Scientific Workflows in the Cloud

bringing modern container infrastructure to HEP

Lukas Heinrich

yadage

declarative, parametrized computational workflows

hep-ex analyses are characterized by distributed, loosely collaborative development model. A typical analysis involves a number of code bases, processing stages that each come with software requirements and developer teams.

In order to preserve (partial) re-executability, one needs a **parametrized description of the analysis**:

$$\underbrace{\text{result} = f_{\text{analysis}}(\text{data}|\text{model})}_{\textit{high level view of an analysis}} \qquad \underbrace{f_{\text{analysis}}(\text{data}|\cdots)}_{\textit{preserve this}}$$

1. preserve individual processing stages in parametrized form via job templates,

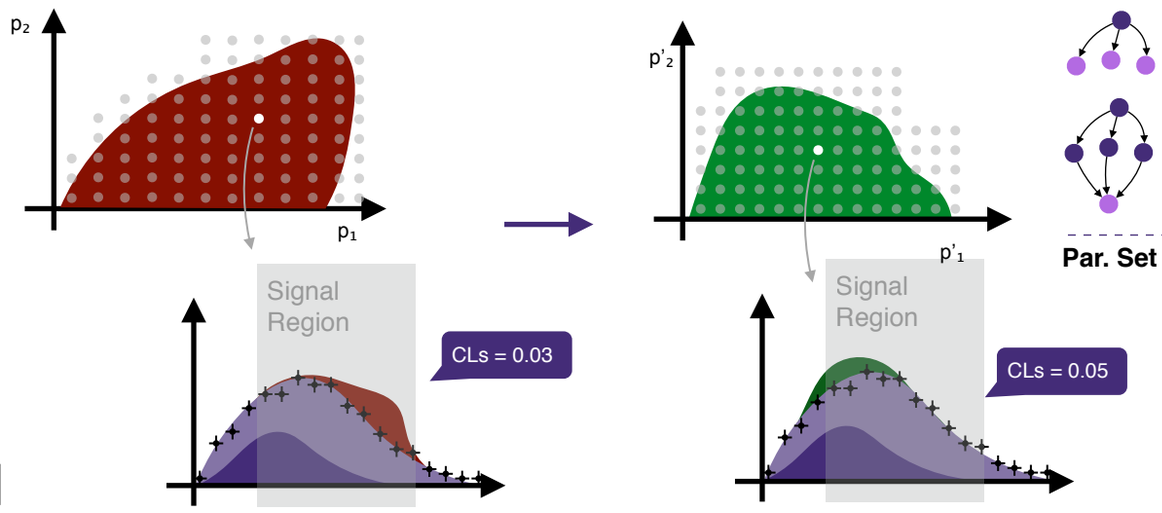
2. capture logic, and dependency structure between processing steps.

RECAST

Analysis Reinterpretation as a Service

Reinterpretations are an efficient way to learn about the viability of alternative physics models based on existing analyses. The majority of resources goes into data-taking, background estimation and analysis design. For a reinterpretations, one only needs to run *a single (signal) dataset* through the analysis pipeline.

With archived analyses codes, reinterpretations can be offered as a service.



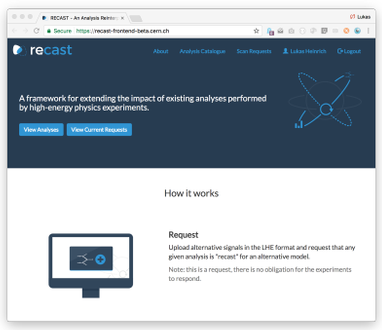
1. Phenomenologists submit request for reinterpretation to **RECAST Frontend** via user interface or API. Requests comes with necessary information for new model (parameter cards, UFO models)

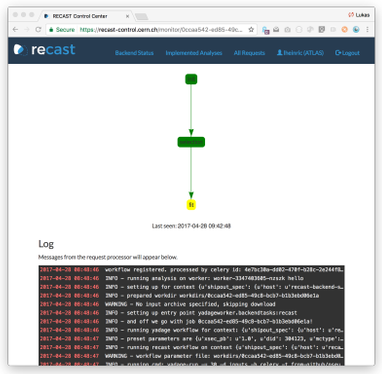
2. Collaboration-internal **RECAST Control Center** allows review of requests.

