ASKAPsoft/Yandasoft deployment in Pawsey's Setonix for ASKAP project

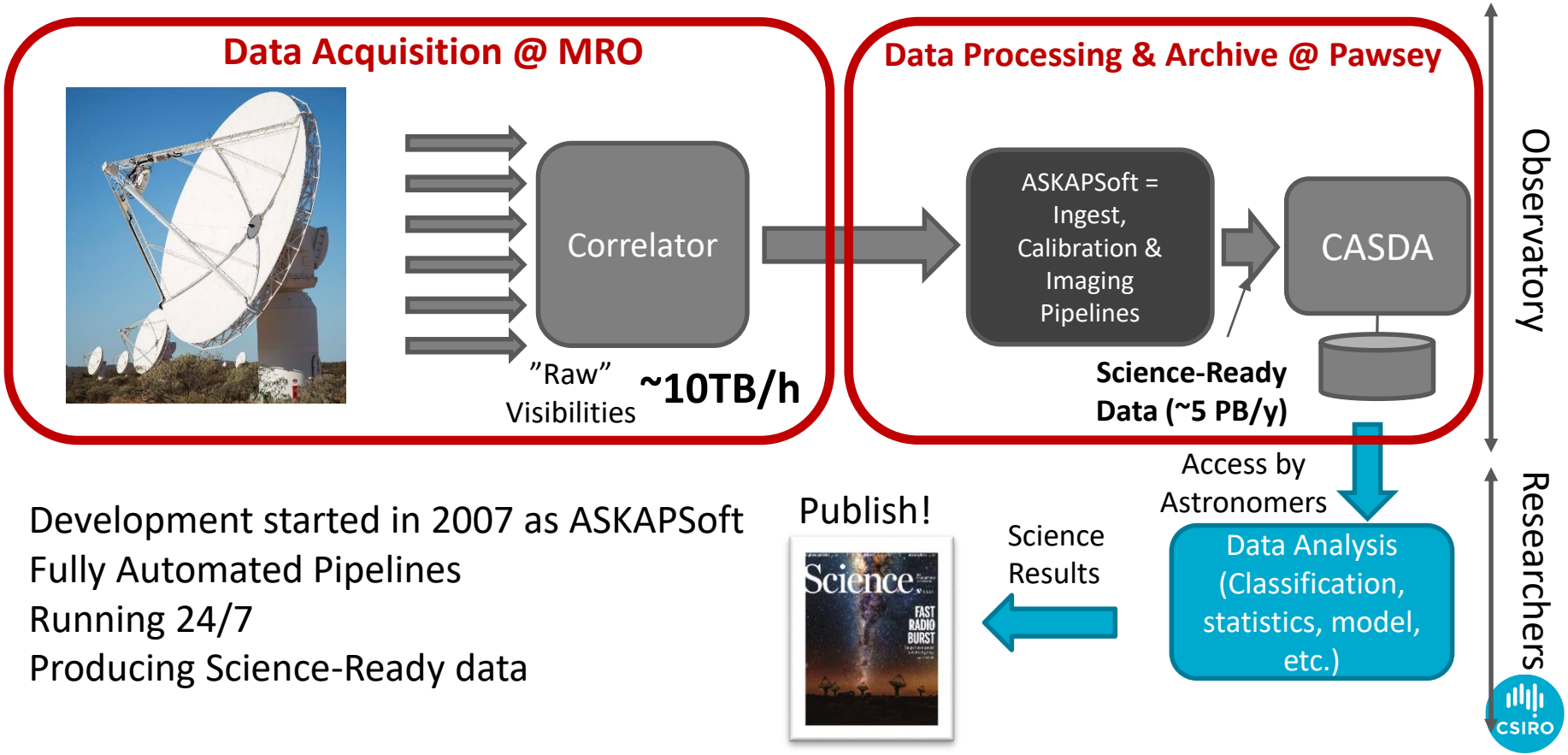
Paulus Lahur, Pascal Elahi, Eric Bastholm | 17 October 2023

