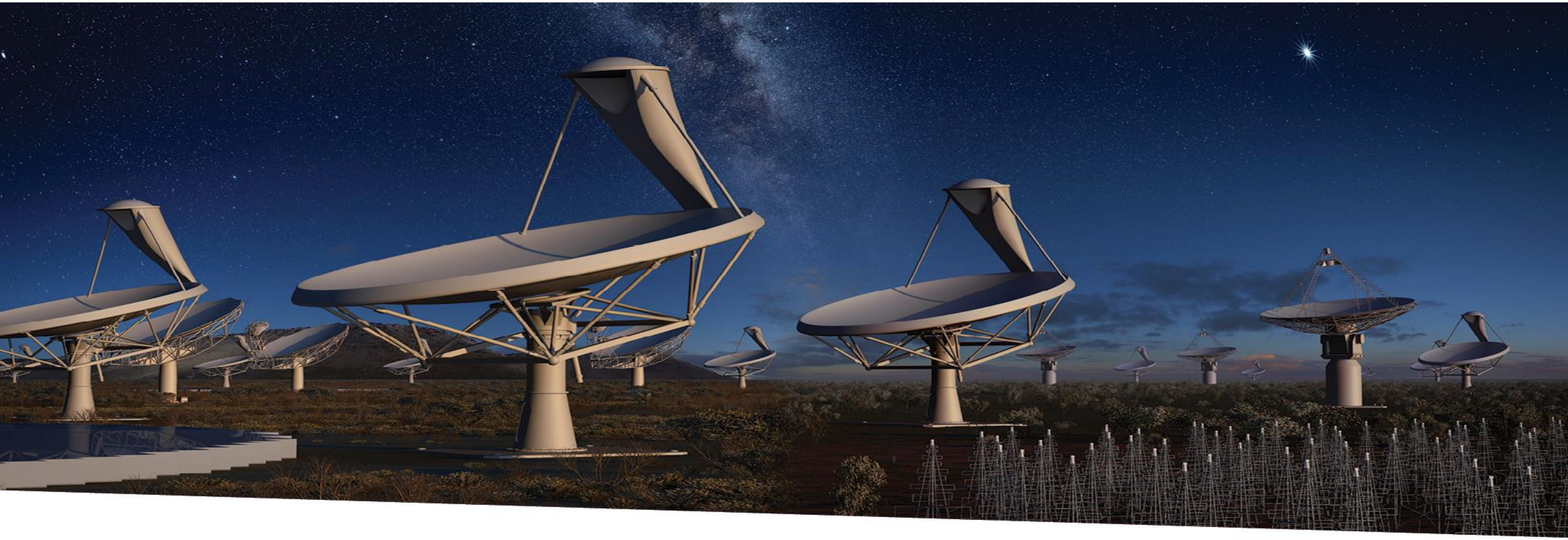
CI-CD Practices with the TANGO-controls framework in the context of the Square Kilometre Array (SKA) Telescope Project



SQUARE KILOMETRE ARRAY

Exploring the Universe with the world's largest radio telescope

Matteo Di Carlo (INAF-OAAB)
Yilmaz U., Harding P., Bartolini M.,
Le Roux G., Dolci M.

SKA Project

- International effort to build two radio interferometers in South Africa and Australia
- One Observatory monitored and controlled from the global headquarters (GHQ) based in the United Kingdom at Jodrell Bank
- Software development process is Agile
  - Mainly incremental and iterative
  - Many teams (18) with a specialized team (known as system team) devoted to support the continuous Integration, test automation and continuous Deployment.

Why CI-CD?

- When many parts of the project are developed independently for a long period of time (weeks or longer),
- Code base and build environments diverges
- When changes are integrated
  - Weeks in verifying that everything works
  - Developers spend time in solving bugs introduced months earlier

Continuous integration

- Set of development practices that requires developers to integrate code into a shared repository several times a day.
- Each check-in is then verified by an automated build, allowing teams to detect problems early.

