HZB Helmholtz Zentrum Berlin

Deployment of Bluesky @ BESSY II

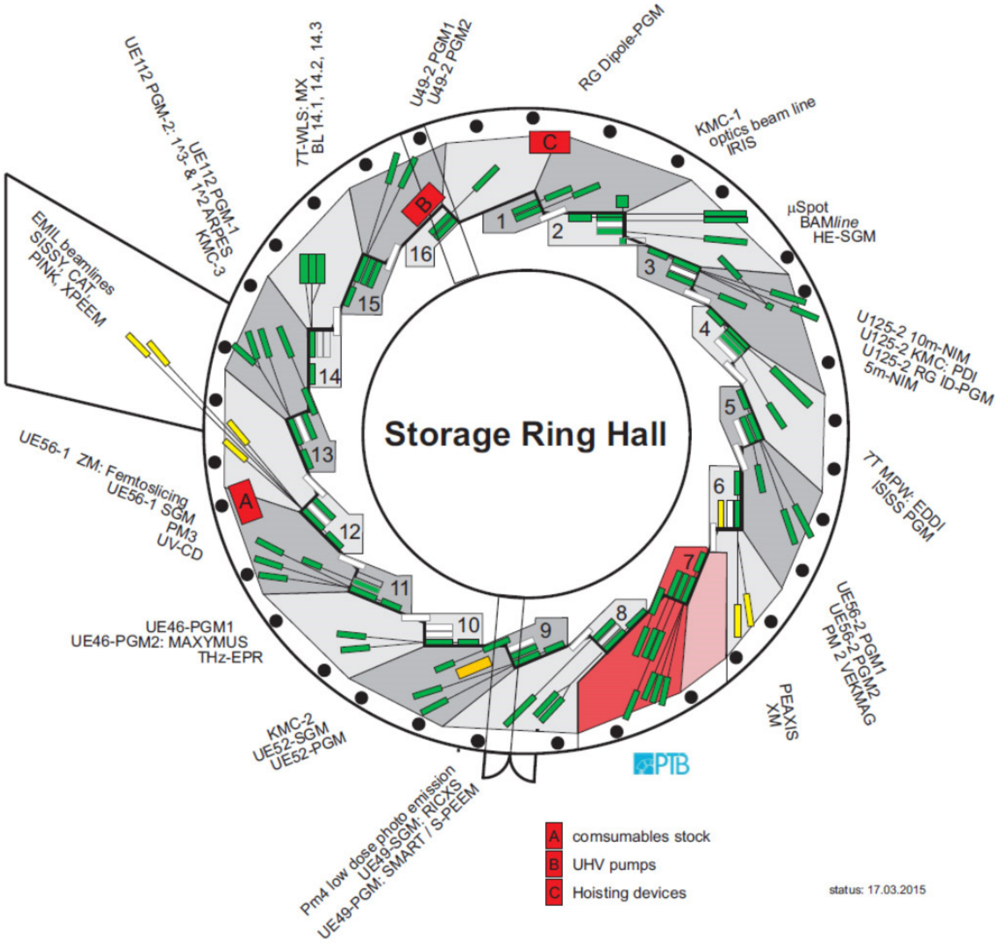
Will Smith, Helmholtz-Zentrum Berlin für Materialien und Energie GmbH *william.smith@helmholtz-berlin.de*

Content

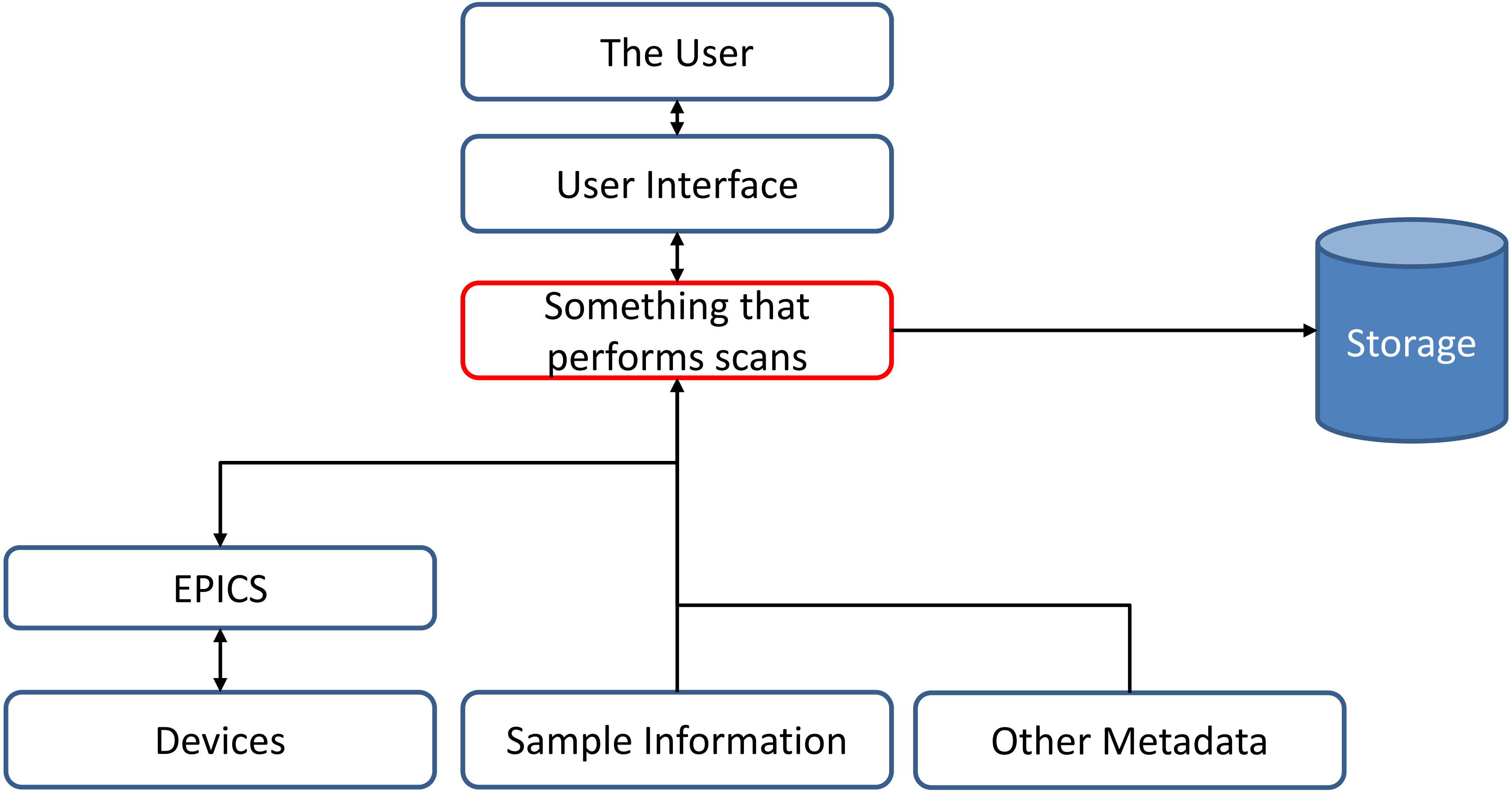
- **BESSY II**
- **Goals**
- **Benefits of Bluesky**
- **Deployment at BESSY II**
  - *Device Integration*
  - *Experimental Flow Control*
  - *Data collection*
  - *UI*
  - *Machine Learning Integration*
- **What’s Next?**

