TANGO Admin tools for SOLEIL – Castor and monitoring

A.Tison: engineer apprentice, developer
G.Abeillé: developer, product owner

SOLEIL II
La science éclaire l'avenir
Science lights up the future

Context

SOLEIL II
La science éclaire l'avenir
Science lights up the future

TANGO Context at SOLEIL

There are currently 40,000 TANGO devices deployed when considering the different control systems at SOLEIL, and for SOLEIL II there will be even more devices than at present.

Moreover, with the construction of SOLEIL II, there will be a need to create test benches, along with their dedicated Tango databases and sets of devices to deploy to test their behavior.

Context

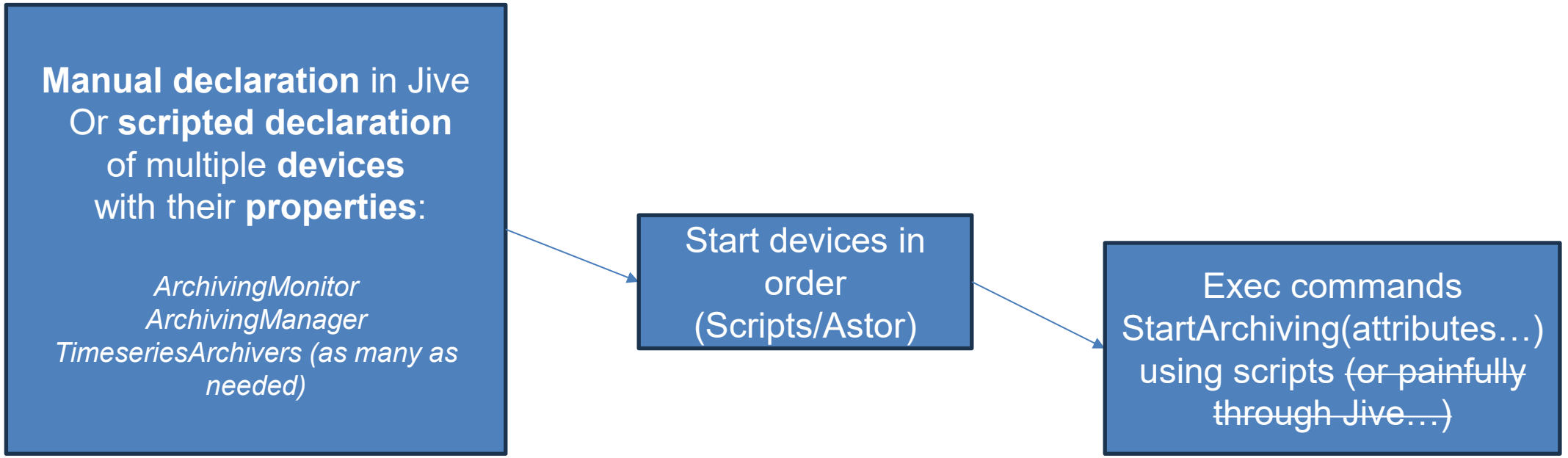
- To operate the TANGO control system, several needs exist:
  - Deploy / start / configure / stop / delete devices
  - Assist with configuring devices that have dynamic attributes (PLCServer, Pandabox, ModbusDataViewer, etc.)
  - Understand the root cause of incidents through monitoring
  - Manage attribute archiving

- Some of these needs — such as deployment, device configuration, and archiving — are currently handled at SOLEIL through bash/Python scripts and GUIs (Jive, Astor, etc.).

- Other needs, such as ensuring monitoring are not being addressed entirely.

Example use case: deploying TANGO archiving

To start TANGO archiving on a new server, we need:



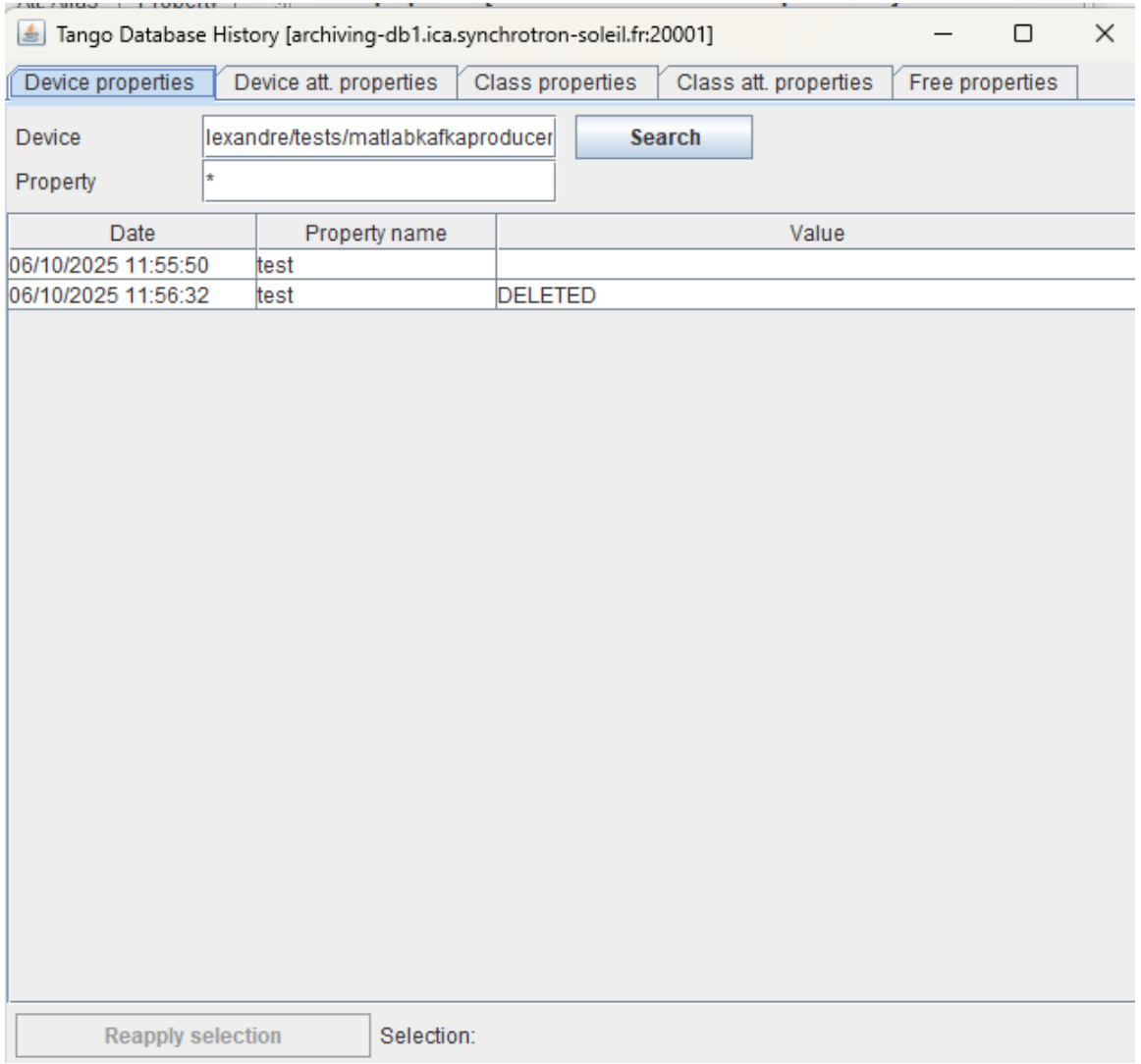
=> 2-3 separate scripts are needed to perform this operation, even though the need for these use cases is extremely common.

Example use case: Monitoring

Recurring issues in operations are:

- Changes in device configuration
- Untracked creation / deletion of devices

-> The current solution is to consult the history provided by Jive for most recent changes, but only for properties.



TANGO device deployment service

CAstor (*means Beaver in French*)

