HSF WORKSHOP 2019, JLAB

ALBERT PUIG NAVARRO, ON BEHALF OF MANY PEOPLE*

*SPECIAL THANKS TO CHRIS BURR AND DARIO BERZANO, FROM WHOM I STOLE SLIDES, GRAPHICS AND IDEAS

STARTERKIT

FNSNF
FONDS NATIONAL SUISSE
SCHWEIZERISCHER NATIONALFONDS
FONDO NAZIONALE SVIZZERO
SWISS NATIONAL SCIENCE FOUNDATION

Universität
Zürich^UZH

SUMMARY AND CONCLUSIONS

The Starterkit is an extremely successful software teaching initiative started at LHCb in 2015 following the Software Carpentry techniques

- It currently includes LHCb, ALICE and SHiP (new since last HSF workshop!)
- It covers both documentation and hands-on teaching through yearly workshops

Since its creation, the documentation generated by this initiative has become the de-facto standard documentation for many aspects of the software

While very successful, it is demanding in terms of time and sustainability

- How can recognition be given to the people organising it? Can we keep this up? How can it be stored?

WHAT IS THE STARTERKIT?

The Starterkit initiative is software training for the next generation of physicists

- Mainly run by the “young” people in a voluntary basis, often former attendees
- Large departure from the “traditional” documentation through Twiki’s and tutorial lessons

It covers two sides of software training:

- Documentation: step-by-step tutorials, without magic and starting from the lowest level
  - In the case of experiment-specific software, focus on teaching the basic building blocks so more complex tasks can be easily understood
- Hands-on teaching through workshops (at least once per year)
  - Advanced workshops also carried out once a year

DIFFERENT SOFTWARE, COMMON TEACHING

LHCb

- Most analysts configure software with Python
- Individual analysis repositories, analysis done locally with C++ macros or Python

ALICE

- Base analysis task is a C++ class based on ROOT
- Software stack needs to be built on user laptop

SHiP

- Everything is still under development, so while most user-facing code is Python, most users have to dive into the C++ internals.
- Use of ALICE tooling to install and manage the software framework and its dependencies.

TEACHING MATERIAL

Material is publicly available on the web (CC BY 4.0 + MIT for code snippets)

- General (https://hsf-training.github.io/analysis-essentials/), LHCb (lhcb.github.io/starterkit-lessons) and ALICE (alice-doc.github.io/alice-analysis-tutorial)

Source is on Github and tested/deployed using Travis CI

- Anyone can contribute via Pull Requests: lower entry barrier for students to contribute (a session on fixing minor issues is carried out in the last day of the workshop)
- Code snippets extracted from the doc and tested daily to ensure the they always work

Material is now the official source of documentation for LHCb and ALICE

- Usage extends well beyond the workshops, it rarely becomes outdated

STARTERKIT WORKSHOP

Hosted at CERN, fee of 25 CHF to cover coffee breaks and social event

- Follow the Software Carpentry teaching methodology
- Remote participation was not included due to the hands-on nature of the teaching methods and to ensure high attendance and engagement

Remote participation introduced at ALICE’s request but with remote mics muted

- ALICE is more geographically diverse, with less travel opportunities, so remote attendance was very high, although with low engagement
- Mattermost channel: questions filtered/reported to teachers

SHiP joined for the first time in 2018, for a total of 45 (LHCb) + 25 (ALICE) + 12 (SHiP) participants

- Make use of the remote participation capabilities
- Invite outside experts, *i.e.*, presentation from the ROOT team

