RECAST, REANA, and all that.

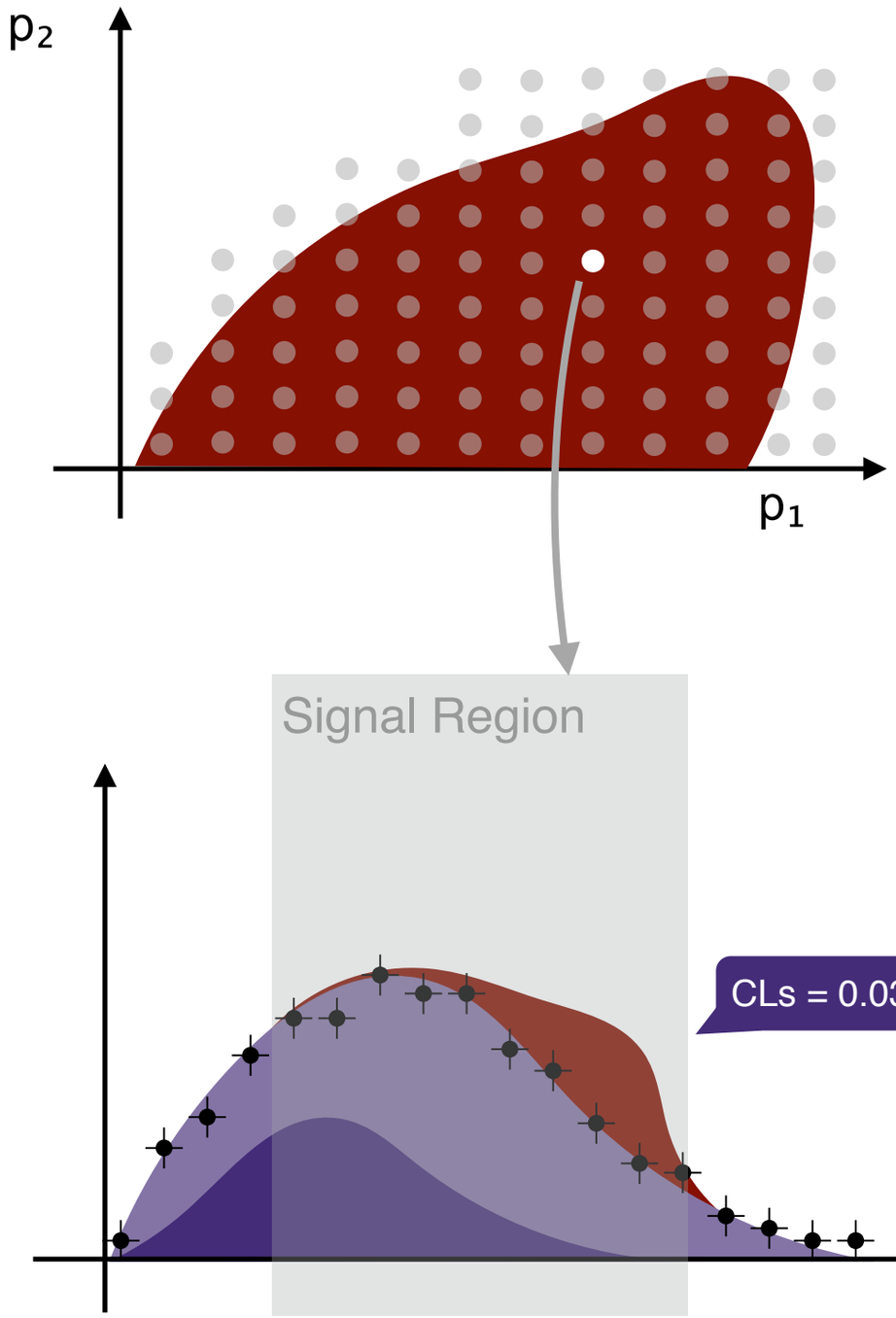
Lukas Heinrich

S2I2 Lightning Talk

NEW YORK UNIVERSITY

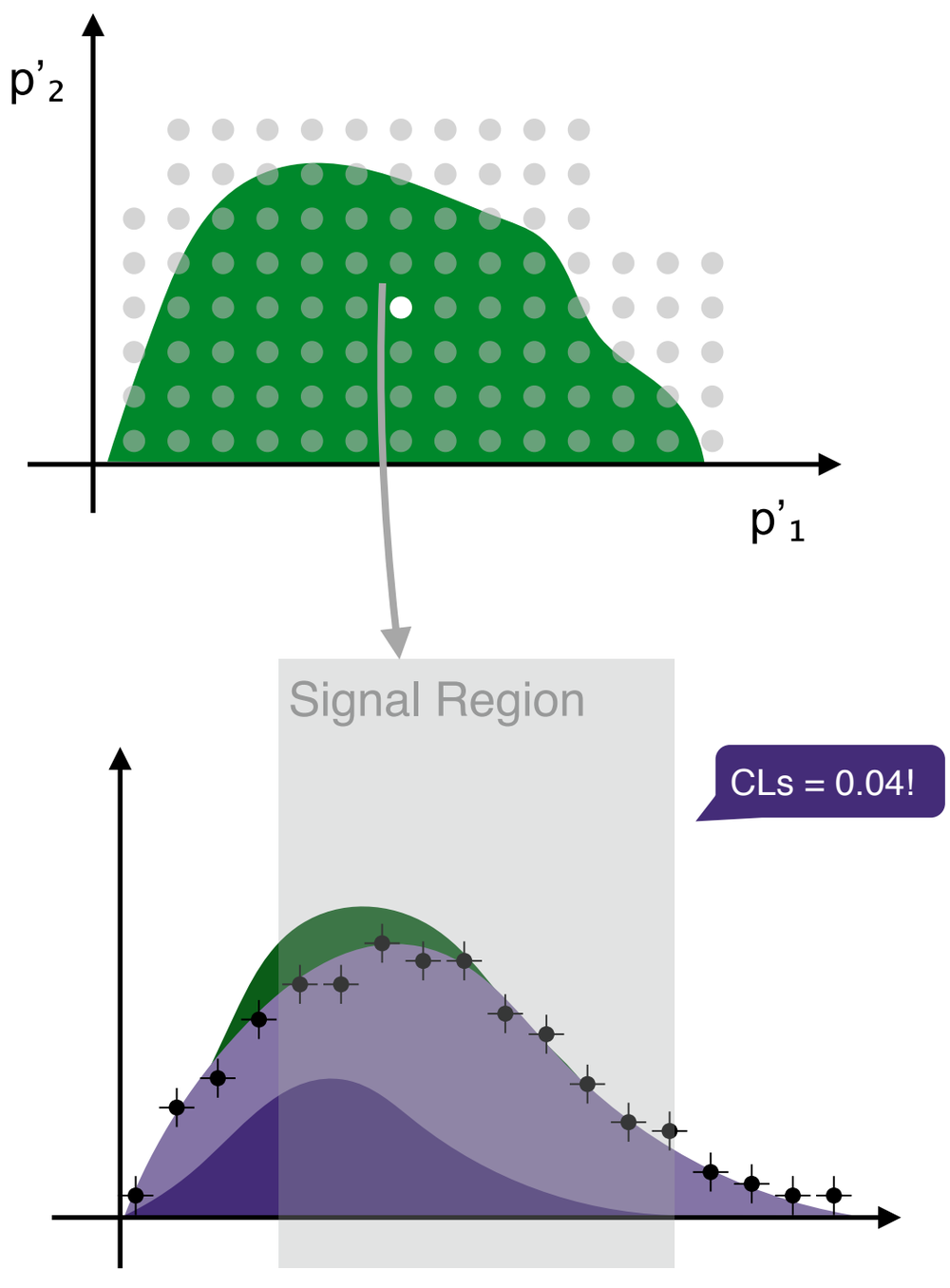
How we implemented RECAST

Interpretations within a specific model often done with few free parameters (e.g. masses, couplings).
Reinterpretation will similarly be performed as *scans* in terms of new parameters.



original analysis (w.r.t **model A**)

