The PHYSLITE to OpenData Framework

ATLAS Open Data Tutorial

26 November 2025

Eirik Gramstad

The ATLAS Open Data chain

- In this session we will look at the framework for **converting from the Research to the Education and Outreach Open Data (skim and noskim)**

- **Total: 65 TB**
  - 36 fb^{-1} of data recorded in 2015 and 2016
  - 2 billion simulated events
- **Data format**: PHYSLITE

- **Total: 2 TB**
- *Unskimmed* samples with only very basic cuts such as event cleaning, good runs list, overlap removal &c
- **Data format**: ROOT NTuples

- Size of skimmed data set varies from **~1.5 GB to ~350 GB**
- **Data format**: ROOT NTuples

PhysliteToOpenDataFramework

- Framework based on CP(*)-algorithms using the latest ATLAS analysis releases
  - Can be run inside a docker container image
- Runs on (any) Run 2 PHYSLITE input
- Gitlab repository set up with continuous integration and unit testing
- Repository is registered in Zenodo with a DOI so that it can be cited with a persistent identifier
- Comes also with scripts for job submission and for preparing education and outreach NTuples

(*) The ATLAS combined performance (CP) groups provide recommendations for use in analysis

How to setup the framework

Several options (in preferred order):

1. If CERN account: Use lxplus!
   a. Instructions are similar as for the Escape Virtual Research Environment [setup]
2. If Docker on your laptop: use docker
   a. Instructions on how to use your own computer and Docker [setup]
3. In non of the above works
   a. login to ESCAPE Virtual Research Environment [setup]
   b. Please create an account now [slide 12 here]

- Please follow the setup instruction for one of the option.
- The remainder of the tutorial will be (more or less) independent on the framework you choose.

Getting Started : Docker 1/2

- You need to download Docker on your machine
- Get the software

```
mkdir OpenData; cd OpenData
mkdir source run build data
cd source
git clone --branch 2025beta \
https://gitlab.cern.ch/atlas-outreach-data-tools/physlitetoopendata.git PhysLiteToOpenData
cp PhysLiteToOpenData/extras/CMakeLists-template.txt CMakeLists.txt
```

- Go to the data directory and download a file from Research Open Data

```
cd ../data
curl -LJO
"https://opendata.cern.ch/record/80013/files/mc20_13TeV_MC_PhPy8EG_NNPDF3_AZNLO_ggZH125_Ht
autau_Zinc_file_index.json_0"
```

Getting Started : Docker 2/2

- Start docker image

```
sudo systemctl start docker
```
→ Starting docker

Register your access token (see slide on how to acquire an access token):

```
echo "<token here>" | docker login gitlab-registry.cern.ch -u username --password-std
```

Run the docker image:

```
docker run -it -v "$(pwd)/data":/workdir/data -v "$(pwd)/run":/workdir/run \
gitlab-registry.cern.ch/atlas-outreach-data-tools/physlitetoopendata:latest bash
```

Mount local folders

Analysis release

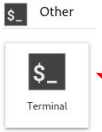
Command to run (*bash* opens a terminal within the container)

```
Configured GCC from: /opt/lcg/gcc/13.1.0-b3d18/x86_64-el9/bin/gcc
Configured AnalysisBase from: /usr/AnalysisBase/25.2.40/InstallArea/x86_64-el9-gcc13-opt
[bash][atlas]:run > ls
data  run  submitDir  submitDir-2025-11-24-0645-4434  submitDir-2025-11-24-0655-53af
```

All set, you're ready to go

Getting started : lxplus/VRE 1/2

- You should already have a verified account in ESCAPE Virtual Research Environment (see talk on resources)
- Click here: https://jhub-vre.cern.ch (Select the **default environment**)
- Open a terminal



- Getting the software:

If on lxplus, do:
`ssh -Y <cern username>@lxplus.cern.ch`
And start from “Getting the software”

```
mkdir OpenData; cd OpenData
mkdir source run build data
cd source
git clone --branch 2025beta \
https://gitlab.cern.ch/atlas-outreach-data-tools/physlitetoopendata.git PhysLiteToOpenData
cp PhysLiteToOpenData/extras/CMakeLists-template.txt CMakeLists.txt
```

- Setting up the release:

```
cd ../
export ATLAS_LOCAL_ROOT_BASE=/cvmfs/atlas.cern.ch/repo/ATLASLocalRootBase
alias setupATLAS='source ${ATLAS_LOCAL_ROOT_BASE}/user/atlasLocalSetup.sh'
setupATLAS -c el9
cd build
asetup AnalysisBase,25.2.40
```

Not needed on lxplus (or any almalinux9 machine)

Getting started : lxplus/VRE 2/2

- Compile the software

```
cmake ../source/
cmake --build .
source x86_64*/setup.sh
```

Important: this must be executed in build directory!

