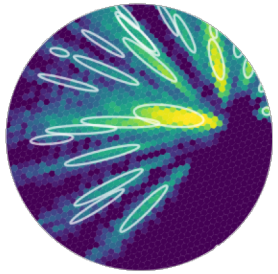
cherenkov telescope array

UNIVERSITY OF OXFORD



ctapipe

Jason J. Watson

SGSO Simulation Workshop

March 4, 2019

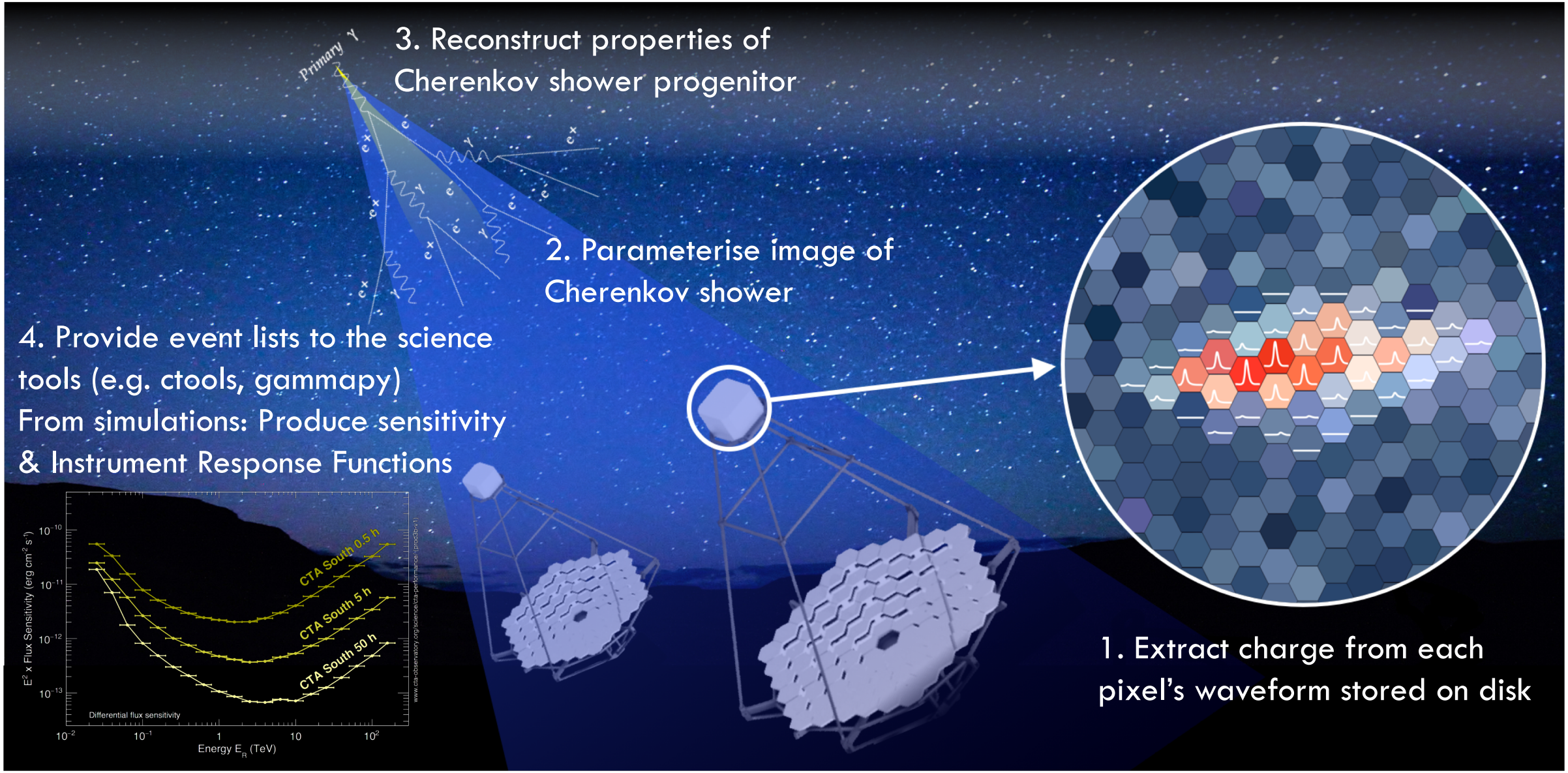
Contents

What is ctapipe?

