I would like to begin by acknowledging

the **Ngunnawal and Ngambri** people as the Traditional Owners of the land here in Canberra that I we are on today,

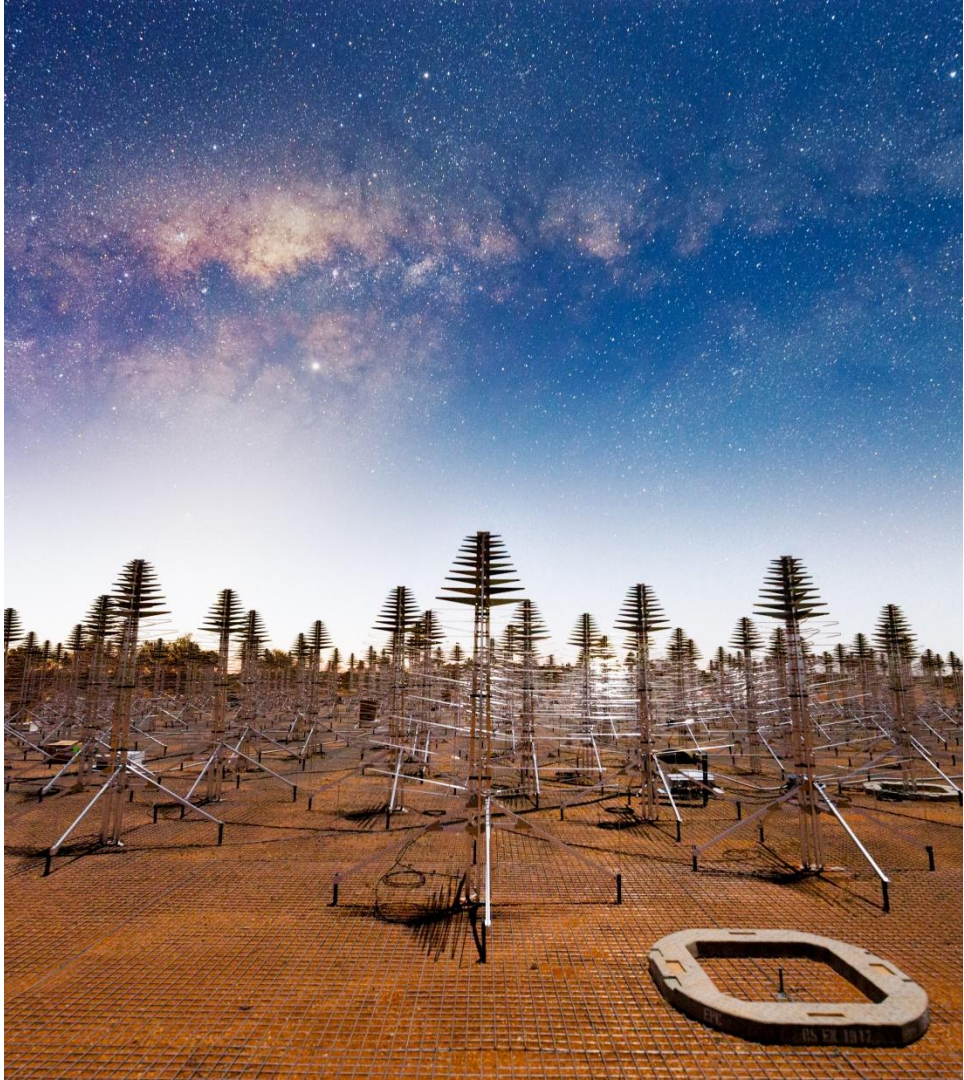
the **Wajarri Yamatji** people who are the Traditional Owners of the land on which the MRO stands, and

and pay my respect to all their Elders past and present.

RACS by Wajarri artist Margaret Whitehurst, 2019 (detail)

Software Testing in SKAO

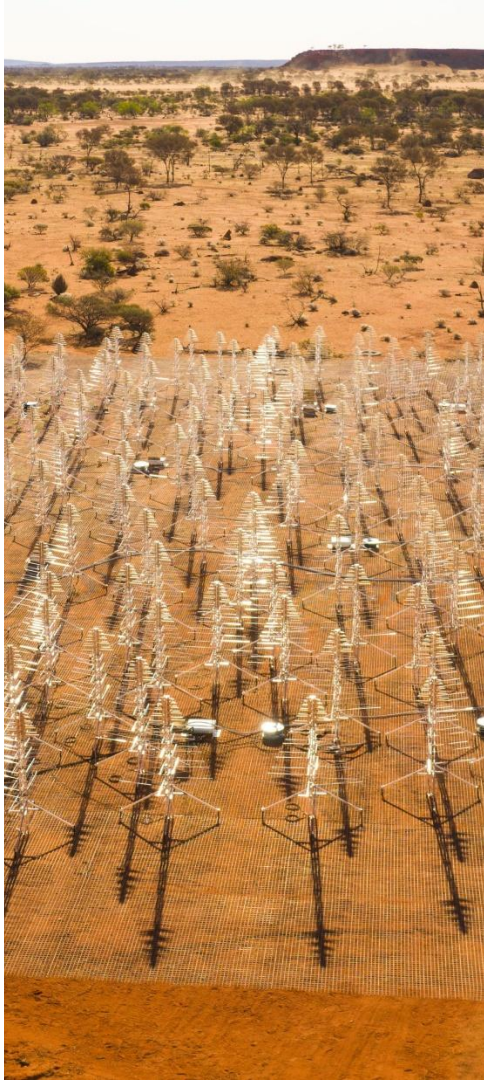
Malte Marquarding CSIRO
ATF CoP 2022



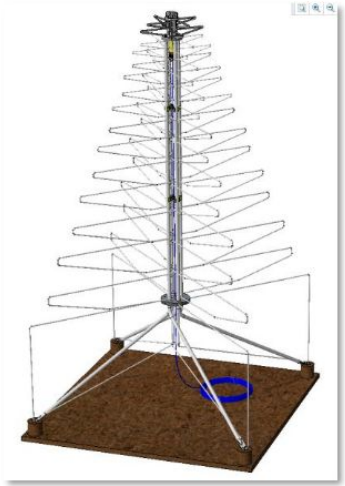
Milky Way overhead the AAVS station
Credit: Michael Goh and ICRAR/Curtin.

SKA overview

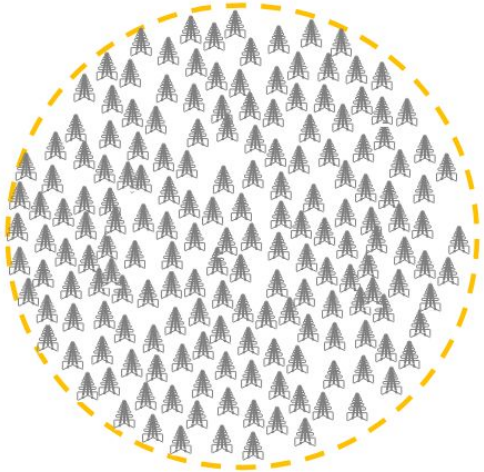
- SKAO – International Governmental Organisation
- Two radio-frequency telescopes
  - MID (South Africa)
  - LOW (Australia)
- Headquarter (Jodrell Bank, UK)
- Under construction
- First “release” target - Array Assembly 0.5



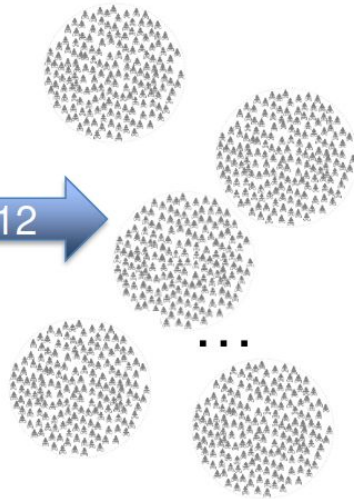
The Aperture Array Verification System (AAVS 2.0)
Credit: ICRAR / Curtin University



256



512



SKA1-Low
Antenna/Receptor

Antenna Beam

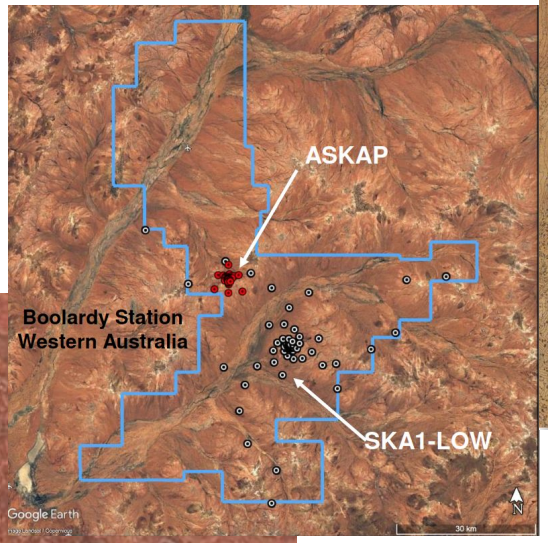
SKA1-Low
"Station"

