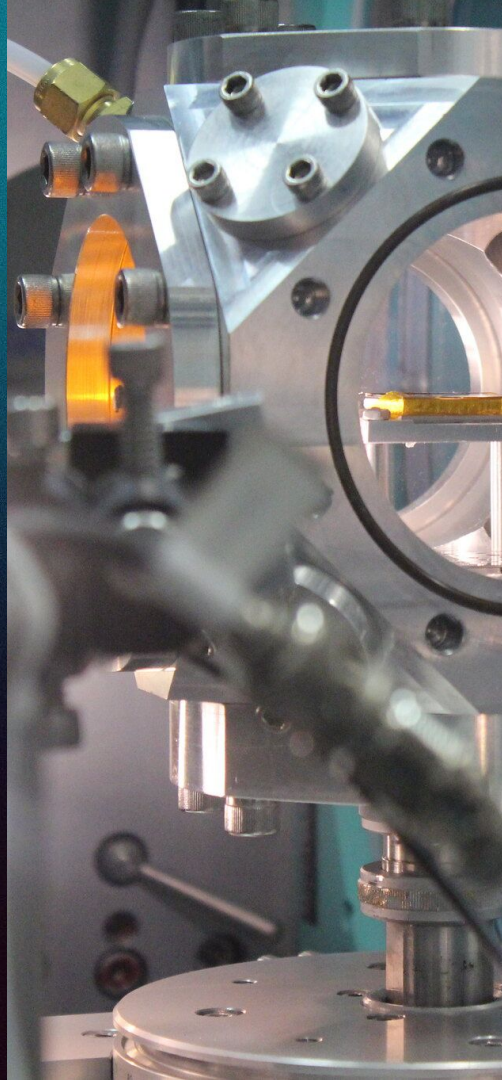
Toward a Unified Data Acquisition Framework for Beamline Instrumentation Using Bluesky and NATS Jetstream

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

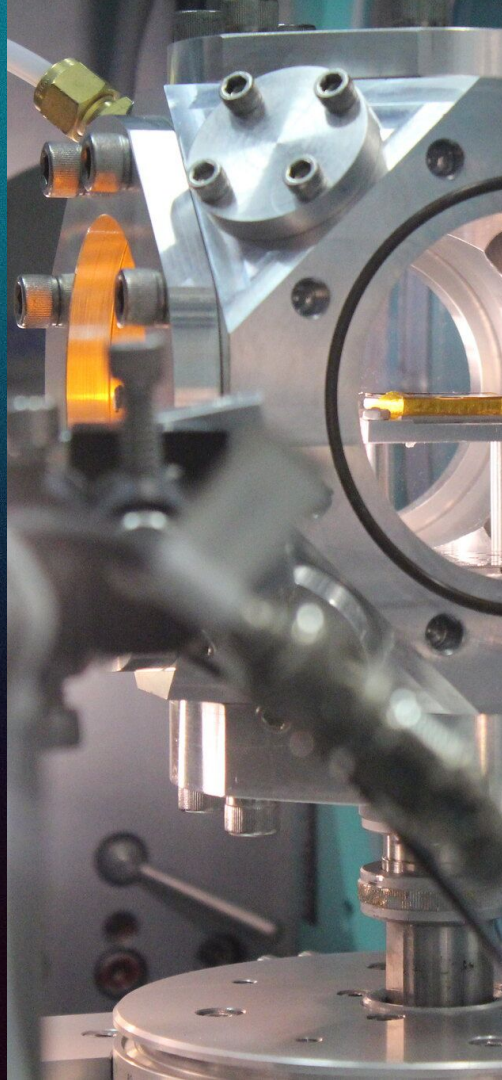
- Introduction
  - History
  - System layout
- Milestones
  - Hardware triggered scan
- Challenges
  - pixi
- Outlook