WORKSHOP SCHEDULE

2 days of general computing, shared lessons

- Teachers and students are shared

3 days of experiment-specific lessons

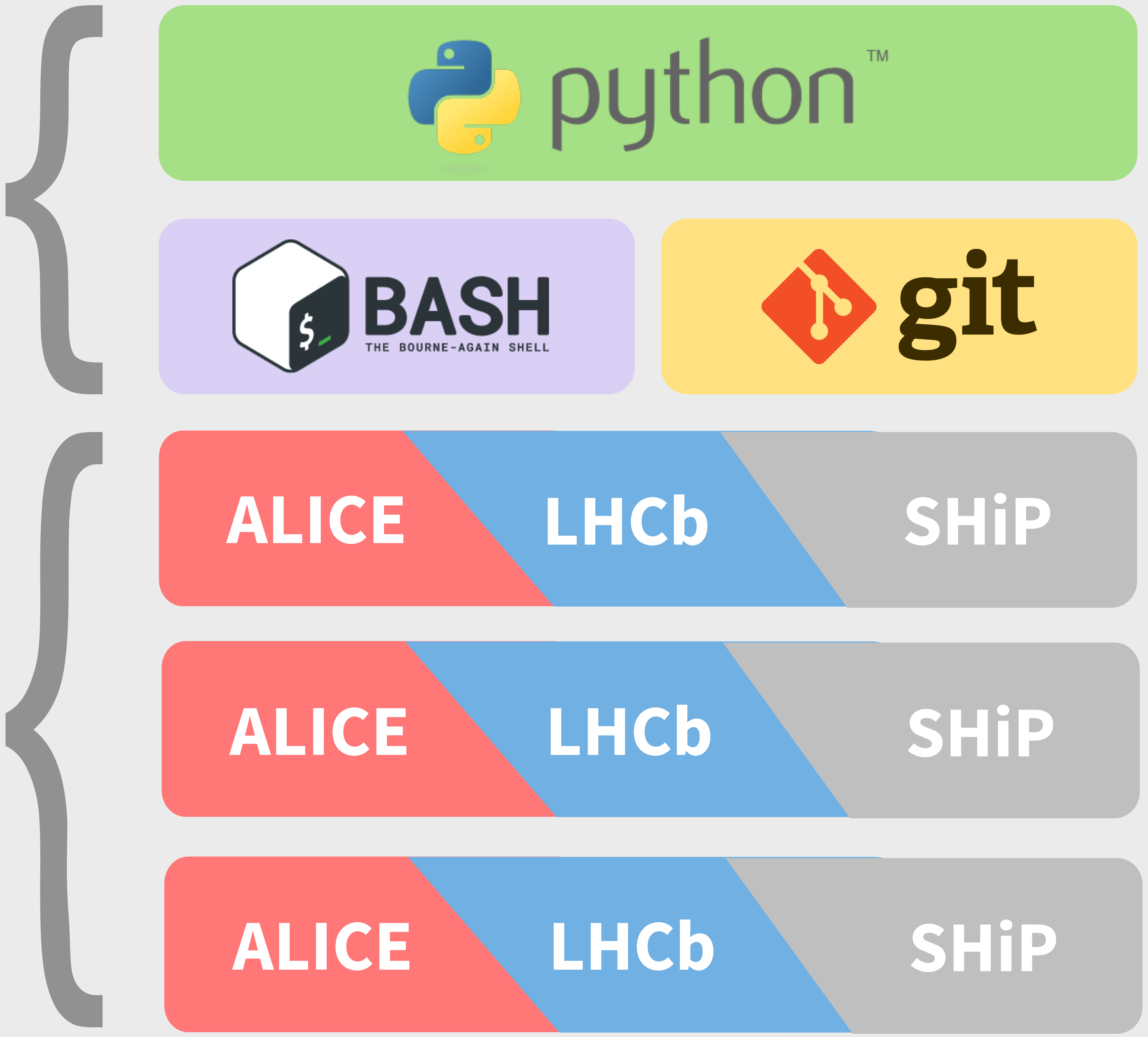
- Some overlaps between ALICE and SHiP

Social event on day 4 🍕🍻

- Key for networking!

Days 1, 2
General computing

Days 3, 4, 5
Experiment specific



FEEDBACK

At the end of each session, feedback from students is gathered with Google

Teaching methodology well-appreciated

- Students really like the one-to-one help, also with solving other specific issues they have
- Some rare cases almost require personal assistance

Generally well paced, and level is considered good

- Some improvements for the general classes, but much better than in 2017
- Even advanced students learn new tricks
- Students would like longer classes

Enjoyed networking between experiments

LONG TERM CHALLENGES

Preparing workshops is time consuming

- Find suitable teachers and helpers: able, capable and available
- Review and refresh the teaching material

Lack of recognition

- Teaching is often regarded as a side task, so students/postdocs can’t spend a lot of time on it (or even encouraged to not pursue these endeavours
- There’s no career rewards (only personal ones!)

Documentation needs to be maintained

- Time consuming, requires some level of expertise to keep up with latest software developments (input from experts is crucial)

A FEW THOUGHTS ON SUSTAINABILITY

Engagement of new students is crucial: former attendees are at the core of future event organisation ("next year" is always mentioned)

- Work required in motivation and encouragement to help building teaching confidence
- "Learning how to teach" workshops/courses necessary

Documentation is a long term resource which belongs to the community (not the authors)

- Crucial to ensure continuity when teachers leave
- Share common documentation across experiments to reduce overhead

Increase reach: decentralised Starterkits?

HOW DO WE BUILD A SUSTAINABLE FUTURE FOR SOFTWARE TEACHING AND LITERACY?

THANK YOU