

Toward a Unified Data Acquisition Framework for Beamline Instrumentation Using Bluesky and NATS Jetstream

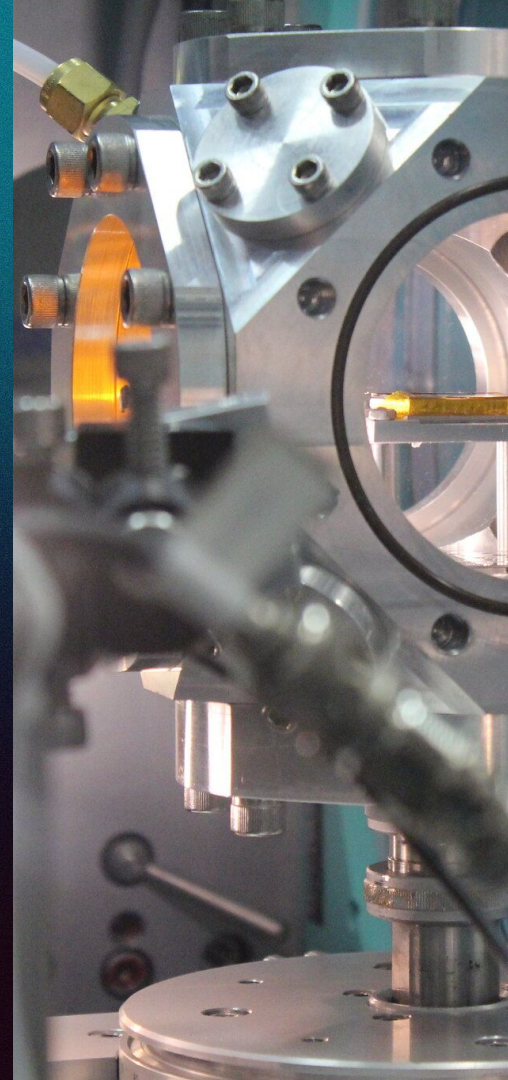


2025-11-06
ROCK-IT Bluesky Workshop, DESY, Germany

Russ Berg, Connor Boyle, Darren Hunter, [Dr. Niko Kivel](#), David Pastl, Jay Van Doornum
Canadian Light Source Inc.

Outline

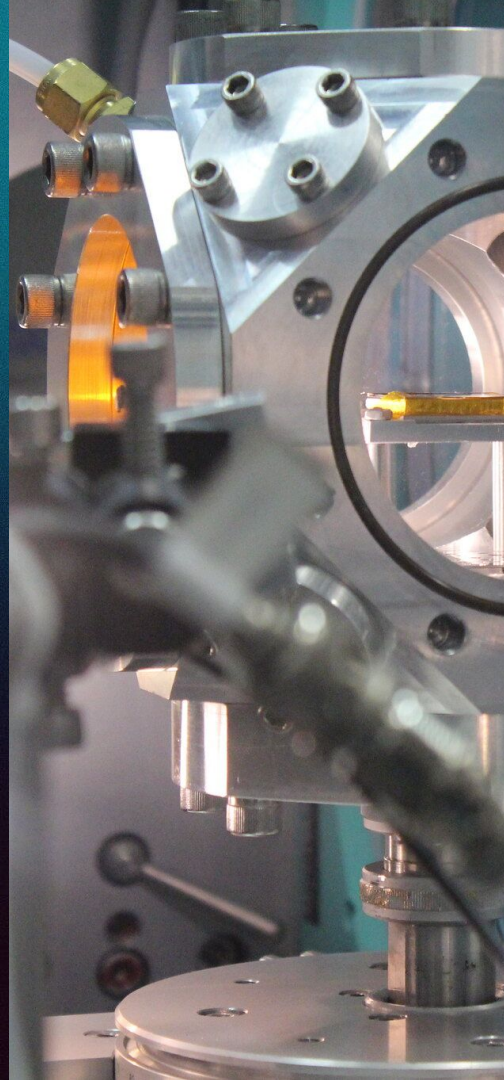
- Introduction
 - History
 - System layout
- Milestones
 - Hardware triggered scan
- Challenges
 - pixi
- Outlook





DAQ History at CLS

- Seven DAQ tools in use
 - External tool (Spec)
 - Inhouse tools (PyAcq, Acquaman, ...)
- High technical dept
- Unsupported or tools with questionable support
- Outdated code bases with ancient libs
- To fix the problem, we're going to introduce yet another tool ...