BESSY II

- Synchrotron focussing on soft X-Rays
- ~50 beamlines
- In operation for over 20 years
- Uses EPICS to control the accelerator and beamlines
- Huge variety of controls solutions at end stations, a lot use spec and LabView or home-grown



Requirements



The User

User Interface

Something that performs scans

Storage

EPICS

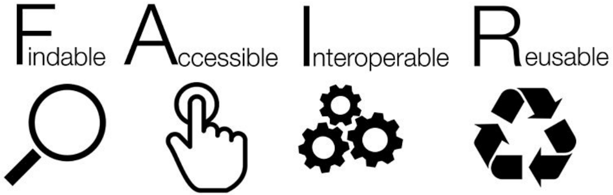
Devices

Sample Information

Other Metadata

Goals

- Use a modern, widely used programming languages
- Use something open source and popular
- Make data acquired at BESSY II Findable, Accessible, Interoperable and Reproducible (FAIR)
- Enable new areas of research in data science
- Produce intuitive tools help increase scientific output
- Write reliable, version controlled software that can be centrally supported



Why Bluesky

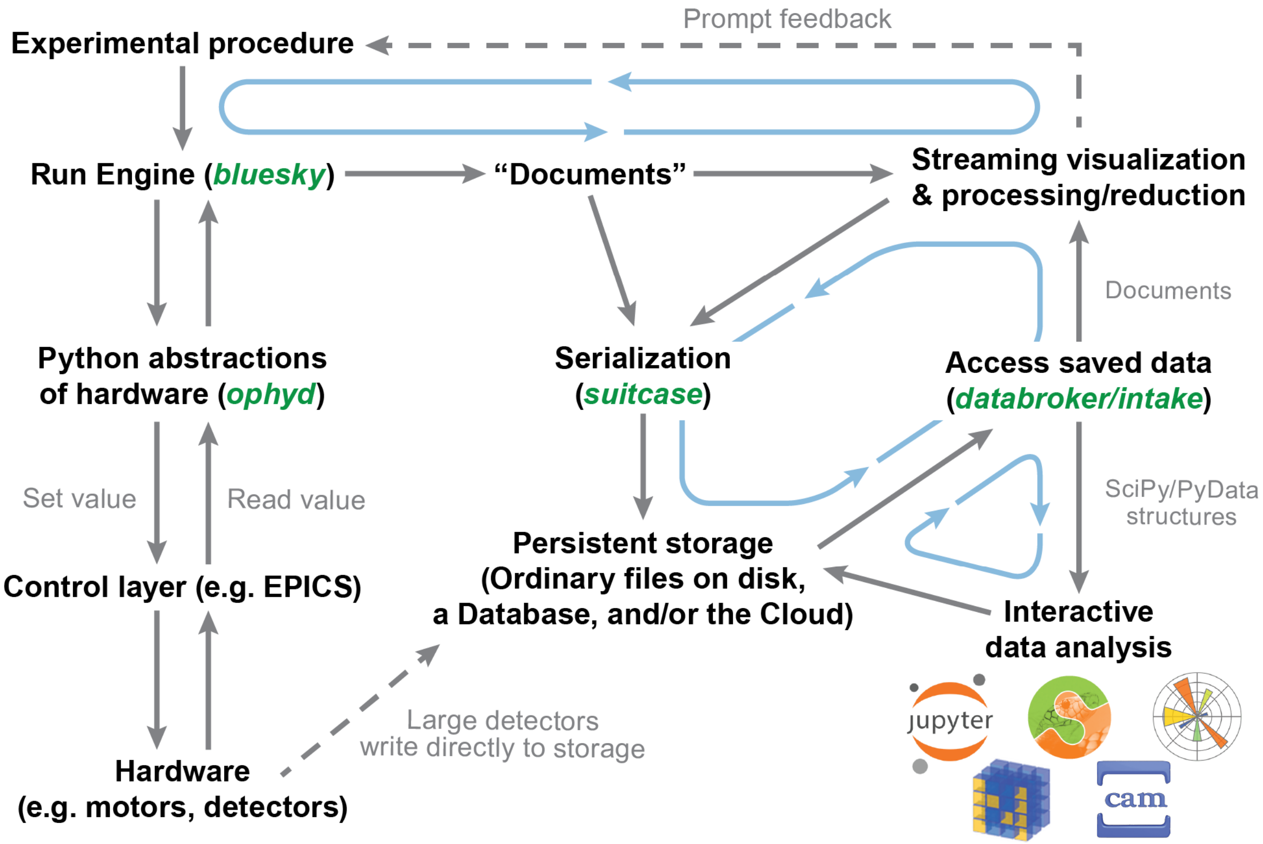
- Python3

- Open source, extremely responsive developer community.

- In use in the US at **NSLS II, APS, LCLS, SSRL, ALS** and a growing community in Europe (**BESSY II**, **MPG/FHI**,**Diamond**, **PSI/SLS** ...) and around the world (**CLS**, **ANSTO**,**PLS II**)

- Native support for EPICS with Ophyd

What is Bluesky?



Prompt feedback

Experimental procedure

Run Engine (*bluesky*)

"Documents"

Streaming visualization & processing/reduction

Documents

Python abstractions of hardware (*ophyd*)

Serialization (*suitcase*)

Access saved data (*databroker/intake*)

Set value

Read value

SciPy/PyData structures

Control layer (e.g. EPICS)

Persistent storage (Ordinary files on disk, a Database, and/or the Cloud)