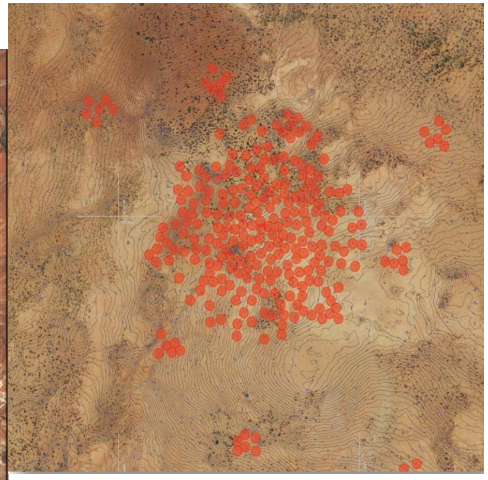
Station Beam

SKA1-Low
"Array"

Correlation and
Tied-array Beams


Carnarvon
Perth
Mandurah
Bunbury
Busselton
Cervantes
Esperance
Plumridge Lakes
Geraldton
Kalgoorlie
• SKA1-LOW: 50 – 350 MHz
• Phase 1: ~130,000 antennas
• across 65km


ASKAP
Boolardy Station
Western Australia
SKA1-LOW
Google Earth
30 km
N



CSIRO

Jama software

Bentley Software

All Software / AssetWise ALIM

TASKTOP

ATLASSIAN Jira

Read the Docs

python

TANGO

Cucumber

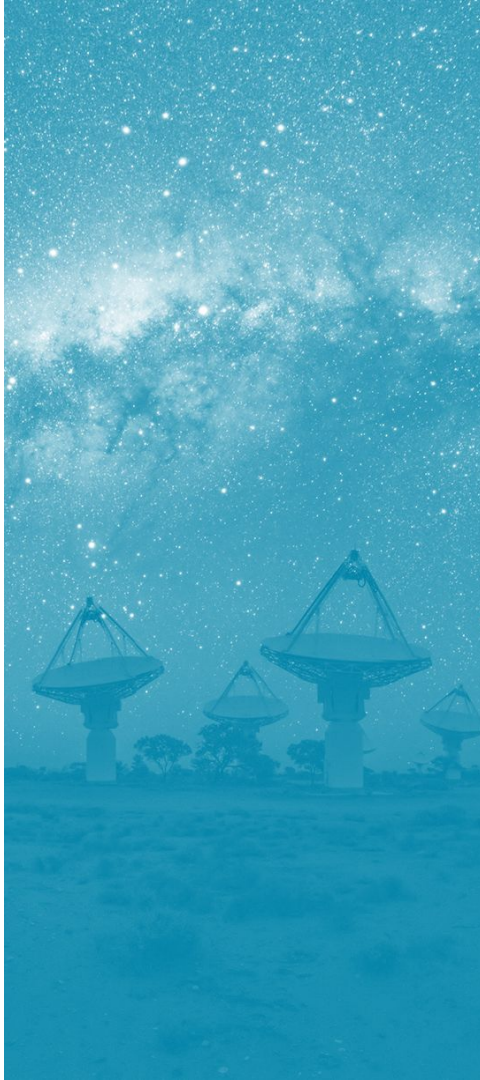
pytest

jupyter

GitLab

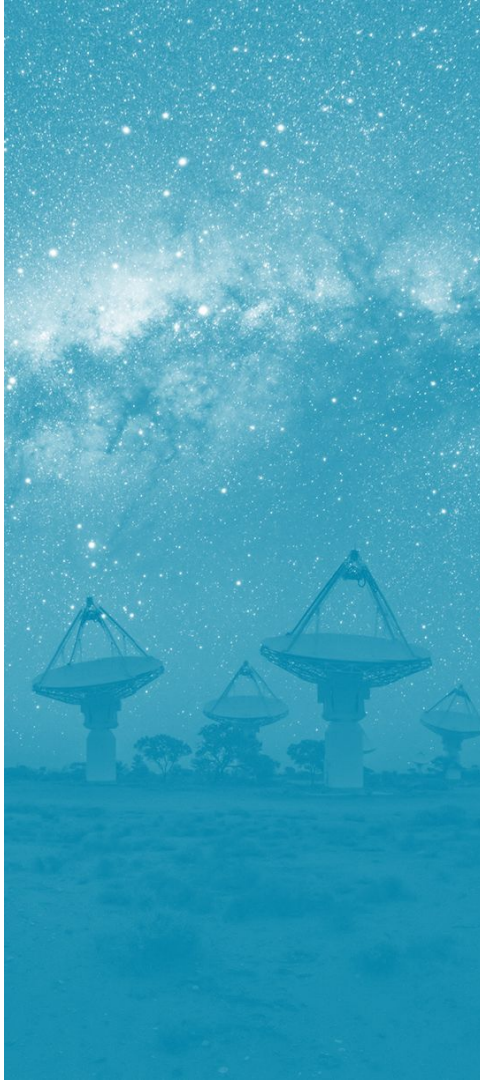
OPEN CONTAINER INITIATIVE

kubernetes



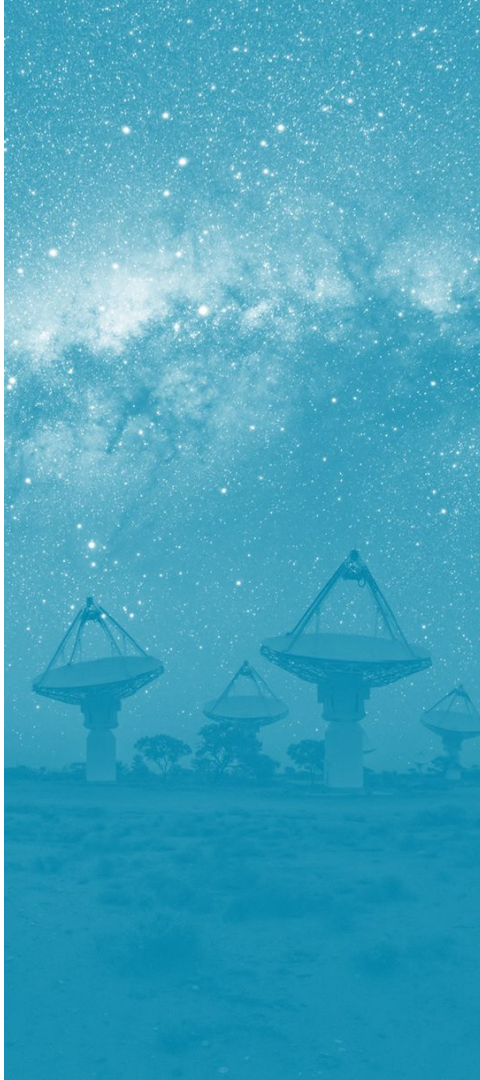
Processes

- MBSE
  - As per introduction yesterday! 
- SAFe
  - Scaled Agile Frame for Enterprise
  - lean agile development
  - all SKA software teams