The practices

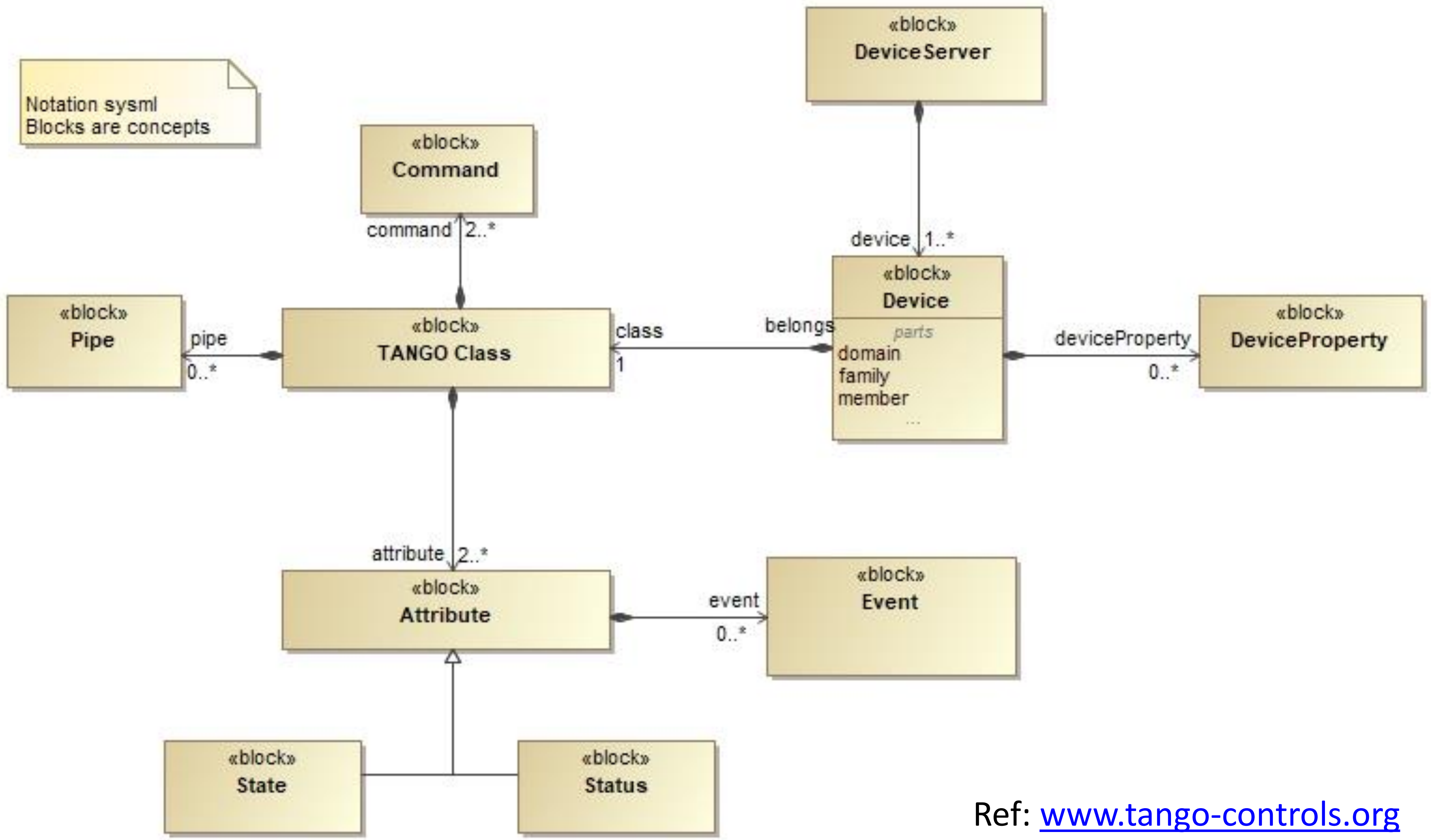
- Single source repository (for each component of the system)
  - minimize the use of branching
- Automate the build (build all in one command)
- Automate testing (together with the build)
- Every commit should build on an integration machine
  - Commit often! (at least once per day)
  - the smaller is the change the easier is the fix
- Build fast (so that a problem in integration can be found quickly)
- Multi-stage deployment: every build software must be tested in different environments

