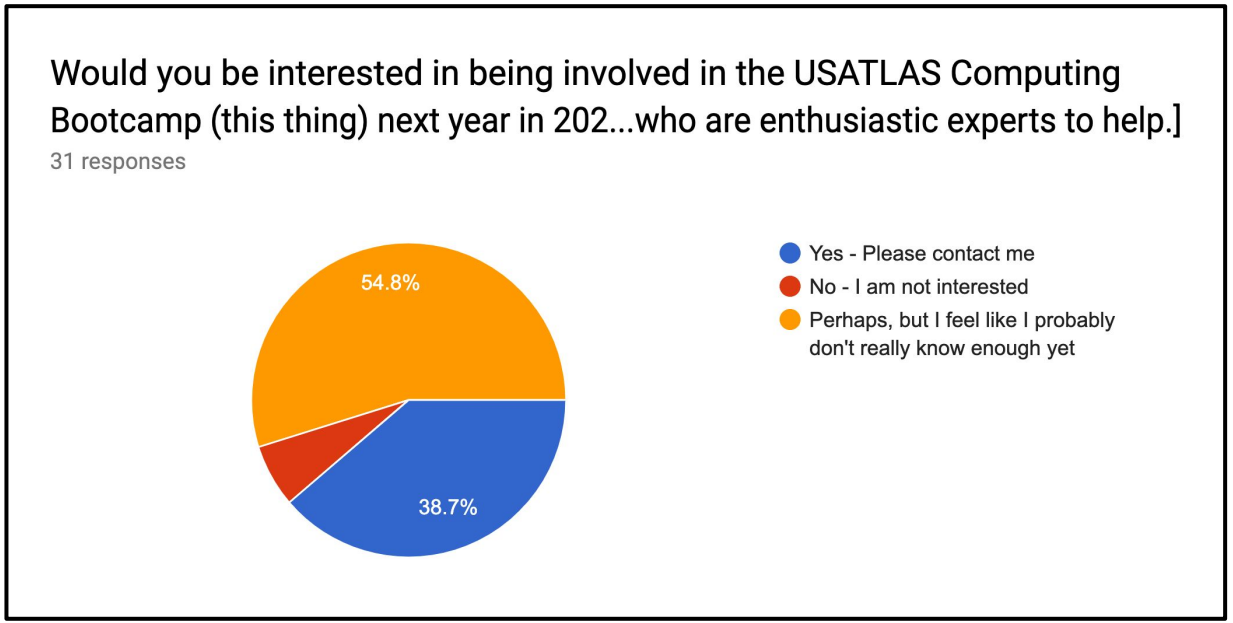
- Three roles for a successful event
  - More details on the HSF-Training Educators page
- ***[1] Facilitator*** : Need 1
  - “Conference organizer”
- ***[2] Instructor(s)*** : Need a few
  - Subject matter “super expert”
  - Develops content/lessons/videos
  - Runs the lessons/keeps track of time
- ***[3] Mentors*** : Need 1 to 5 ratio with attendees
  - People who know a bit more than participants
    - … but can admit when they need to google
  - No need to be super expert
  - Embrace the “learn from teaching” approach

2020



Educators for the Future

- Participants want to be educators
  - Typically recruit 30% of participants to be educators in future



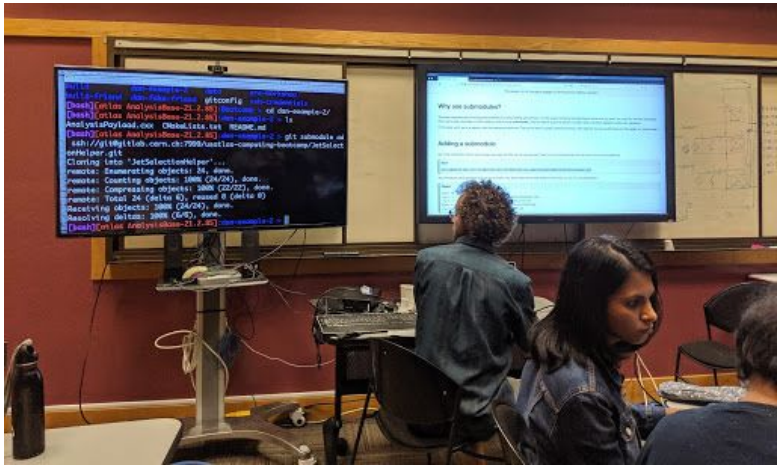
Educators from June
[Virtual Docker Training]

