A Scalable Export Solution for Facility Data

By
Pranali S Patil
Tata Institute of Fundamental Research, Hyderabad

Outline

- Types of Data
- Plan for Facility Data
- Source : Apache Kafka , EPICS Archiver
- Target : Operations Gateway
- Why we need a Data Exporter?
- Data Exporter
  - Overview
  - Data Flow
- Scalability: Problem - Solution

Types of Data

Experimental data

- Intended for (external) users
- Recorded at full rep rate
- Collected On shot data
- High data volumes
- Can originate from any device

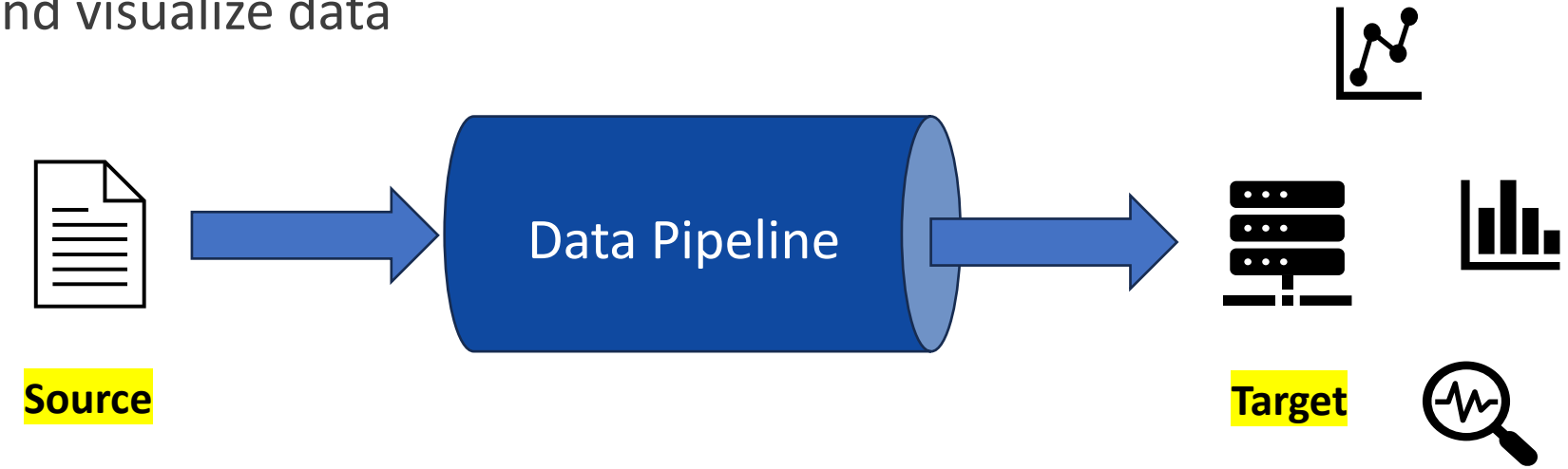
Facility / Operational data

- Intended for facility staff
- Recorded at low rep rate
- Collected periodically
- Low data volumes
- Can originate from any device

Plan for Facility Data

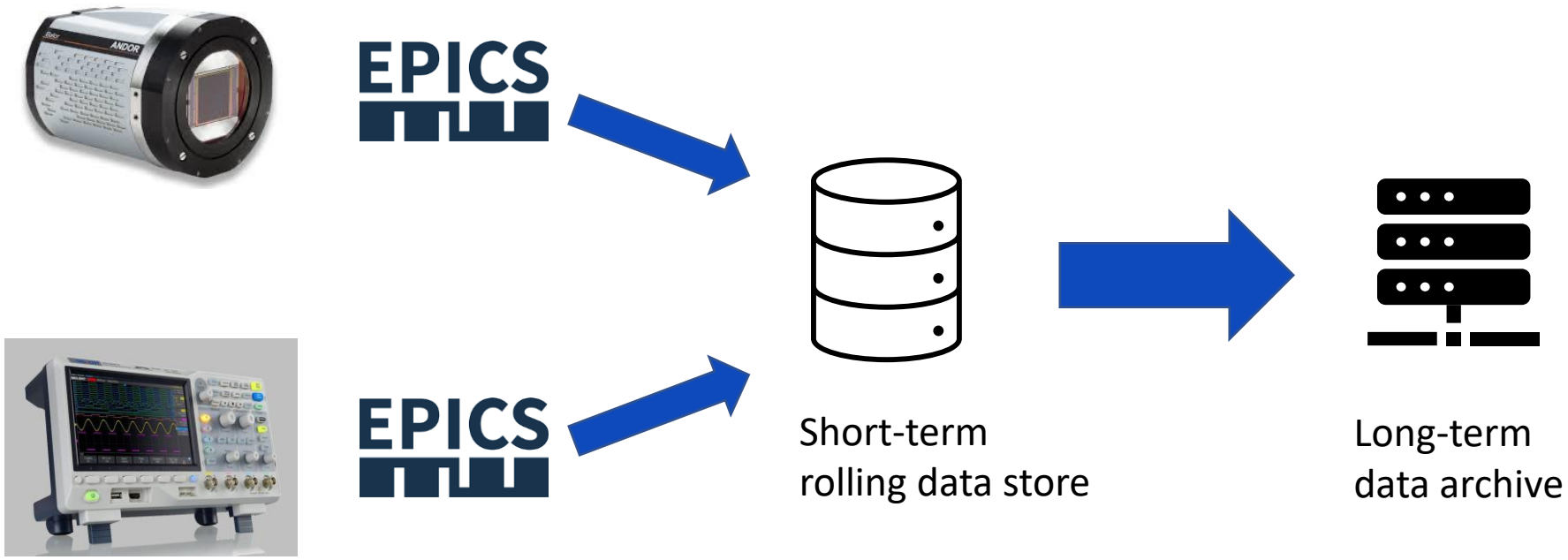
To create a data pipeline to:

1. Capture
2. Aggregate
3. Archive
4. Analyze and visualize data



Plan for Data Management

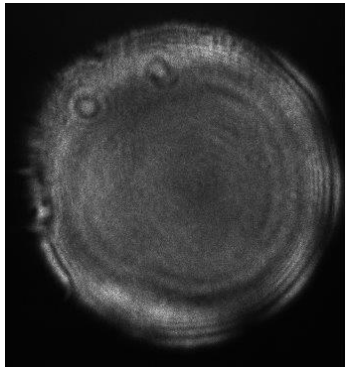
Create a data pipeline for capturing, streaming, and viewing data



Data binding– Event Based

Single Event with corresponding ID

Id	Time	N LEG1 GREEN NF IMAGE	N LEG2 GREEN NF IMAGE	N FLUOR IMAGE	N UNCOMP NF IMAGE
391798	2022-09-28 17:11:43				
391797	2022-09-28 17:11:23				



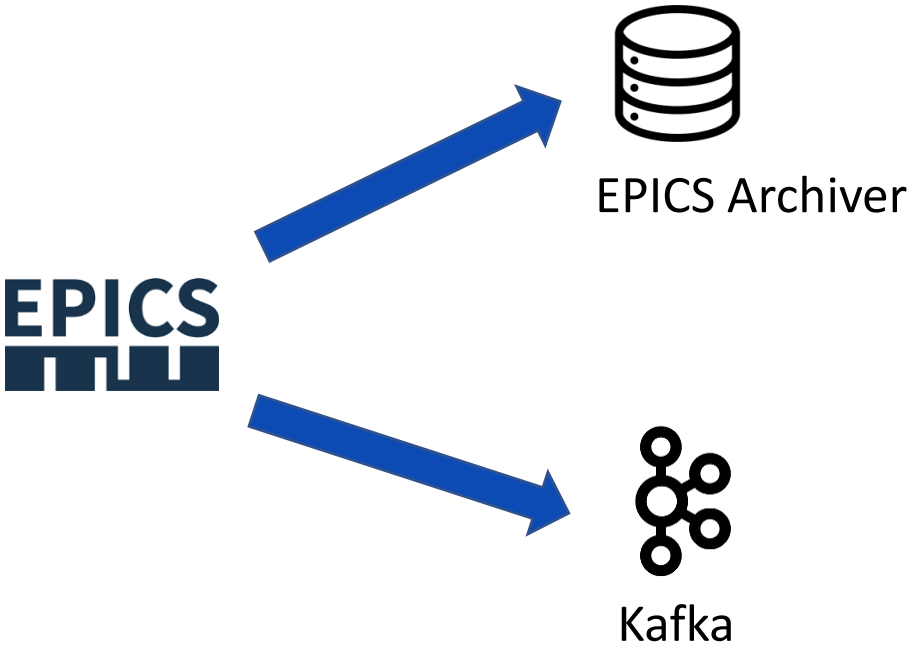
- Timestamp
- Camera model
- Exposure time
- Gain settings
- Camera location
- Filter settings

- Perform event building to match up multiple data streams
- Connect data with important metadata

Data Source

Data Generation:

- Data is produced by devices and instruments, captured in real-time streams (Kafka) and stored in the EPICS Archiver.