Interactive data analysis

Large detectors write directly to storage

Hardware (e.g. motors, detectors)

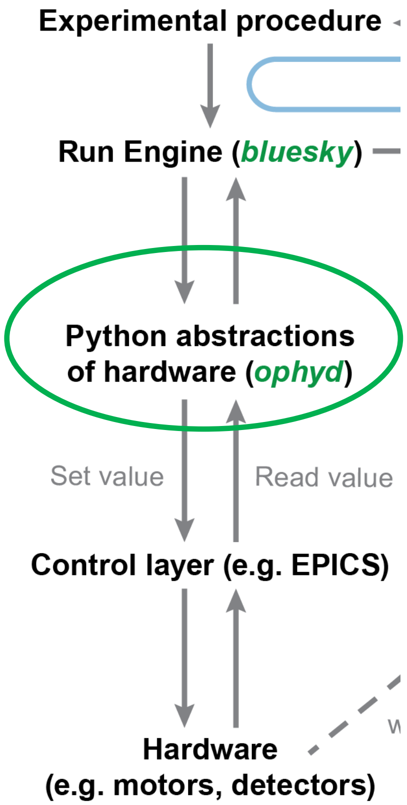
Ophyd Device Integration at BESSY II

Accelerator

- Many repeating segments
- Magnets power supplies, multiplexers, kickers, delay lines, Bunch-Bunch feedback tune values, Beam Position Monitors, Current Injection, Beam Size Monitors

Beamlines

- Variation between beamlines, but using common components and interfaces
- Shutters, Valves, Mirrors, Monochromators, Undulators, Bending Magnets, Apertures, Current Monitors, Cameras
- Devices classes can be reused between beamlines. Reduces workload to deploy
- EMIL in production use, in the process of deployment at AQUARIUS, KMC3, PM4, uSpot. More planned in the near future



Experimental procedure

Run Engine (*bluesky*)

Python abstractions of hardware (*ophyd*)

Set value

Read value

Control layer (e.g. EPICS)

Hardware (e.g. motors, detectors)

Shared Repositories

Aiming to share and reuse as much as possible between deployments.
Took inspiration from what we saw happening at labs in the US

bessyii_devices

- Contains Ophyd device classes
- Same class of device is instantiated at different beamlines

GitLab

bessyii

- Contains Bluesky plans or other scripts that can be reused

<beamline_name>tools

- Bluesky plans or setup scripts specific to a particular beamline

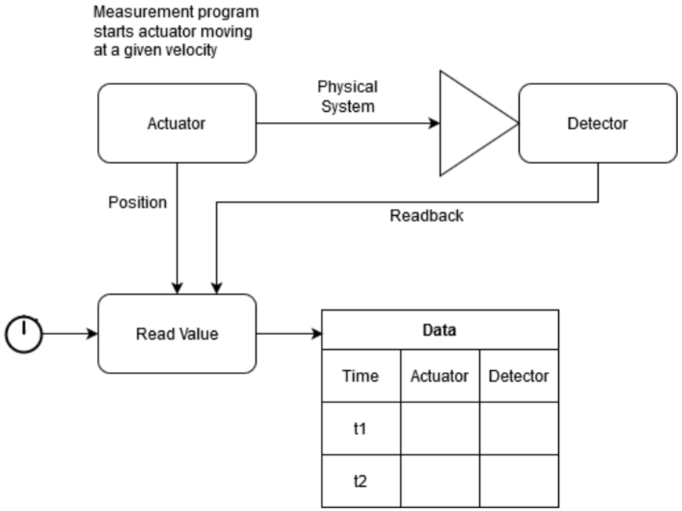
Continuous Energy Scans

- We created a bluesky plan that starts a motor (monochromator), samples the readback and any detectors at fixed intervals.

