# Analysis Streamlining

D Costanzo, K Cranmer, L Heinrich, B Kersevan,
A Krasznahorkay, K Suruliz on behalf of the
ATLAS collaboration

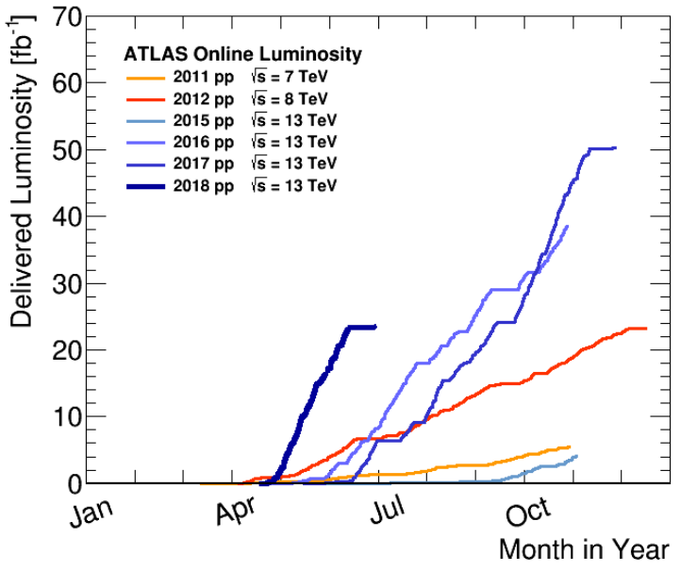
NEW YORK UNIVERSITY

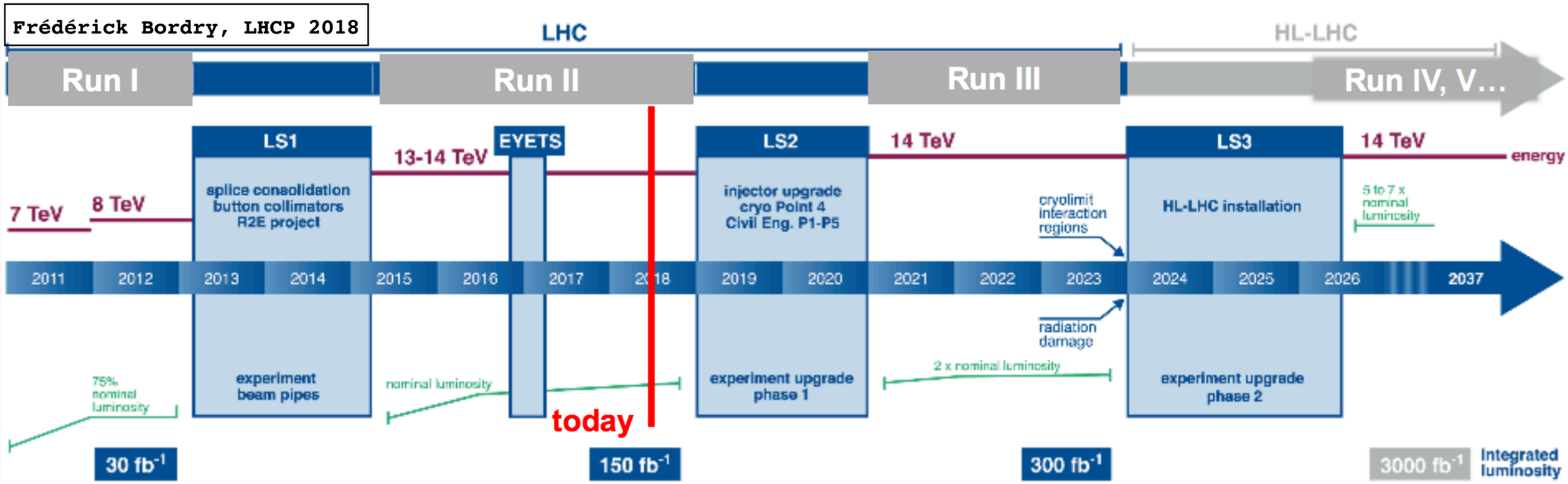
/lukasheinrich

# Analyses at the end of Run 2 of the LHC

The LHC and its experiments are entering an **evolutionary era**:

- no significant increase in √s
- focus is on collecting **luminosity**
  - invest R&D → reduce systematic uncertainties
- We have 10 years experience with analyzing collisions data — take stock of lessons learned
- **right time to streamline the way we do analyses**



# Motivations:

- **make analysis activity as smooth as possible**
  - focus on physics instead of mechanics of running the analysis
  - open up resources for R&D
  - not everyone needs to write their own data analysis framework
  - use well known (industry) tools where possible

