Kubernetes at SKA

Matteo Di Carlo (INAF-OAAB)

TANGO-Workshop @ICALEPCS 2021

About me

- Matteo Di Carlo (matteo.dicarlo@inaf.it)
- Working for INAF-OAAB since 2014
- Since 2015 in the SKA project
- Software engineer, in SKA part of the system team and coordinator of the cop-tango community
- https://orcid.org/0000-0002-3903-9637

ska-tango-examples repository

Demonstrates how to structure a project that pr des some simple Tango devices coded in PyTango h k netes

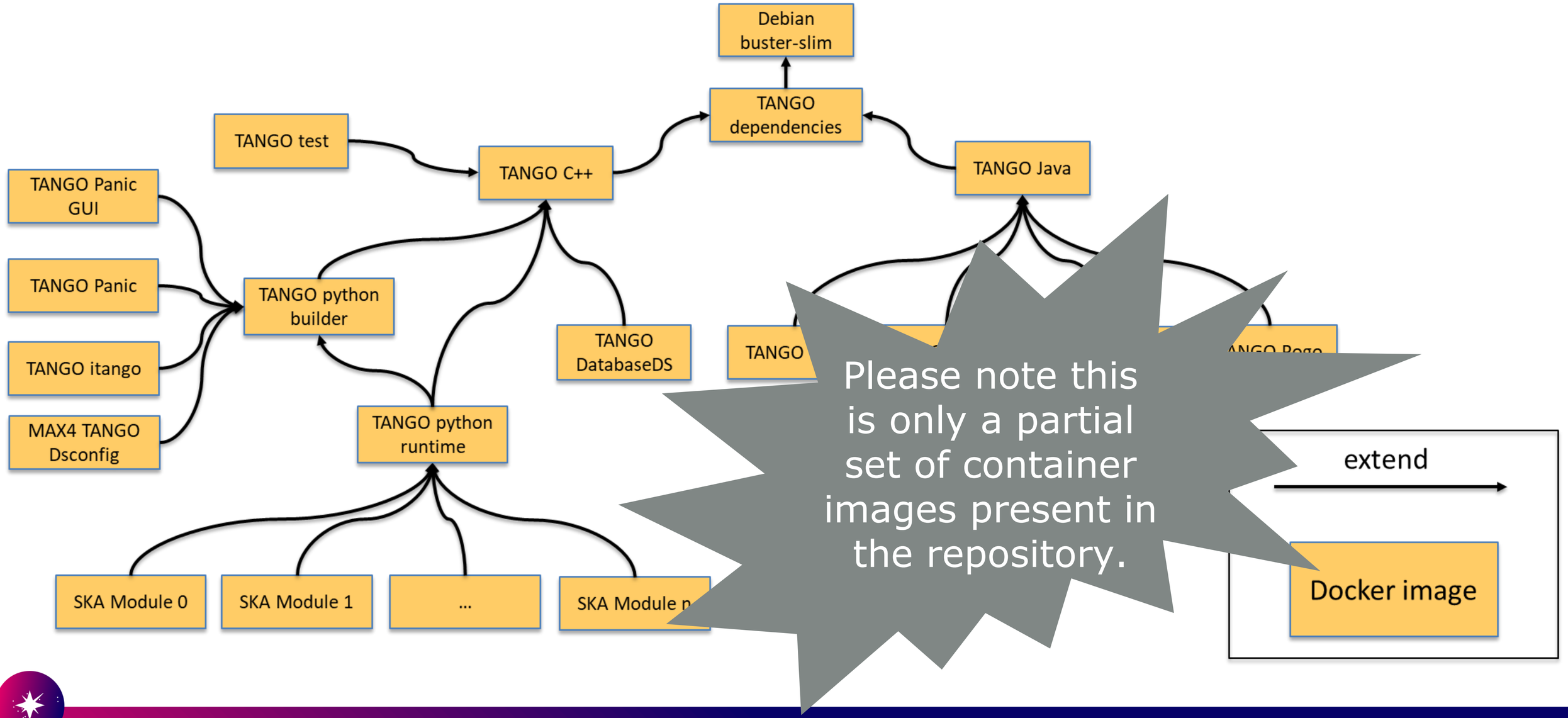
- Development and testing done k
- Environments and test ble and independent of host enviro
- Many authors:
  - https://gitlab.com/ska-telescop s/master
- The code shown in this presentation e: ps://gitlab.com/ska-telescope/ska-tango-examples/-/tre ANGO-workshop

More info: https://docs.google.com/presentation/d/1qA7twT2VQGG5S2nYKABj3hD-9y9e1L7LvvvqMrictgo/edit?usp=sharing

Requirements

- Install docker:
  - Follow the instructions available at ttps://docs.docker.com/get-docker/
- Install minikube:
  - Follow the instructions available at https://gitlab.com/ska-telescope/sdi/deploy-minikube/
- Optionally, install host OS dependencies:
  - Compile the TANGO framework: https://gitlab.com/tango-controls/cppTango/-/blob/main/INSTALL.md

SKA-tango-images - 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*
  - Tool for managing Kubernetes charts
  - Chart is a package of pre-configure
    information for running a Kube