ASKAP: Australian Square Kilometre Array Pathfinder

- Radio telescope
  - 36 dish antennas working together as one giant interferometer.
  - Each antenna is 12m in diameter.
  - Wide field of view: 30x larger than conventional receiver.
  - The longest distance between two antennas is 6km.
- Location
  - "Inyarrimanha Ilgari Bundara" or Murchison Radio-astronomy Observatory (MRO), WA.
  - Traditional owner: Wajarri Yamaji people.
  - Radio-quiet zone

https://www.atnf.csiro.au/projects/askap/index.html

ASKAP Data Processing

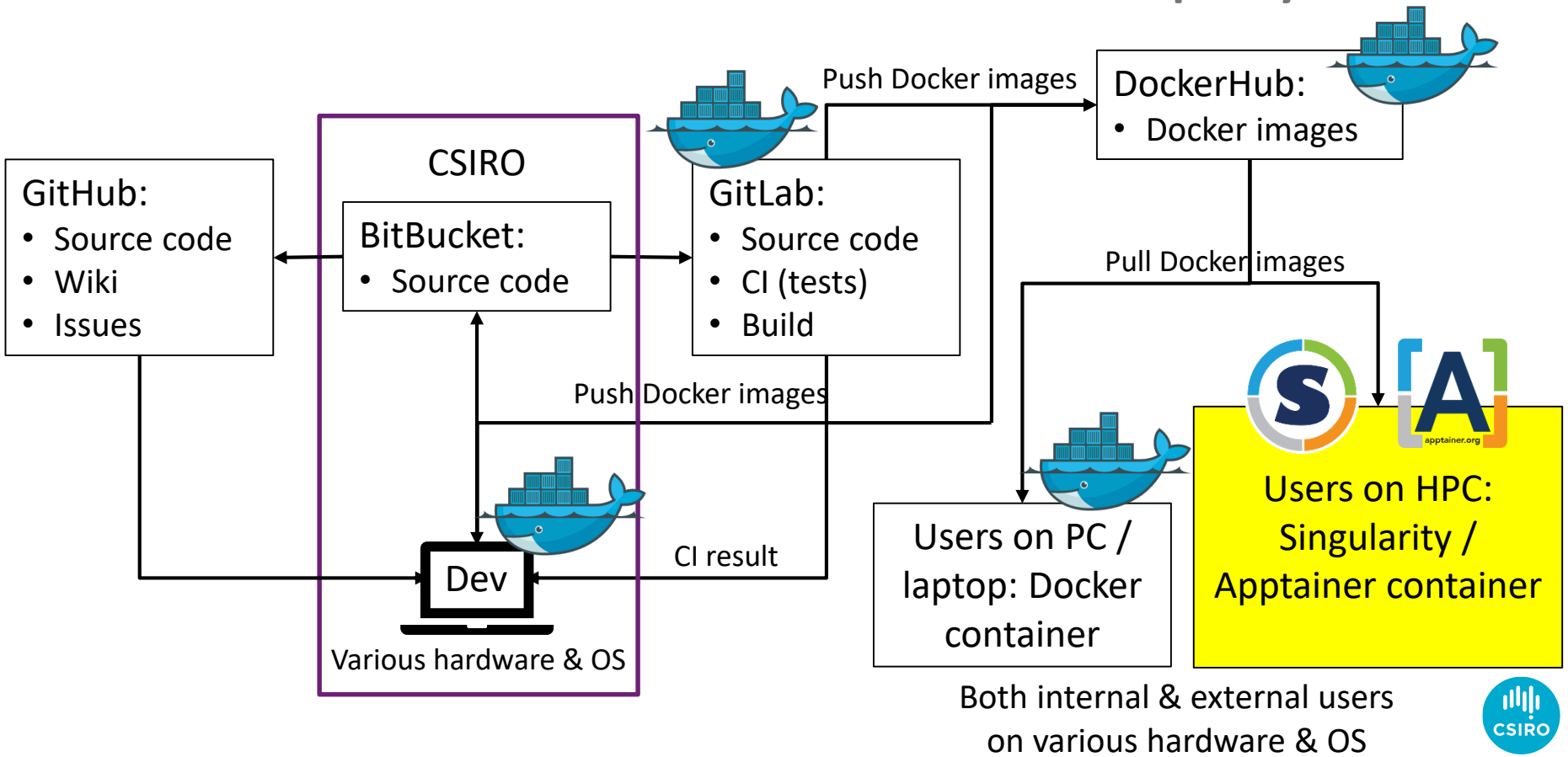


Development started in 2007 as ASKAPSoft
Fully Automated Pipelines
Running 24/7
Producing Science-Ready data

ASKAPsoft & Yandasoft

- ASKAPsoft = Yandasoft + tools specific to ASKAP
  - Ingest, Calibration and Imaging software to process large radio interferometry data in Pawsey's HPC system.
  - Custom-built in CSIRO for ASKAP, written mainly in C++
- Scientific collaborations with external institutions ➔ Collaborators need access to some functionalities of the software in their own HPC systems
- Yandasoft
  - "Yanda" = picture in Wajarri language
  - For any radio astronomy data (not just ASKAP)
  - Open source

Workflow: from Source Code to Deployment



Container and Parallel Computing using MPI

- MPI library inside container must match that of the host system
  - MPICH or OpenMPI of version 2, 3, or 4.
  - Some HPCs have their own implementation of MPICH/OpenMPI.
  - Need to build multiple containers.
- MPI & problem with dependencies
  - The MPI library depends on certain low-level libraries.
  - The low-level libraries might conflict with the ones required for other uses, e.g. for building software inside the container. In recent hardware upgrade (Setonix) this indeed happened.
  - How to resolve such conflict?

Spack: Supercomputer PACKage Manager

- Package management tool for HPC
  - Designed to support multiple versions and configurations of software on a wide variety of platforms and environments, especially HPC
- Developed originally in Lawrence Livermore National Laboratory (LLNL), USA, by Todd Gamblin in 2013. Now open source.
- Widespread use among HPC community.

```
spack install hdf5@1.10.1 %gcc@4.7.3 +debug ^openmpi+cuda fabrics=auto ^hwloc+gl