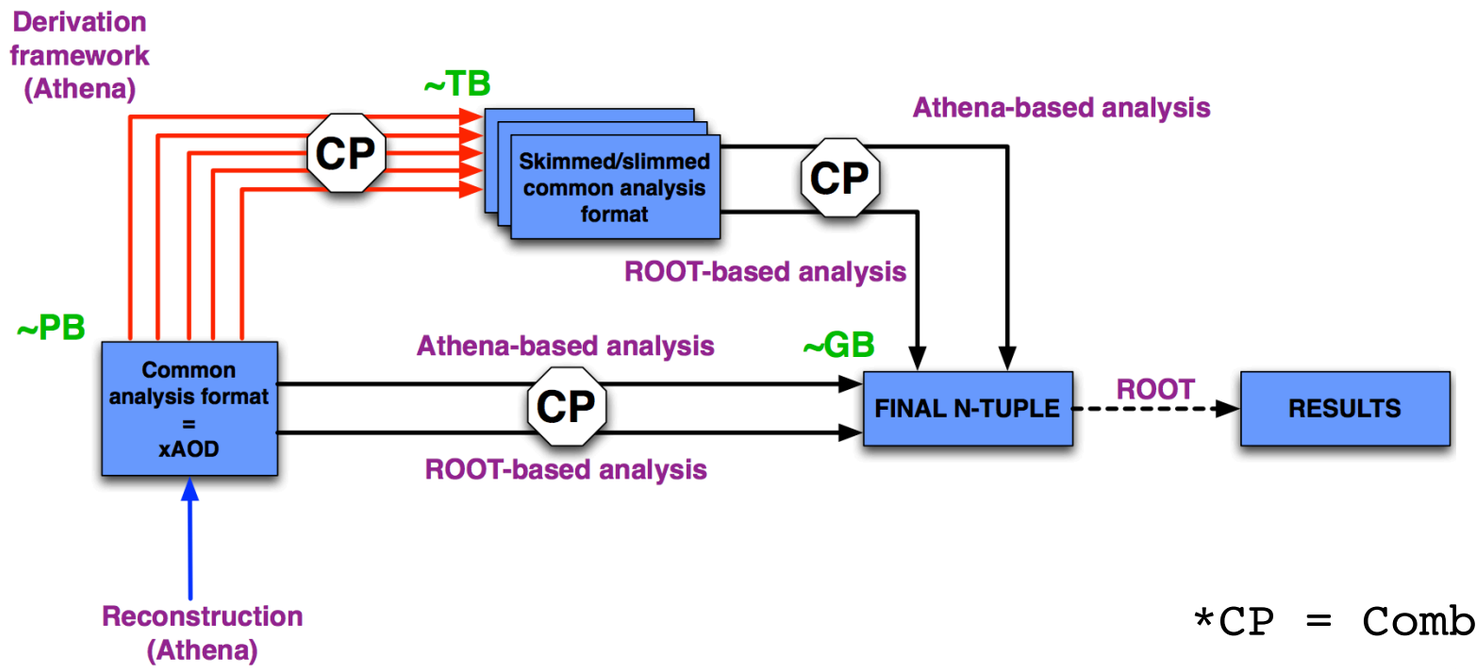


- **Easier Coordination**
  - integrate technical aspects of analyses (code, data, etc) into publication progress tracking

**Streamlined data-taking → streamlined analyses?**

- **Analysis Preservation:**
  - Analyses are valuable — **preserved them to reuse them for new physics and if needed reproduce original results**
  - need infrastructure to make it easy.

# The ATLAS Data Analysis Model




*CP = Combined Performance

After Run-1, ATLAS significantly redesigned its analysis model:

- **Goal:** simplify analysis by centralizing large portions, upstream work: **creating ntuples, applying cuts & calibrations**
  - highly successful **Derivation Framework**
  - works due to **common software stack, well-defined operations ~analysis independent.**
- **Next Goal: How can we harmonize the downstream, fundamentally analysis-dependent pieces of an analysis.**

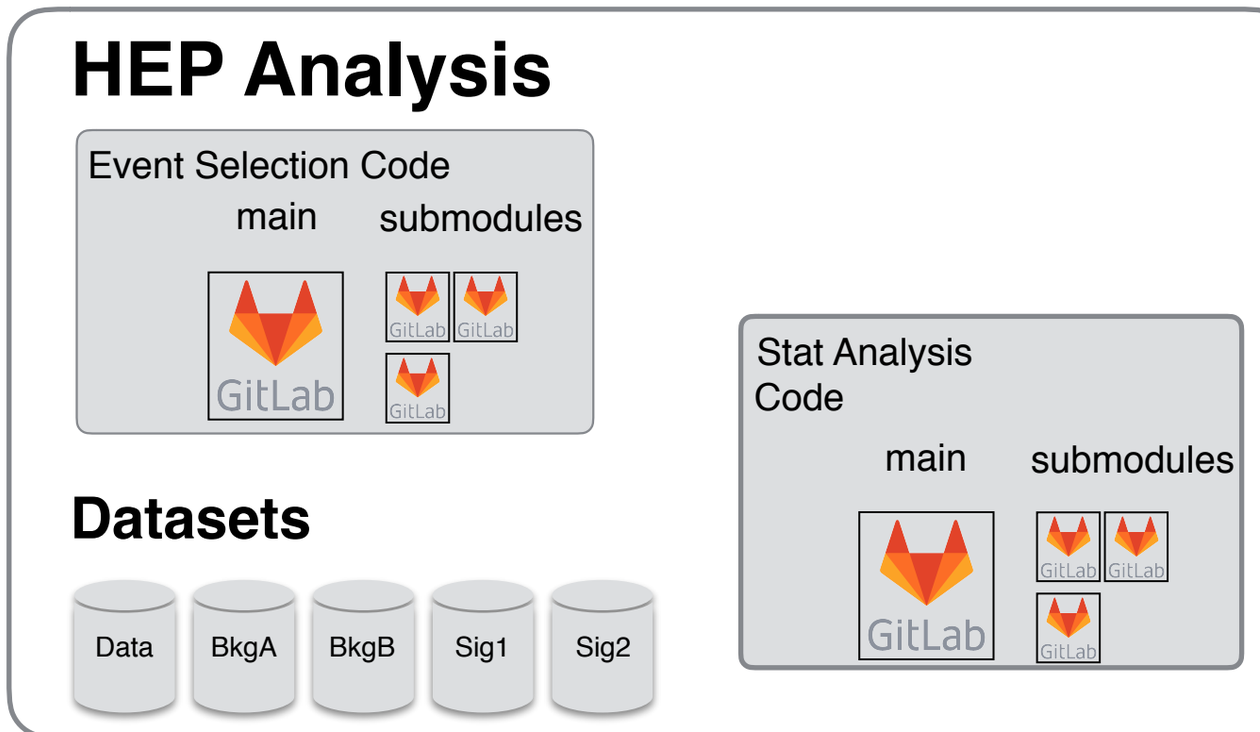
# Typical Analysis Code

**ATLAS** made big transition from custom tools to **git & cmake.** **CERN** phased out SVN — very good on-premise GitLab deployment

- Main reconstruction code in gitlab.cern.ch
- Many users naturally started using GitLab for their anlaysis as well
  - some experimenting with continuous integration
- **Clear Recommendation in ATLAS on what an analysis repository should look like:**
  - pull in dependencies as submodules (e.g. frameworks)

recursive clone == better experience than checking out needed packages by hand

- have as few “toplevel” repos as possible (e.g. if s/w stack is incompatible / very different)



# Enabling Preservation

- ATLAS has multiple flavors of software releases
  - Full Release: the entire offline software
    - mostly needed for reco / production jobs
  - Analysis Release: **slimmed down release**
    - **mainly ROOT + Event Data Model + CP/Physics Tools**