Ref: martinfowler.com/articles/continuousIntegration.html

Delivery vs Deployment

- Continuous delivery
  - Automate the delivery of new releases of software
  - Deployment has to be predictable and sustainable
    - The code must be in a deployable state
    - **Testing** needs to cover enough of your codebase.
- Continuous deployment
  - One step further: every single commit to the software that passes all the stages of the build and test pipeline is deployed into the production environment

TANGO-controls framework

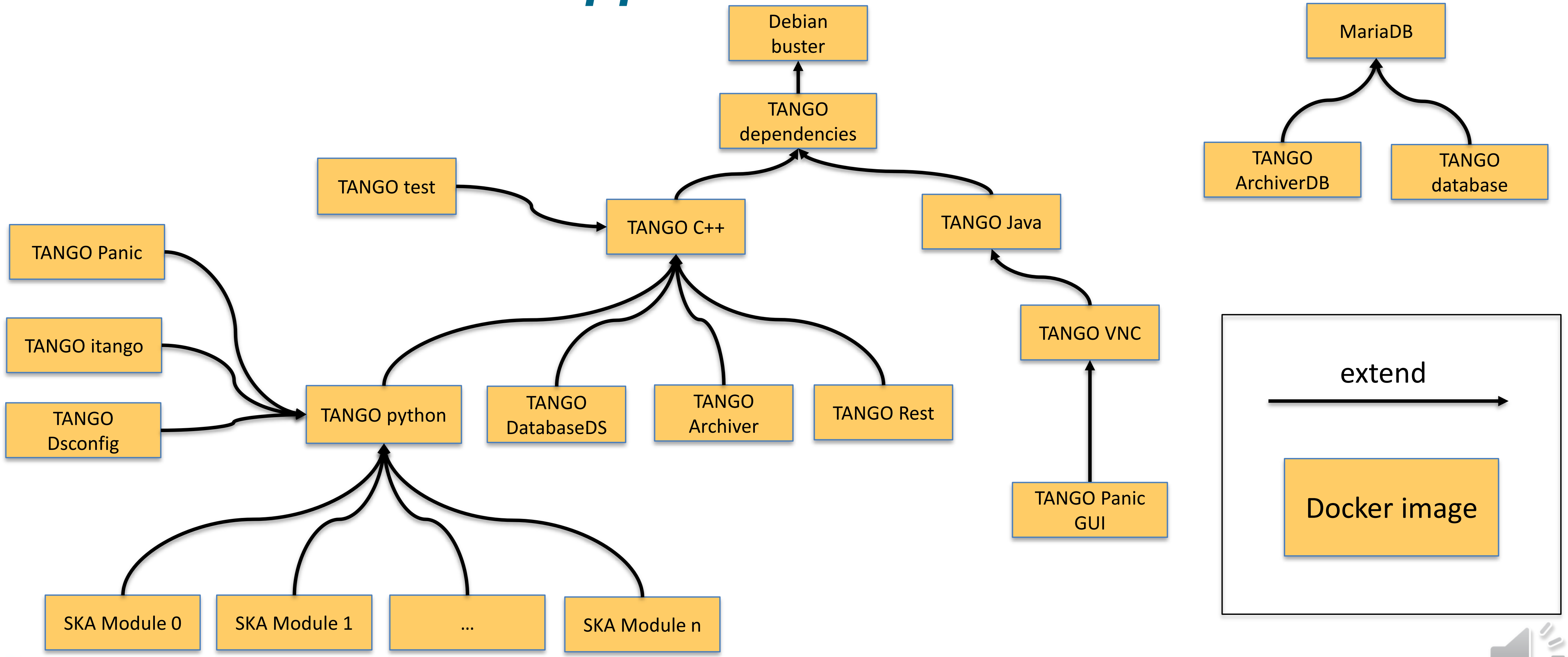


Ref: www.tango-controls.org

Containerization

- SKA == set of elements == a set software modules
- For each module there is one repository
- For each repository there is one docker image
  - – convenient way to package up applications and preconfigured server environments

