Reminder: use of EDM4hep in FCC analyses

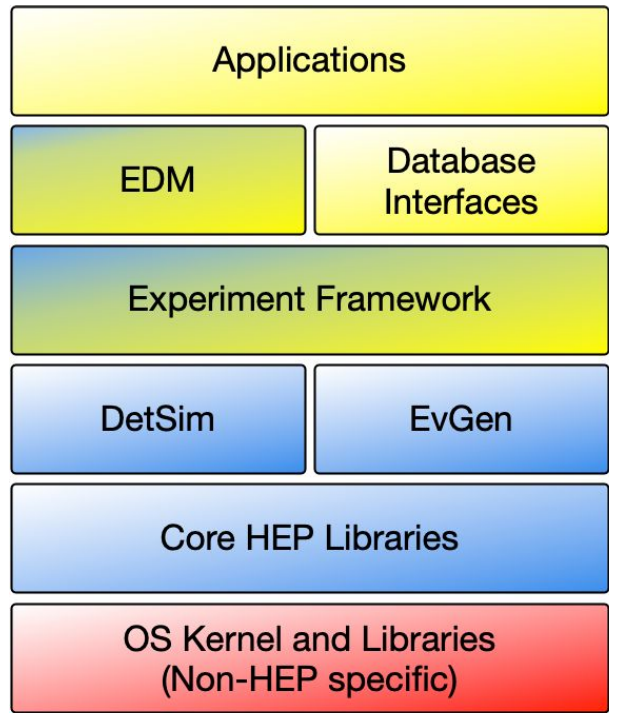
Juraj Smieško (CERN)

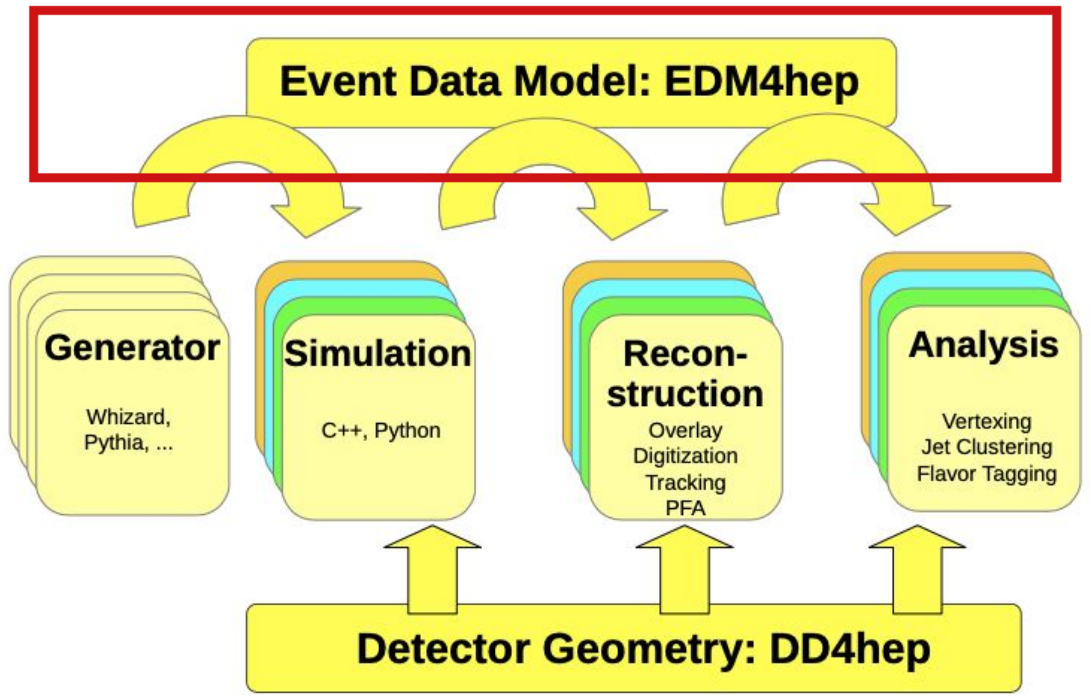
FCC Software Meeting

CERN, 26 Feb 2024

Key4hep

- Set of common software packages, tools, and standards for different Detector concepts
- Common for FCC, CLIC/ILC, CEPC, EIC, …
- Individual participants can mix and match their stack
- Main ingredients:
  - Data processing framework: Gaudi
  - Event data model: EDM4hep
  - Detector description: DD4hep
  - Software distribution: Spack



