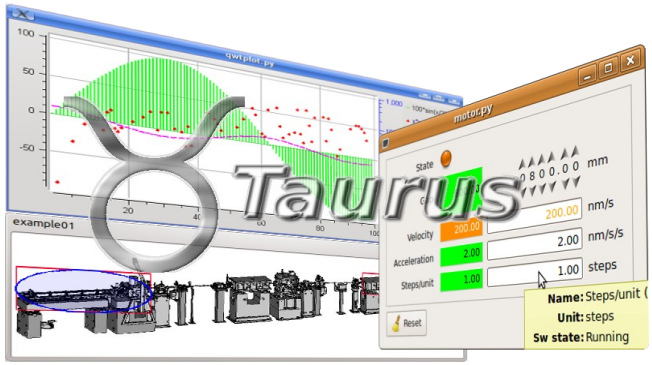
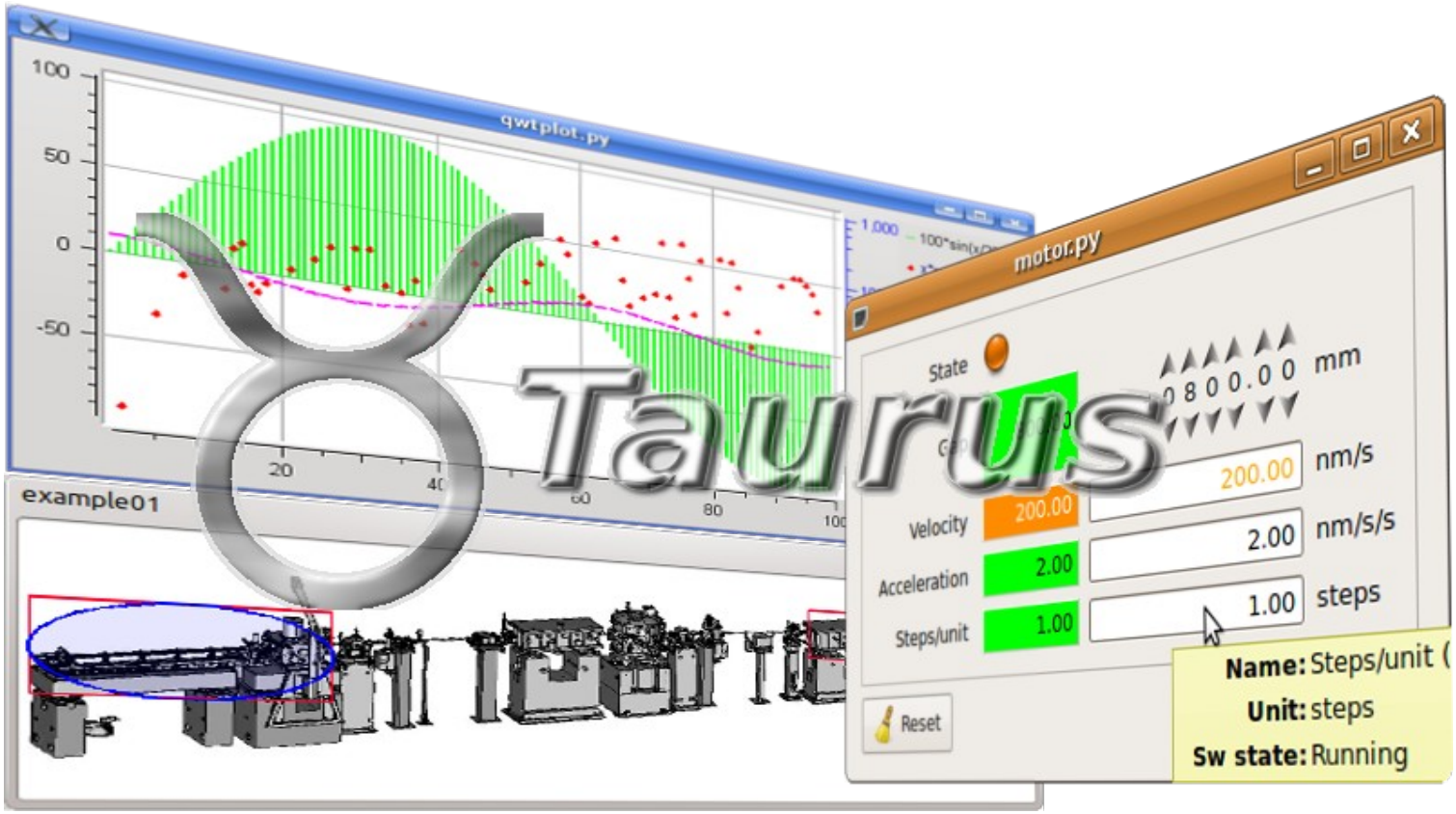
Sardana & Taurus Status



Zbigniew Reszela & Carlos Pascual (Alba Synchrotron, Spain)

on behalf of Sardana & Taurus Communities

Taurus?

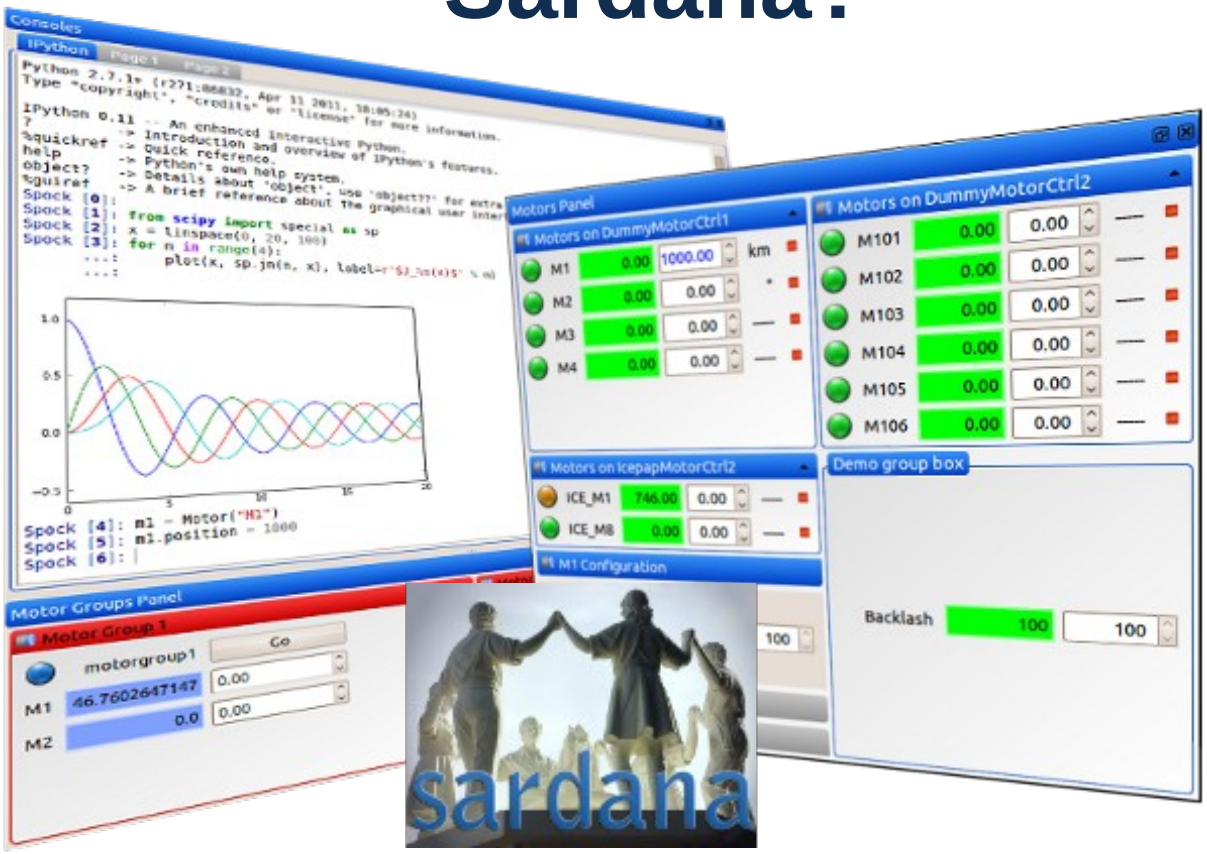


Taurus is a framework for building control and data acquisition **CLIs** and **GUIs**

It is based on **Python** and extends **PyQt**

It supports plugins for various control systems (**Tango**, **EPICS**,...) or data sources (**HDF5**, **Python eval**,...)

