

Data Selection & Delivery for Analysis: ServiceX & FuncADL

Mason Proffitt

October 26, 2020

IRIS-HEP Blueprint Workshop: Future Analysis Systems and Facilities

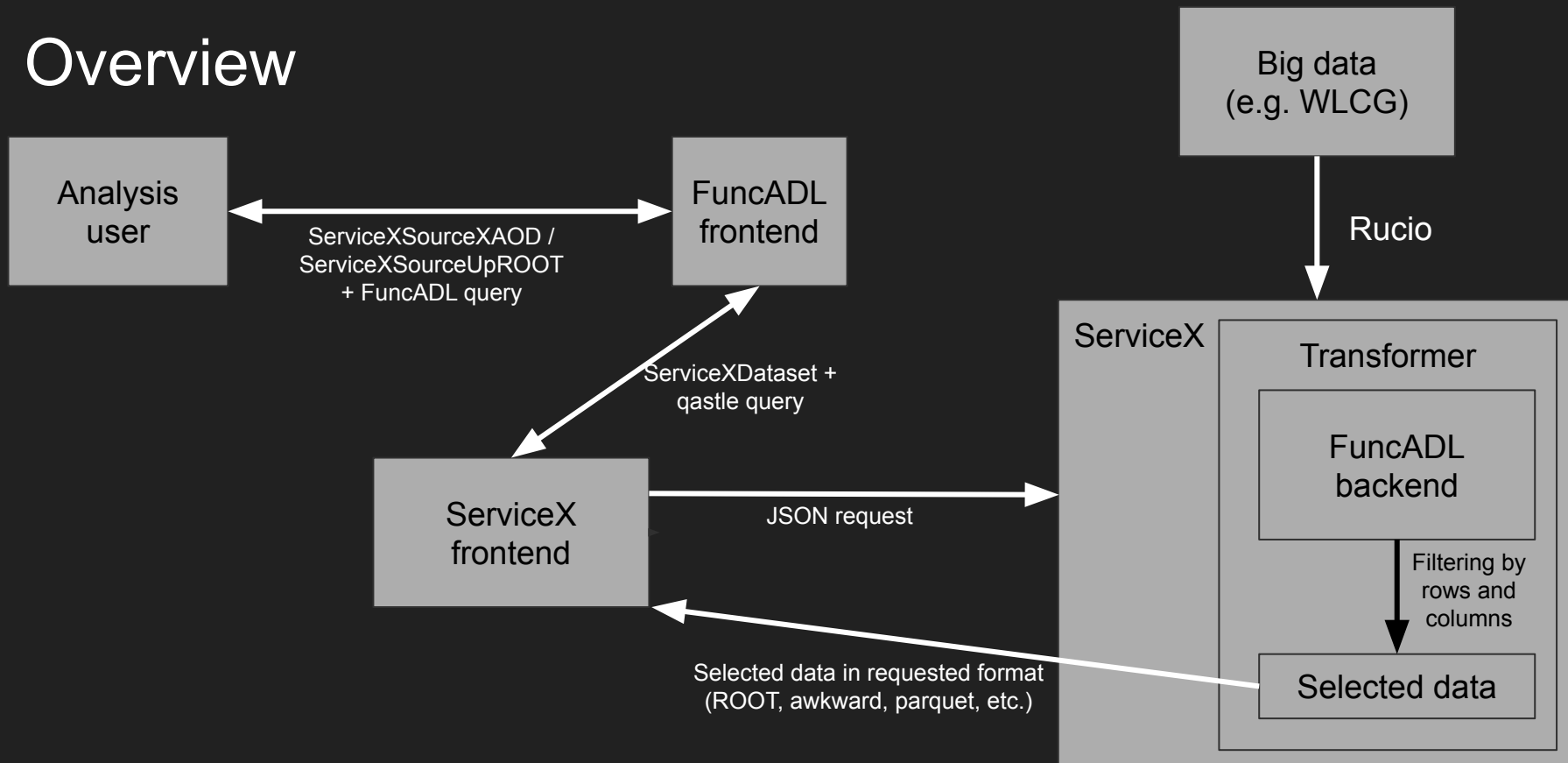
Introduction

- Functional Analysis Description Language (FuncADL):
 - [LINQ](#)-like query language for constructing selections of data
 - Includes operators like `Select()` and `Where()`
 - Queries are written via `func_adl` Python classes
- ServiceX
 - Service that efficiently delivers filtered/transformed data to an analysis user
 - Generally will run on a cluster, but can be run locally