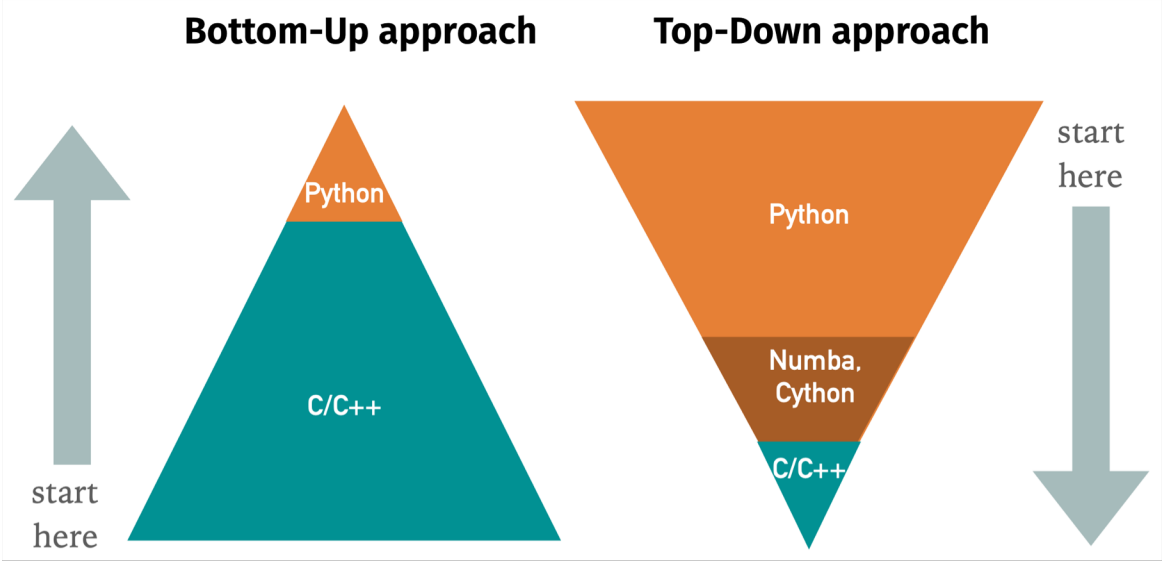
- Intention:
  - A pipeline for the offline low-level processing of CTA Cherenkov Shower data from cameras and Monte Carlo simulations
- Currently:
  - Alpha development stage (v0.6.2)
  - No full, single pipeline implemented yet
  - A library of methods useful for IACT data processing
  - Can read data from cameras and sim_telarray
- Python
- Open source: https://github.com/cta-observatory/ctapipe
- Utilises packages maintained by the large Python data science and astronomy community

What is the scope of ctapipe?