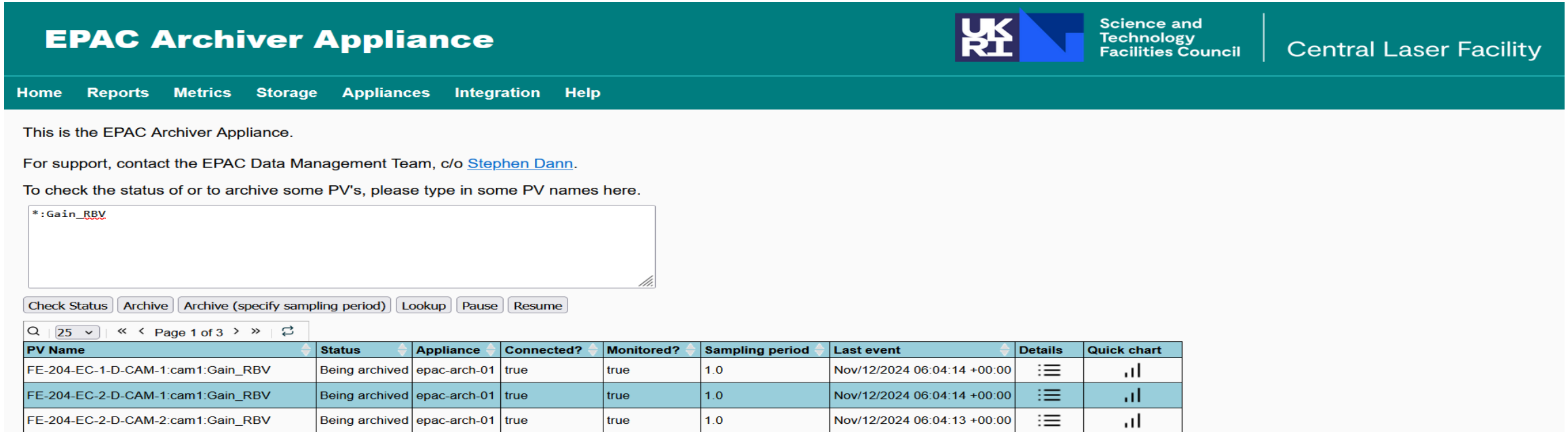


Data Sources:

- **Kafka:** Real-time data streams from devices.
- **EPICS Archiver:** Historical data indexed by timestamps.

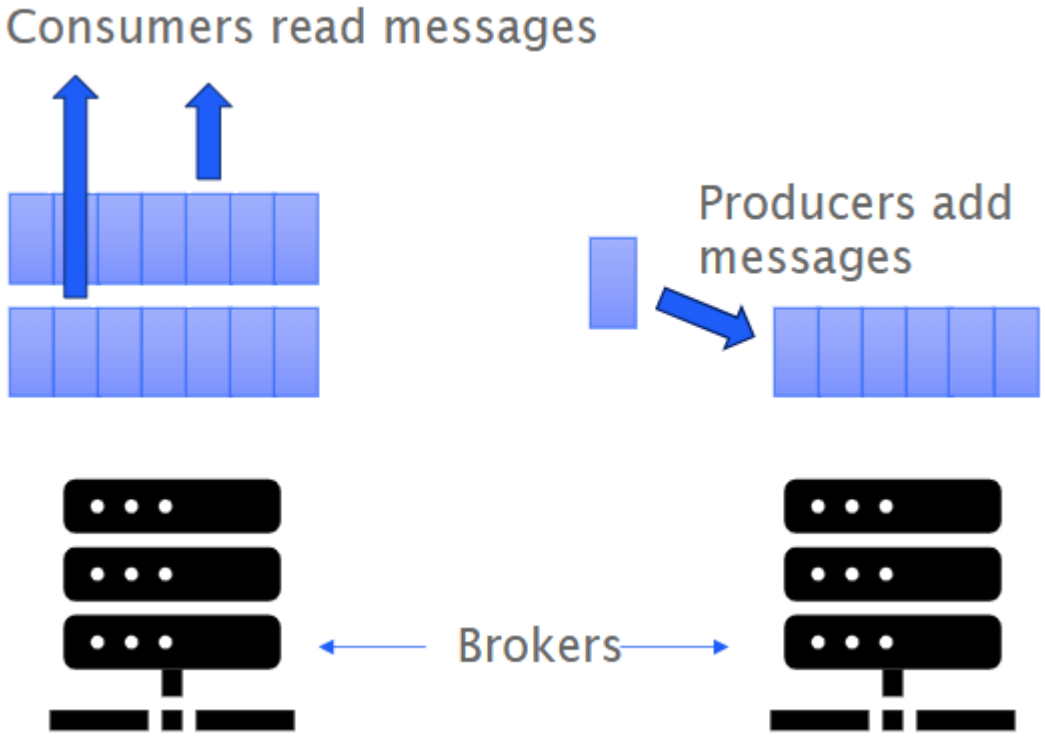
EPICS Archiver Appliance

- Data archiver for EPICS control system
- PVs are configured through the management UI / Rest API
- Mostly used for scalars



Apache Kafka

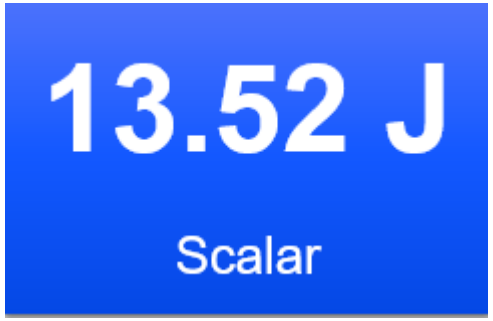
- High performance distributed system for handling **data streams**
- **Data** is split into messages and added to topics
- Each **topic** is a queue
- **Producers** and **consumers** work independently
- Configurable **data retention** and **replication**

