

# **HSF Training :**

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

---

Link to the main  
training portal :  
[hsf-training](https://hsf-training.org/)

Sam Meehan on behalf of  
HSF-Training  
19 November 2020

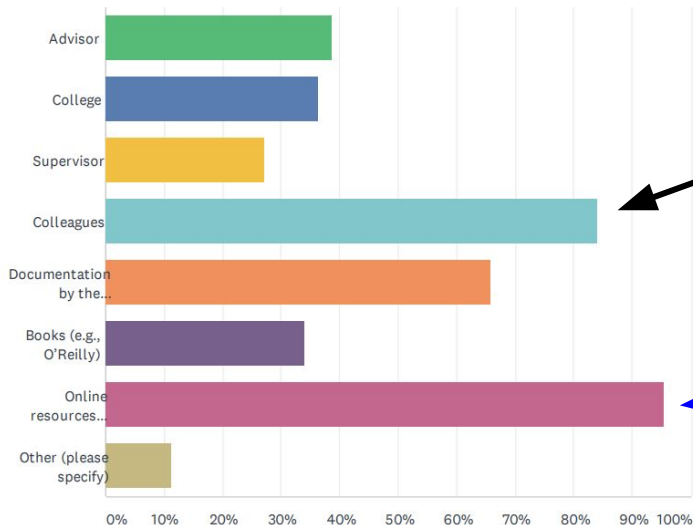
# Reminder from First Talk

Markus Diefenthaler  
Talk at this workshop

## Survey

Q4 Which of the following resources have you used for your software and computing work or research? Select all that apply.

Answered: 44 Skipped: 0

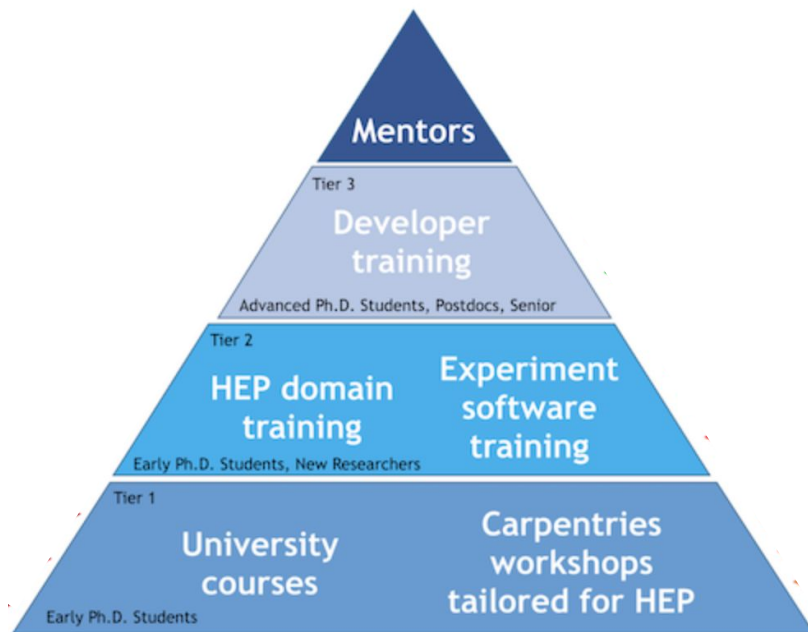


***“That thing my postdoc taught me that once”***

**StackOverflow**  
**[cplusplus.com/docs.python.org](http://cplusplus.com/docs.python.org)**  
**YouTube videos** (from that enterprising faculty member at Univ of XYZ)  
**TWikis** (experiment/lab specific)  
...  
**Non-standard format**  
**Non-targetted for HEP needs**