SOLEIL II
La science éclaire l'avenir
Science lights up the future

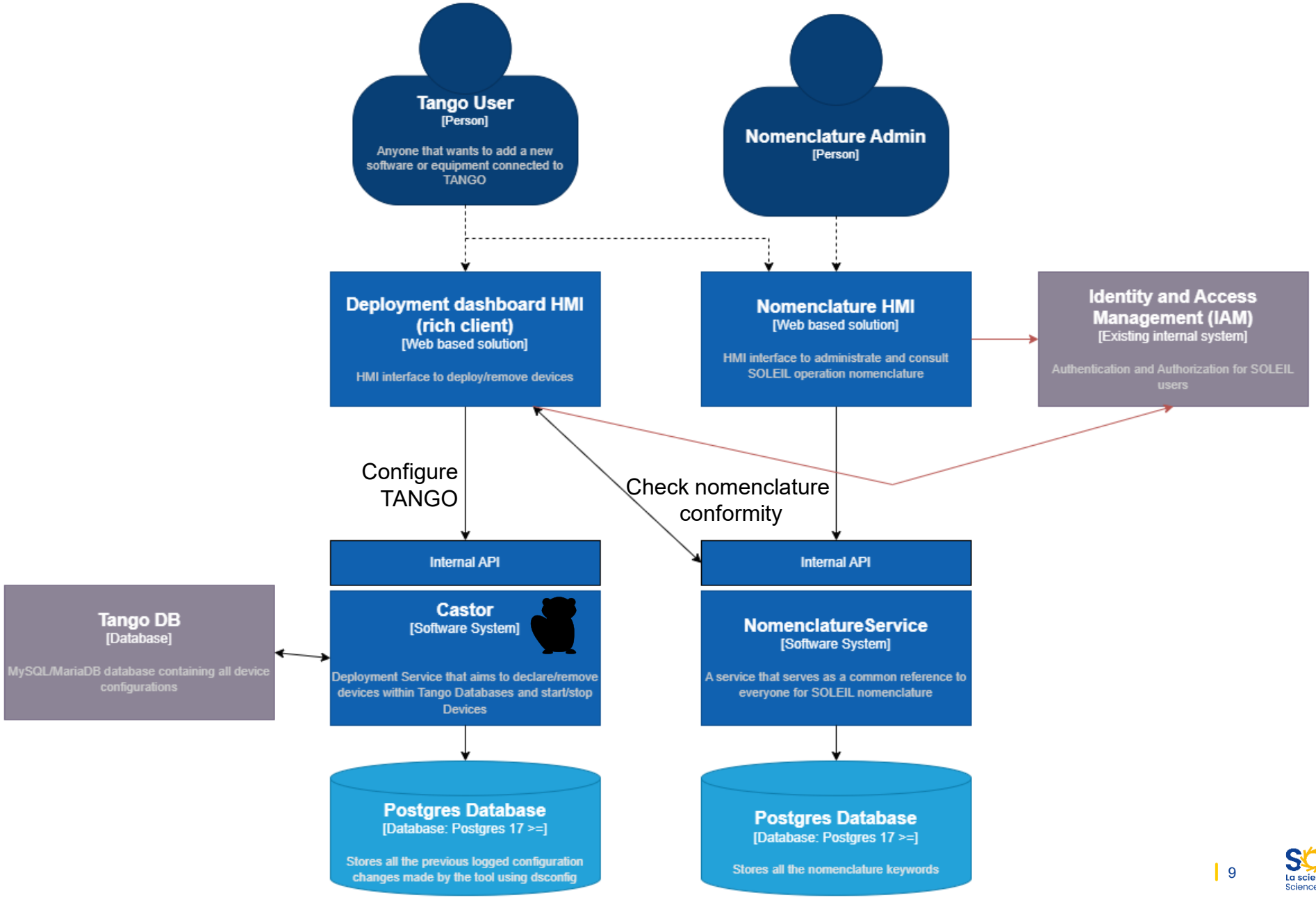
Deployment service: Castor

This service is available on SOLEIL GitLab: https://gitlab.synchrotron-soleil.fr/software-control-system/web/tangops/castor