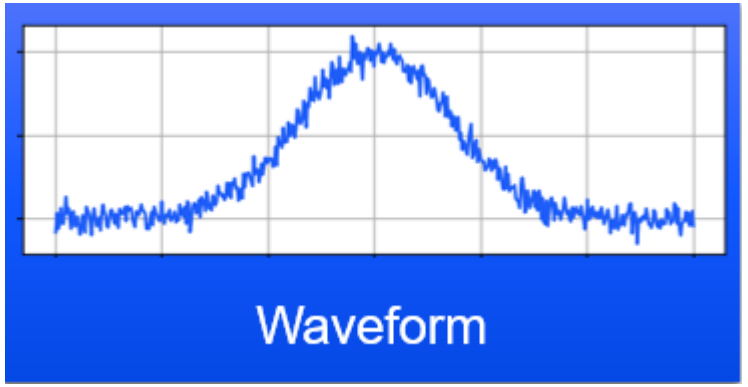


How do we use Kafka?

Each message is a single measurement, including metadata

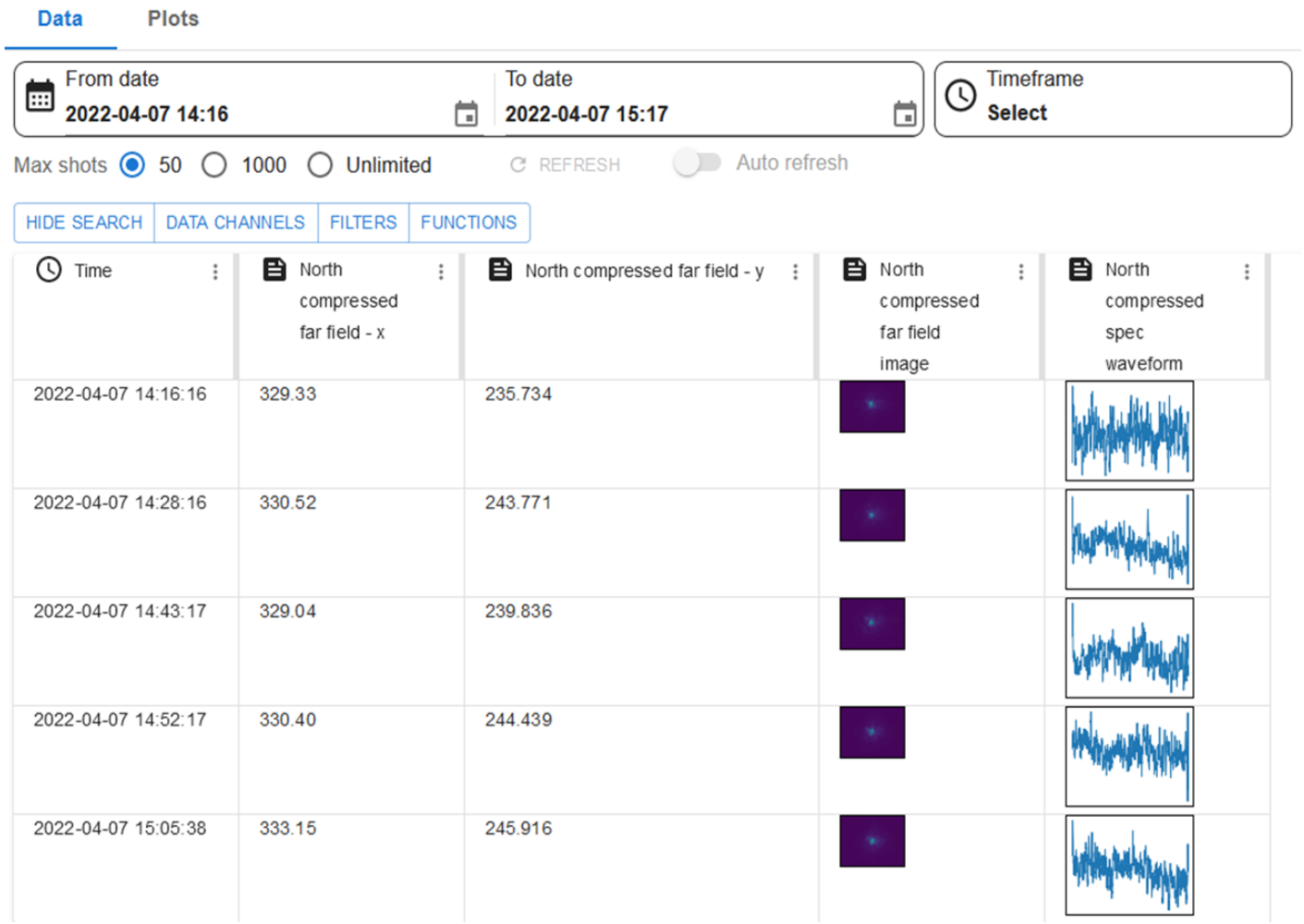
Each topic contains data from one source (or a group)