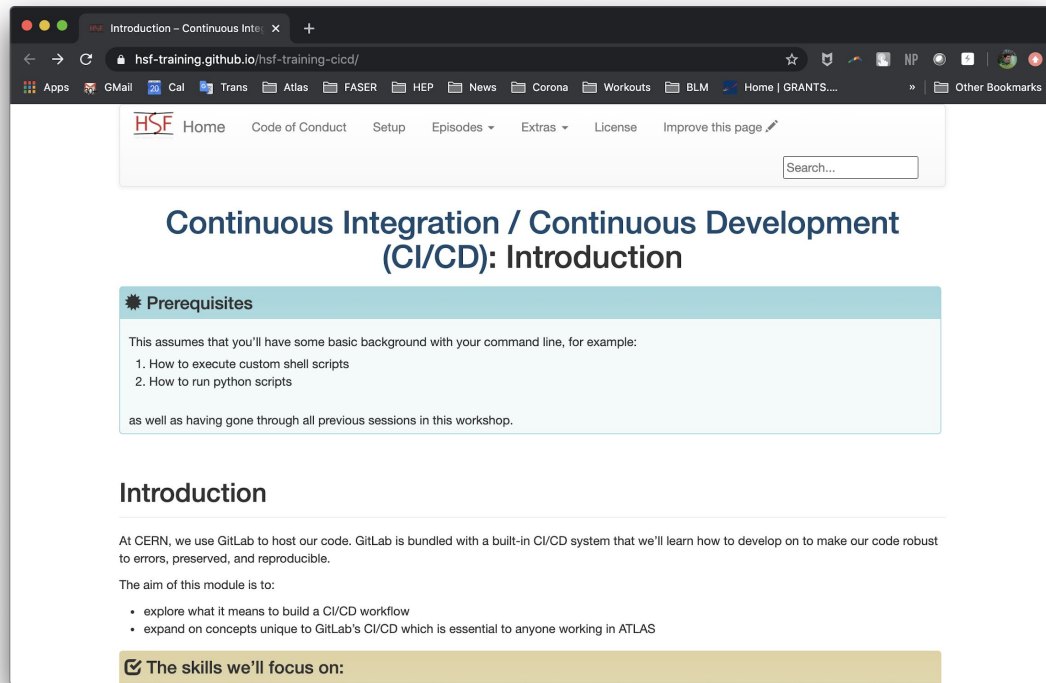
# Mission & Philosophy

- 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**  $\leftrightarrow$  **Scalability**



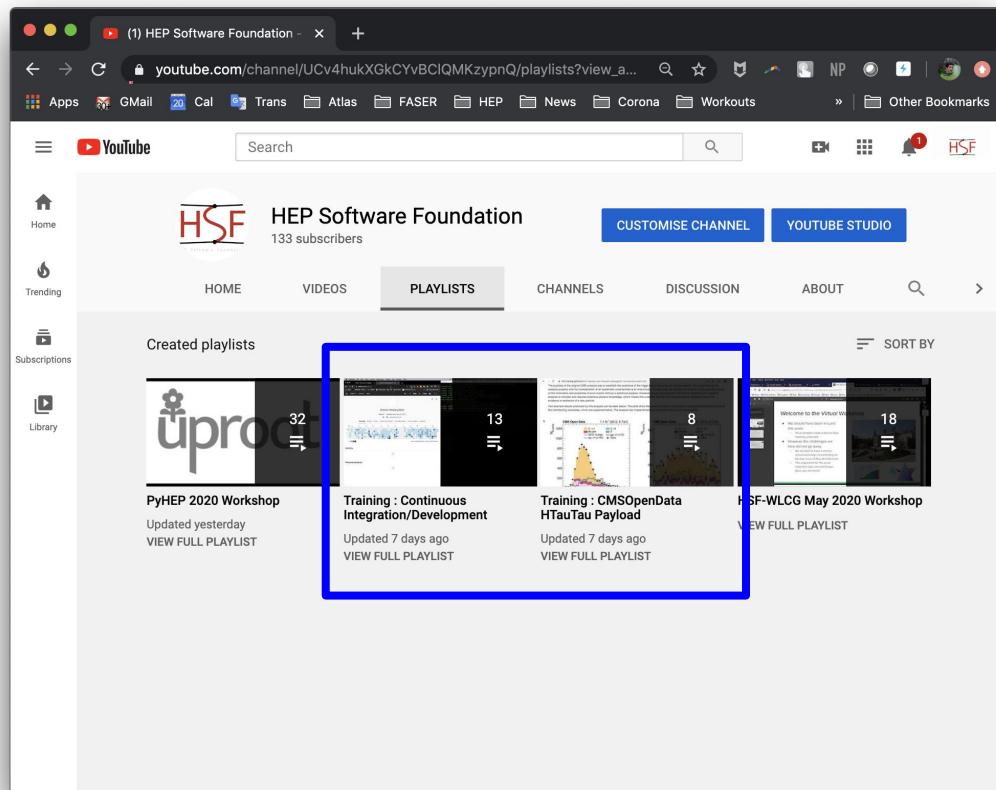
# The Preserved Lessons

- 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; 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**
7. **Advanced**
8. Build sys
9. Continuo
10. Docker a
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 (concepts):**
20. **ing”**
21. **ssh config; tunneli**
22. **Learning**
23. **25. Debuggers (gdb)**
24. **26. Parallel programming**
25. **27. Workflows (e.g. yadage) & Reproducibility (e.g REANA)**
26. **28. Monte Carlo (pythia, sherpa, madgraph, ...)**
27. **29. Simulations (e.g. GEANT)**
28. **30. Documentation (doxygen, sphinx ...)**

**Think you know one of these things?**  
Join us for “meta hackathon” for lesson development to  
be held December 16-18 in the virtual realm!  
(<https://indico.cern.ch/event/975487/>)

# 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

Awesome  
H(tautau) Analysis

[IRIS-HEP blog  
post by Lukas](#)

## Monday CI<sup>⚡</sup>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]

### ***New Subject Matter***

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

## Wednesday


CERN Re-Ana  
[Tibor]