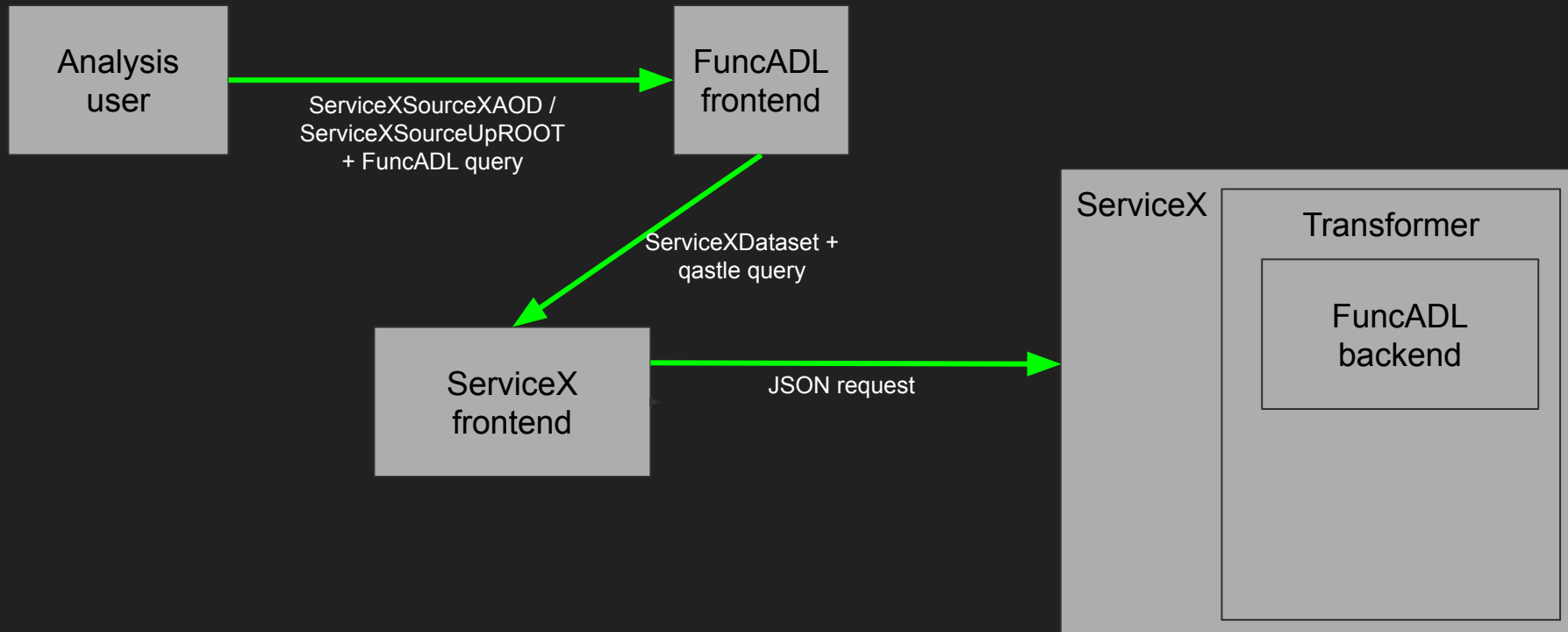
Repositories and modules

- FuncADL frontend (for building a query)
 - [func_adl_servicex](#)
 - [func_adl](#)
- [qastle](#) (Query AST Language Expressions)
 - Plaintext language for communication between FuncADL and ServiceX
- [ServiceX_frontend](#) (`servicex` Python module)
- [ServiceX](#)
- FuncADL backends (actually apply query filters/transformations to data)
 - [func_adl_xAOD](#)
 - [func_adl_uproot](#)