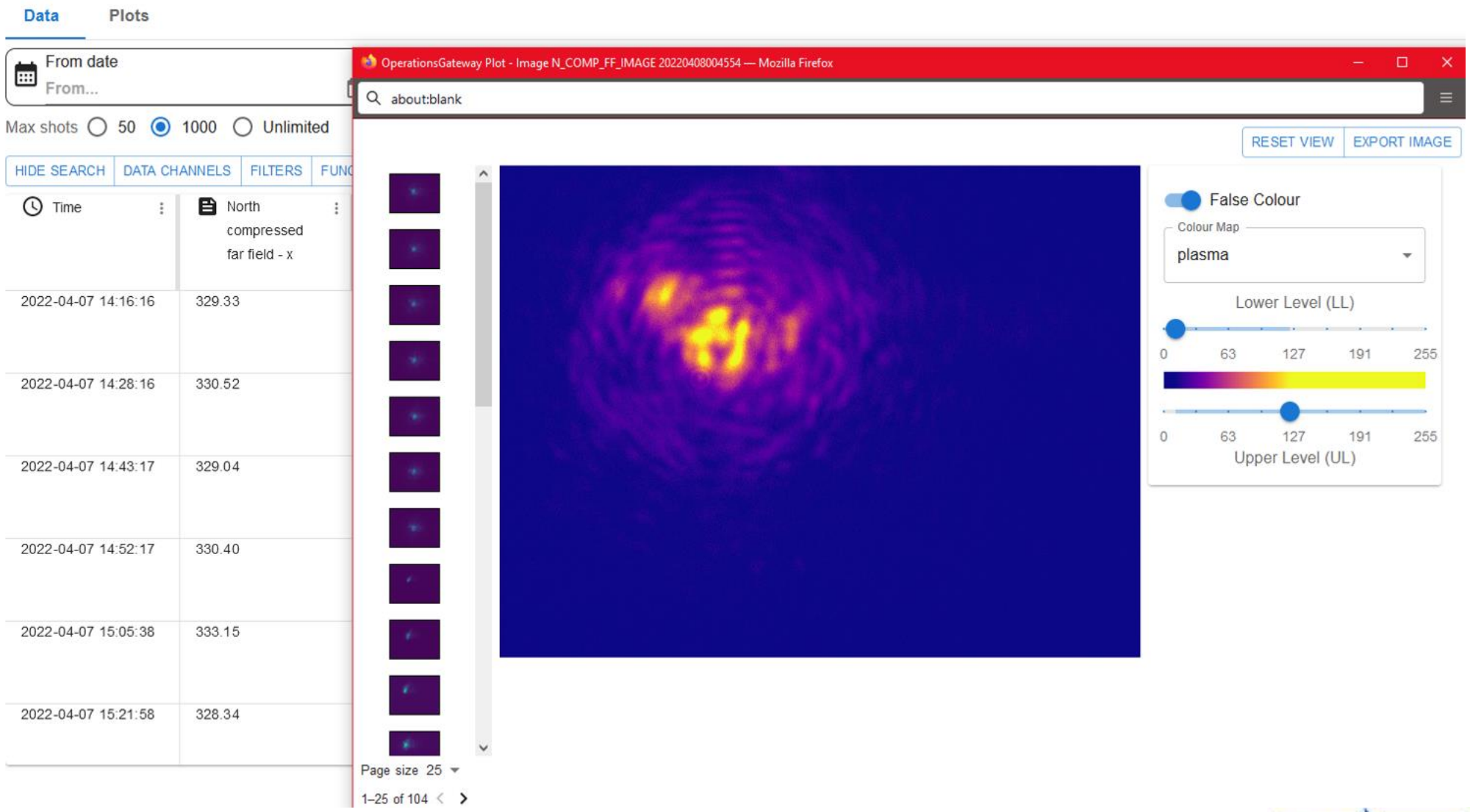


OperationsGateway

- A data discovery, **visualization and analysis tool** for EPAC historic facility data.
- **Store and view** the facility diagnostic data for EPAC
- For CLF operators (not visiting users)
- Being developed by STFC Scientific Computing team