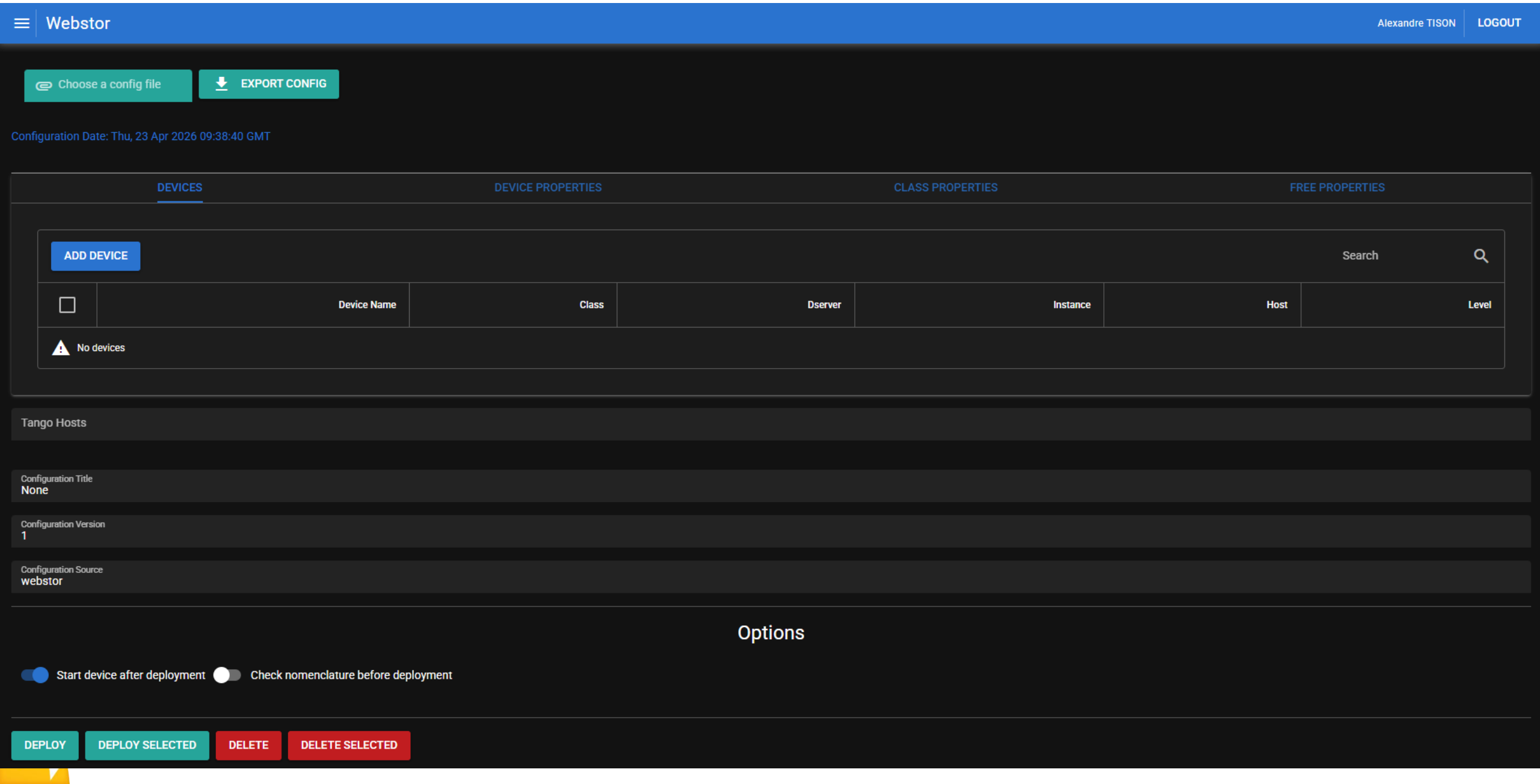
- Written in Python (FastAPI) with a PostGreSQL database. Tested using Docker & Conda
- Can be used as an API or as a CLI tool
- Allows the creation/deletion of devices as well as starting/stopping them
- Creation/Deletion of devices can be done using CSV, JSON or YAML files
- Can deploy devices on multiple tango hosts

A web front-end prototype (Vue.js) is also available (https://gitlab.synchrotron-soleil.fr/software-control-system/web/tangops/webstor-vue).

Castor architecture



Web interface



Deployment service : next steps

- Create an /UPDATE endpoint to enable post-start configuration of devices, covering settings such as polling and logging.
- “dry-run” option: Check changes on TANGO DB before applying them
- Templates for classes, especially for devices with dynamic attributes (PLCDataViewer, …)
- Improve (Web/CLI) interfaces for users

TANGO Database Reporting

SOLEIL II
La science éclaire l'avenir
Science lights up the future

Context

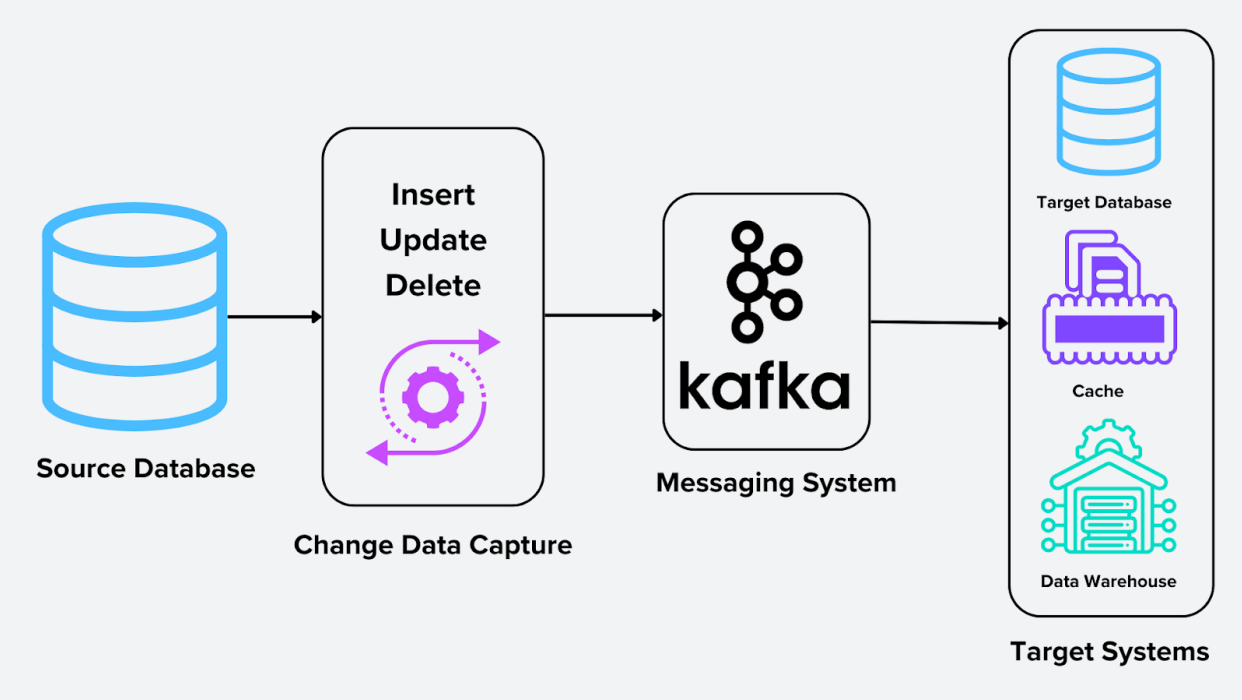
- No monitoring of the TANGO database (e.g., when a device was created or deleted).
- The existing monitoring for properties requires some familiarity with Jive and TANGO and remains limited.
- More subjectively, this same monitoring does not allow you to see the latest changes in a TANGO database at a glance.