```

- Features:
  - Simple package installation
  - Custom versions, configurations & dependencies
  - Non-destructive installs: each specific install is preserved.
  - Packages can peacefully coexist: dependencies are preserved.
- https://spack.io/

CSIRO

Things are Getting More Complicated …

- Build once, deploy everywhere?
  - Does not apply to HPC (at least not without caveat).
  - Although HPC uses generic CPU & GPU, other sub-systems are highly specialised, e.g. data transfer and parallel IO (critical in large computations).
  - To deal with these, manufacturers sometimes sacrifices compatibility.
- More questions …
  - Container is supposed to simplify deployment. Why do things become more complicated instead?
  - Do We Still Need Container Then? Why don't we just build on bare metal?

Advantages of Container + Spack

- Despite the increase in complexity, there are solid advantages of using container and Spack together:
  - Dependency management: Spack manages the dependencies, while container carries all those dependencies inside. Not having to solve dependency problem saves time!
  - Portability: Container can be used on various systems (with certain limits).
  - Reproducibility: Both Spack and Container have recipes that can be used to build a specific version of software. It can be built on various systems.
  - Collaboration: The advantages above allow collaborators to work on the same application and environment, even if they are running on different systems. This might be very hard to achieve if we are to build on bare metal.

Managing Apparent Complexity

- Complexity: apparent vs actual.
- Automate everything that can be automated.
  - Nightly build of dev containers.
- Hide complexity behind simple interface.
  - Script to build various containers in a similar way.
- Make use of multi-stage container build process to fit certain roles.
  - HPC support team: provides (generic) base containers.
  - Container providers: use base container to build dev & production containers. For another HPC, swap only the base container.
  - Developers: work in dev container (also on bare metal).
  - End-users: use the software inside production container.

