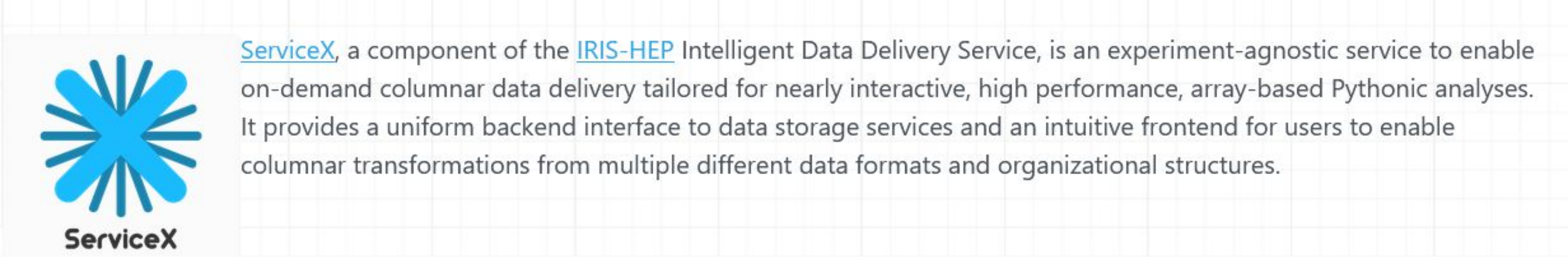
Leveraging ServiceX to Transform PHYSLITE Data into Flat N-tuples with Systematics

Alexander Schmidt

How can this be done?

Encapsulate TopCPToolkit in a docker container which can be accessed on ServiceX by anyone in ATLAS

Intro to ServiceX:



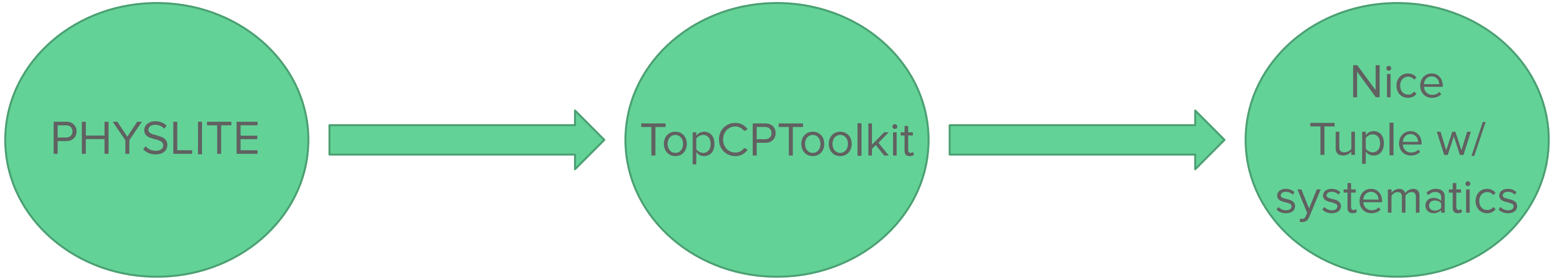
ServiceX, a component of the IRIS-HEP Intelligent Data Delivery Service, is an experiment-agnostic service to enable on-demand columnar data delivery tailored for nearly interactive, high performance, array-based Pythonic analyses. It provides a uniform backend interface to data storage services and an intuitive frontend for users to enable columnar transformations from multiple different data formats and organizational structures.

Example applications on ServiceX:

- Uproot transformer (takes json as query and returns filtered flat ROOT ntuple)
- funcadl_xAOD transformer (uses C++ to extract columns of data from ATLAS xAOD binary files)
- Python transformer (takes python script as query and runs on input files via ServiceX)

Intro to TopCPToolkit:

PHYSLITE is a light-weight file format used to store particle collision event data. PHYSLITE files are designed for C++ making them difficult to use in the python ecosystem. Systematic uncertainties are not directly stored and must be calculated from these files, but there currently exists no python libraries to do that. However, these files can be n-tupilized for compatibility with the python ecosystem. Using TopCPToolkit, these files can be filtered and n-tupilized according to user specifications. Additionally, TopCPToolkit can calculate systematic uncertainties from these files - information that would otherwise be inaccessible once n-tupelized. TopCPToolkit, is a powerful tool, but it can be cumbersome to set up. By making a TopCPToolkit transformer for ServiceX, we can capture the full analysis power of TopCPToolkit while skipping the many steps needed to set it up and access potentially more computing resources than locally available.

