RECAST use-case and demo

Lukas Heinrich

CAP/RECAST/DASPOS Workshop

NEW YORK UNIVERSITY

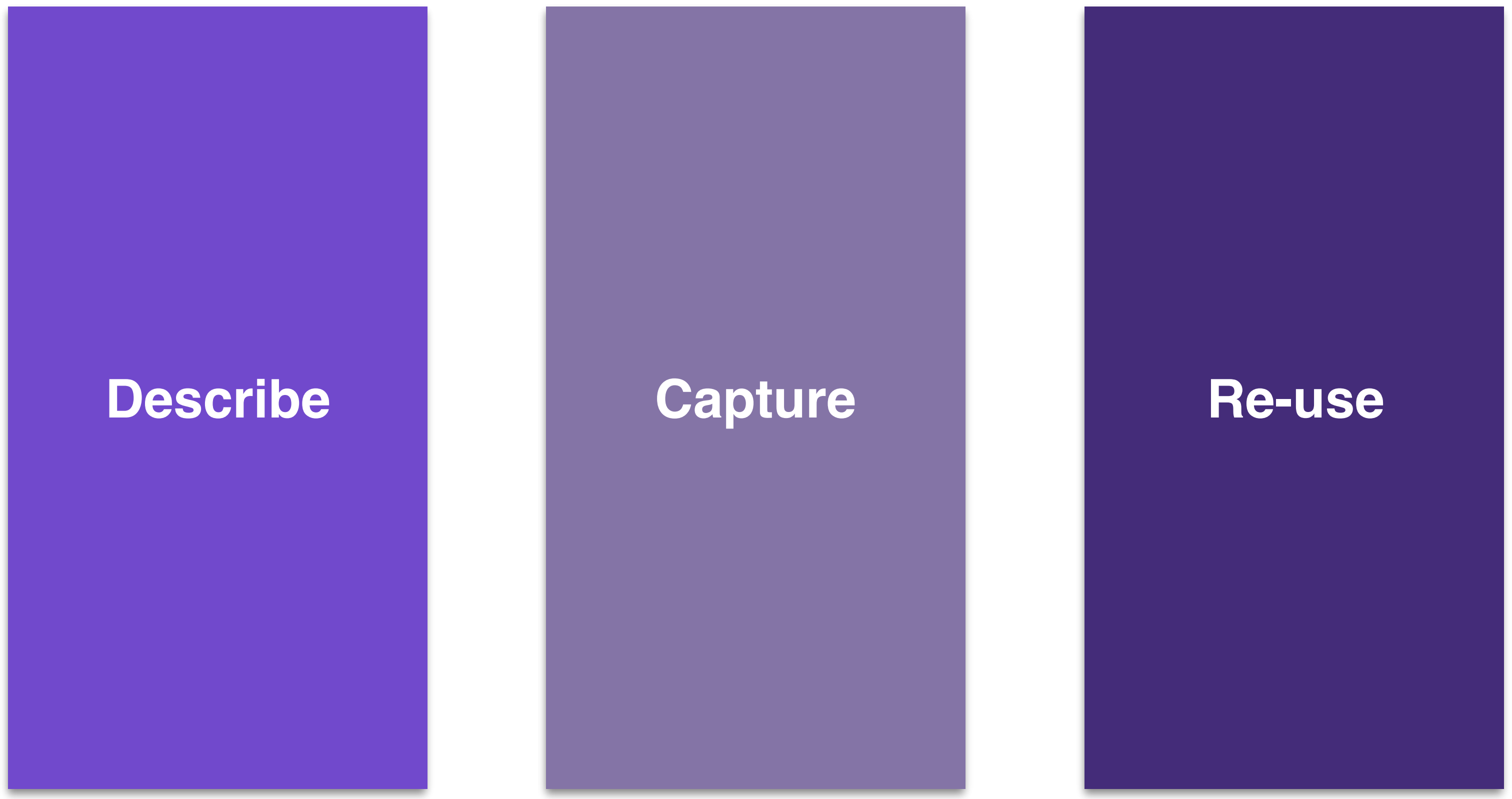
Introduction

Three pillars

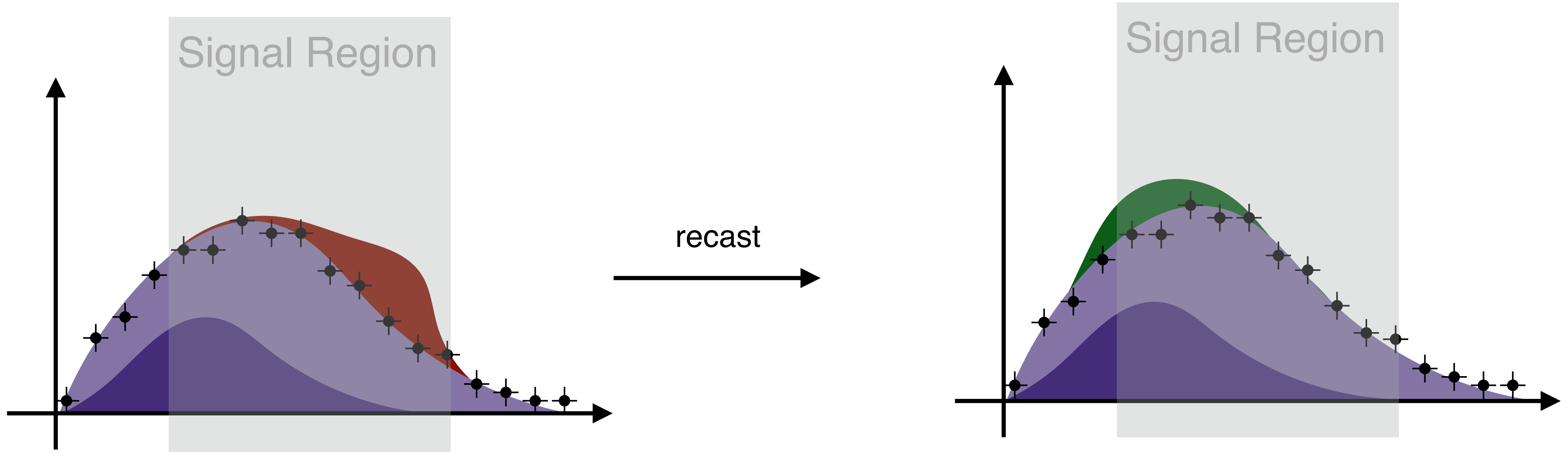
Describe and Capture is about **ingesting** information into CAP about analyses.

The Re-use is pillar about extracting information from CAP and **utilize** that information in new scientific contexts.

If we aim for re-use it informs how describe and capture information.



Reinterpretation/RECAST: running new signal samples through analysis, compare to stored data, background



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

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

Significant demand from outside of collaboration, since published analysis have good efficiency for many models, not just the ones in the paper. Theorists are trying to build own tools, but can only approximate.

To do it properly, we need two ingredients:

1. **1 new fully simulated/reconstructed signal sample**
   - sample needs to go through an *existing* chain of AMI tags e.g. s2608_r7772_r7676_p2666
   - sample size usually not too large, since analysis have good efficiency (i.e. 50-100k events/sample)
2. **Preserved analysis code / workflow**
   - takes over at ntuple level (i.e. derivations) runs event selection, downstream trfs, stat. evaluation etc.

Analysis Ingredients:

↪ measurement of rates and distribution of *pp collision* observables
↪ estimation of expected measurements under model hypotheses
↪ statistical evaluation, interval estimation on model parameters

$$\text{result} = f_{\text{analysis}}(\text{data}|\text{model})$$

observable distributions, confidence intervals on model parameters

reconstruction, event selection, stat. evaluation

collision data from LHC detector

model hypothesis (SM, many SUSY models, etc..)

HepData

CAP
DASPOS

DPHEP
CERN OpenData

UFO/SLHA
(HepData?)

$$\text{result} = f_{\text{analysis}}(\text{data}|\text{model})$$

reconstruction, event selection, stat. evaluation

model hypothesis
(SM, many SUSY models, etc..)

observable distributions,
confidence intervals
on model parameters

collision data from LHC detector

Analysis should be preserve the analysis in a **parametrized form**, not only its concrete application, separately from its application on given dataset.

$$f_{\text{analysis}}(\,\cdot\,), \quad \text{data}, \quad \text{model}$$

Given a **parametrized preservation of an analysis** (even w/ fixed data), we gain ability to extract **new results** using existing resources.