- Downloading the input:

```
cd ../data
curl -LJO
"https://opendata.cern.ch/record/80013/files/mc20_13TeV_MC_PhPy8EG_NNPDF3_AZNLO_ggZH125_Htautau_Zinc_file_index.json_0"
```

All set, you're ready to go

One final thing before running…

- Define the files to run over

```
cd run
ls ../data/DAOD_PHYSLITE.38191136._000001.pool.root.1 &> ../data/inputlist.txt
```

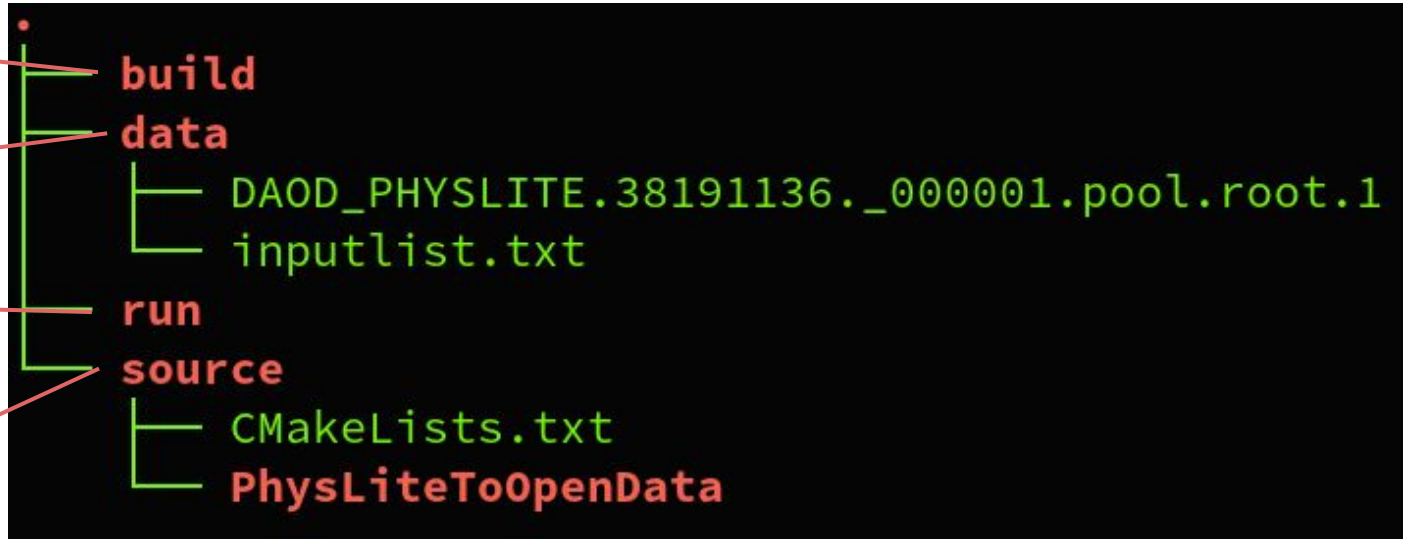
- Now, you should have something like this

Where we compile

Input data

Where we run

The code repository



Let’s try to run over the input file

- From within the run-directory

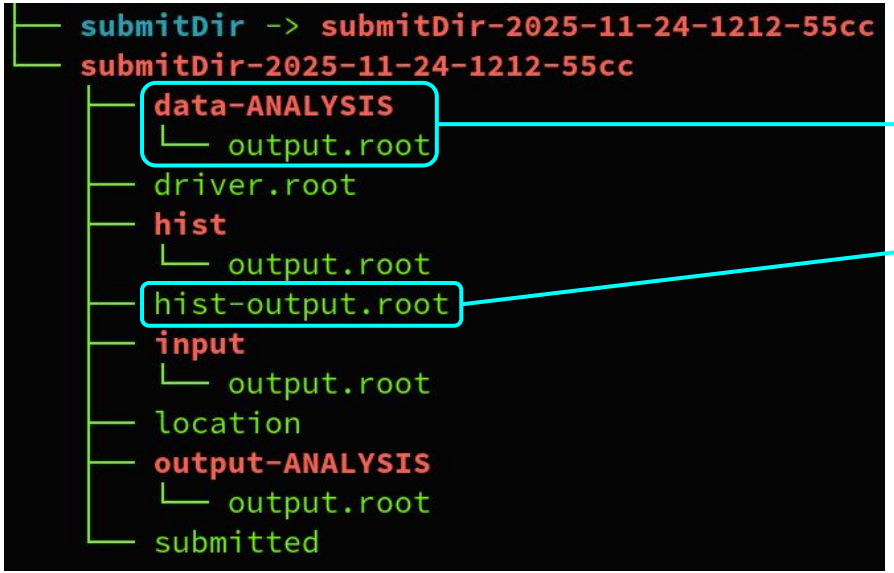
```
OpenDataNtupler.py -i ../data/inputlist.txt
```

