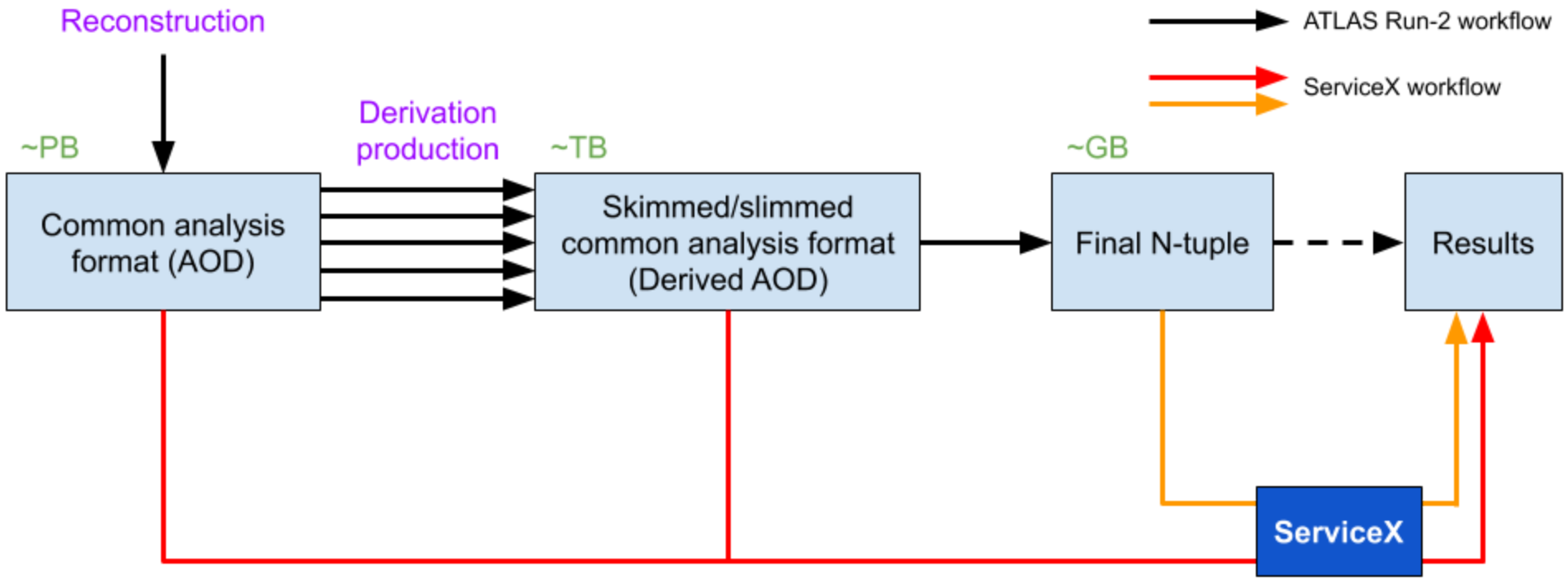
ServiceX: On-Demand Data Transformation and Delivery for the Present and HL-LHC Era