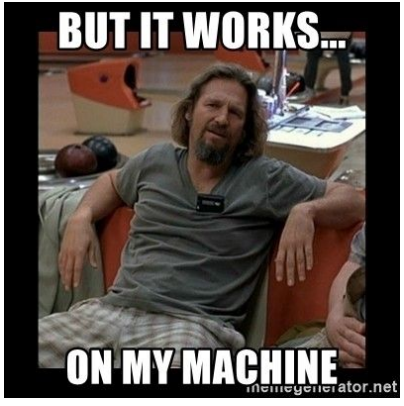
Platforms

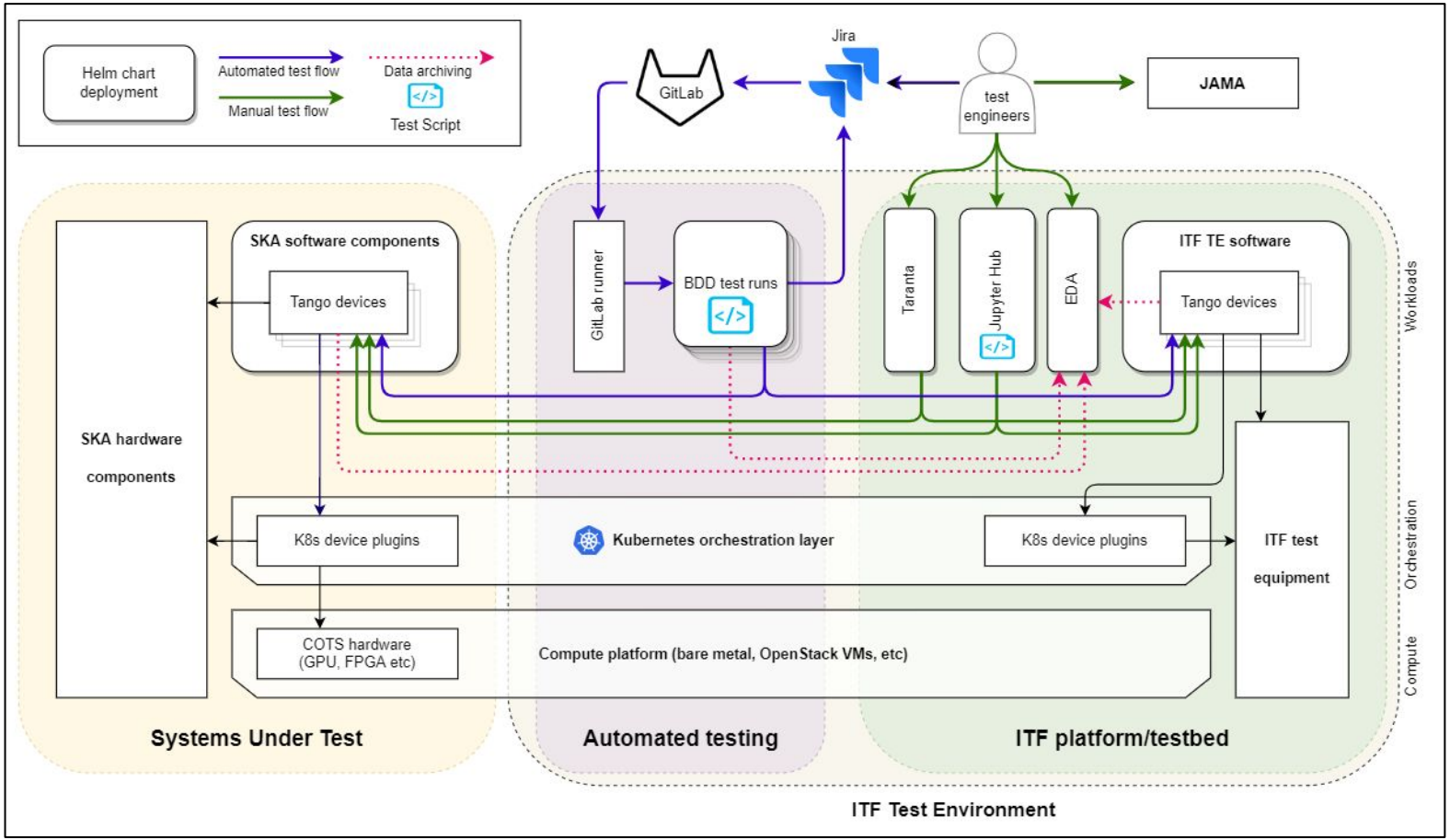
- Cloud-platform
- Prototype System Integration systems
- Integrated Test Facilities
- Central Processing Facilities
- Science Processing Facilities

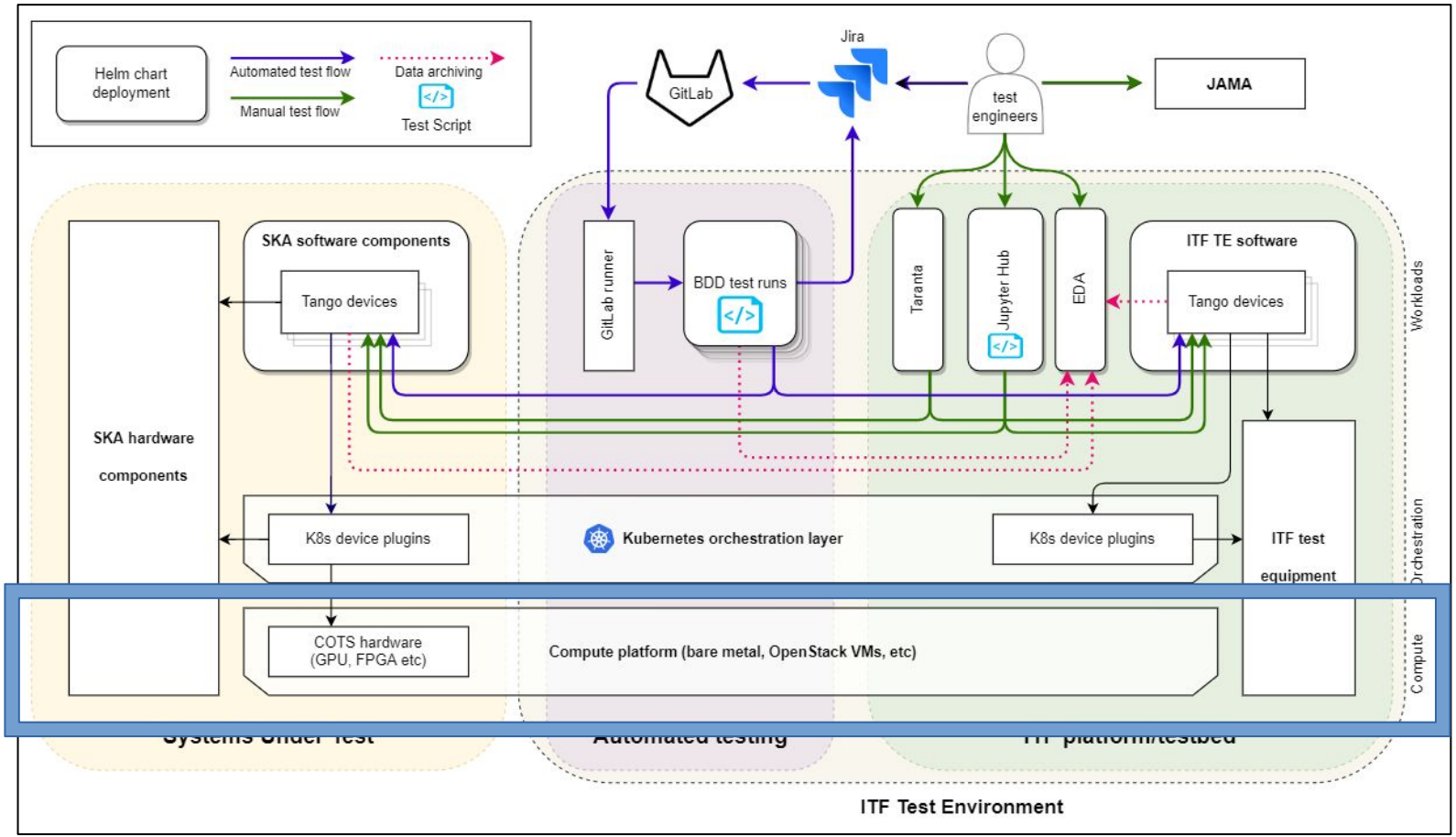


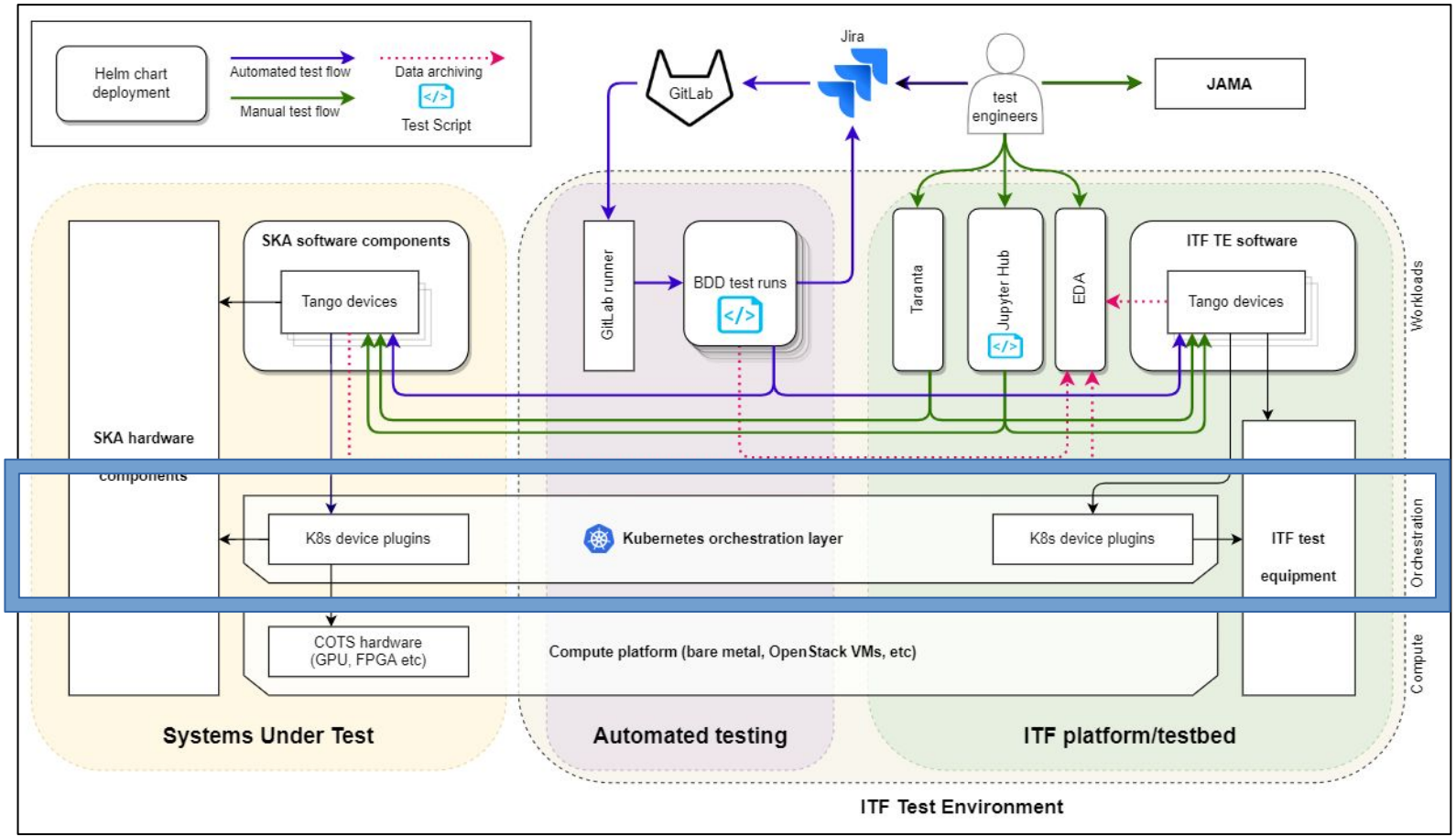
Providing predictable environments for SKA software