Lunch

Catch-up Time

Pre-workshop

Monday  
CI<sup>⚡</sup>CD

Tuesday  


Wednesday  


**Free-Form Technical Discussion**

- Loosely organized
- Can be application/experiment specific
- Explicit time for personalized help

H(tautau) Analysis

Lunch

Cont. Integration  
[Giordon]

ATLAS  
[Lukas & Sam]

CMS  
[Savannah & Clemens]

Discussion with  
Theorists + Reception

Docker  
[Danicka]

Lunch

Catch-up Time

ATLAS  
[Lukas & Sam]

CMS  
[Savannah & Clemens]

Dinner  
@ Meyrinoise

CERN Re-Ana  
[Tibor]

Lunch

Catch-up Time

ATLAS  
[Lukas & Sam]

CMS  
[Savannah & Clemens]

[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]



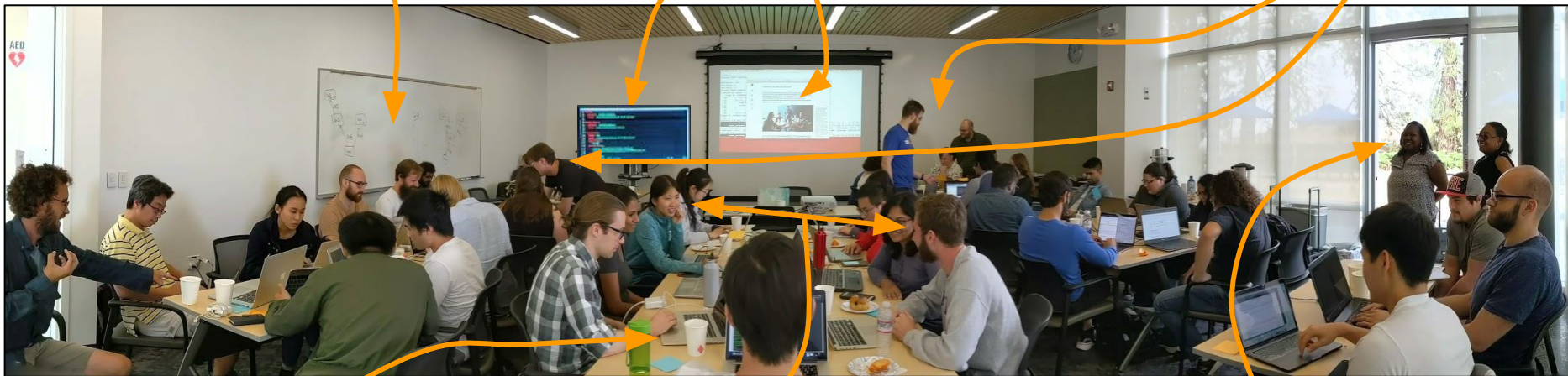


# Location, location, location

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  $\leq 5$ 
  - 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!”



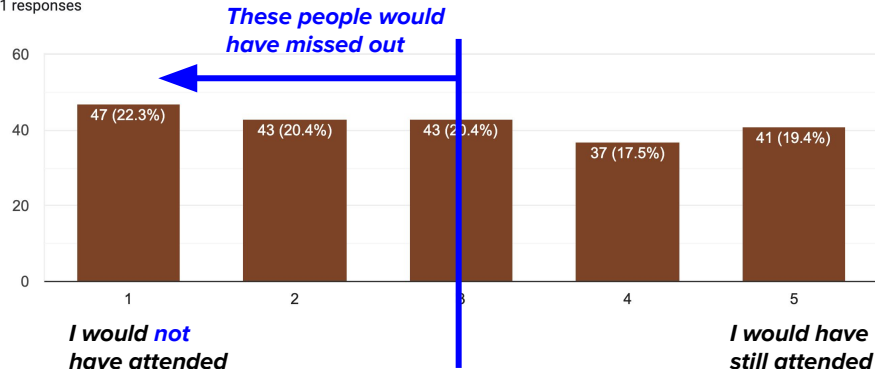


# Example #2 : Virtual

- 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

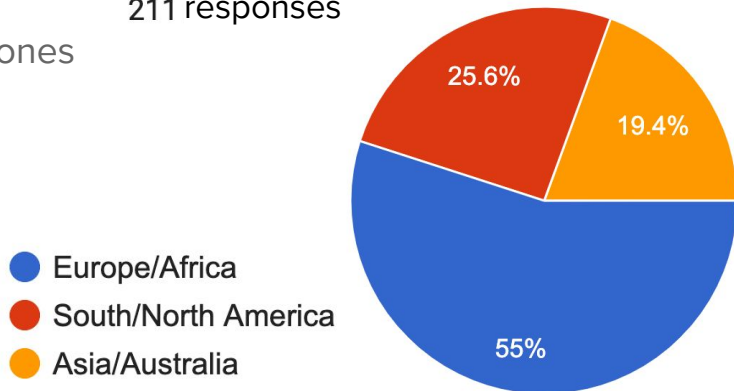
How likely would you have been to attend this bootcamp/workshop had it been held in person at CERN with no external connection?