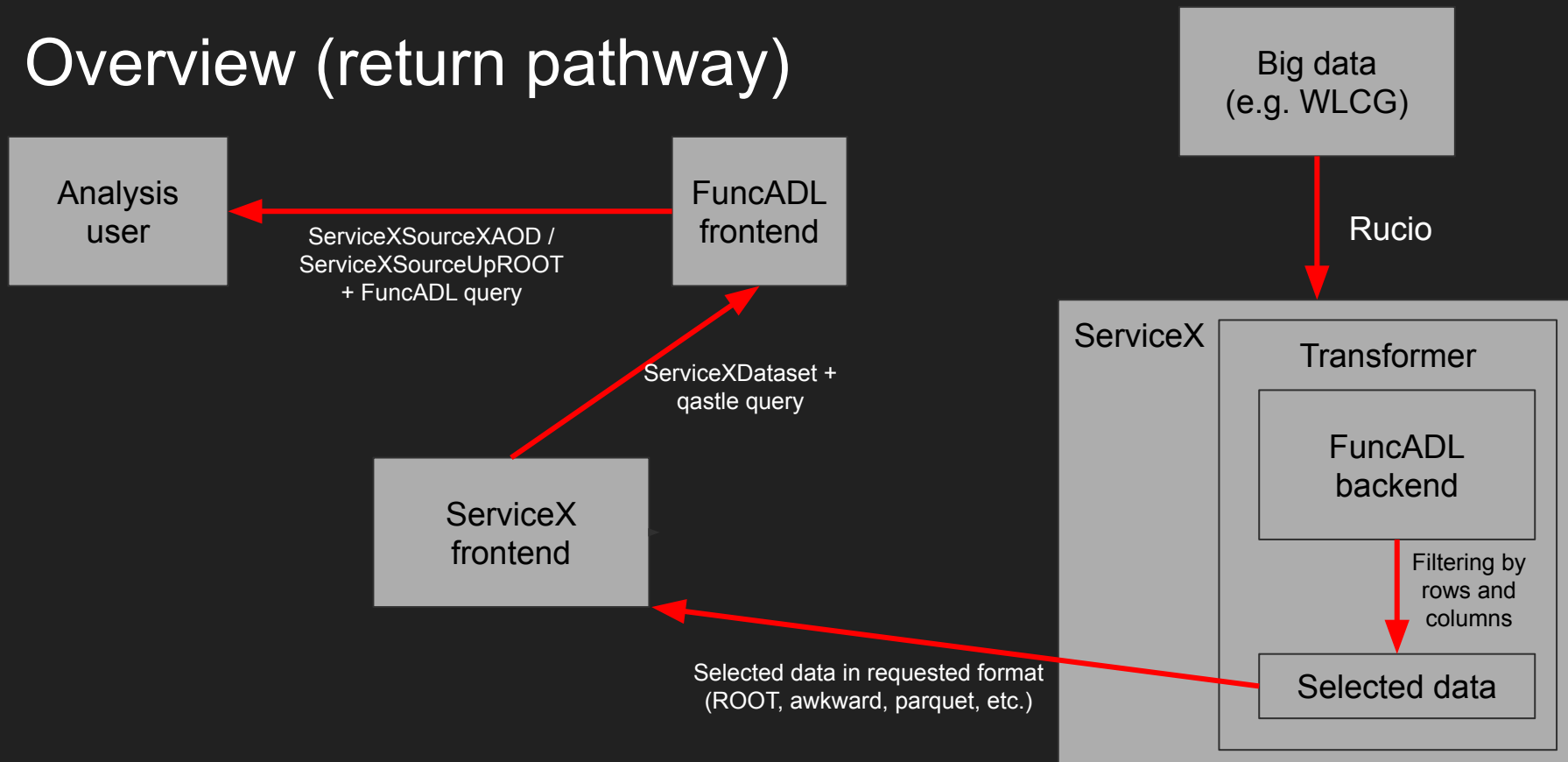
Overview



Overview (query pathway)



