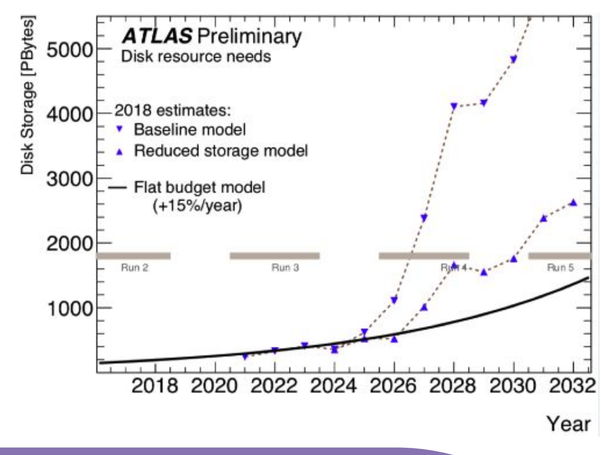
# Functional Analysis Description Language (FuncADL)

Team: Mason Proffitt, Emma Torró, Gordon Watts
Institution: University of Washington, Seattle

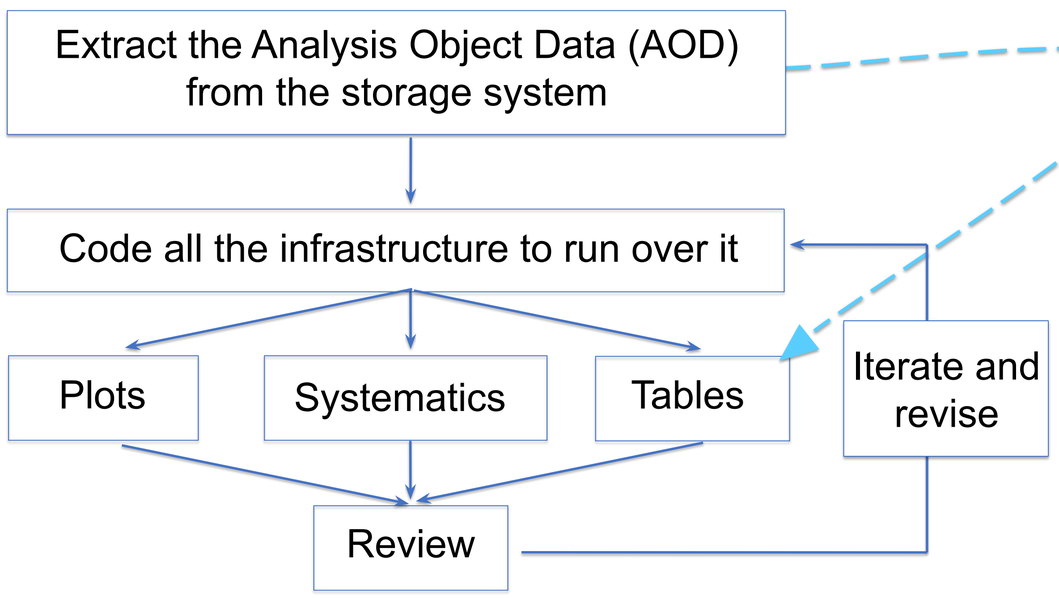
## Introduction

- High Luminosity LHC (HL-LHC) will collect ~10 times as much data as all the LHC runs in 2010-2023
  - New data formats, but we can't rewrite our analysis code for every new file format
- The challenge that we're trying to address:
  - New query-based analysis interface: a functional, declarative analysis language (within Python)
  - Main GitHub repository: https://github.com/iris-hep/func_adl




## Functional ADL

In a large HEP experiment, each analysis team must:




**Analysis language**: goes from AODs to plots and tables

**Declarative analysis:**

- The physicist specifies *what* they want from the data rather than *how* to implement it procedurally
- Columnar operations—no explicit loops

```
In [3]: %%time
        f = EventDataSet(fname)
        events = f.AsATLASEvents()
        Wall time: 0 ns
```
Get input file as ATLAS xAOD

```
In [4]: %%time
        event_info = events \
                    .Select("lambda e: (e.EventInfo('EventInfo'), e.Jets('AntiKt4EMTopoJets'), e.Tracks('InDetTrackParticles').Where(lambda t: t.pt() > 1000.0))")
```
Get electrons and tracks with pt>1 GeV

```
In [7]: jet_info = event_info \
                    .SelectMany('lambda e1: e1[1].Select(lambda j: (e1[0],j,e1[2].Where(lambda t: DeltaR(t.eta(), t.phi(), j.eta(), j.phi())< 0.2)))')
```
SelectMany function changes sequence from event into jets

```
In [8]: jet_data = jet_info \
                    .Select('lambda jinfo: (jinfo[0].runNumber(), jinfo[0].eventNumber(), jinfo[1].pt()/1000.0, jinfo[1].eta(), jinfo[2].Count())')
```
Generate the tuple of data we want to write out

```
In [9]: jet_filtered = jet_data \
                      .Where('lambda jinfo1: jinfo1[2] > 40.0')
```
Keep only jets with pt > 40 GeV