211 responses



## Physical location

211 responses



# 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 your

## ***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

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]

Kickoff/Orientation  
[15-16 CET]

# Monday

Welcome

# Tues/Wed

Work on your own, when you want

# Thursday

Hands-on

Watch and work through  
recorded tutorials  
[payload by Kevin](#)

Kickoff/Orientation  
[15-16 CET]

## **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

Block 1:  
[8-10 CET]

Block 2:  
[10-12 CET]

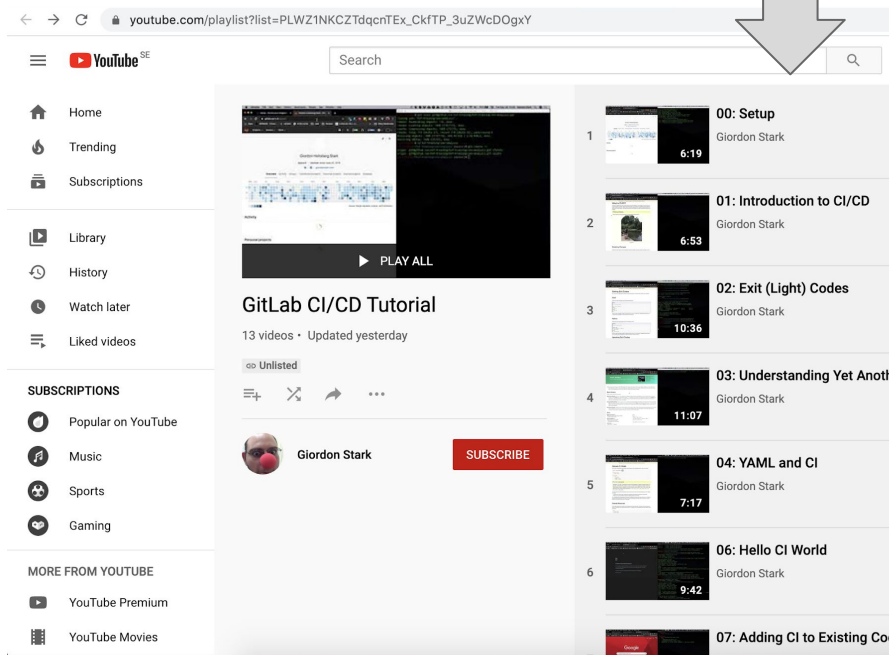
Block 3:  
[12-14 CET]

Block 4:  
[14-16 CET]

# Almost "In Person"

## GitLab CI/CD Videos

- 13 videos following the tutorials



youtube.com/playlist?list=PLWZ1NKCZTdqenTEx\_CkFTP\_3uZWcDOgxy

YouTube

Home  
Trending  
Subscriptions  
Library  
History  
Watch later  
Liked videos

SUBSCRIPTIONS

Popular on YouTube  
Music  
Sports  
Gaming

MORE FROM YOUTUBE

YouTube Premium  
YouTube Movies

**GitLab CI/CD Tutorial**  
13 videos · Updated yesterday

Unlisted

Giordon Stark

SUBSCRIBE

00: Setup  
Giordon Stark  
6:19

01: Introduction to CI/CD  
Giordon Stark  
6:53

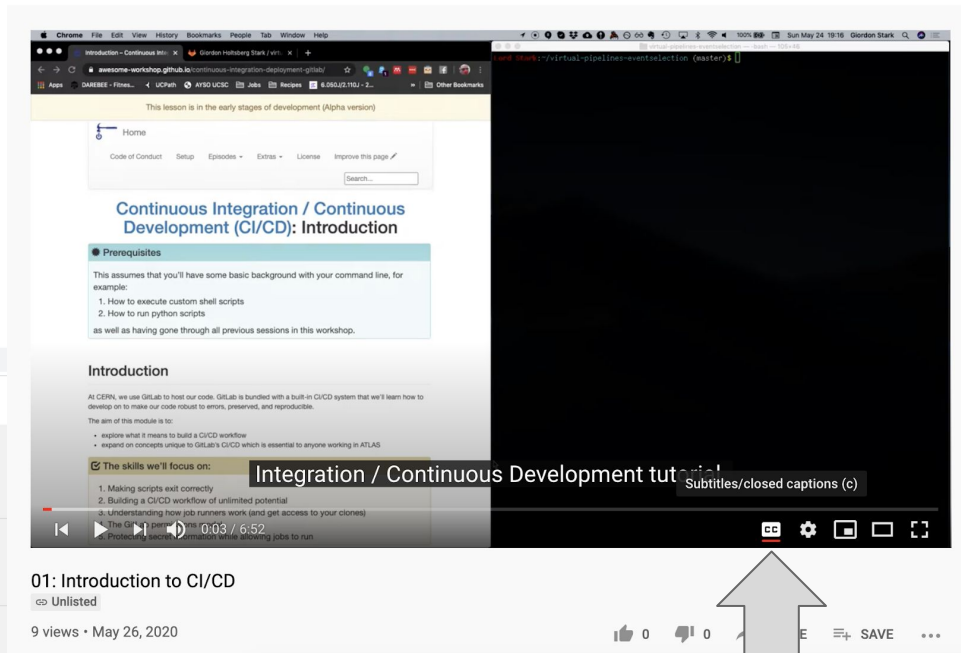
02: Exit (Light) Codes  
Giordon Stark  
10:36