Tip when running on ESCAPE Virtual Research Environment

- Add `-n 100` to only process 100 events
  - if not you may experience problems or have to wait for a long time

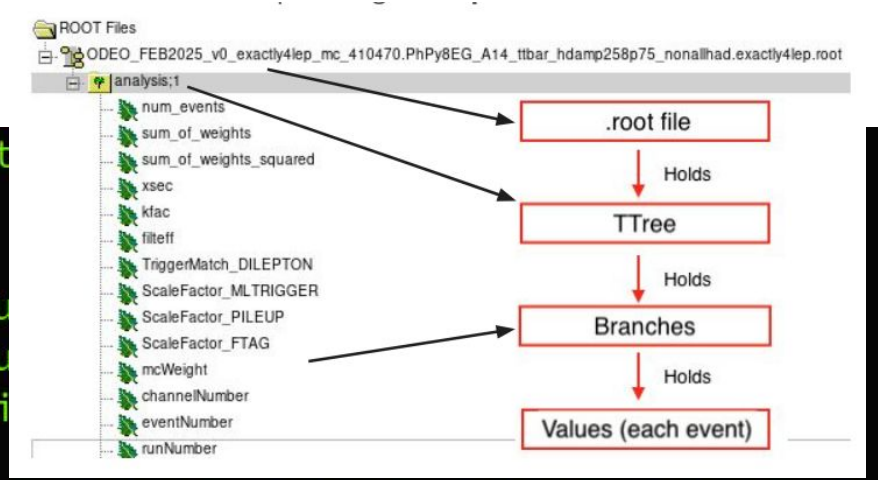
Looking at the output

- The output is saved in the *submitDir* directories
  - The folders will have time stamps, and the *submitDir* will always point to the latest



Looking at the output: ROOT NTuple

- The ntuple (should be familiar if you followed the Education and Outreach part of the tutorial)
  - If not; look at Monday's tutorials




```
Attaching file submitDir/data-ANALYSIS/out
(TFile *) 0x26586870
root [1] _file0->ls()
TFile**         submitDir/data-ANALYSIS/ou
 TFile*         submitDir/data-ANALYSIS/ou
  KEY: TTree    analysis;1      My analysi
root [2] analysis->Print()
******************************************************************************
*Tree    :analysis  : My analysis ntuple                                      *
*Entries :    39830 : Total =        53082905 bytes  File  Size =   18762530 *
*        :          : Tree compression factor =   2.83                       *
******************************************************************************
```

Looking at the output: Bookkeeping histograms

- The histograms contain information about the processed files, number of events included, run time etc.