Reinterpretation of Single Analysis under multiple models $\longrightarrow$

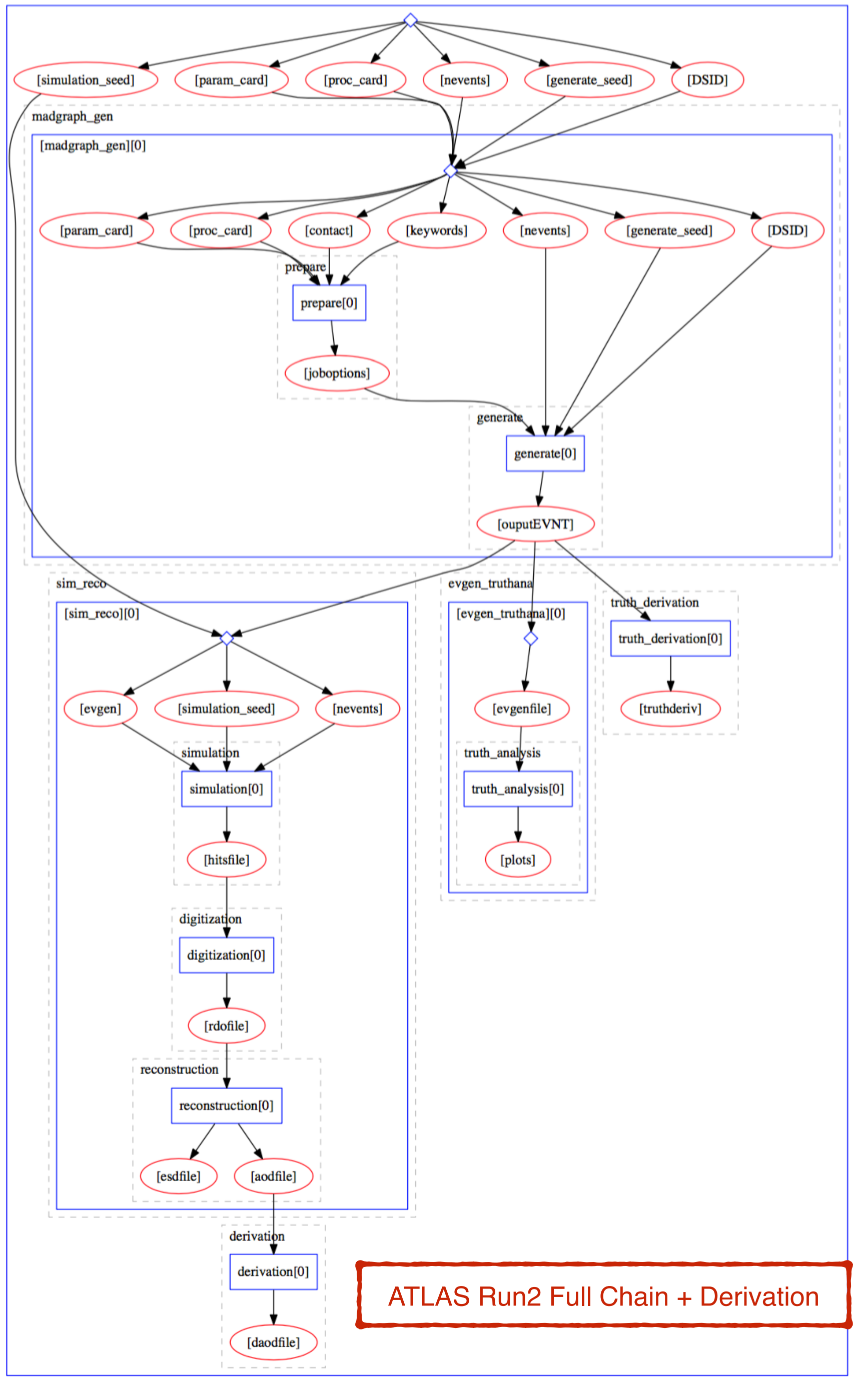
Combination of multiple analyses w.r.t. one model (increased stat. power) $\downarrow$