SKA-Docker - Containerized environment for TANGO-controls application



Kubernetes and Helm

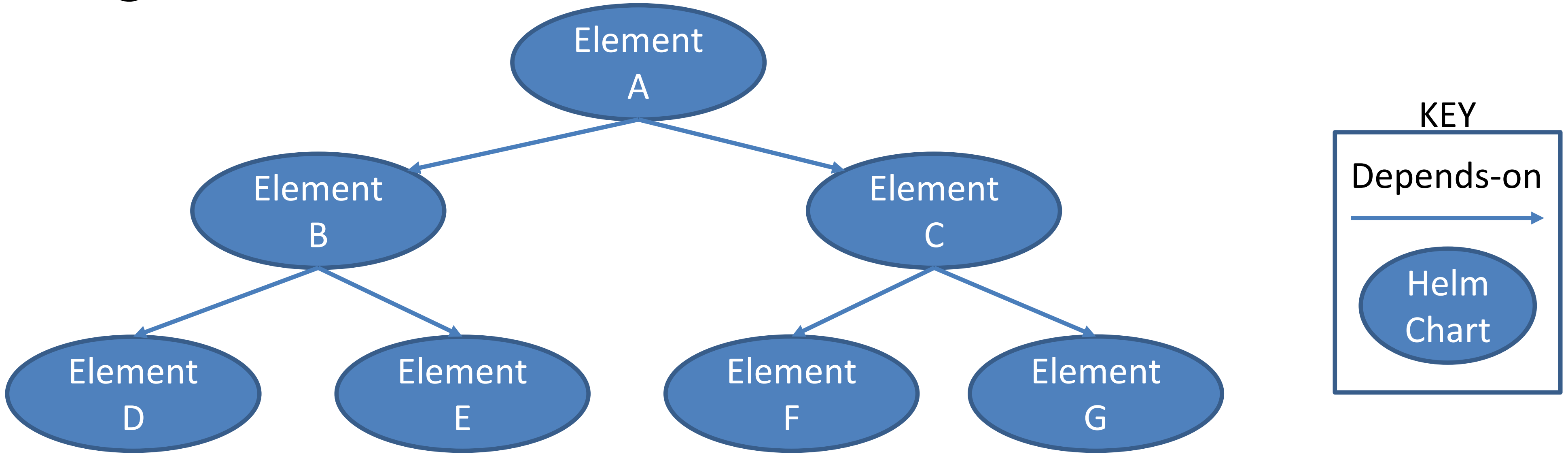
- Kubernetes (k8s) for container orchestration (kubernetes.io)
  - Service == TANGO Device Server
- *Helm for packaging SKA k8s applications (helm.sh)*
  - Tool for managing Kubernetes charts
  - Chart is a package of pre-configured Kubernetes resources (set of information for running a Kubernetes application)

For each SKA element there must be an helm chart for running it in k8s!

Use of Makefiles for lifecycle management (one command for build images, start application using helm, test application and clean)!