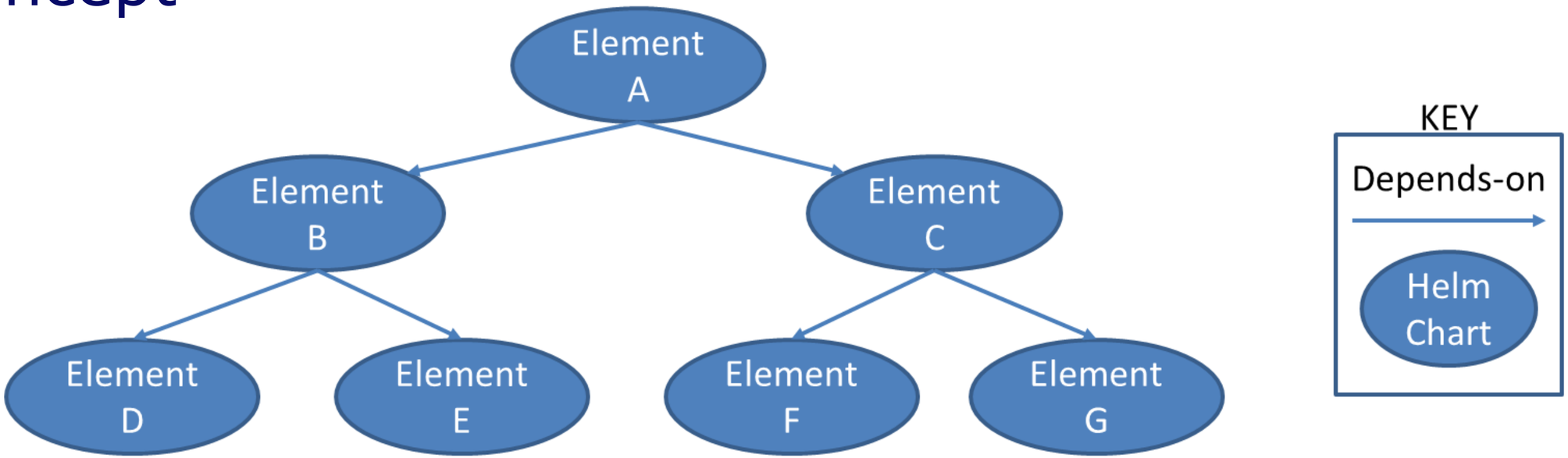
We are working in this area at the moment for standardising a common set of Makefile targets. This will bring every repository to have a standardized structure (set of folder, common files, etc.)

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)!

Architecture for integration (with Helm)

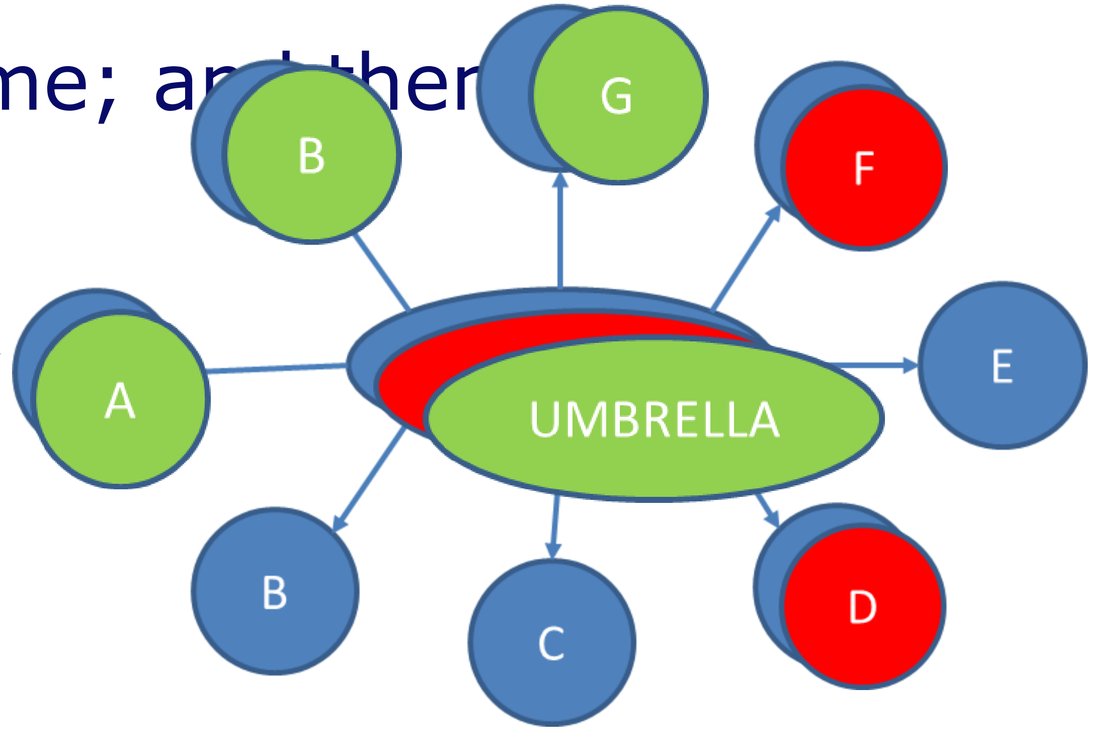
- Helm has the concept of dependency
  - An helm chart can have one or more sub-charts
- The integration of SKA elements can be done with this concept



Umbrella charts

- Operational as[illegible] using dependencies: the sub-charts
  - [illegible] then
  - so[illegible] ame; and then
    - [illegible] der.

