Analysis at the HL-LHC: Data Delivery, ServiceX, and Addressing Our Analysis Challenges

Ben Galewsky (UI), Ilija Vukotic (UChicago), **Gordon Watts (UW/Seattle)**, Roger Janusiak (UW/Seattle), Peter Onyisi (UTexas), KyungEon Choi (UTexas), Artur Cordeiro Oudot Choi (UW/Seattle), Mason Proffitt(UW/Seattle)

And others that have contributed!

Supported by grants from the NSF: OAC-1836650 and PHY-2323298

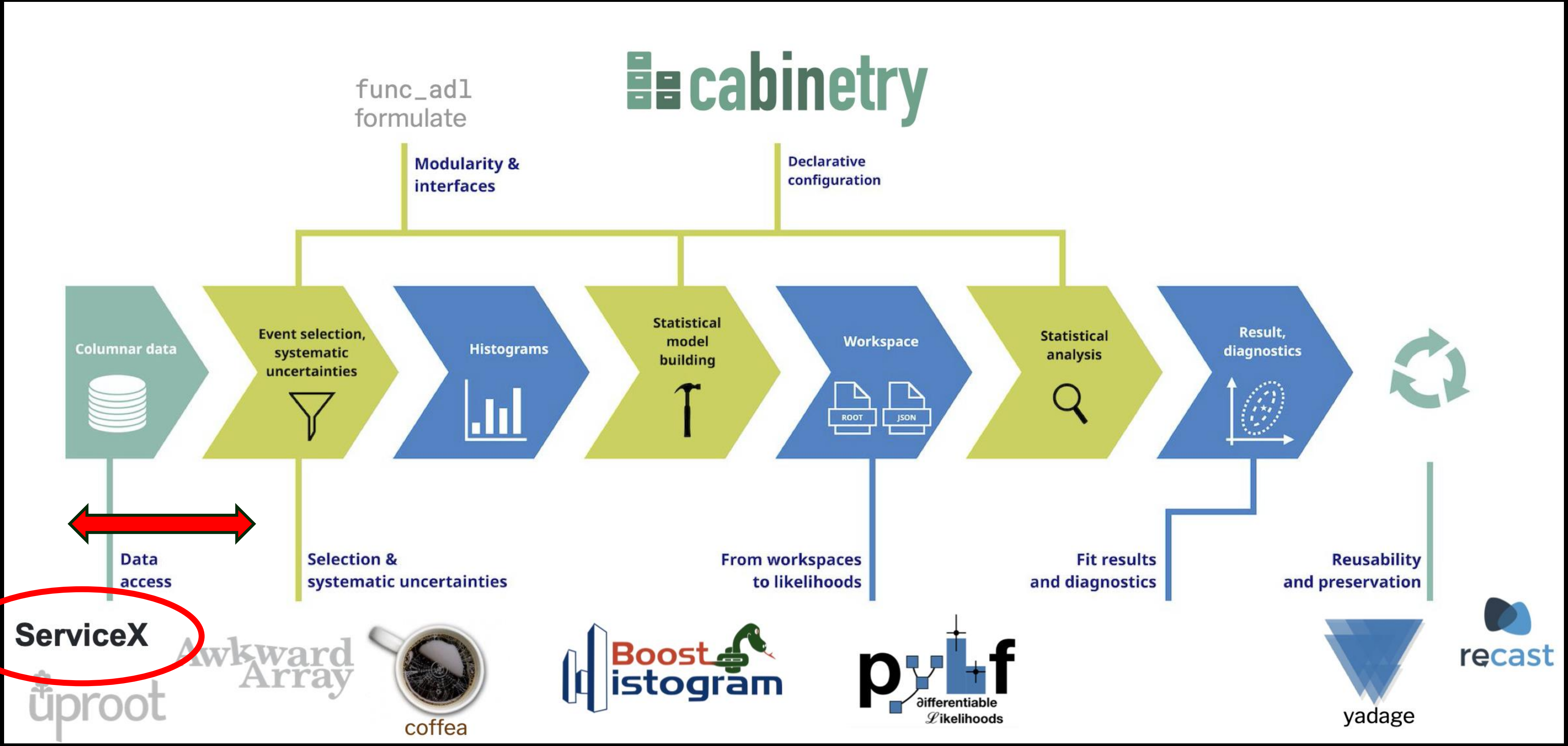
ServiceX – Introduction

ServiceX: Data Delivery for HL-LHC Analysis At Scale

On Demand
Experiment Agnostic Design
Accessible Anywhere

"Faster time to insight"

Where does it fit in Analysis?



ServiceX

Goals

- Produce focused datasets for analysis
- Produce training datasets for ML (high- and low-level features)
- Quick small datasets/skims to answer specific systematic studies
- Operate Quickly
- Simplify user skimming and thinning

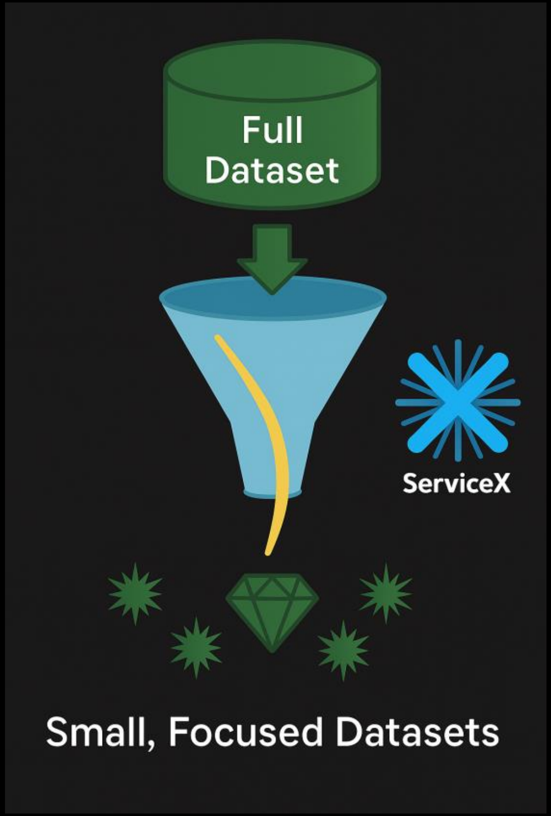
Non-Goals

- Produce Histograms
- Run statistical models
- General Purpose Batch Facility
- Produce datasets that require significant computing

ServiceX: Data Delivery for HL-LHC Analysis At Scale

On Demand

- **Quick access to data**
  - Less need to develop giant derivations with everything you might need under any circumstance
  - Small, ephemeral, focused datasets possible
- **Keep queries around, not giant datasets**
  - Try to eliminate these very large intermediate datasets we have in analysis
- **Depends on dataset generation being cheap:** easy to do, and quick



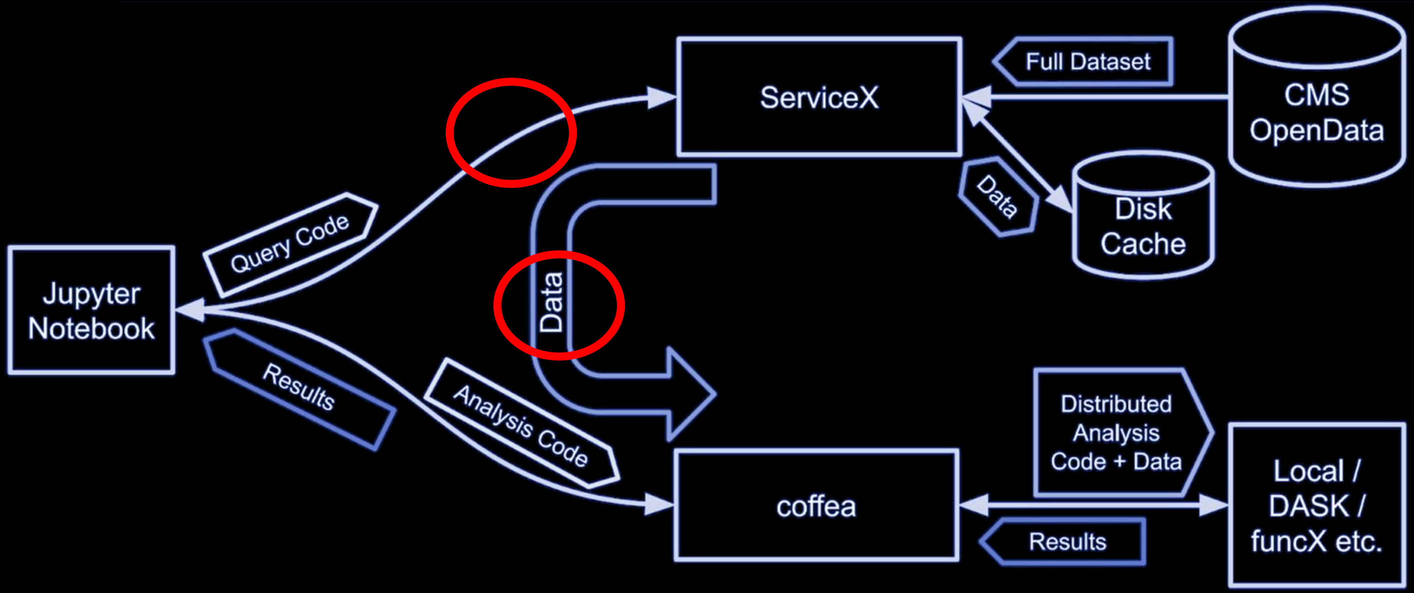
ServiceX: Data Delivery for HL-LHC Analysis At Scale

Experiment Agnostic Design

- ServiceX manages
  - file delivery
  - processing (transformation),
  - output dataset management and access.
- Supports **arbitrary input formats**
  - Support for ATLAS proprietary formats like xAOD, preliminary support for CMS miniAOD
  - Support for ROOT TTree and some initial support for RNTuple, etc.
  - Supported by exiting experiment frameworks (e.g. EventLoop and AnalysisBase release in ATLAS) in docker images
- Output Formats include TTree, RNtuple, and parquet
- Input data source: pluggable architecture current supports Rucio, http, and xrootd.

ServiceX: Data Delivery for HL-LHC Analysis At Scale

Accessible Anywhere



- WebAPI Interface to start transforms
- Amazon's Object Store API for accessing resulting data (S3) in local object store
  - compatible with **python ecosystem** and **ROOT framework**

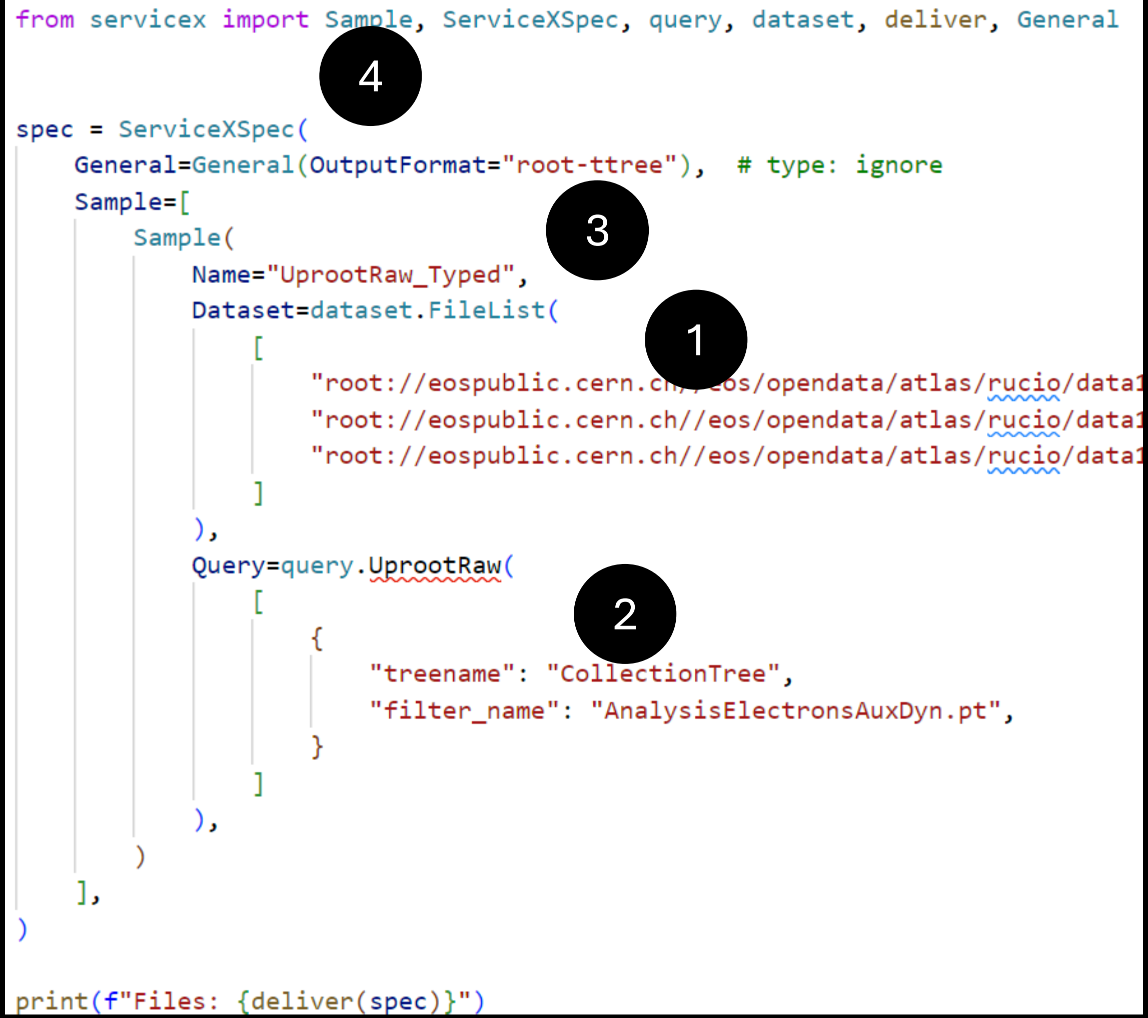
- Accessible from laptop to Analysis Facility
- Support software for local downloads
- Direct access to data in S3 for AF's