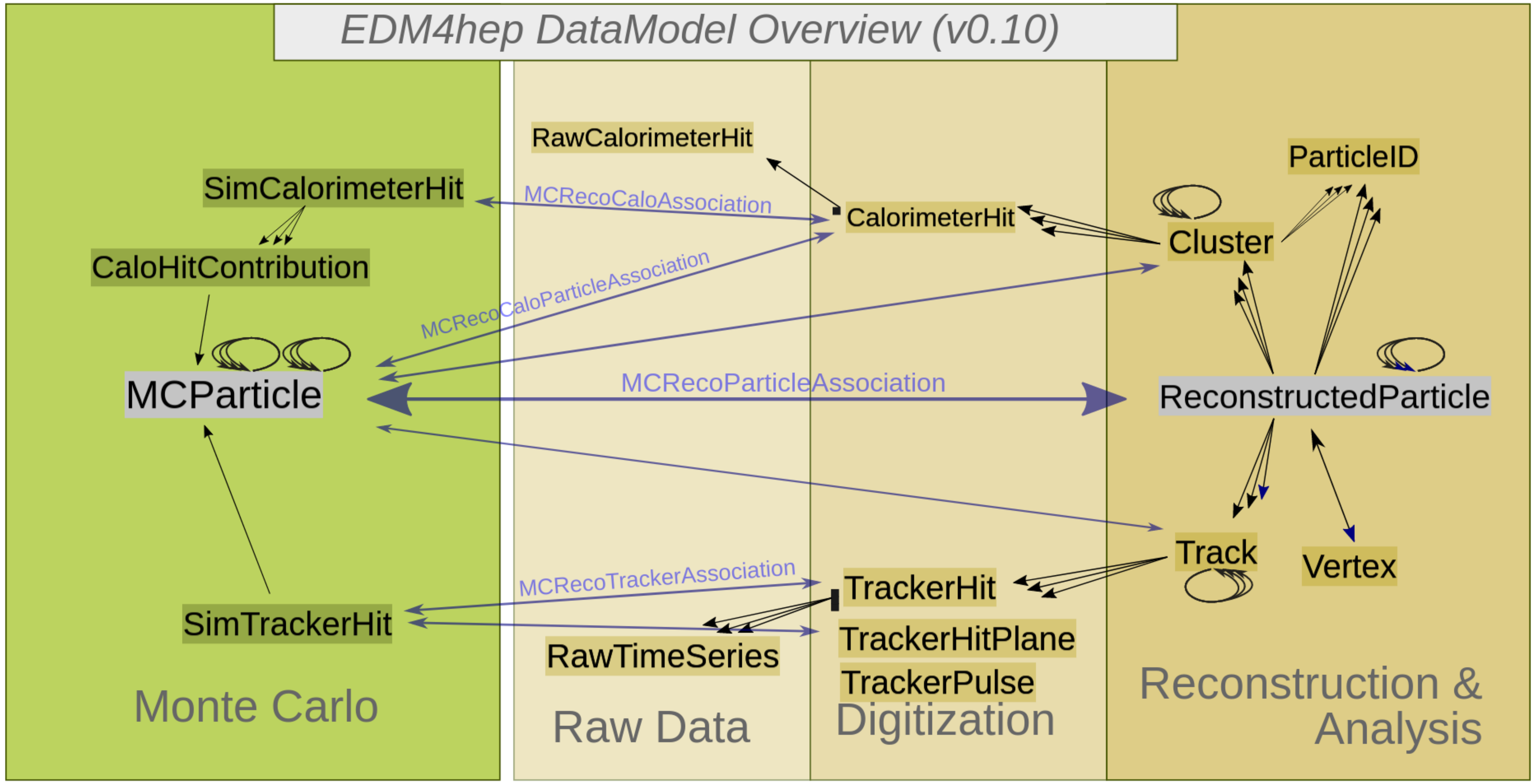


Source: Frank Gaede

EDM4hep I.

Describes event data with the set of standard objects.

- Specification in a single YAML file
- Generated with the help of Podio



EDM4hep II.

Example object:

```
#-------------  CalorimeterHit
edm4hep::CalorimeterHit:
  Description: "Calorimeter hit"
  Author: "EDM4hep authors"
  Members:
    - uint64_t cellID  // detector specific (geometrical) cell id
    - float energy [GeV]              // energy of the hit
    - float energyError [GeV]         // error of the hit energy
    - float time [ns]                // time of the hit
    - edm4hep::Vector3f position [mm] // position of the hit in world coordinates
    - int32_t type                   // type of hit
```

- Current version: `v0.10.5`
- Objects can be extended / new created
- Bi-weekly discussion: Indico

EDM4hep 1.0

The EDM4hep will reach version 1.0 soon, breaking changes and fixes are introduced.