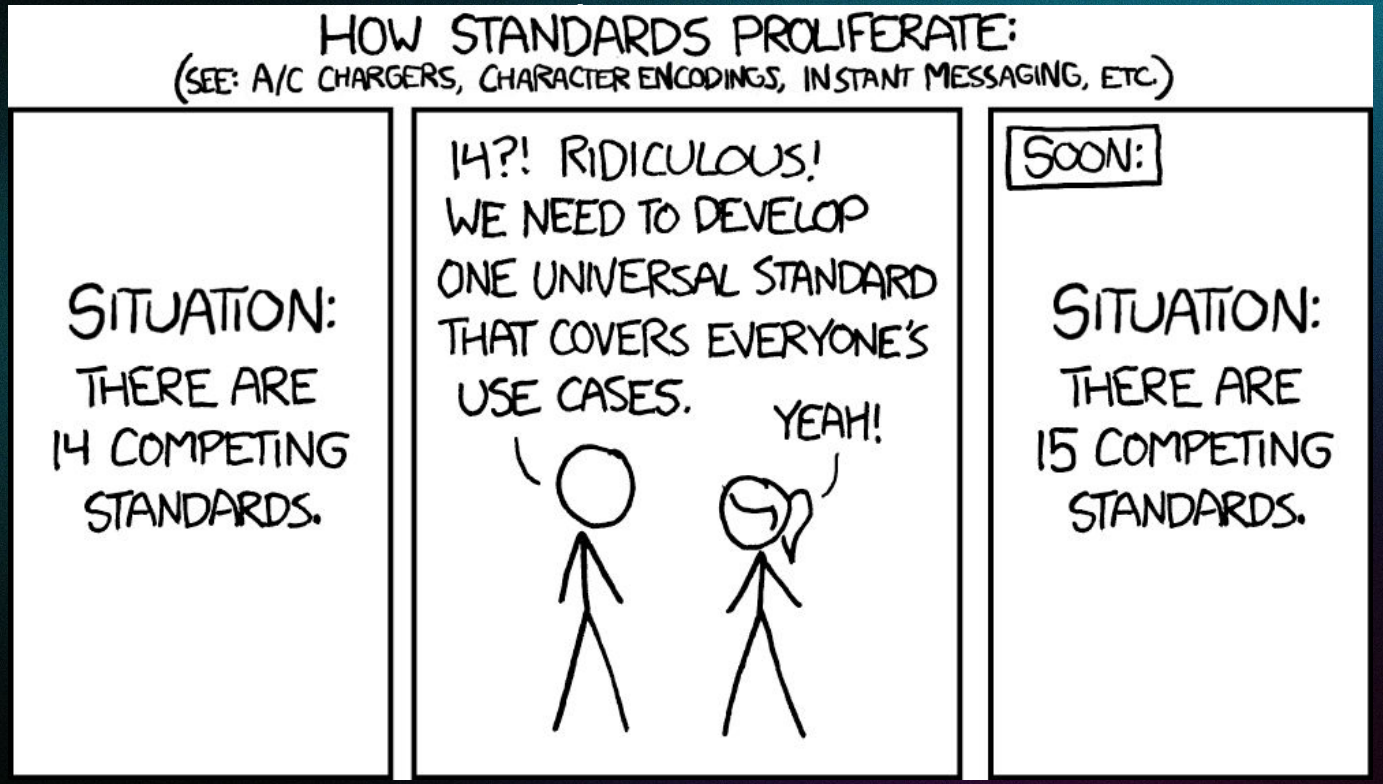

Northwestern Passages
NUNAVUT
NORTHWEST TERRITORIES
YUKON
Hudson Bay
Canada
ALBERTA
MANITOBA
BRITISH COLUMBIA
SASKATCHEWAN
Edmonton
NEWFOUNDLAND AND LABRADOR
ONTARIO
QUEBEC
Banff
Calgary
Whistler
Vancouver
Victoria
Winnipeg
NORTH DAKOTA
WASHINGTON
MONTANA
MINNESOTA
Québec City
NB
PE
Portland
Montreal
SOUTH DAKOTA
WISCONSIN
MAINE
NOVA SCOTIA
VT
OREGON
IDAHO
WYOMING
MICHIGAN
Toronto
NH
NEW YORK
MA
Boston
IOWA
Chicago
NEBRASKA
CT
RI

DAQ History at CLS

- Seven DAQ tools inuse
  - 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- Outdated code bases with ancient libs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