Integration with Helm

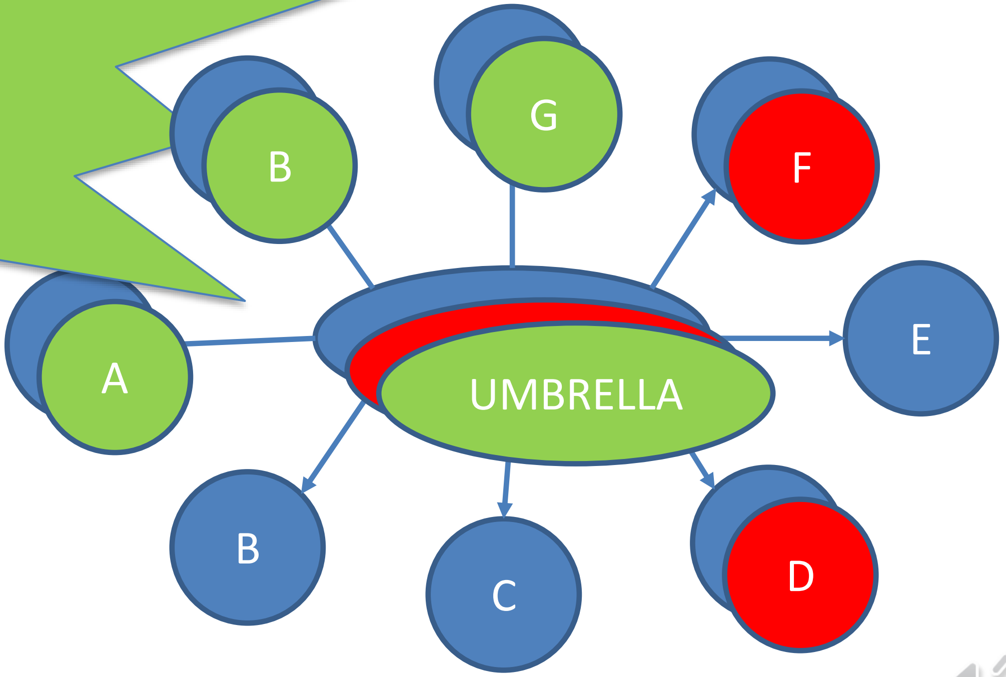
- Helm has the concept of dependency
  - An helm chart can have one or more sub-charts
- The integration of SKA elements can be done with this concept