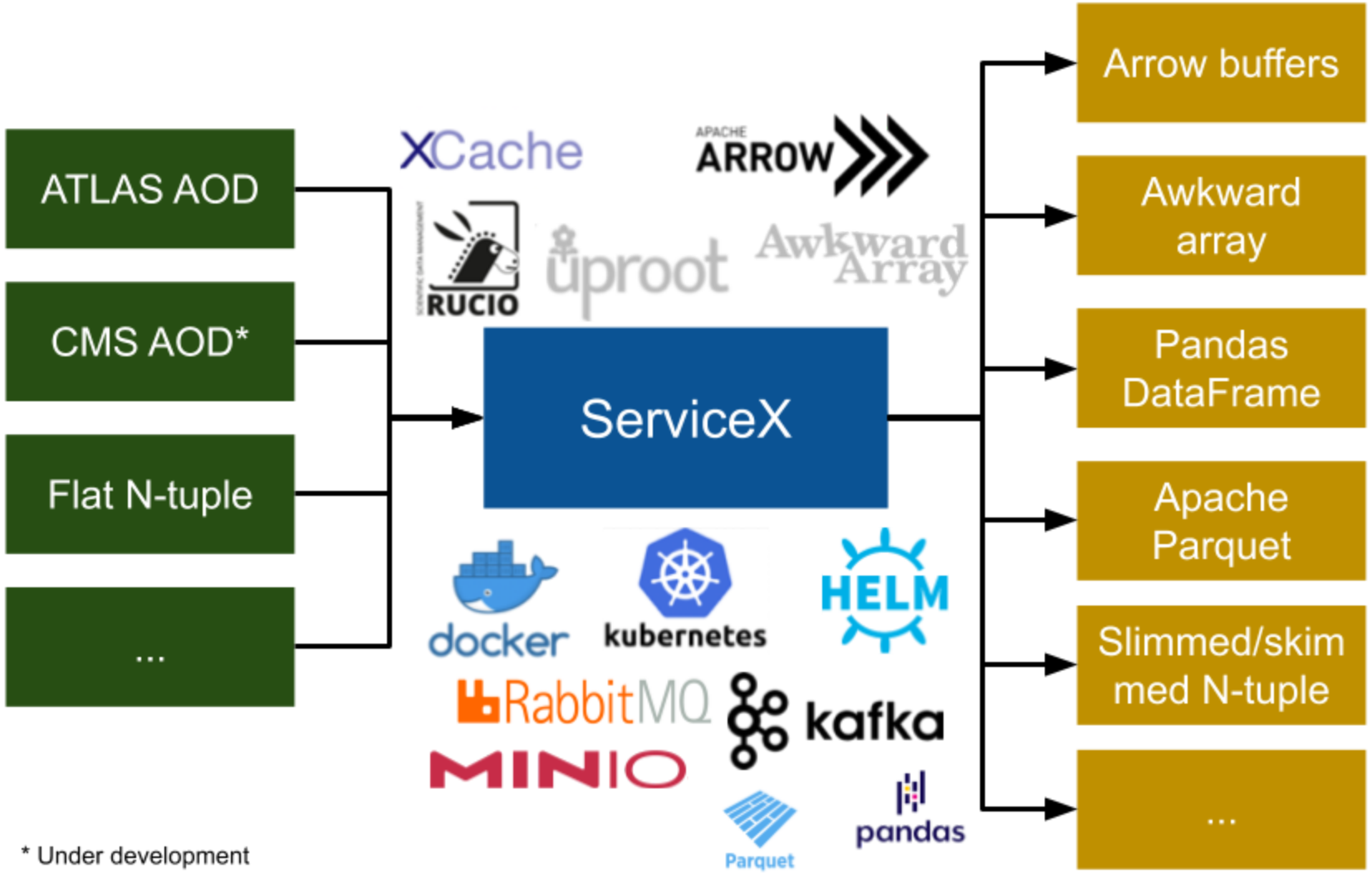
KyungEon Choi (UT Austin), Ben Galewsky (UIUC), Rob Gardner (Chicago), Mason Proffitt (UW), Gordon Watts (UW), Marc Weinberg (Chicago), Ilija Vukotic (Chicago), Chris Weaver (Chicago)

What is ServiceX?