Sardana?

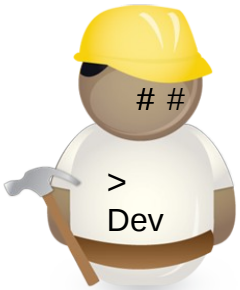


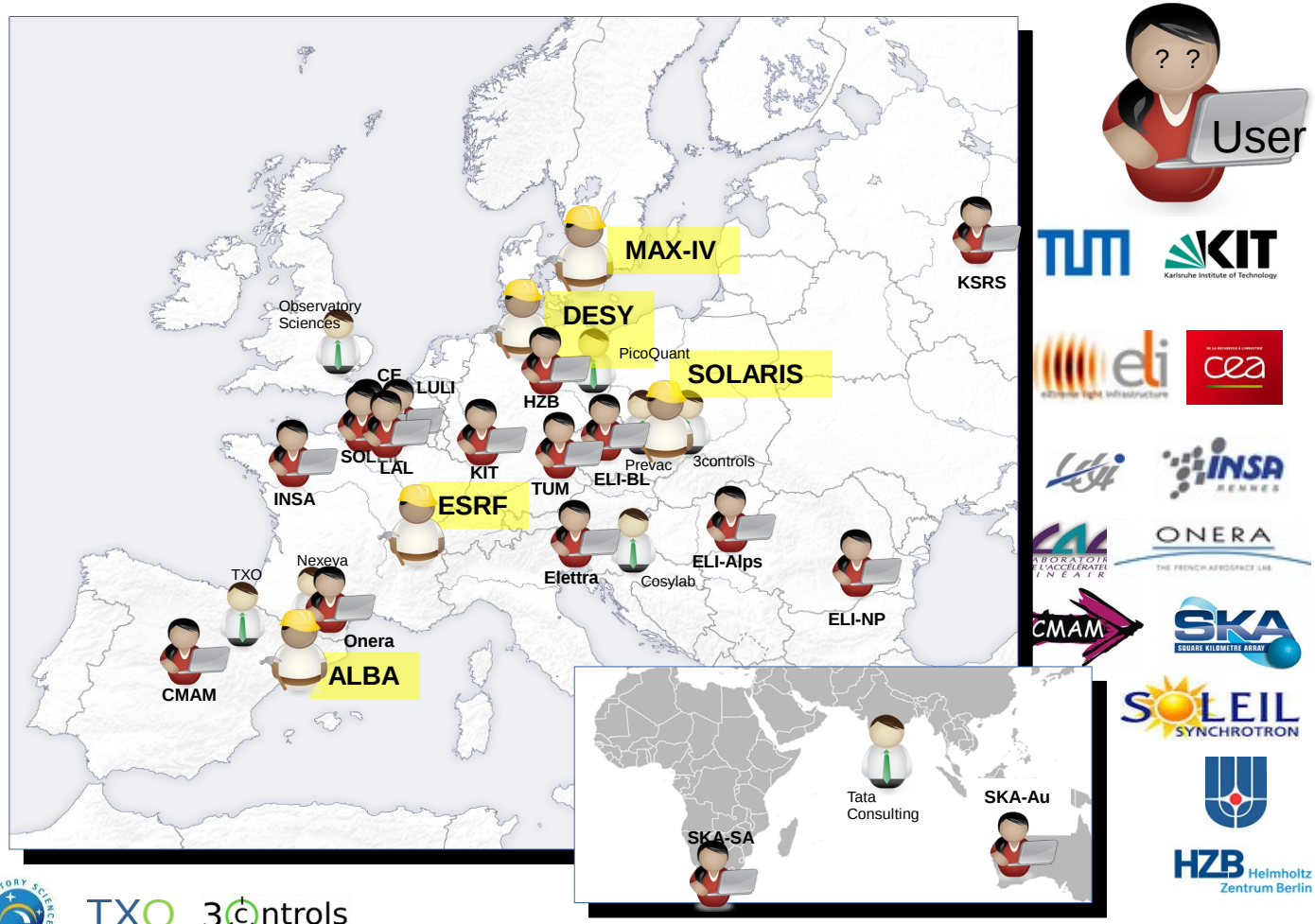
Sardana is a SCADA for scientific installations originally developed at ALBA.

It is built on top of **Taurus** and **PyTango**.

It provides **automation** of procedures and **synchronization** in a distributed control system.

Sardana & Taurus Communities





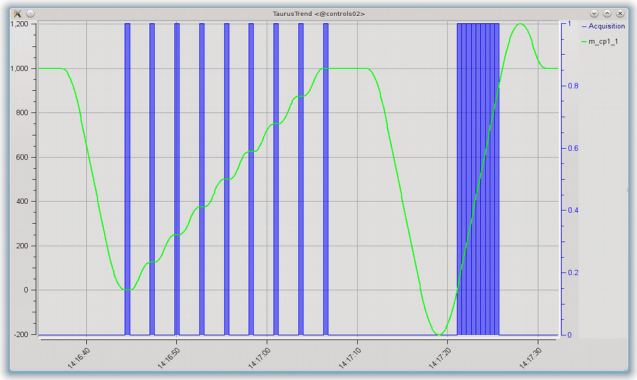
Sardana status...

News since last Tango Meeting:

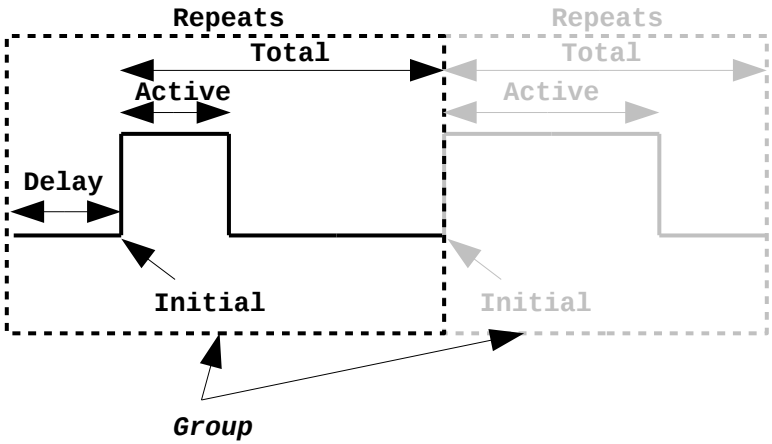
- Two releases: 2.3.2 and 2.4.0
- Update on Continuous Scans
- General Hooks
- Centralized Logging with Elastic
- Macro Logging for Users
- Element references are URIs
- Coming Soon:
  - Jul18 release and Jan19 release
  - Improve 1D and 2D experimental channels integration
  - 3rd Party Repository → Plugins Register

Continuous Scans

- Released one year ago with a set of **standard macros** underpinned by the measurement group configuration allowing **software and hardware synchronization in time and position domains**.
- New macros were added for **timescan** and **meshct.** Thanks to Roberto Homs!
- Interface for developing custom scans with arbitrary waypoints and synchronization description is **missing**.
- Results from ALBA setups presented on ICALEPCS 2017 – WEBPL06:
  - Experiment **time reduction** from several hours to few minutes.
  - **Standardized solutions** – lower maintenance cost.