Quick Examples

PHOTOTHEQUE IN2P3 / CNRS

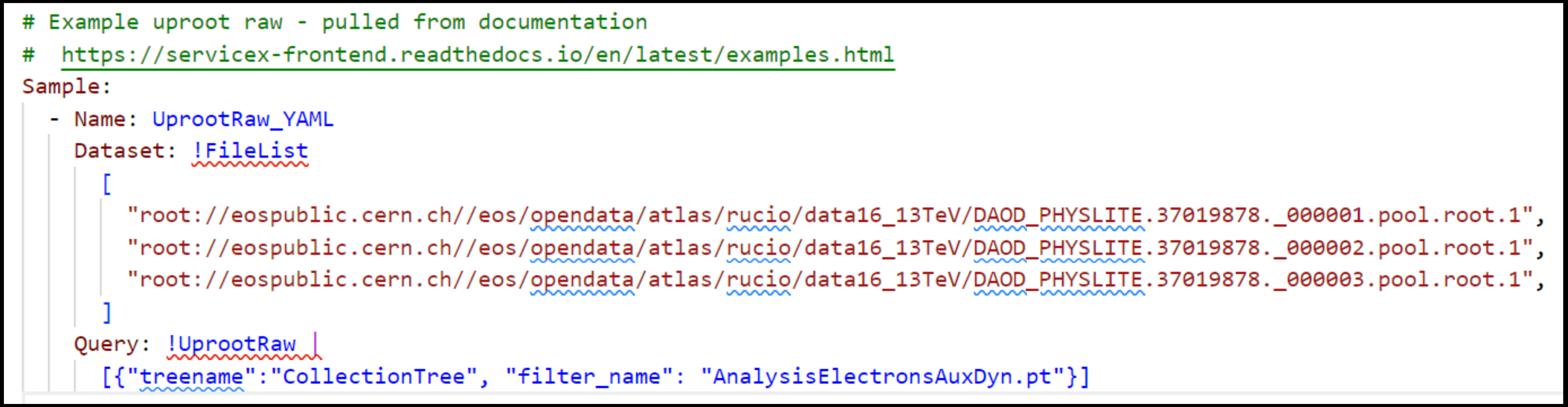
Components of a Transformation Request

1. The Input Data
2. The transformation to apply
3. The Output Format
4. How the transformation is specified



```
from servicex import Sample, ServiceXSpec, query, dataset, deliver, General


spec = ServiceXSpec(
    General=General(OutputFormat="root-ttree"),  # type: ignore
    Sample=[
        Sample(
            Name="UprootRaw_Typed",
            Dataset=dataset.FileList(
                [
                    "root://eospublic.cern.ch//eos/opendata/atlas/rucio/data1
                    "root://eospublic.cern.ch//eos/opendata/atlas/rucio/data1
                    "root://eospublic.cern.ch//eos/opendata/atlas/rucio/data1
                ]
            ),
            Query=query.UprootRaw(
                [
                    {
                        "treename": "CollectionTree",
                        "filter_name": "AnalysisElectronsAuxDyn.pt",
                    }
                ]
            ),
        )
    ],
)

print(f"Files: {deliver(spec)}")
```

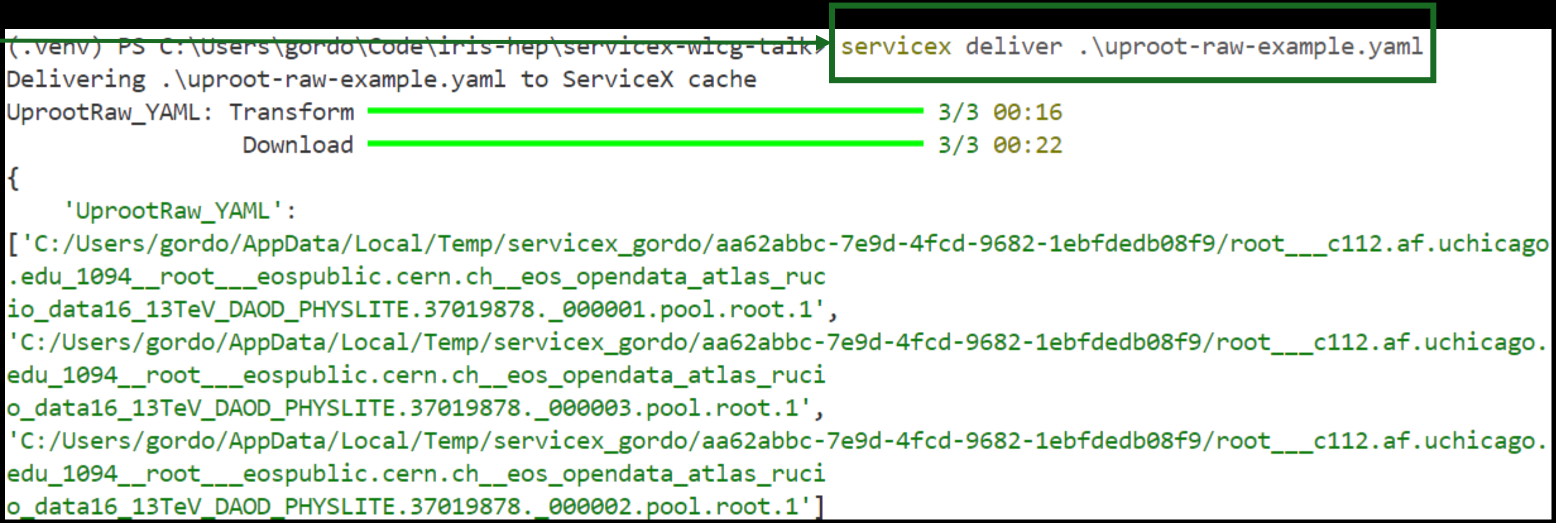
Transform specified as a YAML file



```
# Example uproot raw - pulled from documentation
#  https://servicex-frontend.readthedocs.io/en/latest/examples.html
Sample:
  - Name: UprootRaw_YAML
    Dataset: !FileList
      [
        "root://eospublic.cern.ch//eos/opendata/atlas/rucio/data16_13TeV/DAOD_PHYSLITE.37019878._000001.pool.root.1",
        "root://eospublic.cern.ch//eos/opendata/atlas/rucio/data16_13TeV/DAOD_PHYSLITE.37019878._000002.pool.root.1",
        "root://eospublic.cern.ch//eos/opendata/atlas/rucio/data16_13TeV/DAOD_PHYSLITE.37019878._000003.pool.root.1",
      ]
    Query: !UprootRaw
      [{"treename":"CollectionTree", "filter_name": "AnalysisElectronsAuxDyn.pt"}]
```

Command-line invocation

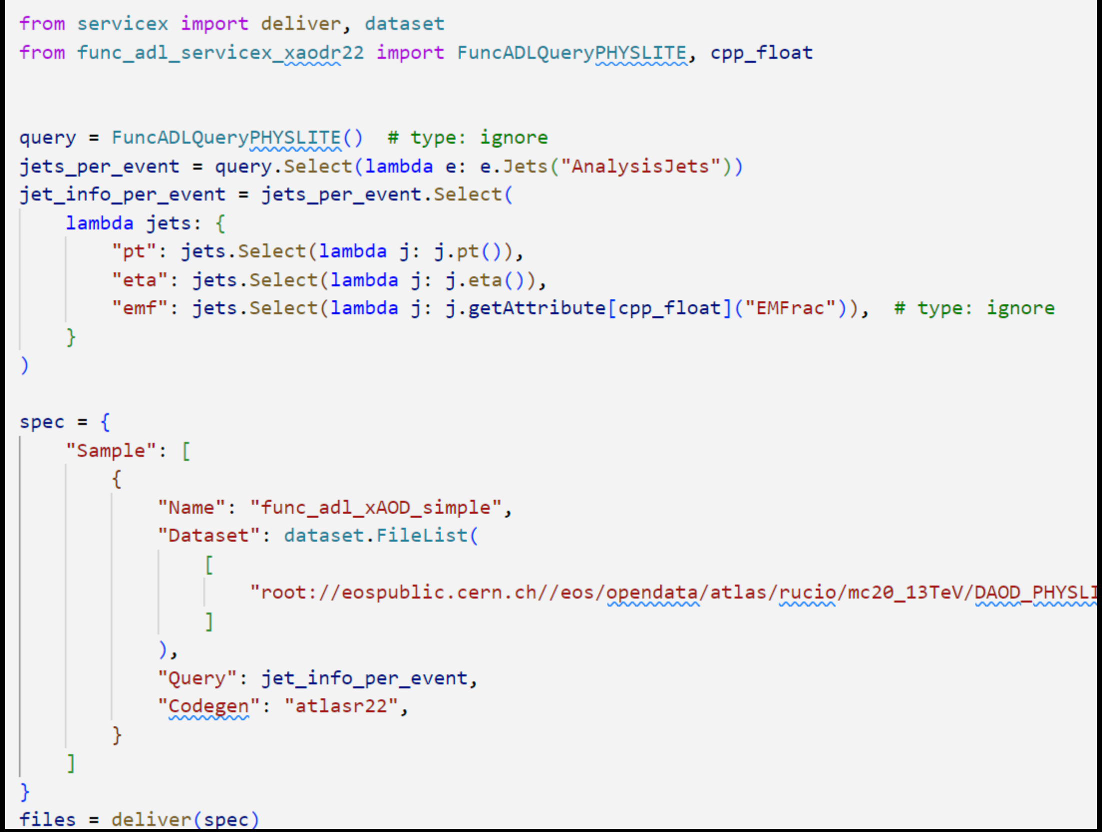
3 Output Files



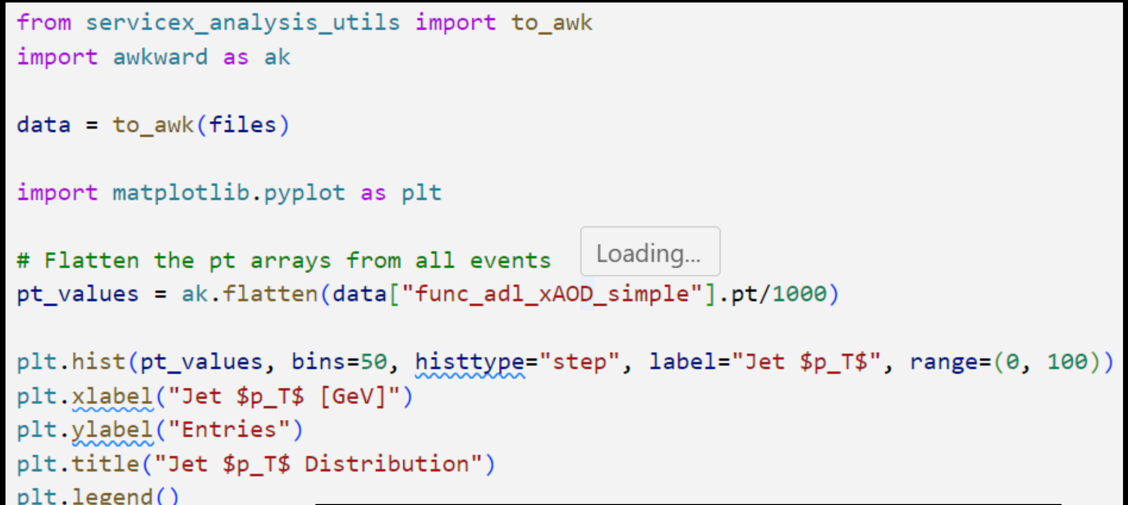
```
(.venv) PS C:\Users\gordo\Code\iris-hep\servicex-wlcg-talk> servicex deliver .\uproot-raw-example.yaml
Delivering .\uproot-raw-example.yaml to ServiceX cache
UprootRaw_YAML: Transform ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3/3 00:16
                 Download ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3/3 00:22
{
    'UprootRaw_YAML':
['C:/Users/gordo/AppData/Local/Temp/servicex_gordo/aa62abbc-7e9d-4fcd-9682-1ebfdedb08f9/root___c112.af.uchicago
.edu_1094__root___eospublic.cern.ch__eos_opendata_atlas_ruc
io_data16_13TeV_DAOD_PHYSLITE.37019878._000001.pool.root.1',
'C:/Users/gordo/AppData/Local/Temp/servicex_gordo/aa62abbc-7e9d-4fcd-9682-1ebfdedb08f9/root___c112.af.uchicago.
edu_1094__root___eospublic.cern.ch__eos_opendata_atlas_ruci
o_data16_13TeV_DAOD_PHYSLITE.37019878._000003.pool.root.1',
'C:/Users/gordo/AppData/Local/Temp/servicex_gordo/aa62abbc-7e9d-4fcd-9682-1ebfdedb08f9/root___c112.af.uchicago.
edu_1094__root___eospublic.cern.ch__eos_opendata_atlas_ruci
o_data16_13TeV_DAOD_PHYSLITE.37019878._000002.pool.root.1']
```

- Easily checked into git
- Tracking of "static" datasets
- Avoid complex logic in dataset generation
- Query isn't always most readable...

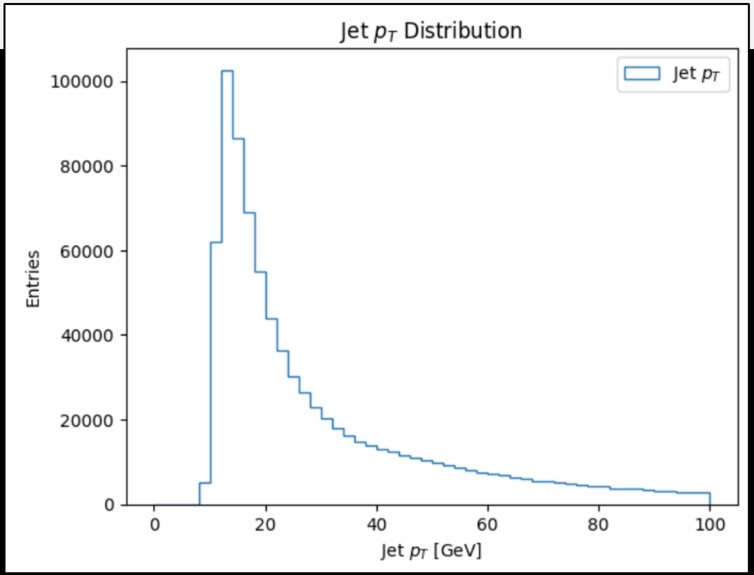
Cells of a Jupyter Notebook



```
from servicex import deliver, dataset
from func_adl_servicex_xaodr22 import FuncADLQueryPHYSLITE, cpp_float


query = FuncADLQueryPHYSLITE()  # type: ignore
jets_per_event = query.Select(lambda e: e.Jets("AnalysisJets"))
jet_info_per_event = jets_per_event.Select(
    lambda jets: {
        "pt": jets.Select(lambda j: j.pt()),
        "eta": jets.Select(lambda j: j.eta()),
        "emf": jets.Select(lambda j: j.getAttribute[cpp_float]("EMFrac")),  # type: ignore
    }
)

spec = {
    "Sample": [
        {
            "Name": "func_adl_xAOD_simple",
            "Dataset": dataset.FileList(
                [
                    "root://eospublic.cern.ch//eos/opendata/atlas/rucio/mc20_13TeV/DAOD_PHYSLI
                ]
            ),
            "Query": jet_info_per_event,
            "Codegen": "atlasr22",
        }
    ]
}
files = deliver(spec)
```



```
from servicex_analysis_utils import to_awk
import awkward as ak

data = to_awk(files)

import matplotlib.pyplot as plt

# Flatten the pt arrays from all events
pt_values = ak.flatten(data["func_adl_xAOD_simple"].pt/1000)

plt.hist(pt_values, bins=50, histtype="step", label="Jet $p_T$", range=(0, 100))
plt.xlabel("Jet $p_T$ [GeV]")
plt.ylabel("Entries")
plt.title("Jet $p_T$ Distribution")
plt.legend()
plt.show()
```



[Notebook]

Used to perform quick studies of datasets exploring various "problems".