Solution: Change Data Capture

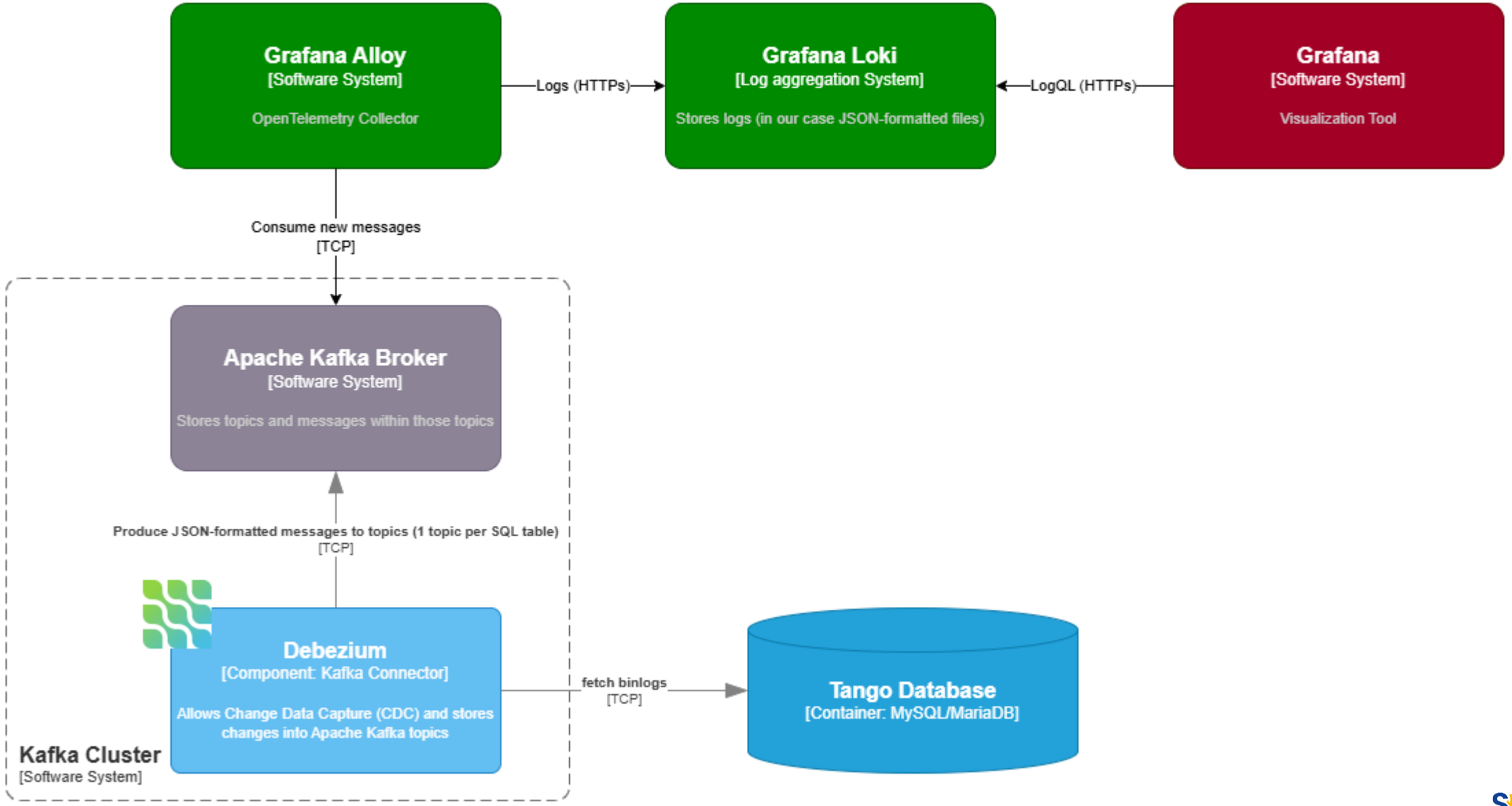
Change Data Capture (**CDC**) is a mechanism whose purpose is to **keep a history of changes** made to the data in one or more tables of a database.
This history makes it possible to reconstruct all the steps that led to the state of a table at a given point in time.