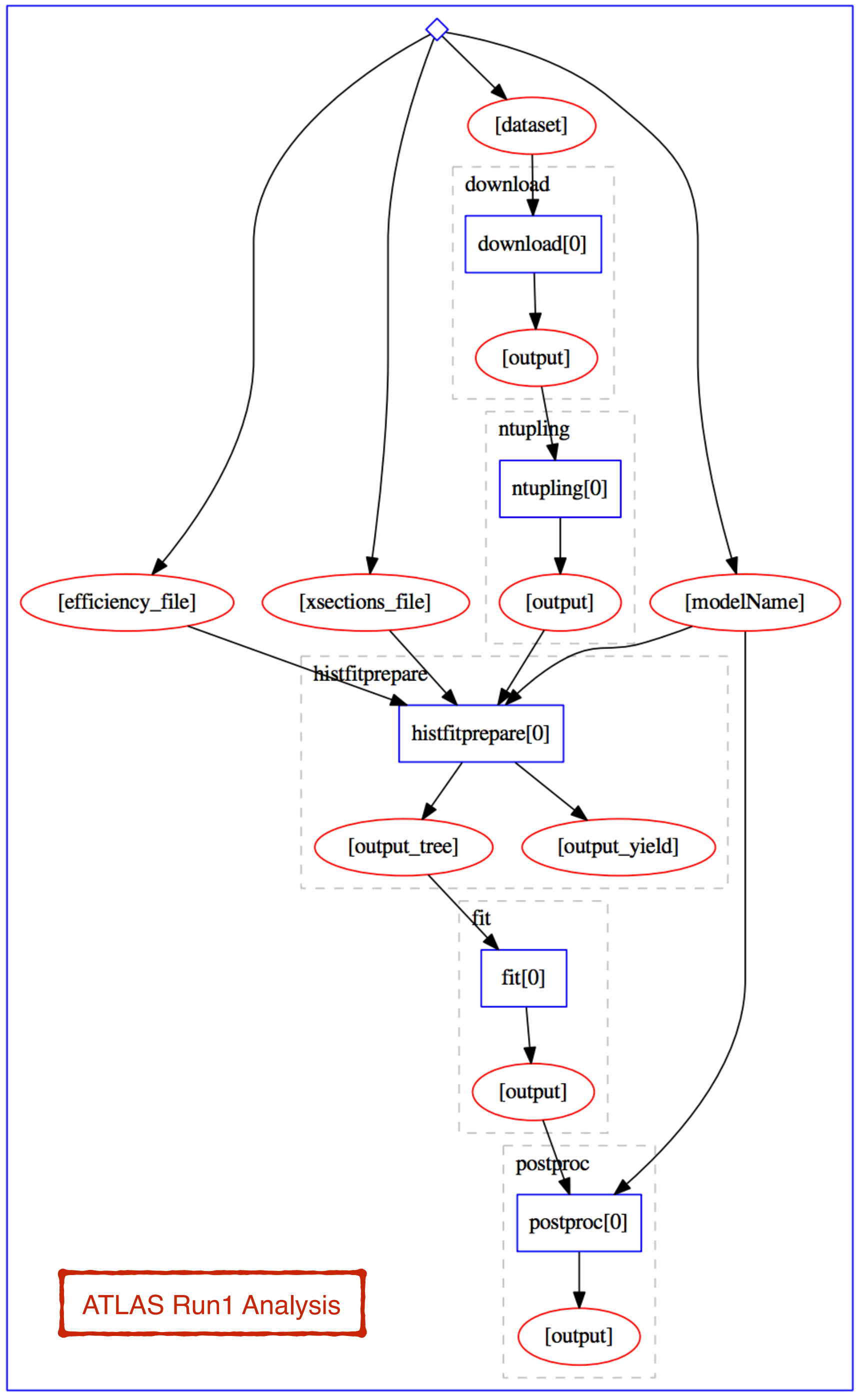
$$\begin{array}{cc}
f_{\text{a}}(\text{data}|\text{model}_1) & f_{\text{a}}(\text{data}|\text{model}_2) \\
f_{\text{b}}(\text{data}|\text{model}_1) & f_{\text{b}}(\text{data}|\text{model}_2) \\
f_{\text{c}}(\text{data}|\text{model}_1) & f_{\text{c}}(\text{data}|\text{model}_2)
\end{array}$$