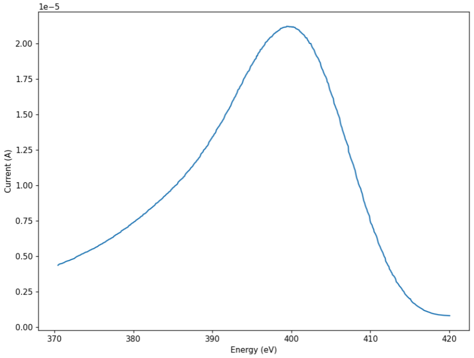
- Assumes motor or mono moves slowly enough that error caused by polling the detectors sequentially is insignificant.

- Could be improved with hardware trigger system

```
RE(flyscan([det],motor,370,470,0.1))
```


Measurement program starts actuator moving at a given velocity
Actuator
Physical System
Detector
Position
Readback
Read Value

Data		
Time	Actuator	Detector
t1		
t2		



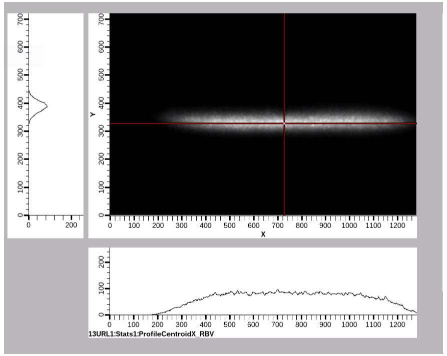
Working with EPICS areaDetector

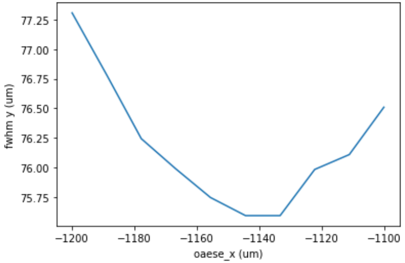
- We want to get parameters from cameras. Position, FWHM etc

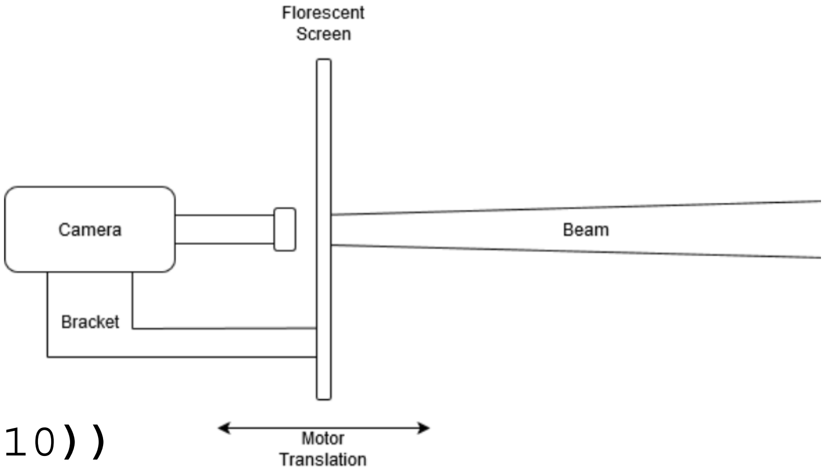
- Cameras and Detectors can be given a standard interface with EPICS areaDetector

- Ophyd already provides an interface to that

- Analysed image from areaDetector to find focus of the end station by scanning position just using the built in *scan* plan.