Some of the changes/fixes underway:

- Interfaces
- `ReconstructedParticle.type` → `ReconstructedParticle.PDG`
- Reverse the direction of the ParticleID relation(s)
- Vector of weights in EventHeader

```
edm4hep::TrackerHit:
  Description: "Tracker hit interface class"
  Author: "Thomas Madlener, DESY"
  Members:
    - uint64_t cellID // ID of the sensor that created this hit
    - int32_t type // type of the raw data hit
    - int32_t quality // quality bit flag of the hit
    - float time [ns] // time of the hit
    - float eDep [GeV] // energy deposited on the hit
    - float eDepError [GeV] // error measured on eDep
    - edm4hep::Vector3d position [mm] // hit position
  Types:
    - edm4hep::TrackerHit3D
    - edm4hep::TrackerHitPlane
```

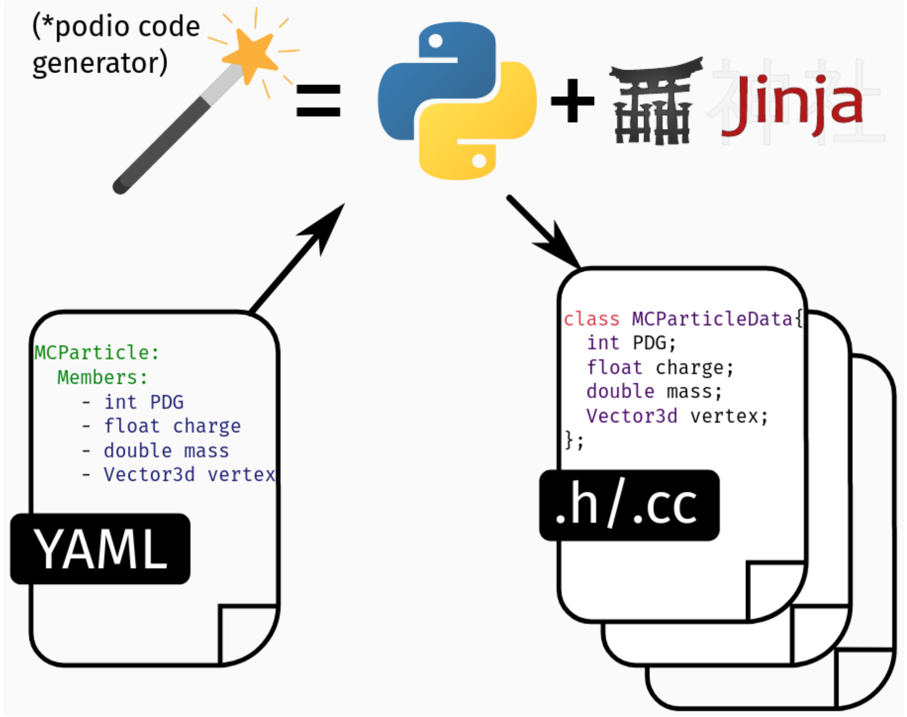
New release of FCCAnalyses 0.9 — preserves state before EDM4hep 1.0 changes

- Will arrive in stable Key4hep stack soon

Podio

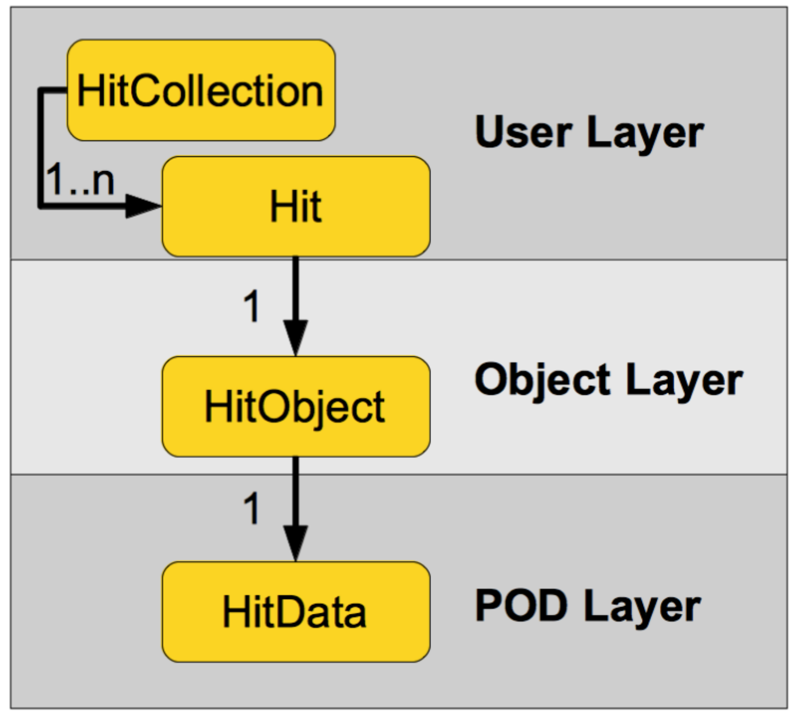
Generates Event Data Model and serves as I/O Layer

- Generates EDM from YAML files
- Employs plain-old-data (POD) data structures
- I/O machinery consists of three layers
  - POD Layer - actual data structures
  - Object Layer - helps resolve the relations
  - User Layer - full fledged EDM objects
- Supports multiple backends:
  - ROOT, SIO, ...
- Current version: `0.99`



```yaml
MCParticle:
  Members:
    - int PDG
    - float charge
    - double mass
    - Vector3d vertex
```

```cpp
class MCParticleData{
  int PDG;
  float charge;
  double mass;
  Vector3d vertex;
};
```