**Offline Release**
- **MC Generators**
- **Reco & Trigger**
- **...**

20GB

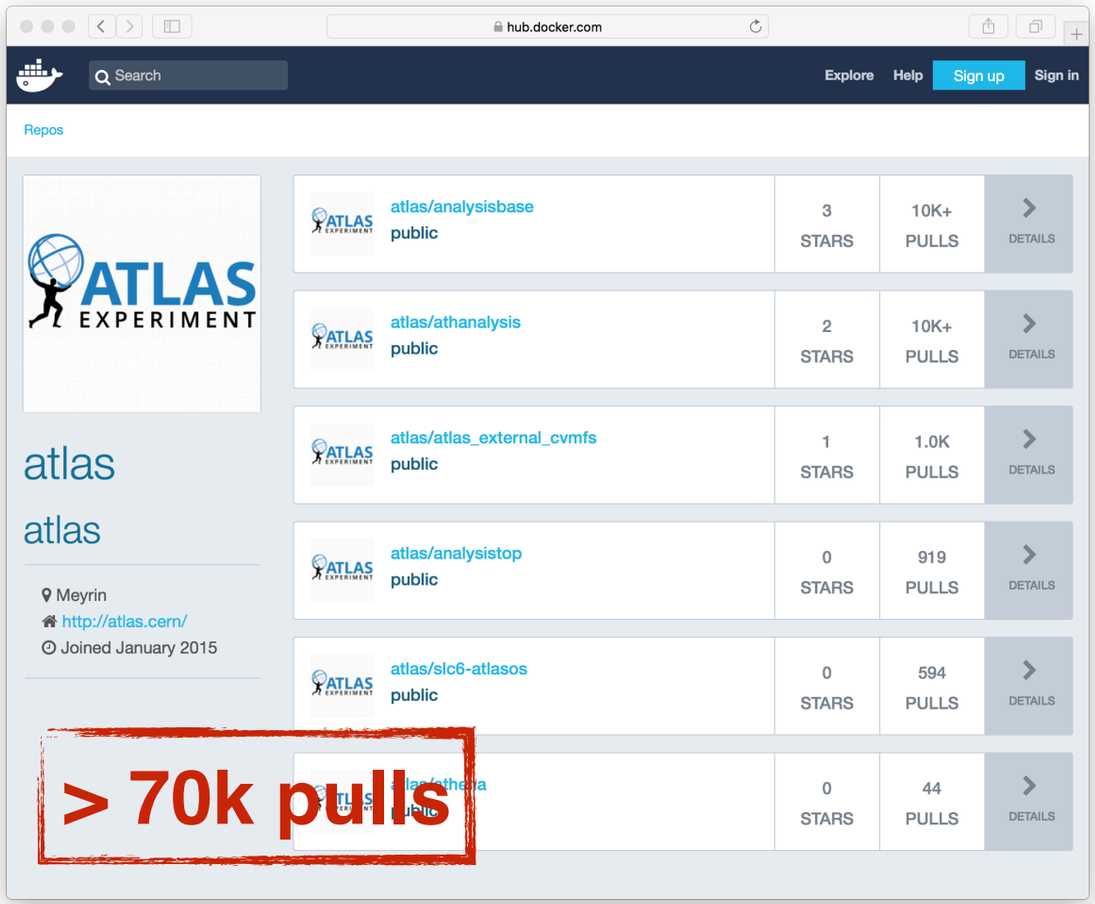
**Analysis Release**

2GB

- **ATLAS provides official Docker images of its analysis releases**
  - small size helps drive adoption
  - provides easy way to get reproducible software environment
  - Integration into **GitLab CI**
  - Preservation becomes **easy!**:

**Dockerfile** 190 Bytes

```
1  FROM atlas/athanalysis:21.2.23
2  ADD . /xampp/XAMPPmonoH
3  WORKDIR /xampp/build
4  RUN source ~/release_setup.sh &&  \
5      sudo chown -R atlas /xampp && \
6      cmake ../XAMPPmonoH && \
7      make -j1
```