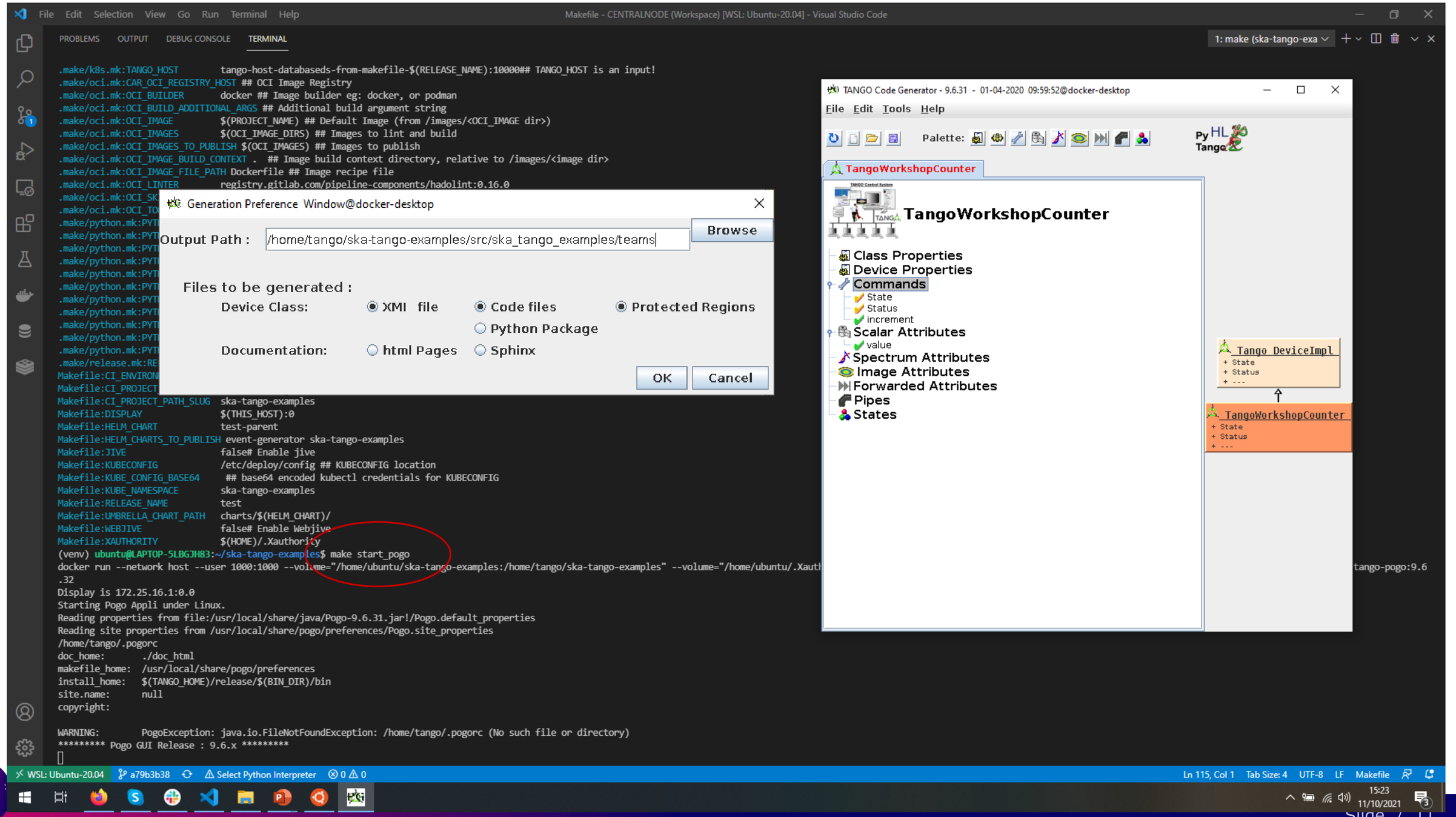
For every SKA element, there is at least an umbrella chart for integration testing



Live demo

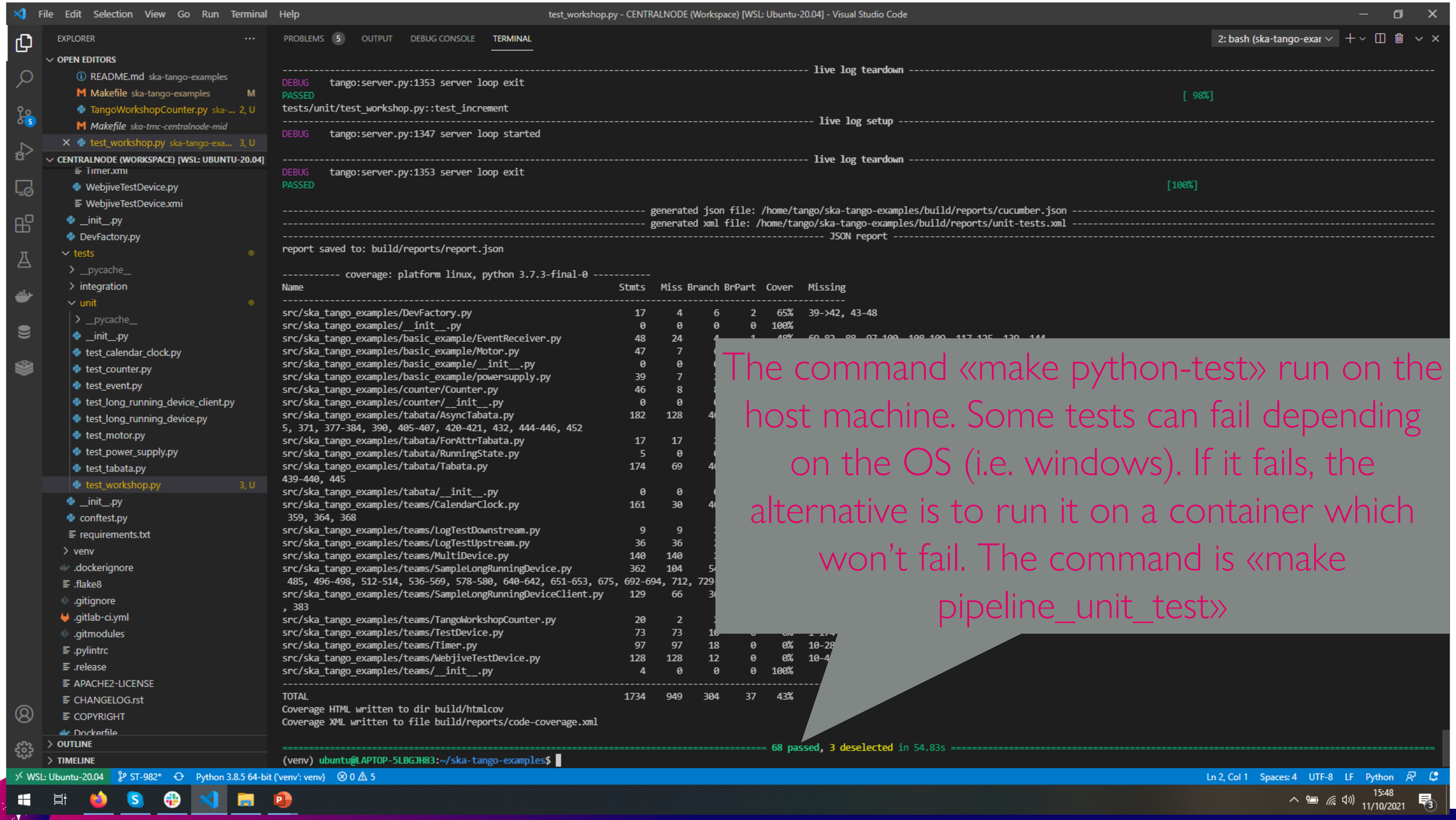
Development of a device step 1: POGO

- make start_pogo
  - Remember to export the DISPLAY and Xauthority environment variables
- In the home folder there folder where you started Pogo
- Generate the device