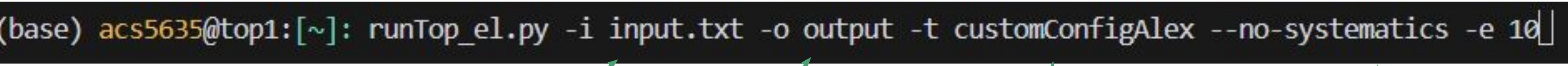


PHYSLITE

TopCPToolkit

Nice Tuple w/ systematics

How TopCPToolkit is normally used:



```
(base) acs5635@top1:[~]: runTop_el.py -i input.txt -o output -t customConfigAlex --no-systematics -e 10
```

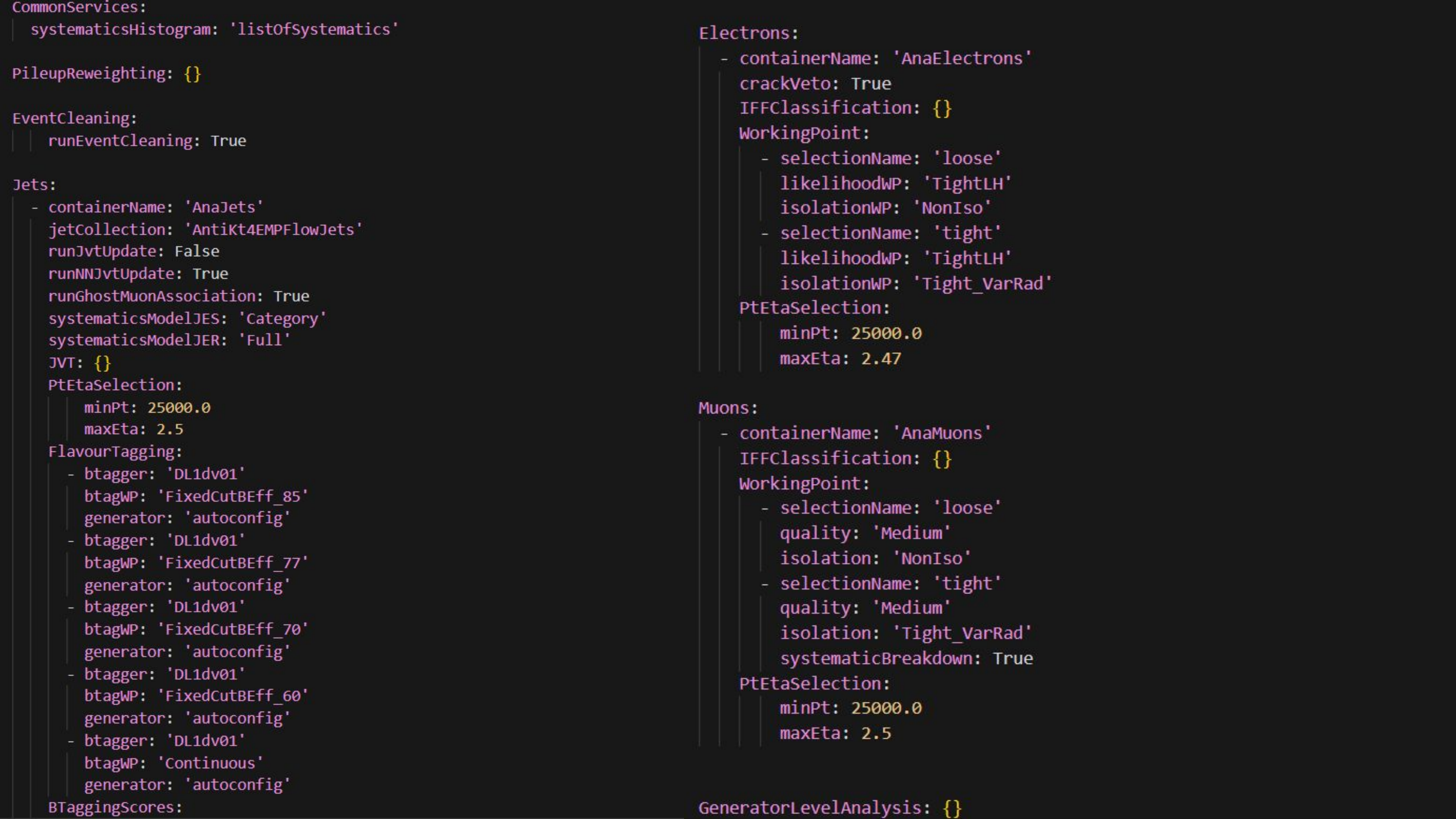
Run the event loop