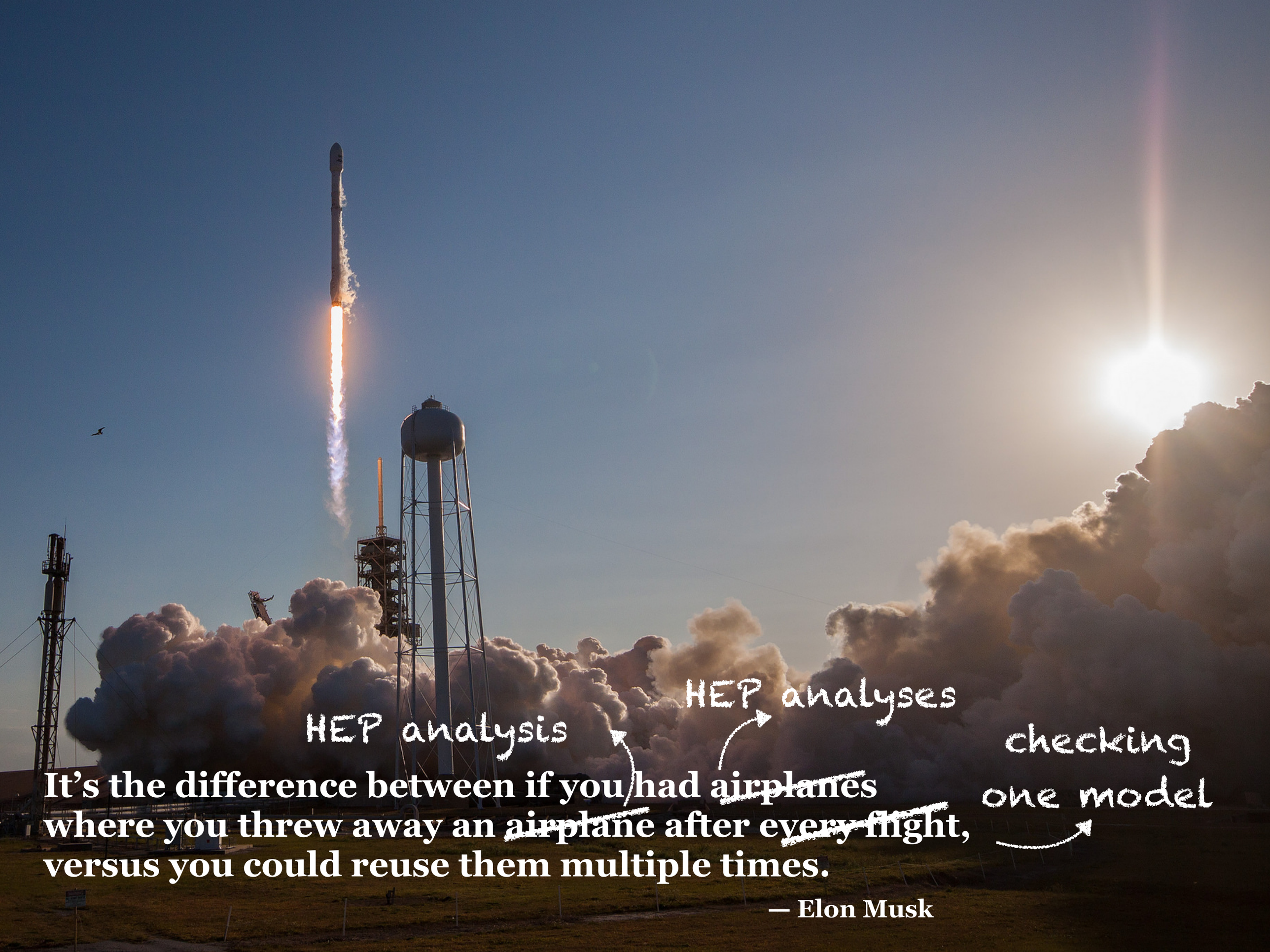
RECAST



original analysis (recast to **model B**)

It’s the difference between if you had airplanes where you threw away an airplane after every flight, versus you could reuse them multiple times.

— Elon Musk



It's the difference between if you had ~~airplanes~~ where you threw away an ~~airplane~~ after ~~every flight~~, versus you could reuse them multiple times.

— Elon Musk

What we needed

1. capturing data processing steps
2. capturing analysis pipelines
3. re-executing pipelines at scale

Goals and Design Principles

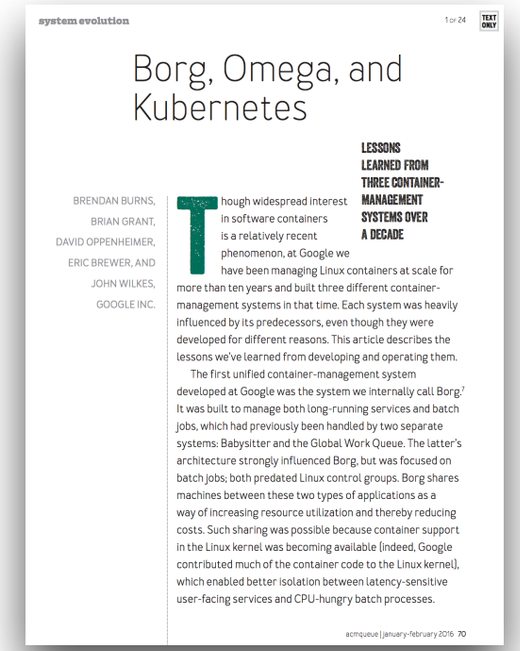
1. only capture the semantics — not the implementation
2. make it declarative
3. make it portable
4. use industry standards, where possible
5. simple data formats