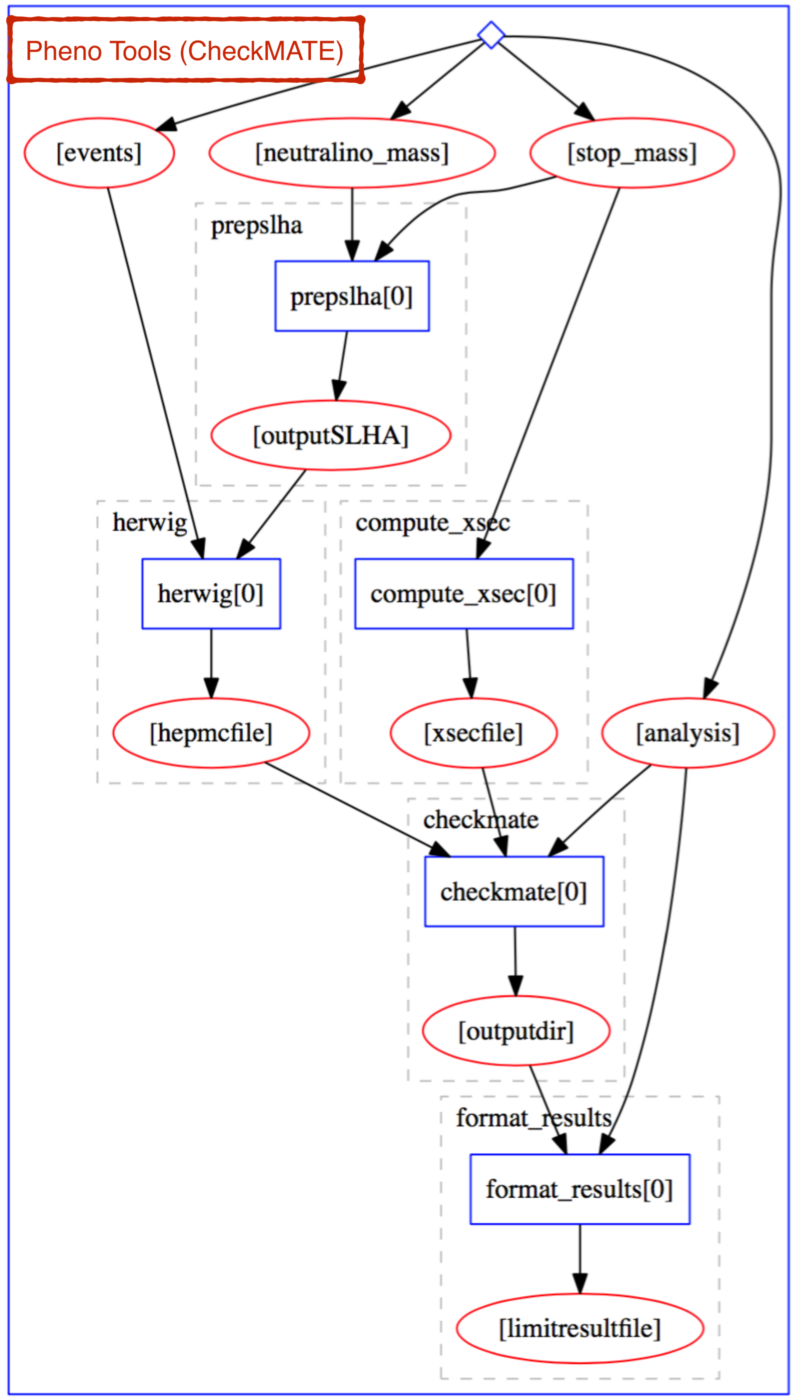
Introduction

Backend uses cluster of ~100VCPU on cluster of worker nodes to execute analysis workflows.

Backend runs on **yadage workflow engine** that build **directed acyclic graphs (DAG)** of tasks with software preserved and shipped via **Docker Containers**. Workflows are defined via JSON/YAML description language. Can (and already are for some examples) stored in **CERN Analysis Preservation Portal**