List of PHYSLITE files. Can be addresses of the files on the internet

Name of the output file which will be in root format

Folder with one or more yaml files that specify what information we want from these files

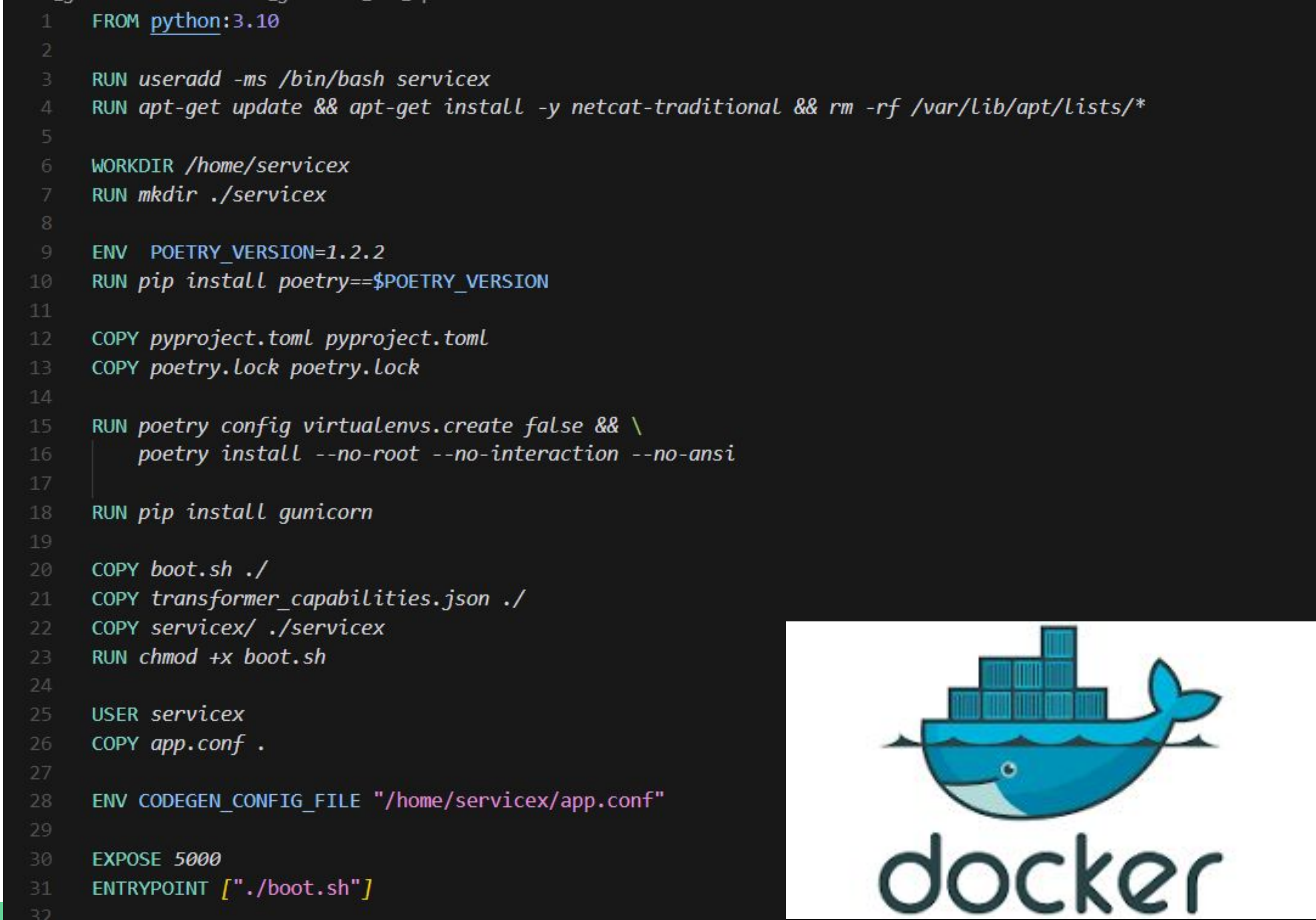
A list of other options (e.g number of events)