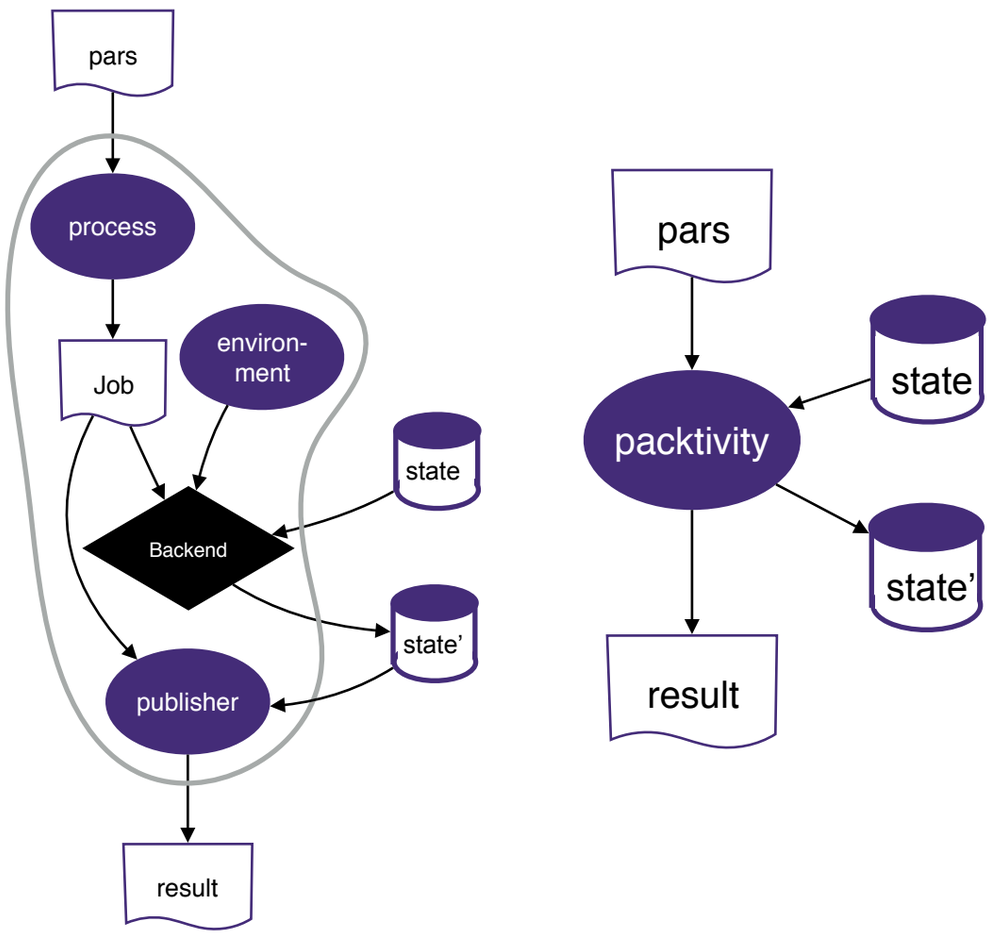
containers are here to stay — it's the right abstraction

processing steps, as JSON functions w/ side-effects

packaged activities — packtivities

in industry this is called *functions as a service (FaaS)*

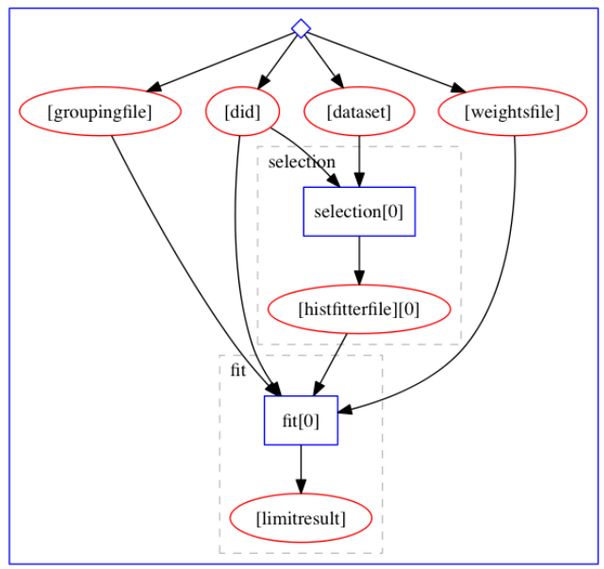




Google's Approach to distributed computing (link)

```
process:
  process_type: 'string-interpolated-cmd'
  cmd: 'rivet -a {analysis} -H {outputyoda} {inputhepmc}'
publisher:
  publisher_type: 'frompar-pub'
  outputmap:
    yodafile: outputyoda
environment:
  environment_type: 'docker-encapsulated'
  image: hepstore/rivet
  imagetag: '2.5.3'
```

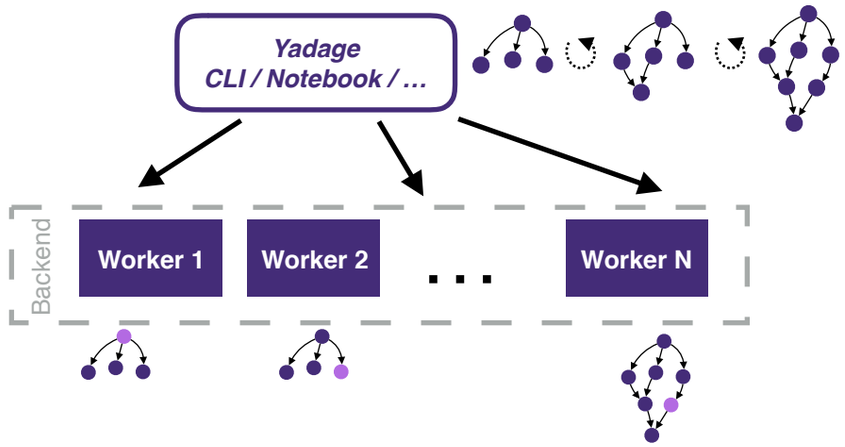
sharable, declarative analysis pipelines



lots of approaches to this problem

Common Workflow Language
WDL: Workflow Description Language
Swift
Makeflow
DAGMan

For ours (***yadage)*** We took inspiration from declarative formats used in industry
(Kubernetes, Heat, Docker, etc, etc.)



- no DSL, JSON/YAML to express workflow
  - indexable, software-definable, human-readable

Use DAG as data-model, but realize topology not known.
Declaratively define recipe how to build graphs.