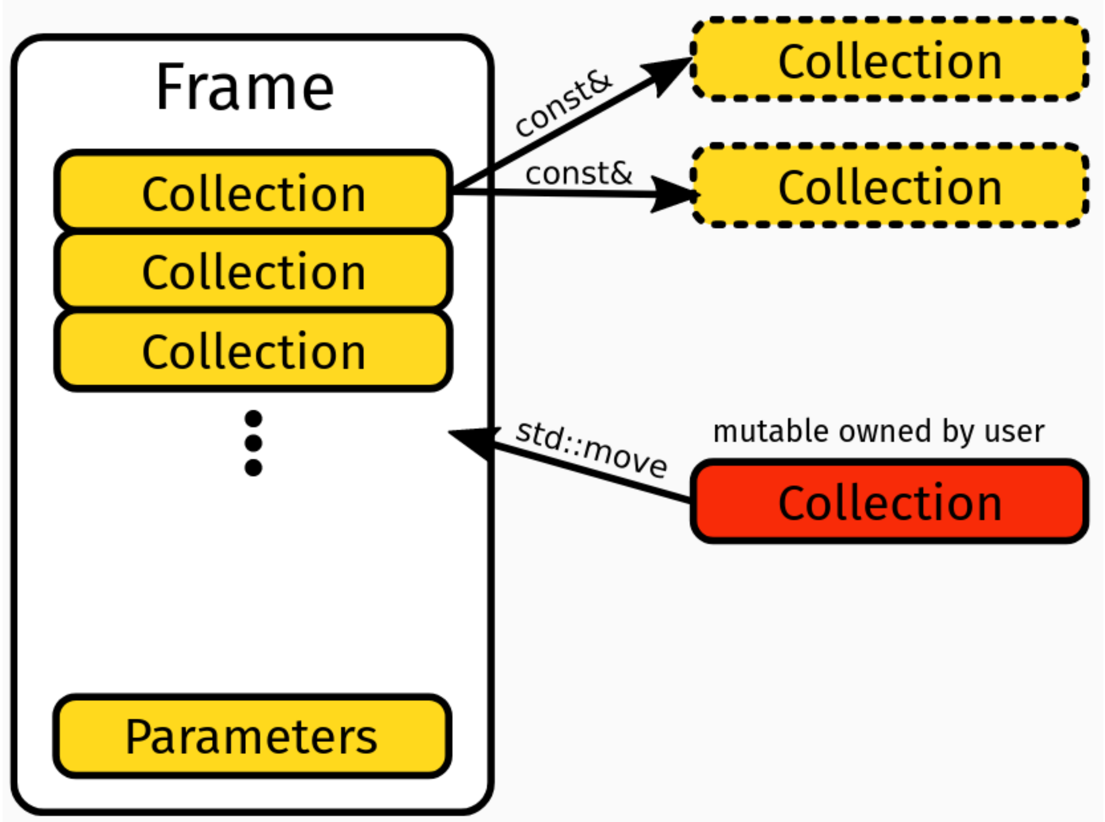


Podio Reader

Constructs the EDM4hep objects for the user

Example usage of Podio Reader in Pyhton:

```python
from podio.root_io import Reader
reader = Reader("one or many input files")
for event in reader.get("events"):
  hits = store.get("hits")
  for hit in hits:
    # ...
```



Datasets

Plethora of processes are pre-generated and available from EOS

Need to be reprocessed to be usable with EDM4hep 1.0

- Two main production campaigns in use:
  - Spring 2021 | EDM4hep `v0.3.1` | Podio `v0.13`
  - Winter 2023 | EDM4hep `v0.7.2` | Podio `v0.16.2`
- Samples are identified by their name, e.g.: `p8_ee_WW_ecm240`
- The production Database is browsable at:
  fcc-physics-events.web.cern.ch
- Example samples list:
  FCCee | Winter 2023 | IDEA | Delphes events
- EOS directory:
  `/eos/experiment/fcc/...`
- Generation handled by EventProducer
  - Heads up: Will change soon (Dirac, iLCDirac)

EOS Space

Intermediate analysis files of common interest can be stored at:

`/eos/experiment/fcc/ee/analyses_storage/...`

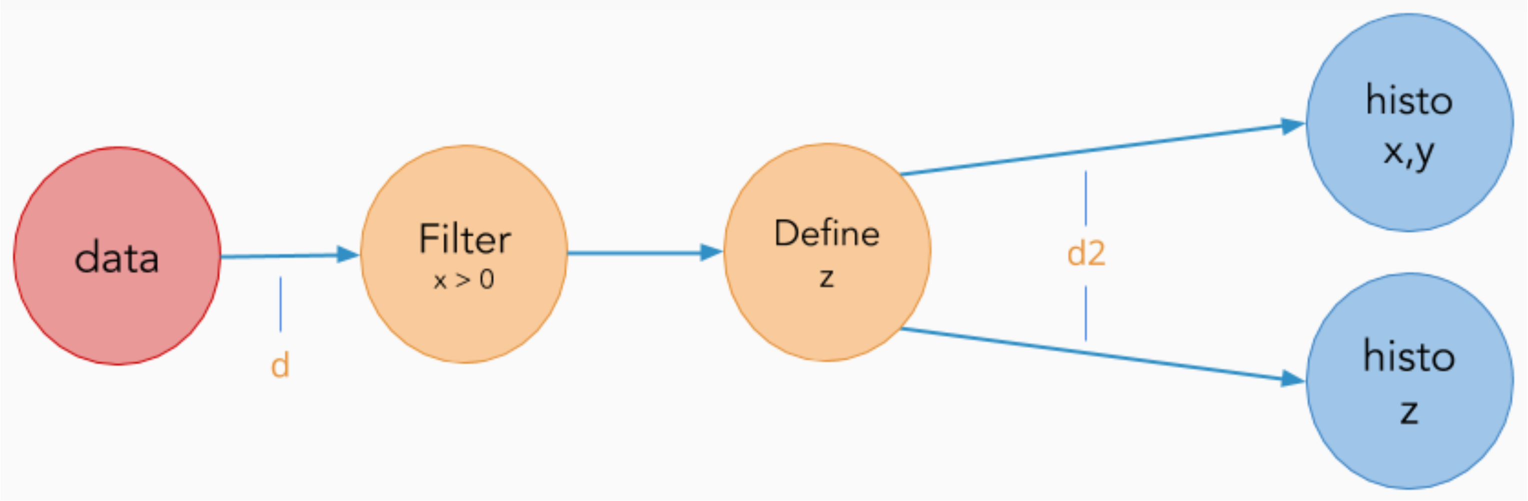
in four subfolders:

- BSM
- EW_and_QCD
- flavor
- Higgs_and_TOP

Access and quotas:

- Read access is is granted to anyone
- Write access needs to be granted: Ask your convener :)
- Total quota for all four directories is `200TB`
- ATM only part of the quota is allocated

ROOT RDataFrame



- Describes processing of data as actions on table columns
    - Defines of new columns
    - Filter rules
    - Result definitions (histogram, graph)
- The actions are lazily evaluated
- Multi threading is available out of the box
- Optimized for bulk processing
- Allows integration of existing C++ libraries

Reading EDM4hep in RDataFrame

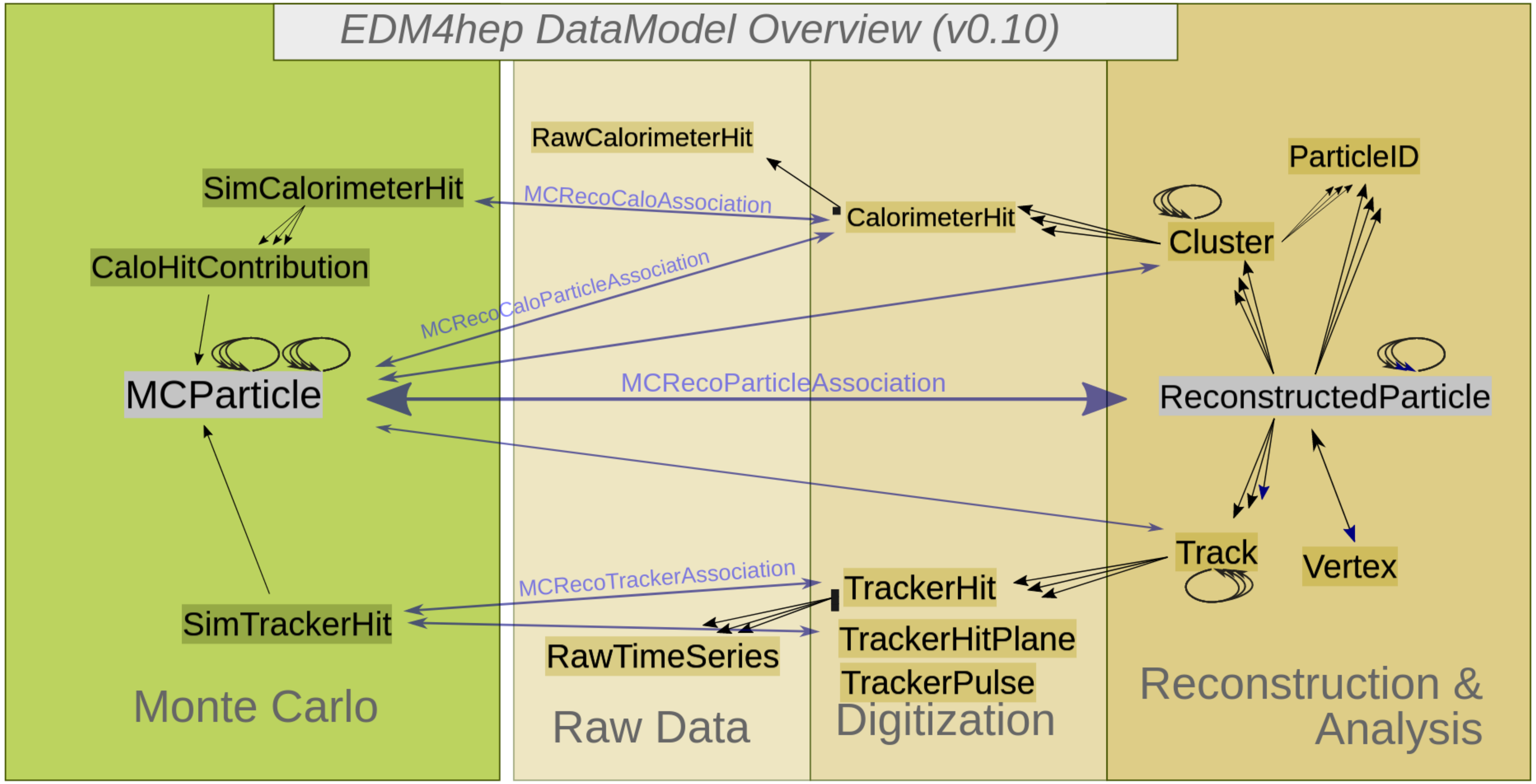
- EDM4hep collection is read in by RDataFrame directly and presented to the user in form:

```
const ROOT::VecOps::RVec<edm4hep::ReconstructedParticleData>& coll
```