Overview (return pathway)



Current status

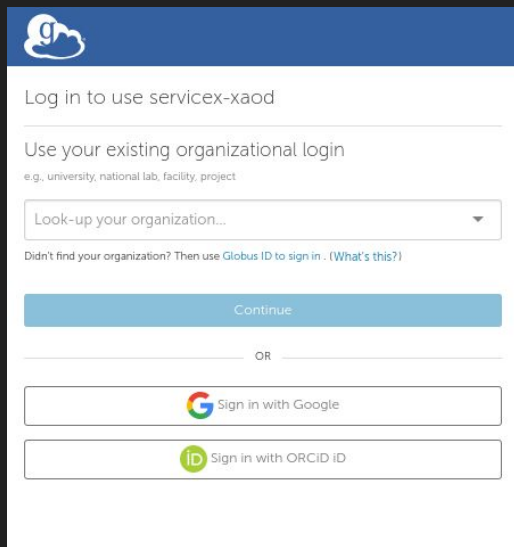
- ServiceX: version 1.0.0-rc.3 (October 8)
 - rc3 brings Globus authentication, website frontend for signing up and API tokens, better documentation, and more
- ServiceX_frontend (`servicex` Python module): version 2.1 (October 21)
- `func_adl_servicex` (FuncADL frontend): version 1.0 (October 21)
- One thing I'm not exactly sure about: timeline for ServiceX rc4, v1.0, etc.
 - I'll have to defer to those working more closely in ServiceX (Ben, Gordon?)

Documentation

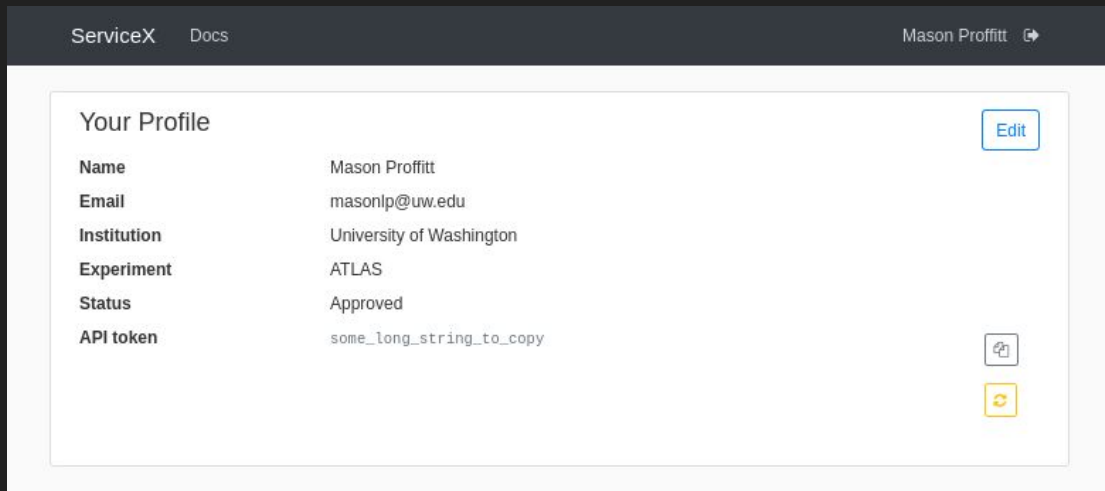
- Read the Docs page:
 - <https://servicex.readthedocs.io/en/latest/>
- Next few slides follow the updated Getting Started guide from:
 - <https://github.com/ssl-hep/ServiceX/pull/211>

Getting started

- Create account on website for appropriate backend
- Get account approved by admin
- Get API key



The screenshot shows the login interface for ServiceX. At the top left is the ServiceX logo. The main heading is "Log in to use servicex-xaod". Below this, it says "Use your existing organizational login" with a subtext "e.g., university, national lab, facility, project". There is a dropdown menu labeled "Look-up your organization...". Below the dropdown is a link: "Didn't find your organization? Then use Globus ID to sign in. (What's this?)". A blue "Continue" button is present. Below the button is an "OR" separator. At the bottom, there are two buttons: "Sign in with Google" and "Sign in with ORCID iD".