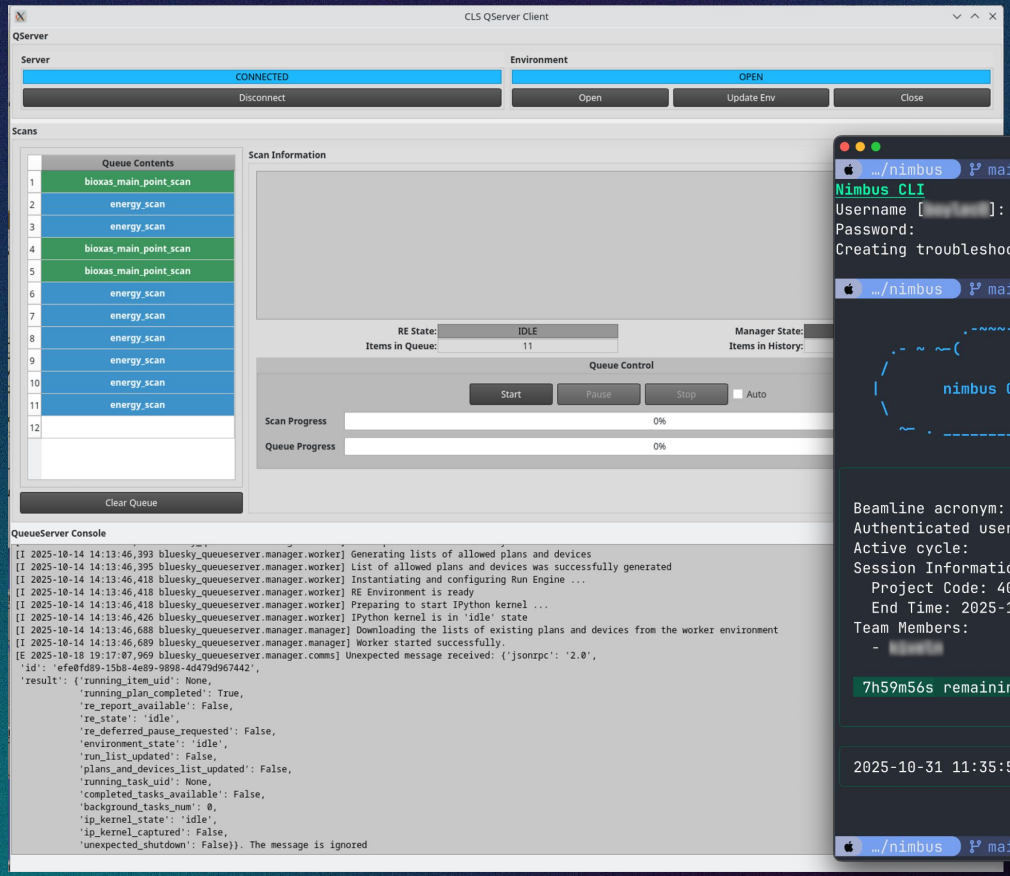
System layout

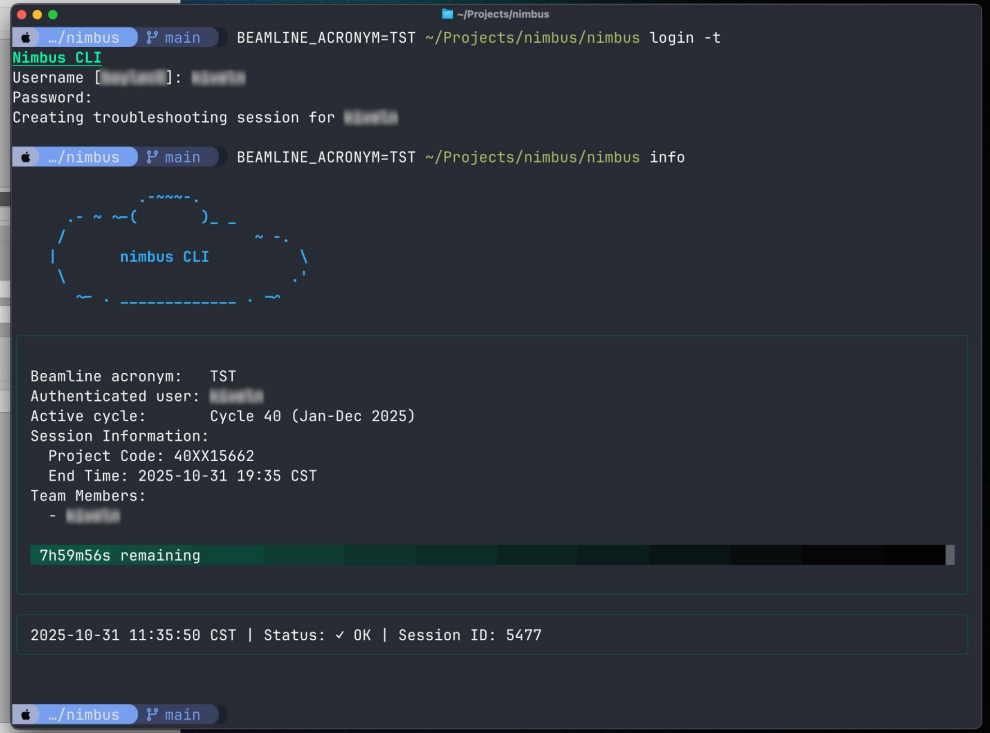
- Queueserver as systemd service (pixi, Ansible)
- Beamline profiles as artifacts (deb, CI/CD)
  - Users can inject custom code via “overlay”
- Client application in separate pixi-environment
  - CLI: Qserver-console ✅
  - GUI: QMonitor / cls_bs_monitor ✅
  - Web: JupyterLab 🚧
- Devices:
  - Exclusively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