Original Illustration Credit: Richard White



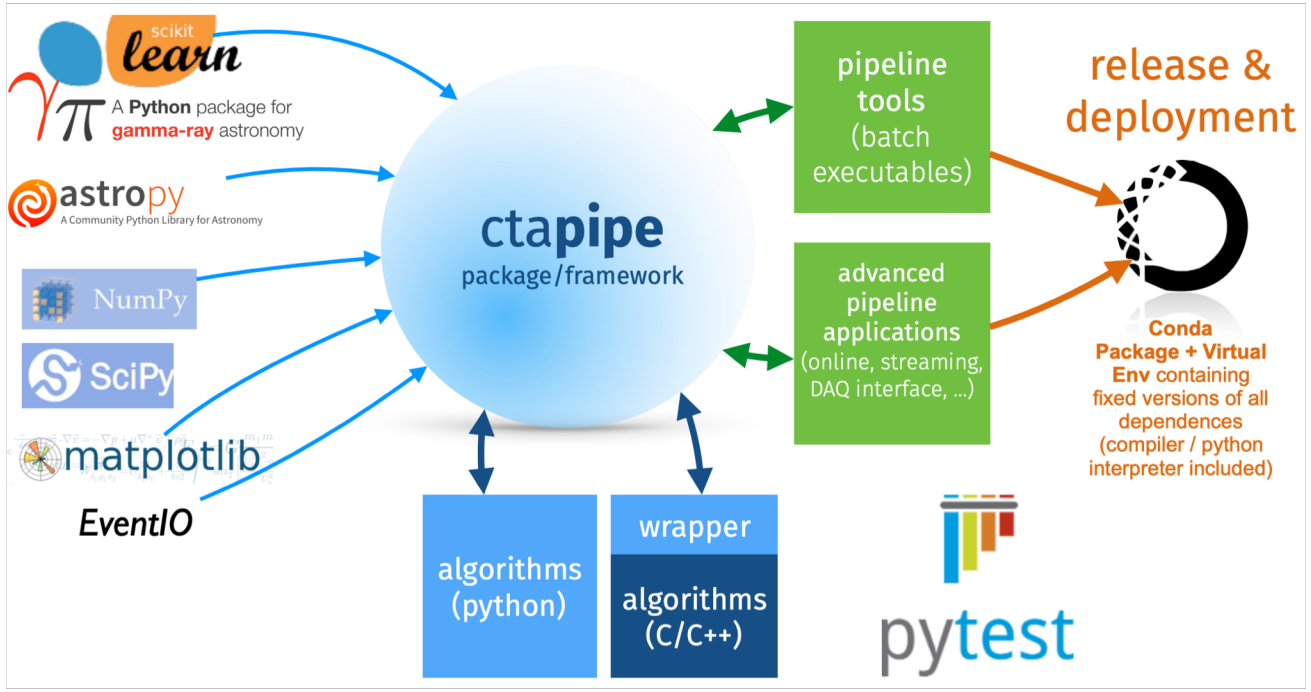
Python as a Pipeline Framework



- Pipeline frameworks often adopt a "Bottom-Up" approach, with the aim of creating statically-compiled optimised code from the start, which may be wrapped with a higher-level Python interface
- Due to its dynamically typed nature, Python is often considered slow. It is therefore unusual to use Python as the core language for a pipeline framework
- However, Python is easy to learn, easy to contribute to, and has intuitive syntax
- Python can also be extended with static compiled languages (such as C, C++ or Fortran)
- There is an extensive amount of scientific computing resources available in Python, utilising optimised low-level C and Fortran operations
- cta**pipe** has therefore adopted the "Top-Down" approach, writing Python code which utilises these extensive resources to achieve processing speeds matching (and often surpassing) hand-written static code.
- Areas requiring further optimisation are identified via profiling, and static code is written to remove bottleneck
  - "premature optimization is the root of all evil"

Python Scientific Resources

- NumPy: **the** fundamental package for scientific processing in Python, providing a contiguous, n-dimensional array object, used to pass data between statically-typed extensions.
- SciPy: expands on the operations one can perform with the NumPy array, providing extensive functionality useful for scientific computing, including statistical operations, interpolation, and signal processing.
- Astropy: developed by the astronomy community to consolidate various common astronomy procedures into a single package.
- Matplotlib: supplies extensive 2D plotting capabilities for Python, similar to those found in MATLAB, and is compatible with NumPy arrays.



Extending Python with C

- Many options!
- https://intermediate-and-advanced-software-carpentry.readthedocs.io/en/latest/c++-wrapping.html
- Numba & Cython
  - Python or "Python-like" code which is compiled into C code, allowing for an improvement in speed
  - https://jakevdp.github.io/blog/2012/08/24/numba-vs-cython/
- By hand
  - Python is itself written in C (CPython)
  - https://docs.python.org/3/extending/extending.html
- SWIG (Simplified Wrapper and Interface Generator)
  - Wraps low-level code such that methods can be called from high-level code
  - Entire libraries can easily be wrapped
- ctypes
  - Easy interface to C-written methods
  - https://pgi-jcns.fz-juelich.de/portal/pages/using-c-from-python.html

Github Version Control

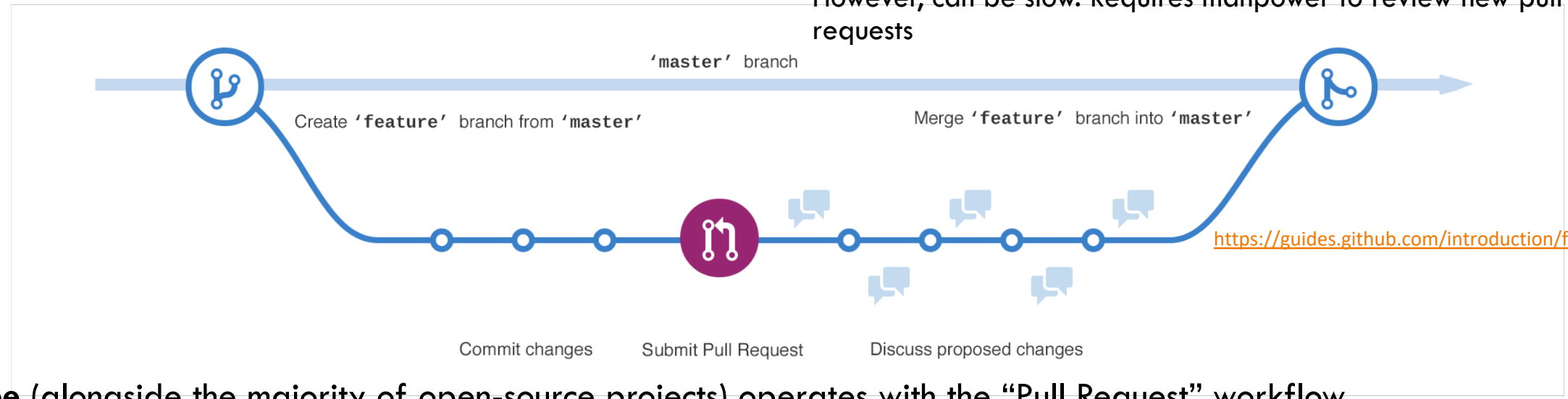
- ctapipe is open source, and is on Github: https://github.com/cta-observatory/ctapipe
- There are two typical Github workflows (that I am aware of):