File Edit Selection View Go Run Terminal Help
Makefile - CENTRALNODE (Workspace) [WSL: Ubuntu-20.04] - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: make (ska-tango-exa
.make/k8s.mk:TANGO_HOST tango-host-databaseds-from-makefile-$(RELEASE_NAME):10000## TANGO_HOST is an input!
.make/oci.mk:CAR_OCI_REGISTRY_HOST ## OCI Image Registry
.make/oci.mk:OCI_BUILDER docker ## Image builder eg: docker, or podman
.make/oci.mk:OCI_BUILD_ADDITIONAL_ARGS ## Additional build argument string
.make/oci.mk:OCI_IMAGE $(PROJECT_NAME) ## Default Image (from /images/<OCI_IMAGE dir>)
.make/oci.mk:OCI_IMAGES $(OCI_IMAGE_DIRS) ## Images to lint and build
.make/oci.mk:OCI_IMAGES_TO_PUBLISH $(OCI_IMAGES) ## Images to publish
.make/oci.mk:OCI_IMAGE_BUILD_CONTEXT . ## Image build context directory, relative to /images/<image dir>
.make/oci.mk:OCI_IMAGE_FILE_PATH Dockerfile ## Image recipe file
.make/oci.mk:OCI_LINTER registry.gitlab.com/pipeline-components/hadolint:0.16.0
Generation Preference Window@docker-desktop
Output Path :
/home/tango/ska-tango-examples/src/ska_tango_examples/teams
Browse
Files to be generated :
Device Class:
XMI file
Code files
Protected Regions
Python Package
Documentation:
html Pages
Sphinx
OK
Cancel
Makefile:CI_PROJECT_PATH_SLUG ska-tango-examples
Makefile:DISPLAY $(THIS_HOST):0
Makefile:HELM_CHART test-parent
Makefile:HELM_CHARTS_TO_PUBLISH event-generator ska-tango-examples
Makefile:JIVE false# Enable jive
Makefile:KUBECONFIG /etc/deploy/config ## KUBECONFIG location
Makefile:KUBE_CONFIG_BASE64 ## base64 encoded kubectl credentials for KUBECONFIG
Makefile:KUBE_NAMESPACE ska-tango-examples
Makefile:RELEASE_NAME test
Makefile:UMBRELLA_CHART_PATH charts/$(HELM_CHART)/
Makefile:WEBJIVE false# Enable Webjive
Makefile:XAUTHORITY $(HOME)/.Xauthority
(venv) ubuntu@LAPTOP-5LBGJH83:~/ska-tango-examples$ make start_pogo
docker run --network host --user 1000:1000 --volume="/home/ubuntu/ska-tango-examples:/home/tango/ska-tango-examples" --volume="/home/ubuntu/.Xauth
tango-pogo:9.6.32
Display is 172.25.16.1:0.0
Starting Pogo Appli under Linux.
Reading properties from file:/usr/local/share/java/Pogo-9.6.31.jar!/Pogo.default_properties
Reading site properties from /usr/local/share/pogo/preferences/Pogo.site_properties
/home/tango/.pogorc
doc_home: ./doc_html
makefile_home: /usr/local/share/pogo/preferences
install_home: $(TANGO_HOME)/release/$(BIN_DIR)/bin
site.name: null
copyright:
WARNING: PogoException: java.io.FileNotFoundException: /home/tango/.pogorc (No such file or directory)
******** Pogo GUI Release : 9.6.x ********
TANGO Code Generator - 9.6.31 - 01-04-2020 09:59:52@docker-desktop
File Edit Tools Help
Palette:
PyHL Tango
TangoWorkshopCounter
TangoWorkshopCounter
Class Properties
Device Properties
Commands
State
Status
increment
Scalar Attributes
value
Spectrum Attributes
Image Attributes
Forwarded Attributes
Pipes
States
Tango DeviceImpl
+ State
+ Status
+ ---
TangoWorkshopCounter
+ State
+ Status
+ ---
WSL: Ubuntu-20.04 a79b3b38 Select Python Interpreter 0 0
Ln 115, Col 1 Tab Size: 4 UTF-8 LF Makefile
15:23
11/10/2021

Development of a device step 2: coding & testing

- Create a virtualenv
- Write your device and create a test (test-drive approach)
- Remember to lint:
  - make python-format && make python-lint
- Use DeviceTestContext or MultiDeviceTestContext
  - make python-test



The command «make python-test» run on the host machine. Some tests can fail depending on the OS (i.e. windows). If it fails, the alternative is to run it on a container which won't fail. The command is «make pipeline_unit_test»