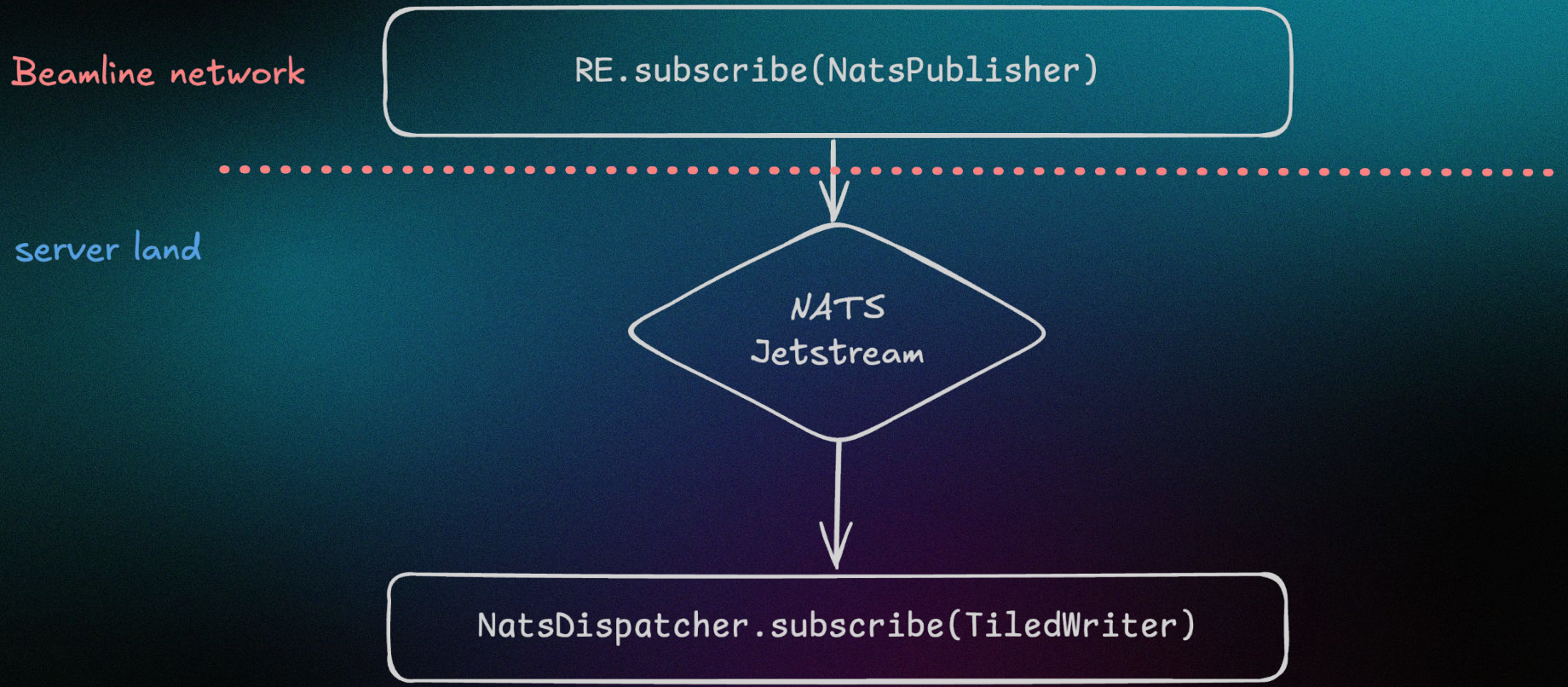




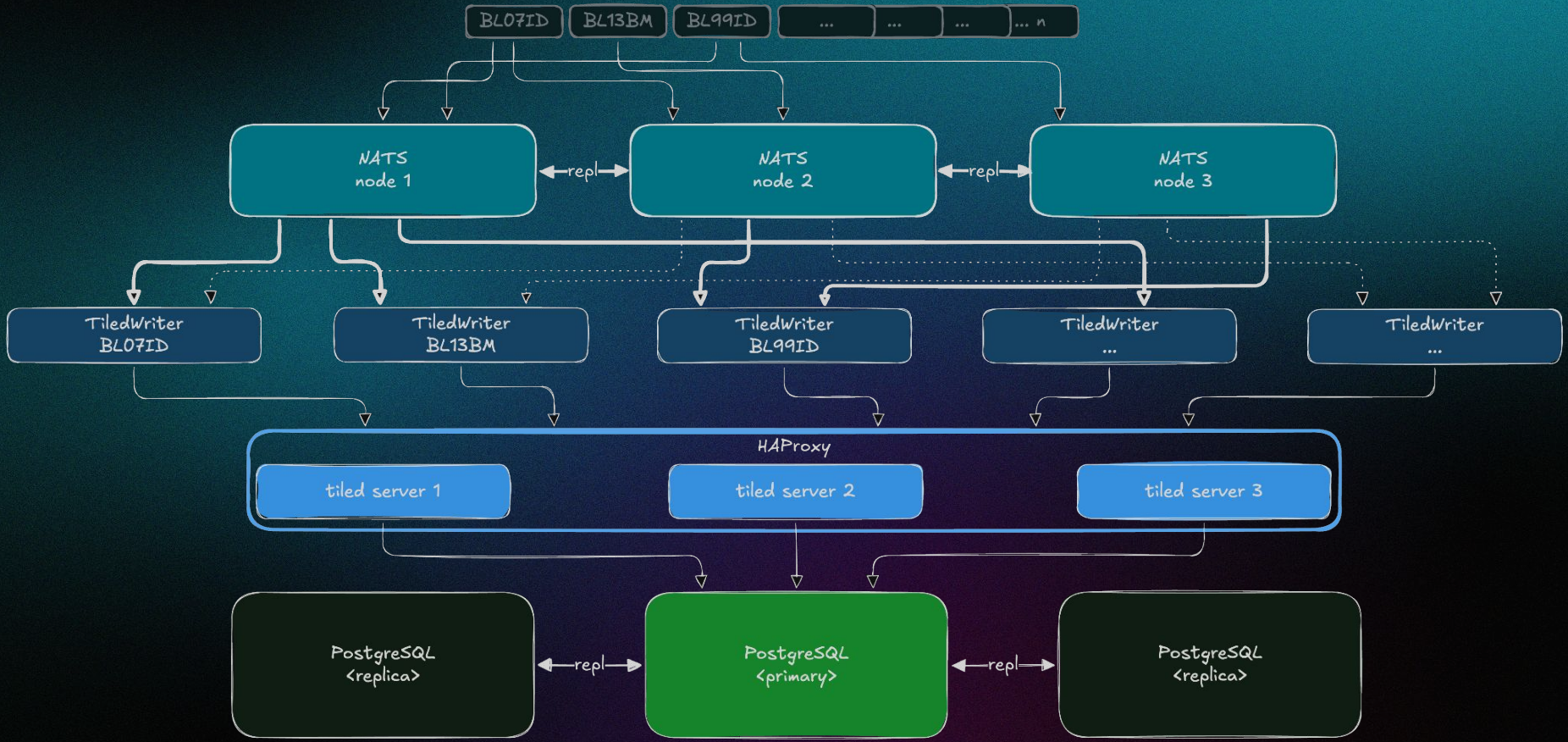
System layout

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