Push/Pull

- Fast integration of new code
- No/limited review procedure
  - No dedicated place for discussion of contribution
  - No enforcement of coding standards
  - No ensuring of working code

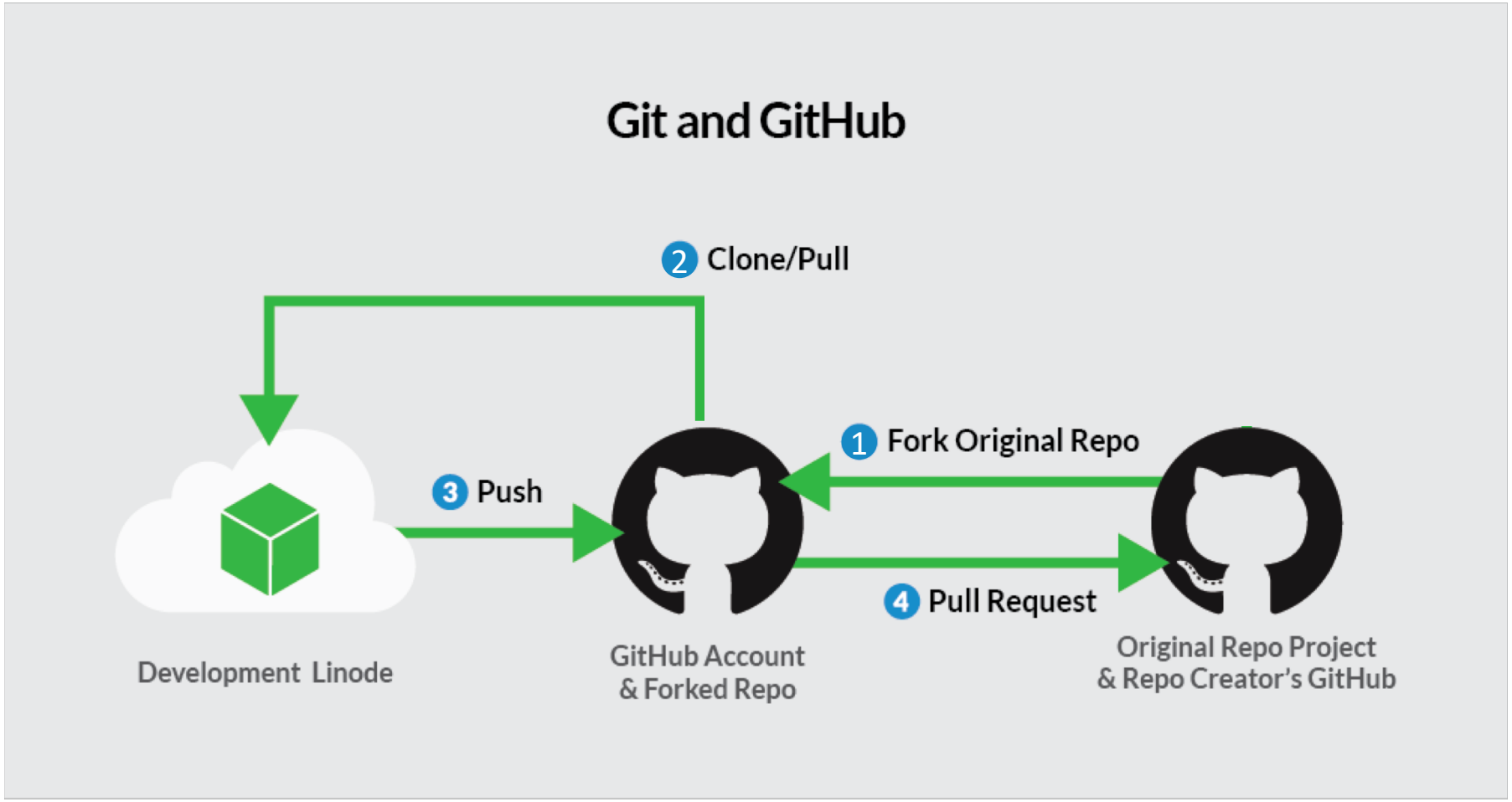
Pull Request

- New code is reviewed before acceptance into main repository
- Contributions can be discussed
- Coding standards can be enforced
- Possibility for automated unit tests to ensure new code works, and does not break existing code
- However, can be slow. Requires manpower to review new pull requests