Data delivery from **grid** to **users**: Preselection and column selection, Columnar data in various formats, On-the-fly data transformation, Experiment-agnostic, On-demand delivery

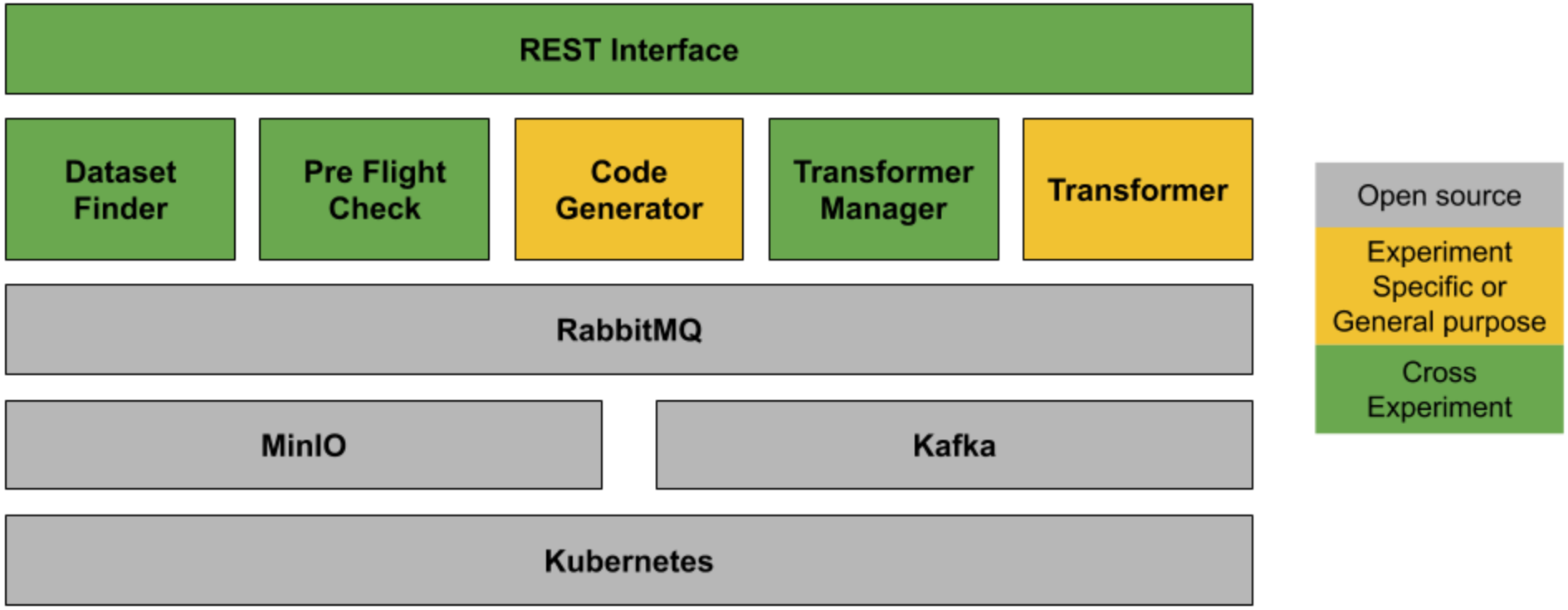


ServiceX Input/Output and Building Blocks

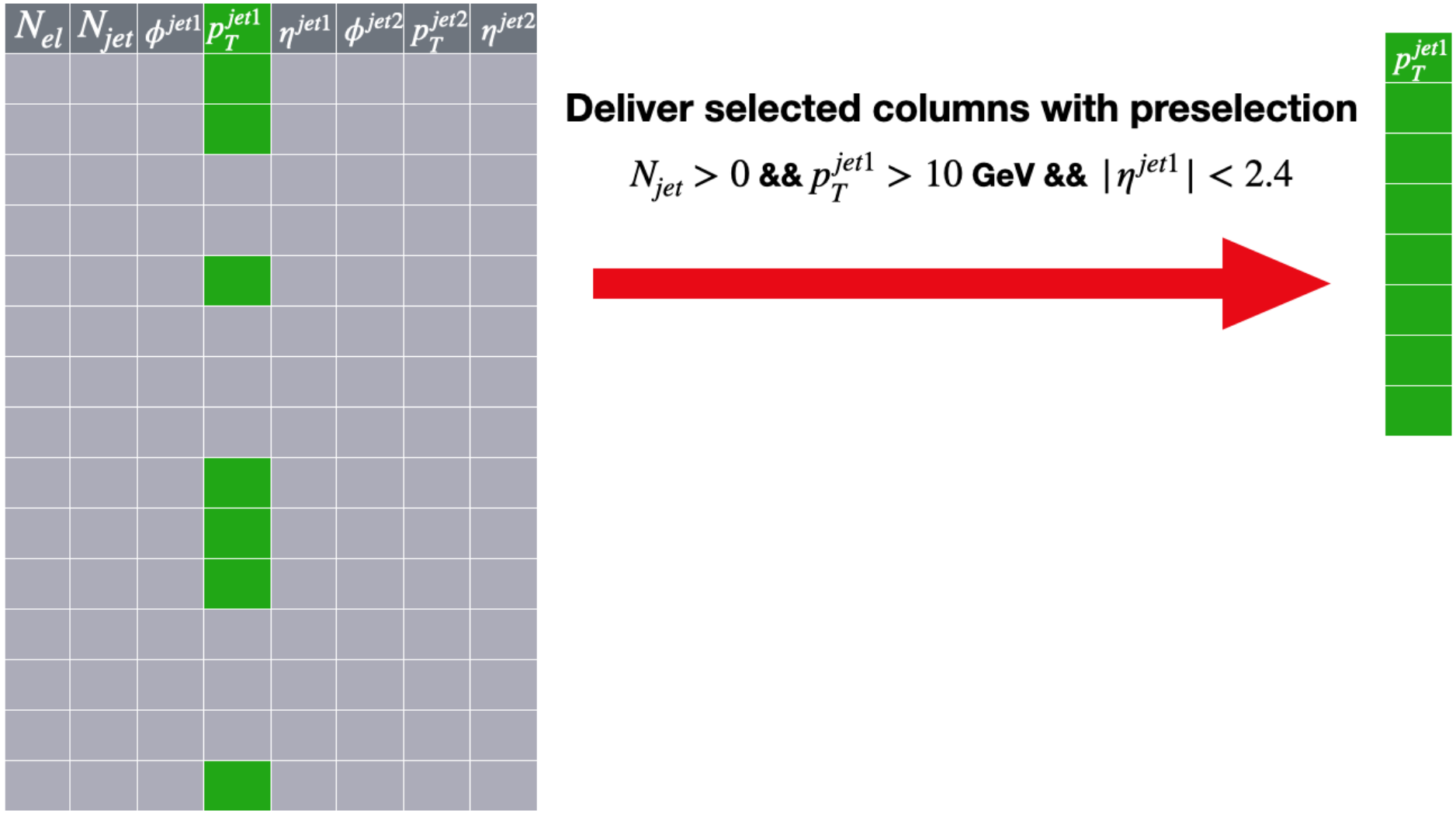


* Under development

ServiceX Backend Architecture



ServiceX Transformer



ServiceX Transformers

- Transformers ready now
    - ATLAS AOD (xAOD) Transformer
    - Uproot (flat N-tuple/CMS nanoAOD) Transformer
- Transformers under development
    - CMS AOD (mini AOD) Transformer
    - PyROOT Transformer

ServiceX Frontend Library

Client library to access ServiceX backend

- Accepts `qastle` formatted query for preselection and column selection
- Supported output: `awkward`, `pandas DF`, `parquet`, `root-file`
- Support simultaneous queries
- Start download as soon as ready
- Local caching of query data

ServiceX Frontend Library

Prerequisites

- python 3.6 or later
- `ServiceX` end points

ServiceX Frontend Library

Uproot Transformer Demo

- Input data type: Flat N-tuple
- Input data size: 76 GB (6 files)
- ServiceX deployed at SSL-RIVER cluster Kubernetes cluster
- Dataset at ATLAS Tier-2 site