Recursively composable, without cooperation: No global namespace such as *targets etc..*

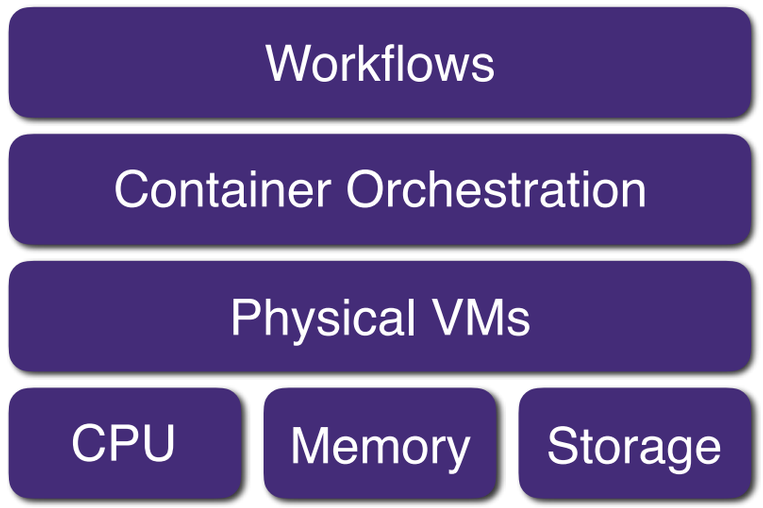
run pipelines on generic infrastructure

Given CPU, Memory, Storage — use minimal complex to run container workflows

OpenStack Magnum — on-demand Kubernetes Clusters on OpenStack

OpenStack Manila — distributed storage based on Ceph

Kubernetes Jobs API — FaaS directly without a batch system



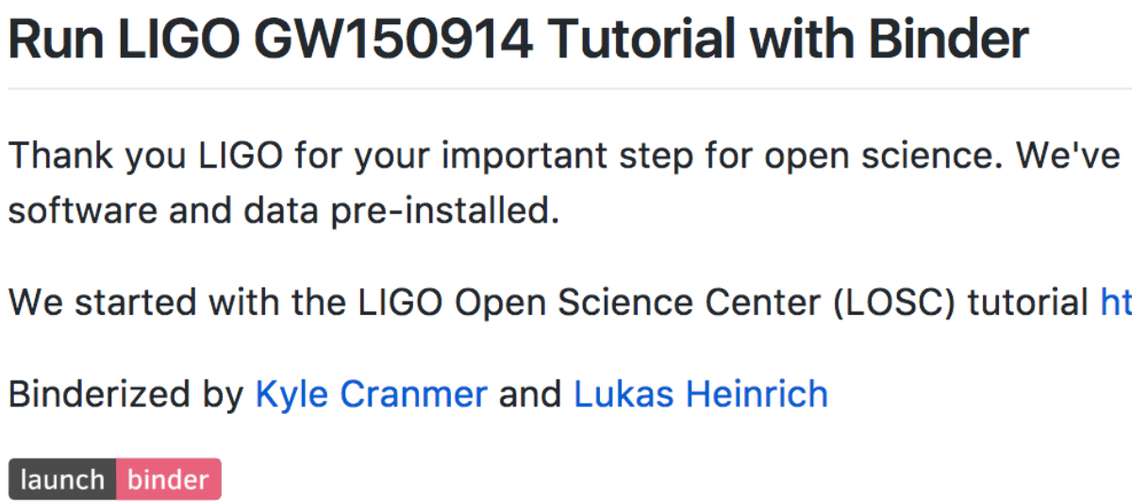
workflows as a service (WaaS?)

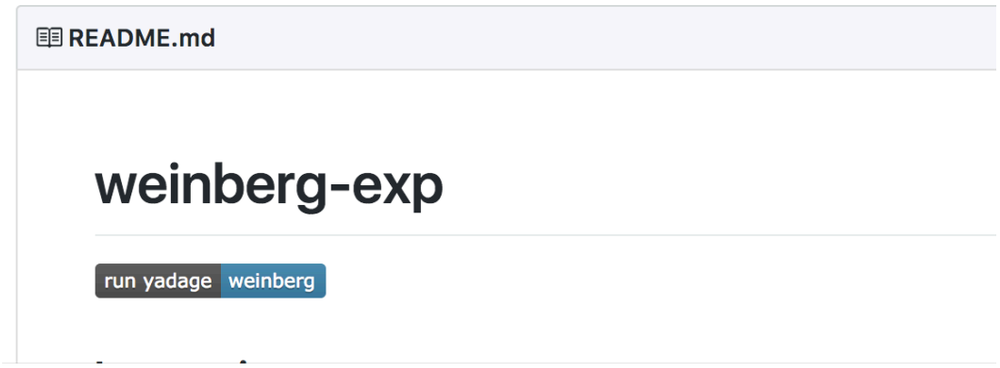
REANA — CERN project to offer workflows as a service

You come with your declaratively defined pipeline based on containers — let CERN run it for you at scale. Goal: make it frictionless