# Take Repositories

## Your Analysis

Event Selection Code

main

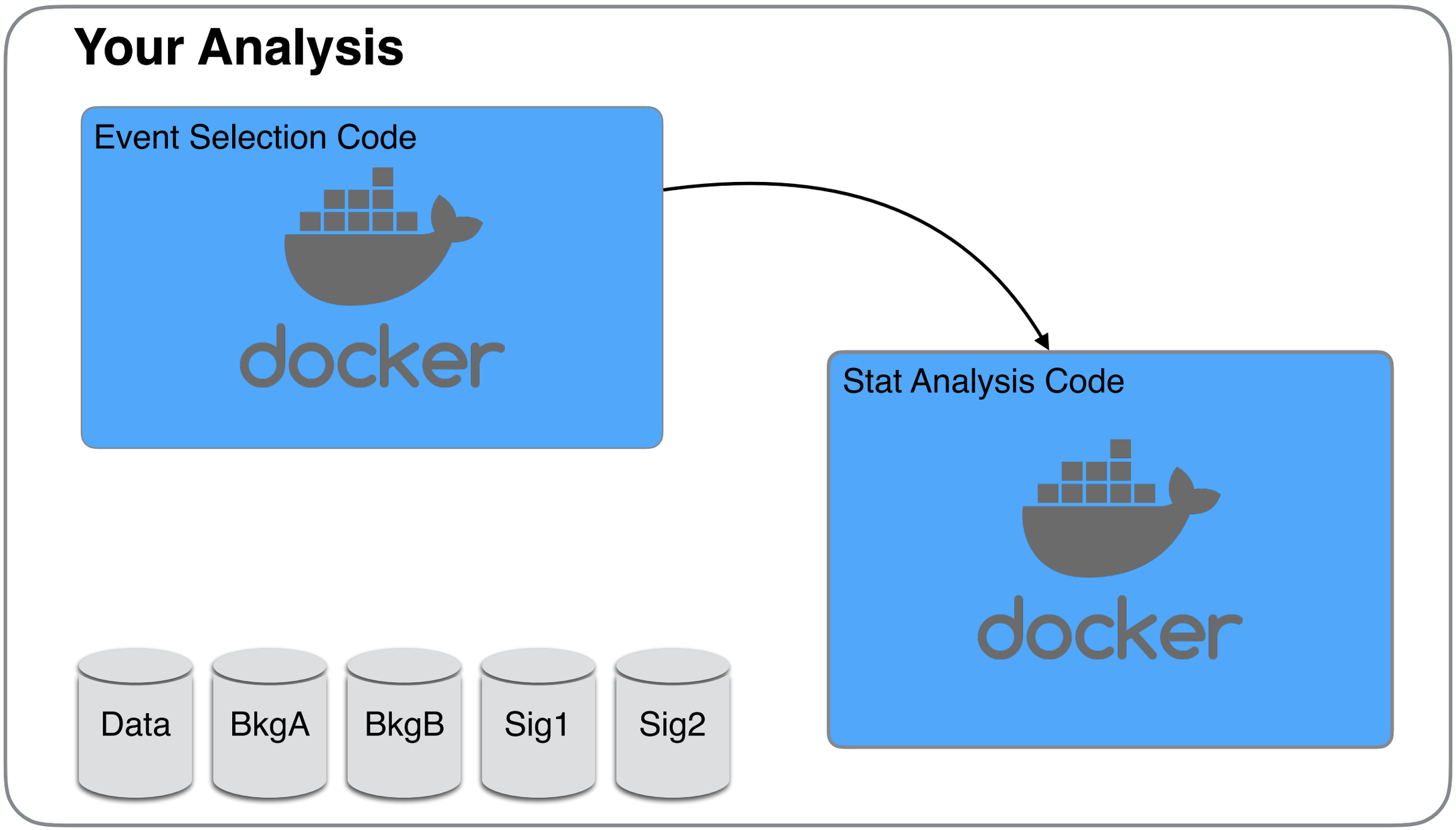

submodule


Stat Analysis Code

main


submodule


# Capture into self-consistent runtimes

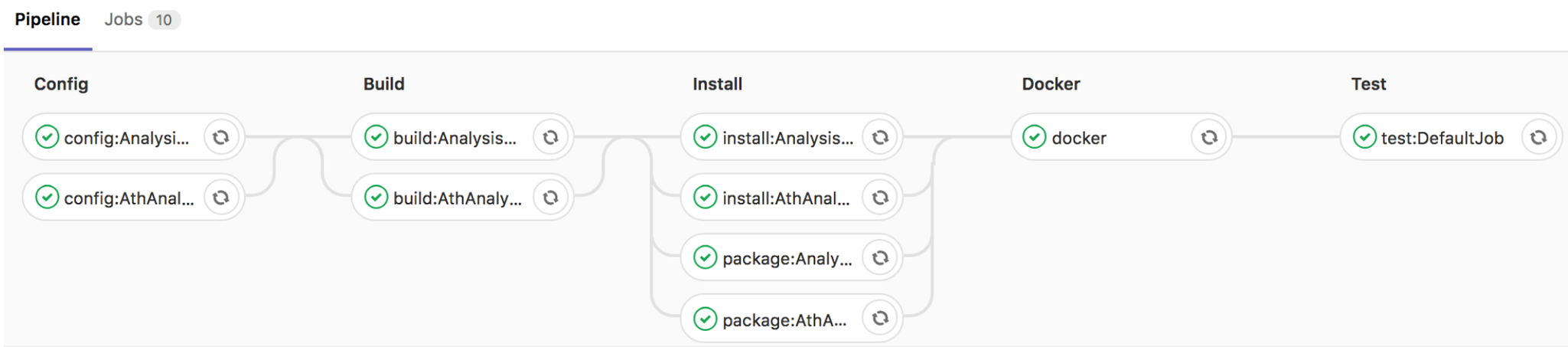


# Enabling Preservation — The Repo Template

Setting up CI and Preservaton by hand is still a barrier: **Use project templates to seed initial repository with analysis preservation built-in**

- uses popular template tool: **cookiecutter**
- **soon: GitLab integration for custom templates**
- nice default CI pipeline (build matrix, Image building, test jobs)
- building integration with internal Analysis Database (Glance)

```
cookiecutter atlas-asg/AnalysisRepositoryTemplate.git
```

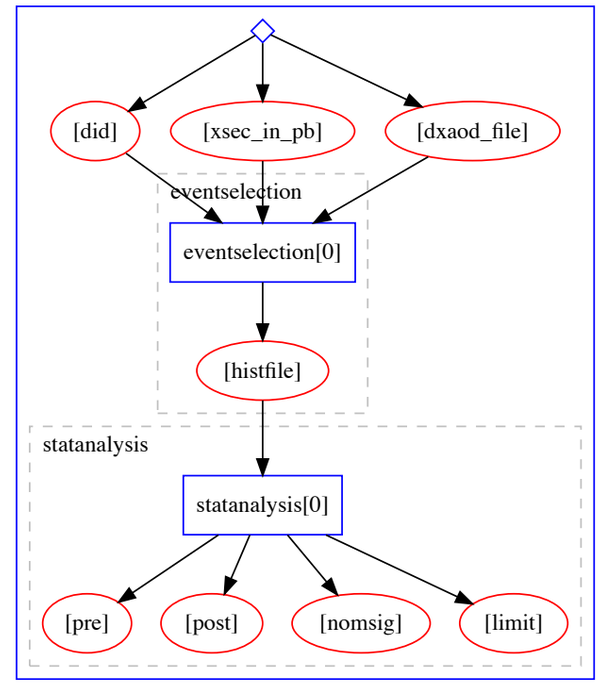



**Continuous Analysis Preservation — from the first commit**

NEW YORK UNIVERSITY

# Preserving Workflows

Just building images is not enough: need to preserve the workflows (**what to run & in what order)**