- ctapipe (alongside the majority of open-source projects) operates with the "Pull Request" workflow

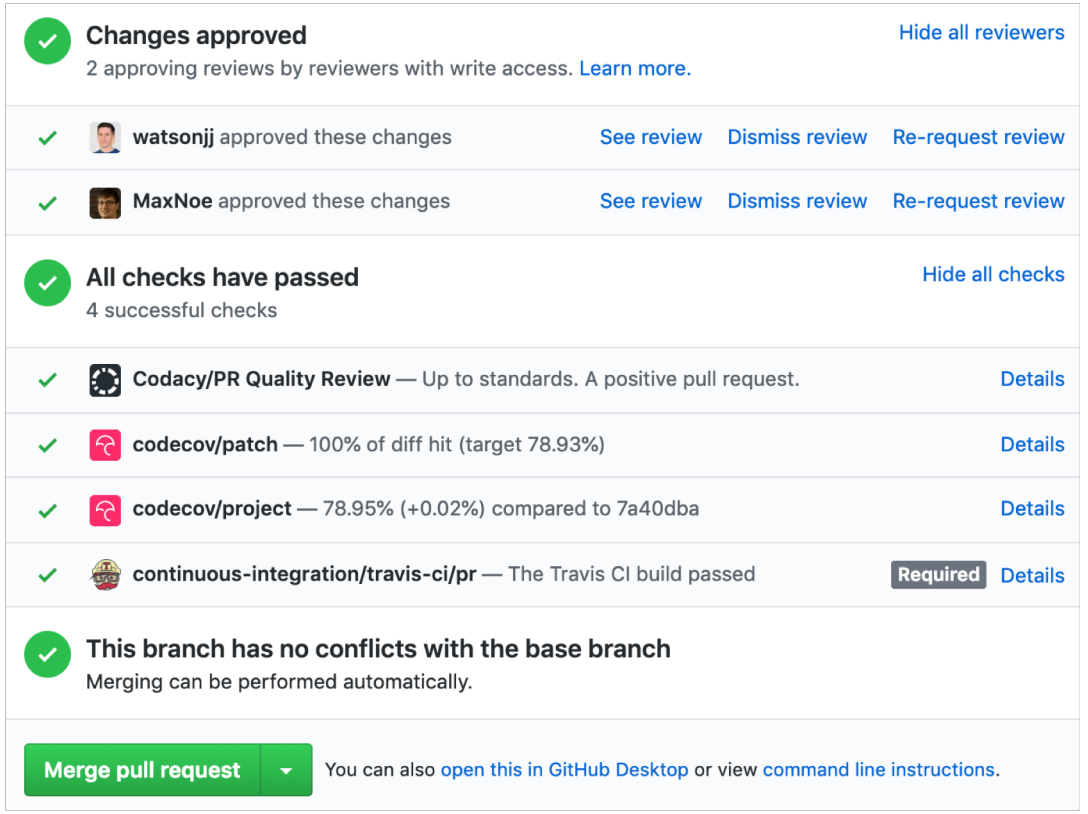
Github Workflow (Fork & Pull Request)



Modified from: https://www.linode.com/docs/development/version-control/how-to-install-git-and-clone-a-github-repository/

Using the Github Workflow in ctapipe

- The automated unit tests (TravisCI) and Pull request discussion have been paramount in a cohesive development
- Integration of new code was very slow to begin with - Only one person had permissions to accept Pull Requests
- We now (recently in the last few months) have a core-developer team (~5 people) who can review Pull Requests
- Pull requests are checked for unit tests (TravisCI), coverage (codecov), and coding standards (Codacy)
- Each Pull Request requires 2 reviews before merge
- Key to a smooth development: Lots of reviewers!



Summary

- Utilising Python for a pipeline framework in a “Top-Down” design is proving to be successful for cta**pipe**
  - The scientific resources available in Python are extensive, impressive, and easy to use

- The “Pull-Request” Github workflow is extremely reliable for collaborative development
  - Ensuring high quality and working code

- It is important to have multiple trusted reviewers of code
  - Ensuring speedy acceptance of contributions