Former Participants in an "Awesome Workshop"

Clemens Lange
Sam Meehan
Brendan Regnery
Kevin Nelson
Adam Parker
Giordon Stark
Savannah Thais
Leonora Vesterbacka
Johan Bonilla
Jackson Burzynski
Meirin Evans
Philipp Gadow
Lukas Heinrich
Konstantin Lehmann
Patrick McCormack
Robin Newhouse
Mason Proffitt
Amber Roepe

Example #1 : In Person

- Attendance : few dozen
- Positives
  - Active/efficient engagement of participants
  - Professional networking and additional “events”
- Negatives
  - Travel costs (education should not be exclusive)
  - Long lead time for planning logistics
    - Related to travel/room booking
  - Requires participant “sacrifice”
- Important things
  - Room setup is crucial
    - Two projects/screens
    - Not an auditorium
    - Ample power



Pre-workshop

Awesome H(tautau) Analysis

[IRIS-HEP blog post by Lukas]

Monday

CI/CD

- Kickoff/Orientation
- Cont. Integration [Giordon]
- Lunch
- Cont. Integration [Giordon]
- ATLAS [Lukas & Sam]
- CMS [Savannah & Clemens]
- Discussion with Theorists + Reception

Tuesday

- Docker [Danicka]
- Lunch
- Catch-up Time
- ATLAS [Lukas & Sam]
- CMS [Savannah & Clemens]
- Dinner @ Meyrinoise

Wednesday

reana

- CERN Re-Ana [Tibor]
- Lunch
- Catch-up Time
- ATLAS [Lukas & Sam]
- CMS [Savannah & Clemens]

Pre-workshop

Monday

CI/CD

Tuesday

Wednesday

reana

Awesome H(tautau) Analysis

Kickoff/Orientation

Cont. Integration [Giordon]

Lunch

Cont. Integration [Giordon]

ATLAS [Lukas & Sam]

CMS [Savannah & Clemens]

Discussion with Theorists + Reception

Docker [Danicka]

Lunch

Catch-up Time

CERN Re-Ana [Tibor]

Lunch

Catch-up Time

New Subject Matter

- Driven by the Instructor
- Mentors move around room to spot-check participation and debug simple issues

IRIS-HEP blog post by Lukas

Pre-workshop

Monday

CI/CD

Tuesday

Wednesday

reana

Free-Form Technical Discussion
- Loosely organized
- Can be application/experiment specific
- Explicit time for personalized help

H(tautau) Analysis

Docker
[Danicka]

CERN Re-Ana
[Tibor]

Lunch

Lunch

Lunch

Cont. Integration
[Giordon]

Catch-up Time

Catch-up Time

ATLAS
[Lukas & Sam]

CMS
[Savannah & Clemens]

ATLAS
[Lukas & Sam]

CMS
[Savannah & Clemens]

ATLAS
[Lukas & Sam]

CMS
[Savannah & Clemens]