Let's have a ServiceX instance for Uproot end-point

Let's have a ServiceX instance for Uproot end-point

In [ ]:

```
import servicex
dataset_uproot = "user.kchoi:user.kchoi.ttHML_80fb_ttH"
tree_name = "nominal"
uproot_transformer_image="sslhep/servicex_func_adl_uproot_transformer:develop"
s_uproot =
servicex.ServiceXDataset(dataset=dataset_uproot,image=uproot_transformer_image,
tree_name=tree_name)
```

Let's prepare a query

Let's prepare a query

In [ ]:

```
import tcut_to_qastle
query_uproot =
tcut_to_qastle.translate("lep_Pt_1>20e3&&lep_Pt_2>20e3&&lep_isolationFixedCutLo
ose_0 && abs(Mlll012-91.2e3)>15e3","lep_Pt_1, lep_Pt_2")
# print(query_uproot)
```

Now request for a delivery

Now request for a delivery

In [ ]:

```
1 out_uproot = s_uproot.get_data_pandas_df(query_uproot)
2 # out_uproot = s_uproot.get_data_parquet(query_uproot)
```

Let's check the output

Let's check the output

In [ ]:
```
1 print(out_uproot)
```

You can now interface the output for ML, hist fitting, or just plotting

You can now interface the output for ML, hist fitting, or just plotting

In [ ]:

```
1 import matplotlib.pyplot as plt
2 plot = plt.hist(out_uproot.lep_Pt_1, bins=100, range=(15e3, 200e3))
```

ServiceX Frontend Library

ATLAS AOD Transformer Demo

- Input data type: ATLAS xAOD
- Input data size: 74 GB (17 files)
- ServiceX deployed at UChicago Kubernetes cluster
- Dataset at ATLAS Tier-2 site

Let's have a ServiceX instance for xAOD end-point

Let's have a ServiceX instance for xAOD end-point

In [ ]:

```
1 dataset_xaod =
"mc15_13TeV:mc15_13TeV.361106.PowhegPythia8EvtGen_AZNLOCTEQ6L1_Zee.merge.DAOD_S
TDM3.e3601_s2576_s2132_r6630_r6264_p2363_tid05630052_00"
2 xaod_transformer_image="sslhep/servicex_func_adl_xaod_transformer:v0.4"
3 s_xaod =
servicex.ServiceXDataset(dataset=dataset_xaod,image=xaod_transformer_image)
4
5 query_xaod = "(call ResultTTree (call Select (call SelectMany (call
EventDataset (list 'localds:bogus')) (lambda (list e) (call (attr e 'Jets')
'AntiKt4EMTopoJets'))) (lambda (list j) (/ (call (attr j 'pt')) 1000.0))) (list
'JetPt') 'analysis' 'junk.root')"
```

Now request for a delivery

Now request for a delivery

In [ ]:

```
out_xaod = s_xaod.get_data_pandas_df(query_xaod)
# out_xaod = s_xaod.get_data_awkward(query_xaod)
# out_xaod = s_xaod.get_data_rootfiles(query_xaod)
```