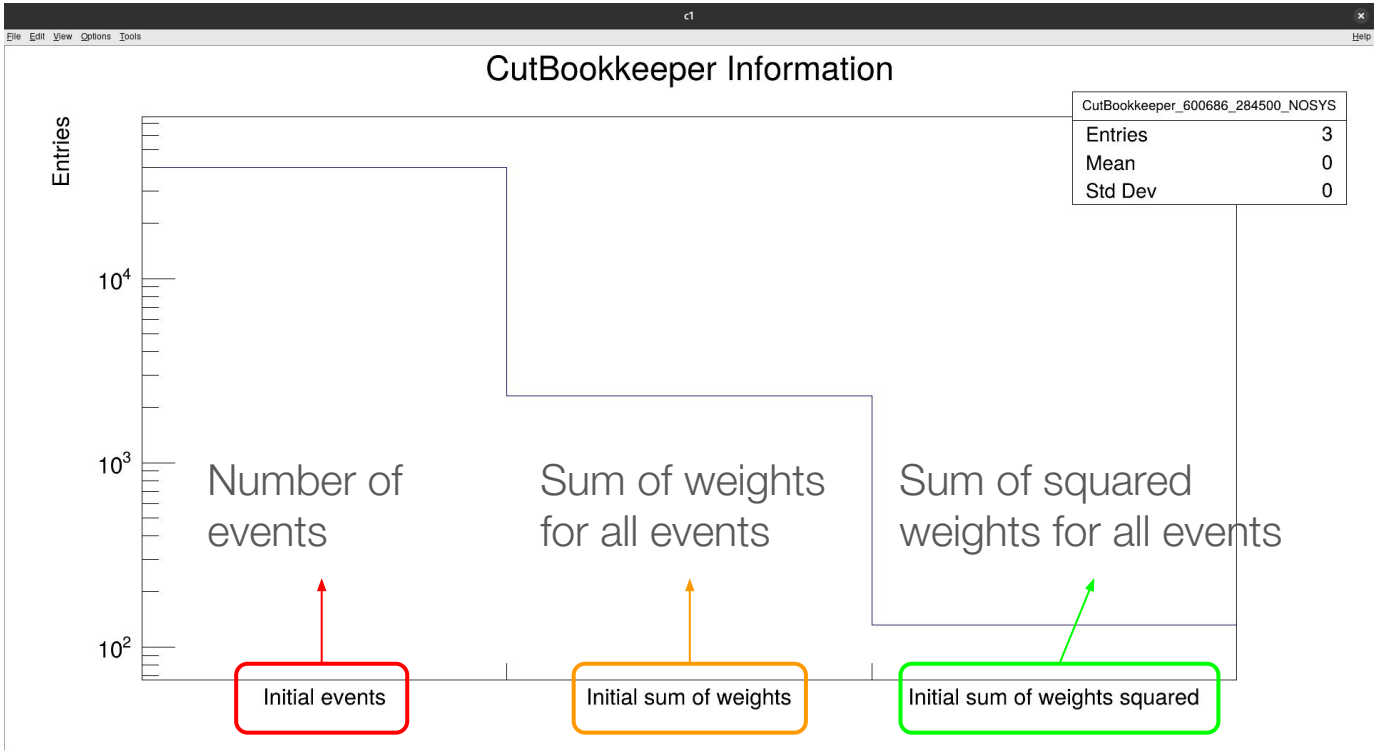
```
Attaching file submitDir/hist-output.root as _file0...
(TFile *) 0x10e25ad0
root [1] _file0->ls()
TFile**         submitDir/hist-output.root
 TFile*         submitDir/hist-output.root
  KEY: TList    EventLoop_OutputStream_ANALYSIS;1       Doubly linked list
  KEY: TTree    EventLoop_FileExecuted;1        executed files
  KEY: TTree    EventLoop_JobStats;1    EventLoop job statistics
  KEY: TH1F     CutBookkeeper_600686_284500_NOSYS;1     CutBookkeeper Information
  KEY: TH1D     EventLoop_RunTime;1     worker run-time summary
  KEY: TH1D     EventLoop_EventCount;1  number of events per algorithm
```

We will look at this histogram. It contains the **original number of events before any selection have been applied**. Important for **scaling the MC to the integrated luminosity of the data set** ("sum of weights")

$$w_\sigma = \frac{\int L\mathrm{d}t\ \sigma\ \eta\ k\ w}{\sum_i w_i}$$

Looking at the output: Bookkeeping histograms

- The CutBookkeeper histogram



Creating an Education and Outreach Open Data NTuple

- To create an NTuple on the same format as the officially released **Education and Outreach Open Data NTuple** we need to
  - Add the information on the number of events/sum of weights to the ntuples
  - Define a “skim”.
    - Apply some specific selection or apply no selection (aka. *noskim*)
- There’s a script for this in
  - */source/PhysliteToOpenData/scripts/skimOpenDataNtuples.py*
- First, get the latest version of the script

```
cd source/PhysLiteToOpenData/
git checkout origin/updateSkimScript -- scripts/skimOpenDataNtuples.py
```