  - This is per event
  - No convenient access to relationships
- Example of a simple function:

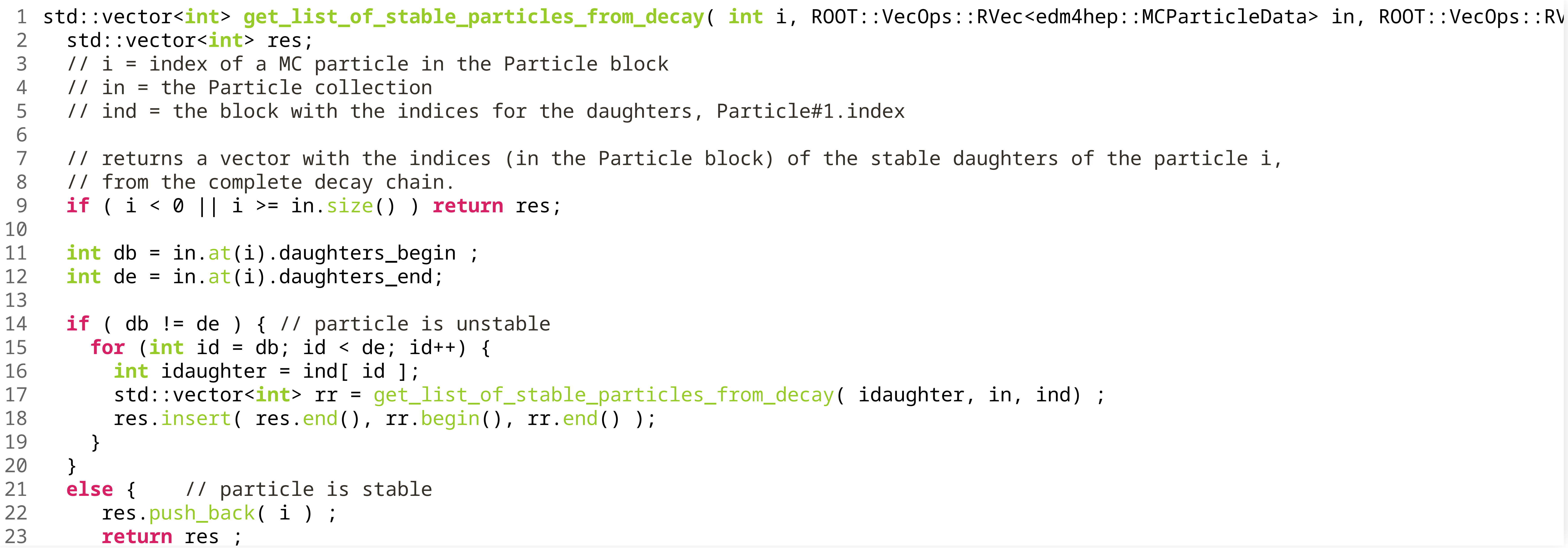
```
float getMass(const ROOT::VecOps::RVec<edm4hep::ReconstructedParticleData>& in) {
  ROOT::Math::LorentzVector<ROOT::Math::PxPyPzE4D<double>> result;

  for (auto & p: in) {
    ROOT::Math::LorentzVector<ROOT::Math::PxPyPzE4D<double>> tmp;
    tmp.SetPxPyPzE(p.momentum.x, p.momentum.y, p.momentum.z, p.energy);
    result+=tmp;
  }

  return result.M();
}
```