03: Understanding Yet Another Markup Language  
Giordon Stark  
11:07

04: YAML and CI  
Giordon Stark  
7:17

06: Hello CI World  
Giordon Stark  
9:42

07: Adding CI to Existing Code



Chrome File Edit View History Bookmarks People Tab Window Help

awesome-workshop gitlab continuous-integration-development-gitlab

This lesson is in the early stages of development (Alpha version)

Home

Code of Conduct Setup Episodes Extras License Improve this page

Continuous Integration / Continuous Development (CI/CD): Introduction

Prerequisites

This assumes that you'll have some basic background with your command line, for example:

1. How to execute custom shell scripts
2. How to run python scripts

as well as having gone through all previous sessions in this workshop.

Introduction

At CERN, we use GitLab to host our code. GitLab is bundled with a built-in CI/CD system that we'll learn how to develop on to make our code robust to errors, preserved, and reproducible.

The aim of this module is to:

- explore what it means to build a CI/CD workflow
- expand on concepts unique to GitLab's CI/CD which is essential to anyone working in ATLAS

The skills we'll focus on:

1. Making scripts exit correctly
2. Building a CI/CD workflow of unlimited potential
3. Understanding how job runners work (and get access to your clones)

The GitLab CI/CD system is a powerful tool for automating your build and deployment jobs to run

Integration / Continuous Development tutorial

Subtitles/closed captions (c)

01: Introduction to CI/CD  
Unlisted

9 views · May 26, 2020

0 0

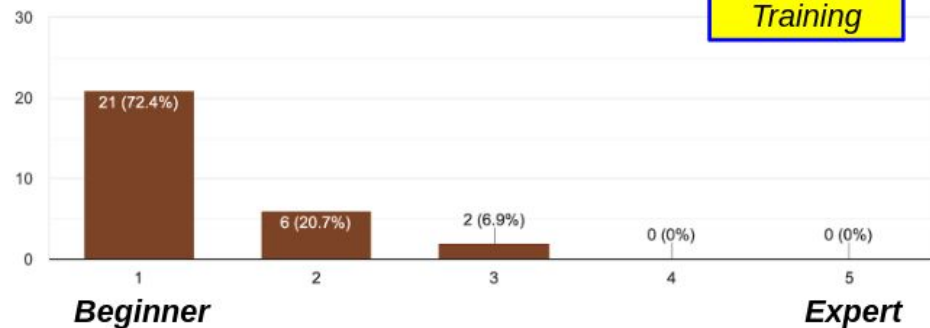
SAVE

turn on caption here!

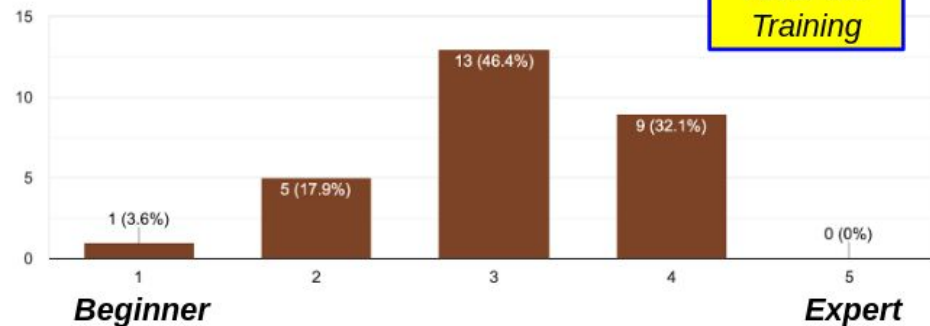
# Does it work?

- 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!