In practice:

- This mechanism is supported by most modern databases (MariaDB, MySQL) through specific log files (binlogs) that contain SQL queries
- And the changes they produced a message on each change (INSERT, UPDATE, ..).



CDC Solution with Apache Kafka (Debezium)



Concrete example of JSON-formatted change event message


```
{
 "before": {
  "device": "sys/database/2",
  "name": "tes",
  "domain": "",
  "family": "",
  "member": "",
  "count": 1,
  "value": "",
  "updated": "2025-07-18T08:17:59Z",
  "accessed": "2025-07-18T08:17:59Z"
 },
 "after": {
  "device": "sys/database/2",
  "name": "tes",
  "domain": "",
  "family": "",
  "member": "",
  "count": 1,
  "value": "qer",
  "updated": "2025-07-18T08:18:05Z",
  "accessed": "2025-07-18T08:18:05Z"
 },
"op": "u",
"ts_ms": 1752826765376,
 "source": {
  "version": "2.4.2.Final",
  "connector": "mysql",
  "name": "mysql-01",
  "ts_ms": 1752826685000,
  "snapshot": "false",
  "db": "tango",
  "table": "property_device",
  "server_id": 1,
  "file": "mysql-bin.000003",
  "pos": 39200,
  "row": 0,
  "thread": 123
 }
}
```

Issues met with the current Database Device

The Database device that feeds the MySQL/MariaDB database performs queries that programs like Debezium can correctly interpret, but which clutter the information received by the end user.

Example: creating a device triggers a pre-emptive deletion of the device:

```
// first delete the tuple (device,name) from the device table

sql_query_stream << "DELETE FROM device WHERE name LIKE \"" << tmp_device << "\"";
DEBUG_STREAM << "DataBase::AddDevice(): sql_query " << sql_query_stream.str() << std::endl;
simple_query(sql_query_stream.str(), "db_add_device()", al.get_dch());

// then insert the new value for this tuple

sql_query_stream.str("");
if(server_device->length() < 4)
{
    sql_query_stream << "INSERT INTO device SET name=\"" << tmp_device << "\",domain=\"" << domain
                     << "\",family=\"" << family << "\",member=\"" << member
                     << "\",exported=0,ior=\"nada\",host=\"nada\",server=\"" << tmp_server
                     << "\",pid=0,class=\"" << tmp_class << "\",version=\"0\",started=NULL,stopped=NULL";
}
```