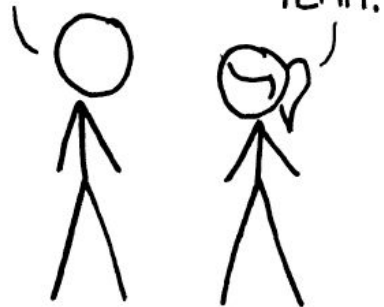


DAQ History at CLS

HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC.)

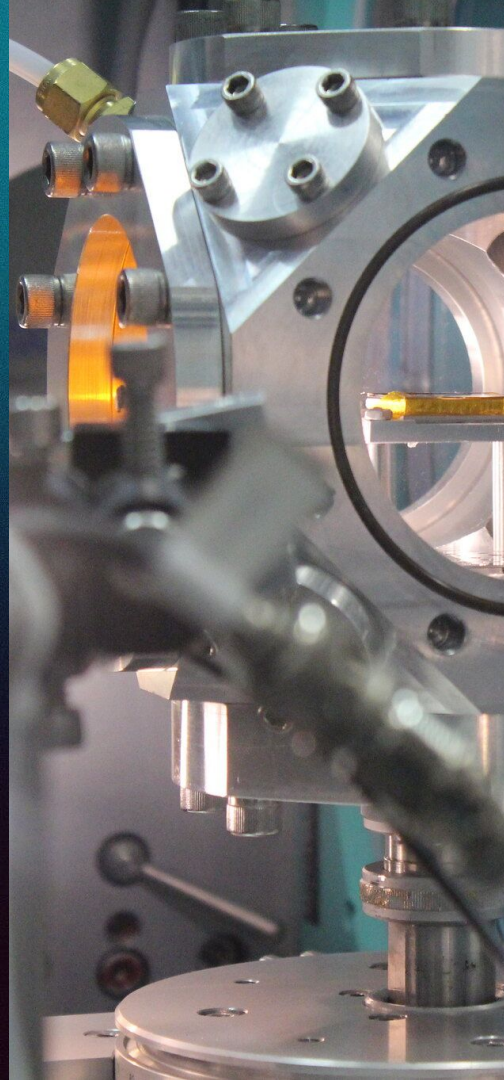
SITUATION:
THERE ARE
14 COMPETING
STANDARDS.

14?! RIDICULOUS!
WE NEED TO DEVELOP
ONE UNIVERSAL STANDARD
THAT COVERS EVERYONE'S
USE CASES.



SOON:

SITUATION:
THERE ARE
15 COMPETING
STANDARDS.



System layout

- Queueserver as systemd service (pixi, Ansible)
- Beamline profiles as artifacts (deb, CI/CD)
 - Users can inject custom code via “overlay”
- Client application in separate pixi-environment
 - CLI: Qserver-console ✓
 - GUI: QMonitor / cls_bs_monitor ✓
 - Web: JupyterLab 🚧
- Devices:
 - Exclusively ophyd-async