OperationsGateway

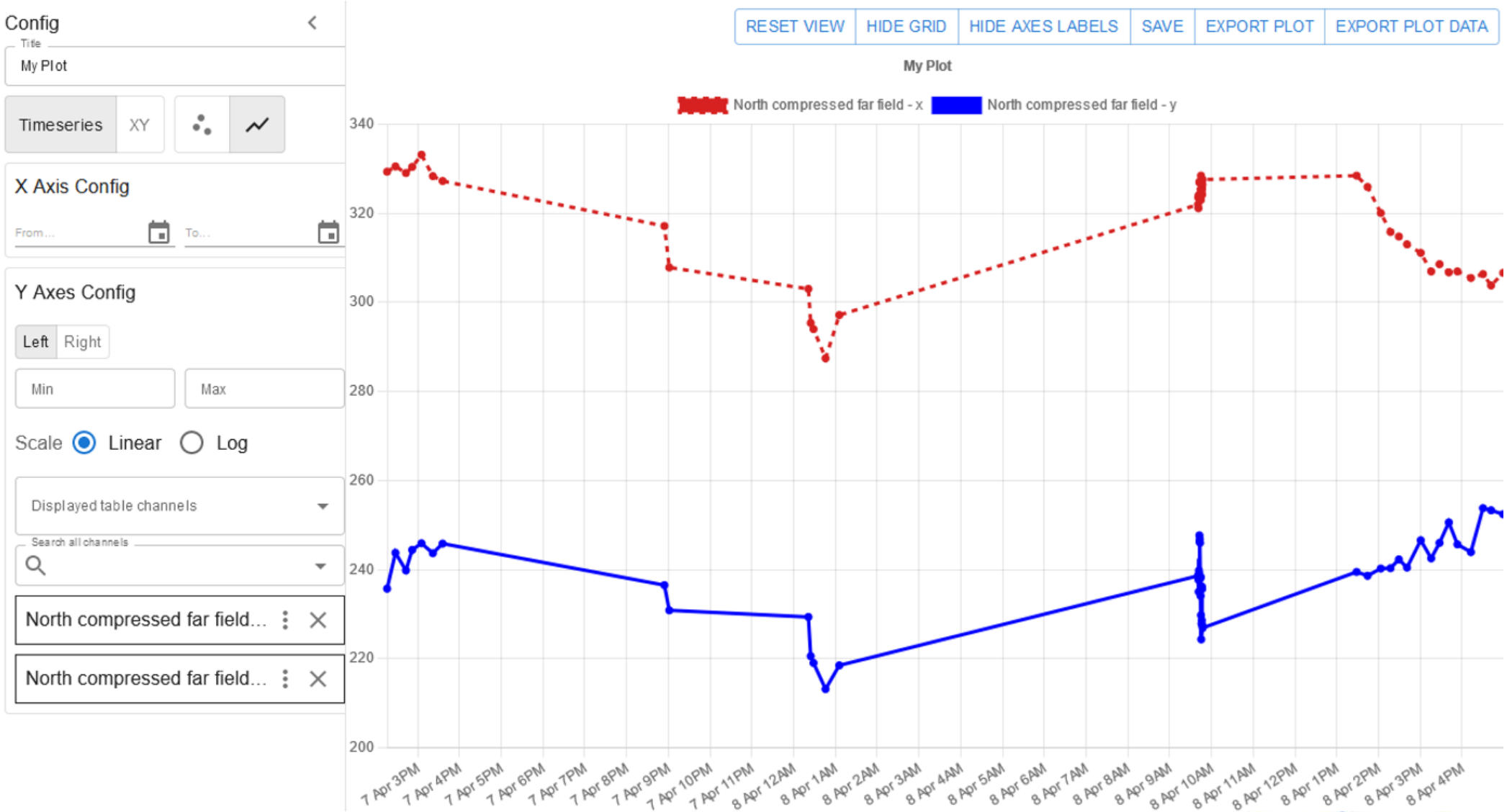


OperationsGateway

EPIC
Extreme Photonics Innovation Centre

OperationsGateway

central laser facility



tifr

UKRI Science and Technology Facilities Council

Data Exporter

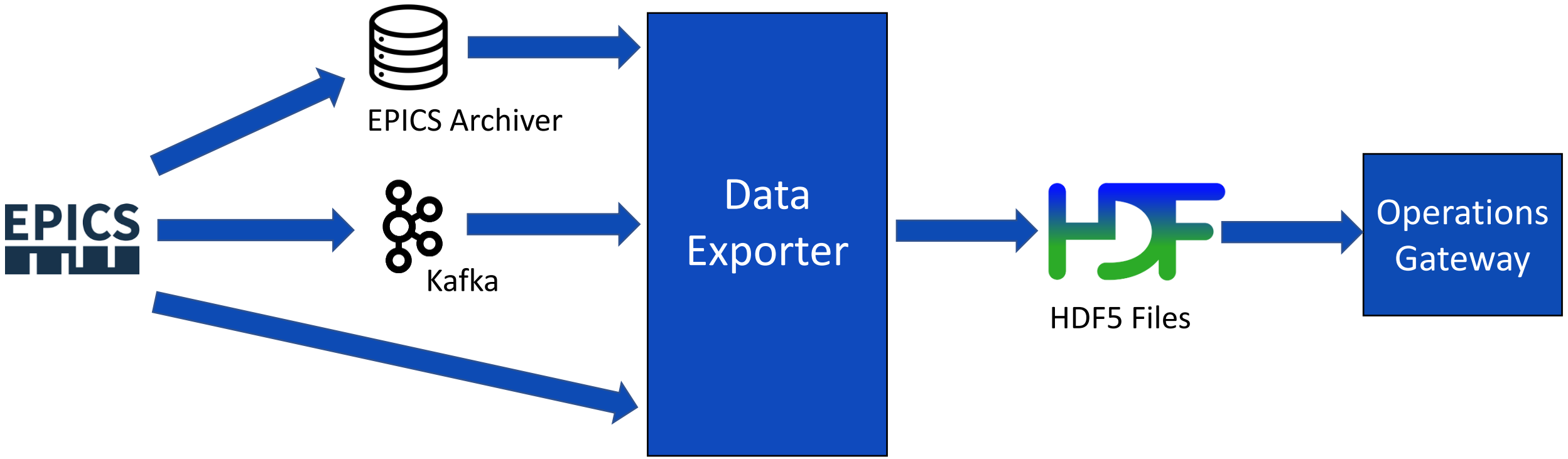
Problems:

- Capture data based on **Pulse ID and timestamp**
- Get data from **2 data sources** (EPICS Archiver and Kafka)
- Deserialize data from Kafka
- Data needs to be in HDF5 file format when uploading to Operations Gateway
- **Extra Metadata** needed in files to upload:
  - Active experiment, active area, Pulse ID and timestamp
- Capture data in variable rep rate

Solution:

- **Data exporter** is a Python application used to collect data based on Pulse ID and timestamp.
- Runs periodically

Data Exporter: Overview



Data Retrieval from Archiver

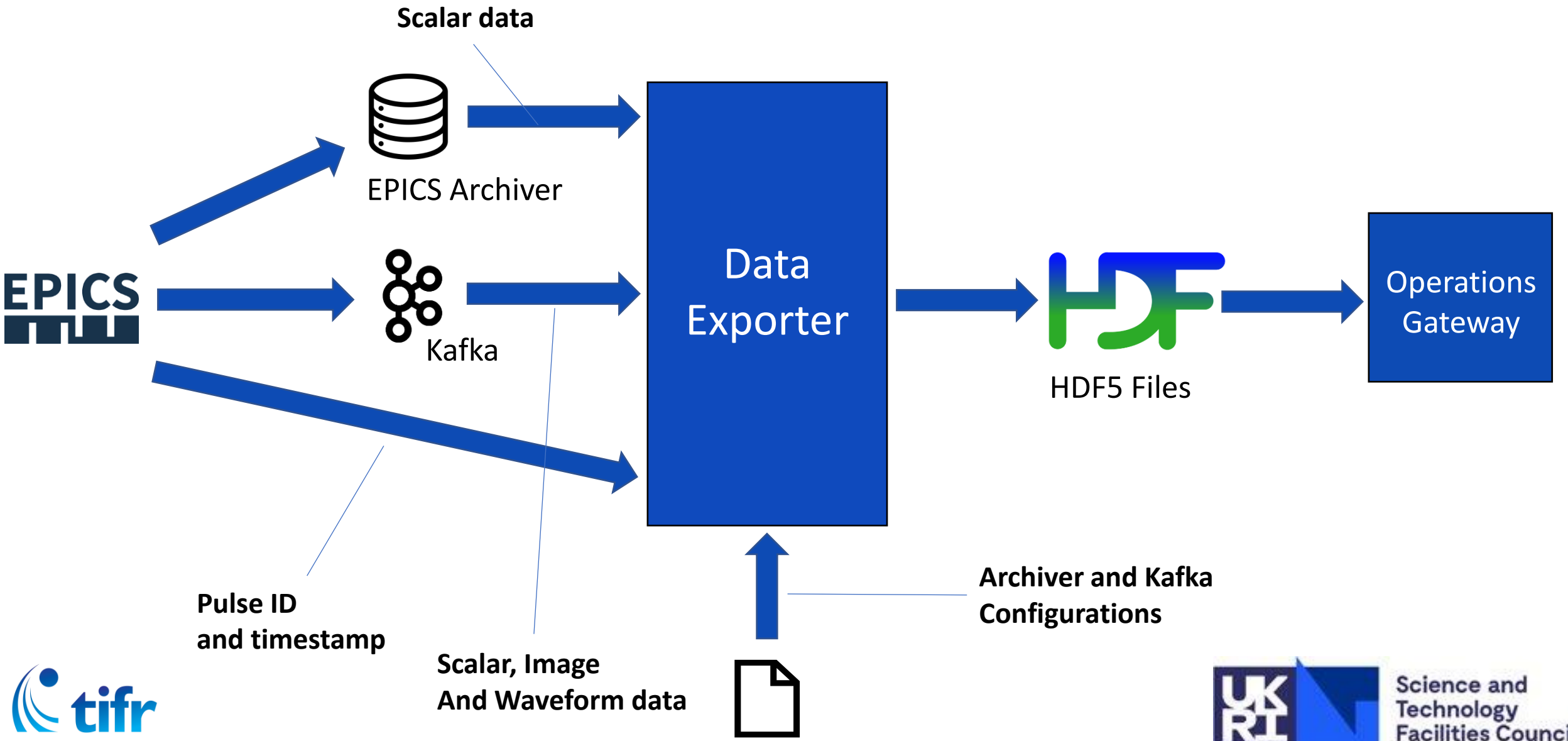
- Archiver has a **data retrieval URL**
- Archiver URL can be configured to **get data** for PVs with metadata
- Need to **send a request** to archiver with PV ('s) and timestamp
- Response from archiver is in **JSON** format.
- The response has 2 sections:
  - **meta** – Metadata - Name, Precision, etc.
  - **data** – Value, Timestamp(nanos, secs), Alarm fields(Severity, Status), etc.


```
{
  "meta": {
    "name": "EPAC-DEV:CAM1:cam1:AcquireTime",
    "PREC": "3"
  },
  "data": [
    {
      "secs": 1731327792,
      "val": 0.005,
      "nanos": 994324767,
      "severity": 0,
      "status": 0,
      "fields": {
        "cnxlostepsecs": "0",
        "startup": "true",
        "cnxregainedepsecs": "1731327814"
      }
    },
    {
      "secs": 1731568026,
      "val": 0.005,
      "nanos": 185526394,
      "severity": 0,
      "status": 0,
      "fields": {
        "cnxlostepsecs": "0",
        "startup": "true",
        "cnxregainedepsecs": "1731568046"
      }
    }
```