Recent Advances

TopCPToolkit Transformation

```
CommonServices:
  systematicsHistogram: 'listOfSystematics'

PileupReweighting: {}

EventCleaning:
    cionEventCleaning: False
    runGRL: False

Electrons:
  - containerName: 'AnaElectrons'
    crackVeto: True
    IFFClassification: {}
    WorkingPoint:
      - selectionName: 'loose'
        identificationWP: 'TightLH'
        isolationWP: 'NonIso'
        noEffSF: True
      - selectionName: 'tight'
        identificationWP: 'TightLH'
        isolationWP: 'Tight_VarRad'
        noEffSF: True
    PtEtaSelection:
        minPt: 25000.0
        maxEta: 2.47
        useClusterEta: True

# After configuring each container, many variables will be saved automatically.
Output:
  treeName: 'reco'
  vars: []
  metVars: []
  containers:
      # Format should follow: '<suffix>:<output container>'
      el_: 'AnaElectrons'
      '': 'EventInfo'
  commands:
    # Turn output branches on and off with 'enable' and 'disable'

AddConfigBlocks: []
```

```
Electrons:
  - containerName: 'AnaElectrons'
    crackVeto: True
    IFFClassification: {}
    WorkingPoint:
      - selectionName: 'loose'
        identificationWP: 'TightLH'
        isolationWP: 'NonIso'
        noEffSF: True
      - selectionName: 'tight'
        identificationWP: 'TightLH'
        isolationWP: 'Tight_VarRad'
        noEffSF: True
    PtEtaSelection:
        minPt: 25000.0
        maxEta: 2.47
        useClusterEta: True
```

ATLAS has developed the **TopCPToolkit** to help with analyzing PHYS and PHYSLITE data

- Started in top group, spread to large parts of experiment.
- Configured by yaml!
- Total control over what is written out
- Uses experiment’s calibration framework

We have turned this into a science image

- The “query” is the yaml configuration file.
- Makes light-weight transformations easy…

Developing a new Transform/Query Type

This is advanced coding

- must be done in concert with ServiceX developers
- site administrators.

But it isn’t technically difficult!

Codegen

Science Image (docker)no

Simple python API

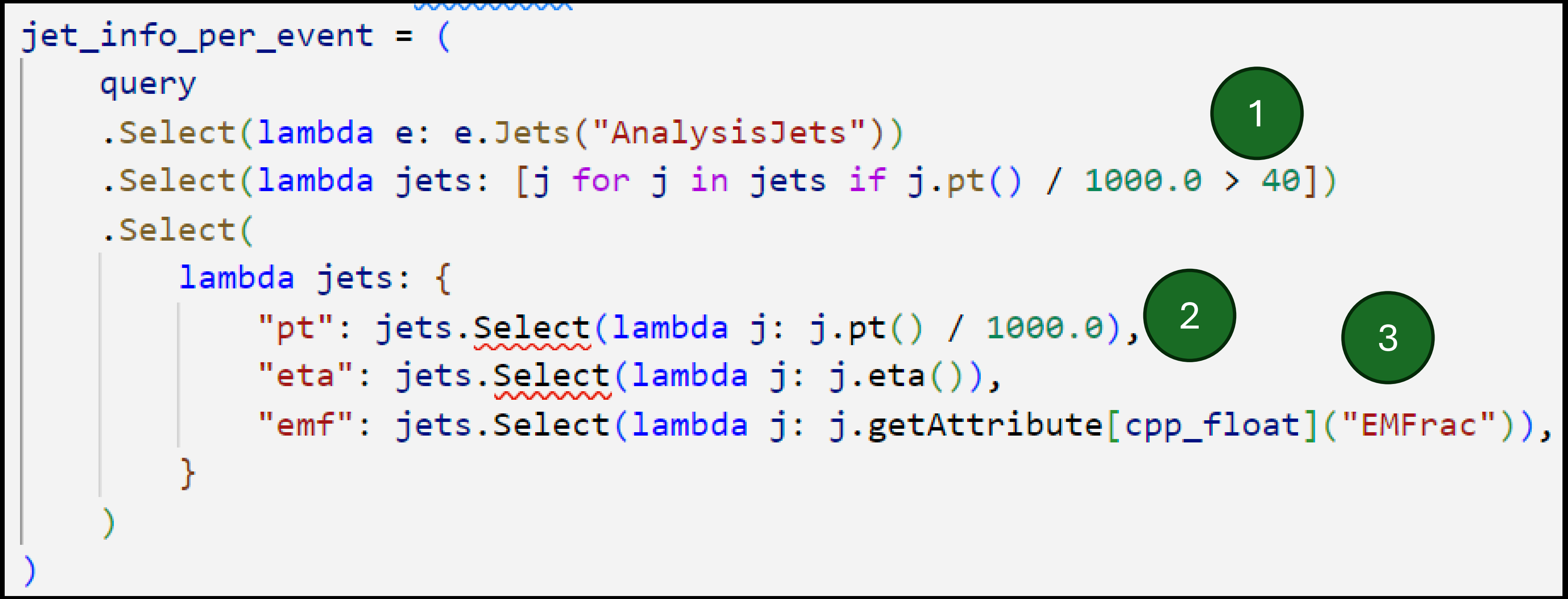
- Translates the query string
- configuration files
- scripts to run the transform

Docker image that can execute the scripts

- Can be based on whatever you want
- Does need to have x509 support!

Func_adl – Supporting rich types

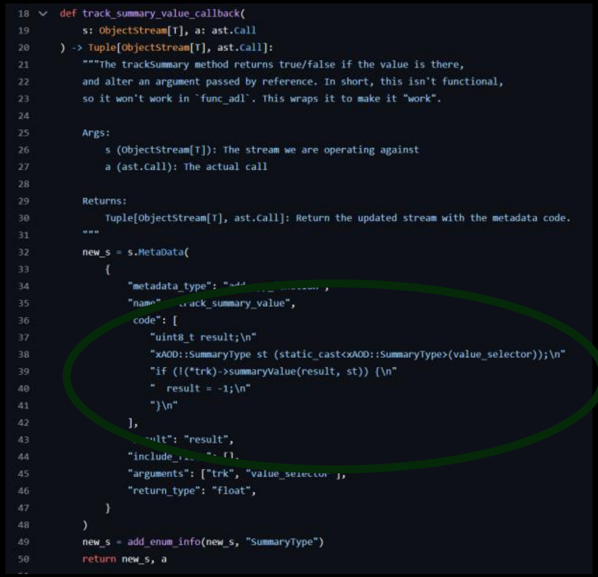
- Loosely based on SQL
- Allows fairly complex expressions
- Translated into C++ (AnalysisBase/miniAOD) or uproot/awkward (root input)
- Typed python – so you can take advantage of an IDE's type checking…



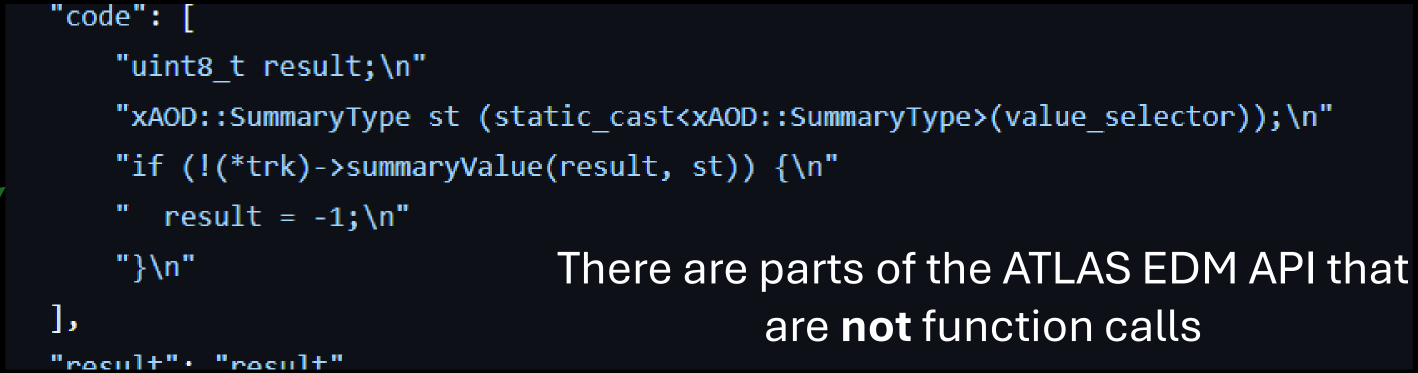
```
jet_info_per_event = (
    query
    .Select(lambda e: e.Jets("AnalysisJets"))
    .Select(lambda jets: [j for j in jets if j.pt() / 1000.0 > 40])
    .Select(
        lambda jets: {
            "pt": jets.Select(lambda j: j.pt() / 1000.0),
            "eta": jets.Select(lambda j: j.eta()),
            "emf": jets.Select(lambda j: j.getAttribute[cpp_float]("EMFrac")),
        }
    )
)
```

Recent support added for C++ constants, enums…

Allow leaky abstractions in C++…



```
def track_summary_value_callback(
    s: ObjectStream[T], a: ast.Call
) -> Tuple[ObjectStream[T], ast.Call]:
    """The trackSummary method returns true/false if the value is there,
    and alter an argument passed by reference. In short, this isn't functional,
    so it won't work in `func_adl`. This wraps it to make it "work".

    Args:
        s (ObjectStream[T]): The stream we are operating against
        a (ast.Call): The actual call

    Returns:
        Tuple[ObjectStream[T], ast.Call]: Return the updated stream with the metadata code.
    """
    new_s = s.MetaData(
        {
            "metadata_type": "add
            "name"  rack_summary_value",
             code": [
               "uint8_t result;\n"
               "xAOD::SummaryType st (static_cast<xAOD::SummaryType>(value_selector));\n"
               "if (!(*trk)->summaryValue(result, st)) {\n"
               "  result = -1;\n"
               "}\n"
            ],
              ult": "result",
            "include_
            "arguments": ["trk", "value_selector"],
            "return_type": "float",
        }
    )
    new_s = add_enum_info(new_s, "SummaryType")
    return new_s, a
```



```
"code": [
    "uint8_t result;\n"
    "xAOD::SummaryType st (static_cast<xAOD::SummaryType>(value_selector));\n"
    "if (!(*trk)->summaryValue(result, st)) {\n"
    "  result = -1;\n"
    "}\n"
],
"result": "result"
```

There are parts of the ATLAS EDM API that are **not** function calls

- Can inject C++ into the code stream in a composable way
- Allows you to build a library to work around non-functional parts of an experiment's EDM

Other Improvements

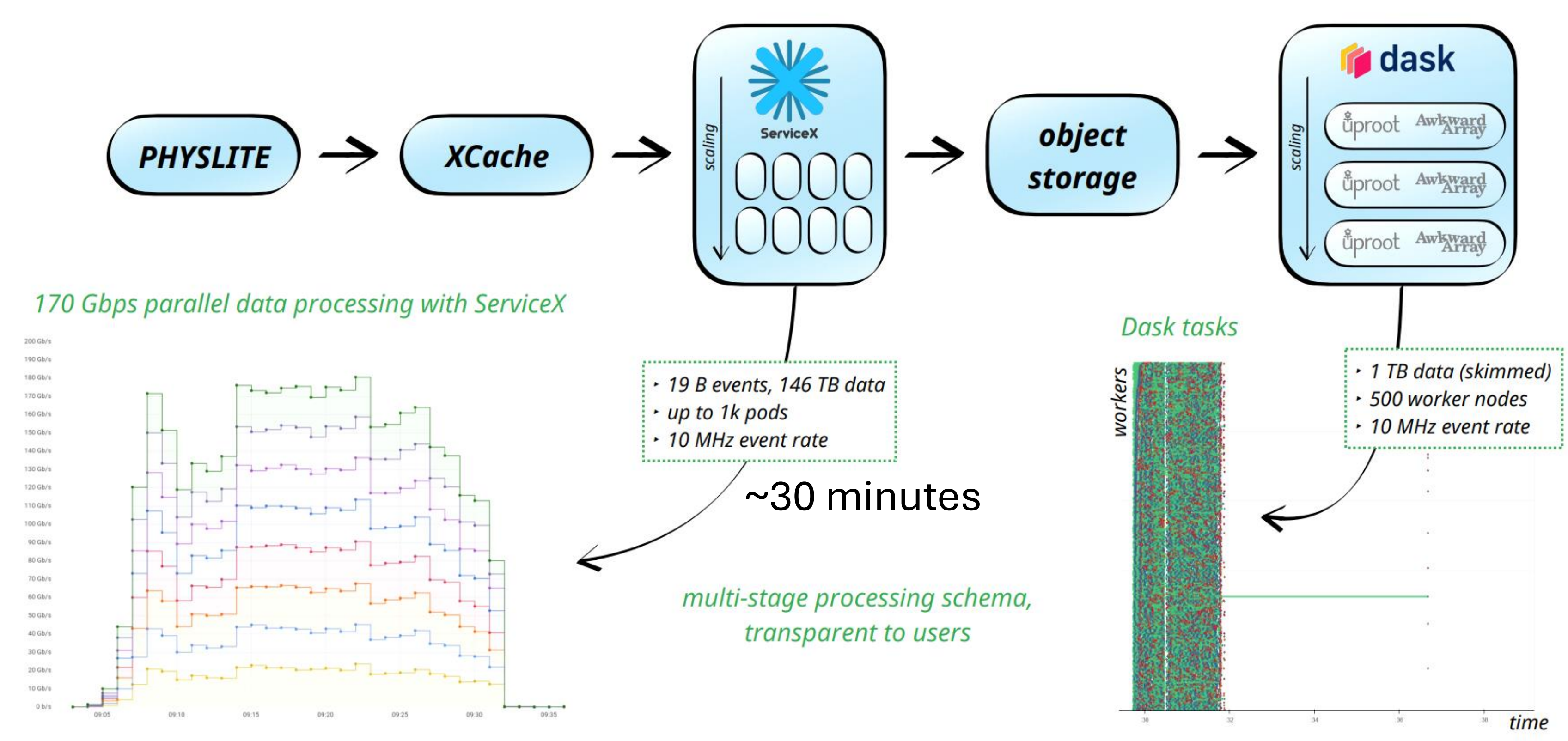
- RNTuple Support
    - Input for any uproot based transformer works now
    - In some cases requires experiment support first
    - Output works for uproot transformers, and converters for others coming.
- User login now uses keycloak and CERN!
    - No longer require an approval
    - We'd love to get rid of the access token if we could…
        - On Analysis Facilities?
- Making ServiceX Less Of A Black Box
    - It is a highly distributed system…
    - Depends on file delivery over the GRID…
    - Not easy, but far from impossible to write bad transform code…
    - Lots of work on improving error messages and reporting…
- Metadata
    - Improved command line interface and API
    - Access lots more information about transforms
    - MCP Plug-in for Ilija's talk!? (talk)