Helm sub-charts Architecture

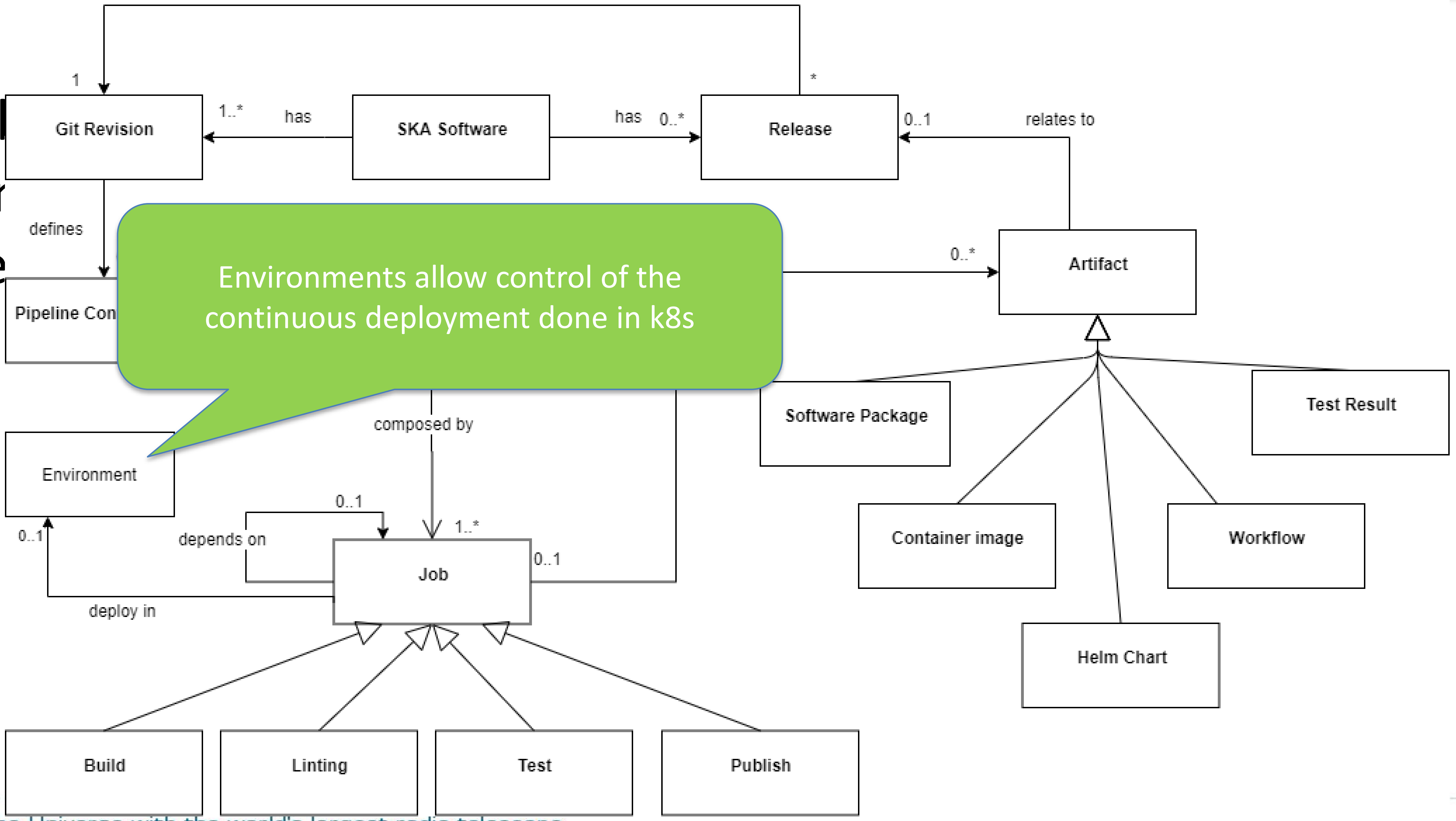
- Operational aspects of using depe dencies: the sub-charts are
  - aggregated into single s
  - sorted by type fo
  - created/upd