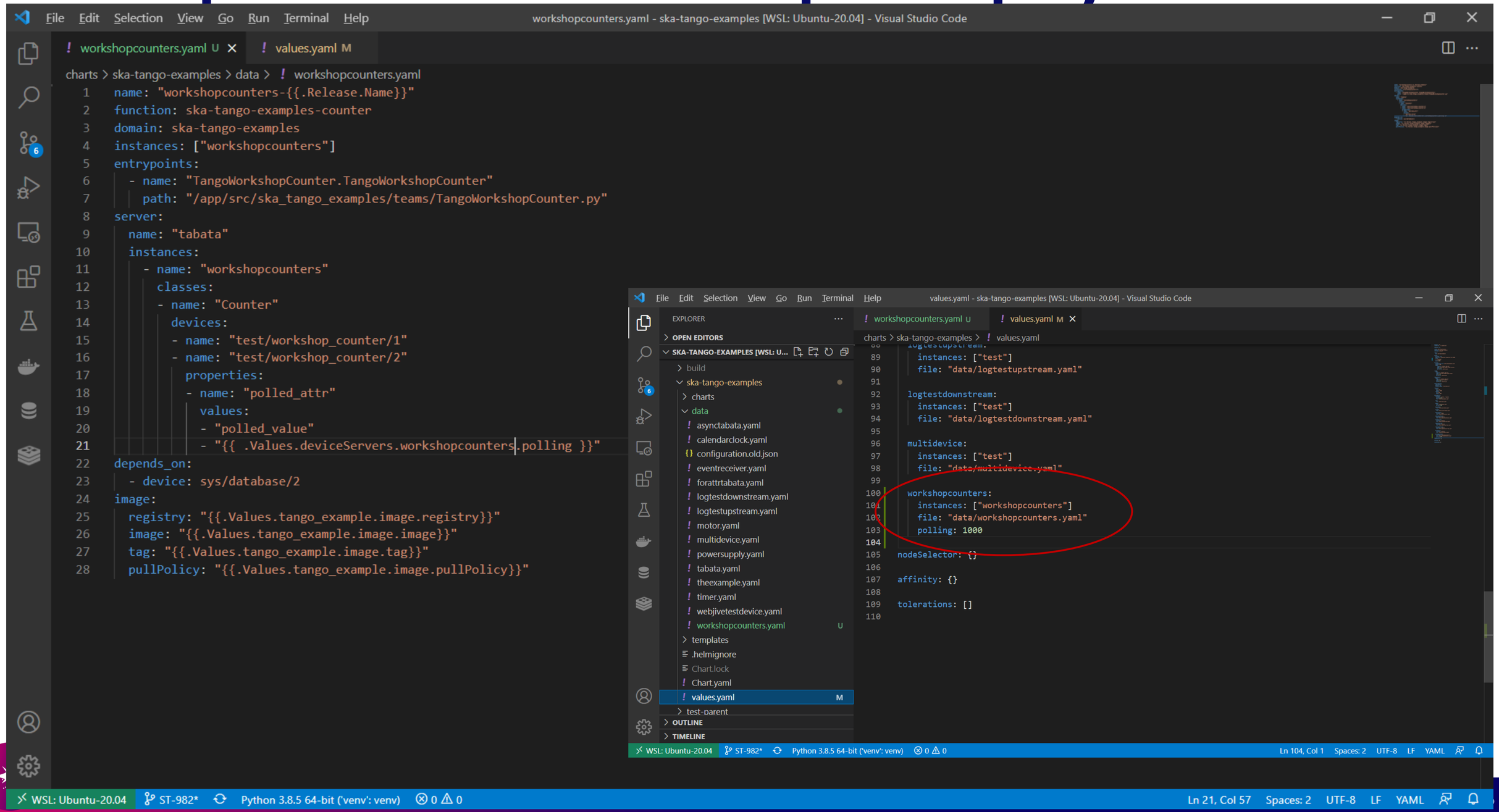
Development of a device step 3: deployment

- In order to install the examples, two charts have been created: one called ska-tango-examples which is the real application and the umbrella chart, called test-parent, used for testing.
- The ska-tango-examples uses the ska-tango-base chart for setting up the TANGO eco-system (mysql database and databaseds device) and the ska-tango-util library chart which helps in the definition of the TANGO device servers
- More information on: https://gitlab.com/ska-telescope/ska-tango-images

How does it work?

- Define the device server and devices in a yaml file inside the folder data according to the documentation
- Reference that file in the main values file
- Use it in the template deviceservers.yaml
- It is possible to define dependency which will create one init container for each of them

Development of a device step 3: deployment



Development of a device step 4: install on minikube

- «make oci-build» builds the container image (with docker)
- «make install-chart» installs the deployment

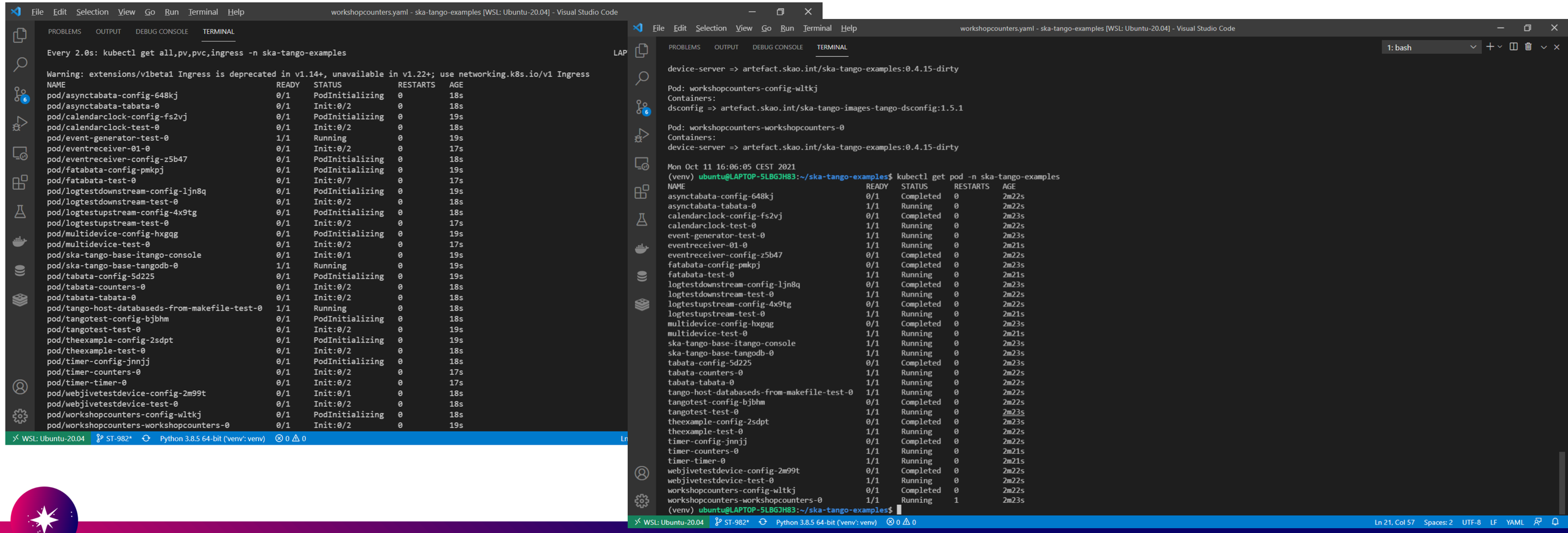
```
Name:         ska-tango-examples
Labels:       <none>
Annotations:  <none>
Status:       Active

No resource quota.

No LimitRange resource.
install-chart: install charts/test-parent/ release: test in Namespace: ska-tango-examples with params: --set global.minikube=true --set global.tango_host=tango-host-databaseds-from-makefile-test:10000
--set ska-tango-base.display=172.25.29.163:0 --set ska-tango-base.xauthority=/home/ubuntu/.Xauthority --set ska-tango-base.jive.enabled=false --set webjive.enabled=false --set tango_example.tango_examp
le.image.tag=0.4.15-dirty --set event_generator.events_generator.image.tag=0.4.15-dirty --values gilab_values.yaml
helm upgrade --install test \
--set global.minikube=true --set global.tango_host=tango-host-databaseds-from-makefile-test:10000 --set ska-tango-base.display=172.25.29.163:0 --set ska-tango-base.xauthority=/home/ubuntu/.Xauthority -
-set ska-tango-base.jive.enabled=false --set webjive.enabled=false --set tango_example.tango_example.image.tag=0.4.15-dirty --set event_generator.events_generator.image.tag=0.4.15-dirty --values gilab_
values.yaml \
 charts/test-parent/ --namespace ska-tango-examples; \
 rm gilab_values.yaml
Release "test" does not exist. Installing it now.
NAME: test
LAST DEPLOYED: Mon Oct 11 16:04:58 2021
NAMESPACE: ska-tango-examples
STATUS: deployed
REVISION: 1
TEST SUITE: None
(venv) ubuntu@LAPTOP-5LBGJH83:~/ska-tango-examples$
```