Use Cases and Scalability Testing

200 Gb/s Test

[2024 IRIS-HEP retreat]

(click graphic for talk slide)



170 Gbps parallel data processing with ServiceX

- *19 B events, 146 TB data*
- *up to 1k pods*
- *10 MHz event rate*

~30 minutes

multi-stage processing schema, transparent to users

- *1 TB data (skimmed)*
- *500 worker nodes*
- *10 MHz event rate*

- First time we consistently ran on large data sets
- Learned from many small issues encountered
  - Dropped internal messages
  - Race conditions
  - Configuration issues
  - Facility provisioning issues

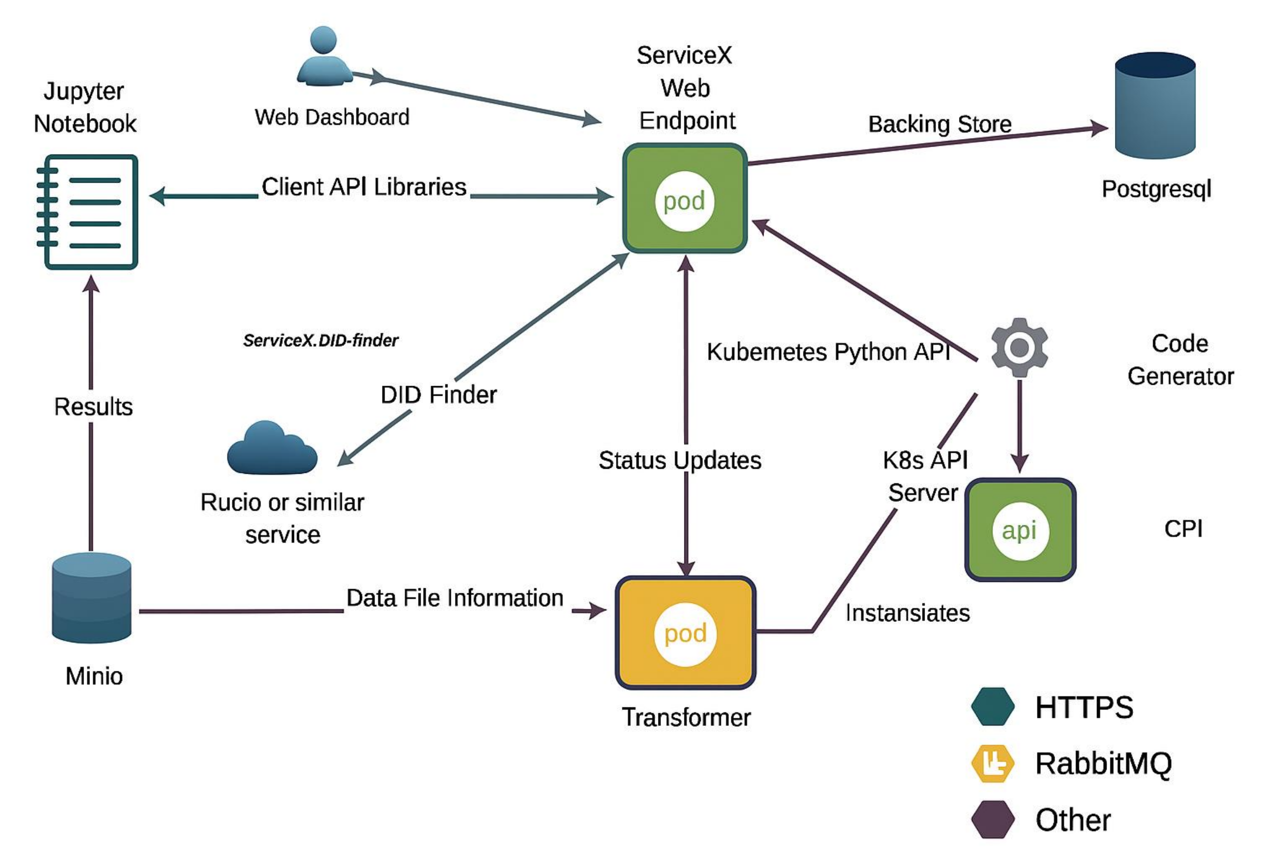
Thanks to many people in IRIS-HEP and the facilities and US OPS programs for contributing to this test!

Stability Improvements

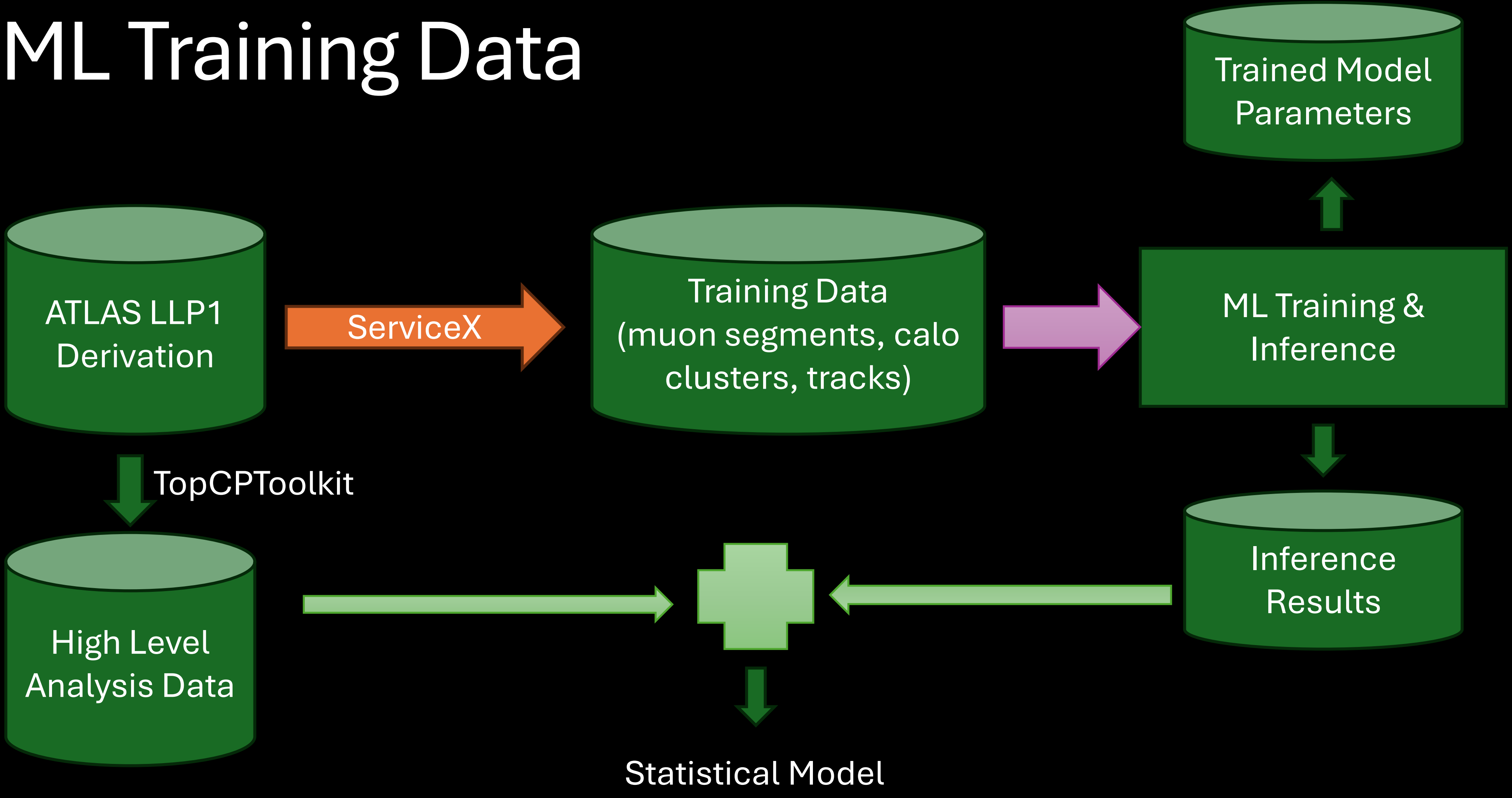
- A lot of careful internal work
- Converting the “easy” https connections to message queue for crucial messages
- Protecting the database with transactions
- Understanding bottlenecks in the database
- Decoupling the frontend REST API from backend operations
- Better resource sharing with rest of k8s cluster
- Improed XCache integration
- Now regularly test with a 50 TB dataset

We would not have seen any of these problems without the stress testing!

Designed for scalability



ML Training Data



Trained Model Parameters

ATLAS LLP1 Derivation

ServiceX

Training Data (muon segments, calo clusters, tracks)

ML Training & Inference

TopCPToolkit

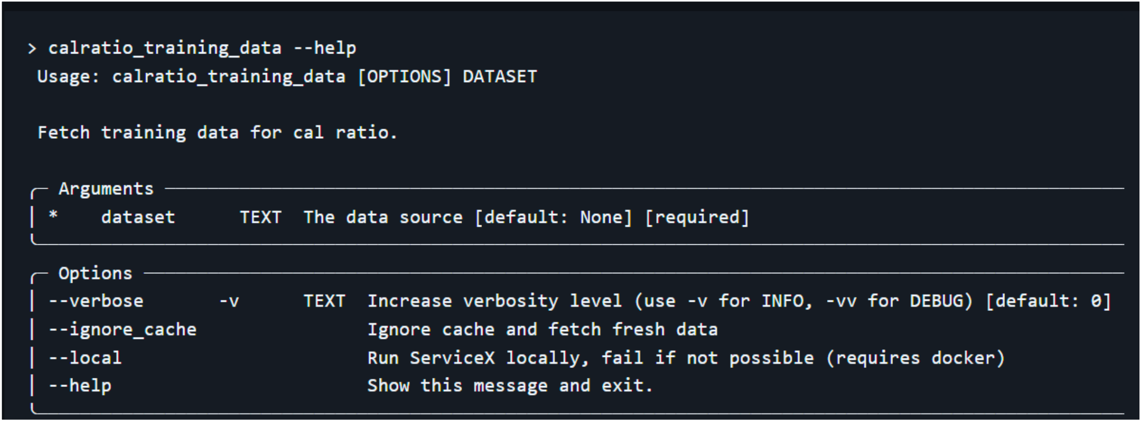
Inference Results

High Level Analysis Data

Statistical Model

Training Extractor

With ServiceX we have a hope of building a command-line slim/skim/delivery of the training data



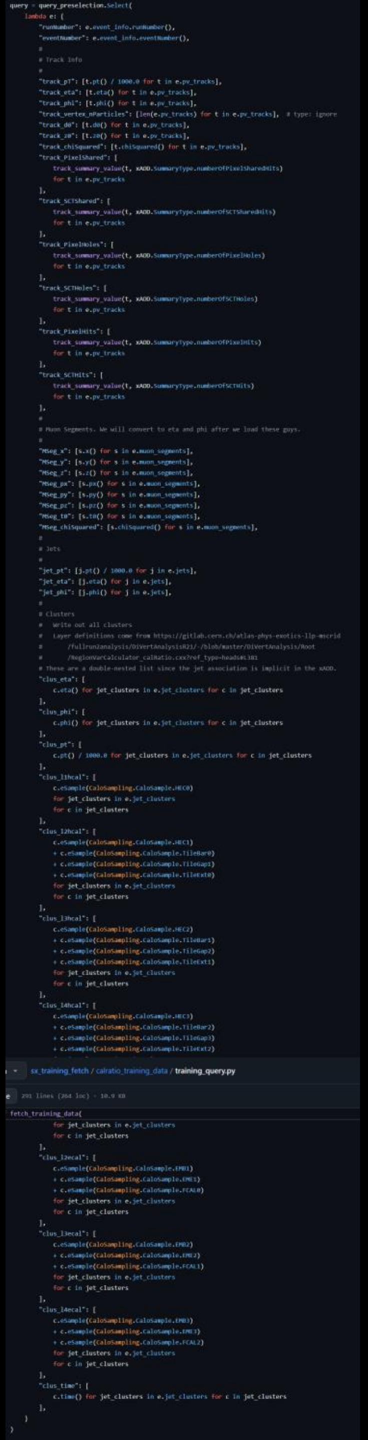
```
> calratio_training_data --help
 Usage: calratio_training_data [OPTIONS] DATASET

 Fetch training data for cal ratio.

╭─ Arguments ──────────────────────────────────────────────────────────────────────────────╮
│ *    dataset      TEXT  The data source [default: None] [required]                       │
╰──────────────────────────────────────────────────────────────────────────────────────────╯
╭─ Options ────────────────────────────────────────────────────────────────────────────────╮
│ --verbose       -v      TEXT  Increase verbosity level (use -v for INFO, -vv for DEBUG) [default: 0] │
│ --ignore_cache                Ignore cache and fetch fresh data                          │
│ --local                       Run ServiceX locally, fail if not possible (requires docker) │
│ --help                        Show this message and exit.                                │
╰──────────────────────────────────────────────────────────────────────────────────────────╯
```

Contains:

- Jet
- Tracks, including quality information (e.g. # of pixel hits)
- Muon segments
- Calorimeter Clusters