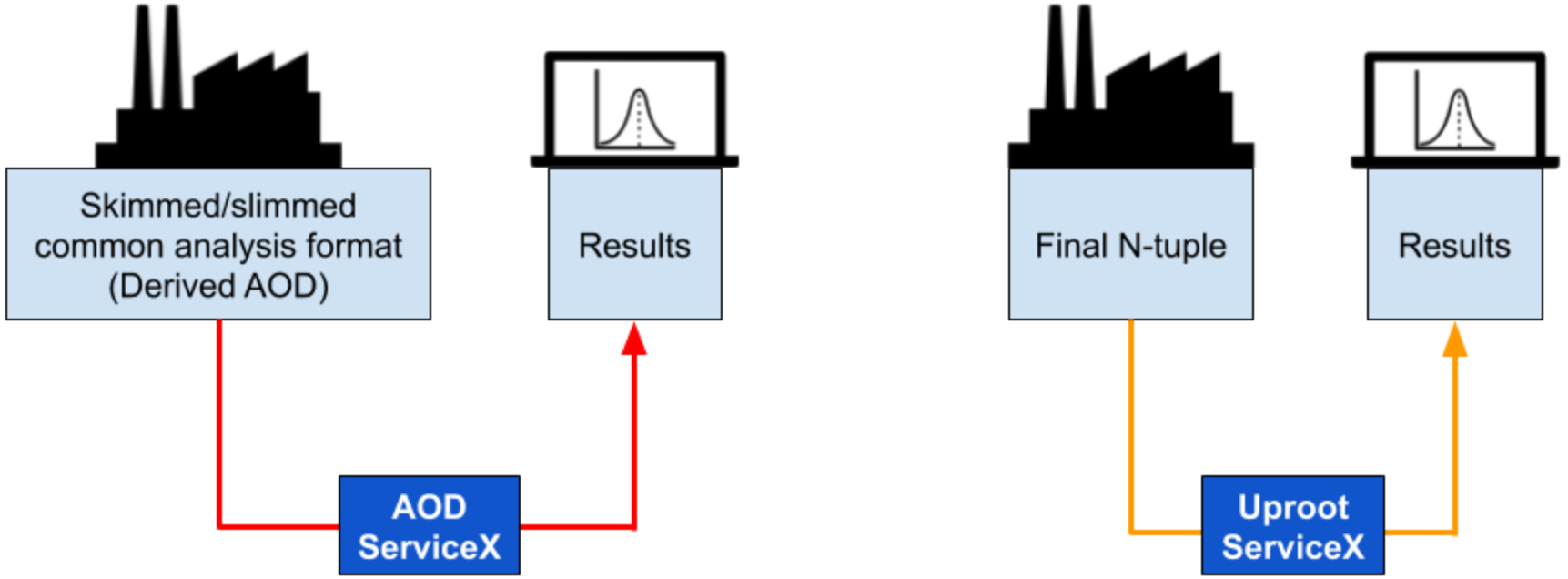
Let's check the output

Let's check the output

In [ ]:

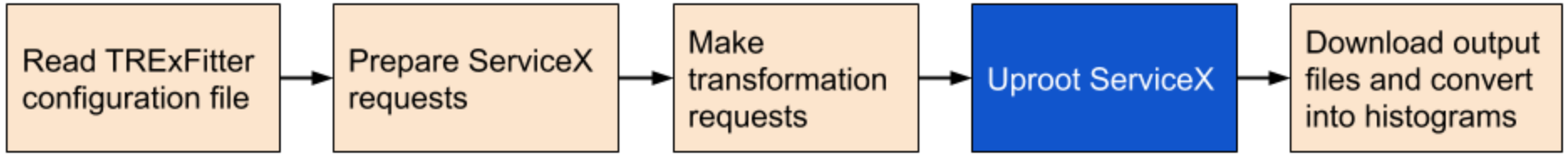
```
1 print(out_xaod)
```

ServiceX workflow



ServiceX Use case - ServiceX for TRExFitter

- TRExFitter is a framework for binned template profile likelihood fits
- Generating input histograms from ROOT N-tuples can be quite expensive for large N-tuples
- Workflow of ServiceXforTRExFitter



➡ No local storage need and easy to scale

ServiceX Use case - ServiceX for NtupleMaker

- A custom PyROOT tranformer to slim/skim ROOT TTree and add new columns (or branches) like a 4-vector reconstructed top
- Not a primary use case of ServiceX, but shows potential adaptability of ServiceX

Very close to the public release of ServiceX!

Very close to the public release of ServiceX!

Useful Links

- ServiceX documentation
- GitHub repos
    - Try ServiceX
    - ServiceX Frontend
    - qastle language
    - TCut to qastle wrapper