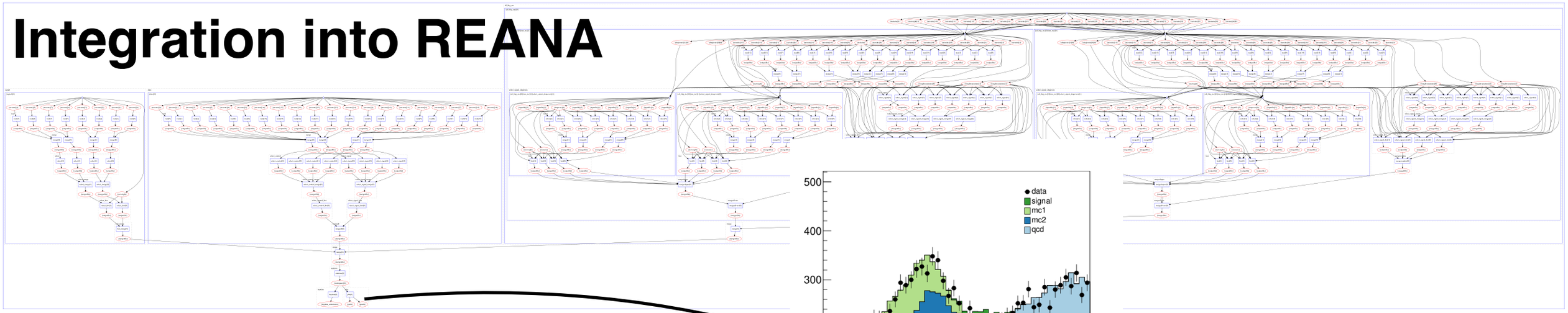



- **Users becoming more familiar with the concept thanks to e.g. CI pipelines**
- **workflow as graph of containerized analysis jobs**
- **Complex workflows (10k) jobs possible**
- **…but for many applications partial analysis preservation sufficient & easier**
  - **RECAST**

```
eventselection:
  environment: reanahub/reana-demo-atlas-recast-eventselection
  cmd: |
    source /home/atlas/release_setup.sh
    source /analysis/build/x86*/setup.sh
    cat << 'EOF' > recast_xsecs.txt
    id/I:name/C:xsec/F:kfac/F:eff/F:relunc/F
    {inp[init.did]} name {inp[init.xsec]} 1.0 1.0 1.0
    EOF
    echo {inp[init.url]} > recast_inputs.txt
    myEventSelection {workdir[submit]} recast_inputs.txt recast_xsecs.txt 30 #(=lumi)
  output:
    signalfile: workdir['submit/hist-sample.root']
statanalysis:
  environment: reanahub/reana-demo-atlas-recast-statanalysis
  cmd: |
    source /home/atlas/release_setup.sh
    python /code/make_ws.py /code/data/data.root {inp[eventselection.signalfile]} /code/data/background.root
    resultdir={workdir[fitresults]}
    mkdir -p $resultdir
    python /code/plot.py /code/results/meas_combined_meas_model.root $resultdir/pre.png $resultdir/post.png
    python /code/set_limit.py results/meas_combined_meas_model.root \
           $resultdir/limit.png $result/limit_data.json \
           $resultdir/limit_data_nomsignal.json
  output:
    limit:  workdir['fitresults/limit_data.json']
```

job/data dependency

# Integration into REANA

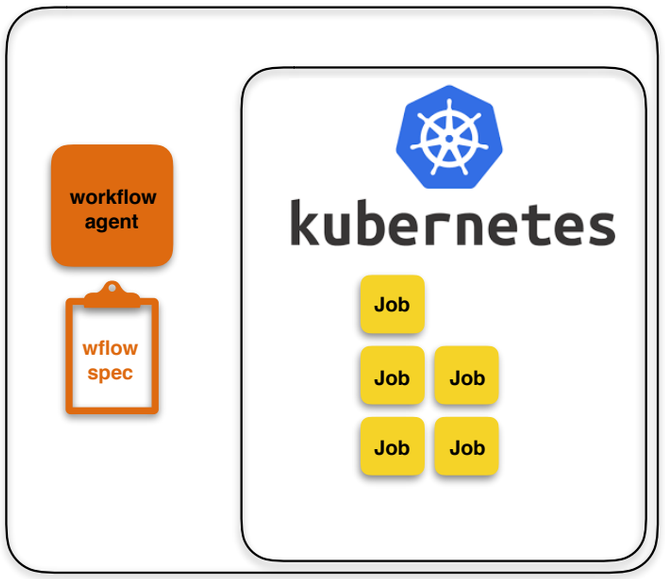


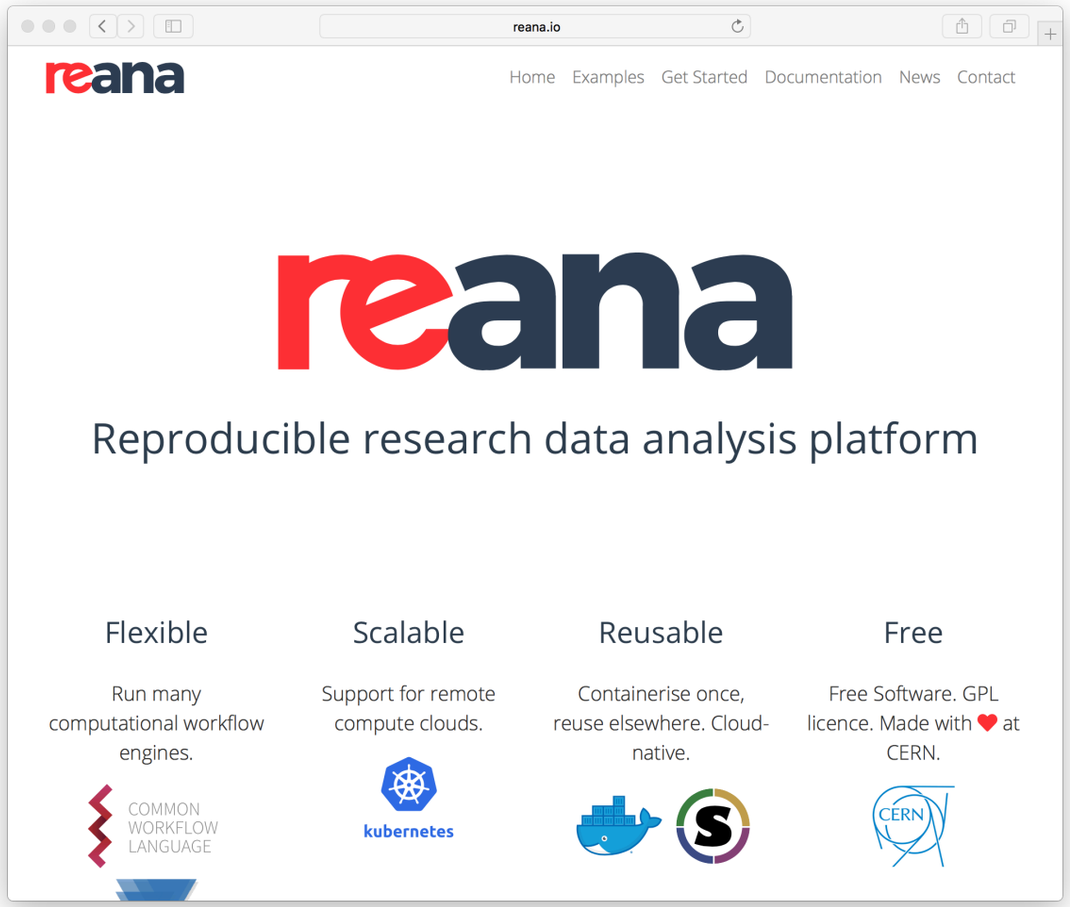
result of analysis
run on REANA




Close collaboration with REANA to provide service to users to run preserved analysis in the cloud.