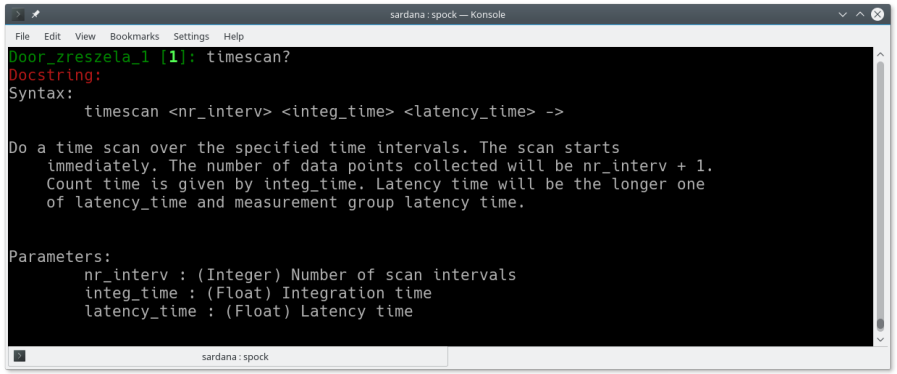


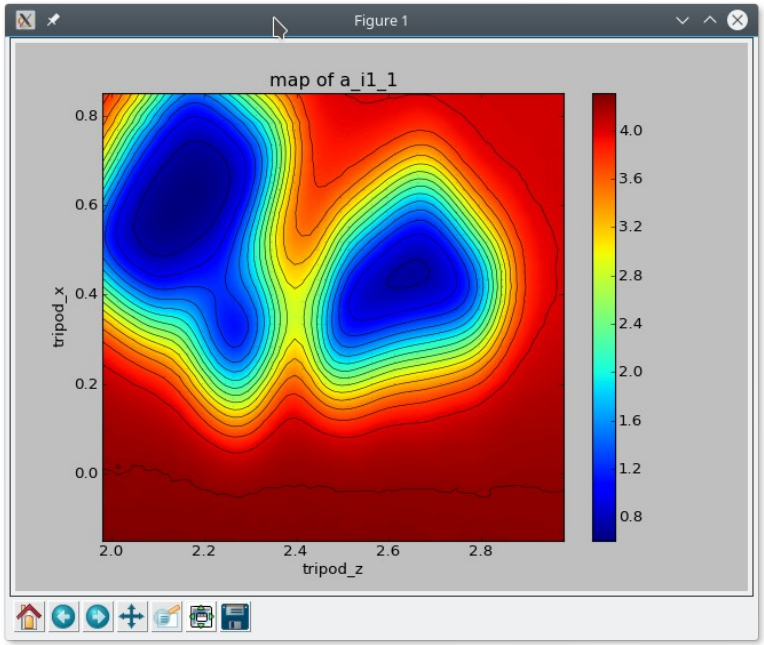
ascanct, a2scanct, a3scanct, a4scanct, dscanct, d2scanct, d3scanct, d4scanct, ...



Continuous Scans

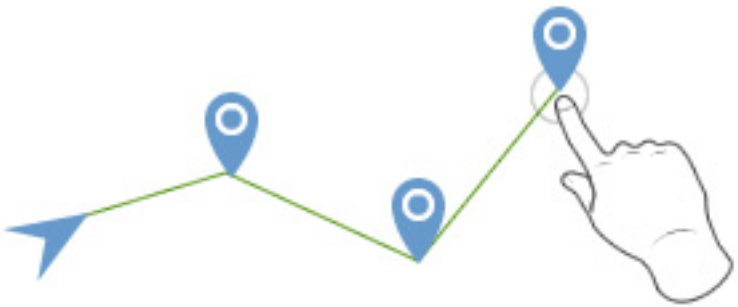
- Released one year ago with a set of **standard macros** underpinned by the measurement group configuration allowing **software and hardware synchronization in time and position domains**.
- New macros were added for **timescan** and **meshct**. Thanks to Roberto Homs!
- Interface for developing custom scans with arbitrary waypoints and synchronization description is **missing**.
- Results from ALBA setups presented on ICALEPCS 2017 – WEBPL06:
  - Experiment **time reduction** from several hours to few minutes.
  - **Standardized solutions** – lower maintenance cost.





Continuous Scans

- Released one year ago with a set of **standard macros** underpinned by the measurement group configuration allowing **software and hardware synchronization in time and position domains**.
- New macros were added for **timescan** and **meshct**. Thanks to Roberto Homs!
- Interface for developing custom scans with arbitrary waypoints and synchronization description is **missing**.
- Results from ALBA setups presented on ICALEPCS 2017 – WEBPL06:
  - Experiment **time reduction** from several hours to few minutes.
  - **Standardized solutions** – lower maintenance cost.



```
from sardana.pool import SynchParam, SynchDomain

def waypoint_generator(self):

    synchronization = [
        {
            SynchParam.Initial: {SynchDomain.Position: 0},
            SynchParam.Active: {SynchDomain.Time: 0.1},
            SynchParam.Total: {SynchDomain.Time: 3}
        },
        {
            SynchParam.Initial: {SynchDomain.Position: 3},
            SynchParam.Active: {SynchDomain.Time: 0.1},
            SynchParam.Total: {SynchDomain.Time: 1},
        },
        {
            SynchParam.Initial: {SynchDomain.Position: 4},
            SynchParam.Active:  {SynchDomain.Time: 0.1},
            SynchParam.Total:  {SynchDomain.Time: 1},
        }
    ]

    step = {}
    step["start_positions"] = [0]
    step["positions"] = [5]
    step["synchronization"] = synchronization

    yield step
```

BRAINSTORMING

Continuous Scans

- Released one year ago with a set of **standard macros** underpinned by the measurement group configuration allowing **software and hardware synchronization in time and position domains**.
- New macros were added for **timescan** and **meshct**. Thanks to Roberto Homs!
- Interface for developing custom scans with arbitrary waypoints and synchronization description is **missing**.
- Results from ALBA setups presented on ICALEPCS 2017 – WEBPL06:
  - Experiment **time reduction** from several hours to few minutes.
  - **Standardized solutions** – lower maintenance cost.

ICALEPCS2017
Barcelona · Spain, October 8-13 · Palau de Congressos de Catalunya

	Step Scan	Con-tinuous Scan 2013	Con-tinuous Scan 2017
MSPD Angular range: 100° Integration time: 25ms Intervals: 10000	~9 h 26 min	~42 min	~41 min 45 s
CLAESS Energy range: 1keV (8969 keV – 9969 keV) Integration time: 29 ms Intervals: 4000	~1 h 3 min	~3 min	~2 min 40 s
BOREAS (XMCD/XMLD) Energy range: 65 eV (755eV – 820eV) Integration time: 124ms Intervals: 4000	~1 h 25 min	~3 min	~2 min
BOREAS (reflectivity) Specular range: 47° Integration time: 0.2 s Intervals: 470	~17 min	-	~6 min 30 s
Magnetic Field Map Z axis range: 2.7 m Integration time: 0.06s Intervals: 2700	~7 h 30 min	-	~4 min

General Hooks







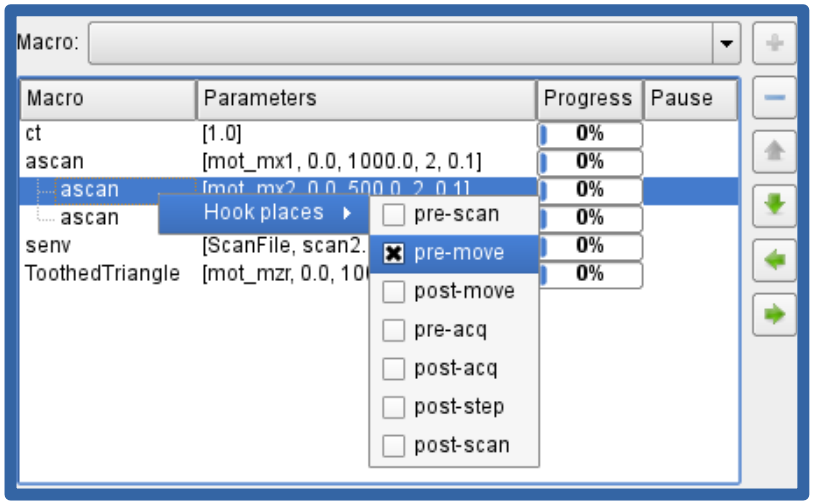
- What are hooks? **Code that will be executed at given points of the macro.**
- **Programmatic** and **graphical** way of attaching hooks was already possible.
- Now also possible by means of **persistent configuration** on different levels: global, door (session) or macro.
- Configuration via: **`lsgh`**, **`defgh`** and **`udefgh`**
- Examples:
  - wait for optimal beam conditions in pre-acq e.g. check shutter, check current
  - data analysis in post-scan
- Thanks to Teresa Nuñez from DESY!