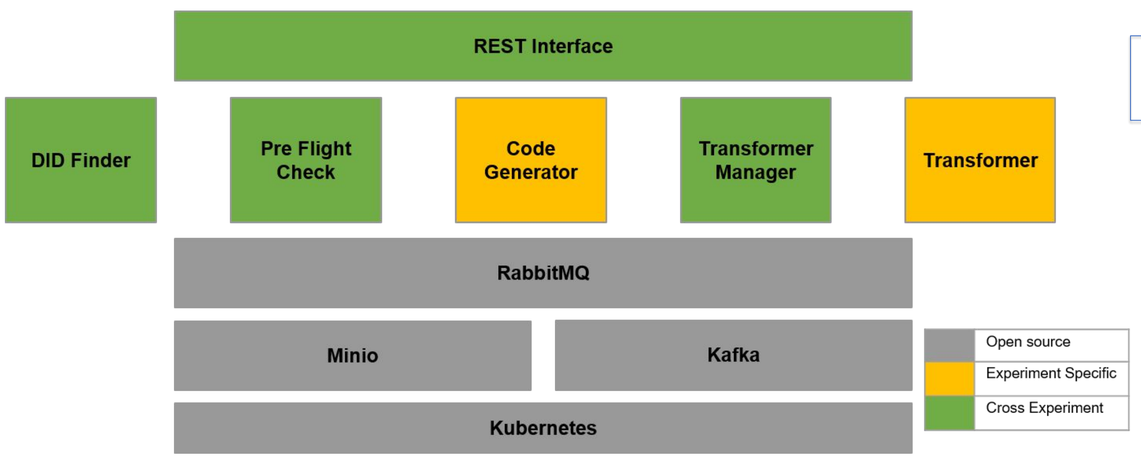
```
In [10]: %%time
         training_df = jet_filtered.AsPandasDF(columns=['Run', 'Event', 'JetPt', 'JetEta', 'nTracks'])
         Wall time: 0 ns
```
Turn it into a pandas data frame

```
In [11]: %%time
         df = training_df.value()
         Wall time: 38.6 s
```
Execute

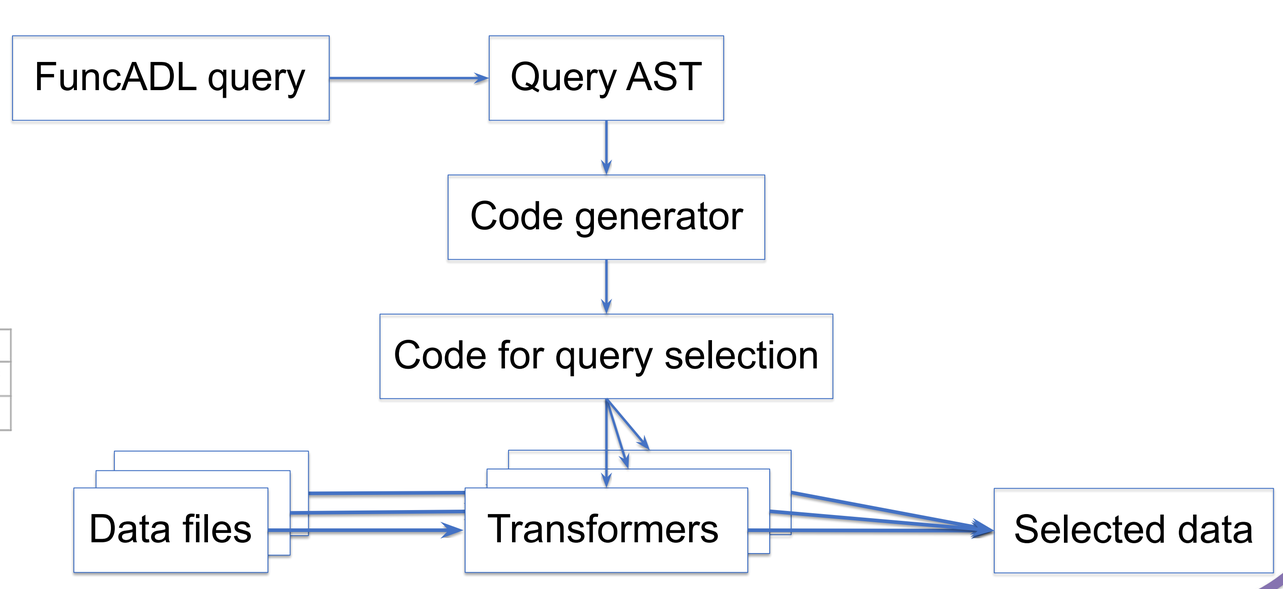
- Developing declarative analysis language based on queries
- Can be run on different file formats; currently implemented for:
  - xAOD (ATLAS)
  - Flat ROOT TTrees, using either uproot or RDataFrame
- Model being used in a full Run 2 analysis in ATLAS
- Benefits for the LHC community:
  - Analysis preservation that goes beyond the lifetimes of experiments
  - Facilitating the abstraction, design, combination, interpretation, and overall communication of the contents of LHC analyses

## Integration with ServiceX

- ServiceX is a suite of services developed by IRIS-HEP that provides high-performance data delivery for analysis
- Low-level extraction and selection of data on different file formats is done by FuncADL
- This is accomplished by two types of services: code generators and transformers
  - Data query from user is passed to code generator as an abstract syntax tree (AST)
  - Code generator translates the AST into selection code appropriate for the underlying file format
  - Transformers run the generated code and return the selected data
- Separation of these steps allows parallelizing transformation on large datasets with common generated code




Components of ServiceX. FuncADL provides the backend implementations for the services in yellow.




This project is supported by National Science Foundation under Cooperative Agreement OAC-1836650. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.