columnflow: Fully automated analysis via flow of columns over distributed resources

Marcel Rieger
on behalf of the
c/f -Team

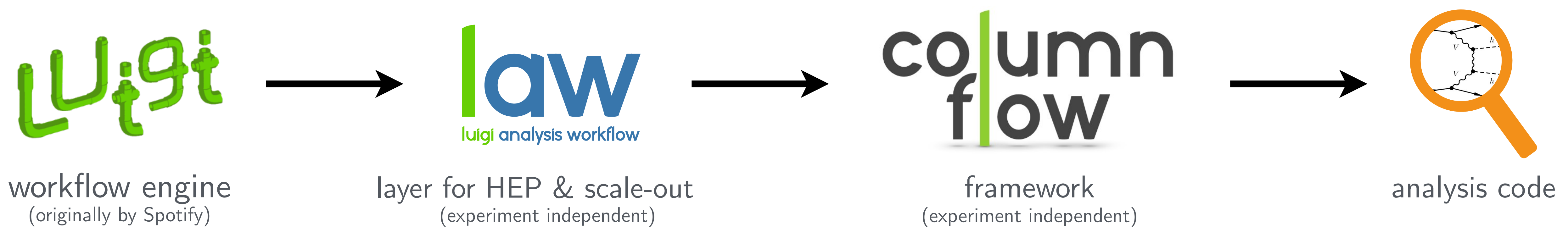
General idea

- Python-based framework for nano-like inputs
- End-to-end **orchestration** & **automation**
- **No reliance** on single local cluster or local storage
- Adapt to any remote cluster and storage system
  - ▻ HTCondor, Slurm, CMS-CRAB, LSF
  - ▻ Store via `file://`, `xrootd://`, `gsiftp://`, `webdav://`
- **Persistent intermediate outputs**
  - ▻ Debugging, reuse, sharing across groups

Key concepts

- Experiment **agnostic core**
  - ▻ Organize experiment-specific recipes in extensions
- Use awkward arrays as interface, parquet as file format
  - ▻ Give **users full control** over processing tools (NumPy, TensorFlow, coffea-nano-format, pandas, ...)
- High degree of **code-reuse** and collaboration
- Define **workflows** with luigi + **law**, metadata with order
- Control and execution via **CLI**, **scripts** and **notebooks**

Automation stack



Parallelization over ...

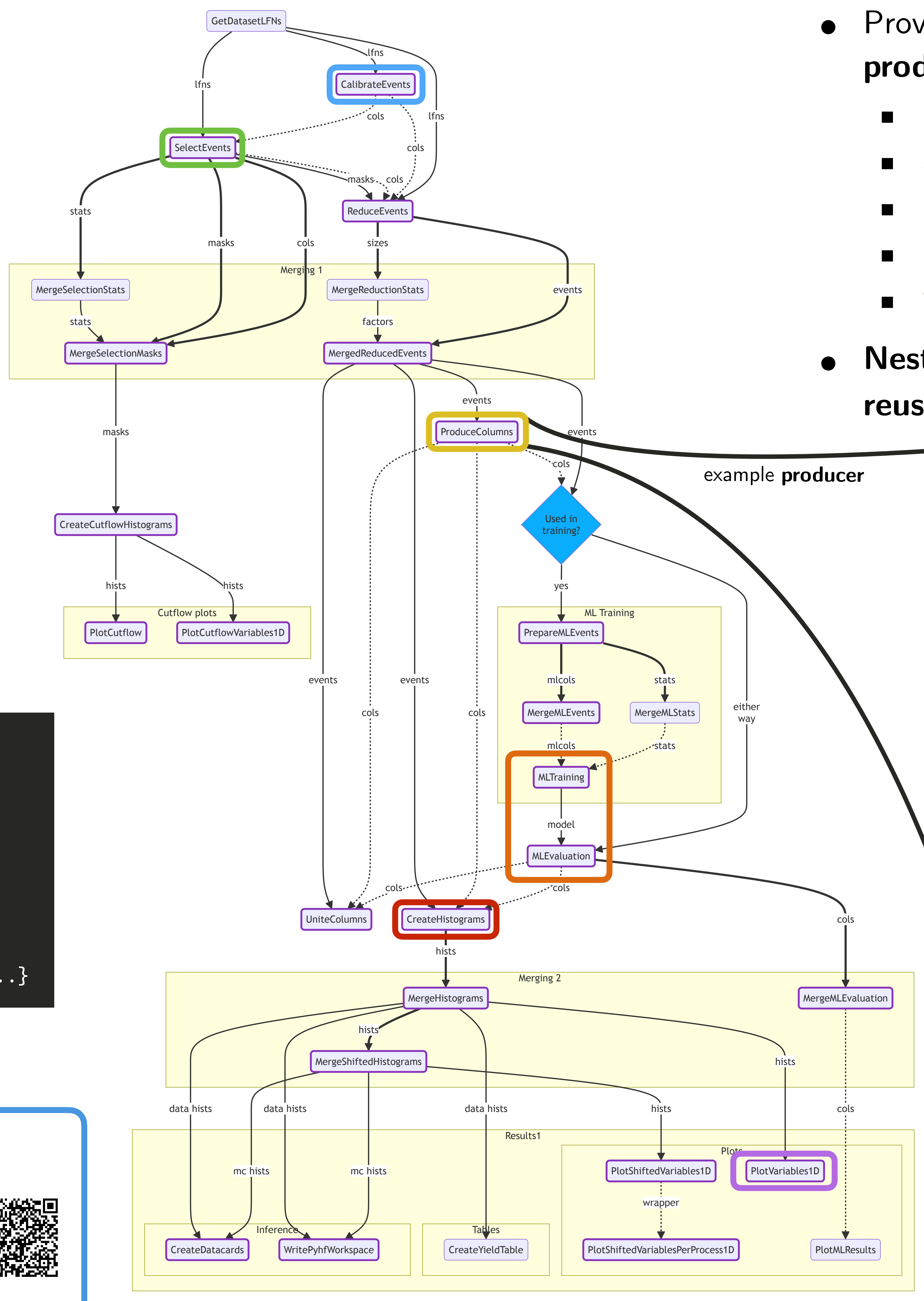
- Campaigns & datasets
- Files
- Systematics