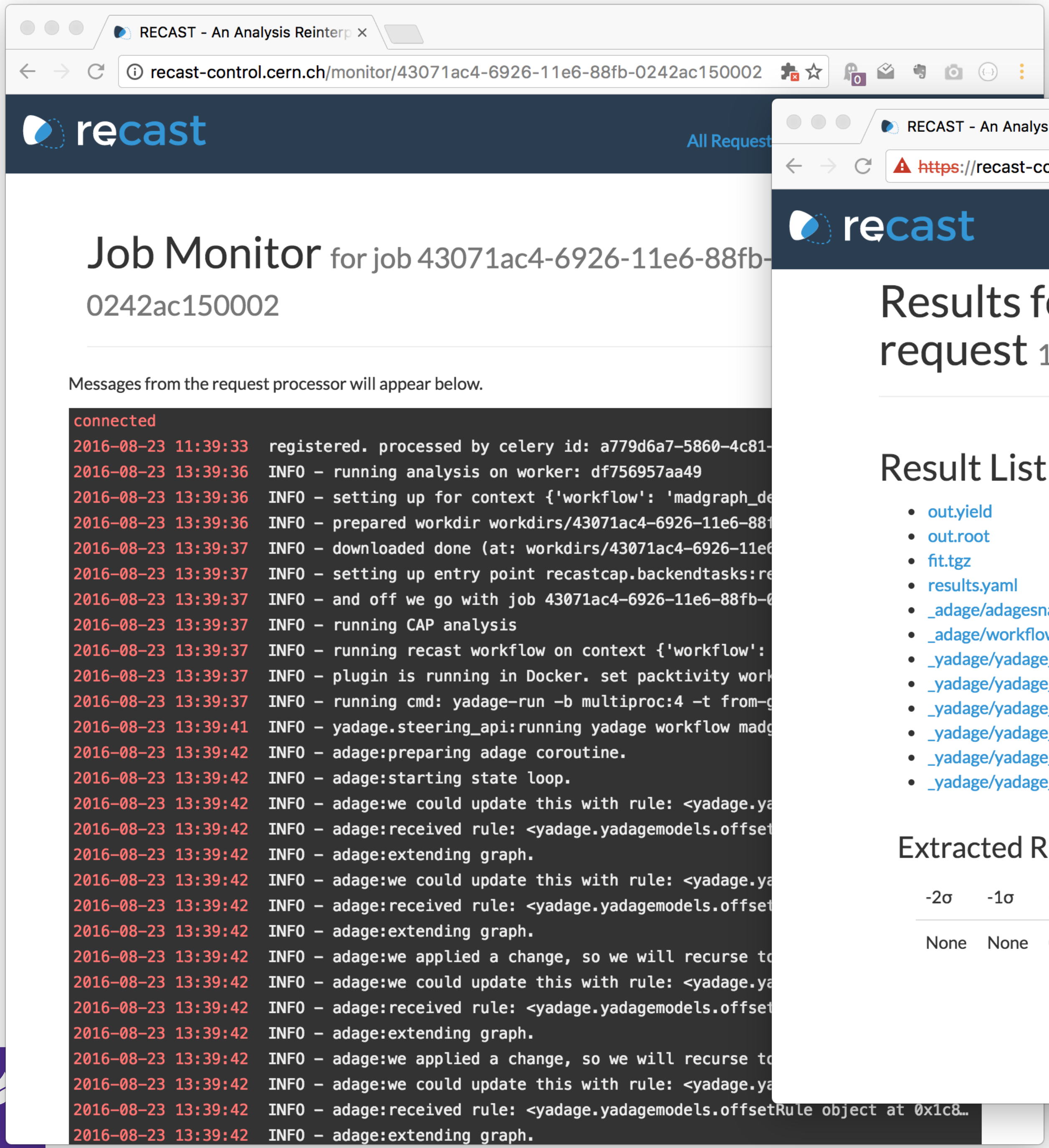


Introduction

RECAST is an old idea to formalize / streamline the process. Let interested parties formulate new models / parameter point grids for existing analyses, have centralized backend that allows to re-run analysis pipeline without additional workload on original analysis team.

- capture analysis code / during development, when team knows best how to run it
- capture workflow logic of separate parts (selection / fit)

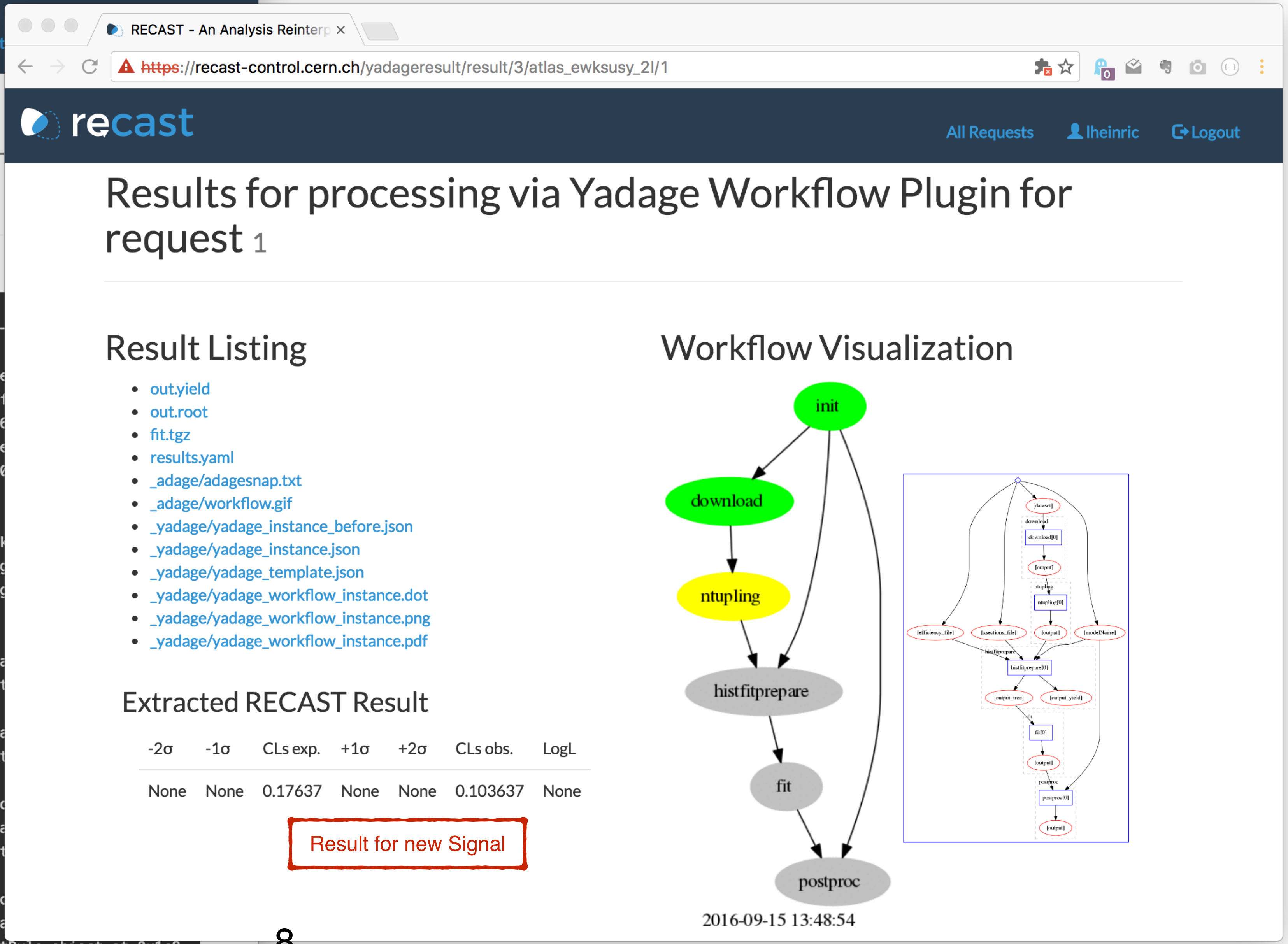
Result: allows us to run newly generated signals automatically using semi-automated framework.