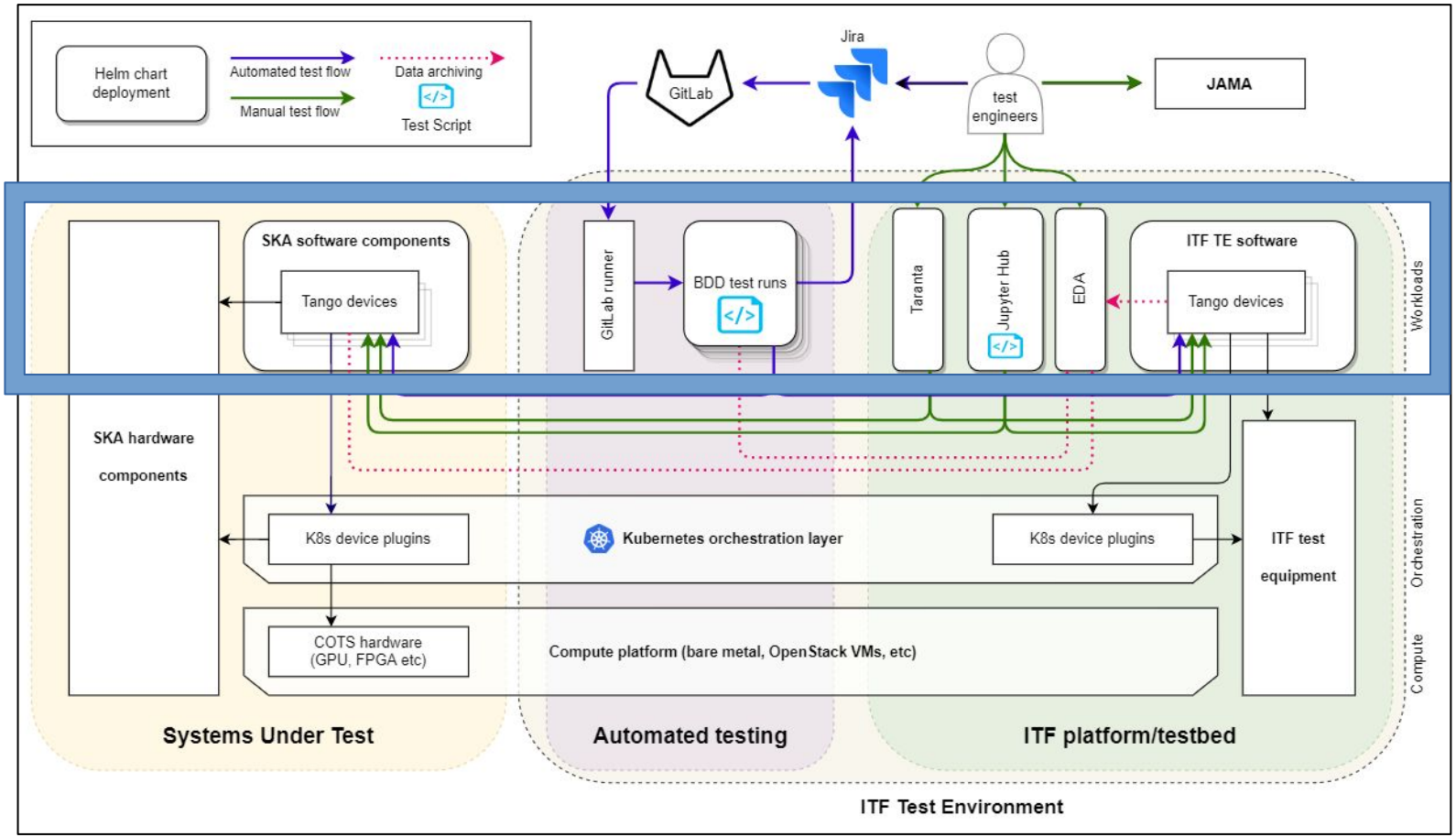
- An environment is what the software “sees”
  - Devices, storage, networks, other services, etc
- Modern SW environments are an abstraction over the underlying hardware
- DevOps role is to design, construct and maintain these abstractions, e.g…
  - Development
  - Continuous Integration environments
  - PSI Low - GPUs, high-performance NICs, FPGA and Custom HW
  - ITF
- We aim to minimise friction as software moves between environments
- By providing consistent and predictable ways to interact with them
  - e.g. requesting and accessing specialised hardware