General Hooks

```
import time
from sardana.macroserver.macro import macro


def hook(self):
    time.sleep(1)


@macro([["start", Type.Integer, None, "start point"],
        ["stop", Type.Integer, None, "end point"],
        ["step", Type.Integer, 1, "step"]])
def captain_hook(self, start, stop, step):
        loop_macro, _ = self.createMacro("loop", start,
                                         stop, step)
        loop_macro.hooks = [(hook, ["pre-acq"])]
        self.runMacro(loop_macro)
```



- What are hooks? **Code that will be executed at given points of the macro.**
- **Programmatic** and **graphical** way of attaching hooks was already possible.
- Now also possible by means of **persistent configuration** on different levels: global, door (session) or macro.
- Configuration via: **`lsgh`**, **`defgh`** and **`udefgh`**
- Examples:
  - wait for optimal beam conditions in pre-acq e.g. check shutter, check current
  - data analysis in post-scan
- Thanks to Teresa Nuñez from DESY!

General Hooks

```
File Edit View Search Terminal Help

p06/door/hasp029rack.01 [31]: lsgh
No general hooks

p06/door/hasp029rack.01 [32]: defgh 'check_beam' pre-scan post-scan
Defining general hook
check_beam

p06/door/hasp029rack.01 [33]: lsgh
   Hook place      Hook(s)
 ------------ ------------
    post-scan   check_beam
     pre-scan   check_beam

p06/door/hasp029rack.01 [34]: lsenv _GeneralHooks
            Name                                          Value   Type
 --------------- ---------------------------------------------- ------
   _GeneralHooks   [('check_beam', ['pre-scan', 'post-scan'])]   list

p06/door/hasp029rack.01 [35]: defgh 'mv exp_dmy01 10' pre-scan
Defining general hook
mv exp_dmy01 10

p06/door/hasp029rack.01 [36]: lsgh
   Hook place           Hook(s)
 ------------ -----------------
    post-scan        check_beam
     pre-scan        check_beam
                mv exp_dmy01 10

p06/door/hasp029rack.01 [37]: udefgh
Undefine all general hooks

p06/door/hasp029rack.01 [38]: lsgh
No general hooks
```

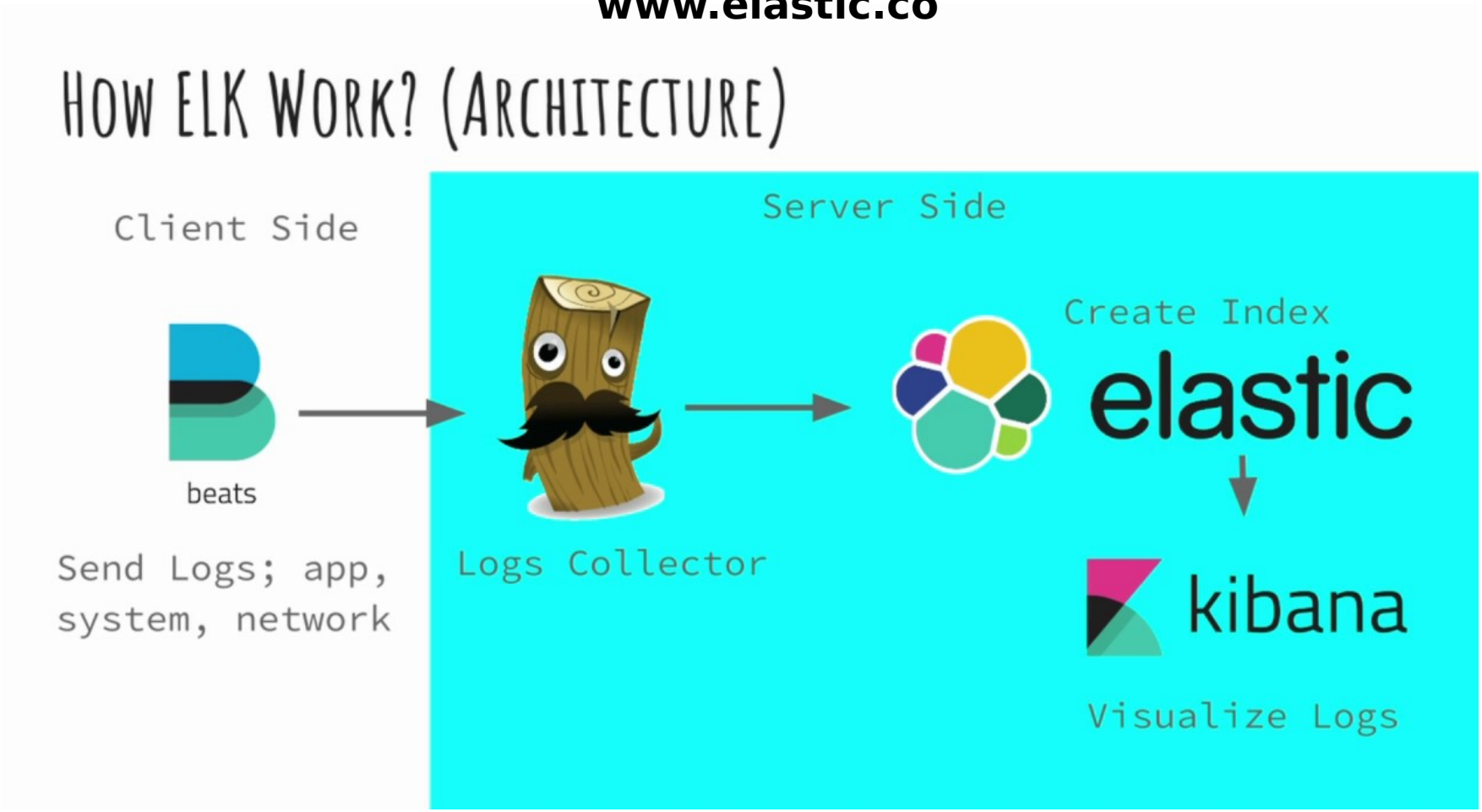
- What are hooks? **Code that will be executed at given points of the macro.**
- **Programmatic** and **graphical** way of attaching hooks was already possible.
- Now also possible by means of **persistent configuration** on different levels: global, door (session) or macro.
- Configuration via: **`lsgh`**, **`defgh`** and **`udefgh`**
- Examples:
  - wait for optimal beam conditions in pre-acq e.g. check shutter, check current
  - data analysis in post-scan
- Thanks to Teresa Nuñez from DESY!

Centralized Logs with Elastic

- Servers log to multiple and distributed files.
- Debugging requires prior files merging :(
- Sardana does not use Tango logging but Python logging.