Creating an Education and Outreach Open Data NTuple

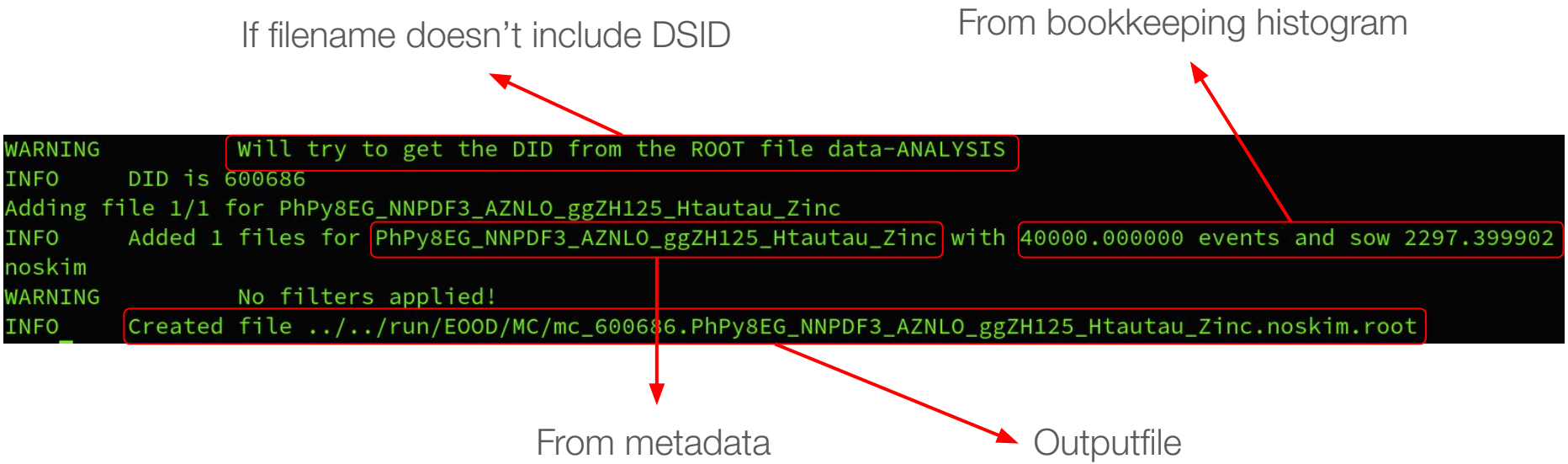
- It can take as input
  a. The username, tag etc. used for generating NTuples on the The Worldwide LHC Computing Grid (WLCG)
  b. Local folder for local runs (as we just did)

```
Options:
  -h, --help            show this help message and exit
  -d DIRECTORY, --directory=DIRECTORY
                        Directory where files are downloaded/produced. If
                        option -g is 0 this option should point to the
                        directory of the submitDir directories from local
                        running.
  -t TAG, --tag=TAG     Tag used in grid processing. If option -g is 0 the tag
                        is used to identify the submitDir directory with the
                        given tag. If left empty it will process all submiDir
                        directories in the directory folder (specified by -d)
  -r PRIORITYTAG, --prioritytag=PRIORITYTAG
                        Comma seperated list of priority tags. If data set is
                        found with any of these tags use those in stead of the
                        ones with nominal tag specified through -t (--tag)
                        option. Tags are prioritized in the order they are
                        specified (first tag most important)
  -u UNAME, --uname=UNAME
                        Username of grid jobs (i.e. your CERN username)
  -o OUTDIR, --outdir=OUTDIR
                        Directory to store output ntuples
  -c NCPU, --ncpu=NCPU  Number of cpus to use in merging. Default: 0 (i.e.
                        single threaded)
  -j SPECFILE, --specfile=SPECFILE
                        If specification file already exists
  -s SKIM, --skim=SKIM  If want to makes skimmed samples (look in the
                        applySkim() function for all supported skims)
  -p PROC, --proc=PROC  Specify data or mc if only want to run one of them.
                        Default is all (i.e. both)
  -k CSV, --csv=CSV     Creates CSV files as output. Default is not to create
                        CSV. Be aware that this may cause memory issues if not
                        a very aggressive skimmming is applied (e.g. 4
                        leptons).
  -g GRID, --grid=GRID  If files comes from grid jobs. If False assuming
                        output is from local running
```

Creating an Education and Outreach Open Data NTuple

- Try to make an unskimmed ntuple
  - from the run directory:

```
python -i ../source/PhysliteToOpenData/scripts/skimOpenDataNtuples.py -g 0 -d ./ -o ./EOOD
```