Development of a device step 4: wait and watch

- «make wait» wait for all pods to be running
- «make watch» to see what's happening



Development of a device step 5: tests against a real deployment

- Done with the “make test-deployment” target in the Makefile
- The make test:
  - compress the tests folder
  - create a new pod (using the image of the repository)
  - run the pytest with the true-context option
  - once done, it retrieves the files generated.
- Please note that some tests can be done only with a real deployment and therefore they are marked as “post-deployment” tests.

File Edit Selection View Go Run Terminal Help — output.txt - ska-tango-examples [WSL: Ubuntu-20.04] - Visual Studio Code

SOURCE CONTROL

ska-tango-exa... ST-982*

Message (Ctrl+Enter to commit on 'ST-982')

Changes 6

- output.txt U
- values.yaml charts/ska-tango-examples M
- workshopcounters.yaml charts/ska-tan... U
- TangoWorkshopCounter.py src/ska_ta... U
- TangoWorkshopCounter.xmi src/ska_t... U
- test_workshop.py tests/unit 3, U

.make Git ST-982*

Message (Ctrl+Enter to commit on 'a79b3...

Makefile | requirements.txt | requirements-dev.txt | values.yaml M | test_workshop.py 3, U | output.txt (Untracked) U | output.txt U | workshop

output.txt

```
1761  INFO     ska_tango_examples.DevFactory:DevFactory.py:44 Creating Proxy for test/asynctabata/1
1762  PASSED                   [ 97%]
1763  tests/unit/test_tabata.py::test_fatabata
1764  ------------------------------ live log setup -------------------------------
1765  INFO     root:conftest.py:43 true context: True
1766  ------------------------------ live log call -------------------------------
1767  INFO     root:test_tabata.py:93 None
1768  INFO     ska_tango_examples.DevFactory:DevFactory.py:44 Creating Proxy for test/tabata/1
1769  INFO     ska_tango_examples.DevFactory:DevFactory.py:44 Creating Proxy for test/fatabata/1
1770  INFO     ska_tango_examples.DevFactory:DevFactory.py:44 Creating Proxy for test/counter/prepare
1771  INFO     ska_tango_examples.DevFactory:DevFactory.py:44 Creating Proxy for test/counter/work
1772  INFO     ska_tango_examples.DevFactory:DevFactory.py:44 Creating Proxy for test/counter/rest
1773  INFO     ska_tango_examples.DevFactory:DevFactory.py:44 Creating Proxy for test/counter/cycles
1774  INFO     ska_tango_examples.DevFactory:DevFactory.py:44 Creating Proxy for test/counter/tabatas
1775  PASSED                   [ 98%]
1776  tests/unit/test_workshop.py::test_increment PASSED                   [100%]
1777
1778  ============================== warnings summary ===============================
1779  ../usr/local/lib/python3.7/dist-packages/tango/utils.py:181
1780    /usr/local/lib/python3.7/dist-packages/tango/utils.py:181: DeprecationWarning: `np.str` is a deprecated alias for the builtin `str`. To
1781    Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
1782      CmdArgType.DevString: numpy.str,
1783
1784  -- Docs: https://docs.pytest.org/en/stable/warnings.html
1785  ------------ generated json file: /app/build/reports/cucumber.json -------------
1786  ------------ generated xml file: /app/build/reports/unit-tests.xml -------------
1787  ---------------------------------- JSON report ----------------------------------
1788  report saved to: build/reports/report.json
1789
1790  ---------- coverage: platform linux, python 3.7.3-final-0 -----------
1791  Coverage HTML written to dir build/htmlcov
1792  Coverage XML written to file build/reports/code-coverage.xml
1793
1794  ============= 76 passed, 1 skipped, 1 warning in 92.17s (0:01:32) ==============
1795  build/
1796  build/code_analysis.stdout
1797  build/htmlcov/
1798  build/htmlcov/src_ska_tango_examples_teams_LogTestUpstream_py.html
1799  build/htmlcov/src_ska_tango_examples_tabata___init___py.html
1800  build/htmlcov/status.json
```

WSL: Ubuntu-20.04 | ST-982* | Python 3.8.5 64-bit ('venv': venv) | ⊗ 0 ⚠ 3 | Ln 1778, Col 81 | Spaces: 2 | UTF-8 | LF | Plain Text