The screenshot shows the user profile page for "Mason Proffitt". The page title is "Your Profile" and there is an "Edit" button. The profile information is as follows:

Name	Mason Proffitt
Email	masonip@uw.edu
Institution	University of Washington
Experiment	ATLAS
Status	Approved
API token	some_long_string_to_copy

There are two icons on the right side of the profile: a copy icon and a refresh icon.

Getting started

- Create or download `.servicex` config file:

```
api_endpoints:  
  - endpoint: https://xaod.servicex.ssl-hep.org/  
  token: some_long_string_to_copy  
  type: xaod
```

xAOD example

```
In [1]: import servicex as sx
        from func_adl_servicex import ServiceXSourceXAOD
```

```
In [*]: %%time
```

```
dataset_name = "mc15_13TeV:mc15_13TeV.361106.PowhegPythia8EvtGen_AZNLOCTEQ6L1_Zee.merge.DAOD_STDM3.e3601_s2576_s213:
sx_dataset = sx.ServiceXDataset(dataset_name, "xaod")
src = ServiceXSourceXAOD(sx_dataset)
df = src \
     .SelectMany('lambda e: e.Jets("AntiKt4EMTopoJets")') \
     .Select('lambda j: j.pt()/1000.0') \
     .AsPandasDF('JetPt') \
     .value()

print(df)
```

uproot example

```
In [1]: import pandas as pd
import servicex as sx
from func_adl_servicex import ServiceXSourceUpROOT
```

```
In [2]: %%time
```

```
dataset_name = "data15_13TeV:data15_13TeV.00282784.physics_Main.deriv.DA0D_PHYSLITE.r9264_p3083_p4165_tid21568807_00"
sx_dataset = sx.ServiceXDataset(dataset_name, "uproot")
src = ServiceXSourceUpROOT(sx_dataset, "CollectionTree")
data = src.Select("lambda e: {'JetPT': e['AnalysisJetsAuxDyn.pt']}") \
    .AsParquetFiles('junk.parquet') \
    .value()
df = pd.read_parquet(data[0])

print(df)
```

```
          JetPT
0  [56970.56, 57738.047, 24149.762, 15421.779, 14...
1  [123299.94, 89595.32, 75777.94, 18421.592, 164...
2  [172519.64, 115030.47, 111144.8, 97817.69, 934...
3  [28965.395, 15481.423, 14233.97, 16032.507, 12...
4  [288785.3, 189529.4, 80025.805, 43544.61, 1581...
...
54238 [347737.28, 313428.75, 46344.66, 33925.395, 20...
54239 [45954.137, 41864.71, 15005.428]
54240 [76411.27, 66487.41, 60403.04, 51341.3, 41749....
54241 [33027.637, 24204.908, 18219.818]
54242 []
```

```
[54243 rows x 1 columns]
```

```
CPU times: user 1.6 s, sys: 904 ms, total: 2.5 s
Wall time: 3min 5s
```