Centralized Logs with Elastic

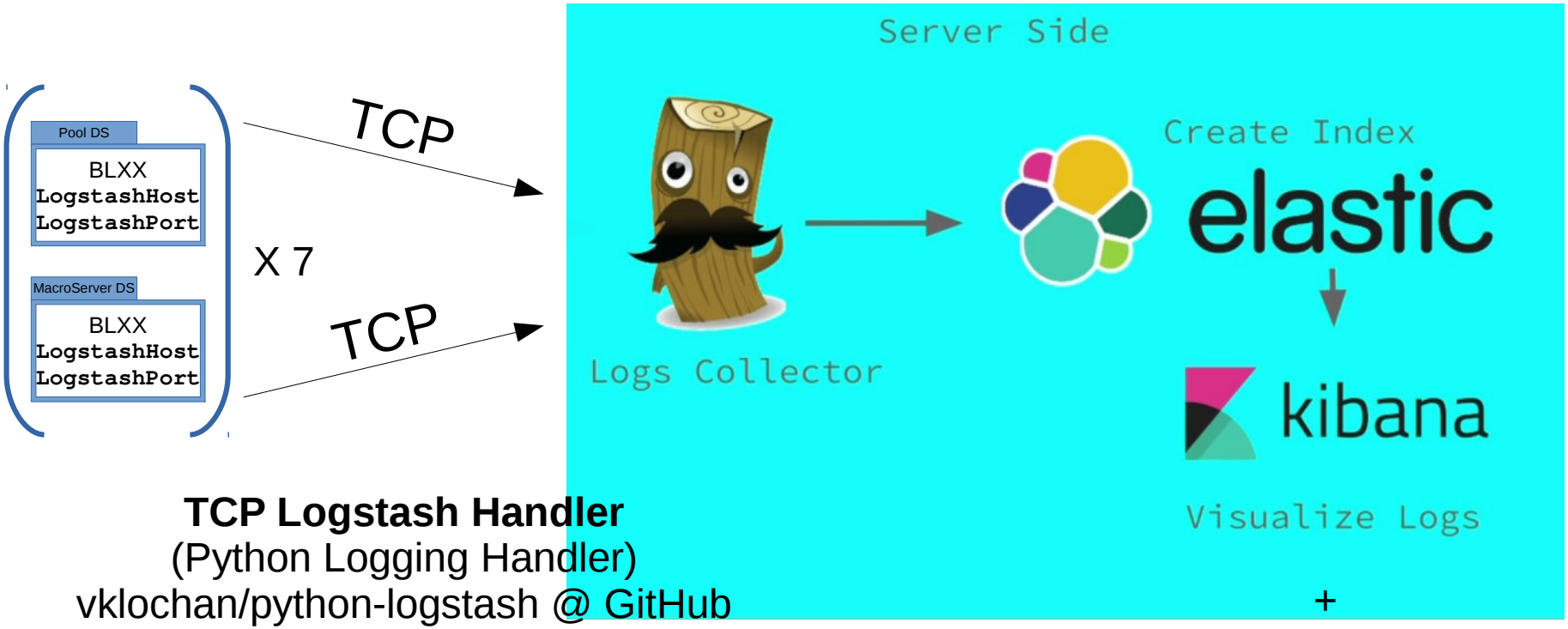
www.elastic.co



Centralized Logs using ELK (Elasticsearch, Logstash, Kibana) Stack - Estu Fardani - FOSSASIA 2017

Centralized Logs with Elastic

Proof Of Concept



TCP Logstash Handler
(Python Logging Handler)
vklochan/python-logstash @ GitHub
eht16/python-logstash-async @ GitHub