- Leverage Workflows and **auto-built** Images
- cloud-native analysis platform
- multiple workflow languages



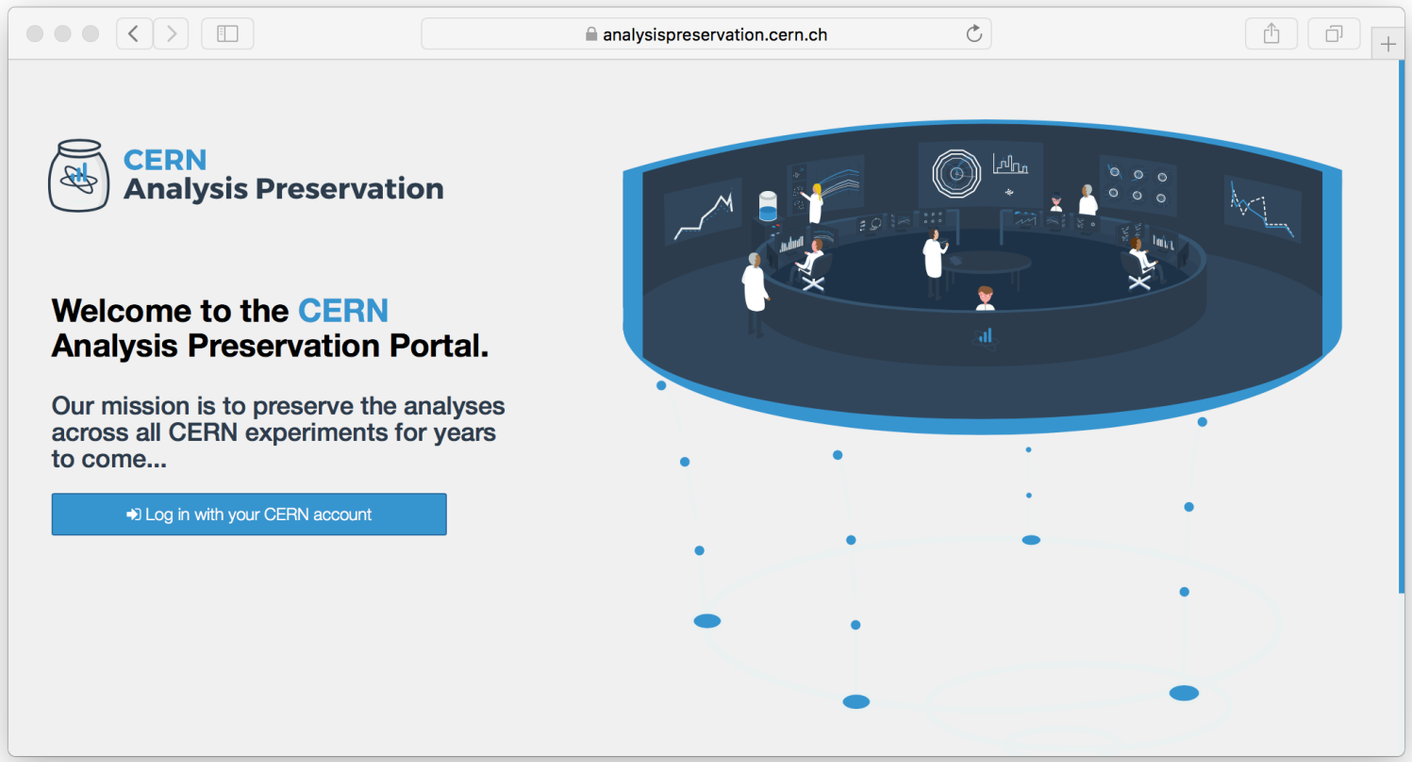




NEW YORK UNIVERSITY

# Integration into CERN Analysis Preservation

LHC-wide archive for preserved analyses. **Streamline ingestion.**

```
analysis: {

}
```

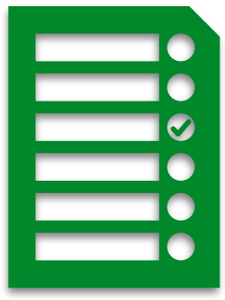


# Integration into CERN Analysis Preservation

LHC-wide archive for preserved analyses. **Streamline ingestion.**

auto-ingest
from existing DB

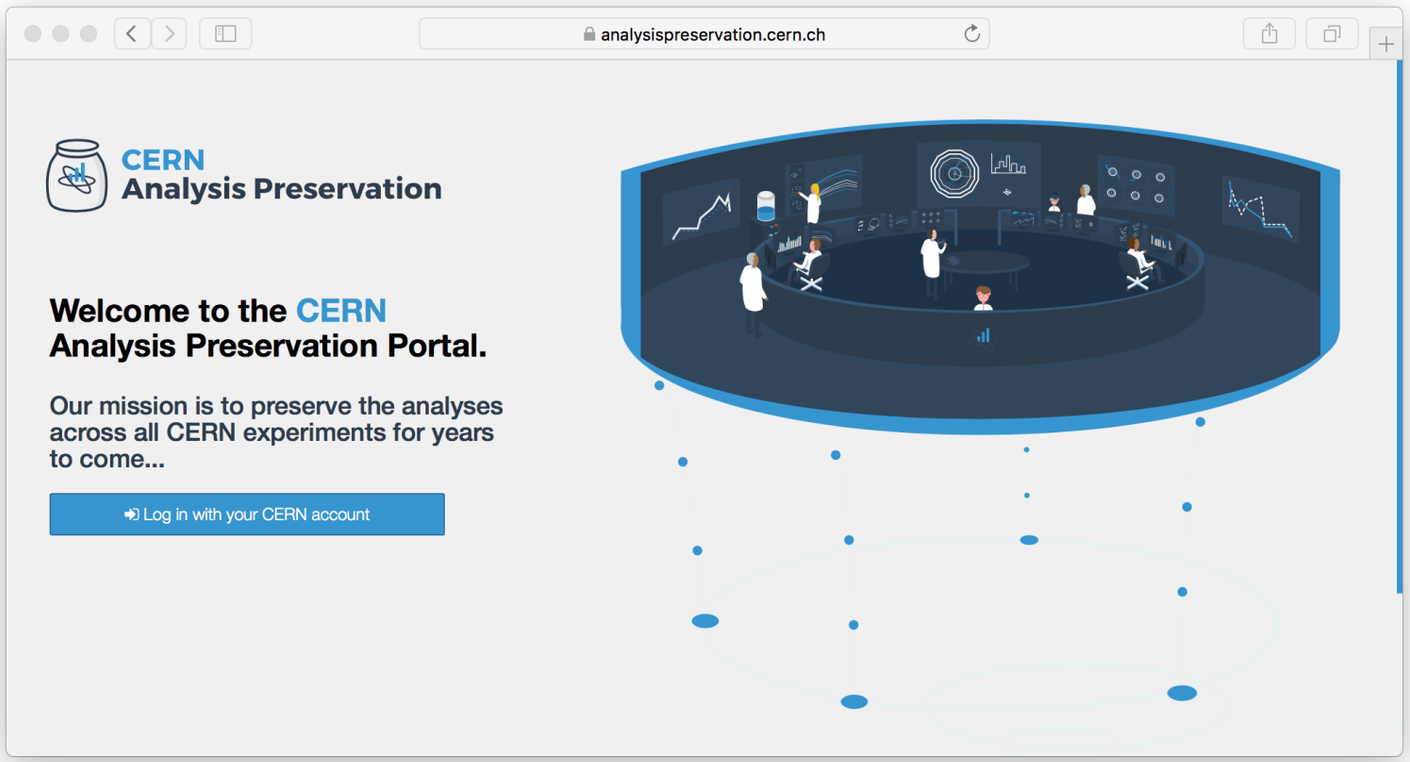
```
analysis: {

   metadata: {
      ...
   },

}
```