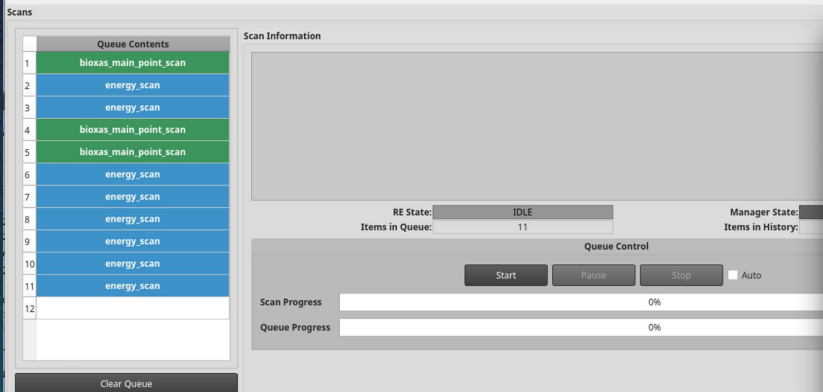
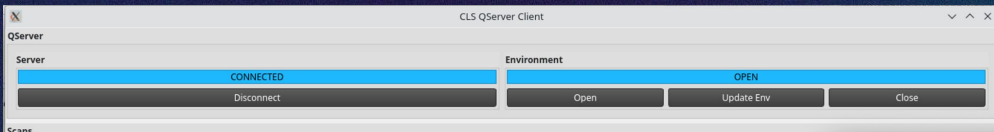


System layout

- RE metadata is enriched from kv-store
 - Populated via **nimbus**
 - Project/Session info from UserPortal (API)
- Custom Providers
 - SessionPathProvider → file storage path
 - `/<root>/<PRJ>/<BL>/<Session>`
 - SubjectProvider → NATS subject
 - `events.<BL>.<PRJ>.{start, event, ..., stop}`
- Session validation via **Preprocessor**