- In the course of the analysis the EDM4hep slowly decays into more trivial objects


EDM4hep DataModel Overview (v0.10)
RawCalorimeterHit
SimCalorimeterHit
MCRecoCaloAssociation
CalorimeterHit
ParticleID
Cluster
CaloHitContribution
MCRecoCaloParticleAssociation
MCParticle
MCRecoParticleAssociation
ReconstructedParticle
Track
Vertex
MCRecoTrackerAssociation
TrackerHit
SimTrackerHit
RawTimeSeries
TrackerHitPlane
TrackerPulse
Monte Carlo
Raw Data
Digitization
Reconstruction &
Analysis

Relations

- One collection can contain one-to-one or one-to-many relations to other collections, e.g.:
  - `CaloHit ⇶ CaloHitContribution`
  - `MCParticle ⇶ MCParticle`
- Typically relationships between derived objects (Sim. side separated from Reco. side)
- Example analyzer (FCC Tutorials link):



```cpp
std::vector<int> get_list_of_stable_particles_from_decay( int i, ROOT::VecOps::RVec<edm4hep::MCParticleData> in, ROOT::VecOps::RV
  std::vector<int> res;
  // i = index of a MC particle in the Particle block
  // in = the Particle collection
  // ind = the block with the indices for the daughters, Particle#1.index

  // returns a vector with the indices (in the Particle block) of the stable daughters of the particle i,
  // from the complete decay chain.
  if ( i < 0 || i >= in.size() ) return res;

  int db = in.at(i).daughters_begin ;
  int de = in.at(i).daughters_end;

  if ( db != de ) { // particle is unstable
    for (int id = db; id < de; id++) {
      int idaughter = ind[ id ];
      std::vector<int> rr = get_list_of_stable_particles_from_decay( idaughter, in, ind) ;
      res.insert( res.end(), rr.begin(), rr.end() );
    }
  }
  else {     // particle is stable
     res.push_back( i ) ;
     return res ;
```

Associations

- One-to-one relationships between two collection types, e.g.:
  - MCParticle ↔ ReconstructedParticle
  - SimTrackerHit ↔ TrackerHit
- Relationships between Simulation and Reconstruction side
- Example analyzer: Association between RecoParticle and MCParticle (link):

```
ROOT::VecOps::RVec<int>
ReconstructedParticle2MC::getRP2MC_index(const ROOT::VecOps::RVec<int>& recind,
                                         const ROOT::VecOps::RVec<int>& mcind,
                                         const ROOT::VecOps::RVec<edm4hep::ReconstructedParticleData>& reco) {
  ROOT::VecOps::RVec<int> result;
  result.resize(reco.size(),-1.);
  for (size_t i=0; i<recind.size();i++) {
    result[recind.at(i)]=mcind.at(i);    // recind.at(i) is the index of a reco'ed particle in the ReconstructedParticl
                                         // mcind.at(i) is the index of its associated MC particle, in the Particle col
  }

  return result;
}
```

Documentation

Multiple sources of documentation

- FCC Tutorials: https://hep-fcc.github.io/fcc-tutorials/
  - Focused on providing a tutorial on a specific topic
- Code reference: https://hep-fcc.github.io/FCCAnalyses/doc/latest/
  - Provides details about implementation of individual analyzers
- Manual pages:
  - Info about commands directly in the terminal: `man fccanalysis`
- FCCAnalyses website, FCCSW website