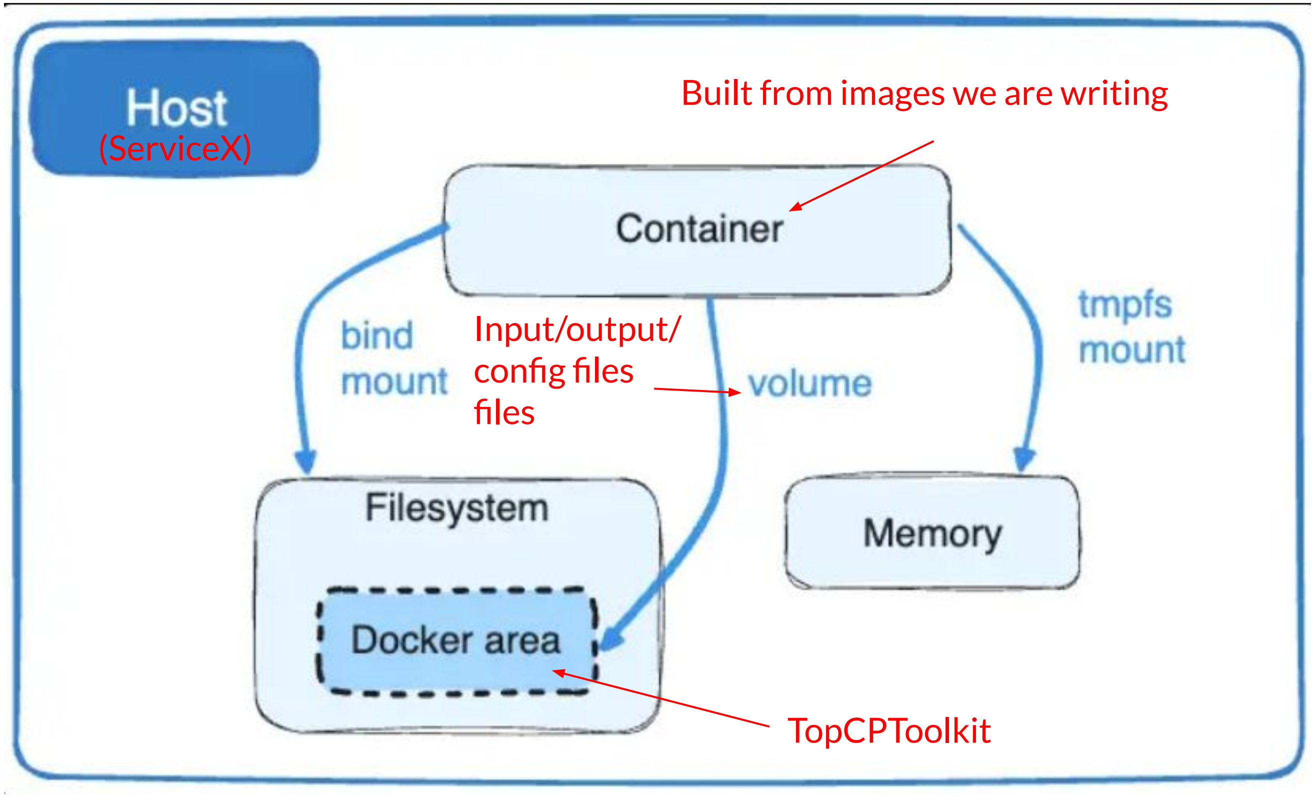
```yaml
CommonServices:
  systematicsHistogram: 'listOfSystematics'

PileupReweighting: {}

EventCleaning:
    runEventCleaning: True

Jets:
  - containerName: 'AnaJets'
    jetCollection: 'AntiKt4EMPFlowJets'
    runJvtUpdate: False
    runNNJvtUpdate: True
    runGhostMuonAssociation: True
    systematicsModelJES: 'Category'
    systematicsModelJER: 'Full'
    JVT: {}
    PtEtaSelection:
        minPt: 25000.0
        maxEta: 2.5
    FlavourTagging:
      - btagger: 'DL1dv01'
        btagWP: 'FixedCutBEff_85'
        generator: 'autoconfig'
      - btagger: 'DL1dv01'
        btagWP: 'FixedCutBEff_77'
        generator: 'autoconfig'
      - btagger: 'DL1dv01'
        btagWP: 'FixedCutBEff_70'
        generator: 'autoconfig'
      - btagger: 'DL1dv01'
        btagWP: 'FixedCutBEff_60'
        generator: 'autoconfig'
      - btagger: 'DL1dv01'
        btagWP: 'Continuous'
        generator: 'autoconfig'
    BTaggingScores:
```