Creating an Education and Outreach Open Data NTuple

- Skimming
  - Very easy to create new selections
  - Defined using ROOT's RDataFrame (see slides for an introduction)
  - Can use any of the variables in the ntuples to add new selection
  - You can add as many selection as you like

```
if skim == "exactly4lep":
    df_flat = df.Filter("n_lep == 4","Exactly 4 leptons")
elif skim == "4lep":
    df_flat = df.Filter("n_lep >= 4","At least 4 leptons")
elif skim == "3lep":
    df_flat = df.Filter("n_lep >= 3","At least 3 leptons")
elif skim == "2to4lep":
    df_flat = df.Filter("n_lep >= 2 && n_lep <= 4","2, 3 or 4 leptons")
elif skim == "exactly3lep":
    df_flat = df.Filter("n_lep == 3","Exactly 3 leptons")
elif skim == "2bjets70":
    df_flat = df.Filter("n_bjet70 >= 2","At least two b-jets at 70% WP")
elif skim == "2muons":
    df_flat = df.Filter("n_muo >= 2","At least two muons")
elif skim == "GamGam":
    df_flat = df.Filter("n_pho >= 2","At least two photons")
elif skim == "3J1LMET30":
    df_flat = df.Filter("n_lep >= 1","At least one lepton")
    df_flat = df.Filter("n_jet >= 3","At least three jets")
    df_flat = df.Filter("met > 30","MET > 30 GeV")
elif skim == "2J2LMET30":
    df_flat = df.Filter("n_lep >= 2","At least two leptons")
    df_flat = df.Filter("n_jet >= 2","At least two jets")
    df_flat = df.Filter("met > 30","MET > 30 GeV")
elif skim == "1LMET30":
    df_flat = df.Filter("n_lep >= 1","At least 1 loose leptons")
    df_flat = df.Filter("met > 30","MET > 30 GeV")
elif skim == "qcdjet":
    df_flat = df.Filter("trigJ || trigMJ", "Passes jet or multijet triggers")
else:
    print("WARNING \t No filters applied!")
    df_flat = df.Filter("1.0","nofilter")
```

Name of the skim

```
python -i ../source/PhysliteToOpenData/scripts/skimOpenDataNtuples.py -g 0 -s 1LMET30 -d ./run/ -o ./run/EOOD
```

Making changes to the framework : Code structure

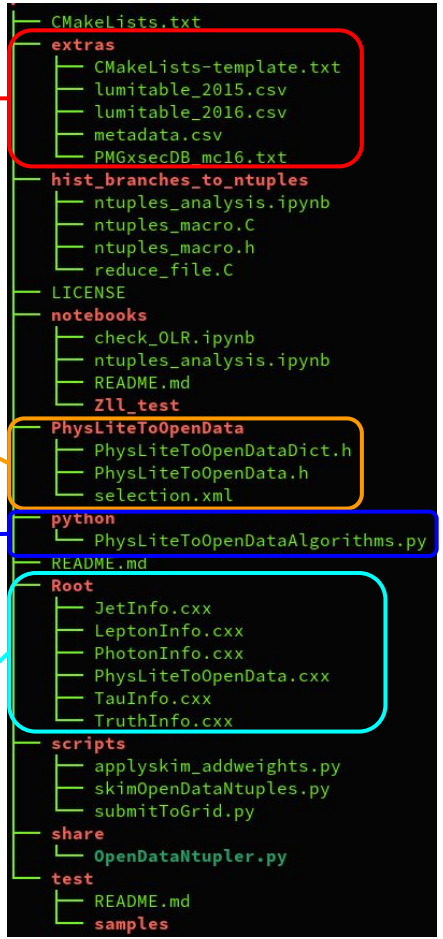
- Let's look at the content of the software repository
  - Will mainly focus on what's in the **Root**, **PhysliteToOpenData** and **python** folders

Metadata files

Header files

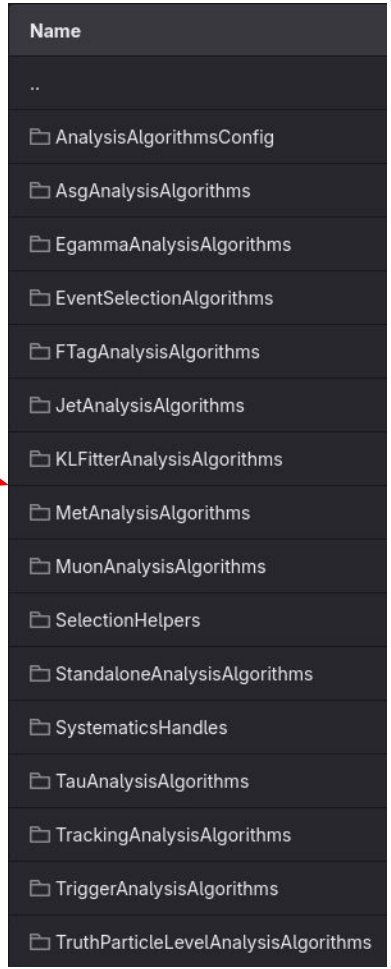
Main configuration file

Source files.
File for each object (Jet, Lepton, Photon, Tau, Truth)
Main file for event variables and initialization (*PhysliteToOpenData.cxx*)



Making changes to the framework: CP algorithms

- The ATLAS combined performance (CP) groups provide recommendations for use in analysis
  - Recommendations for data quality, pileup reweighting, calibrations and uncertainties for physics objects etc.
  - The CP groups provide tools to implement the recommendations wrapped up in ***CP algorithms***
  - the CP algorithms are centrally located in the ATLAS Athena code base
  - The algorithm blocks (to configure the tools) can be found in its respective directory under the python subdirectory and is called **AnalysisConfig.py*
- Useful links for more documentation
  - ATLAS Software Documentation on CP algorithms [link]
  - Example of the implementation of most available algorithm sequence blocks [link]