-> Modification of the code to comply with Debezium : link

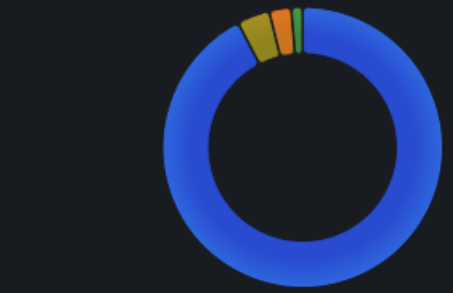
Demo with Loki/Grafana stack

Last 6 hours | Refresh

Devices deleted

4

Overall modifications inside the TANGO DB



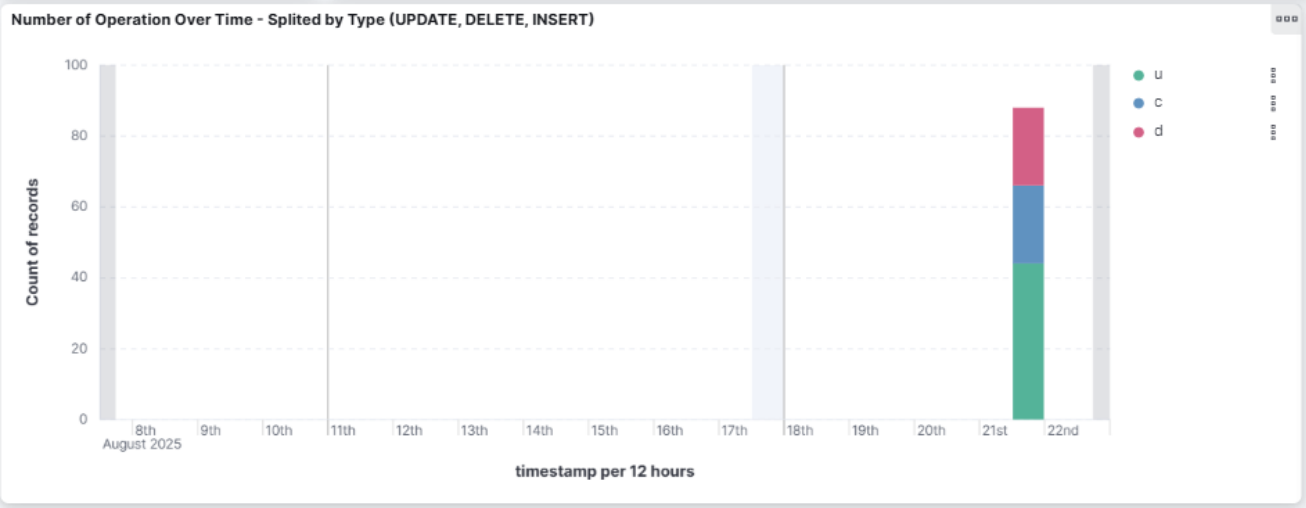
Device changes

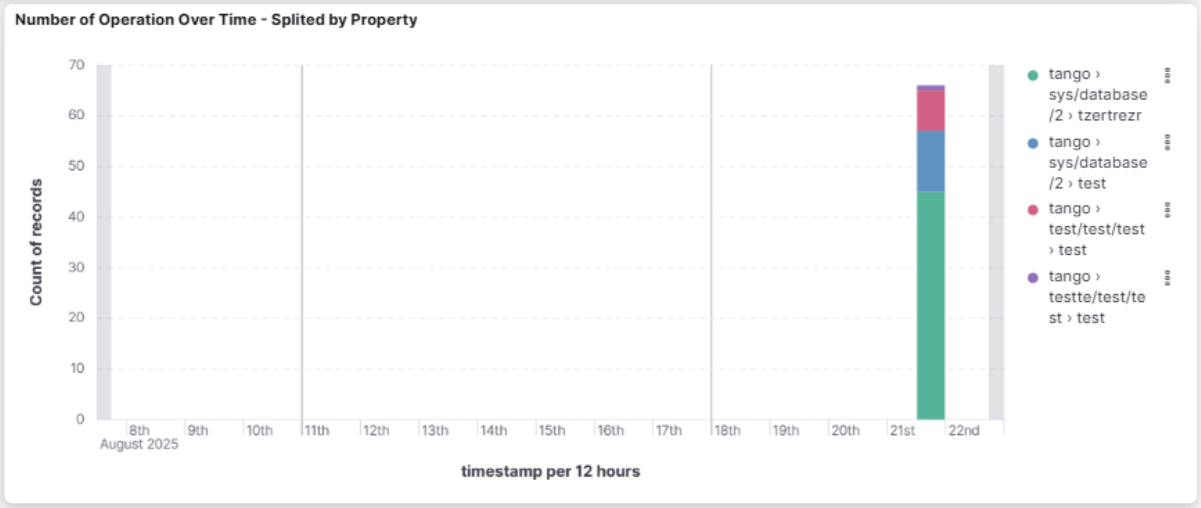
Time	op	MySQL Database	Device
2026-04-23 09:34:29	deleted	db	sys/tg_test/1
2026-04-23 09:34:29	deleted	db	dserver/TangoTest/test
2026-04-23 09:33:45	created	db	dserver/Mu/haha
2026-04-23 09:33:45	created	db	test/tango/mu.haha
2026-04-23 09:32:45	updated	db	sys/database/2
2026-04-23 09:32:45	updated	db	dserver/DataBase/2