Dinner
@ Meyrinoise

Discussion with Theorists + Reception

IRIS-HEP blog post by Lukas

Location, location, location

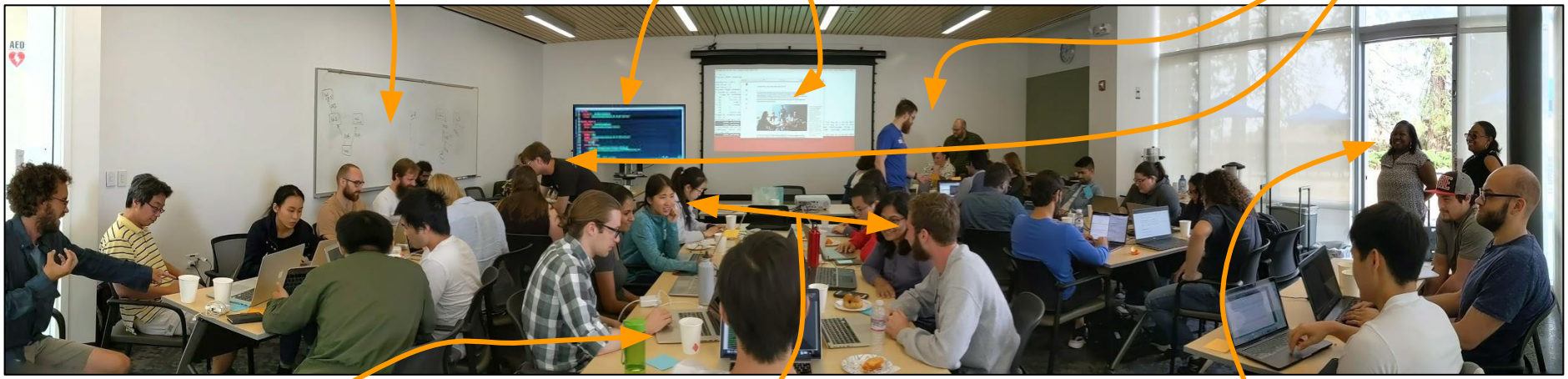
- Success of the workshop is highly dependent on the location
  - Is this event “vidyo-able” and can be held remotely?
    - ~~No [Sam’s opinion in Aug 2019]~~ → Maybe [Sam’s new opinion]

Location, location, location

Whiteboards for describing concepts (e.g. git branching/merging)

Two screens :
[1] Projection of material - students follow along as well
[2] Display of instructor terminal - coding on the fly

Large/open space ➔ instructors can move around and help participants



Big tables to allow for {notebook, laptop, coffee/snacks}

NOT an auditorium - participants face each other ➔ promotes discussion

Awesome local coordination/help

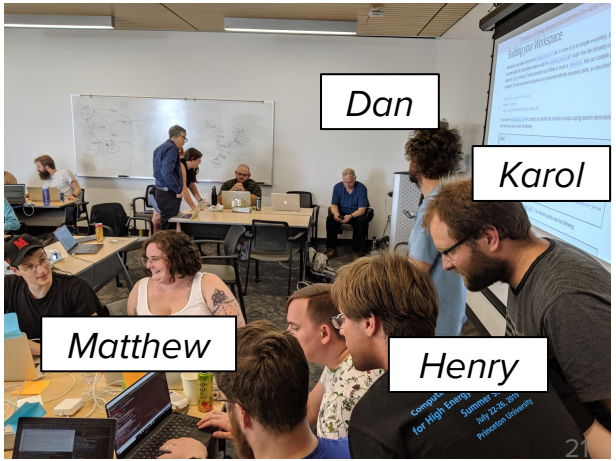
The Golden Ratio

- Ratio of Participant : Educator <= 5
  - This is *essential* to allow for the “hands on” aspect of the workshop to be successful
- Large time commitment on behalf of the educators
  - Can’t just “do your talk” and then leave

Zach : “I’m confused that ...”

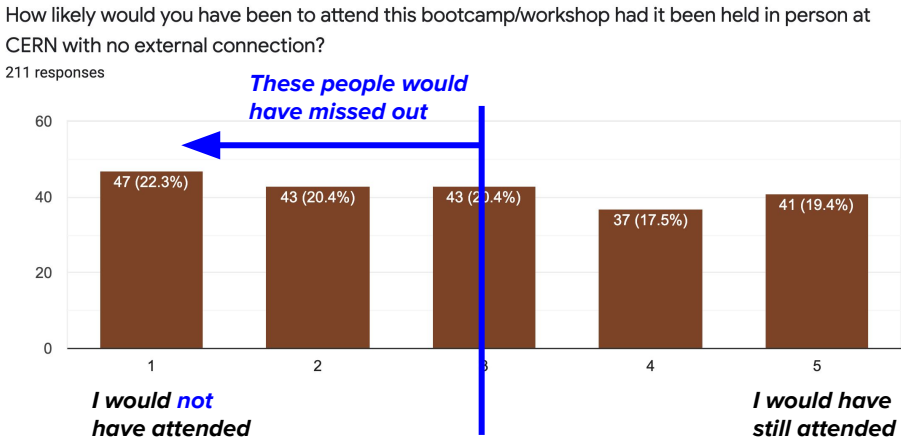
Zach : “Yeah, I already tried that ...”