▻ Typically $\mathcal{O}$(10k) 60min jobs, **however**, on **standard resources**

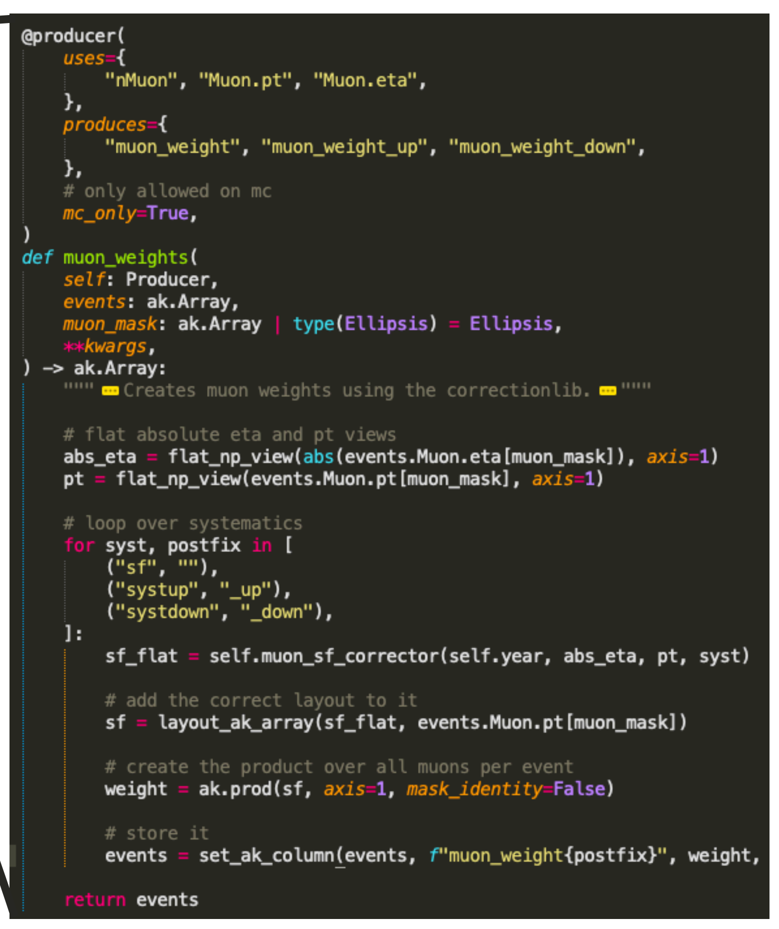
▻ HTCondor, CRAB, ...

Example graph*

(* Just a suggestion, can be easily altered or amended by analyses)



example **producer**



```python
@producer(
    uses={
        "nMuon", "Muon.pt", "Muon.eta",
    },
    produces={
        "muon_weight", "muon_weight_up", "muon_weight_down",
    },
    # only allowed on mc
    mc_only=True,
)
def muon_weights(
    self: Producer,
    events: ak.Array,
    muon_mask: ak.Array | type(Ellipsis) = Ellipsis,
    **kwargs,
) -> ak.Array:
    """ Creates muon weights using the correctionlib. """

    # flat absolute eta and pt views
    abs_eta = flat_np_view(abs(events.Muon.eta[muon_mask]), axis=1)
    pt = flat_np_view(events.Muon.pt[muon_mask], axis=1)

    # loop over systematics
    for syst, postfix in [
        ("sf", ""),
        ("systup", "_up"),
        ("systdown", "_down"),
    ]:
        sf_flat = self.muon_sf_corrector(self.year, abs_eta, pt, syst)

        # add the correct layout to it
        sf = layout_ak_array(sf_flat, events.Muon.pt[muon_mask])

        # create the product over all muons per event
        weight = ak.prod(sf, axis=1, mask_identity=False)

        # store it
        events = set_ak_column(events, f"muon_weight{postfix}", weight,

    return events
```

Simple customization

- Provide simple functions, **producers**, to create
  - ▪ **calibrated** (*updated*) **columns**
  - ▪ **selection masks**
  - ▪ **new columns**
  - ▪ **ML training & evaluation**
  - ▪ **variables**
- **Nesting** enables for easy **reuse** and **capsulation**

- Using bare **awkward arrays**
- Implementation and **choice of tools** fully **up to user**

Graph execution

- **Single command** can trigger the full pipeline from **inputs to plots**
- **Example**

```
> law run cf.PlotVariables1D \
    --version dev1 \
    --datasets ttbar,dy \
    --calibrators jec,jer \
    --selector full \
    --producers muon_weights \
    --variables jet*_{eta,pt} \
    --workflow {crab,htcondor,...}
```

Documentation

github.com/columnflow

columnflow.readthedocs.io