Glance
Record

**Metadata**

- **authors**
- **used methods**
- **...**




NEW YORK UNIVERSITY

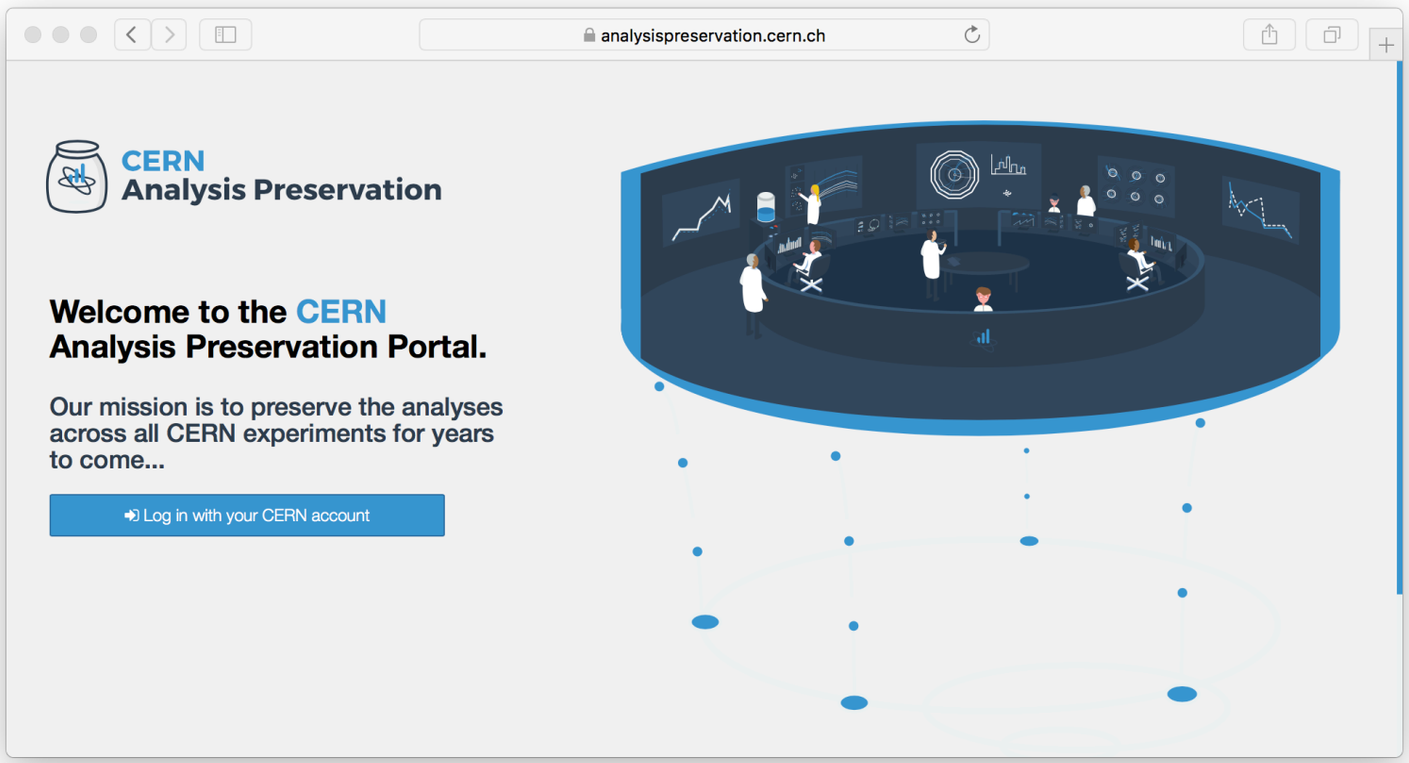
# Integration into CERN Analysis Preservation

LHC-wide archive for preserved analyses. **Streamline ingestion.**

```
analysis: {

   metadata: {
       ...
   },
   data: {
       ...
   },

}
```

ingest from group share (EOS)