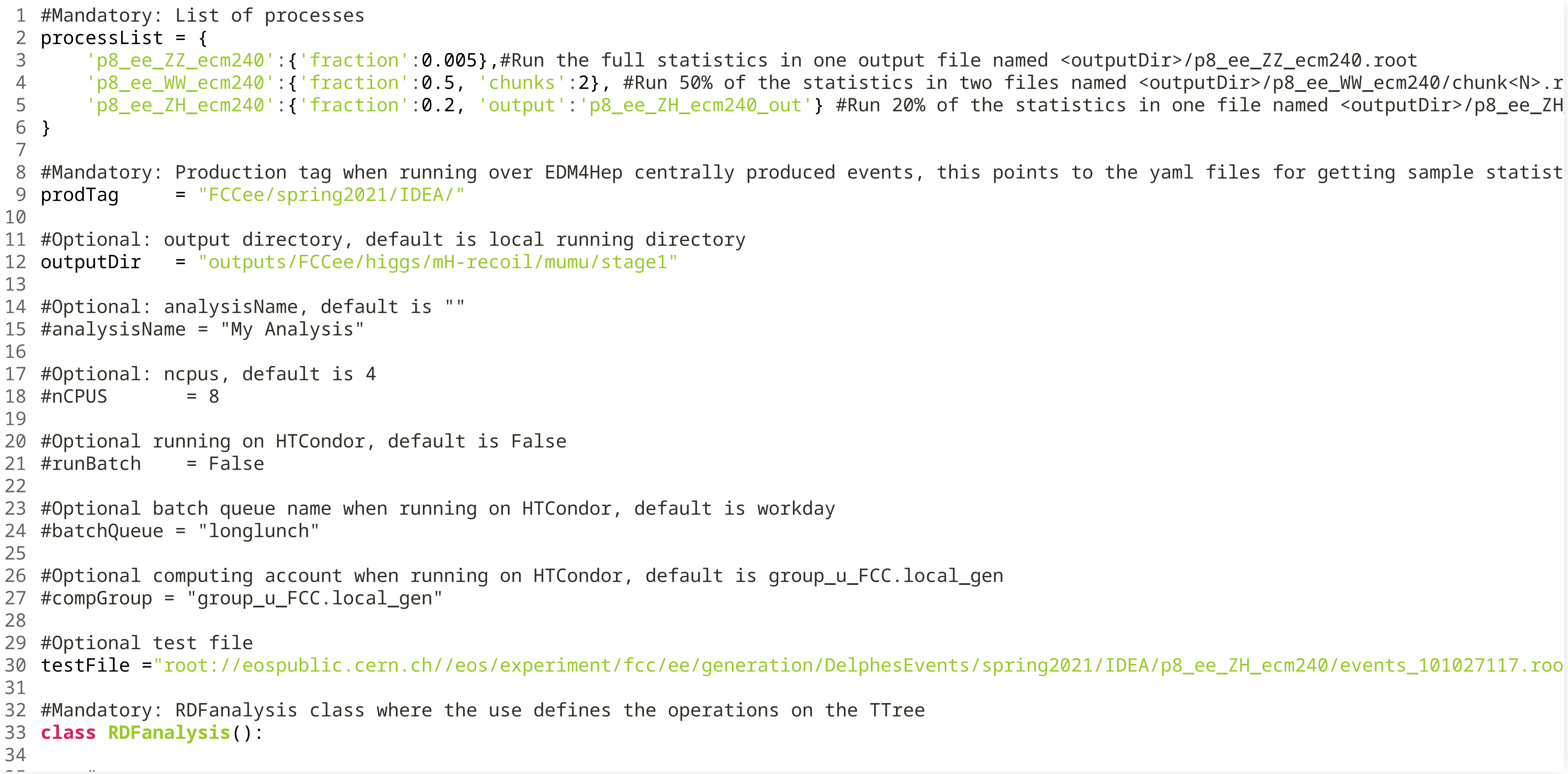
Conclusions

- Primary focus of EDM4hep is in Reconstruction
- Current strategy in FCCAnalyses is to slowly decay EDM4hep into more basic objects/structures
- To resolve relationships might require working with indexes across multiple collections
  - Remedy: EDM4hep RDataSource
- EDM4hep 1.0 is coming soon
  - All pre-generated samples will need to be reprocessed

Backup

Example analysis

The Higgs boson mass and σ(ZH) from the recoil mass with leptonic Z decays (link)



```
#Mandatory: List of processes
processList = {
    'p8_ee_ZZ_ecm240':{'fraction':0.005},#Run the full statistics in one output file named <outputDir>/p8_ee_ZZ_ecm240.root
    'p8_ee_WW_ecm240':{'fraction':0.5, 'chunks':2}, #Run 50% of the statistics in two files named <outputDir>/p8_ee_WW_ecm240/chunk<N>.r
    'p8_ee_ZH_ecm240':{'fraction':0.2, 'output':'p8_ee_ZH_ecm240_out'} #Run 20% of the statistics in one file named <outputDir>/p8_ee_ZH
}

#Mandatory: Production tag when running over EDM4Hep centrally produced events, this points to the yaml files for getting sample statist
prodTag     = "FCCee/spring2021/IDEA/"

#Optional: output directory, default is local running directory
outputDir   = "outputs/FCCee/higgs/mH-recoil/mumu/stage1"

#Optional: analysisName, default is ""
#analysisName = "My Analysis"

#Optional: ncpus, default is 4
#nCPUS       = 8

#Optional running on HTCondor, default is False
#runBatch    = False

#Optional batch queue name when running on HTCondor, default is workday
#batchQueue = "longlunch"

#Optional computing account when running on HTCondor, default is group_u_FCC.local_gen
#compGroup = "group_u_FCC.local_gen"

#Optional test file
testFile ="root://eospublic.cern.ch//eos/experiment/fcc/ee/generation/DelphesEvents/spring2021/IDEA/p8_ee_ZH_ecm240/events_101027117.roo

#Mandatory: RDFanalysis class where the use defines the operations on the TTree
class RDFanalysis():

```