Data retrieval from Kafka

- To get data from Kafka, we have developed custom library in Rust with a Python API.
- Input is topic name, Pulse ID and timestamp
- Data is captured, deserialized and streamed to Python

Data flow


Scalar data
EPICS Archiver
EPICS
Kafka
Data Exporter
HDF5 Files
Operations Gateway
Pulse ID
and timestamp
Scalar, Image
And Waveform data
Archiver and Kafka
Configurations

Scalability challenges

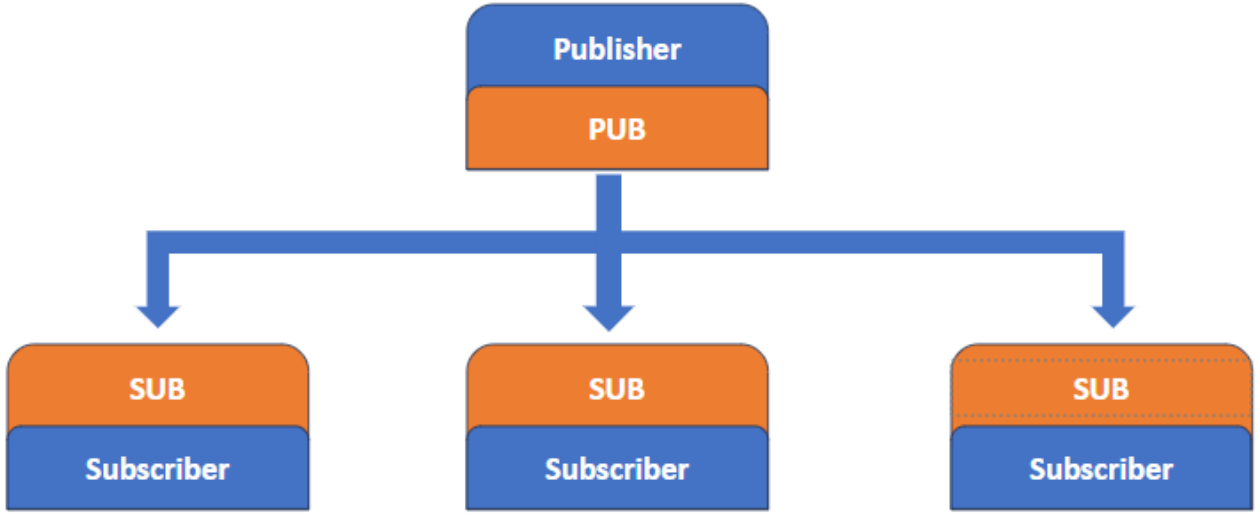
- **Large number of devices** (~150) across the facility
- Data is collected continuously - e.g., every 1 min or every 3 mins
- Writing the big chunk of data to HDF5 files
- One single exporter won't be enough
- Data capture with multiple instances:
  - Must be captured at same time with same Pulse ID

Leader – follower architecture

- Data exporter instances will be run as **one leader** and **more than one followers**
- Followers need to **connect** to the leader
- Leader broadcasts **pulse id and timestamp** to the followers
- Leader and followers **both run** using the pulse id and timestamp

ZeroMQ architecture

- **Leader** listens a socket
- **Followers** connects to the leader
- Socket type - **Publish-Subscribe**:
- **Leader** broadcasts data to multiple followers
- One-to-many distribution of information



Summary

A data management pipeline for facility data where EPICS devices generate data, stored in the **EPICS Archiver** and streamed via **Kafka**. **Data Exporter** synchronizes data from both sources using a laser pulse ID, saving it in **HDF5 format**. HDF5 files are **uploaded** to Operations Gateway. The data is then processed and visualized through the **Operations Gateway**.

Work in Progress:

- Need to add support for metadata
- Upload functionality to Operations Gateway
- Testing with real devices
- Planning to make the tool operational in mid 2025

Thank You!

References:

- EPICS Archiver Appliance — archiverdocs 0.1 documentation
- https://zeromq.org/