RECAST - An Analysis Reinterp
recast-control.cern.ch/monitor/43071ac4-6926-11e6-88fb-0242ac150002
recast
All Request

Job Monitor for job 43071ac4-6926-11e6-88fb-0242ac150002

Messages from the request processor will appear below.

```
connected
2016-08-23 11:39:33  registered. processed by celery id: a779d6a7-5860-4c81-
2016-08-23 13:39:36  INFO - running analysis on worker: df756957aa49
2016-08-23 13:39:36  INFO - setting up for context {'workflow': 'madgraph_de
2016-08-23 13:39:36  INFO - prepared workdir workdirs/43071ac4-6926-11e6-88f
2016-08-23 13:39:37  INFO - downloaded done (at: workdirs/43071ac4-6926-11e6
2016-08-23 13:39:37  INFO - setting up entry point recastcap.backendtasks:re
2016-08-23 13:39:37  INFO - and off we go with job 43071ac4-6926-11e6-88fb-0
2016-08-23 13:39:37  INFO - running CAP analysis
2016-08-23 13:39:37  INFO - running recast workflow on context {'workflow':
2016-08-23 13:39:37  INFO - plugin is running in Docker. set packtivity work
2016-08-23 13:39:37  INFO - running cmd: yadage-run -b multiproc:4 -t from-g
2016-08-23 13:39:41  INFO - yadage.steering_api:running yadage workflow madg
2016-08-23 13:39:42  INFO - adage:preparing adage coroutine.
2016-08-23 13:39:42  INFO - adage:starting state loop.
2016-08-23 13:39:42  INFO - adage:we could update this with rule: <yadage.ya
2016-08-23 13:39:42  INFO - adage:received rule: <yadage.yadagemodels.offset
2016-08-23 13:39:42  INFO - adage:extending graph.
2016-08-23 13:39:42  INFO - adage:we could update this with rule: <yadage.ya
2016-08-23 13:39:42  INFO - adage:received rule: <yadage.yadagemodels.offset
2016-08-23 13:39:42  INFO - adage:extending graph.
2016-08-23 13:39:42  INFO - adage:we applied a change, so we will recurse to
2016-08-23 13:39:42  INFO - adage:we could update this with rule: <yadage.ya
2016-08-23 13:39:42  INFO - adage:received rule: <yadage.yadagemodels.offset
2016-08-23 13:39:42  INFO - adage:extending graph.
2016-08-23 13:39:42  INFO - adage:we applied a change, so we will recurse to
2016-08-23 13:39:42  INFO - adage:we could update this with rule: <yadage.ya
2016-08-23 13:39:42  INFO - adage:received rule: <yadage.yadagemodels.offsetRule object at 0x1c8…
2016-08-23 13:39:42  INFO - adage:extending graph.
```


RECAST - An Analysis Reinterp
https://recast-control.cern.ch/yadageresult/result/3/atlas_ewksusy_2l/1
recast
All Requests
lheinric
Logout