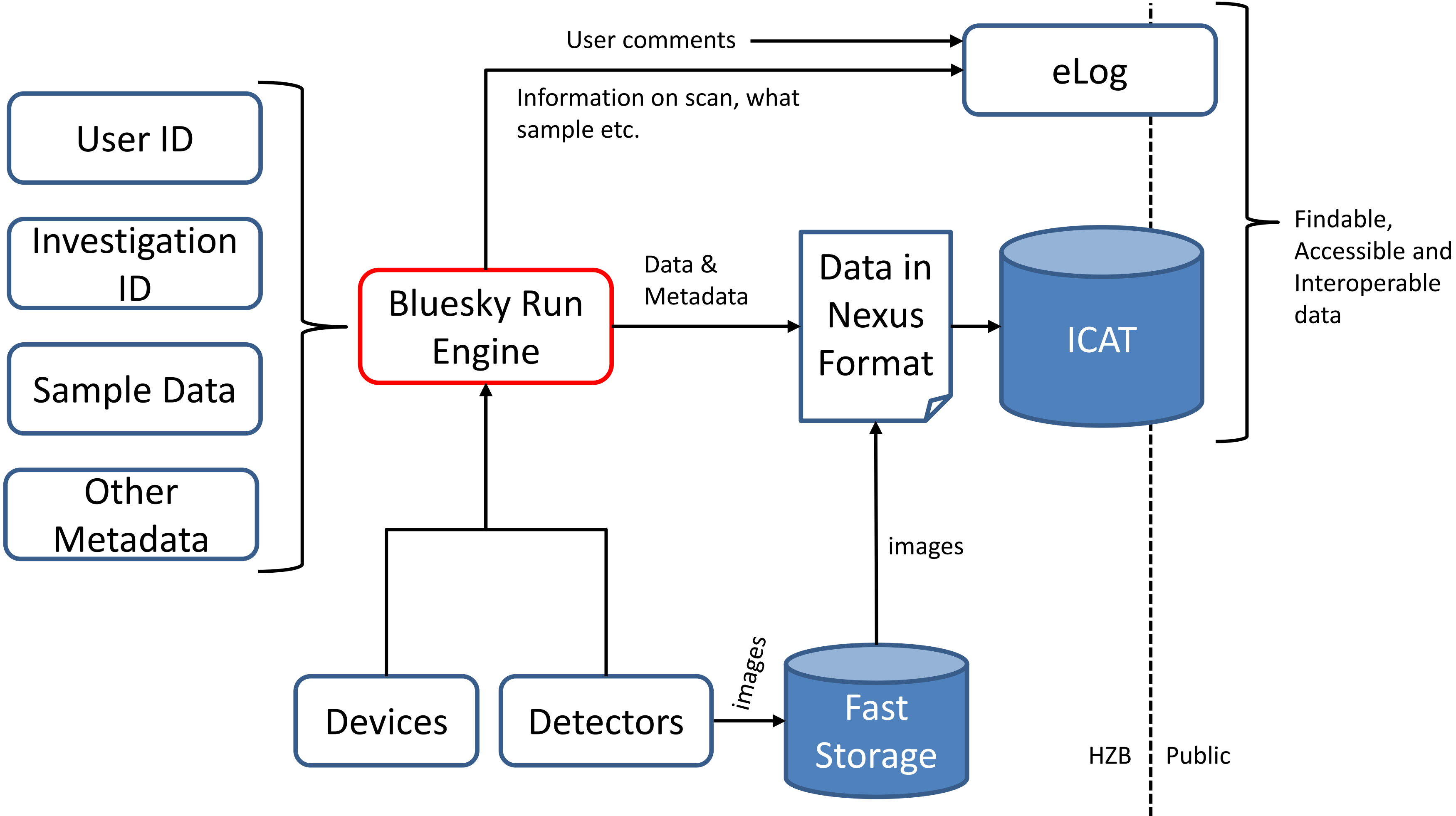






```
RE(scan([camera],motor,-1200,-1100,10))
```

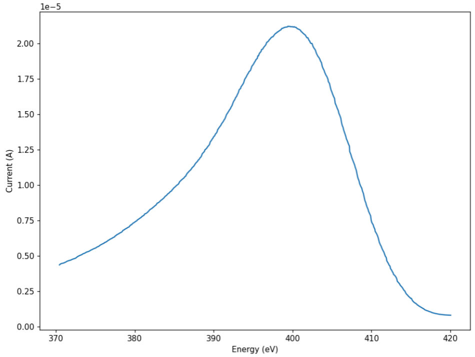
FAIR Case Study at EMIL (WEBL05)



User Interface

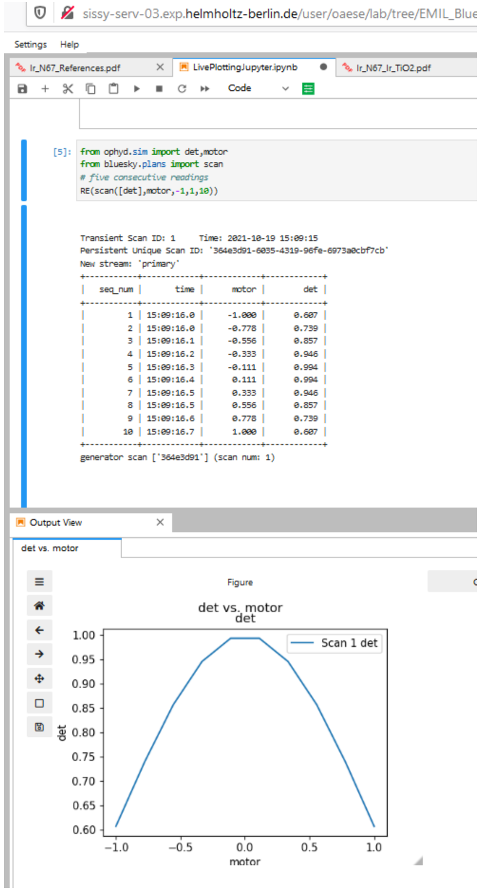
- **Ipython**
  - Closest to spec
  - Tab completion
  - Produces live visualisations

```
RE(scan([det],motor,-1,1,10))
```



User Interface

- **Jupyter Notebook**
  - Web interface
  - Allows you to save and share your notebook
  - Nice for working on data and trying things out
  - Brilliant for making tutorials
  - Can produce live visualisations