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

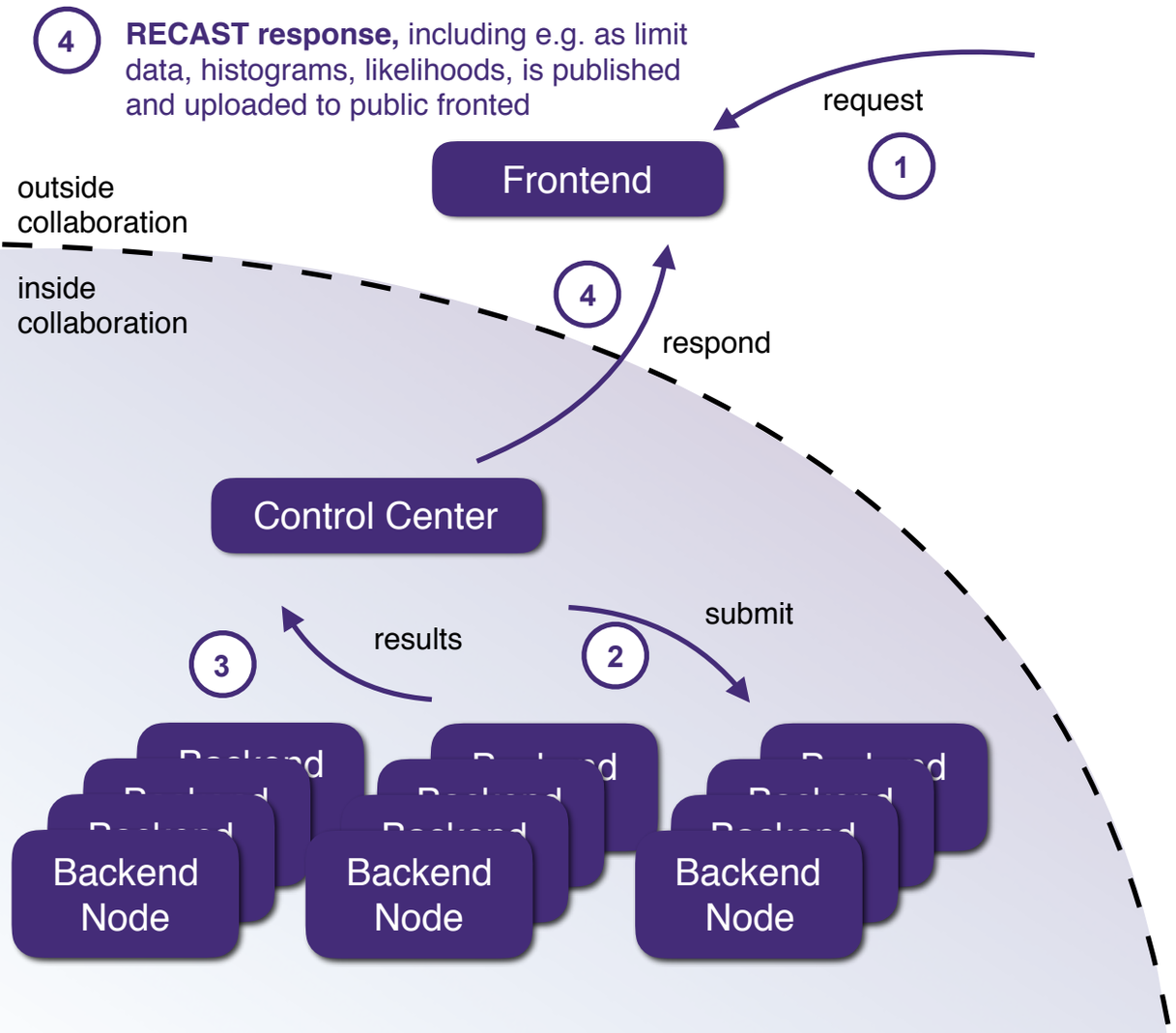
   Analysis workflow description provided by **CERN Analysis Preservation (CAP)**