How advanced is your knowledge and abilities when using Docker?  
29 responses



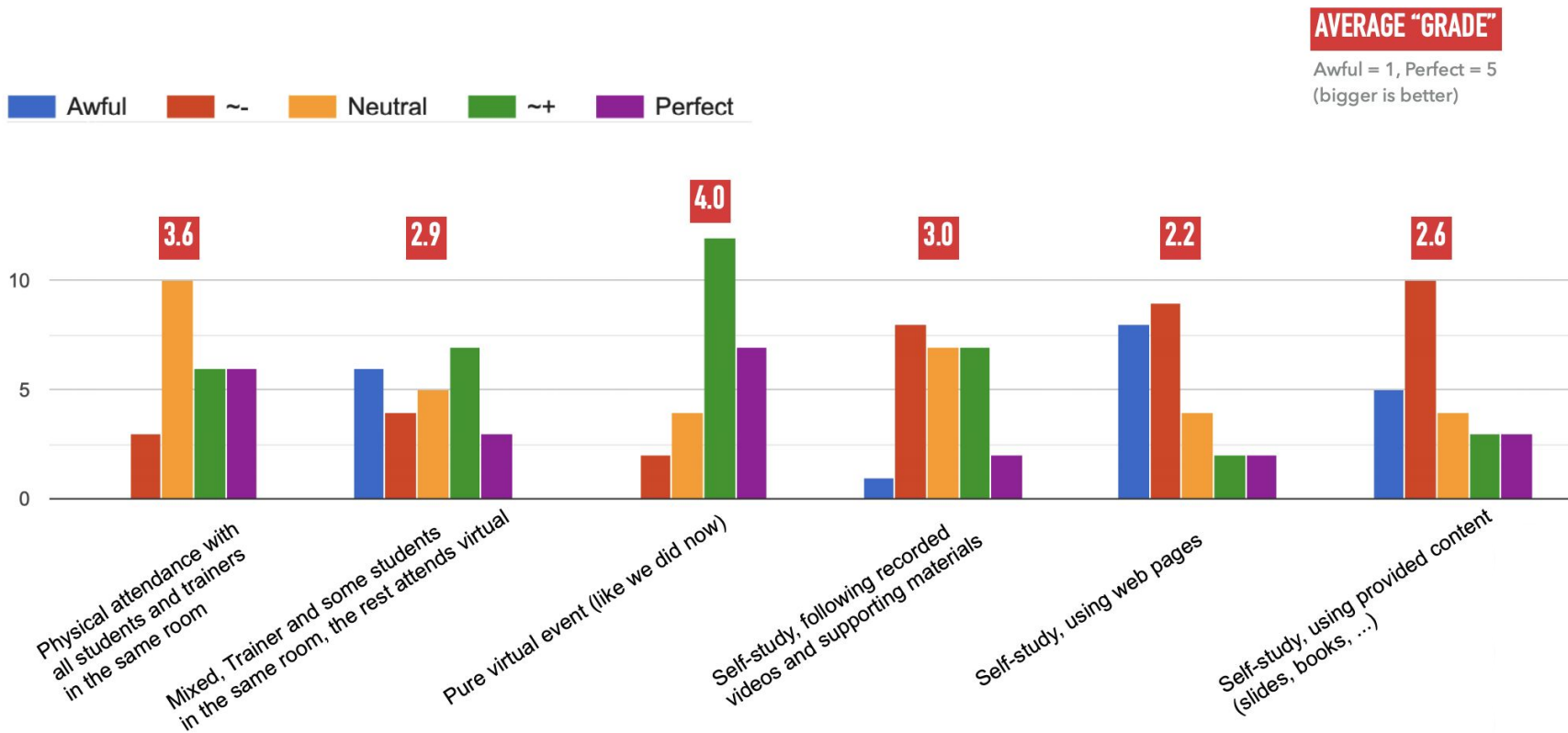
How advanced is your knowledge and abilities when using Docker?  
28 responses