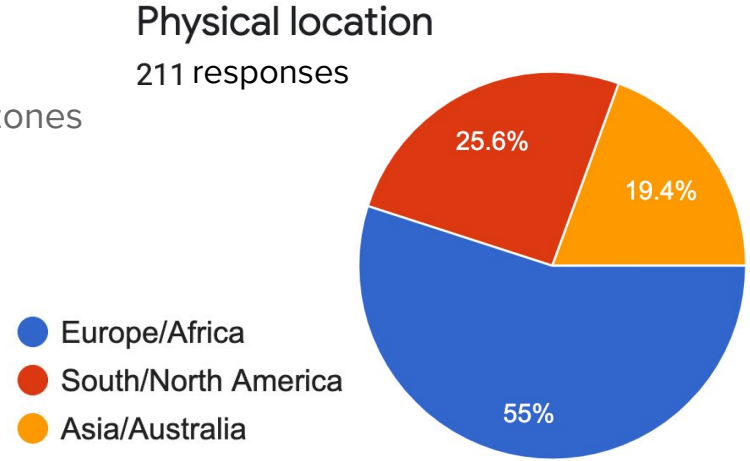
Zach : “Ahhhh, that makes sense!”



Example #2 : Virtual

- Attendance : few ***hundred***
- Positives
  - Broader reach : >100 registrants for both events
    - 2 times greater likelihood to participate
  - No travel costs ➜ critical for some supervisors
  - Don't need to plan in as much advance
  - Materials are more fully preserved (i.e. videos)
- Negatives
  - Difficult educator/participant interactions
  - Need mentors spaced in (potentially) different time zones
  - Challenging to keep everyone on same page
  - Higher attrition rate from registrants ➜ participants
- Important things
  - Have well defined roles
  - Effective chat application is essential
    - e.g. mattermost/discord/slack





Monday

Welcome

Kickoff/Orientation
[15-16 CET]

Tues/Wed

Work on your own, when you want

Watch and work through recorded tutorials
payload by Kevin

Watch and work through recorded tutorials
CI/CD by Giordon

Thursday

hands-on

Block 1:
[8-10 CET]

Block 2:
[10-12 CET]

Block 3:
[12-14 CET]

Block 4:
[14-16 CET]

Monday

Welcome

Work on y

New Subject Matter

- Pre-recorded by the Instructor and posted to YouTube
- Participants work at their own pace
- Active assistance provided via Slack (or something like it) by Instructors & Mentors

Kickoff/Orientation
[15-16 CET]

Watch and work through recorded tutorials
payload by Kevin

Watch and work through recorded tutorials
CI/CD by Giordon

Block 1:
[8-10 CET]

Block 2:
[10-12 CET]

Block 3:
[12-14 CET]

Block 4:
[14-16 CET]

Monday

Welcome

Kickoff/Orientation
[15-16 CET]

Tues/Wed

Work on your own, when you want

Watch and work through recorded tutorials
payload by Kevin

Free-Form Technical Discussion

- Use individual sessions and assign 1 mentor : 5 participants per Zoom/Vidyo/Bluejeans session
- Can center discussion on some “challenge topics” or let it be driven by participants

Thursday

Hands-on

Block 1:
[8-10 CET]

Block 2:
[10-12 CET]