Making changes to the framework: Trigger example

In PhysLiteToOpenDataAlgorithms.py

```
# Include and setup trigger algorithm:
# https://gitlab.cern.ch/atlas/athena/-/blob/main/PhysicsAnalysis/Algorithms/T
configSeq += makeConfig( 'Trigger')
configSeq.setOptionValue('.triggerChainsForSelection', triggerChains)
configSeq.setOptionValue('.triggerMatchingChainsPerYear',triggerChainsPerYear)
configSeq.setOptionValue('.electronID', 'Tight')
configSeq.setOptionValue('.electronIsol', 'FCTight')
configSeq.setOptionValue('.photonIsol', 'FixedCutTight')
configSeq.setOptionValue('.muonID', 'Medium')
configSeq.setOptionValue('.electrons', 'AnalysisElectrons')
configSeq.setOptionValue('.muons', 'AnalysisMuons')
configSeq.setOptionValue('.photons', 'AnalysisPhotons')
configSeq.setOptionValue('.taus', 'AnalysisTaus')
configSeq.setOptionValue('.triggerChainsPerYear',triggerChainsPerYear)
# Add the decorations, but no filtering done based on the triggers
configSeq.setOptionValue('.noFilter',True)

configSeq.printOptions()
```

```
triggerChains = [
    #2015
    'HLT_2e12_lhloose_L12EM10VH',
    'HLT_2mu10',
    'HLT_mu18_mu8noL1',
    "HLT_e24_lhmedium_L1EM20VH",
    "HLT_e60_lhmedium",
    "HLT_e120_lhloose",
    "HLT_mu20_iloose_L1MU15",
    "HLT_mu40",
    "HLT_xe70_mht",
    "HLT_g120_loose",
    "HLT_g200_etcut",
    "HLT_tau80_medium1_tracktwo_L1TAU60",
    "HLT_tau35_medium1_tracktwo_tau25_medium1_tracktwo",
    #2016
    'HLT_2e17_lhvloose_nod0',
    'HLT_2mu14',
    'HLT_mu22_mu8noL1',
    "HLT_e26_lhtight_nod0_ivarloose",
    "HLT_e60_lhmedium_nod0",
    "HLT_e140_lhloose_nod0",
    "HLT_mu26_ivarmedium",
    "HLT_mu50",
    "HLT_xe90_mht_L1XE50",
    "HLT_xe110_mht_L1XE50",
    "HLT_g140_loose",
    "HLT_g300_etcut",
    "HLT_tau160_medium1_tracktwo",
    "HLT_tau35_medium1_tracktwo_tau25_medium1_tracktwo_L1TAU20IM_2TAU12IM"
]
```

In [TriggerAnalysisAlgorithms/python/TriggerAnalysisConfig.py]

```
def makeTriggerSelectionAlg(self, config, decisionTool):

    # Set up the trigger selection:
    alg = config.createAlgorithm( 'CP::TrigEventSelectionAlg', 'TrigEventSelectionAlg' )
    alg.tool = '%s/%s' % \
        ( decisionTool.getType(), decisionTool.getName() )
    alg.triggers = self.triggerChainsForSelection
    alg.selectionDecoration = 'trigPassed'
    alg.noFilter = self.noFilter
    alg.noL1 = self.noL1

    for t in self.triggerChainsForSelection :
        t = t.replace(".", "p").replace("-", "_")
        config.addOutputVar ('EventInfo', 'trigPassed_' + t, 'trigPassed_' + t, noSys=True)
```

In Root/PhysliteToOpenData.cxx

```
// Di-electron trigger chains
if( m_event->eventInfo->auxdata< bool >("trigPassed_HLT_2e12_lhloose_L12EM10VH") ) m_trigDE = true;
// Di-muon trigger chains
if( m_event->eventInfo->auxdata< bool >("trigPassed_HLT_2mu10") ) m_trigDM = true;
if( m_event->eventInfo->auxdata< bool >("trigPassed_HLT_mu18_mu8noL1") ) m_trigDM = true;
```

Can access those variables in C++

Making changes to the framework: Adding loose taus

- I would like to be able to select *loose* taus in my NTuple
  - Implement the necessary changes so that a new variable ***tau_isLoose*** gets written to the NTuples
  - Need to change the *PhysLiteToOpenDataAlgorithms.py*, *Root/TauInfo.cxx* and *PhysLiteToOpenData/PhysLiteToOpenData.h*
  - TauAnalysisConfig.py
  - **Bonus**: Add also scale factors for the loose taus