Code is built as a library, enabling Jupyter notebook studies

- Amazingly helpful for understanding missing data
- And developing more complex algorithms and checking them (e.g. jet-segment association)



```
query = query_preselection.Select(
    lambda e: {
        "runNumber": e.event_info.runNumber(),
        "eventNumber": e.event_info.eventNumber(),
        #
        # Track info
        #
        "track_pT": [t.pt() / 1000.0 for t in e.pv_tracks],
        "track_eta": [t.eta() for t in e.pv_tracks],
        "track_phi": [t.phi() for t in e.pv_tracks],
        "track_vertex_nParticles": [len(e.pv_tracks) for t in e.pv_tracks],  # type: ignore
        "track_d0": [t.d0() for t in e.pv_tracks],
        "track_z0": [t.z0() for t in e.pv_tracks],
        "track_chiSquared": [t.chiSquared() for t in e.pv_tracks],
        "track_PixelShared": [
            track_summary_value(t, xAOD.SummaryType.numberOfPixelSharedHits)
            for t in e.pv_tracks
        ],
        "track_SCTShared": [
            track_summary_value(t, xAOD.SummaryType.numberOfSCTSharedHits)
            for t in e.pv_tracks
        ],
        "track_PixelHoles": [
            track_summary_value(t, xAOD.SummaryType.numberOfPixelHoles)
            for t in e.pv_tracks
        ],
        "track_SCTHoles": [
            track_summary_value(t, xAOD.SummaryType.numberOfSCTHoles)
            for t in e.pv_tracks
        ],
        "track_PixelHits": [
            track_summary_value(t, xAOD.SummaryType.numberOfPixelHits)
            for t in e.pv_tracks
        ],
        "track_SCTHits": [
            track_summary_value(t, xAOD.SummaryType.numberOfSCTHits)
            for t in e.pv_tracks
        ],
        #
        # Muon Segments. We will convert to eta and phi after we load these guys.
        #
        "MSeg_x": [s.x() for s in e.muon_segments],
        "MSeg_y": [s.y() for s in e.muon_segments],
        "MSeg_z": [s.z() for s in e.muon_segments],
        "MSeg_px": [s.px() for s in e.muon_segments],
        "MSeg_py": [s.py() for s in e.muon_segments],
        "MSeg_pz": [s.pz() for s in e.muon_segments],
        "MSeg_t0": [s.t0() for s in e.muon_segments],
        "MSeg_chiSquared": [s.chiSquared() for s in e.muon_segments],
        #
        # Jets
        #
        "jet_pt": [j.pt() / 1000.0 for j in e.jets],
        "jet_eta": [j.eta() for j in e.jets],
        "jet_phi": [j.phi() for j in e.jets],
        #
        # Clusters
        #   Write out all clusters
        #   Layer definitions come from https://gitlab.cern.ch/atlas-phys-exotics-llp-mscrid
        #       /fullrun2analysis/DiVertAnalysisR21/-/blob/master/DiVertAnalysis/Root
        #       /RegionVarCalculator_calRatio.cxx?ref_type=heads#L381
        # These are a double-nested list since the jet association is implicit in the xAOD.
        "clus_eta": [
            c.eta() for jet_clusters in e.jet_clusters for c in jet_clusters
        ],
        "clus_phi": [
            c.phi() for jet_clusters in e.jet_clusters for c in jet_clusters
        ],
        "clus_pt": [
            c.pt() / 1000.0 for jet_clusters in e.jet_clusters for c in jet_clusters
        ],
        "clus_l1hcal": [
            c.eSample(CaloSampling.CaloSample.HEC0)
            for jet_clusters in e.jet_clusters
            for c in jet_clusters
        ],
        "clus_l2hcal": [
            c.eSample(CaloSampling.CaloSample.HEC1)
            + c.eSample(CaloSampling.CaloSample.TileBar0)
            + c.eSample(CaloSampling.CaloSample.TileGap1)
            + c.eSample(CaloSampling.CaloSample.TileExt0)
            for jet_clusters in e.jet_clusters
            for c in jet_clusters
        ],
        "clus_l3hcal": [
            c.eSample(CaloSampling.CaloSample.HEC2)
            + c.eSample(CaloSampling.CaloSample.TileBar1)
            + c.eSample(CaloSampling.CaloSample.TileGap2)
            + c.eSample(CaloSampling.CaloSample.TileExt1)
            for jet_clusters in e.jet_clusters
            for c in jet_clusters
        ],
        "clus_l4hcal": [
            c.eSample(CaloSampling.CaloSample.HEC3)
            + c.eSample(CaloSampling.CaloSample.TileBar2)
            + c.eSample(CaloSampling.CaloSample.TileGap3)
            + c.eSample(CaloSampling.CaloSample.TileExt2)
```

sx_training_fetch / calratio_training_data / training_query.py

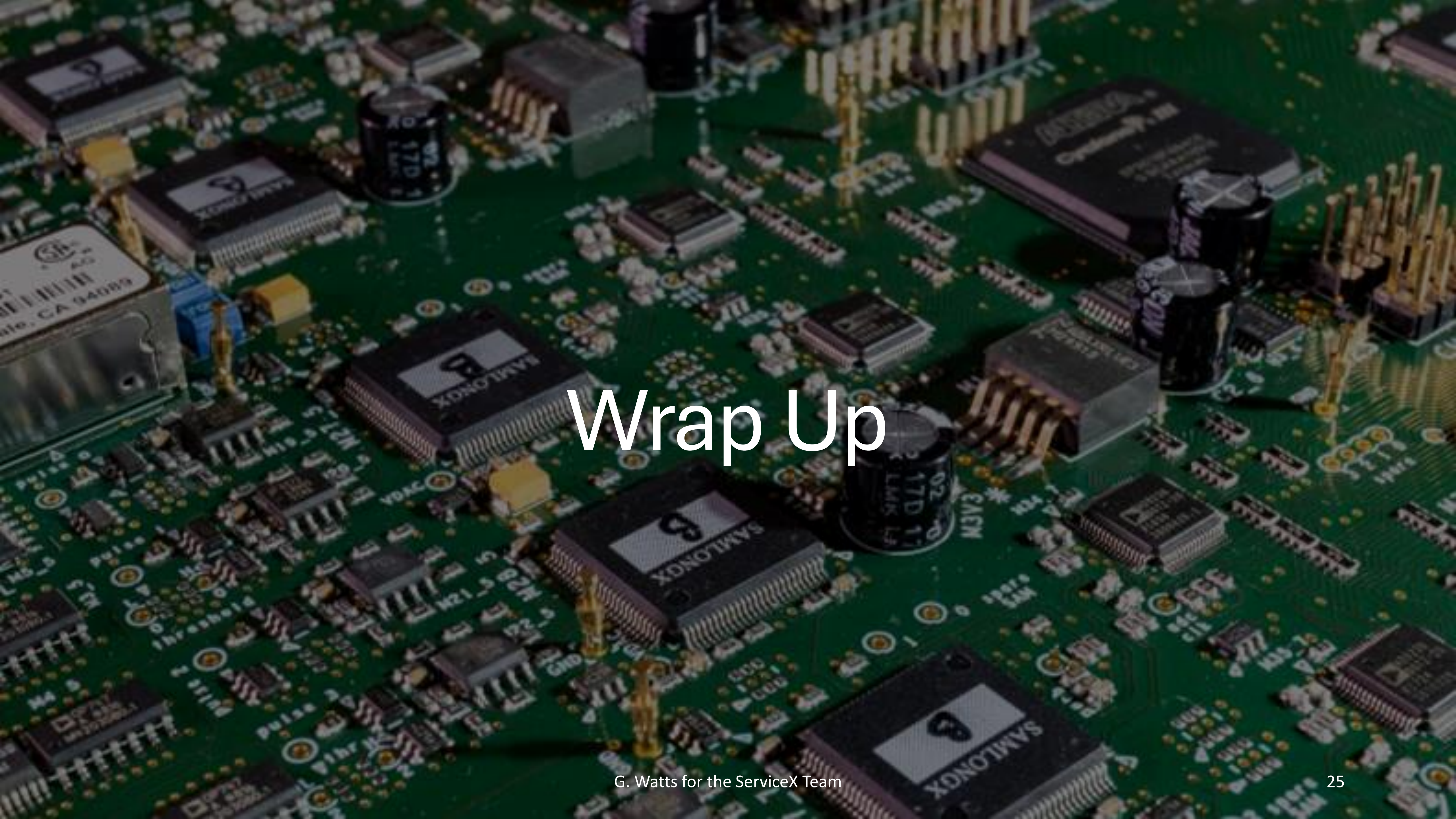
291 lines (264 loc) · 10.9 KB

```
fetch_training_data(
            for jet_clusters in e.jet_clusters
            for c in jet_clusters
        ],
        "clus_l2ecal": [
            c.eSample(CaloSampling.CaloSample.EMB1)
            + c.eSample(CaloSampling.CaloSample.EME1)
            + c.eSample(CaloSampling.CaloSample.FCAL0)
            for jet_clusters in e.jet_clusters
            for c in jet_clusters
        ],
        "clus_l3ecal": [
            c.eSample(CaloSampling.CaloSample.EMB2)
            + c.eSample(CaloSampling.CaloSample.EME2)
            + c.eSample(CaloSampling.CaloSample.FCAL1)
            for jet_clusters in e.jet_clusters
            for c in jet_clusters
        ],
        "clus_l4ecal": [
            c.eSample(CaloSampling.CaloSample.EMB3)
            + c.eSample(CaloSampling.CaloSample.EME3)
            + c.eSample(CaloSampling.CaloSample.FCAL2)
            for jet_clusters in e.jet_clusters
            for c in jet_clusters
        ],
        "clus_time": [
            c.time() for jet_clusters in e.jet_clusters for c in jet_clusters
        ],
    }
)
```

A Few Other Slim & Skim Use Cases

- Analysis Selection in $t\bar{t}\mathrm{H}$:
    - Start with large ntuple using Panda and standard ATLAS tools, stored on Grid RSEs.
    - The prototyping of the analysis requires a small fraction of the original dataset (due to not using systematic branches)
    - The final result has tuned signal regions etc.
    - Achieve a very large reduction of file size using ServiceX, such that we do not need to download the entire ntuple dataset.
- Various People, small studies:
    - Small exploratory studies with Grid datasets where, again, we only want a few branches from PHYSLITE or some ntuple. We can prepare the skim for a standard ATLAS MC sample in minutes in the best case, generally less than an hour.

- Access to certain ATLAS datasets with unusual data structures not handled by standard ntupling tools but handled by the FuncADL xAOD backend.
- Also aware of people setting up to use it to study compression settings for analysis!



Wrap Up

Conclusions

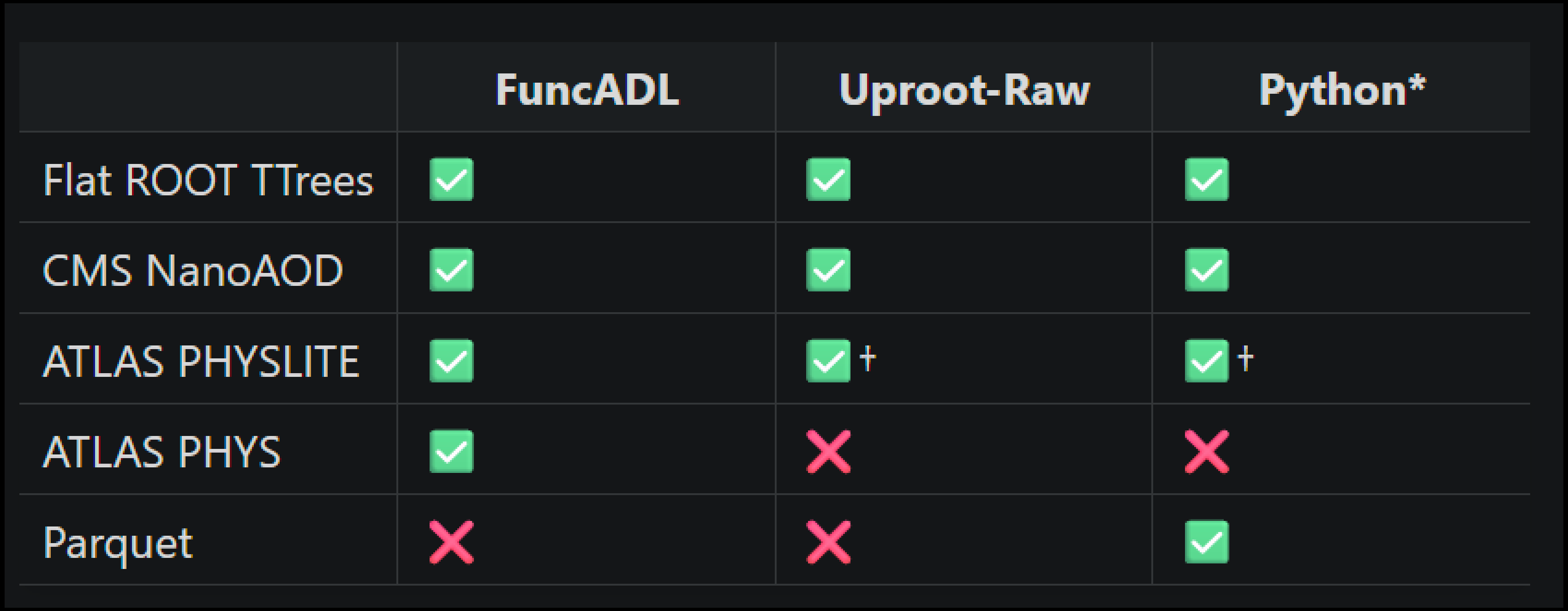
- Roadmap
  - Publicly published on github
  - But better idea by joining the #servicex channel in the iris-hep slack.
  - And the ServiceX 2-week meeting
- Huge amount of work on usability, scalability, and robustness after the 200 gpbs test
  - Looking forward to the next edition!
  - Current tests show system is quite robust
- Usage Documentation
- Scalability
  - Uses k8s resources efficiently
- Used by a small number of analyses and experiments
  - Working hard with new users
  - And improving our documentation
- Use cases are now driving new features
- The core of the system is stable
  - Concentrating development on things like new transformers
  - Supporting new use cases