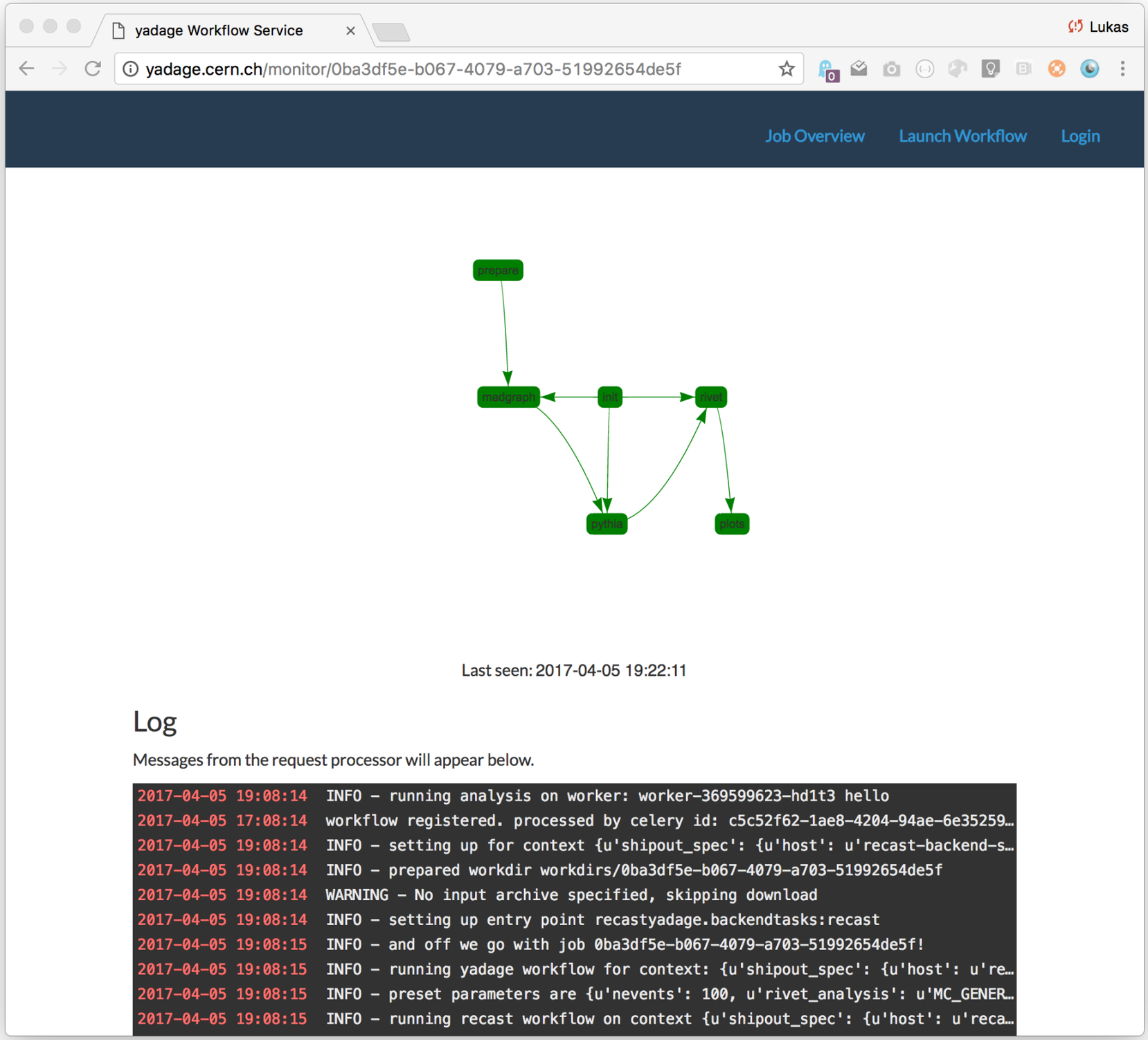
similar philosophy as Binder/Swarm (notebooks as a service)

also similar to Genomics pipelines as a service shown by Karan Bhatia / Google





workflows as a service (WaaS?)