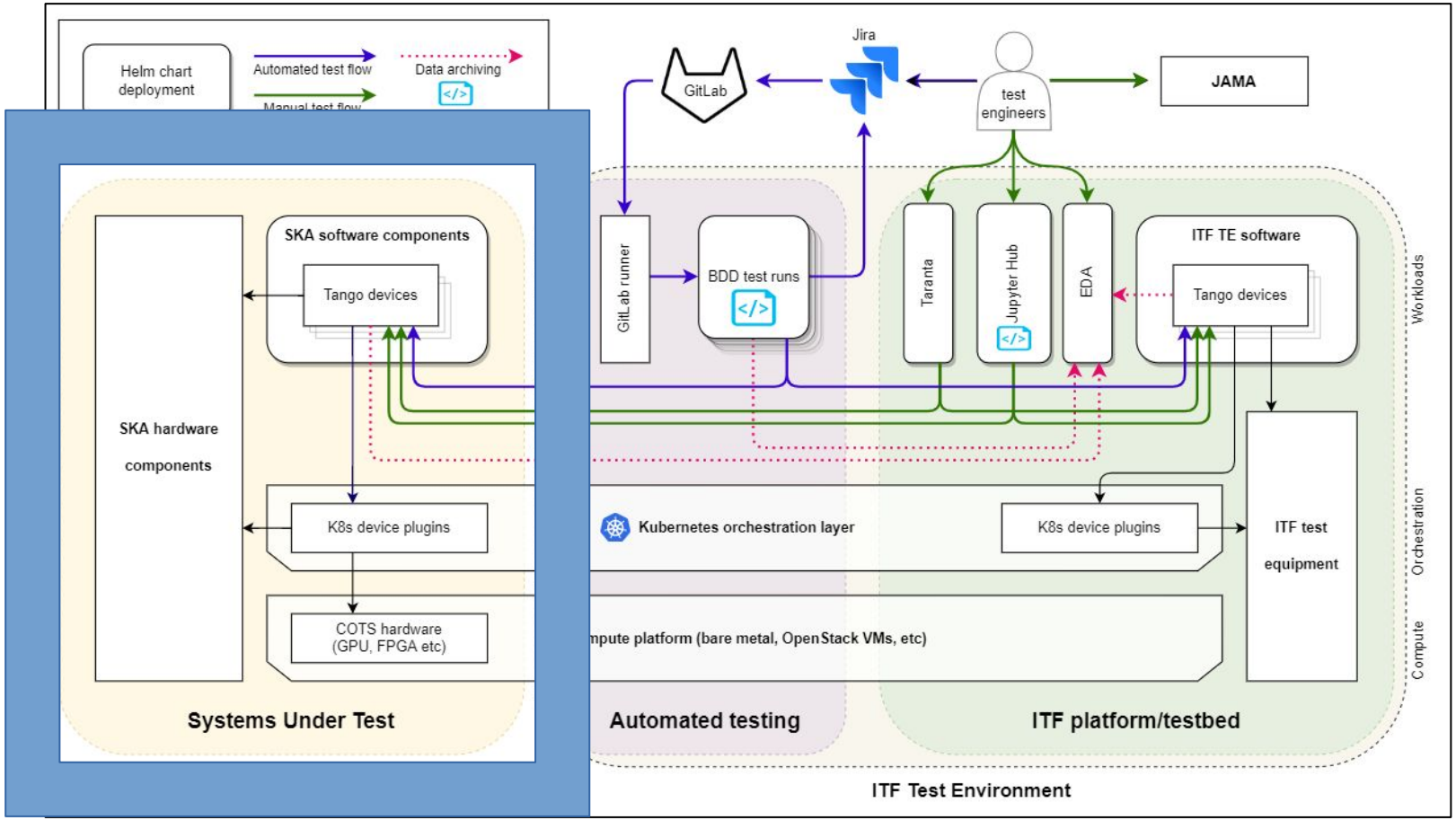


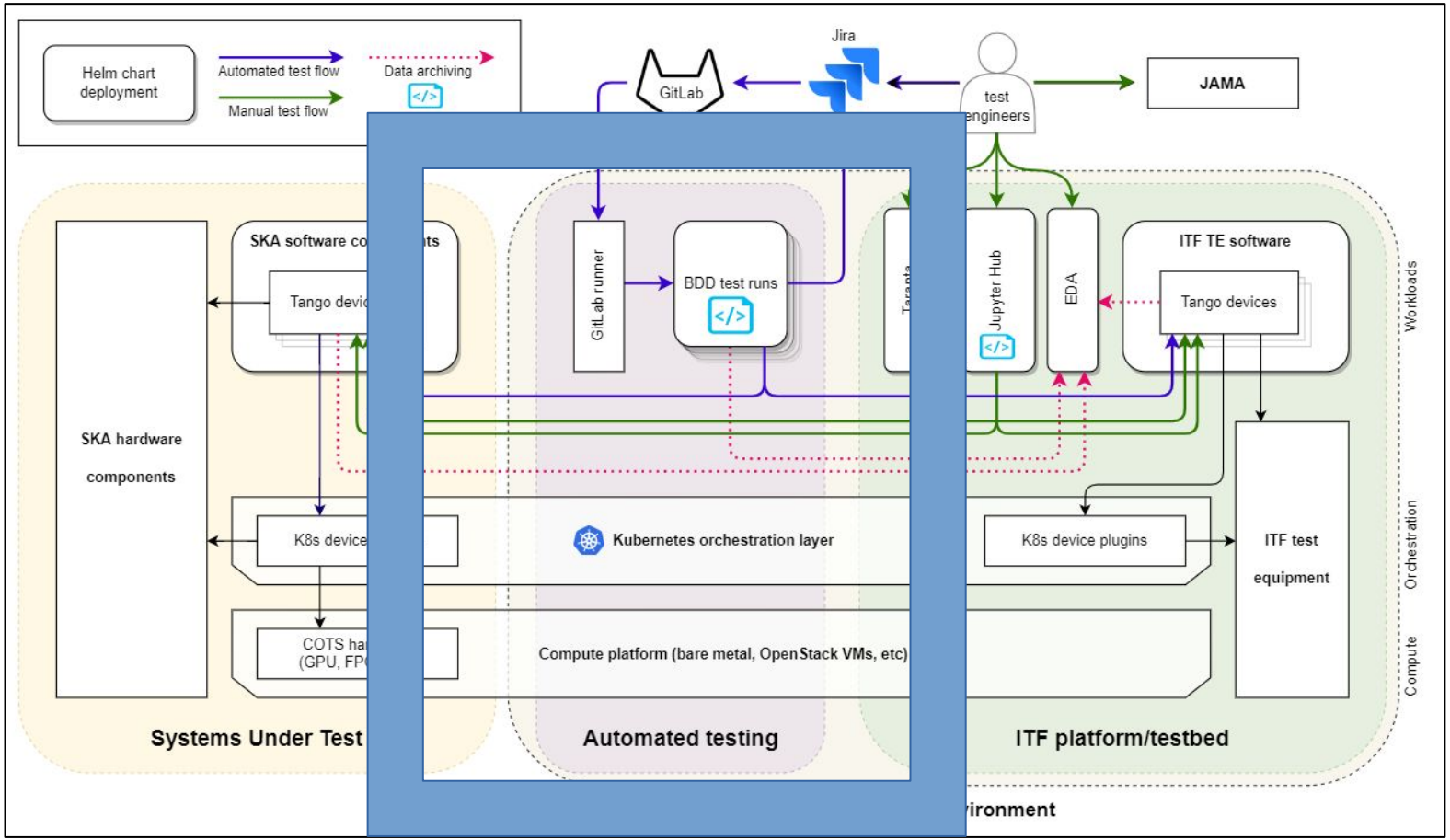

Helm chart deployment
Automated test flow
Manual test flow
Data archiving
Test Script
GitLab
Jira
test engineers
JAMA
SKA software components
Tango devices
GitLab runner
BDD test runs
Taranta
Jupyter Hub
EDA
ITF TE software
Tango devices
SKA hardware components
K8s device plugins
Kubernetes orchestration layer
K8s device plugins
ITF test equipment
COTS hardware (GPU, FPGA etc)
Compute platform (bare metal, OpenStack VMs, etc)
Systems Under Test
Automated testing
ITF platform/testbed
Workloads
Orchestration
Compute
ITF Test Environment

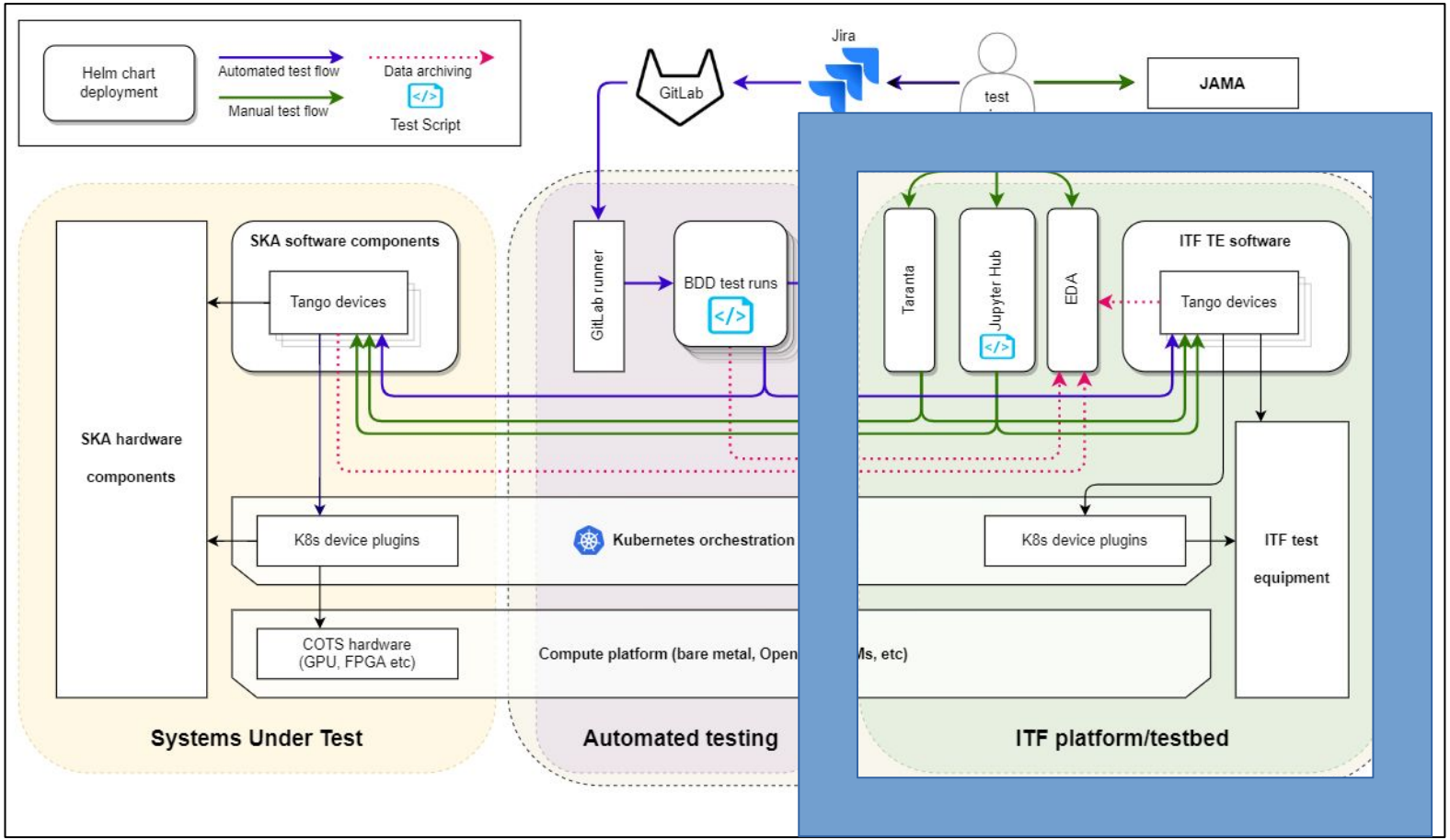

Helm chart deployment
Automated test flow
Manual test flow
Data archiving
Test Script
GitLab
Jira
test engineers
JAMA
SKA software components
Tango devices
GitLab runner
BDD test runs
Taranta
Jupyter Hub
EDA
ITF TE software
Tango devices
Workloads
SKA hardware components
K8s device plugins
Kubernetes orchestration layer
K8s device plugins
ITF test equipment
Orchestration
COTS hardware (GPU, FPGA etc)
Compute platform (bare metal, OpenStack VMs, etc)
Compute
Systems Under Test
Automated testing
ITF platform/testbed
ITF Test Environment