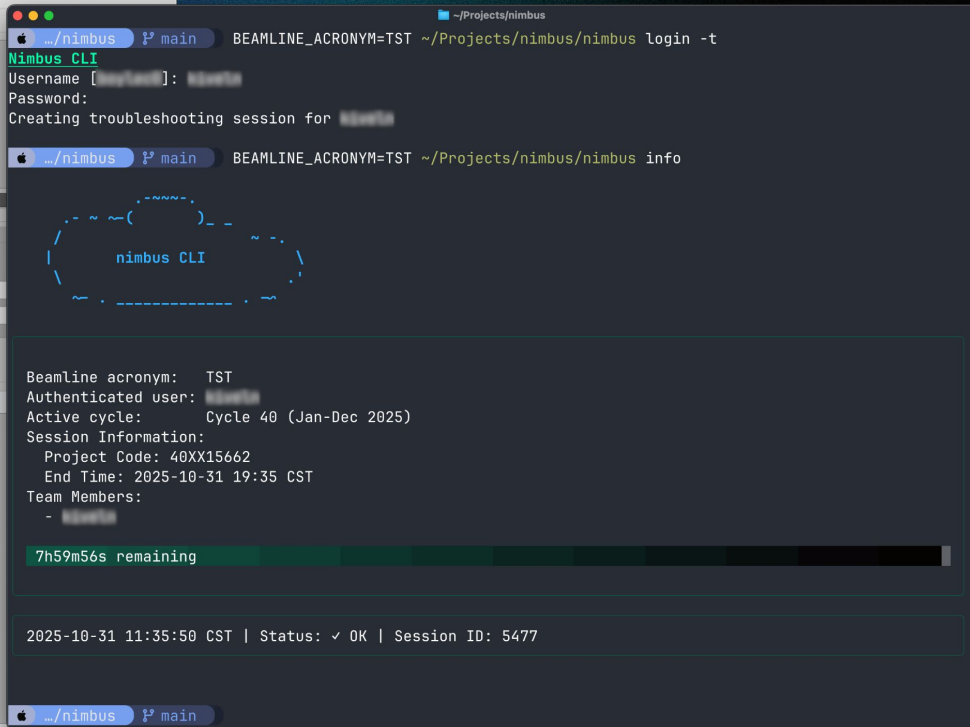


Tools, new and improved



QueueServer Console

```
[I 2025-10-14 14:13:46,393 bluesky_queueserver.manager.worker] Generating lists of allowed plans and devices
[I 2025-10-14 14:13:46,395 bluesky_queueserver.manager.worker] List of allowed plans and devices was successfully generated
[I 2025-10-14 14:13:46,418 bluesky_queueserver.manager.worker] Instantiating and configuring Run Engine ...
[I 2025-10-14 14:13:46,418 bluesky_queueserver.manager.worker] RE Environment is ready
[I 2025-10-14 14:13:46,418 bluesky_queueserver.manager.worker] Preparing to start IPython kernel ...
[I 2025-10-14 14:13:46,426 bluesky_queueserver.manager.worker] IPython kernel is in 'idle' state
[I 2025-10-14 14:13:46,688 bluesky_queueserver.manager.worker] Downloading the lists of existing plans and devices from the worker environment
