CI-CD Practices with the TANGO-controls framework in the context of the Square Kilometre Array (SKA) Telescope Project
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
• Agile development process is Agile
  – Mainly incremental and iterative
  – Specialized team (known as system team) devoted to support the continuous Integration, test automation and continuous Deployment.
TANGO-controls framework

Ref: www.tango-controls.org
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.
CI – In practice

• 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 (testing, staging and so on)

Ref: martinfowler.com/articles/continuousIntegration.html
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.

• “If it hurts, do it more often, and bring the pain forward”
  – Ref: J. Humble, D. Farley, "Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation"
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.

• “If it hurts, do it more often, and bring the pain forward”
  – Ref: J. Humble, D. Farley, "Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation"

![Diagram of Blue-green deployment with Router, Active environment, and Not active environment]
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.
Containerization

• SKA can be seen as a set of elements which can be seen as a set software modules
  – Every module is a component with its own repository

• For each component there is one docker image
  – standard unit of software that packages up code and all its dependencies so the component runs quickly and reliably across different computing environments

• Each container includes one application (device server)

• Containerized TANGO environment
  – in order to allow all the teams to work in the same environment
    – Makefile for starting and testing (one single command “make test”)

The integration environment is based on Kubernetes (k8s) orchestration and Helm

Helm is a tool for managing Kubernetes charts

Chart is a package of pre-configured Kubernetes resources (set of information necessary to create an instance of a Kubernetes application)

Ref:
- kubernetes.io
- helm.sh
K8s and Helm Repository

Integration repository

- k8s template
- json device configuration
- helm chart

Environment (cloud, openstack, ...)

KEY

- CICD Infrastructure
- a file or set of files
- Environment (cloud, openstack, ...)

Data flow

Composition
K8s and Helm Repository

One folder (chart) for each team

Includes images versions

Integration tests runs in a chart

k8s template

json device configuration

helm chart

Integration repository

KEY

CICD Infrastructure

a file or set of files

Environment (cloud, openstack, ...)

data flow

composition
K8s and Helm Repository

One folder (chart) for each team

Includes images versions

Integration tests runs in a chart

Integration repository

k8s template

json device configuration

helm chart

GitLab + Ansible

Environment (cloud, openstack, ...)

data flow

composition

KEY

CICD Infrastructure

a file or set of files
Infrastructure – runtime view

KEY

Source-Code repository

server

code repository

Artifacts repository

request-reply

Exploring the Universe with the world’s largest radio telescope
Infrastructure – runtime view

Source-Code repository

CICD Server

every 5 minutes or manually

KEY

server request-reply code repository Artifacts repository
Infrastructure – runtime view

Source-Code repository

CICD Server

every 5 minutes or manually

every 3 seconds

CI Executor

KEY

server

code repository

Artifacts repository

request-reply

Exploring the Universe with the world's largest radio telescope
Infrastructure – runtime view

Source-Code repository

CICD Server

every 5 minutes or manually

every 3 seconds

CI Executor

docker images, python packages, other packages

Artifacts repository

KEY

request-reply

server  code repository  Artifacts repository

Exploring the Universe with the world's largest radio telescope
Infrastructure – runtime view

Source-Code repository

CICD Server

CI Executor

donner images, python packages, other packages

Environment

Artifacts repository

every 5 minutes or manually

every 3 seconds

KEY

server code repository Artifacts repository

request-reply

Exploring the Universe with the world's largest radio telescope
Infrastructure – runtime view

- **Source-Code repository**
- **CICD Server**
- **Gitlab Runner**
- **Nexus**

**CI Executor**
- request-reply every 5 minutes or manually
- request-reply every 3 seconds

**Environment**
- K8s
- Code repository
- Artifacts repository

**KEY**
- server
- code repository
- Artifacts repository

docker images, python packages, other packages

Deploy
Dashboard

• Monitor the projects in SKA with at least the following information:
  – Latest build status and date
  – Latest green build date
  – Test coverage
  – Testing report

• Progressive web application
  – Automatic deployment
  – Pipeline with 2 phases: retrieve data and generate
Monitor the projects in SKA with at least the following information:

- Latest build status and date
- Latest green build date
- Test coverage
- Testing report

Progressive web application

- Automatic deployment
- Pipeline with 2 phases: retrieve data and generate Dashboad
Conclusion

• It’s work in progress:
  – Improve performance of the tests
  – more environments

• But we already implemented many best practices:
Conclusion

• It’s work in progress:
  – Improve performance of the tests
  – more environments

• But we already implemented many best practices:
  – One component-one repository
    • minimal use of branching (short lived)
  – Build (and testing) of each component is automated
  – Every commit triggers a build
  – Artifacts transferred in an integration repository
    • which run a kubernetes cluster and more tests are done
  – Common repository for artifacts and test results
    • very easy to download the latest changes for each component
  – Integration environment is accessible for every developer
Thanks