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

Muons:
  - containerName: 'AnaMuons'
    IFFClassification: {}
    WorkingPoint:
      - selectionName: 'loose'
        quality: 'Medium'
        isolation: 'NonIso'
      - selectionName: 'tight'
        quality: 'Medium'
        isolation: 'Tight_VarRad'
        systematicBreakdown: True
    PtEtaSelection:
        minPt: 25000.0
        maxEta: 2.5


GeneratorLevelAnalysis: {}
```

Intro to Docker:



```dockerfile
FROM python:3.10

RUN useradd -ms /bin/bash servicex
RUN apt-get update && apt-get install -y netcat-traditional && rm -rf /var/lib/apt/lists/*

WORKDIR /home/servicex
RUN mkdir ./servicex

ENV  POETRY_VERSION=1.2.2
RUN pip install poetry==$POETRY_VERSION

COPY pyproject.toml pyproject.toml
COPY poetry.lock poetry.lock

RUN poetry config virtualenvs.create false && \
    poetry install --no-root --no-interaction --no-ansi

RUN pip install gunicorn

COPY boot.sh ./
COPY transformer_capabilities.json ./
COPY servicex/ ./servicex
RUN chmod +x boot.sh

USER servicex
COPY app.conf .

ENV CODEGEN_CONFIG_FILE "/home/servicex/app.conf"