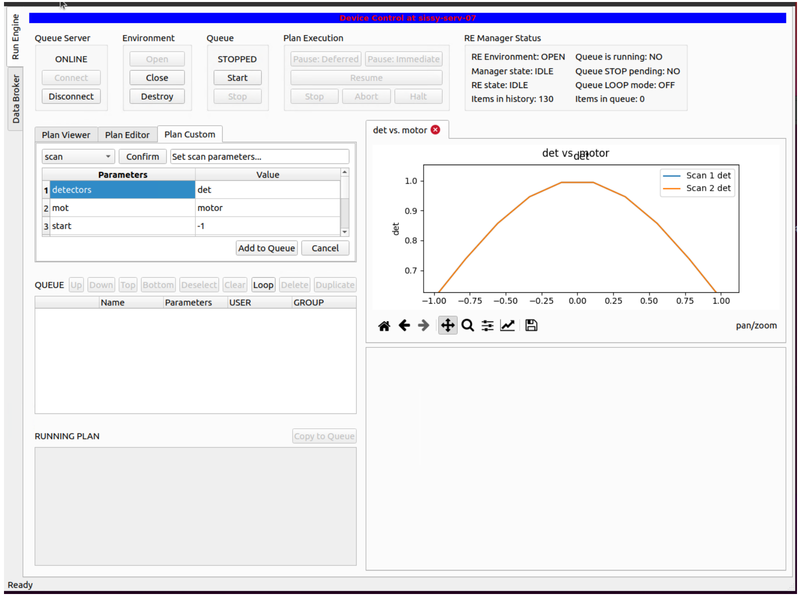


```
[5]: from ophyd.sim import det,motor
     from bluesky.plans import scan
     # five consecutive readings
     RE(scan([det],motor,-1,1,10))
```

```
Transient Scan ID: 1     Time: 2021-10-19 15:09:15
Persistent Unique Scan ID: '364e3d91-6035-4319-96fe-6973a0cbf7cb'
New stream: 'primary'
+-----------+------------+------------+------------+
|   seq_num |       time |      motor |        det |
+-----------+------------+------------+------------+
|         1 | 15:09:16.0 |     -1.000 |      0.607 |
|         2 | 15:09:16.0 |     -0.778 |      0.739 |
|         3 | 15:09:16.1 |     -0.556 |      0.857 |
|         4 | 15:09:16.2 |     -0.333 |      0.946 |
|         5 | 15:09:16.3 |     -0.111 |      0.994 |
|         6 | 15:09:16.4 |      0.111 |      0.994 |
|         7 | 15:09:16.5 |      0.333 |      0.946 |
|         8 | 15:09:16.5 |      0.556 |      0.857 |
|         9 | 15:09:16.6 |      0.778 |      0.739 |
|        10 | 15:09:16.7 |      1.000 |      0.607 |
+-----------+------------+------------+------------+
generator scan ['364e3d91'] (scan num: 1)
```

User Interface

- **QT GUI**
  - Nicest for simple use cases
  - Minimal training needed
  - Browsing the database is easy
  - Performs best if it’s decoupled from the run engine

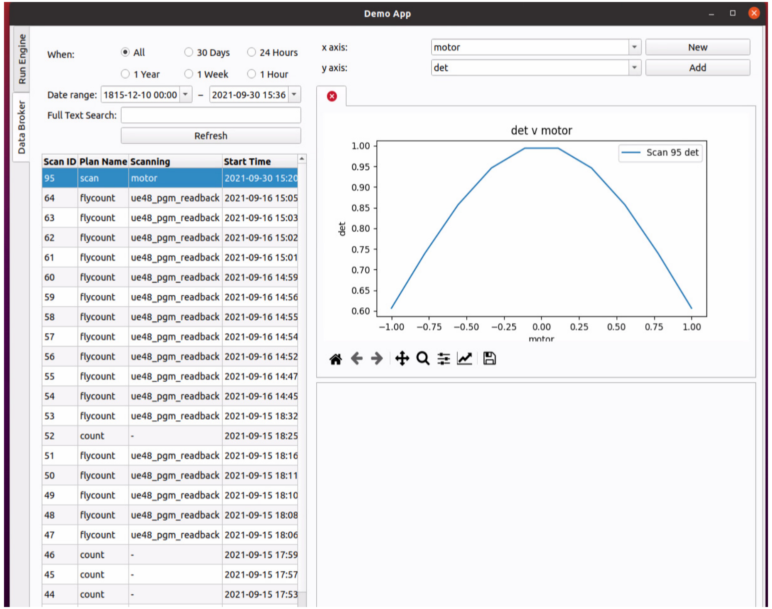


RE Environment: OPEN
Manager state: IDLE
RE state: IDLE
Items in history: 130
Queue is running: NO
Queue STOP pending: NO
Queue LOOP mode: OFF
Items in queue: 0

	Parameters	Value
1	detectors	det
2	mot	motor
3	start	-1

User Interface

- **QT GUI**
  - Nicest for simple use cases
  - Minimal training needed
  - Browsing the database is easy
  - Performs best if it’s decoupled from the run engine



When: ● All ○ 30 Days ○ 24 Hours ○ 1 Year ○ 1 Week ○ 1 Hour

Date range: 1815-12-10 00:00 – 2021-09-30 15:36

Full Text Search:

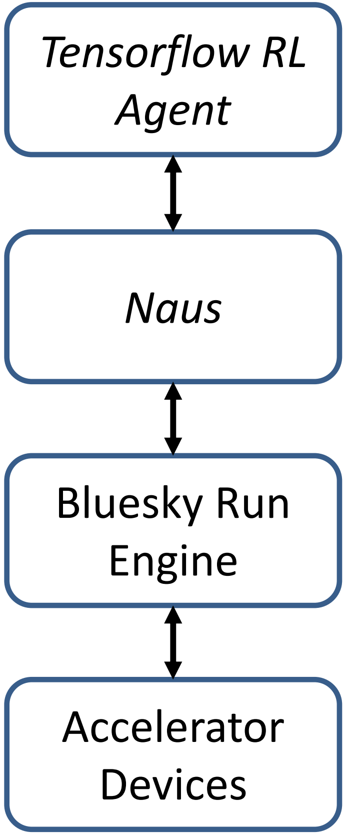
x axis: motor

y axis: det

Scan ID	Plan Name	Scanning	Start Time
95	scan	motor	2021-09-30 15:20
64	flycount	ue48_pgm_readback	2021-09-16 15:05
63	flycount	ue48_pgm_readback	2021-09-16 15:03
62	flycount	ue48_pgm_readback	2021-09-16 15:02
61	flycount	ue48_pgm_readback	2021-09-16 15:01
60	flycount	ue48_pgm_readback	2021-09-16 14:59
59	flycount	ue48_pgm_readback	2021-09-16 14:56
58	flycount	ue48_pgm_readback	2021-09-16 14:55
57	flycount	ue48_pgm_readback	2021-09-16 14:54
56	flycount	ue48_pgm_readback	2021-09-16 14:52
55	flycount	ue48_pgm_readback	2021-09-16 14:47
54	flycount	ue48_pgm_readback	2021-09-16 14:45
53	flycount	ue48_pgm_readback	2021-09-15 18:32
52	count	-	2021-09-15 18:25
51	flycount	ue48_pgm_readback	2021-09-15 18:16
50	flycount	ue48_pgm_readback	2021-09-15 18:11
49	flycount	ue48_pgm_readback	2021-09-15 18:10
48	flycount	ue48_pgm_readback	2021-09-15 18:08
47	flycount	ue48_pgm_readback	2021-09-15 18:06
46	count	-	2021-09-15 17:59
45	count	-	2021-09-15 17:57
44	count	-	2021-09-15 17:53

ML Integration Studies (THAL01)

- **Reducing Harmonic Orbit Perturbations**
  - Tensorflow RL agent communicates with RE over ZMQ using and in house package called *Naus*
  - Unfortunately was too slow for production use.
  - Following developments of Bluesky-Queueserver and Bluesky-Adaptive hoping to build on this.
  - See paper at this conference: THAL01 by Luis Vera Ramirez and Pierre Schnizer


Tensorflow RL Agent

Naus

Bluesky Run Engine

Accelerator Devices

What‘s Next?

- Integrate more devices for the accelerator and end stations
- Roll out to other beamlines
- Deploy first working version of GUI
- Continue working on making data FAIR

Conclusion

- Considerable progress in the last year at BESSY II deploying Bluesky for the accelerator and Beamlines
- Joining a growing and active community of users and developers
- Set of powerful tools that we can customise for our users specific needs
- Helping us realise our FAIR data goals

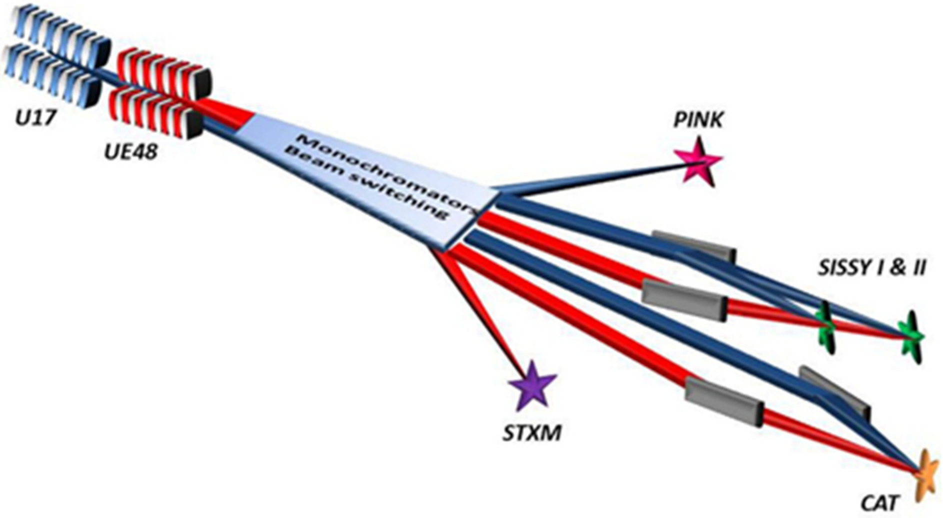
Thanks

- **Colleagues at BESSY II**
  - Huiling He
  - Luis Vera Ramirez
  - Pierre Schnizer
  - Rolland Müller
  - Sebastian Kazarski
  - Sebastian Sachse
  - Simone Vadilonga
- **The developers of the Bluesky packages!**

https://blueskyproject.io/

Deployment at EMIL Beamlines

- Energy Materials In-Situ Laboratory
- 5 Endstations
- Hard and Soft Beam
- 100’s of devices and components (although lots of repeats)
- Beamline uses EPICS
- SISSY and PINK use EPICS, CAT uses SpecsLab
- SISSY Lab has UHV sample transfer system, various instruments



Options

SPEC

- Used by most beamlines at BESSY II, and many around the world.
- Simple command line interface
- Flexible and very well tested
- It’s own language
- Licenced and not free
- Single developer
- No error checking or minimal tabbed completion

Sardana

- Based on python, open source
- Used by Alba, PETRA III, ESRF, ELI NP, MAX IV, SOLARIS .
- Designed around Tango, could use EPICS but nobody is doing this => no community support

Playing with pytest

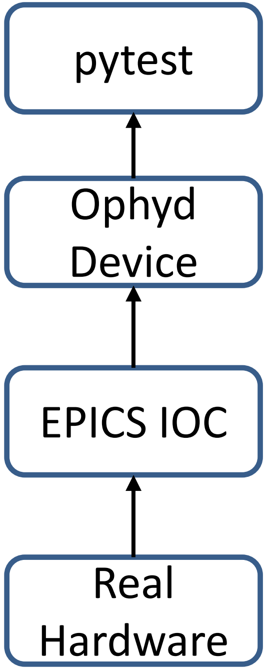
- Ophyd provides python device abstraction

- We can write unit tests using the package pytest and use this to test IOC's and hardware does what we expect

- We have tests to check:
  - Devices exist
  - Devices are connected
  - Motors and valves move when asked
  - Devices operate as expected (when I set a value and read it back is it correct?)