[I 2025-10-14 14:13:46,689 bluesky_queueserver.manager.worker] Worker started successfully.
[E 2025-10-18 19:17:07,969 bluesky_queueserver.manager.worker] Unexpected message received: {'jsonrpc': '2.0',
'id': 'efe8fd89-15b8-4e89-9898-4d479d967442',
'result': {'running_item_uid': None,
'running_plan_completed': True,
're_report_available': False,
're_state': 'idle',
're_deferred_pause_requested': False,
'environment_state': 'idle',
'run_list_updated': False,
'plans_and_devices_list_updated': False,
'running_task_uid': None,
'completed_tasks_available': False,
'background_tasks_num': 0,
'ip_kernel_state': 'idle',
'ip_kernel_captured': False,
'unexpected_shutdown': False}}. The message is ignored
```



System layout

```
RE.subscribe(TiledWriter)
```

System layout

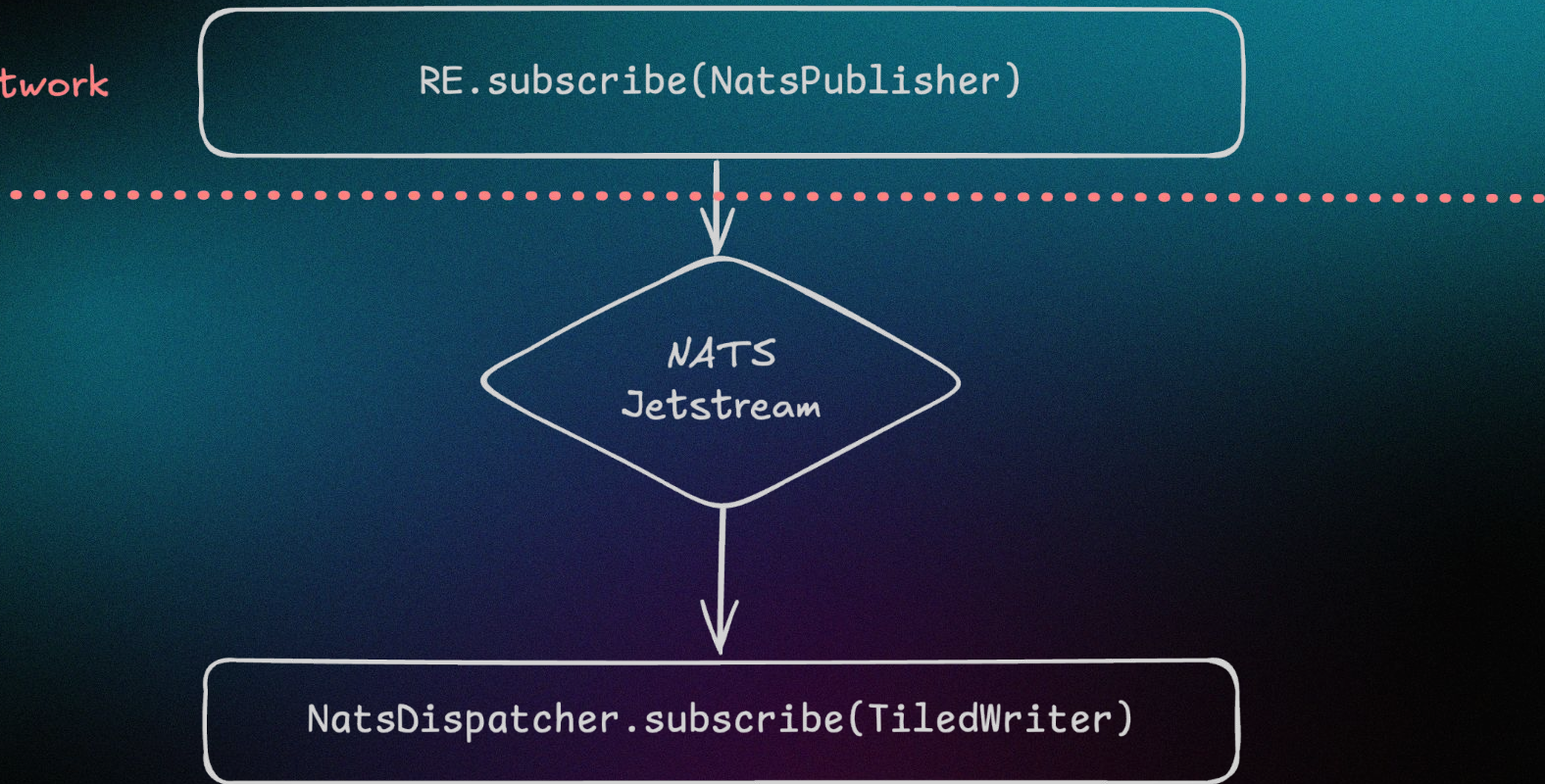
Beamline network

```
RE.subscribe(NatsPublisher)
```