Backup

Input Matrix…

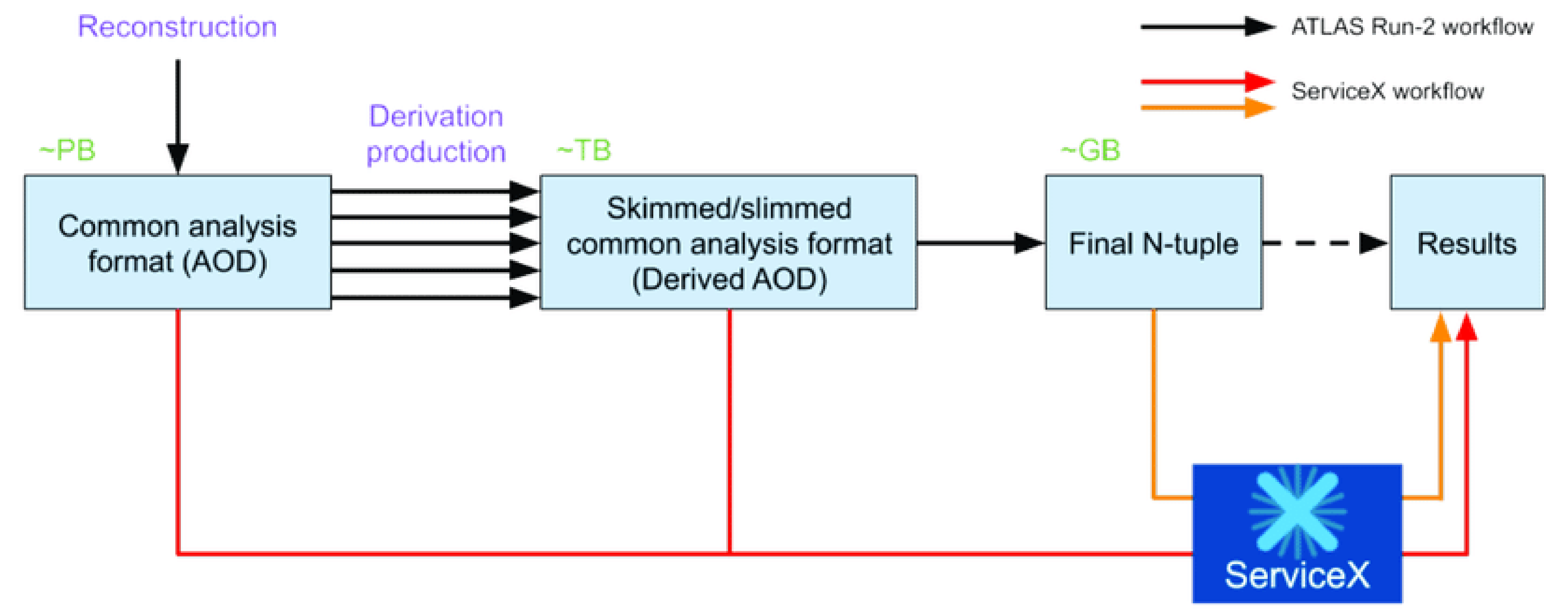
We'd love input to be agnostic of processing query

But inputs are tied to specific software frameworks…

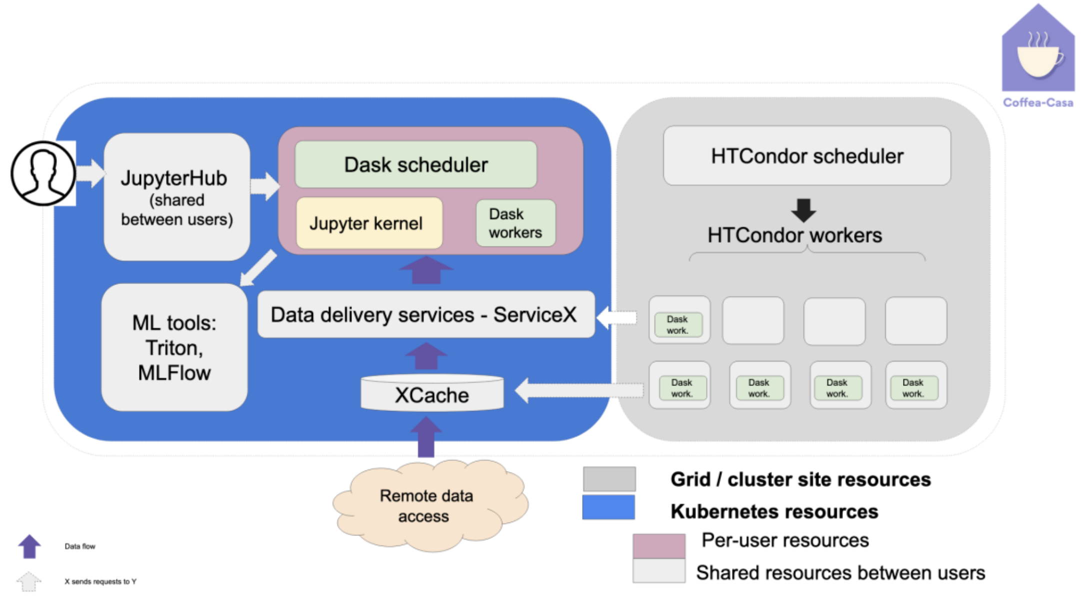


	FuncADL	Uproot-Raw	Python*
Flat ROOT TTrees	✅	✅	✅
CMS NanoAOD	✅	✅	✅
ATLAS PHYSLITE	✅	✅ †	✅ †
ATLAS PHYS	✅	❌	❌
Parquet	❌	❌	✅

Its Place In Analysis



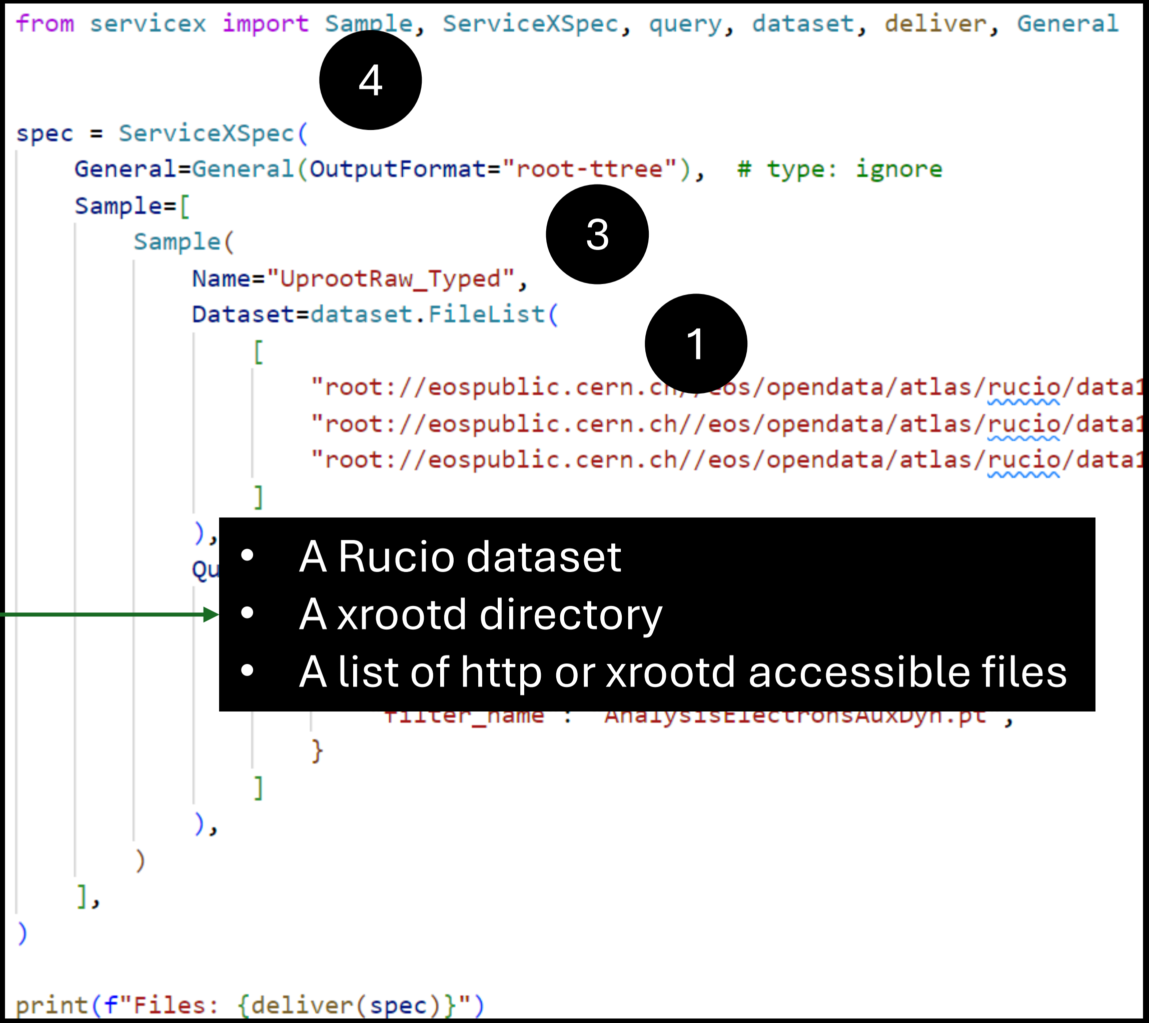
Analysis Facilities & Computing in HEP



- Designed as a service that can be made available in a AF or standalone.
  - Kubernetes based, helm chart published, site-personalization via values.yaml.
- Enhances an AF with temporary high speed parallel storage for skimmed data sets
- The "manages file delivery, processing (transformation), and output dataset" is a lot like the GRID's mission!
  - But this is built for small fast transformations
  - Compute heavy workloads are not efficient use of ServiceX
  - This is *not* a GRID replacement
  - But it does depend on the GRID for data delivery!!

Components of a Transformation Request

1. The Input Data
2. The transformation to apply
3. The Output Format
4. How the transformation is specified

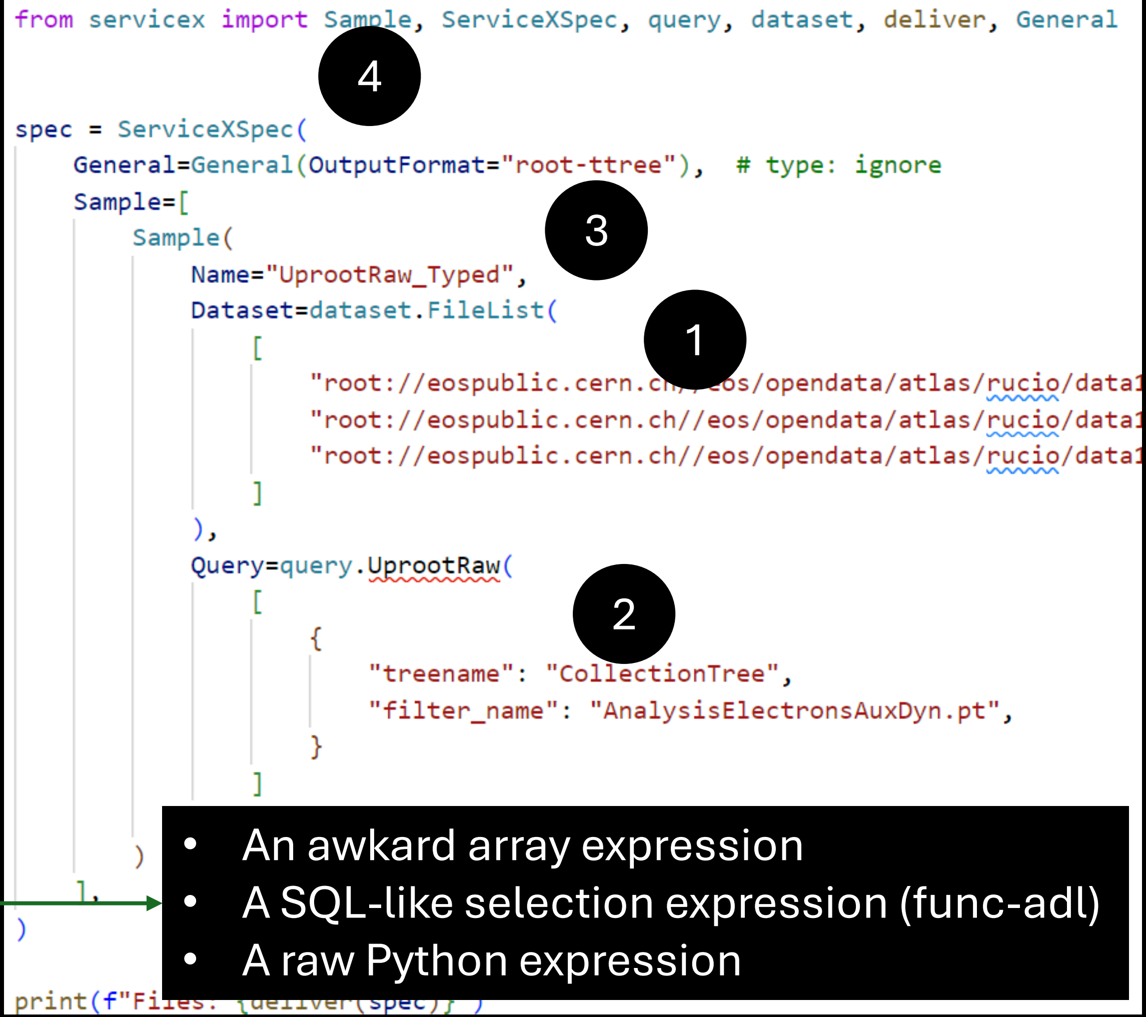


```
from servicex import Sample, ServiceXSpec, query, dataset, deliver, General


spec = ServiceXSpec(
    General=General(OutputFormat="root-ttree"),  # type: ignore
    Sample=[
        Sample(
            Name="UprootRaw_Typed",
            Dataset=dataset.FileList(
                [
                    "root://eospublic.cern.ch//eos/opendata/atlas/rucio/data1
                    "root://eospublic.cern.ch//eos/opendata/atlas/rucio/data1
                    "root://eospublic.cern.ch//eos/opendata/atlas/rucio/data1
                ]
            ),
            Qu
                    "filter_name": "AnalysisElectronsAuxDyn.pt",
                }
            ]
            ),
        )
    ],
)

print(f"Files: {deliver(spec)}")
```

- A Rucio dataset
- A xrootd directory
- A list of http or xrootd accessible files