System layout



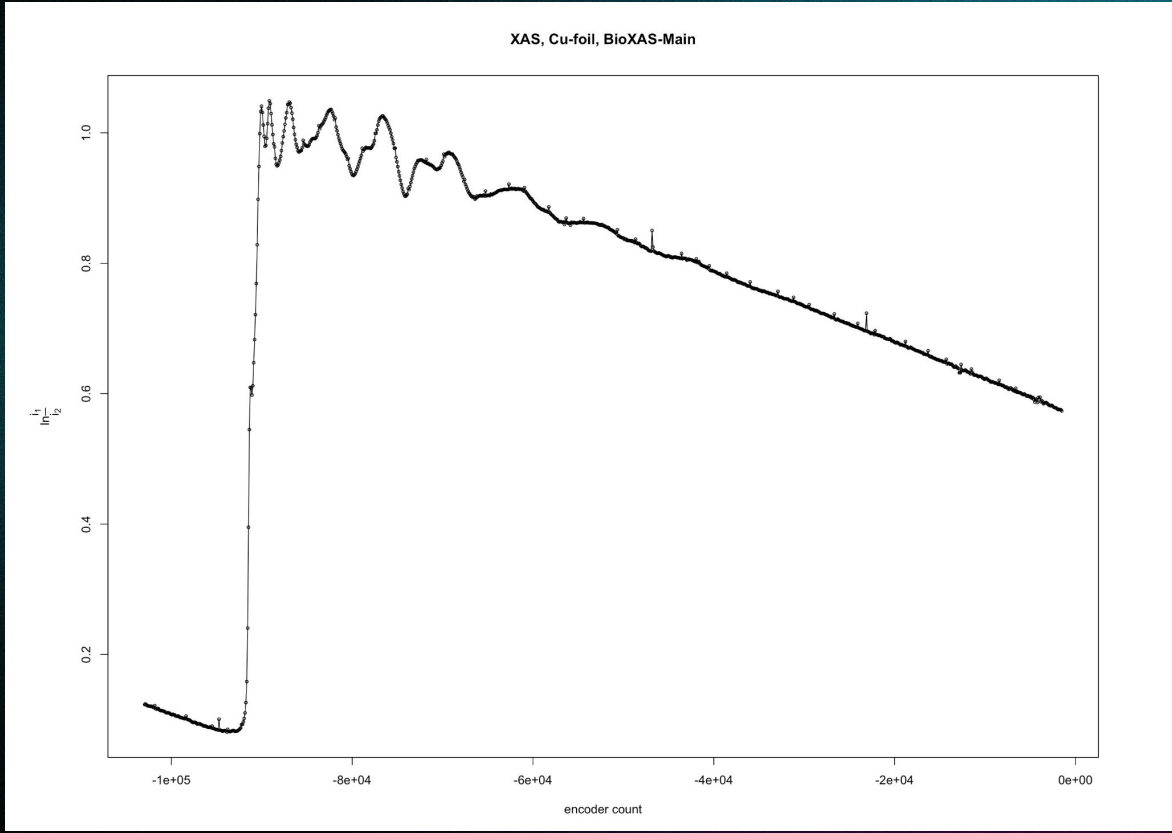
System layout (WIP)



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, cerate additional messages 🚧
  - Community support ❌

Milestone: Bluesky in action



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