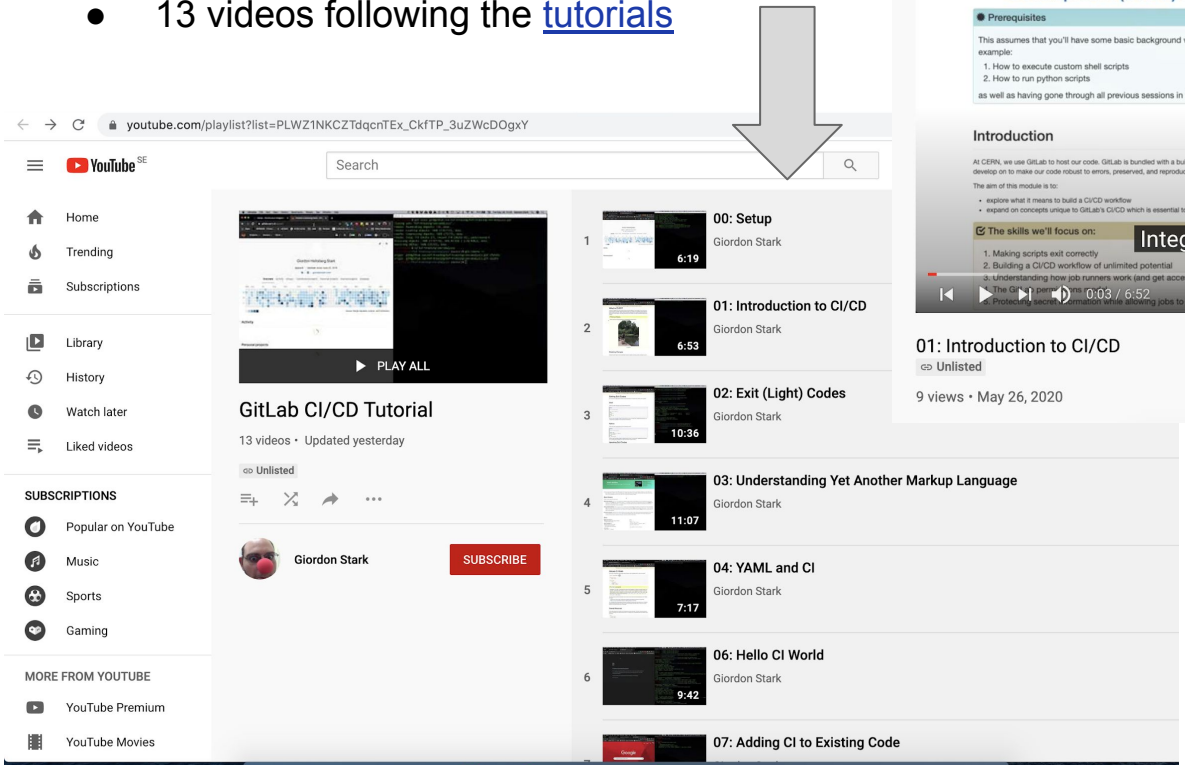
Block 3:
[12-14 CET]

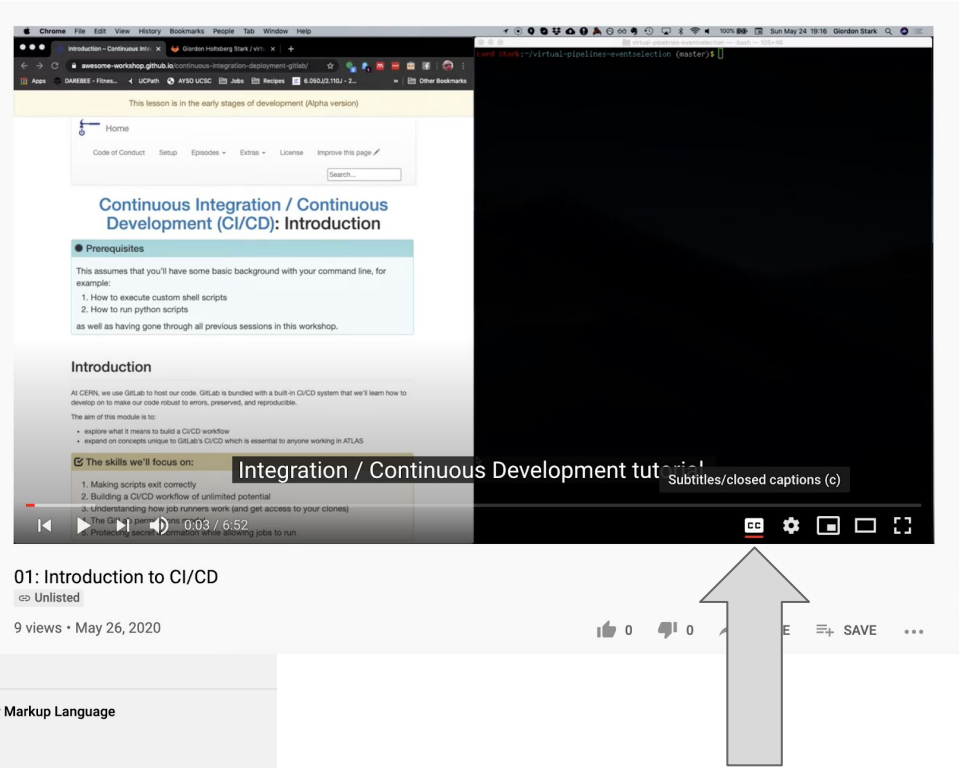
Block 4:
[14-16 CET]