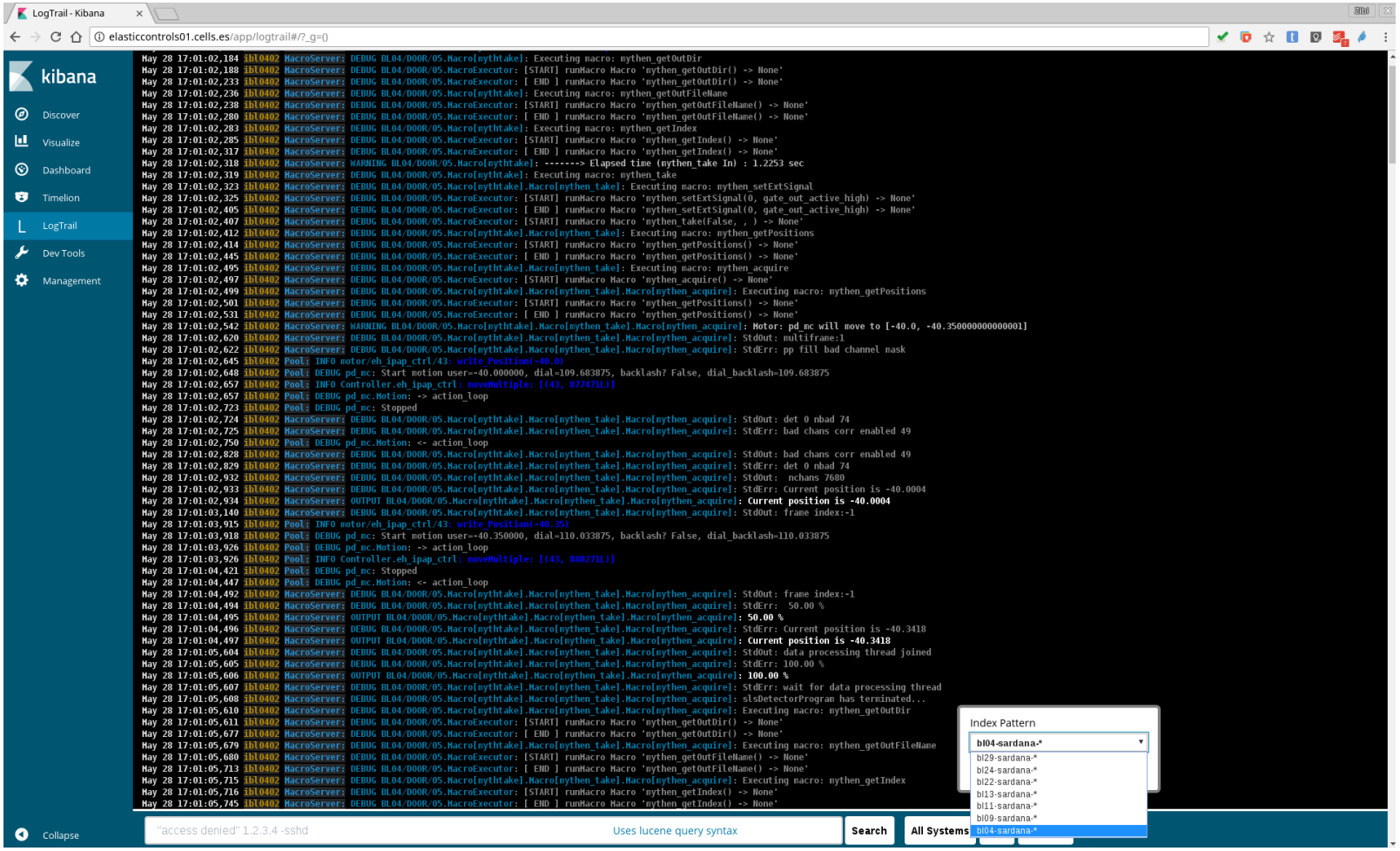
+

LogTrail
Log Viewer plugin for Kibana
sivasamyk/logtrail @ GitHub

Configuration examples: on reszelaz/sardana-elastic @ GitHub

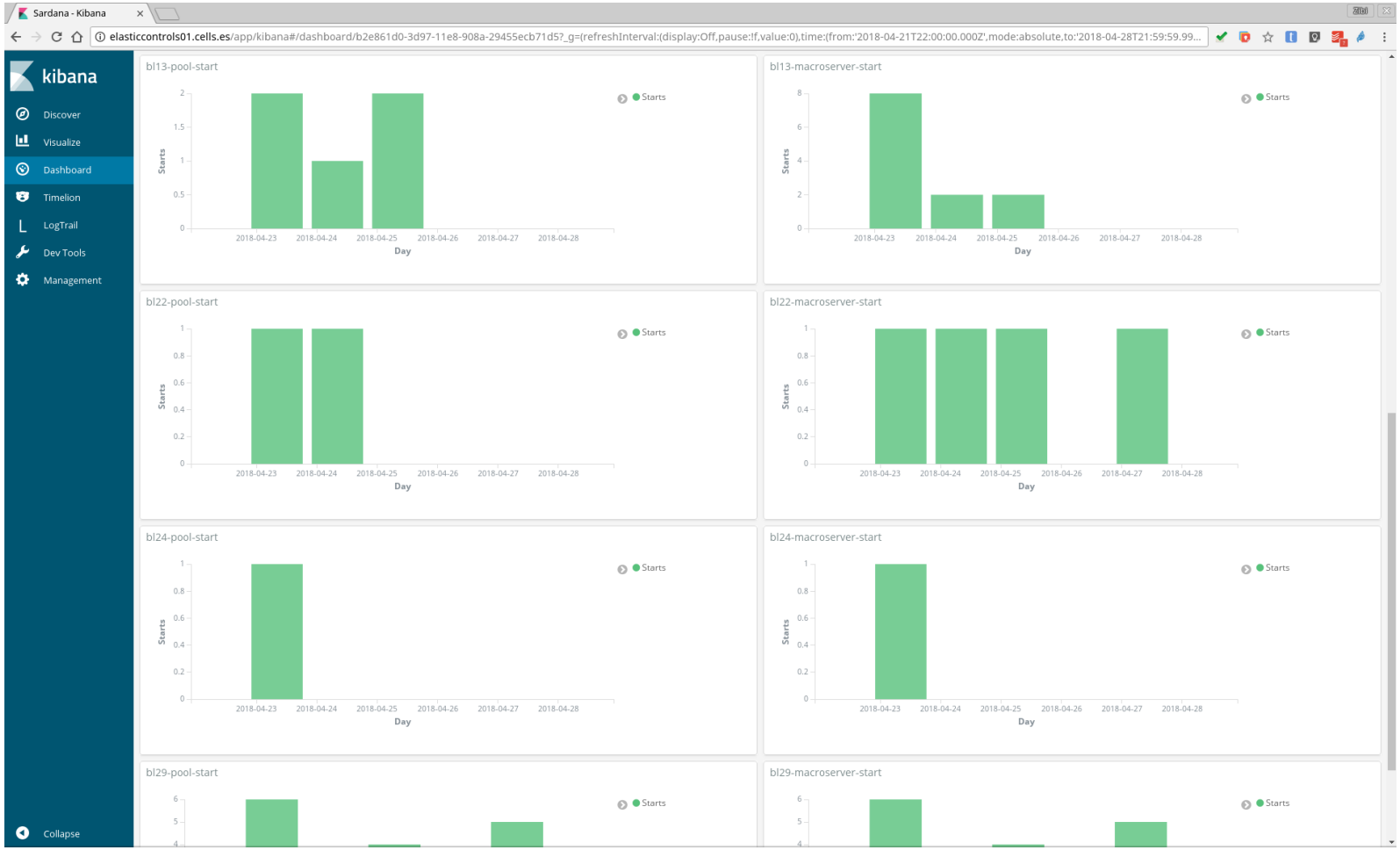
Centralized Logs with Elastic

- Benefits: **remote log analysis**, performance indicators.



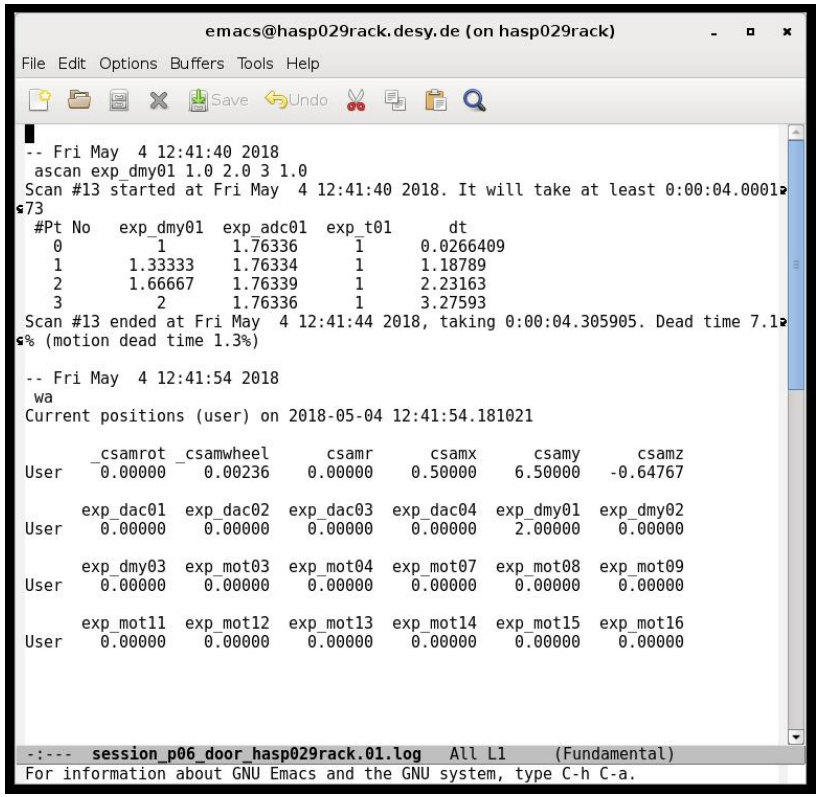
Centralized Logs with Elastic

- Benefits: remote log analysis, **performance indicators**.



Macro Logging for Users

- Macros may provide information via logging streams.
- Users requires this information to be stored in a file **as they see it in spock**.
- Implementation:
  - File log handler attached to the MacroServer
  - **`logmacro`** macro and **`LogMacroFormat`**, **`LogMacroDir`** environment variables.
- Thanks to Teresa Nuñez from DESY!



Elements References are URIs

- How elements refers to each other?
- Previously:

```
host:port/element/controller/axis
myhost:10000/motor/icepap01/7
```

- Now:

```
scheme://host[.domain]:port/element/controller/axis
tango://myhost.mydomain:10000/motor/icepap01/7
```

- Compatibility with Taurus 4

Elements References are URIs

- How elements refers to each other?
- Previously:
  `host:port/element/controller/axis`
  `myhost:10000/motor/icepap01/7`
- Now:
  `scheme://host[.domain]:port/element/controller/axis`
  `tango://myhost.mydomain:10000/motor/icepap01/7`
- Compatibility with Taurus 4

Elements References are URIs

- How elements refers to each other?
- Previously:

```
host:port/element/controller/axis
myhost:10000/motor/icepap01/7
```

- Now:

```
scheme://host[.domain]:port/element/controller/axis
tango://myhost.mydomain:10000/motor/icepap01/7
```

- Compatibility with Taurus 4

1D & 2D channels

COMING SOON

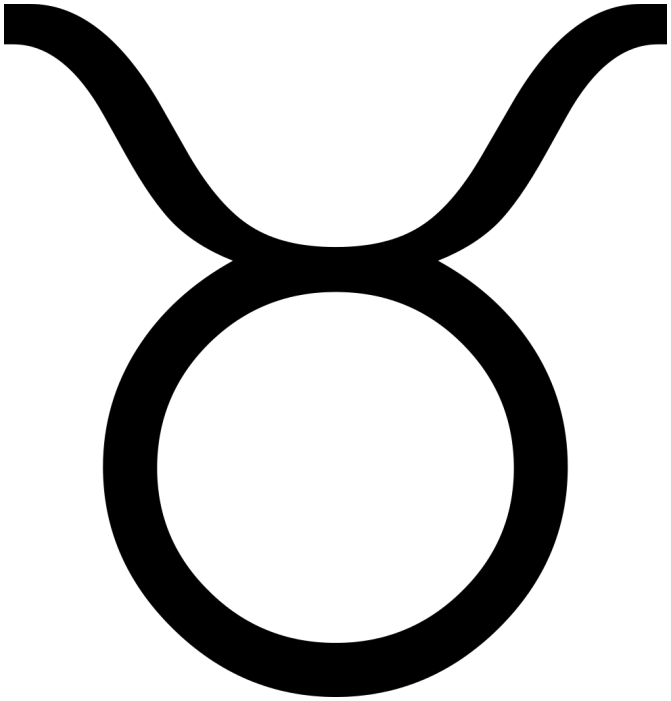
- **SEP2** will be inspired on Lima but not limited to Lima.
- **Configuration**: saving (also per experiment configuration); image e.g. ROI, binning, etc.
- **Acquisition** will be possible on different levels: channel, measurement group, scan.
  - Prior work on synchronization and acquisition are described in **SEP18**.
- **Saving**:
  - Done externally e.g. controller, Lima or directly the detector; reference to the image will be passed to the recorders.
  - Image will be passed via Tango events and saving will be done internally by the recorders.

Plugins Register

COMING SOON

- Plugins: controllers, macros, recorders, widgets/GUIs.
- Currently controllers and macros are in a **unique Git repo** on www.sf.net
- **SEP16**
- Objectives:
  - Enable natural way of working using Git (or other VCS).
  - Do not impose the repository organization.
  - Enable the possibility to use project tools: issue trackers, wikis, ...
  - Do not force the hosting platform.
  - Give visibility to the well maintained plugins.
- Idea:
  - Plugins repositories will have their **own admins**.
  - Sardana org. will **advice** on how to organize the plugin projects.
  - Sardana org. will **maintain the register** of third party plugins.

Taurus...