Concluding Remarks

- ASKAPsoft/Yandasoft uses both container technology and Spack (Supercomputer PACKage Manager) for deployment on Setonix.
- The increase of complexity in tools is more than compensated by significant improvement in dependency management, portability and reproducibility.
- Next steps
  - Consolidate and unify the approach.
  - More automation & flexibility for developers.
  - Improve interface.

Acknowledgement

- People: Space and Astronomy (S&A)
  - ASKAPsoft/Yandasoft Dev Team: Daniel Mitchell, Eric Bastholm, Fatemeh Moghaddam, Juan Guzman, Mark Wieringa, Matt Austin, Matthew Whiting, Max Voronkov, Minh Vuong, Ozgur Cekmer, Paulus Lahur (IMT), Pero Manojlovic, Randika Hewage, Rodrigo Tobar, Stephen Ord, Tim Galvin, Vitaliy Ogarko, Wasim Raja
  - ASKAP Team
  - Pawsey: Pascal Elahi, Maciej Cytowski & the rest of Pawsey Team
  - Information Management & Technology (IMT)
- About the authors
  - Pascal Elahi: Setonix supercomputer in Pawsey, base containers, Spack
  - Eric Bastholm: oversees the entire DevOps process for ASKAPsoft
  - Paulus Lahur: ASKAPsoft & Yandasoft containers

CSIRO

Thank You!

- Further information
  - ASKAP: https://www.atnf.csiro.au/projects/askap/index.html
  - Pawsey: https://pawsey.org.au/
  - Yandasoft on GitHub: https://github.com/ATNF/yandasoft
- Email: Paulus.Lahur@csiro.au