Tip when working in docker

Mount a local copy of the PhysLiteToOpenDataFramework when opening the container:

```
-v "$(pwd)/source/PhysLiteToOpenData":/local_files
```

Make changes to the local files and copy over to container repository. E.g.

```
cp /local_files/PhysLiteToOpenData/PhysLiteToOpenData.h /workdir/source/PhysLiteToOpenData/PhysLiteToOpenData/.
```

Then compile

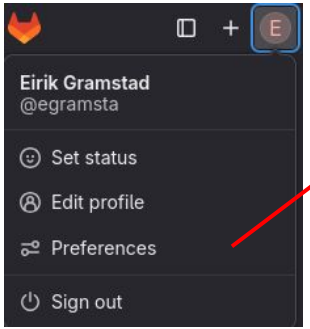
```
cd /workdir/build/
make
```

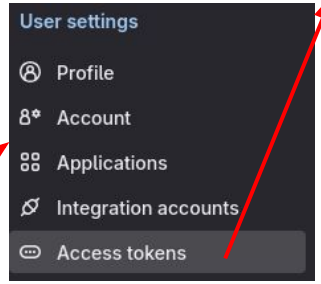
Making changes to the framework: Adding loose taus

- Need help?
  - Have a look here

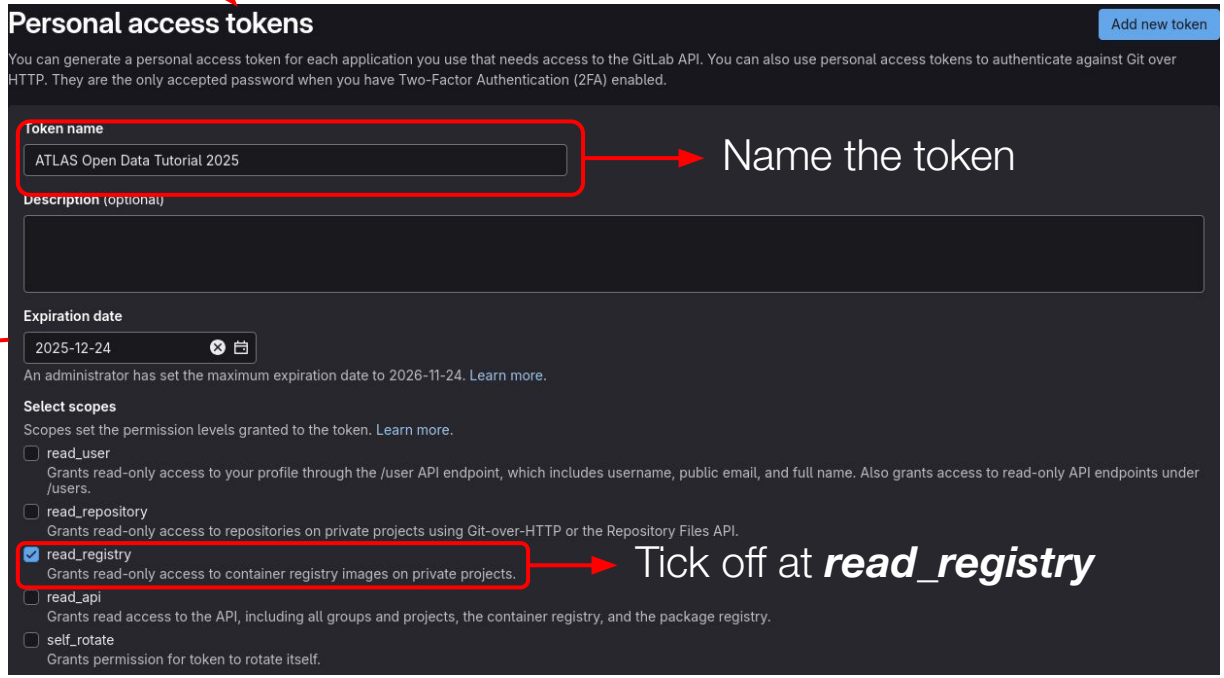
BACKUP

Acquire an access token





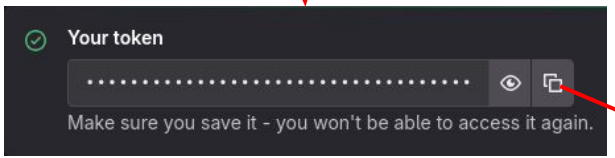
Add new token



Name the token

Tick off at ***read_registry***

Create token



Copy the token