# Virtual vs. In Person

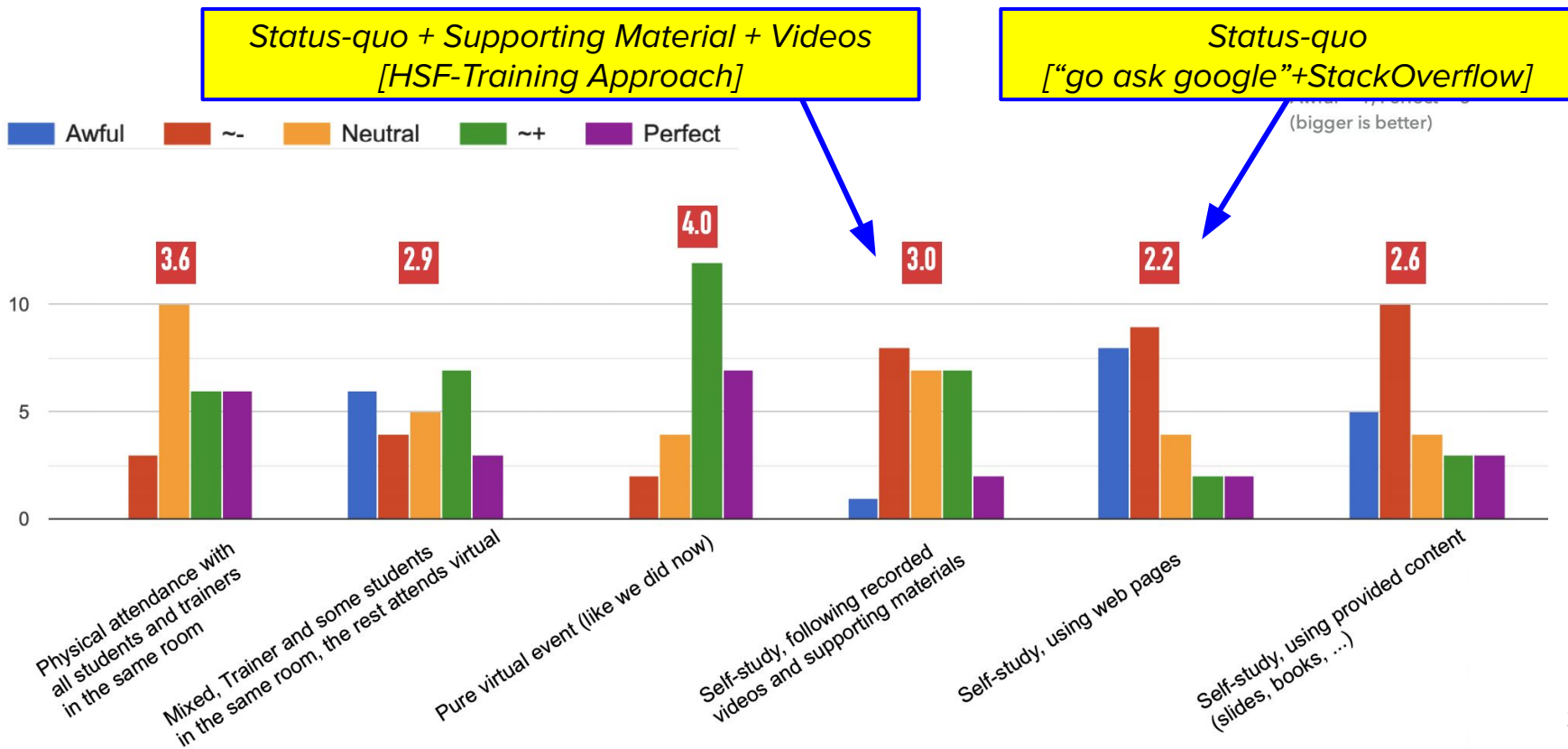
From c++ event in early November :  
<https://indico.cern.ch/event/969325/>

## PLS LET US KNOW WHICH TRAINING FORMAT FOR SUCH EVENTS WOULD FIT YOU BEST



# Virtual vs. In Person

PLS LET US KNOW WHICH TRAINING FORMAT FOR SUCH EVENTS WOULD FIT YOU BEST





# Virtual vs. In Person

PLS LET US KNOW WHICH TRAINING FORMAT FOR SUCH EVENTS WOULD FIT YOU BEST

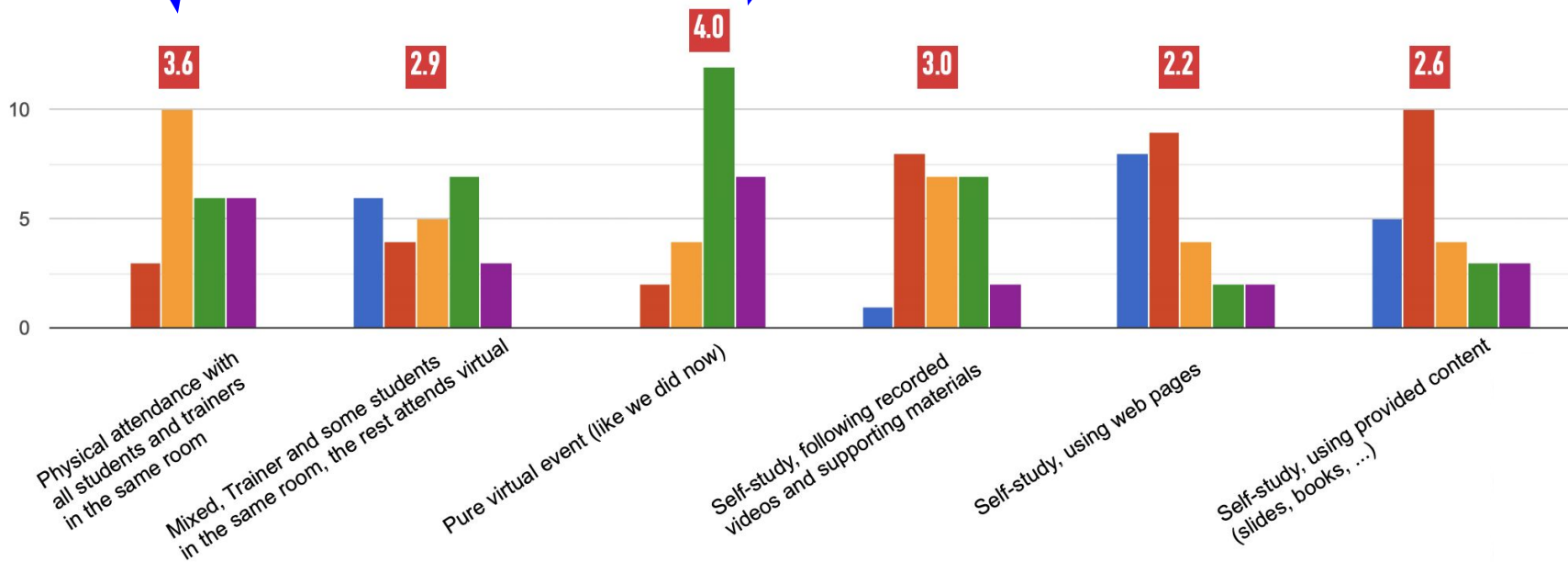
*In-Person Workshops*  
*[pre-covid preference]*