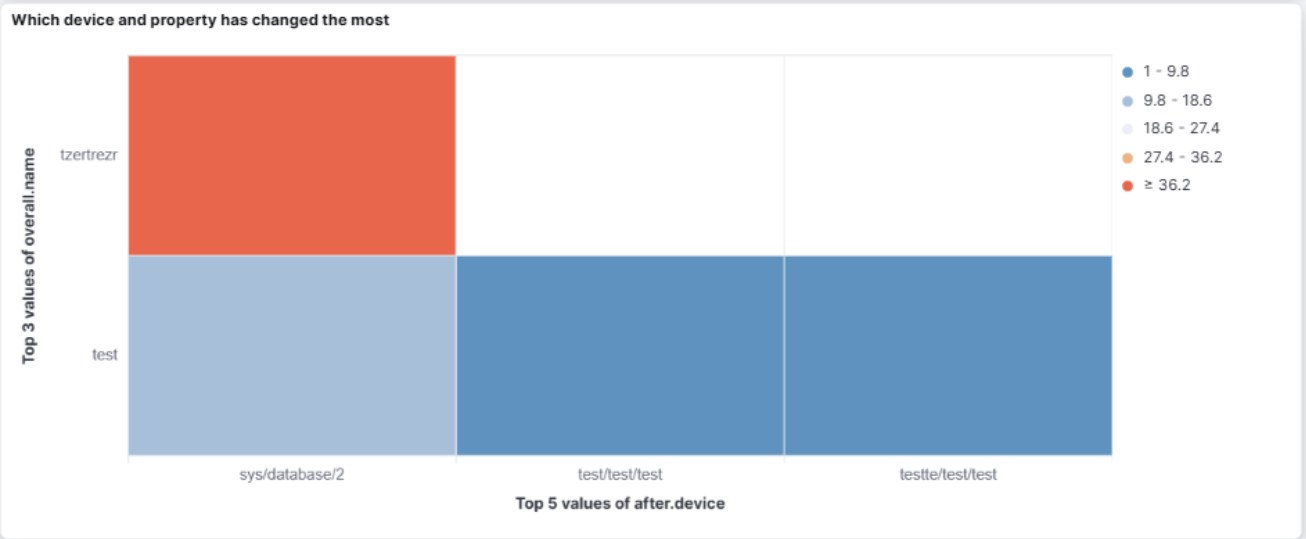
Device properties changes

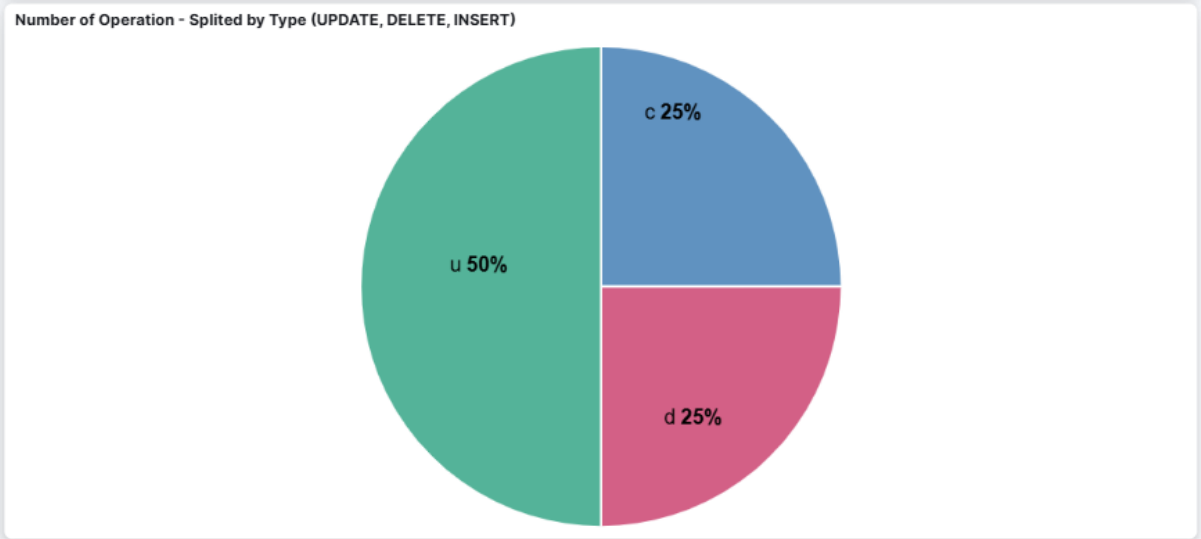
Time	MySQL Database	op	value_after	value_before	Device ↓	Property name
2026-04-23 09:34:36.431	db	deleted			test/tango/mu.haha	t
2026-04-23 09:34:13.385	db	deleted		false	test/tango/mu.haha	test device prop
2026-04-23 09:34:09.878	db	created			test/tango/mu.haha	t
2026-04-23 09:34:05.369	db	updated	false	true	test/tango/mu.haha	test device prop
2026-04-23 09:34:02.364	db	updated	true	yes	test/tango/mu.haha	test device prop
2026-04-23 09:33:58.355	db	updated	yes		test/tango/mu.haha	test device prop
2026-04-23 09:33:56.896	db	created			test/tango/mu.haha	test device prop

Demo with ELK stack









Next?

- Validate and consolidate Castor and TANGO Database reporting on SOLEIL II tests benches.
- TANGO Database device evolutions:
  - May these changes made for CDC be useful for the TANGO community?
  - We have noticed some other SQL queries optimizations that we can share

Resources

- Debezium: https://debezium.io/documentation/faq/#what_is_debezium