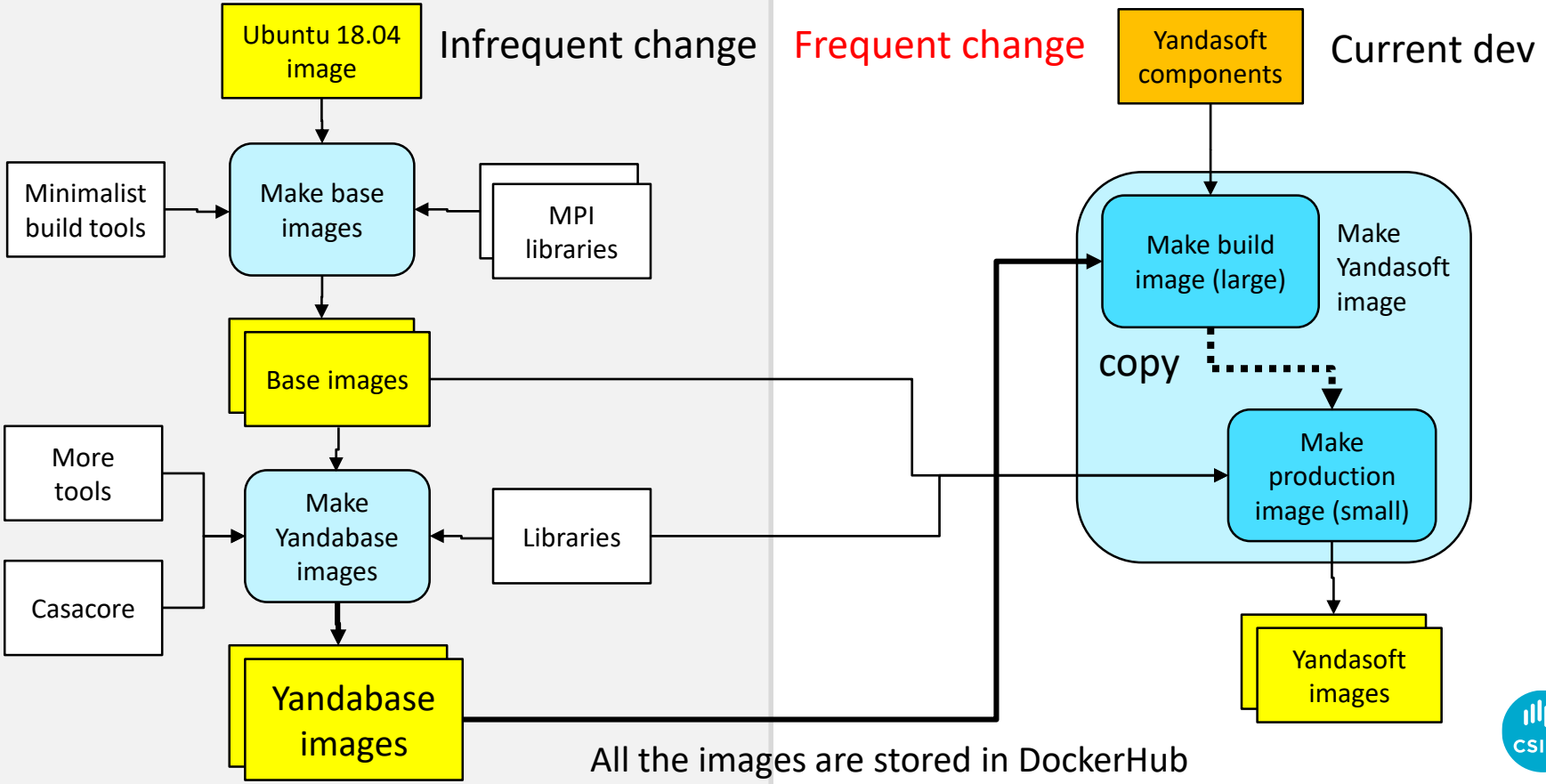
Extra Pages

Pawsey: Setonix

- Manufacturer: HPE Cray
- Model: EX Supercomputer
  - 1592 Dual 2.45GHz AMD EPYC 7763 “Milan” 64-Core CPU Nodes with 256 GB RAM
  - 8 Dual 2.45GHz AMD EPYC 7763 “Milan” 64-Core CPU Nodes with 1 TB RAM
  - 154 Single AMD EPYC 7A53 “Trento” 64-Core GPU Nodes with eight AMD Instinct MI250X GPUs and 256 GB RAM
  - 38 Single AMD EPYC 7A53 “Trento” 64-Core GPU Nodes with eight AMD Instinct MI250X GPUs and 512 GB RAM
  - Eleven Data mover nodes
  - 31 Visualization nodes
  - Nine Login nodes
  - Connected by HPE’s Slingshot interconnect (200Gb/sec)
  - Lustre file systems: `/scratch` (14 PB), `/software`
  - NFS: `/home`

CSIRO

Yandasoft's Multi-Stage Workflow



Science Projects

ASKAP's survey science projects involves >700 astronomers from >200 institutions around the world

- CRAFT (The Commensal Real-time ASKAP Fast Transients survey)
- DINGO (Deep Investigations of Neutral Gas Origins)
- EMU (Evolutionary Map of the Universe)
- FLASH (The First Large Absorption Survey in HI)
- GASKAP-HI (The Galactic ASKAP Spectral Line Survey - Neutral Hydrogen)
- GASKAP-OH (The Galactic ASKAP Spectral Line Survey - ground-state Hydroxyl)
- POSSUM (Polarization Sky Survey of the Universe's Magnetism)
- VAST (An ASKAP Survey for Variables and Slow Transients)
- WALLABY (Widefield ASKAP L-Band Legacy All-Sky Blind Survey)

More info: https://www.atnf.csiro.au/projects/askap/science.html

Continuous Integration (CI)

- Built by Stephen Ord
- Workflow:
  1. Change in source code is pushed by a developer to repository in BitBucket
  2. The change is mirrored to GitLab repository
  3. GitLab's CI is triggered
  4. Result (success / fail) is reported in developers' chat channel. Link back to CI pipeline.
  5. If debug is required, artifacts from build and tests can be downloaded
- Major parts in CI pipeline:
  1. Build: from a reference Docker image of Yandasoft
  2. Unit tests
  3. Functional tests
- Everything works behind the scene. For developers, it just works.

CSIRO