Development of a device step 6: clean

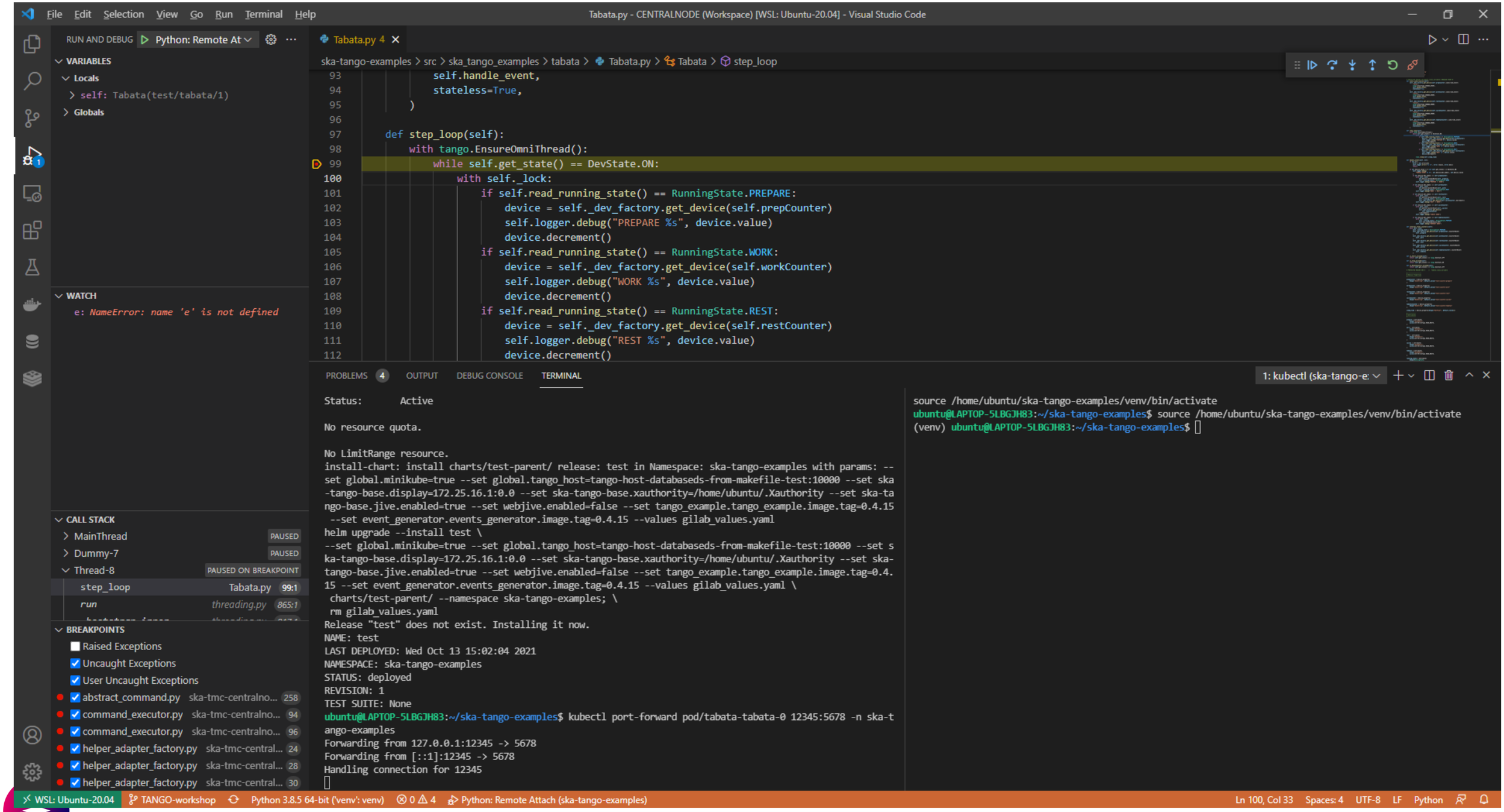
- «make uninstall-chart»
- «make reinstall-chart» if needed

Development of a device: debugging

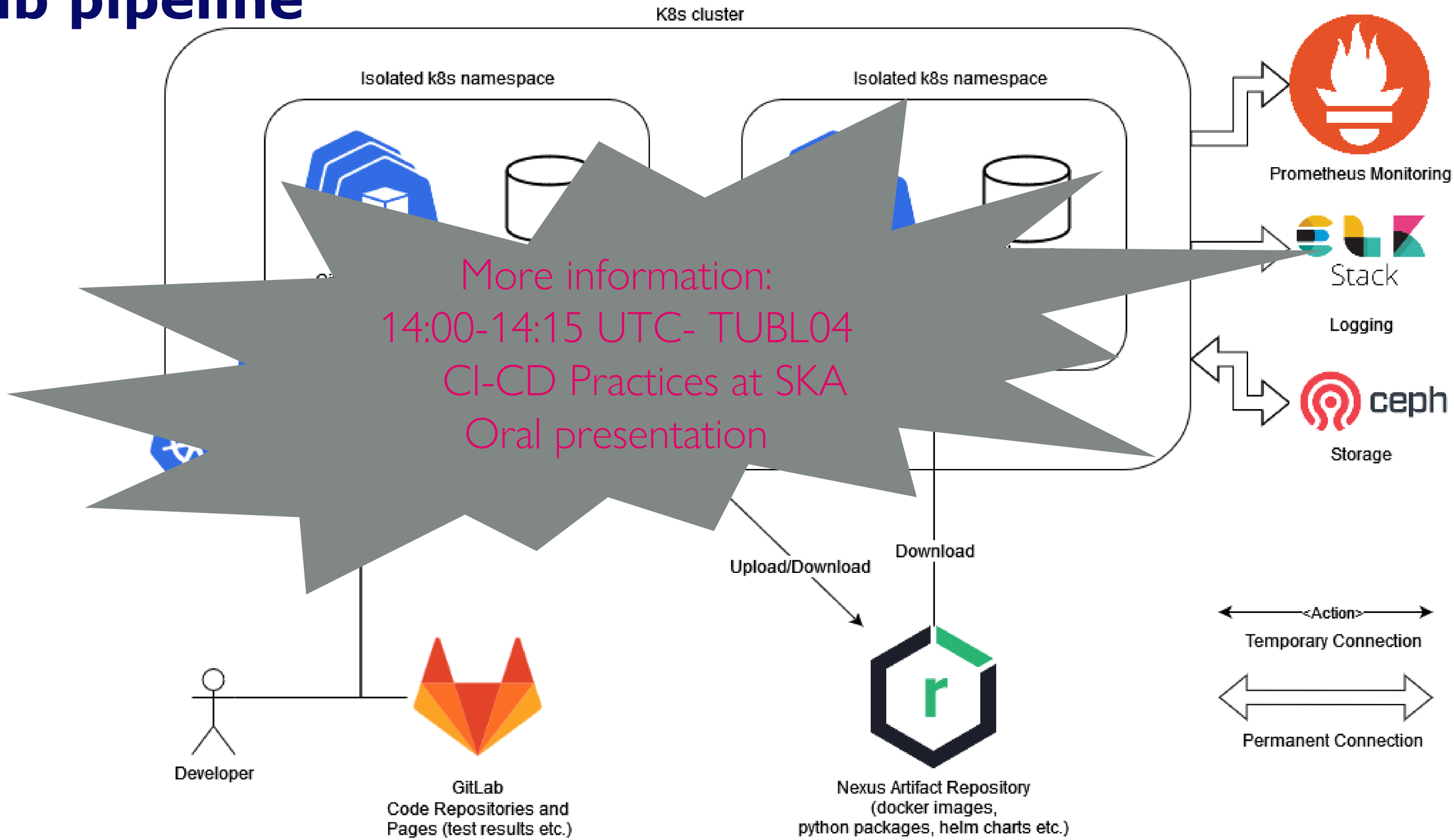
- The ska-tango-examples chart is deployed with the debugger enabled (parameter DEBUG in values.yaml file)
- K8s allows to forward a port from the local machine to a pod (the container where the device server is running)
  - `kubectl port-forward pod/tabata-tabata-0 12345:5678 -n ska-tango-examples`
- Then we can make use of the vscode attach to remote process (library debugpy)