Helm chart deployment
Automated test flow
Manual test flow
Data archiving
Test Script
GitLab
Jira
test engineers
JAMA
SKA software components
Tango devices
GitLab runner
BDD test runs
Taranta
Jupyter Hub
EDA
ITF TE software
Tango devices
Workloads
SKA hardware components
K8s device plugins
Kubernetes orchestration layer
K8s device plugins
ITF test equipment
Orchestration
COTS hardware (GPU, FPGA etc)
Compute platform (bare metal, OpenStack VMs, etc)
Compute
Systems Under Test
Automated testing
ITF platform/testbed
ITF Test Environment

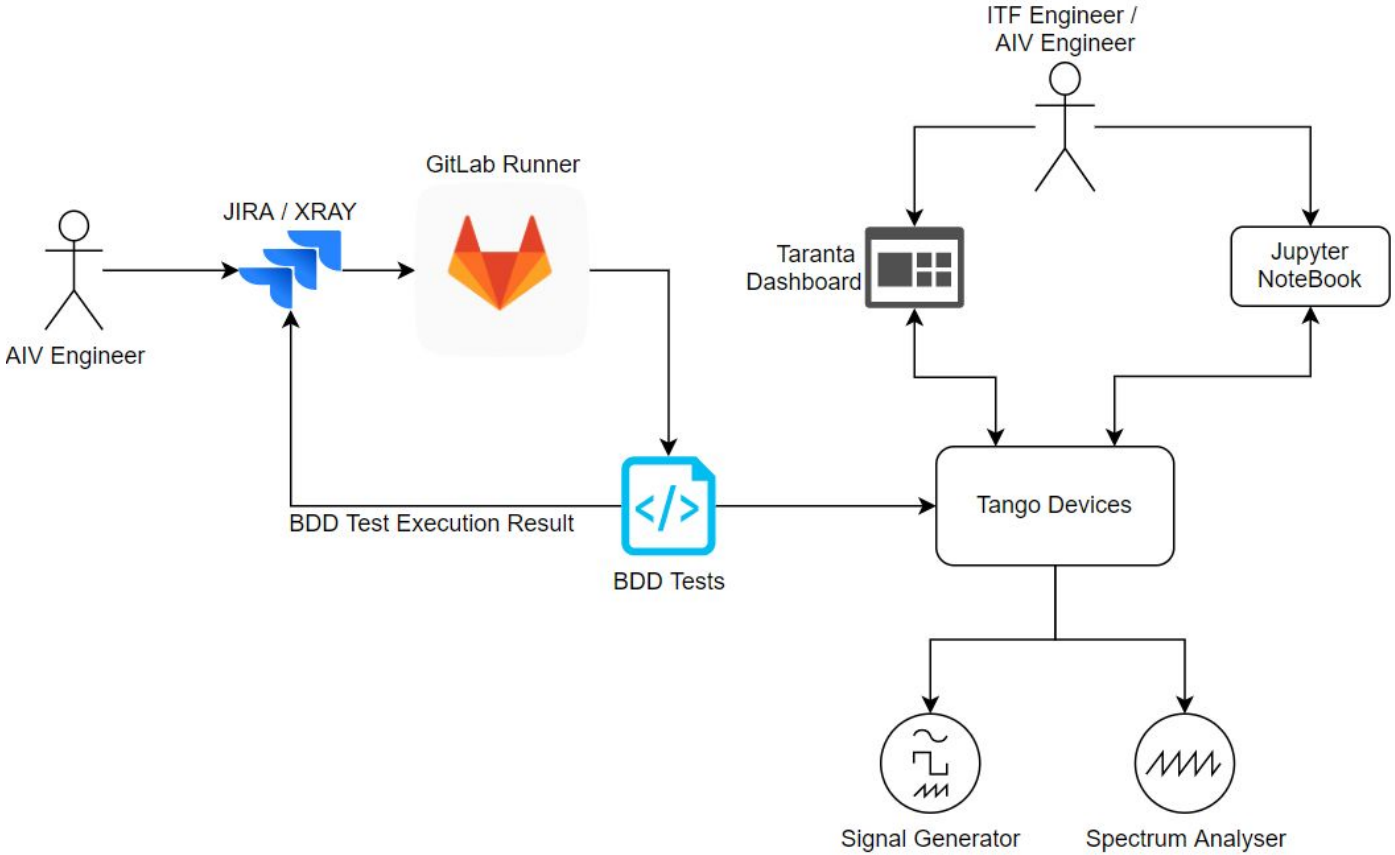

Helm chart deployment
Automated test flow
Manual test flow
Data archiving
Test Script
GitLab
Jira
test engineers
JAMA
SKA software components
Tango devices
GitLab runner
BDD test runs
Taranta
Jupyter Hub
EDA
ITF TE software
Tango devices
Workloads
SKA hardware components
K8s device plugins
Kubernetes orchestration layer
K8s device plugins
ITF test equipment
Orchestration
COTS hardware (GPU, FPGA etc)
Compute platform (bare metal, OpenStack VMs, etc)
Compute
Systems Under Test
Automated testing
ITF platform/testbed
ITF Test Environment


Helm chart deployment
Automated test flow
Manual test flow
Data archiving
GitLab
Jira
test engineers
JAMA
SKA software components
Tango devices
SKA hardware components
K8s device plugins
COTS hardware (GPU, FPGA etc)
Systems Under Test
GitLab runner
BDD test runs
Taranta
Jupyter Hub
EDA
ITF TE software
Tango devices
Kubernetes orchestration layer
K8s device plugins
ITF test equipment
mpute platform (bare metal, OpenStack VMs, etc)
Automated testing
ITF platform/testbed
Workloads
Orchestration
Compute
ITF Test Environment

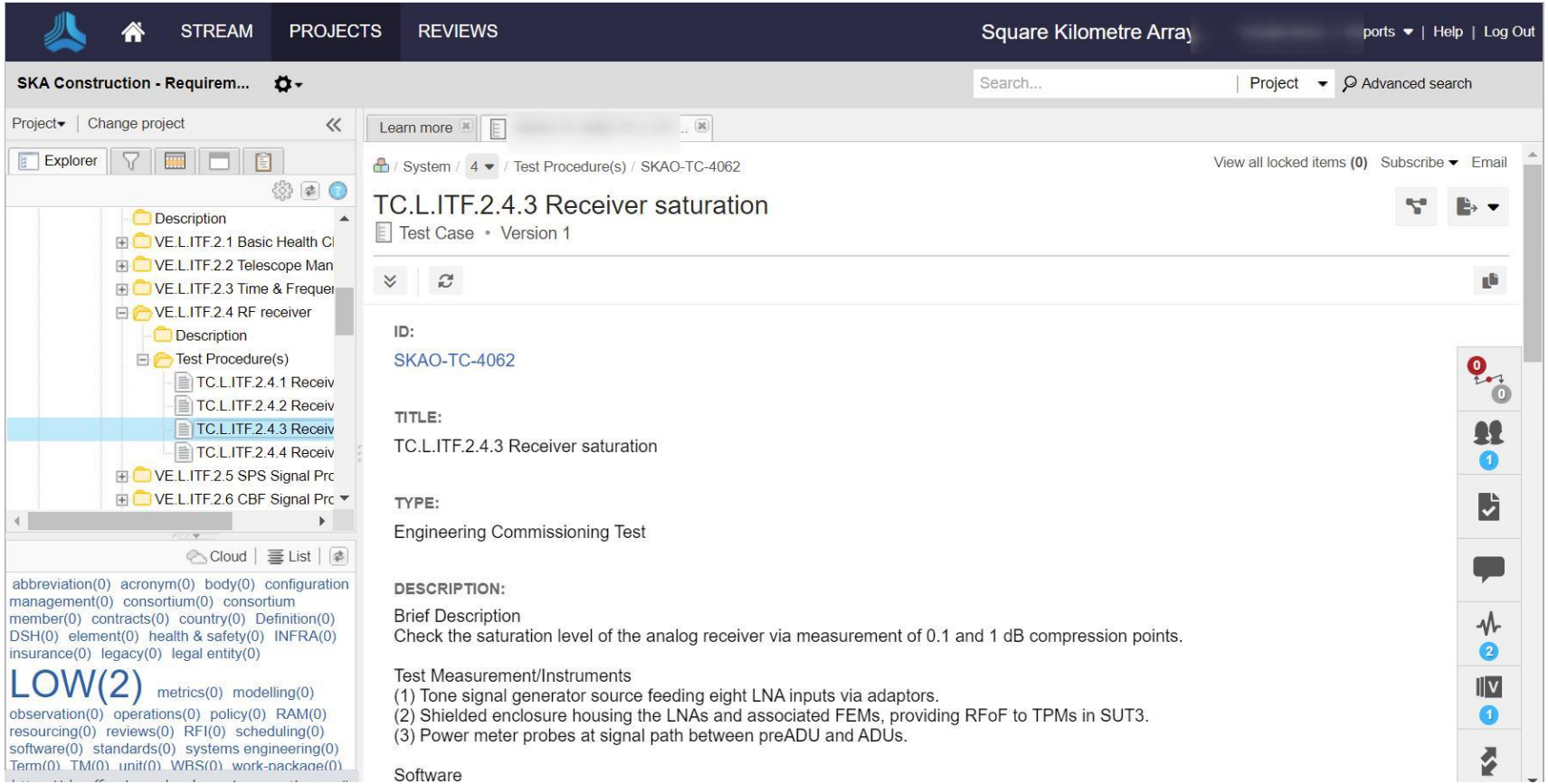

Helm chart deployment
Automated test flow
Manual test flow
Data archiving
GitLab
Jira
test engineers
JAMA
SKA software co
Tango devic
SKA hardware components
GitLab runner
BDD test runs
Jupyter Hub
EDA
ITF TE software
Tango devices
K8s device
Kubernetes orchestration layer
K8s device plugins
ITF test equipment
COTS ha
(GPU, FP
Compute platform (bare metal, OpenStack VMs, etc)
Systems Under Test
Automated testing
ITF platform/testbed
Workloads
Orchestration
Compute
vironment