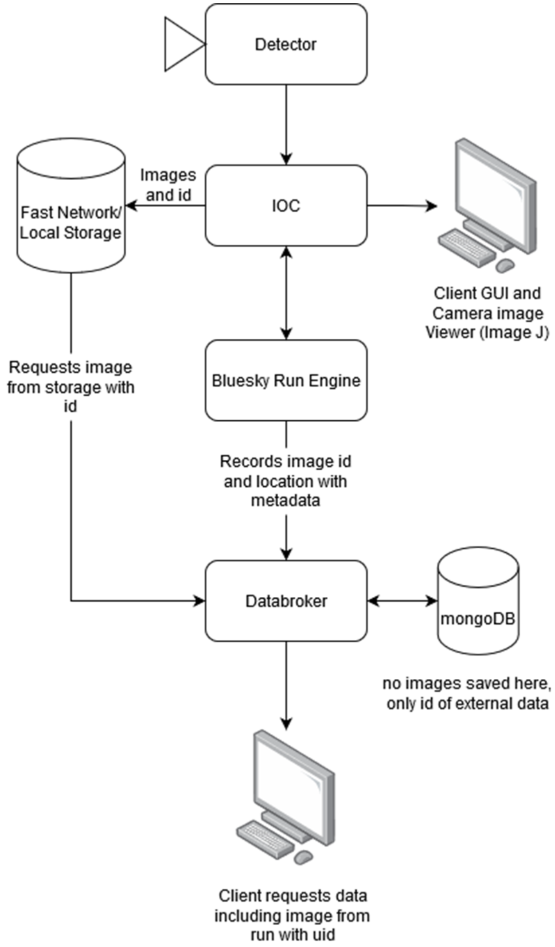
- Challenging to implement in practice because we need hardware to test on and don't have a spare beamline!

- Has proved useful, and found bugs we wouldn't have otherwise



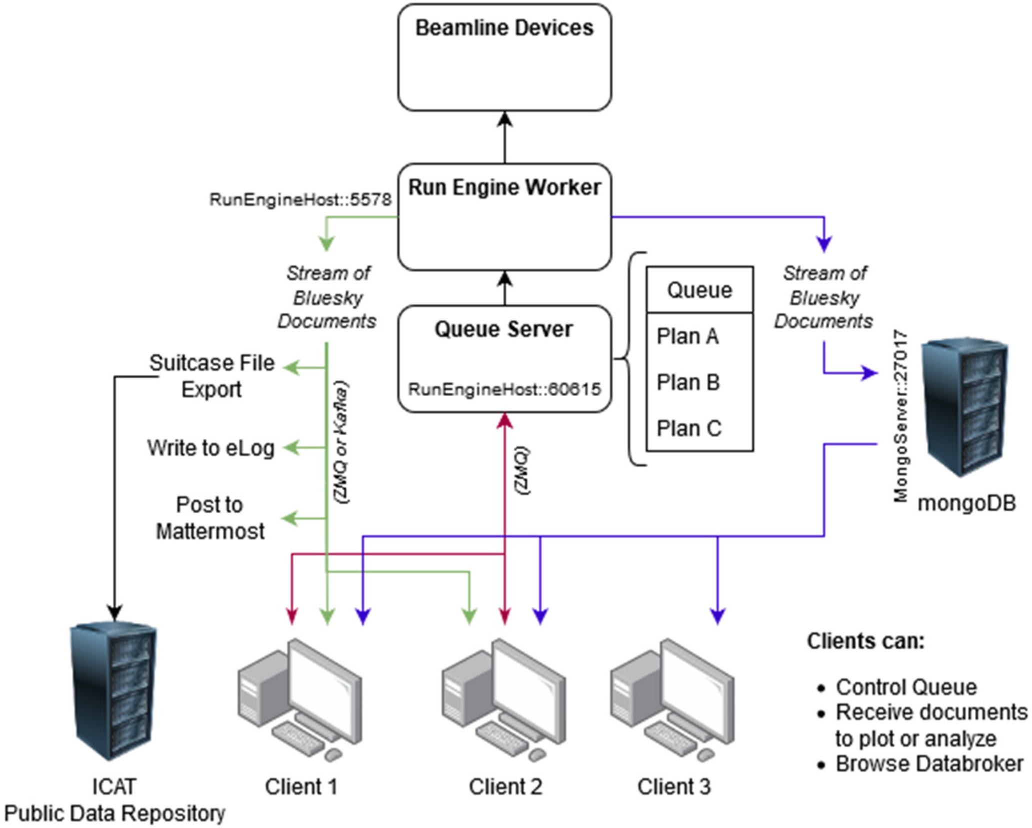
Working with EPICS areaDetector

- Images are acquired by the IOC under the direction of Bluesky
- Images are saved locally with a name that is saved in the mongodb database
- When the user accesses the run in the database the databroker fetches the image from the local storage
- Works with all devices that have EPICS areaDetector Support
- Reduces requirements on network and storage.



Detector
IOC
Images and id
Fast Network/ Local Storage
Client GUI and Camera image Viewer (Image J)
Requests image from storage with id
Bluesky Run Engine
Records image id and location with metadata
Databroker
mongoDB
no images saved here, only id of external data
Client requests data including image from run with uid

Run Engine as a Service



Clients can:

- Control Queue
- Receive documents to plot or analyze
- Browse Databroker