Debugging consideration debug_this_thread

- A standard TANGO Device server does not use Python threads so most method calls are not debuggable unless we make them aware of the debugger.
- In every method we want to debug we must add the following line of code:
  - `debugpy.debug_this_thread()`
- https://github.com/microsoft/debugpy/wiki/API-Reference#debug_this_thread
- Makes the debugger aware of the current thread, and start tracing it. Must be called on any background thread that is started by means other than the usual Python APIs (i.e. the threading module), in order for breakpoints to work on that thread.


File Edit Selection View Go Run Terminal Help
Tabata.py - CENTRALNODE (Workspace) [WSL: Ubuntu-20.04] - Visual Studio Code
RUN AND DEBUG
Python: Remote At
VARIABLES
Locals
self: Tabata(test/tabata/1)
Globals
WATCH
e: NameError: name 'e' is not defined
CALL STACK
MainThread PAUSED
Dummy-7 PAUSED
Thread-8 PAUSED ON BREAKPOINT
step_loop Tabata.py 99:1
run threading.py 865:1
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
User Uncaught Exceptions
abstract_command.py ska-tmc-centralno... 258
command_executor.py ska-tmc-centralno... 94
command_executor.py ska-tmc-centralno... 96
helper_adapter_factory.py ska-tmc-central... 24
helper_adapter_factory.py ska-tmc-central... 28
helper_adapter_factory.py ska-tmc-central... 30
Tabata.py 4
ska-tango-examples > src > ska_tango_examples > tabata > Tabata.py > Tabata > step_loop
93 self.handle_event,
94 stateless=True,
95)
96
97 def step_loop(self):
98 with tango.EnsureOmniThread():
99 while self.get_state() == DevState.ON:
100 with self._lock:
101 if self.read_running_state() == RunningState.PREPARE:
102 device = self._dev_factory.get_device(self.prepCounter)
103 self.logger.debug("PREPARE %s", device.value)
104 device.decrement()
105 if self.read_running_state() == RunningState.WORK:
106 device = self._dev_factory.get_device(self.workCounter)
107 self.logger.debug("WORK %s", device.value)
108 device.decrement()
109 if self.read_running_state() == RunningState.REST:
110 device = self._dev_factory.get_device(self.restCounter)
111 self.logger.debug("REST %s", device.value)
112 device.decrement()
PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL
1: kubectl (ska-tango-e
Status: Active
No resource quota.
No LimitRange resource.
install-chart: install charts/test-parent/ release: test in Namespace: ska-tango-examples with params: --set global.minikube=true --set global.tango_host=tango-host-databaseds-from-makefile-test:10000 --set ska-tango-base.display=172.25.16.1:0.0 --set ska-tango-base.xauthority=/home/ubuntu/.Xauthority --set ska-tango-base.jive.enabled=true --set webjive.enabled=false --set tango_example.tango_example.image.tag=0.4.15 --set event_generator.events_generator.image.tag=0.4.15 --values gilab_values.yaml
helm upgrade --install test \
--set global.minikube=true --set global.tango_host=tango-host-databaseds-from-makefile-test:10000 --set ska-tango-base.display=172.25.16.1:0.0 --set ska-tango-base.xauthority=/home/ubuntu/.Xauthority --set ska-tango-base.jive.enabled=true --set webjive.enabled=false --set tango_example.tango_example.image.tag=0.4.15 --set event_generator.events_generator.image.tag=0.4.15 --values gilab_values.yaml \
charts/test-parent/ --namespace ska-tango-examples; \
rm gilab_values.yaml
Release "test" does not exist. Installing it now.
NAME: test
LAST DEPLOYED: Wed Oct 13 15:02:04 2021
NAMESPACE: ska-tango-examples
STATUS: deployed
REVISION: 1
TEST SUITE: None
ubuntu@LAPTOP-5LBGJH83:~/ska-tango-examples$ kubectl port-forward pod/tabata-tabata-0 12345:5678 -n ska-tango-examples
Forwarding from 127.0.0.1:12345 -> 5678
Forwarding from [::1]:12345 -> 5678
Handling connection for 12345
source /home/ubuntu/ska-tango-examples/venv/bin/activate
ubuntu@LAPTOP-5LBGJH83:~/ska-tango-examples$ source /home/ubuntu/ska-tango-examples/venv/bin/activate
(venv) ubuntu@LAPTOP-5LBGJH83:~/ska-tango-examples$
WSL: Ubuntu-20.04 TANGO-workshop Python 3.8.5 64-bit ('venv': venv) 0 4 Python: Remote Attach (ska-tango-examples)
Ln 100, Col 33 Spaces: 4 UTF-8 LF Python

Gitlab pipeline



Thanks for your attention

We recognise and acknowledge the Indigenous peoples and cultures that have traditionally lived on the lands on which our facilities are located.

SKAO

www.skao.int