Helm chart deployment
Automated test flow
Manual test flow
Data archiving
Test Script
GitLab
Jira
test
JAMA
SKA software components
Tango devices
SKA hardware components
GitLab runner
BDD test runs
Taranta
Jupyter Hub
EDA
ITF TE software
Tango devices
K8s device plugins
Kubernetes orchestration
K8s device plugins
ITF test equipment
COTS hardware (GPU, FPGA etc)
Compute platform (bare metal, Open
Ms, etc)
Systems Under Test
Automated testing
ITF platform/testbed

Test flow


ITF Engineer /
AIV Engineer
GitLab Runner
JIRA / XRAY
AIV Engineer
Taranta
Dashboard
Jupyter
NoteBook
BDD Test Execution Result
BDD Tests
Tango Devices
Signal Generator
Spectrum Analyser

JAMA


STREAM
PROJECTS
REVIEWS
Square Kilometre Array
Help
Log Out
SKA Construction - Requirem...
Search...
Project
Advanced search
Project
Change project
Explorer
Description
VE.L.ITF.2.1 Basic Health Cl
VE.L.ITF.2.2 Telescope Man
VE.L.ITF.2.3 Time & Freque
VE.L.ITF.2.4 RF receiver
Description
Test Procedure(s)
TC.L.ITF.2.4.1 Receiv
TC.L.ITF.2.4.2 Receiv
TC.L.ITF.2.4.3 Receiv
TC.L.ITF.2.4.4 Receiv
VE.L.ITF.2.5 SPS Signal Pro
VE.L.ITF.2.6 CBF Signal Pro
Cloud
List
abbreviation(0) acronym(0) body(0) configuration management(0) consortium(0) consortium member(0) contracts(0) country(0) Definition(0) DSH(0) element(0) health & safety(0) INFRA(0) insurance(0) legacy(0) legal entity(0)
LOW(2)
metrics(0) modelling(0) observation(0) operations(0) policy(0) RAM(0) resourcing(0) reviews(0) RFI(0) scheduling(0) software(0) standards(0) systems engineering(0) Term(0) TM(0) unit(0) WBS(0) work-package(0)
Learn more
System / 4 / Test Procedure(s) / SKAO-TC-4062
View all locked items (0)
Subscribe
Email
TC.L.ITF.2.4.3 Receiver saturation
Test Case • Version 1
ID:
SKAO-TC-4062
TITLE:
TC.L.ITF.2.4.3 Receiver saturation
TYPE:
Engineering Commissioning Test
DESCRIPTION:
Brief Description
Check the saturation level of the analog receiver via measurement of 0.1 and 1 dB compression points.
Test Measurement/Instruments
(1) Tone signal generator source feeding eight LNA inputs via adaptors.
(2) Shielded enclosure housing the LNAs and associated FEMs, providing RFoF to TPMs in SUT3.
(3) Power meter probes at signal path between preADU and ADUs.
Software

Code

```
Feature: ITF Low environment testing

    @XTP-13588
    Scenario Outline: ITF Test Equipment are reporting correct models
        Given the Test Equipment <fqdn>
        When the Test Equipment is initialised
        Then the Test Equipment reports itself as <identity>

    Examples:
        |fqdn               |identity                                  |
        |low-itf/siggen/1   |Tektronix,TSG4104A,s/nC010133,ver2.03.26   |
        |low-itf/spectana/1 |TEKTRONIX,SPECMON26B,PQ00112, FV:3.9.0031.0|
```

```python
from pytest_bdd import given, scenario, then, when, parsers

import tango


@scenario(
    "features/test_equipment_example.feature",
    "ITF Test Equipment are reporting correct models",
)
def test_itf_test_equipment_are_reporting_correct_models():
    """ITF Test Equipment are reporting correct models."""


@given(parsers.parse("the Test Equipment {fqdn}"), target_fixture="proxy")
def the_test_equipment_fqdn(fqdn: str) -> tango.DeviceProxy:
    """The Test Equipment <fqdn>.

    :param fqdn: the Tango FQDN of the device
    """
    return tango.DeviceProxy(fqdn)


@when("the Test Equipment is initialised")
def the_test_equipment_fqdn_is_initialised(proxy: tango.DeviceProxy) -> None:
    """the Test Equipment is initialised."""
    proxy.adminmode = 0


@then(parsers.parse("the Test Equipment reports itself as {identity}"))
def the_test_equipment_reports_itself_as_identity(
    proxy: tango.DeviceProxy, identity: str
) -> None:
    """the Test Equipment reports itself as <identity>."""
    assert proxy.identity == identity
```

SAFe Program / SP-2575

ITF Test Equipment testing flow integration

Edit | Comment | Assign | More | Implementing | Done | Workflow

Details

Description

Define BDD tests and Test plans, and implement the tests against the TANGO devices exposed through ~~SP-2576~~

Test Coverage

Add Tests | Execute | ...

TEST COVERAGE FOR THE FOLLOWING ANALYSIS SCOPE

Scope: Version; **Version:** Earliest Unreleased; **Environment:** ITF-LOW

OK

Filter(s)

Show 10 entries Columns

Key	Summary	Test Runs	Test Status
XTP-7690	Test a Signal Generator frequency and amp can be set	0	PASS
XTP-9041	Test a Signal Generator output can be turned OFF	0	PASS
XTP-9042	Test a Signal Generator output can be turned ON	0	PASS
XTP-9312	Test a Signal Generator frequency can not be set with a negative value	0	PASS
XTP-13587	Test connection between Signal Generator and Spectrum Analyser	0	PASS
XTP-13588	Test an ITF Test Equipment is reporting correct model	0	PASS
XTP-13599	Test that a Spectrum Analyser can be configured to plot a trace	0	PASS

Solution and System Acceptance Tests (skampi tests) / Test Plan: XTP-9052 / Test Execution: XTP-13890 / Test: XTP-13587

Test connection between Signal Generator and Spectrum Analyser

Import Execution Results | Export to Cucumber | Return to Test Execution | Next

There are no Test Run Custom Fields defined.

Test Description

Test Issue Links 2

Test Type: Cucumber

Scenario Type: Scenario Outline

Scenario:

```
Given the Signal Generator is online
And the Signal Generator is initialised
And the Signal Generator RF Output is OFF
And the Spectrum Analyser is online
And the Spectrum Analyser is initialised
When the user specifies the Spectrum Analyser start frequency as <start_frequency> Hz
And the user specifies the Spectrum Analyser stop frequency as <stop_frequency> Hz
And the user specifies the Signal Generator frequency as <frequency> Hz
And the user specifies the Signal Generator power as <power> dBm
And the user turns the Signal Generator RF Output ON
Then the Spectrum Analyser peak frequency is <frequency> Hz
And the Spectrum Analyser peak power is no more than <power> dBm

    Examples:
          | start_frequency | stop_frequency |  frequency  | power |
          | 25000000.0      | 75000000.0     |  50000000.0 | -20.0 |
          | 50000000.0      | 200000000.0    | 100000000.0 | -10.0 |
```