*Virtual Workshops*  
*[covid-forced]*

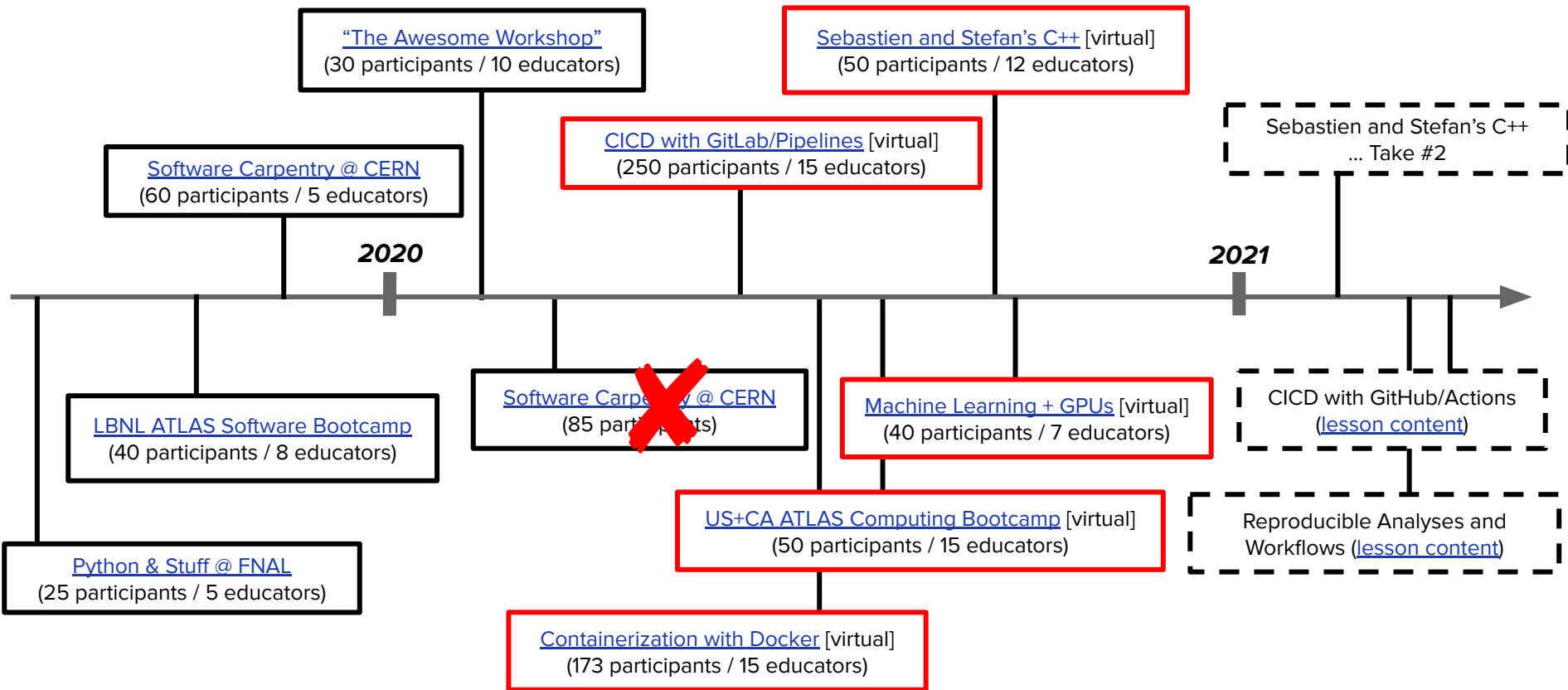
**AVERAGE "GRADE"**

Awful = 1, Perfect = 5  
(bigger is better)

Awful ~- Neutral ~+ Perfect



# All Trainings to Date



# Conclusions

- Are we filling a niche that wasn't filled before? No
  - HEP PhD  $\leftrightarrow$  “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”)