Almost "In Person"

[GitLab CI/CD Videos]

- 13 videos following the [tutorials]

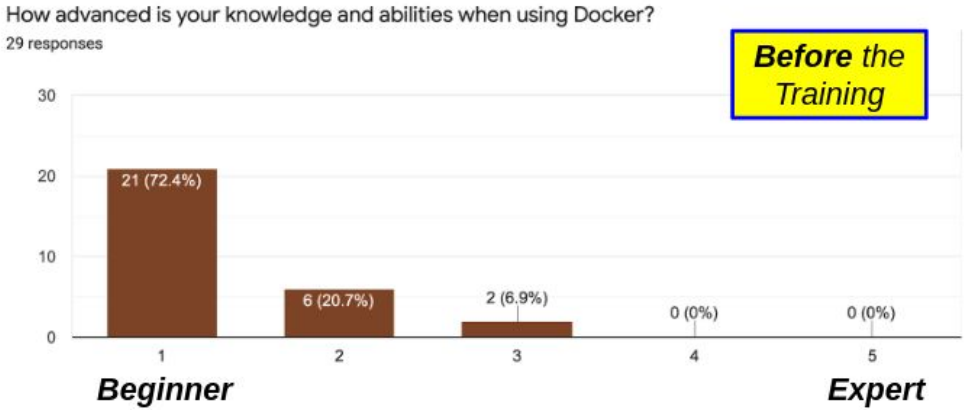


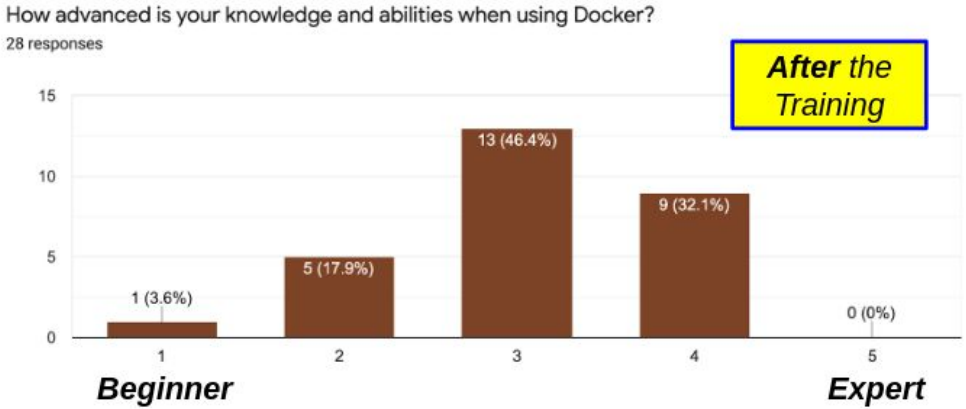


turn on caption here!

Does it work?

- We do our best to diligently collect before/after data via surveys
  - Pre-survey
    - Demographics
    - How much do you know?
  - Post-survey
    - How much do you **now** know?
    - What can we do better next time?
  - Would like to have further out “follow up” surveys (takes more work …)
- Self-reported learning *does* happen!





Conclusions

- Are we filling a niche that wasn't filled before? No
  - HEP PhD ← → "learning to compute"
- Are we making that niche more uniform/accessible/efficient/approachable?
  - Definitely - Look at the statistics
- For the immediate future
  - Develop/fill out core curriculum
    - Challenge : Teaching of c++
  - Understand what factorizes
    - What is "someone else's responsibility"?
- For the further future
  - Establish official MoU with SWC
  - Formalize HEP education (e.g. "career path")