EXPOSE 5000
ENTRYPOINT ["./boot.sh"]
```

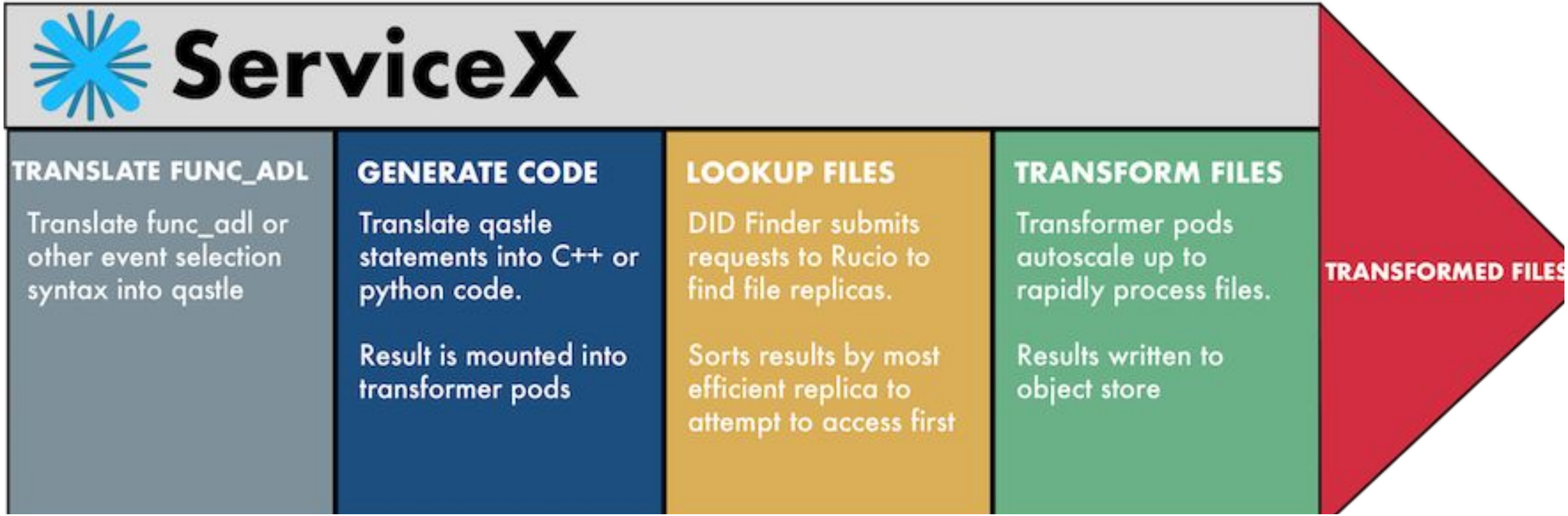


(ServiceX)

Built from images we are writing

Input/output/
config files
files

TopCPToolkit



ServiceX

TRANSLATE FUNC_ADL

Translate func_adl or other event selection syntax into qastle

GENERATE CODE

Translate qastle statements into C++ or python code.

Result is mounted into transformer pods

LOOKUP FILES

DID Finder submits requests to Rucio to find file replicas.

Sorts results by most efficient replica to attempt to access first

TRANSFORM FILES

Transformer pods autoscale up to rapidly process files.

Results written to object store

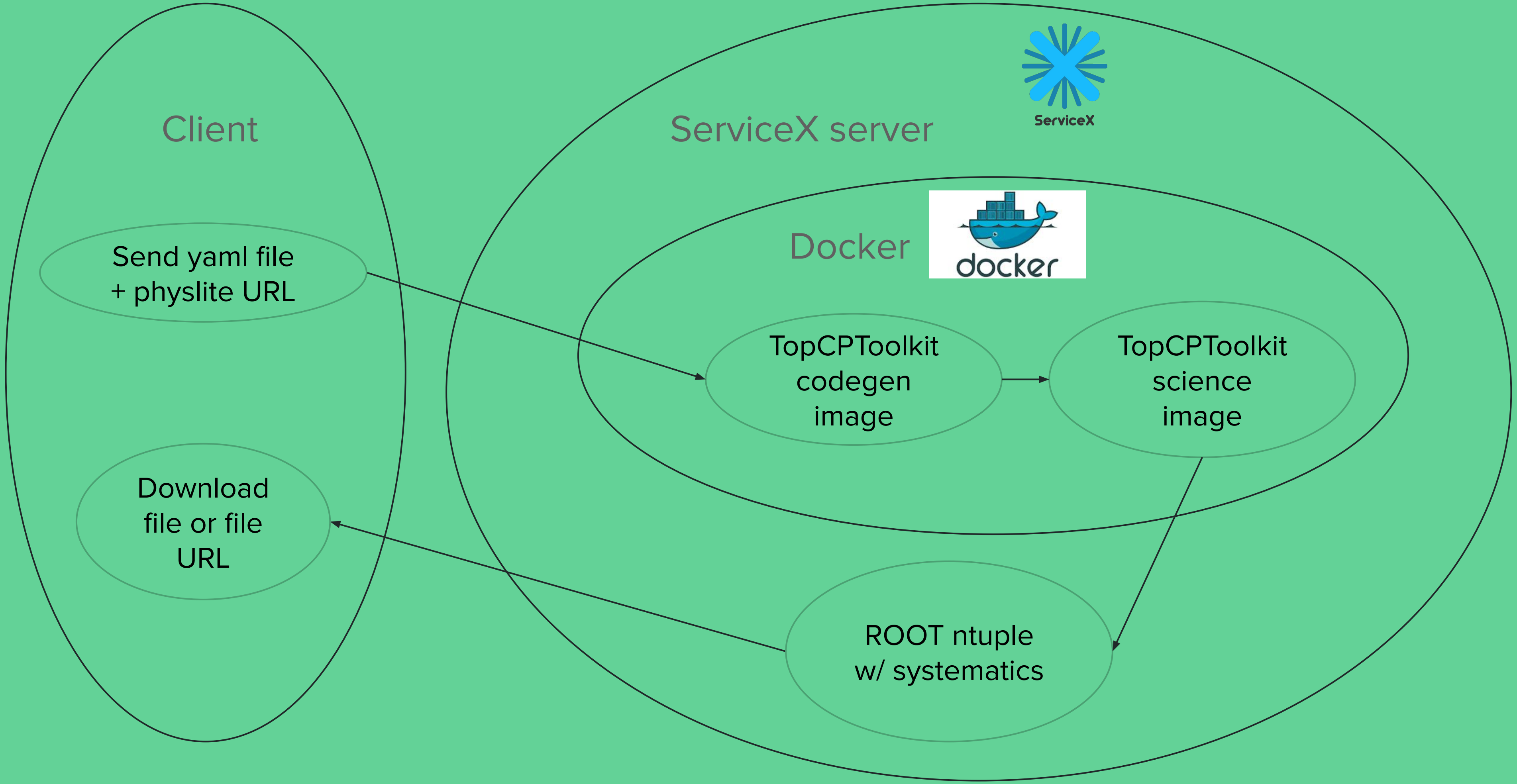
TRANSFORMED FILES

Send in yaml query file

Need a second docker image called “codegen” image

Look for specific PHYSLITE files

TopCPToolkit image called “science” image



Client

Send yaml file + physlite URL