Components of a Transformation Request

1. The Input Data
2. The transformation to apply
3. The Output Format
4. How the transformation is specified

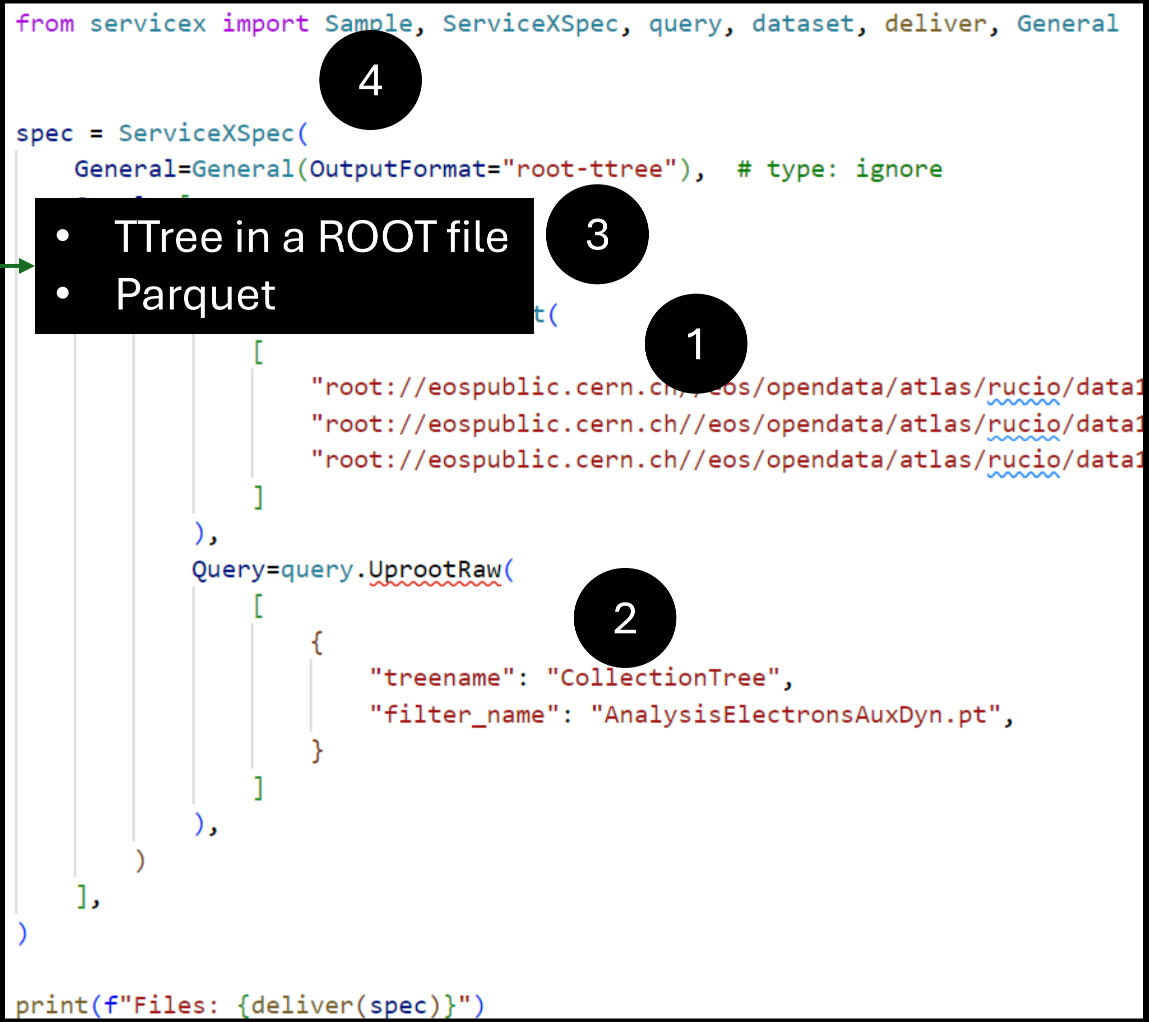


```python
from servicex import Sample, ServiceXSpec, query, dataset, deliver, General


spec = ServiceXSpec(
    General=General(OutputFormat="root-ttree"),  # type: ignore
    Sample=[
        Sample(
            Name="UprootRaw_Typed",
            Dataset=dataset.FileList(
                [
                    "root://eospublic.cern.ch//eos/opendata/atlas/rucio/data1
                    "root://eospublic.cern.ch//eos/opendata/atlas/rucio/data1
                    "root://eospublic.cern.ch//eos/opendata/atlas/rucio/data1
                ]
            ),
            Query=query.UprootRaw(
                [
                    {
                        "treename": "CollectionTree",
                        "filter_name": "AnalysisElectronsAuxDyn.pt",
                    }
                ]
        )
    ],
)

print(f"Files: {deliver(spec)}")
```

- An awkard array expression
- A SQL-like selection expression (func-adl)
- A raw Python expression

Components of a Transformation Request

1. The Input Data
2. The transformation to apply
3. The Output Format
4. How the transformation is specified

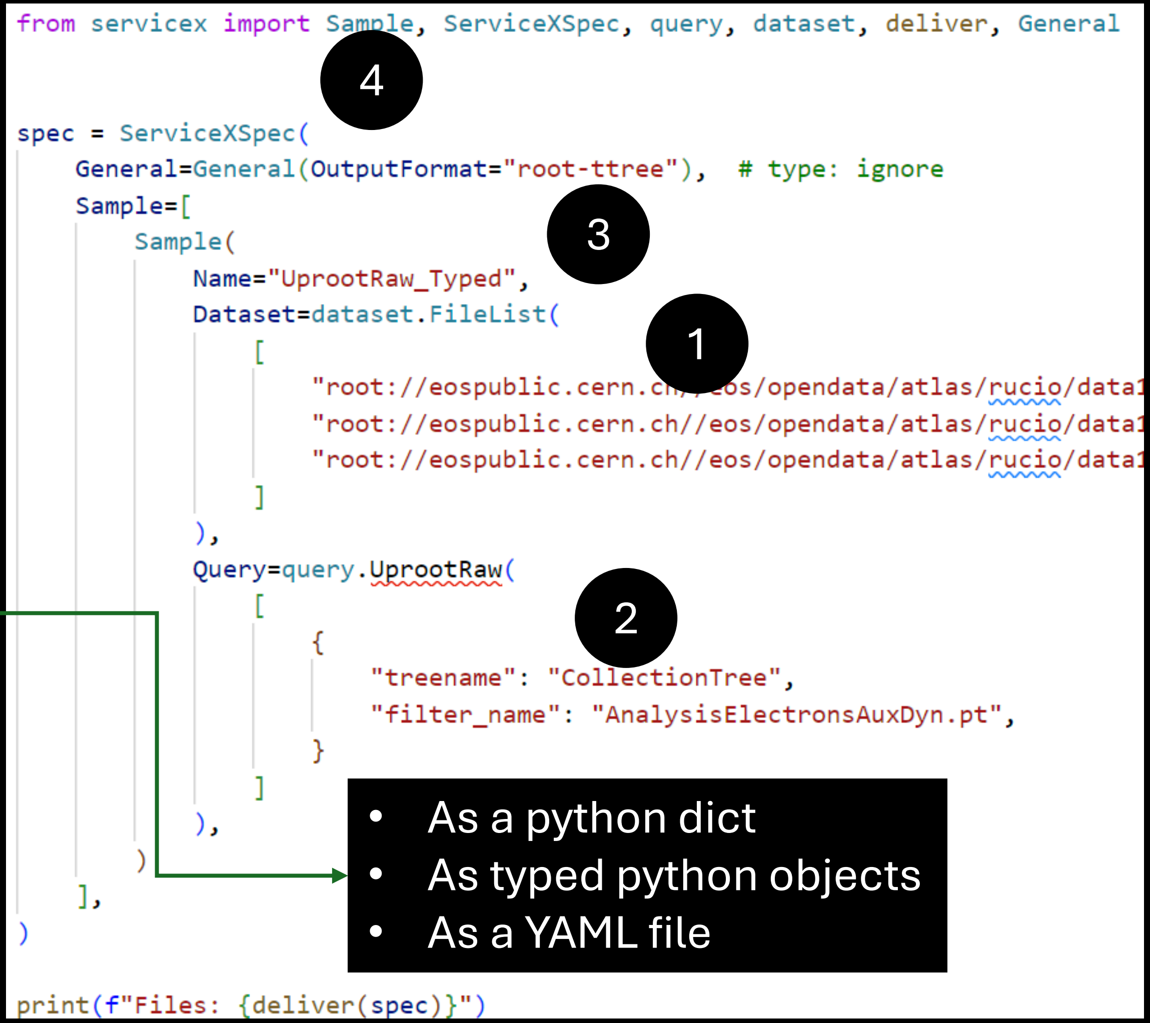


```python
from servicex import Sample, ServiceXSpec, query, dataset, deliver, General


spec = ServiceXSpec(
    General=General(OutputFormat="root-ttree"),  # type: ignore



                                   t(
                [
                    "root://eospublic.cern.ch//eos/opendata/atlas/rucio/data1
                    "root://eospublic.cern.ch//eos/opendata/atlas/rucio/data1
                    "root://eospublic.cern.ch//eos/opendata/atlas/rucio/data1
                ]
            ),
            Query=query.UprootRaw(
                [
                    {
                        "treename": "CollectionTree",
                        "filter_name": "AnalysisElectronsAuxDyn.pt",
                    }
                ]
            ),
        )
    ],
)

print(f"Files: {deliver(spec)}")
```

- TTree in a ROOT file
- Parquet

Components of a Transformation Request

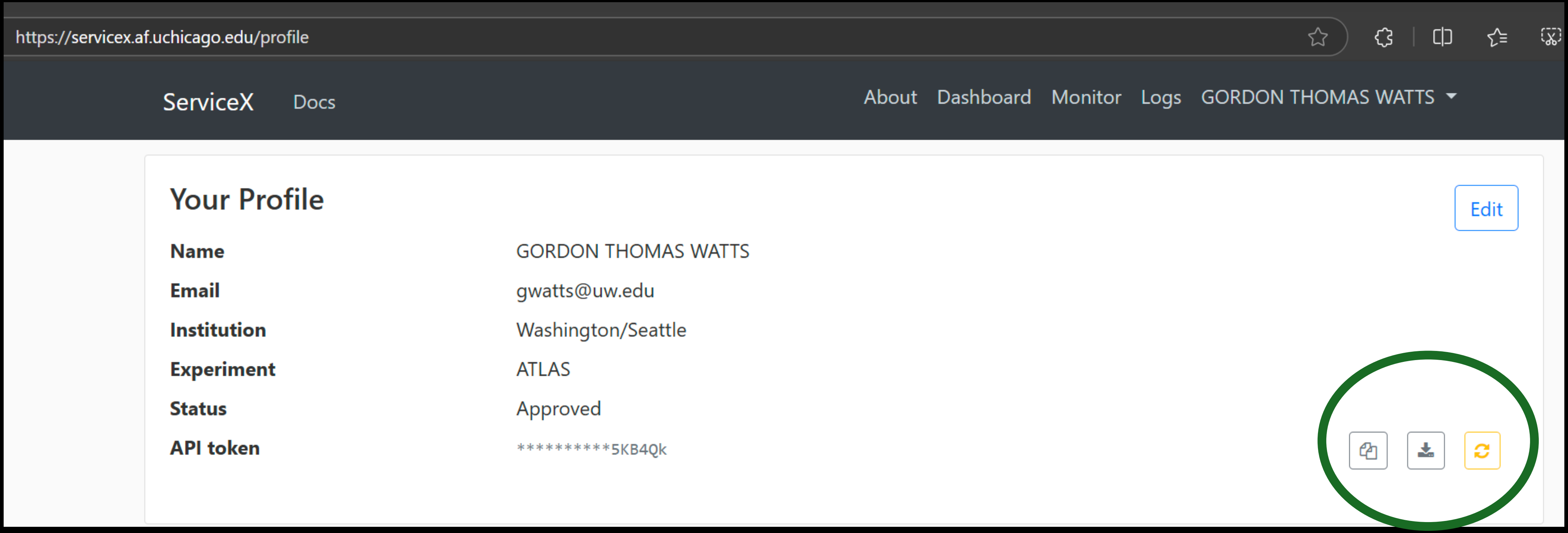
1. The Input Data
2. The transformation to apply
3. The Output Format
4. How the transformation is specified



```python
from servicex import Sample, ServiceXSpec, query, dataset, deliver, General


spec = ServiceXSpec(
    General=General(OutputFormat="root-ttree"),  # type: ignore
    Sample=[
        Sample(
            Name="UprootRaw_Typed",
            Dataset=dataset.FileList(
                [
                    "root://eospublic.cern.ch//eos/opendata/atlas/rucio/data1
                    "root://eospublic.cern.ch//eos/opendata/atlas/rucio/data1
                    "root://eospublic.cern.ch//eos/opendata/atlas/rucio/data1
                ]
            ),
            Query=query.UprootRaw(
                [
                    {
                        "treename": "CollectionTree",
                        "filter_name": "AnalysisElectronsAuxDyn.pt",
                    }
                ]
            ),
        )
    ],
)

print(f"Files: {deliver(spec)}")
```

- As a python dict
- As typed python objects
- As a YAML file

First… Don’t forget your access token!