**Data**
- **ntuples**
- **fixed aux meas.**
- **UFO models**
- **...**



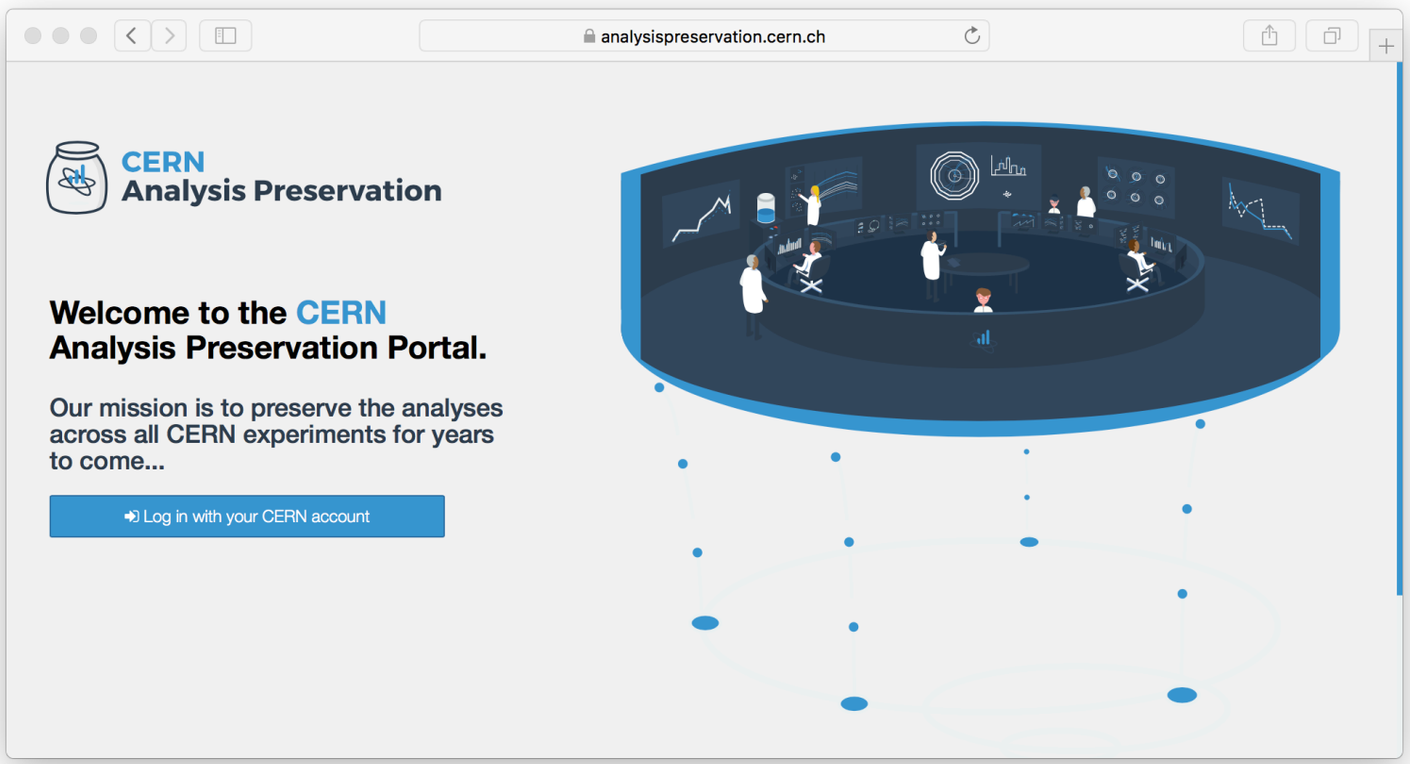
# Integration into CERN Analysis Preservation

LHC-wide archive for preserved analyses. **Streamline ingestion.**

```
analysis: {

   metadata: {
       ...
   },
   data: {
       ...
   },
   implementation: {
       ...
   }
}
```

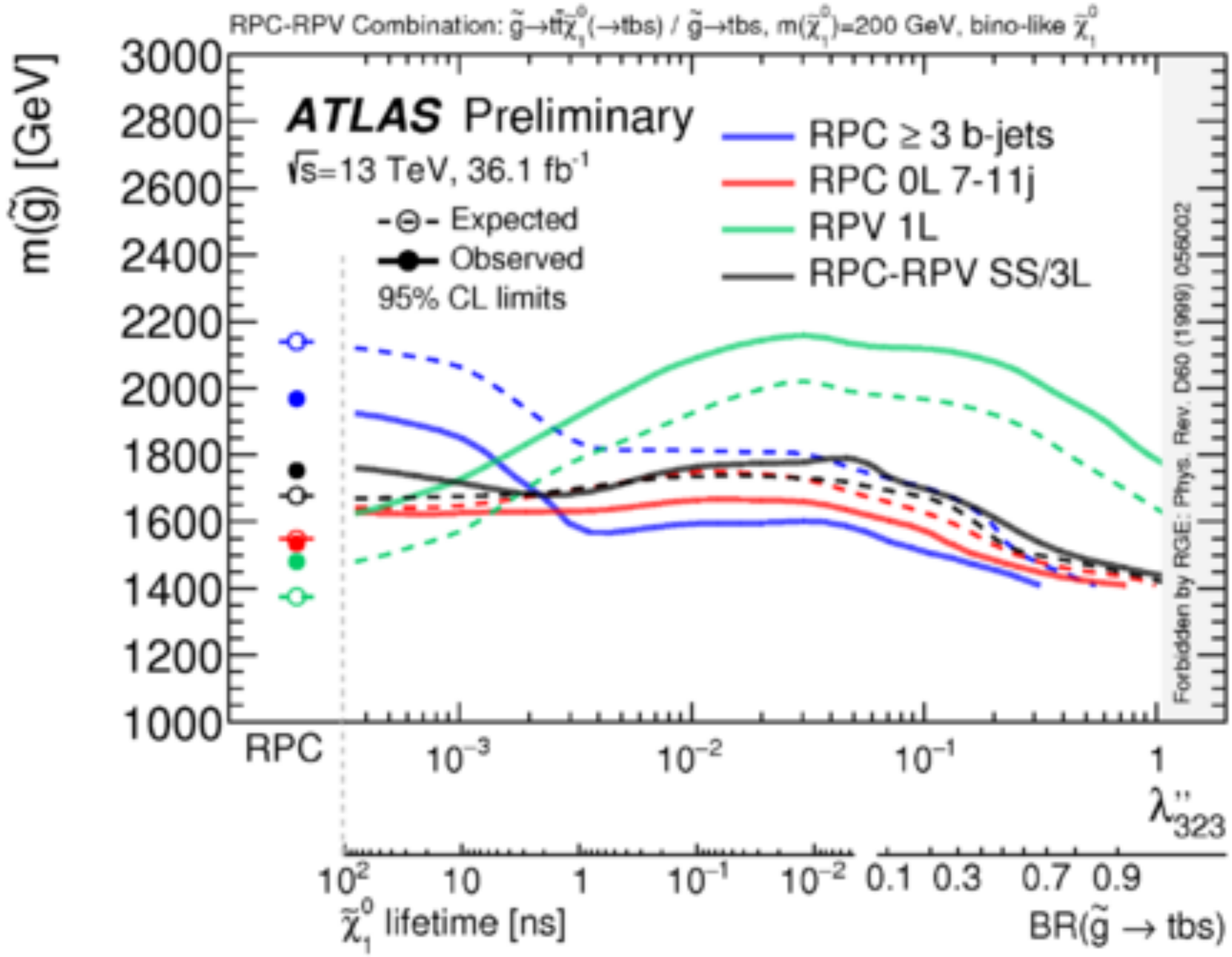

auto-ingest repo, docker images etc

**Implementation**
- **workflow**
- **docker images**
- **source code**




NEW YORK UNIVERSITY

# **Result:** New Physics from old analyses




Reinterpretation enabled by containerized, streamlined analyses

# The Future

## Streamline Distributed Computing Infrastructure for analyses

- develop code locally → Merge Request to repo
- image built in continuous integration system
- run container-based jobs on the grid

- ongoing work to enable containers on the grid
  - Hot idea: federated Kubernetes (= cloud-native grid)
- accelerate commit → build → run cycle to be painless