Workflows in Analysis Preservation

workflow

`eventsel.yml`

`fit.yml`

`workflow.yml`

software

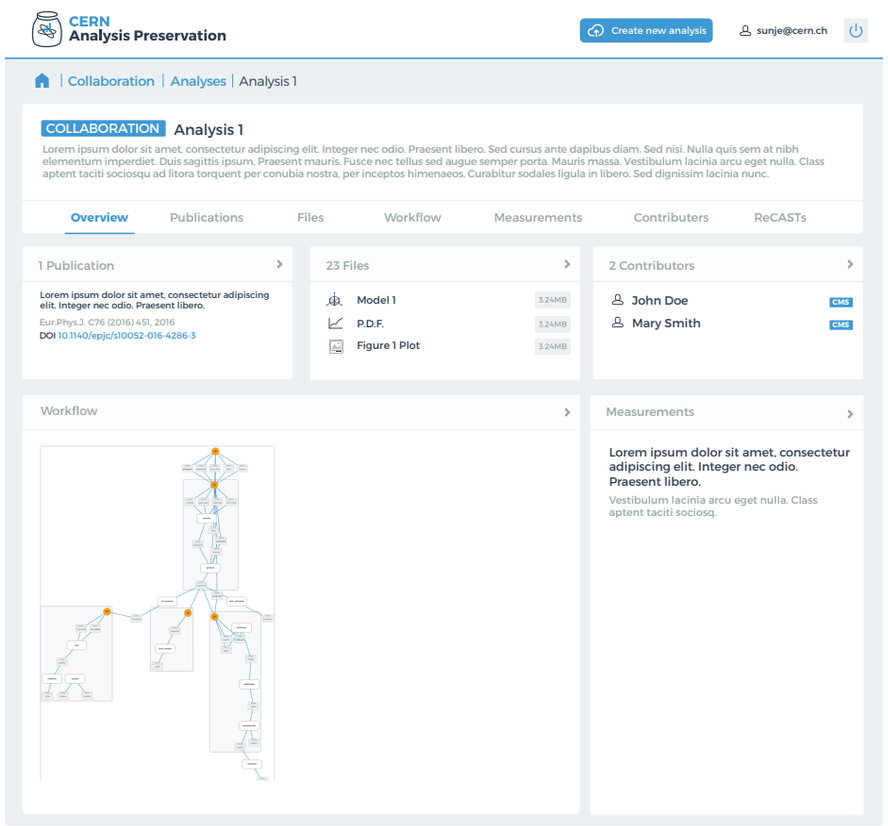
`docker img`

`docker img`

data

`data, bkgds`

import analysis workflow



CERN
Analysis Preservation

NEW YORK UNIVERSITY

RECAST — an application on top of workflows

Now that we have HEP analysis (partially) as declarative workflows in CAP leverage them to do physics.

1. Phenomenologists submit request for reinterpretation to **RECAST Frontend** via user interface or API. Requests comes with necessary information for new model (parameter cards, UFO models)

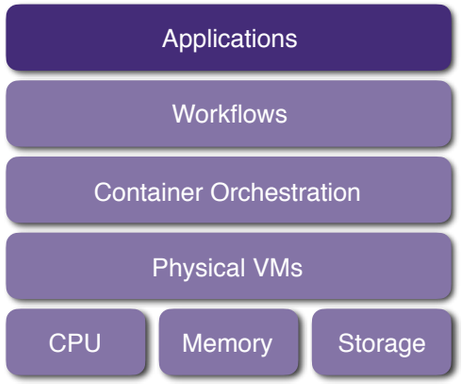
2. Collaboration-internal **RECAST Control Center** allows review of requests.

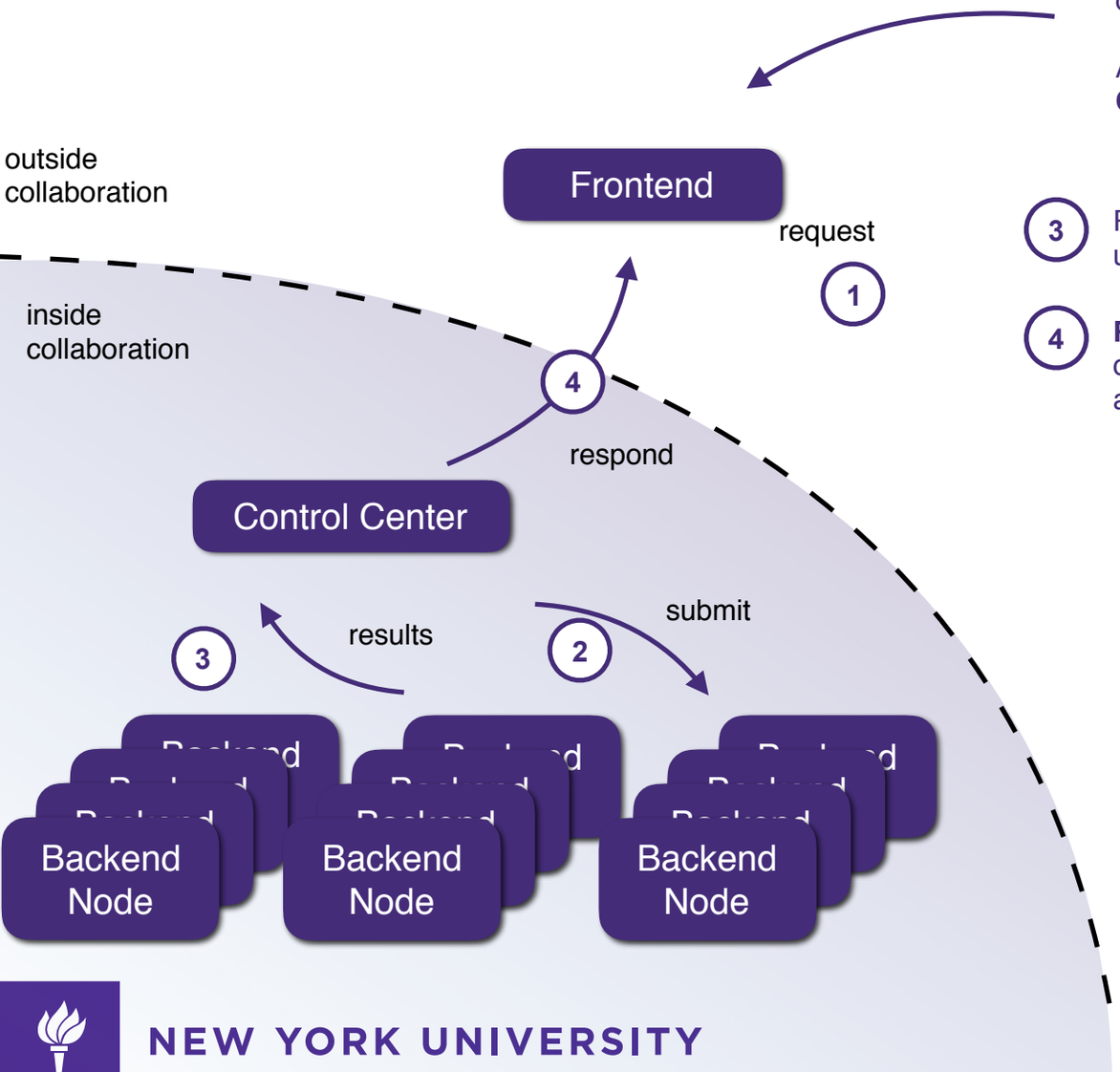
   When approved, alternative models are generated and analyses re-executed on a workflow-aware, container-based computing backend (REANA).