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



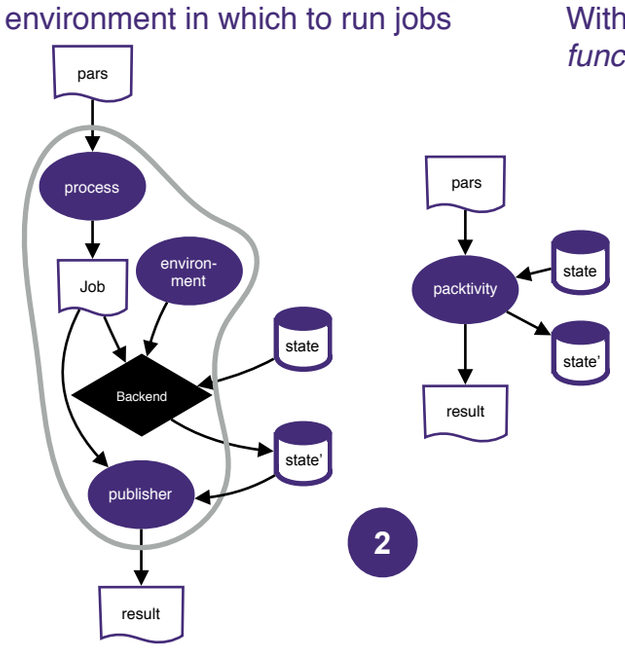


4. **RECAST response,** including e.g. as limit data, histograms, likelihoods, is published and uploaded to public fronted



1. for each processing step, the needed information is:

process: recipe to generate jobs given parameters passed as JSON
environment: spec of computing environment in which to run jobs



publisher: recipe to generate JSON of relevant result data (e.g. file paths to data fragments).

With this, we have a *packaged activity* ***(packtivity),*** *which functions as a JSON API with side-effects.*

2

```yaml
process:
  process_type: 'interpolated-script-cmd'
  interpreter: bash
  script: |
    source ~/.bashrc
    cd WSMaker
    source setup.sh MonoH
    /recast_auth/getkrb.sh
    mkdir -p {fitinputdir}
    hadd {fitinputdir}/LimitHistograms.MonoH.vvbb.13TeV.root {signalfile} {zerolepbkg}
    xrdcp {onelepbkg}  {fitinputdir}/LimitHistograms.MonoH.lvbb.13TeV.root
    xrdcp {twolepbkg}  {fitinputdir}/LimitHistograms.MonoH.llbb.13TeV.root
    ./scripts/MonoH/RunMonoHbbStatsForRecast.sh {fitinputdir} {nametag} model
    grep "\+2sigma" -A6 limits_{nametag}_model.log |grep -v Injected > {limitfile}
publisher:
    publisher_type: interpolated-pub
    publish:
      limitfile: '{limitfile}'
environment:
  environment_type: 'docker-encapsulated'
  image: lukasheinrich/monohstat
  resources:
    - CVMFS
    - GRIDProxy
```

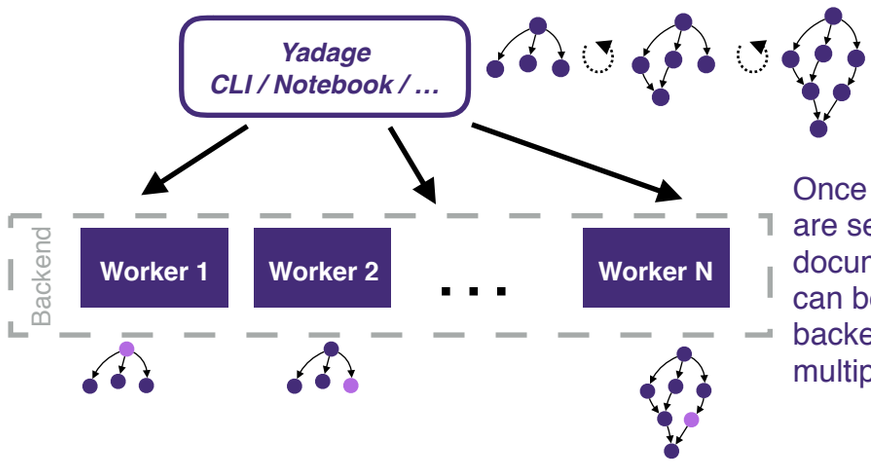
```yaml
process:
  process_type: 'string-interpolated-cmd'
  cmd: 'rivet -a {analysis} -H {outputyoda} {inputhepmc}'
publisher:
  publisher_type: 'frompar-pub'
  outputmap:
    yodafile: outputyoda
environment:
  environment_type: 'docker-encapsulated'
  image: hepstore/rivet
  imagetag: '2.5.3'
```

Analysis workflows are conveniently modeled as directed acyclic graphs: nodes → packtivities, edges → dependencies. But topology is not always known before run-time.