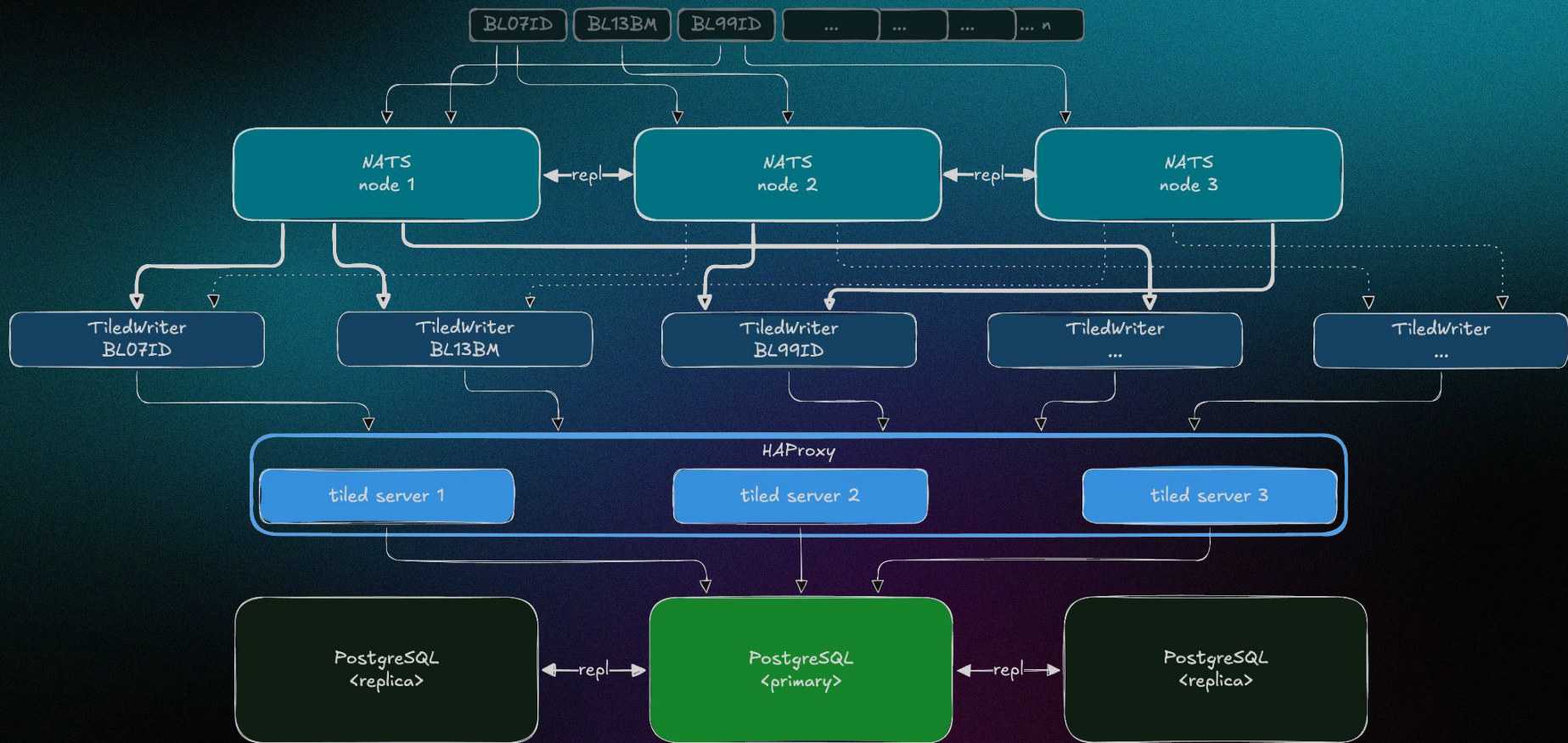
server land

NATS
Jetstream

```
NatsDispatcher.subscribe(TiledWriter)
```

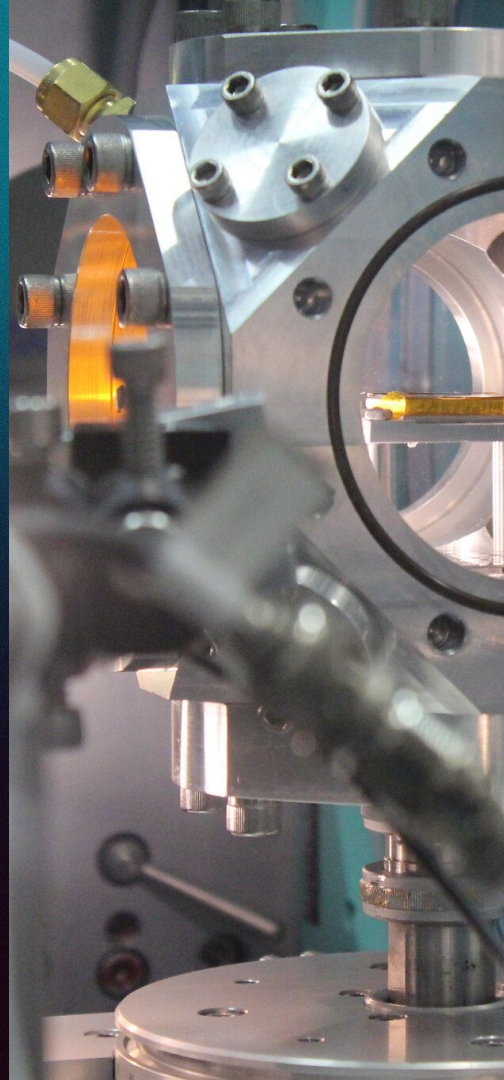


System layout (WIP)