Download file or file URL

ServiceX server

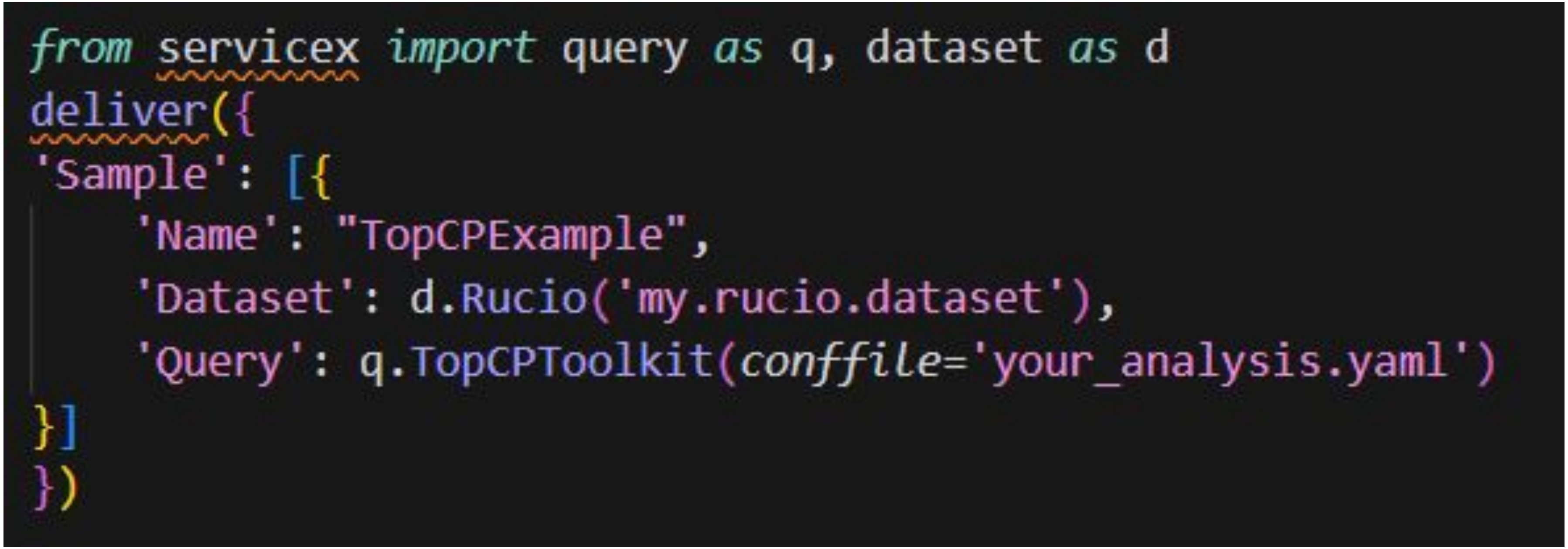
Docker

TopCPToolkit codegen image

TopCPToolkit science image

ROOT ntuple w/ systematics

How TopCP will work over ServiceX (in pseudocode):



```
from servicex import query as q, dataset as d
deliver({
'Sample': [{
    'Name': "TopCPExample",
    'Dataset': d.Rucio('my.rucio.dataset'),
    'Query': q.TopCPToolkit(conffile='your_analysis.yaml')
}]
})
```

Completed so far:

- science image is completed
- codegen image is completed
- X509 proxy image (sets up X509 certificate much more easily)
- ServiceX test harness (which makes testing/developing new transformers easier)

What’s left:

- Make sure science and codegen image communicate properly
- Put finished product on ServiceX server
- Write an example client script to make application easier to use

Questions?