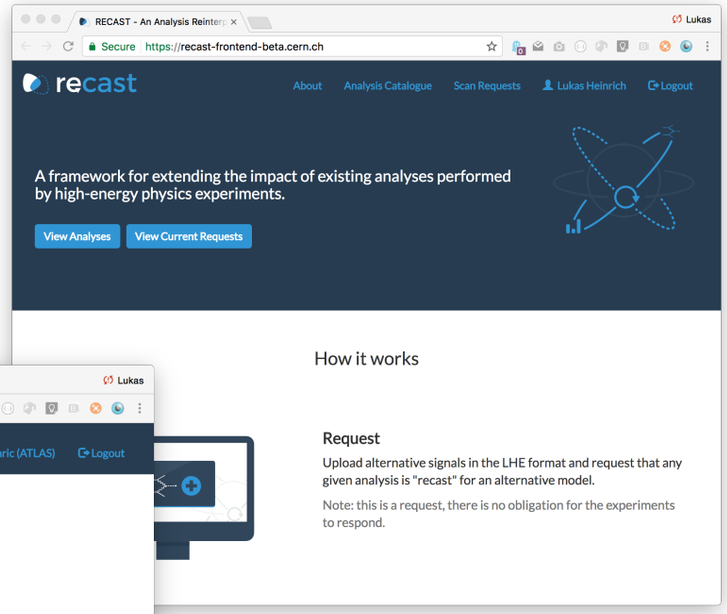
   Analysis workflow description provided by **CERN Analysis Preservation (CAP)**

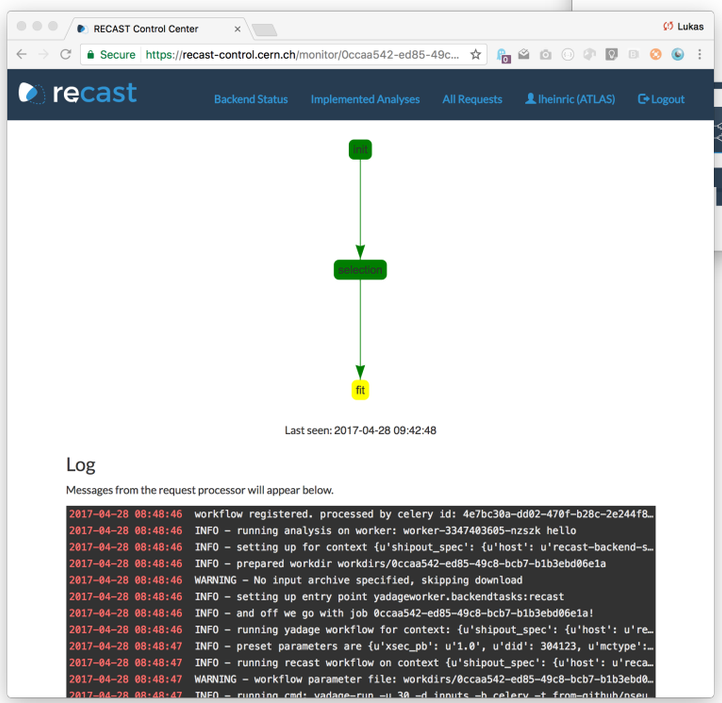
3. Reinterpretation results collected and undergo **approval by the collaboration**.

4. **RECAST response,** including e.g. as limit data, histograms, likelihoods, is published and uploaded to public fronted









NEW YORK UNIVERSITY