To preserve workflow, preserve logic how to build DAG of packtivities at run-time, in a series of stages.

stages add nodes and edges, self-report what information they require (e.g. upstream nodes finished/defined)

```yaml
stages:
  - name: prepslha
    dependencies: ['init']
    scheduler:
      scheduler_type: singlestep-stage
      step: {$ref: 'steps.yml#/prepare_slha'}
      parameters:
        template: /templates/slha_template.slha
        stop_mass: {stages: init, output: stop_mass, unwrap: true}
        neutralino_mass: {stages: init, output: neutralino_mass, unwrap: true}
        outputSLHA: '{workdir}/spectrum.slha'
  - name: herwig
    dependencies: ['prepslha']
    scheduler:
      scheduler_type: singlestep-stage
      step: {$ref: 'steps.yml#/herwig'}
      parameters:
        slhafile: {stages: prepslha, output: outputSLHA, unwrap: true}
        events: {stages: init, output: events, unwrap: true}
        seed: 1234
        hepmcfile: '{workdir}/out.hepmc'
        herwigruncard: '{workdir}/runcard.dat'
  - name: compute_xsec
    dependencies: ['init']
    scheduler:
      scheduler_type: singlestep-stage
      step: {$ref: 'steps.yml#/compute_xsec'}
      parameters:
        stop_mass: {stages: init, output: stop_mass, unwrap: true}
        xsecfile: '{workdir}/xsec.json'
```

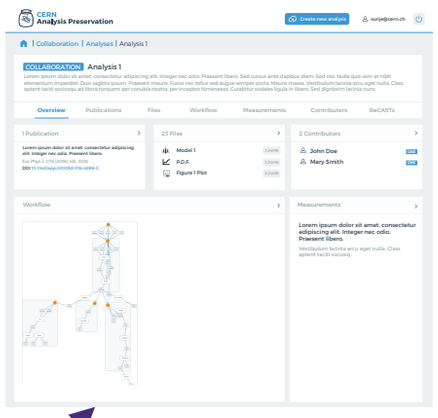


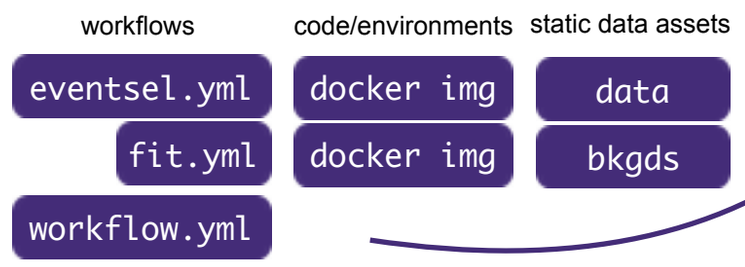
Once defined, yadage workflows are self-contained, portable JSON documents defining workflows that can be executed on a number of backends (ipython clusters, multiprocess pool, …)

CERN Analysis Preservation

Repository for re-usable analyses

CAP is a Invenio-based digital library designed to collect and preserve analysis-specific information. With its tight integration of yadage workflows, it's possible to re-use the preserved analysis for use-cases such as RECAST.





REANA

A generic container-aware workflow service deployable in the cloud.

A joint effort by the RECAST, DASPOS and CAP to provide a scalable, cloud-native distributed workflow service. It utilizes industry standard container workload and orchestration systems (Kubernetes Jobs API) and distributed storage solutions (CephFS) and is designed to support multiple workflow systems (yadage, jupyter notebooks, …). Deployed at CERN on Kubernetes clusters created via various OpenStack technologies (Magnum, Manila, Heat). Scalable to large clusters O(1k nodes) in order to support HEP use-cases such as Monte Carlo generation and analysis re-execution.

ceph EOS CernVM File system kubernetes docker openstack

NEW YORK UNIVERSITY