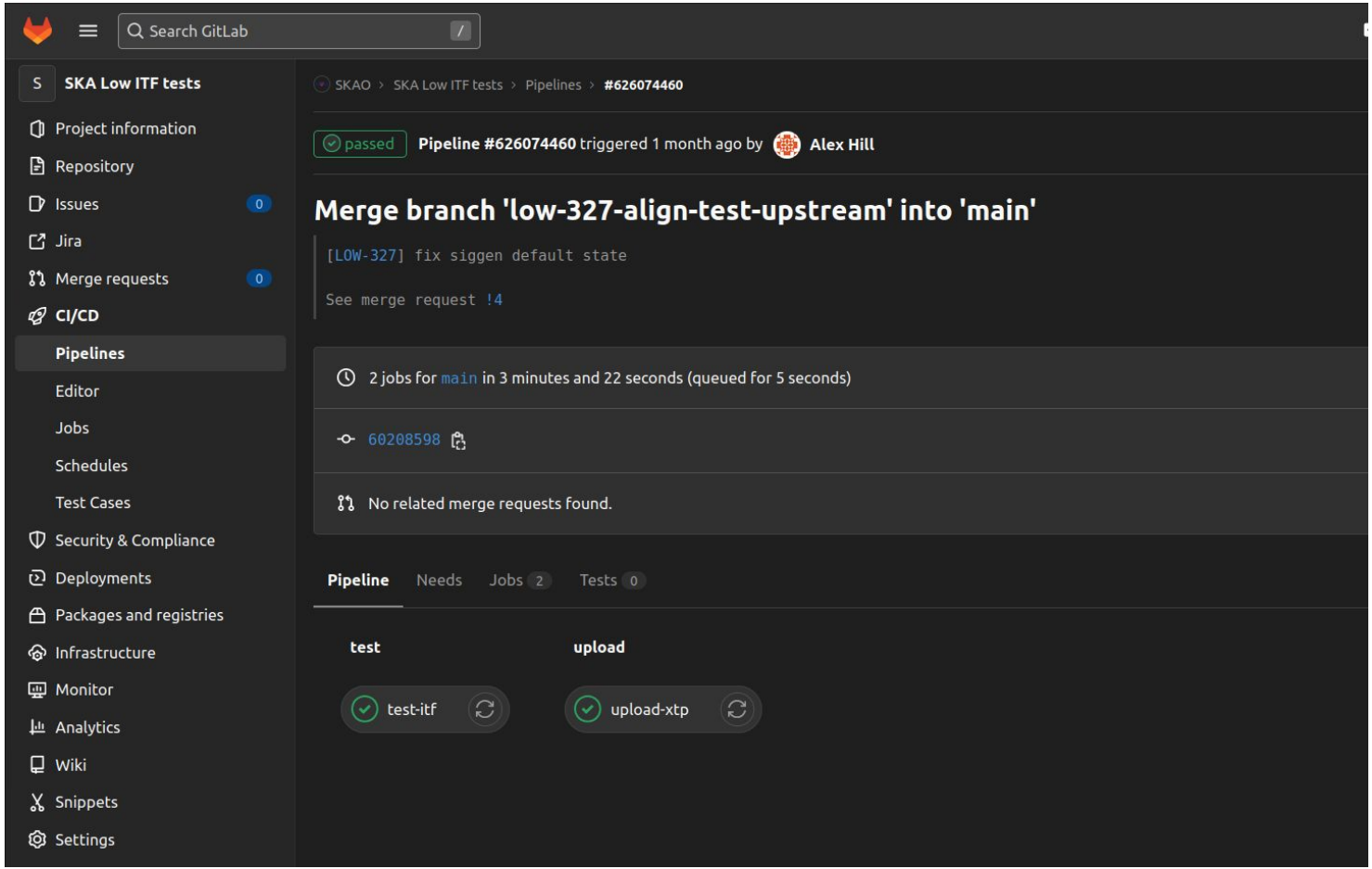
Examples

<start_frequency>	<stop_frequency>	<frequency>	<power>	Duration	Status
25000000.0	75000000.0	50000000.0	-20.0	27979.362 ms	PASS

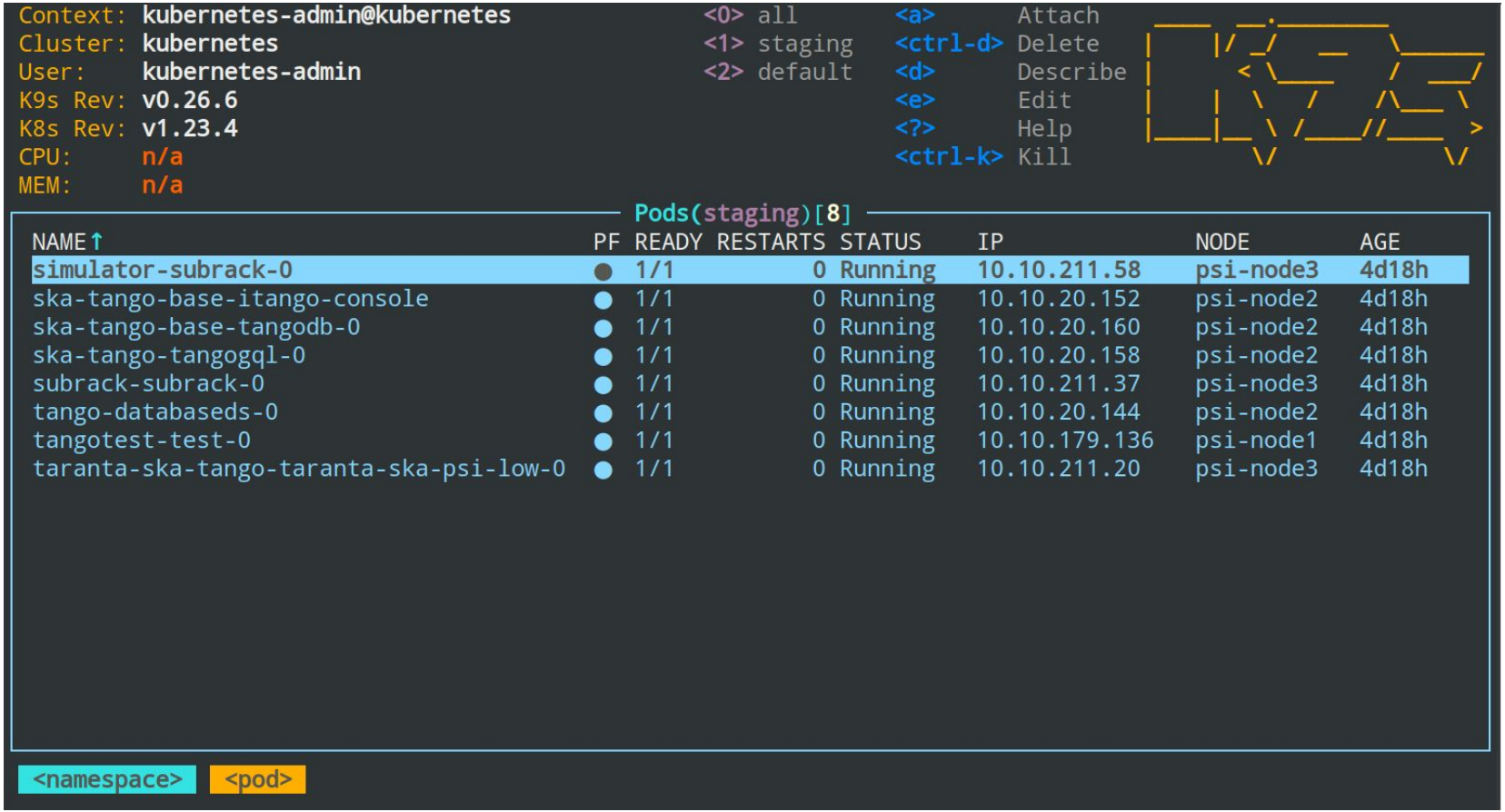
Steps

Given the Signal Generator is online	89.760 ms	PASS
And the Signal Generator is initialised	194.675 ms	PASS
And the Signal Generator RF output is OFF	0.758 ms	PASS
And the Spectrum Analyser is online	51.127 ms	PASS
And the Spectrum Analyser is initialised	2673.217 ms	PASS
When the user specifies the Spectrum Analyser stop frequency as 75000000.0 Hz	8592.205 ms	PASS
And the user specifies the Spectrum Analyser start frequency as 25000000.0 Hz	2998.495 ms	PASS

gitlab continuous integration



k8s


Context: kubernetes-admin@kubernetes
Cluster: kubernetes
User: kubernetes-admin
K9s Rev: v0.26.6
K8s Rev: v1.23.4
CPU: n/a
MEM: n/a
<0> all
<1> staging
<2> default
<a> Attach
<ctrl-d> Delete
<d> Describe
<e> Edit
<?> Help
<ctrl-k> Kill
Pods(staging)[8]
NAME↑ PF READY RESTARTS STATUS IP NODE AGE
simulator-subrack-0 1/1 0 Running 10.10.211.58 psi-node3 4d18h
ska-tango-base-itango-console 1/1 0 Running 10.10.20.152 psi-node2 4d18h
ska-tango-base-tangodb-0 1/1 0 Running 10.10.20.160 psi-node2 4d18h
ska-tango-tangogql-0 1/1 0 Running 10.10.20.158 psi-node2 4d18h
subrack-subrack-0 1/1 0 Running 10.10.211.37 psi-node3 4d18h
tango-databaseds-0 1/1 0 Running 10.10.20.144 psi-node2 4d18h
tangotest-test-0 1/1 0 Running 10.10.179.136 psi-node1 4d18h
taranta-ska-tango-taranta-ska-psi-low-0 1/1 0 Running 10.10.211.20 psi-node3 4d18h
<namespace>
<pod>

Thank You

Malte.Marquarding@csiro.au

Milky Way overhead the AAVS station
Credit: Michael Goh and ICRAR/Curtin.