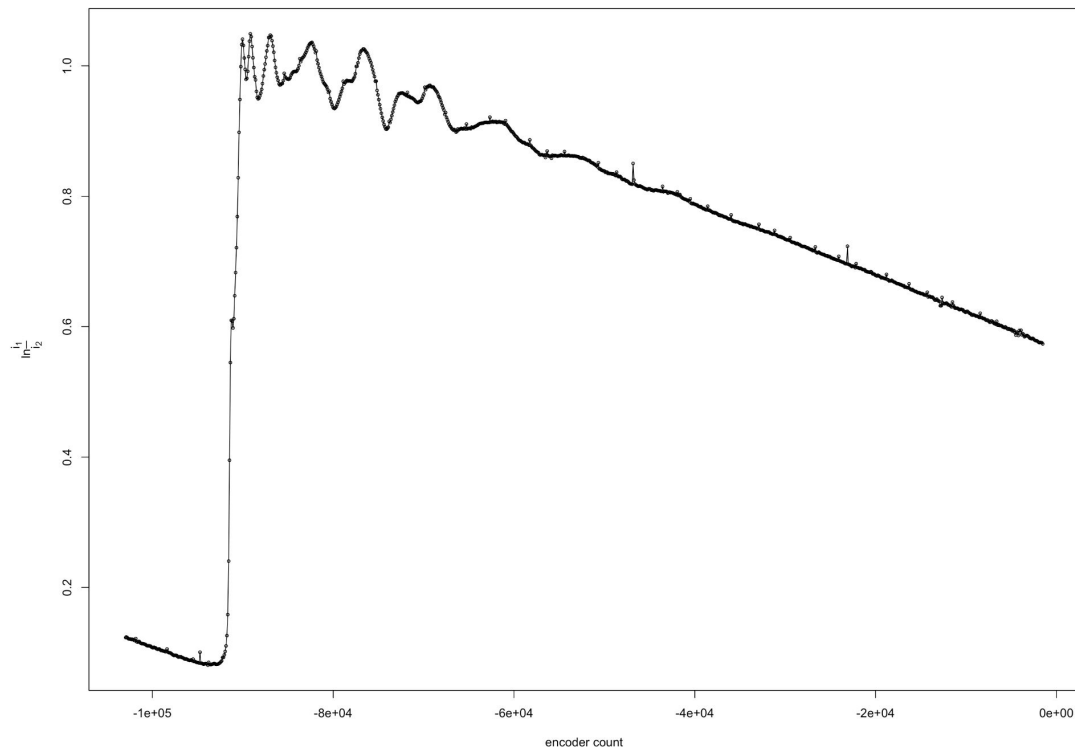
Why a NATS Jetstream at all?

- RE.subscribe(TiledWriter)
 - Safety net ❌
 - Messages consumed ❌
 - Community support ✅
- Jetstream in the middle
 - Messages preserved ✅
 - Recovery via re-streaming ✅
 - Scalable and consumer agnostic ✅
 - Flexible, create additional messages 🚧
 - Community support ❌



Milestone: Bluesky in action

XAS, Cu-foil, BioXAS-Main



Beamline: BioXAS-Main

Mode: fly-scan, hardware trigger

Number of points: ~1000

Time: ~30 s

Captured traces:

- Three ion chamber voltages
- Encoder signal

Trigger source:

- Encoder signal

Challenges | **pixi**: A love-hate story

- Deterministic environments
... if you get it to work at all
- CI friendly ... sometimes
- Conda-forge: no wheels, no trouble?
- One manifest, multiple environments
... that's what is says on the box, works mostly
- Multiple platforms
... **aioca** says: “You shall not pass! ... on aarch64”
- Still better than **pip**? ... sure



Outlook

- Fully NATS Jetstream backed data flow
- WebSocket streaming with NATS kv backend
- Replace **Redis** for metadata caching with NATS kv.
`redis-json-dict` → `kv-json-dict`
- Deploy bluesky to more beamlines