TCut example

- Non-FuncADL frontends can also plug into ServiceX
 - Can use anything that can translate to qastle (a standardized plaintext format for specifying a query)
 - For example: [tcut to qastle](#)

```
In [1]: import servicex
import tcut_to_qastle

In [2]: %%time
query = tcut_to_qastle.translate('nominal', 'lep_pt_1', 'lep_pt_1>1000')
dataset_uproot = "user.kchoi:user.kchoi.tthML_80fb_ttbar"
sx_dataset = servicex.ServiceXDataset(dataset=dataset_uproot, backend_type='uproot')
data = sx_dataset.get_data_parquet(query)

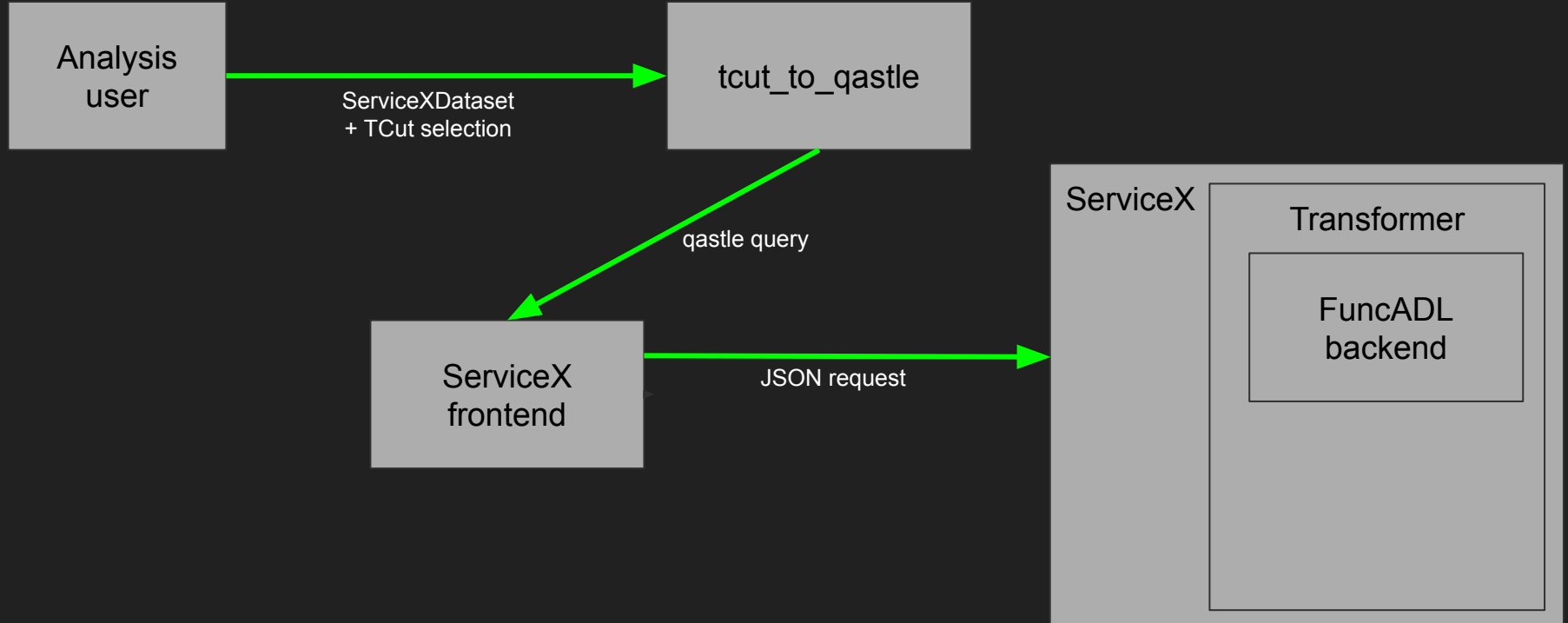
CPU times: user 258 ms, sys: 36.9 ms, total: 295 ms
Wall time: 34.8 s

In [4]: import pandas as pd
df = pd.read_parquet(data[0])
print(df)

      lep_pt_1
0      29697.210938
1      59942.164062
2      18633.767578
3      10757.475586
4       78170.820312
...
141350  31596.203125
141351  31708.304688
141352  40093.730469
141353  39144.351562
141354  18899.412109

[141355 rows x 1 columns]
```

Overview (query using TCut)



Some final comments

- A few caveats:
 - func_adl doesn't support Python 3.8 or 3.9 yet
 - Support for some features varies between xAOD and uproot FuncADL backends, plus some other subtle inconsistencies between the two
 - Debugging experience is far from ideal
 - For example, I'm still working out a "Gateway time-out" error from the earlier xAOD example
 - but work is ongoing to improve all of these...
- But ServiceX 1.0.0-rc.3 has brought huge improvements for usability:
 - Getting the uproot example working perfectly (delivering selected data from the grid) from scratch took mere minutes following the new Getting Started guide!