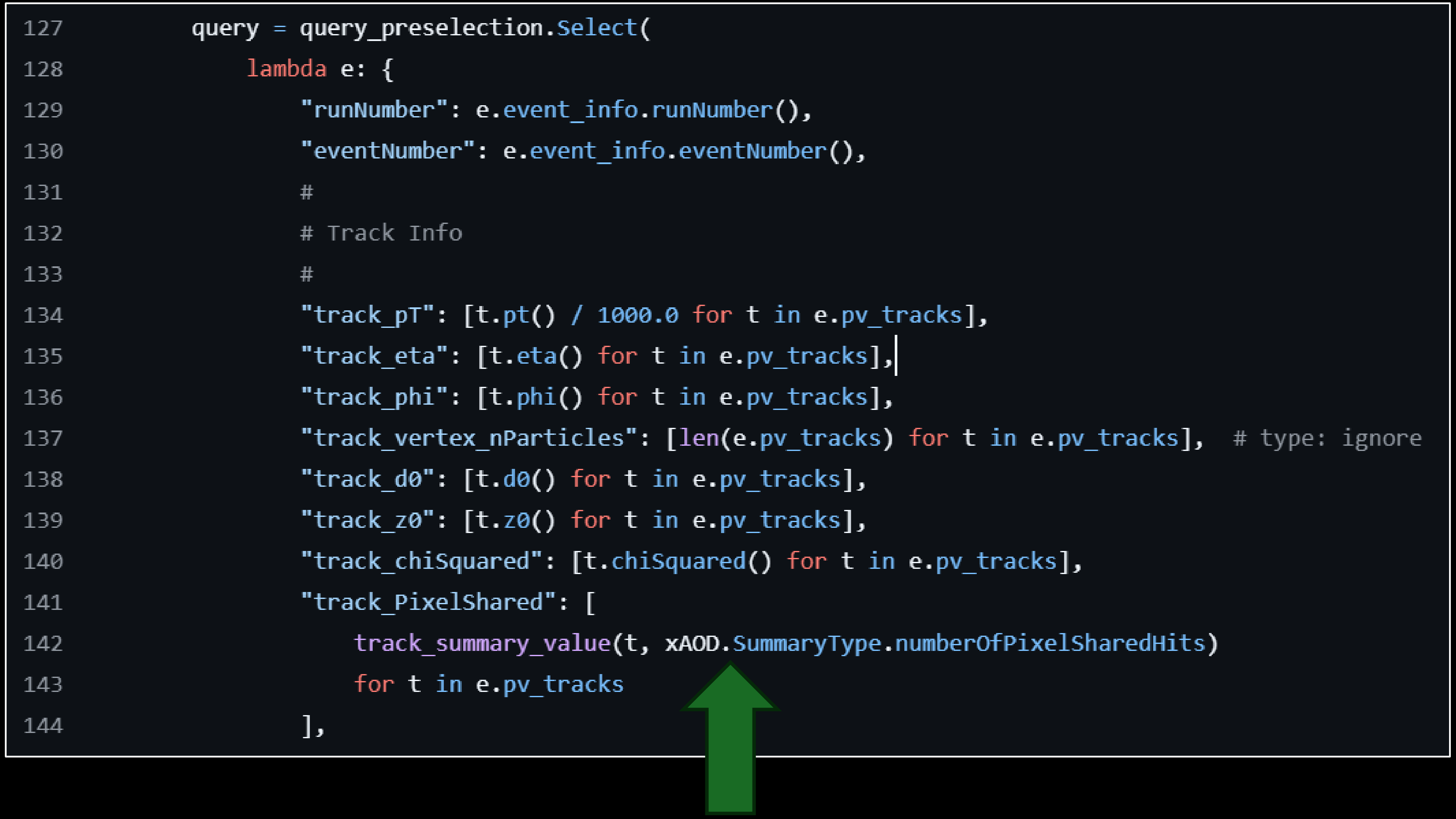


Your Profile

Name GORDON THOMAS WATTS
Email gwatts@uw.edu
Institution Washington/Seattle
Experiment ATLAS
Status Approved
API token `**********5KB4Qk`

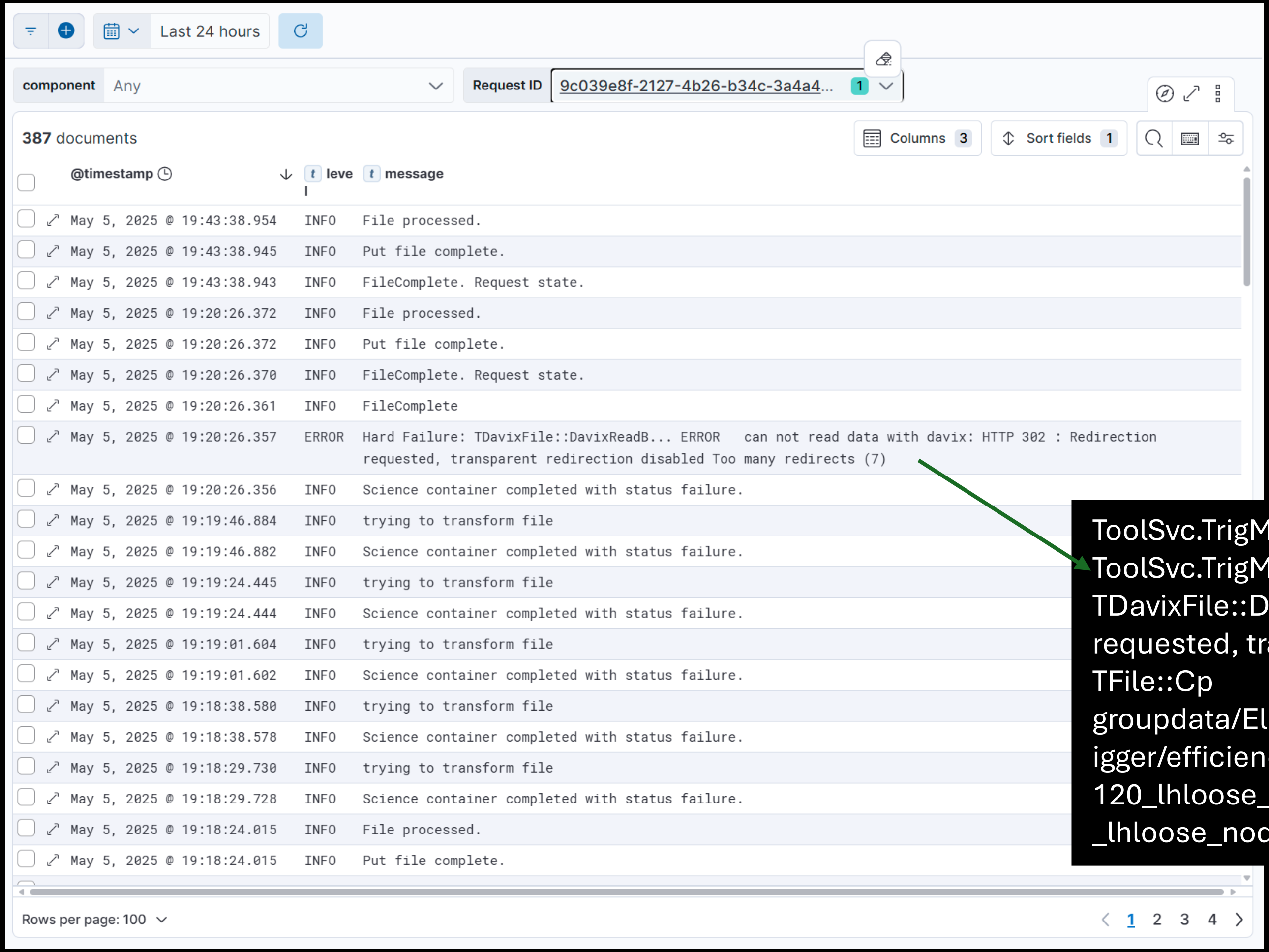
- Place the servicex.yaml (or .servicex) file in your home directory, or in the root directory of your analysis
- Do not check into git!!!

And use it in your expression…



```
        query = query_preselection.Select(
            lambda e: {
                "runNumber": e.event_info.runNumber(),
                "eventNumber": e.event_info.eventNumber(),
                #
                # Track Info
                #
                "track_pT": [t.pt() / 1000.0 for t in e.pv_tracks],
                "track_eta": [t.eta() for t in e.pv_tracks],
                "track_phi": [t.phi() for t in e.pv_tracks],
                "track_vertex_nParticles": [len(e.pv_tracks) for t in e.pv_tracks],  # type: ignore
                "track_d0": [t.d0() for t in e.pv_tracks],
                "track_z0": [t.z0() for t in e.pv_tracks],
                "track_chiSquared": [t.chiSquared() for t in e.pv_tracks],
                "track_PixelShared": [
                    track_summary_value(t, xAOD.SummaryType.numberOfPixelSharedHits)
                    for t in e.pv_tracks
                ],
```

Errors are still hard…



Last 24 hours

component Any

Request ID 9c039e8f-2127-4b26-b34c-3a4a4… 1

387 documents

Columns 3 | Sort fields 1

```
@timestamp                  level  message
May 5, 2025 @ 19:43:38.954  INFO   File processed.
May 5, 2025 @ 19:43:38.945  INFO   Put file complete.
May 5, 2025 @ 19:43:38.943  INFO   FileComplete. Request state.
May 5, 2025 @ 19:20:26.372  INFO   File processed.
May 5, 2025 @ 19:20:26.372  INFO   Put file complete.
May 5, 2025 @ 19:20:26.370  INFO   FileComplete. Request state.
May 5, 2025 @ 19:20:26.361  INFO   FileComplete
May 5, 2025 @ 19:20:26.357  ERROR  Hard Failure: TDavixFile::DavixReadB... ERROR   can not read data with davix: HTTP 302 : Redirection
                                   requested, transparent redirection disabled Too many redirects (7)
May 5, 2025 @ 19:20:26.356  INFO   Science container completed with status failure.
May 5, 2025 @ 19:19:46.884  INFO   trying to transform file
May 5, 2025 @ 19:19:46.882  INFO   Science container completed with status failure.
May 5, 2025 @ 19:19:24.445  INFO   trying to transform file
May 5, 2025 @ 19:19:24.444  INFO   Science container completed with status failure.
May 5, 2025 @ 19:19:01.604  INFO   trying to transform file
May 5, 2025 @ 19:19:01.602  INFO   Science container completed with status failure.
May 5, 2025 @ 19:18:38.580  INFO   trying to transform file
May 5, 2025 @ 19:18:38.578  INFO   Science container completed with status failure.
May 5, 2025 @ 19:18:29.730  INFO   trying to transform file
May 5, 2025 @ 19:18:29.728  INFO   Science container completed with status failure.
May 5, 2025 @ 19:18:24.015  INFO   File processed.
May 5, 2025 @ 19:18:24.015  INFO   Put file complete.
```

Rows per page: 100

Using elastic search

- Captures everything
- User gets links to errors
- Transformer infrastructure can use regular expressions to find most likely cause (e.g. "ERROR")

```
ToolSvc.TrigMatchingTool INFO    initializing ToolSvc.TrigMatchingTool
ToolSvc.TrigMatchingTool INFO    Remap broken links? 0
TDavixFile::DavixReadB... ERROR   can not read data with davix: HTTP 302 : Redirection
requested, transparent redirection disabled Too many redirects (7)
TFile::Cp                 ERROR   cannot read from source file http://cern.ch/atlas-
groupdata/ElectronEfficiencyCorrection/2015_2018/rel21.2/Precision_Summer2020_v1/tr
igger/efficiencySF.SINGLE_E_2015_e24_lhmedium_L1EM20VH_OR_e60_lhmedium_OR_e
120_lhloose_2016_2018_e26_lhtight_nod0_ivarloose_OR_e60_lhmedium_nod0_OR_e140
_lhloose_nod0.TightLLH_d0z0_v13_isolTight.root. readsize=81971 read=0 readop=1
```

(at the end of 18K lines of log file output…)

Running At Scale

At Scale means running on data

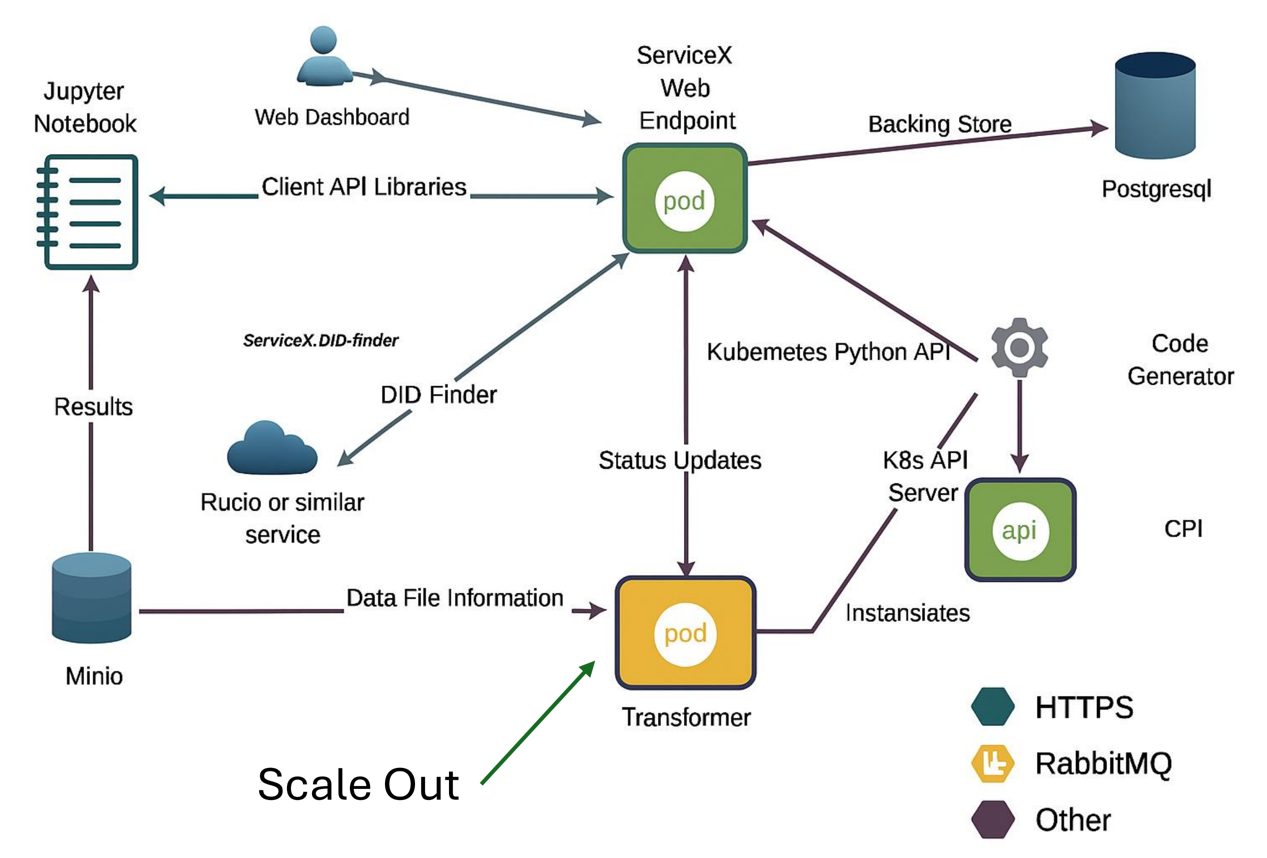
- Run 2 dataset contains ~66K files
- And many many MC datasets

Requires robust handling and communication inside ServiceX

- RabbitMQ for message passing on crucial pathways
- https, on a private backbone, for less important messages.

Database is carefully protected & transactional

Some of this was hard won insight – dropped messages & retries inside k8s network!


Jupyter Notebook
Web Dashboard
ServiceX Web Endpoint
pod
Backing Store
Postgresql
Client API Libraries
ServiceX.DID-finder
DID Finder
Kubemetes Python API
Code Generator
Results
Rucio or similar service
Status Updates
K8s API Server
api
CPI
Minio
Data File Information
pod
Instansiates
Transformer
Scale Out
HTTPS
RabbitMQ
Other