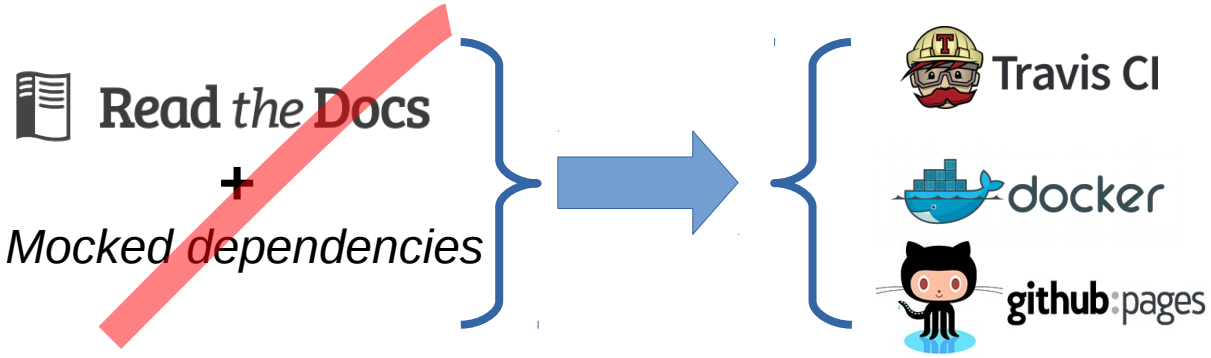


Taurus Status

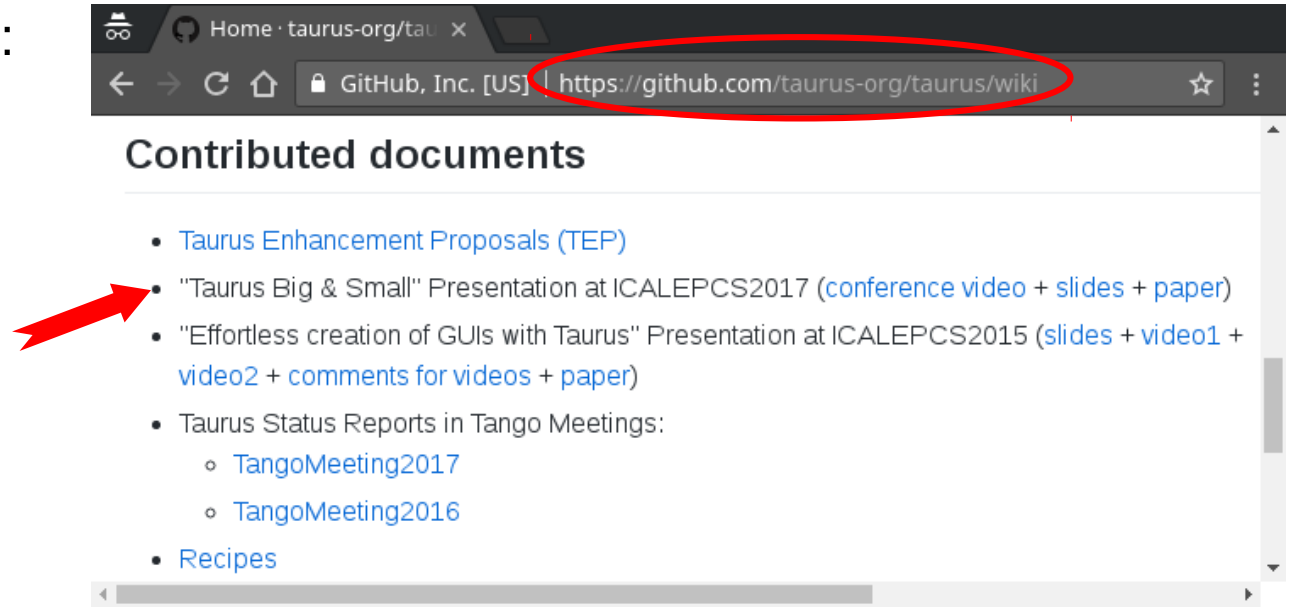
- News since last Tango meeting
  - Documentation: travis-generated docs and wiki
  - Taurus formatter API
  - (slow) Progress in PyQtgraph-based plots (TEP17)
  - Other works in progress:
    - python3 support
    - TEP15
    - Tango event serialization

GithubPages and Wiki

- Taurus docs are now generated by Travis and served by githup pages



- New project wiki:



Taurus Formatter API

- **Requirement:**
  - We want to specify the format with which numerical quantities are displayed by the Taurus widgets (note: Taurus 3 used the “tango format” field for displaying the attribute value)
- **Problem:**
  - The “format” should be a property of the **view** (widget) not of the **model** (attribute).
  - Also, it is **not scheme-agnostic**
- **Solution**:
  - Taurus **attributes** define a ***precision*** property (number of meaningful decimal places)
  - Taurus **widgets** display the value using a given ***Formatter*** (which can be either a format string or a method that returns a format string).
  - The ***defaultFormatter*** uses the attribute precision for the numerical values
  - The taurus *TangoAttribute* infers the precision from the “tango config format”
  - The Tango scheme provides a ***tangoFormatter*** which uses the full “tango config format”
  - The *Formatter* can be set programmatically or (for some widgets) via a context menu

Taurus Formatter API

Consider a Tango Attribute whose value is **123.45 V**

Tango Format	TaurusAttribute precision	Widget Formatter	Widget Format	Display
%6.2f	2	defaultFormatter	{:~.2f}	123.45 V
%4.2e	2	defaultFormatter	{:~.2f}	123.45 V
		tangoFormatter	{:~4.2e}	1.23e2 V
		myFormatter	<myFormatter>	Quantity

```
def myFormatter(dtype=type(None), **kw):
    return dtype.__name__
```

Taurus_pyqtgraph

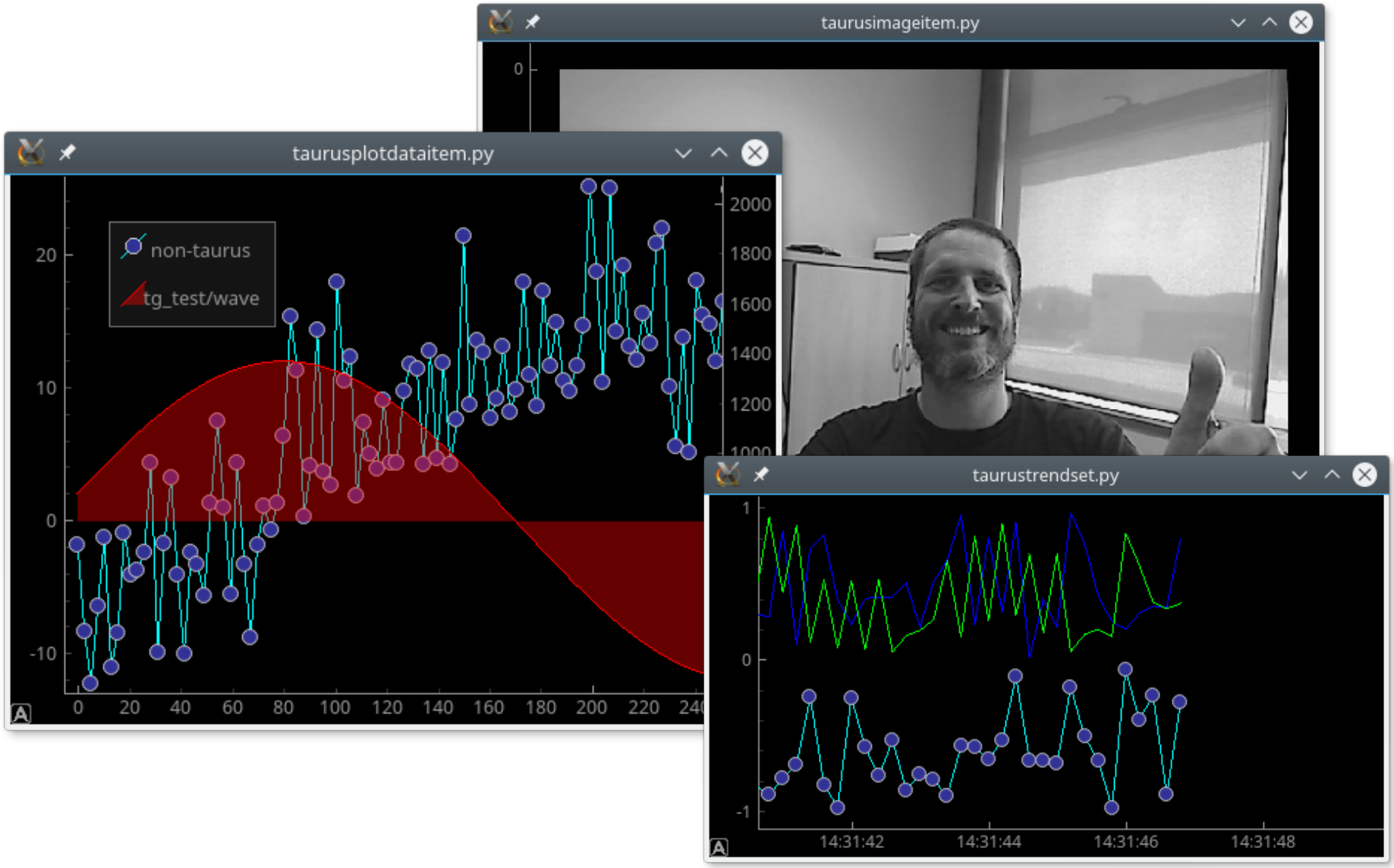
- TEP17 (https://github.com/taurus-org/taurus/pull/452). Goal: provide pyqtgraph, tool-based replacement for:
    - TaurusPlot (PyQwt)
    - TaurusTrend (PyQwt)
    - TaurusImageDialog (guiqwt)
    - TaurusTrend2DDialog (guiqwt)
  - Planned for release **Jan18** but stalled
  - Implemented as a **plugin for taurus:**