For every SKA element, there is at least an umbrella chart for integration testing



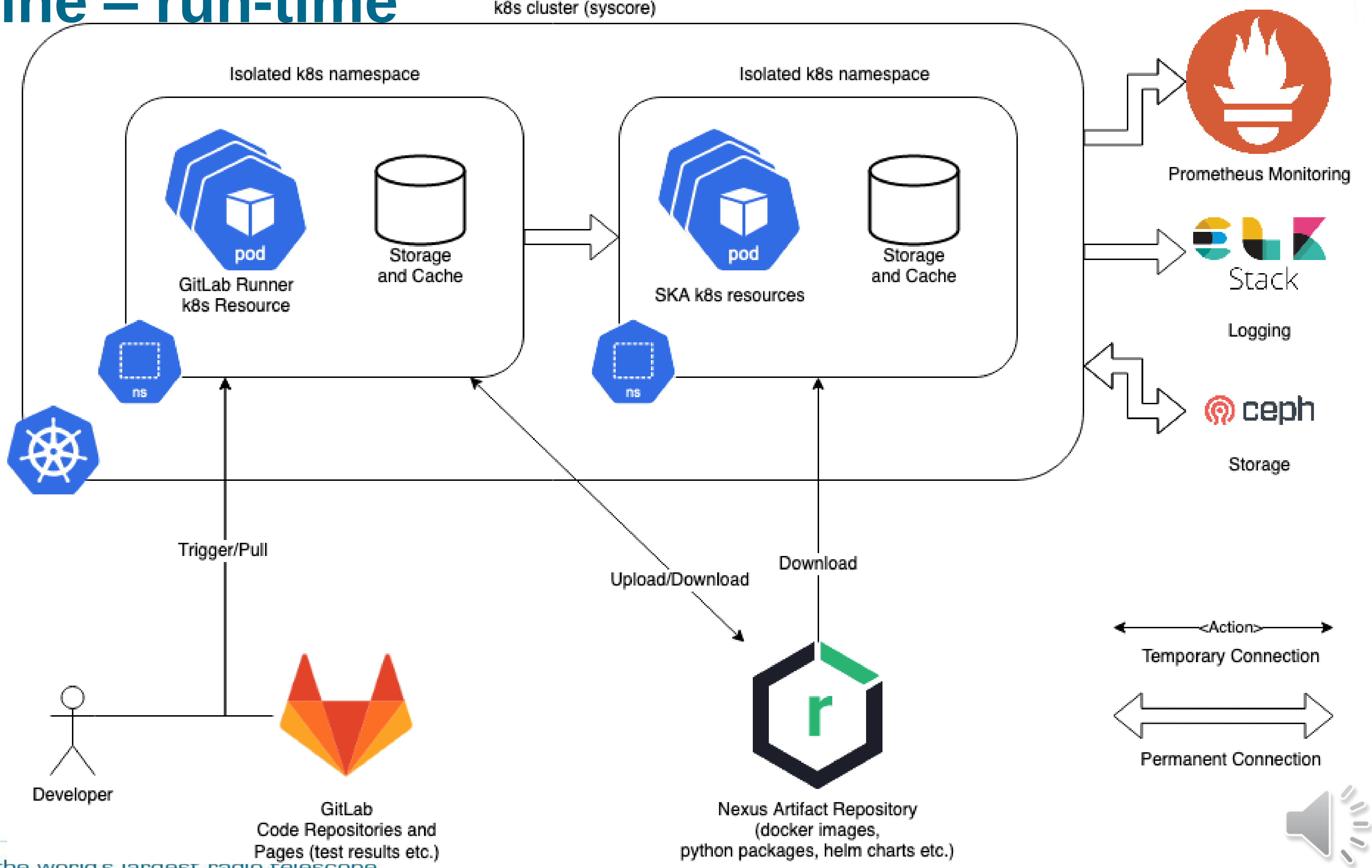
GitLab

- Wel
  mar
  inte



Environments allow control of the continuous deployment done in k8s

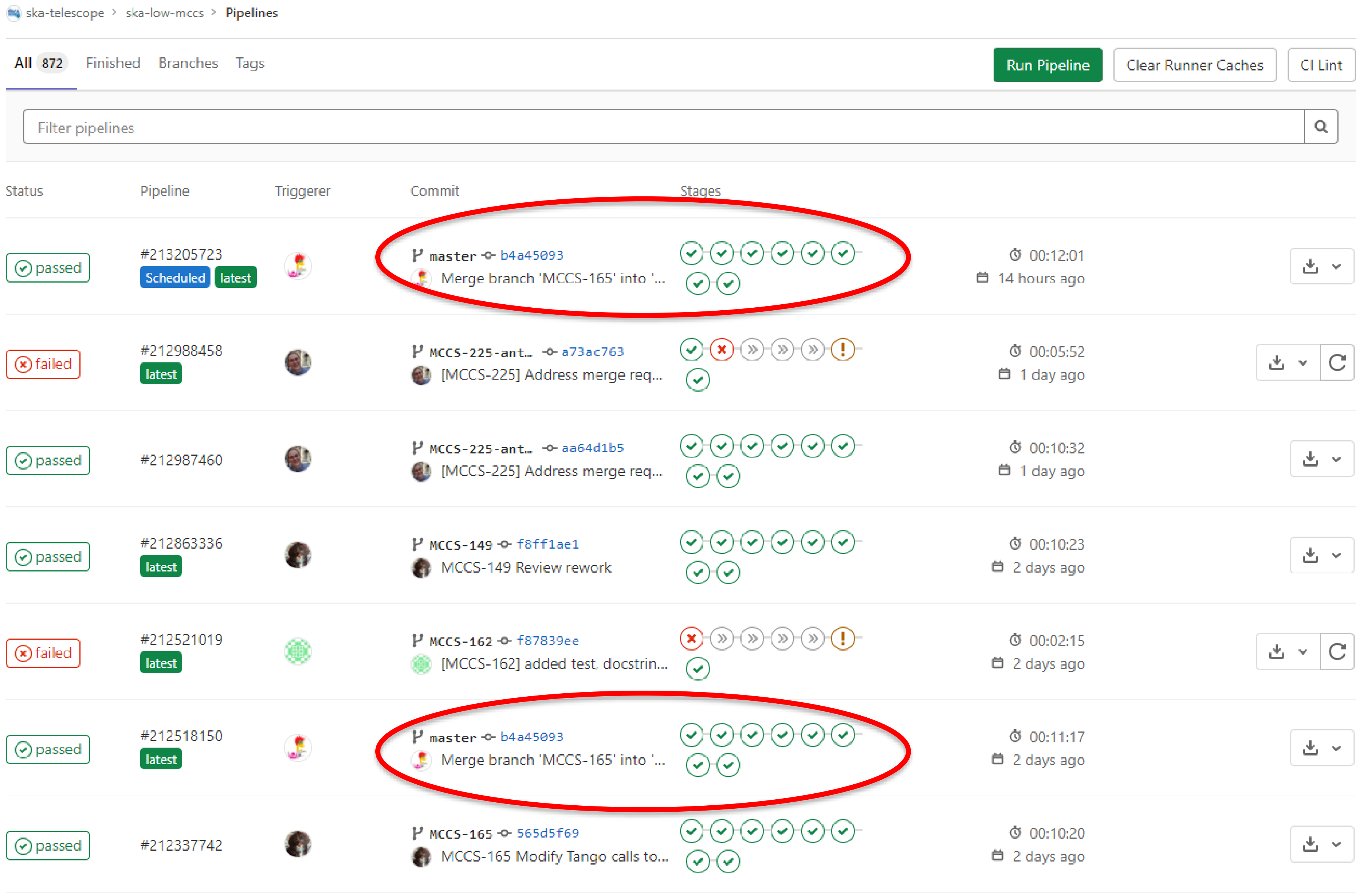
Gitlab pipeline – run-time



Deployment and Testing

- For each repo and for each commit (!):
  - – install the (umbrella) chart in an isolated namespace
  - – wait for every container to be running
  - – For the tests:
    - Create a k8s pod (a container) in the isolated namespace
    - Run pytest inside the above pod
    - Return the tests results
  - – uninstall the (umbrella) chart

Example pipeline



ska-telescope › ska-low-mccs › Pipelines

All 872 | Finished | Branches | Tags — Run Pipeline | Clear Runner Caches | CI Lint

Filter pipelines

Status	Pipeline	Triggerer	Commit	Stages	
passed	#213205723 Scheduled latest		master b4a45093 Merge branch 'MCCS-165' into '...	✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	00:12:01 14 hours ago
failed	#212988458 latest		MCCS-225-ant... a73ac763 [MCCS-225] Address merge req...	✓ ✗ » » » ! ✓	00:05:52 1 day ago
passed	#212987460		MCCS-225-ant... aa64d1b5 [MCCS-225] Address merge req...	✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	00:10:32 1 day ago
passed	#212863336 latest		MCCS-149 f8ff1ae1 MCCS-149 Review rework	✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	00:10:23 2 days ago
failed	#212521019 latest		MCCS-162 f87839ee [MCCS-162] added test, docstrin...	✗ » » » » ! ✓	00:02:15 2 days ago
passed	#212518150 latest		master b4a45093 Merge branch 'MCCS-165' into '...	✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	00:11:17 2 days ago
passed	#212337742		MCCS-165 565d5f69 MCCS-165 Modify Tango calls to...	✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	00:10:20 2 days ago

Explori

Conclusion

- Single source repository (for each component of the system)
  - minimize the use of branching
- Automate the build (build all in one comm
- Automate testing (together with the build)
- Every commit should build on an integration machine
- Build fast (so that a problem in integration
- Multi-stage deployment: every build software must be tested in different environm

Every component has its own repository with story based branching (2 weeks lifecycle)

Large use of Makefile

Every commit trigger a GitLab pipeline that build, lint, execute unit-testing and integration testing

It depends: building the ska-docker images can take 30 minutes, while testing takes up to 10 minutes

Integration testing in Gitlab is done within an isolated k8s namespace which are completely separate each other
One namespace is kept as "blessed" environment

SQUARE KILOMETRE ARRAY

Exploring the Universe with the world's largest radio telescope

Thanks