    http://github.com/taurus-org/taurus_pyqtgraph

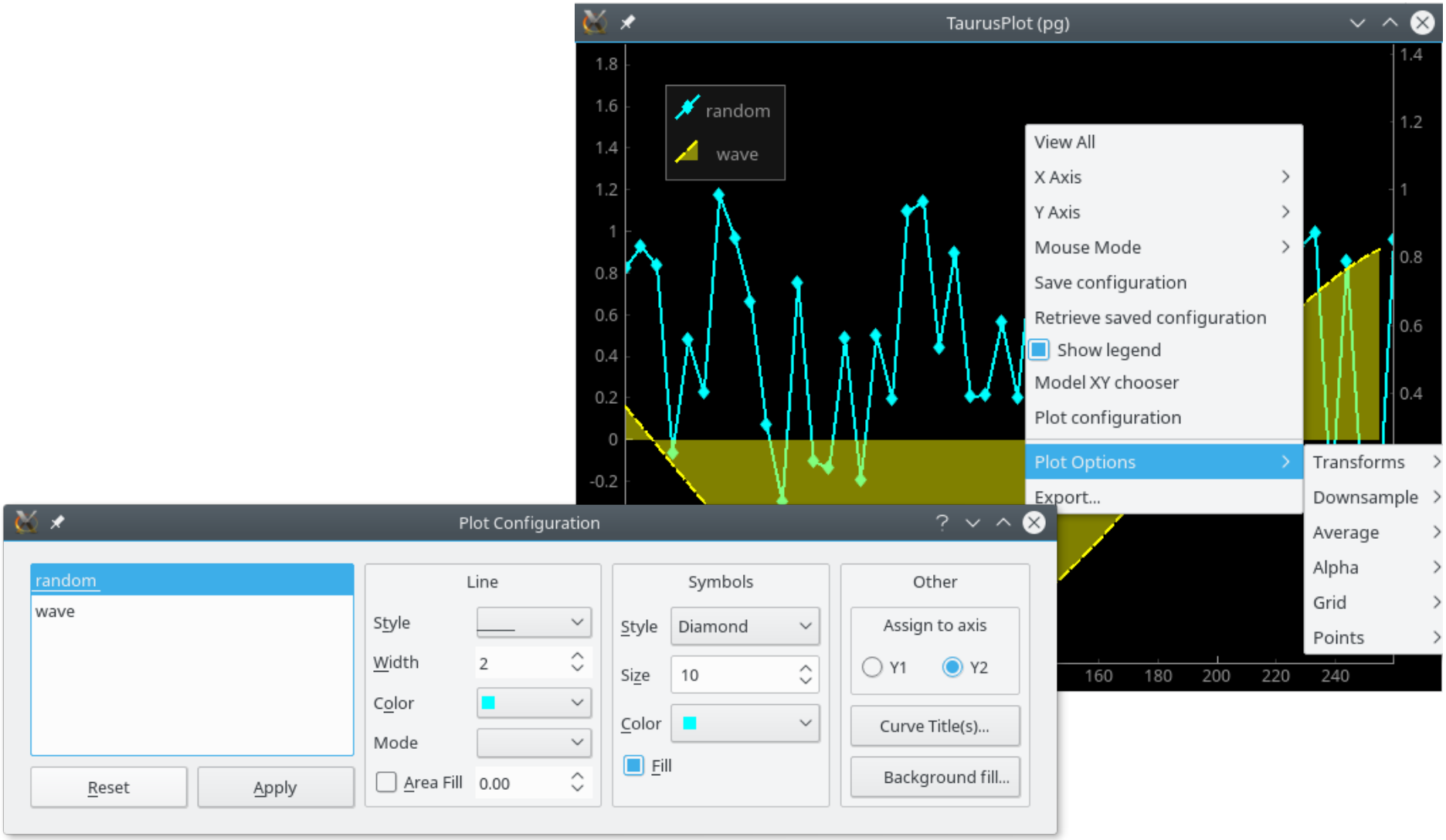
Taurus_pyqtgraph (done)

- Taurus data items (taurus-aware data items attachable to generic pyqtgraph plots):
  - **TaurusPlotDataItem**
  - **TaurusTrendSet**
  - **TaurusImageItem**
- Tools (independent “tools” attachable to generic pyqtgraph plots to extend them)
  - **Model Chooser Tools** (allow the user to select taurus models):
    - **TaurusModelChooserTool**, (for selecting a list of models)
    - **TaurusImageModelChooserTool**, (for selecting a single model)
    - **TaurusXYModelChooserTool** (for selecting a list of X and Y models + titles)
  - **DateAxisItem** (provides an axis that supports date/time)
  - **Y2ViewBox** (provides a secondary Y axis)
  - **CurvesPropertiesTool** (allows the user to change curve color, symbol, width,...)
  - **AutoPanTool** (provides “oscilloscope mode” auto-panning for trends)
  - **ForcedReadTool** (provides user-selectable client-side polling)
  - **PlotLegendTool** (allows the user to show/hide a legend)
- High-level widgets (stand-alone, taurus-aware, save/restore config support,...):
  - **TaurusPlot**
  - **TaurusTrend** (WIP)

Taurus_pyqtgraph



Taurus_pyqtgraph (done)



Taurus_pyqtgraph (missing)

- 1D trends
  - add/remove trends in high-level widget is still buggy
  - archiving support (should be provided by archiving-scheme)
- 2D plots
  - high-level widget not started
- 2D trends
  - not started

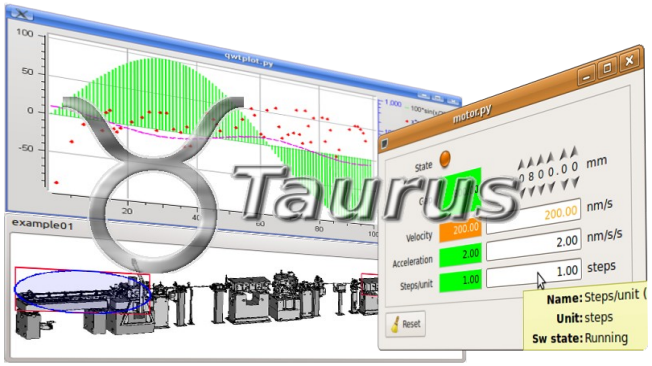
(See complete feature list in taurus_pyqtgraph README)

Other works-in-progress

- Support of Python3 (PR #680, #703)
  - First steps done (thanks MaxIV and @piertoni !)
- TEP15 (use URI fragments to reference value slices)
  - please help decide the syntax!
- Support tango.DevEnum (Issue #742)
- Manage the tango event queue in Taurus (PR #738)

Community Events

- **Sardana & Taurus Workshop** (beginners) at **ICALEPCS 2017** – sardana-org/sardana-training @ GitHub
- **Sardana & Taurus Workshop** at **Tango Meeting 2018**
  Thanks to all Participants and Tango Meeting Organizers!
- **Sardana Follow-up Meetings** - sardana-org/sardana-followup @ GitHub.
  Thanks to Grzegorz Kowalski (Solaris), Antonio Milan (MaxIV) and Teresa Nuñez (DESY)!
- **Sardana Docs Camp** – July 2018 in Barcelona



Thanks to the Community!