Results for processing via Yadage Workflow Plugin for request 1

Result Listing

- out.yield
- out.root
- fit.tgz
- results.yaml
- _adage/adagesnap.txt
- _adage/workflow.gif
- _yadage/yadage_instance_before.json
- _yadage/yadage_instance.json
- _yadage/yadage_template.json
- _yadage/yadage_workflow_instance.dot
- _yadage/yadage_workflow_instance.png
- _yadage/yadage_workflow_instance.pdf

Extracted RECAST Result

-2σ	-1σ	CLs exp.	+1σ	+2σ	CLs obs.	LogL
None	None	0.17637	None	None	0.103637	None

Workflow Visualization

Introduction

Analysis Preservation: two-step process

Modern HEP analysis:

- Multiple steps/code-bases, possibly developed by independent teams, with differing software requirements. Example: one team developing the event selection, another team developing the statistical analysis

Need to capture:

1. Individual processing steps

- code bases
- software environments
- identify binaries, scripts in code base
- templates how to run binaries (semantic description of arguments, naming etc..)
- description of step output, what are the relevant data fragments

2. How to connect these steps

- How to wire individual steps together
- What outputs of which steps, are used as inputs for other steps, …

Introduction

Analysis Preservation: two-step process

For semi-automation, the information needs to detailed, but is also most closely related to day to day work of the physicists.

Developed flexible JSON Schemas (diana-hep/yadage-schemas) and Implementation to describe both parts:

1. Individual processing steps

diana-hep/packtivity

packtivity

build passing | pypi v0.5.7

2. How to connect these steps

diana-hep/yadage

yadage - yaml based adage

pypi package 0.10.5 | build passing | health 92% | docs latest | 403.2MB 8 layers

Introduction

diana-hep/packtivity

packtivity

build passing pypi v0.5.7

parametrized process:

template job from which we can produce concrete job

template: `“./DelphesHepMC <input file> <output file>”`

concrete: `“./DelphesHepMC /input/file/path.hepmc /output/file.root”`

environment:

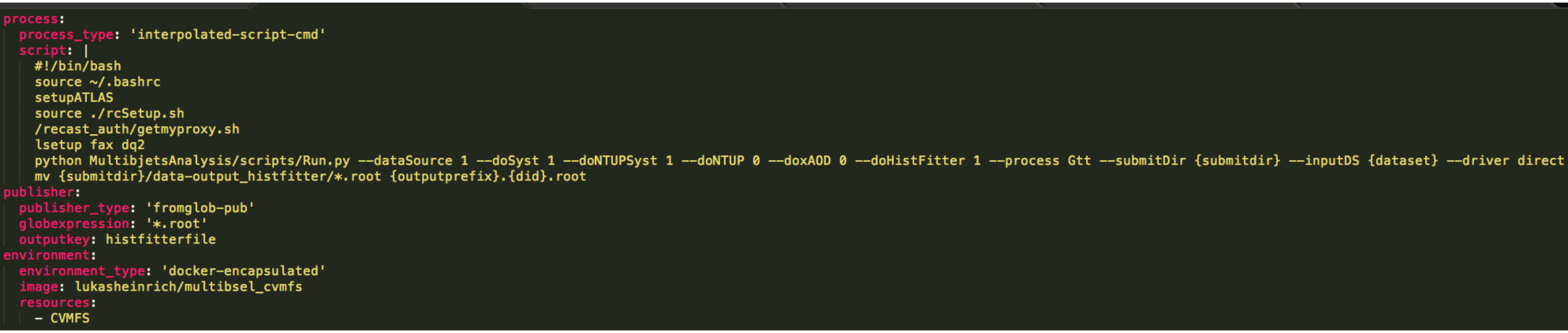
description of computing env in which above job can run.

Multiple options, promising: *Linux Containers* (investigating Umbrella, etc)

publisher:

recipe how to extract parsable result data after job completion

e.g. globbing files in a work directory, just forwarding input parameters that refer to output data etc…



```
process:
  process_type: 'interpolated-script-cmd'
  script: |
    #!/bin/bash
    source ~/.bashrc
    setupATLAS
    source ./rcSetup.sh
    /recast_auth/getmyproxy.sh
    lsetup fax dq2
    python MultibjetsAnalysis/scripts/Run.py --dataSource 1 --doSyst 1 --doNTUPSyst 1 --doNTUP 0 --doxAOD 0 --doHistFitter 1 --process Gtt --submitDir {submitdir} --inputDS {dataset} --driver direct
    mv {submitdir}/data-output_histfitter/*.root {outputprefix}.{did}.root
publisher:
  publisher_type: 'fromglob-pub'
  globexpression: '*.root'
  outputkey: histfitterfile
environment:
  environment_type: 'docker-encapsulated'
  image: lukasheinrich/multibsel_cvmfs
  resources:
    - CVMFS
```

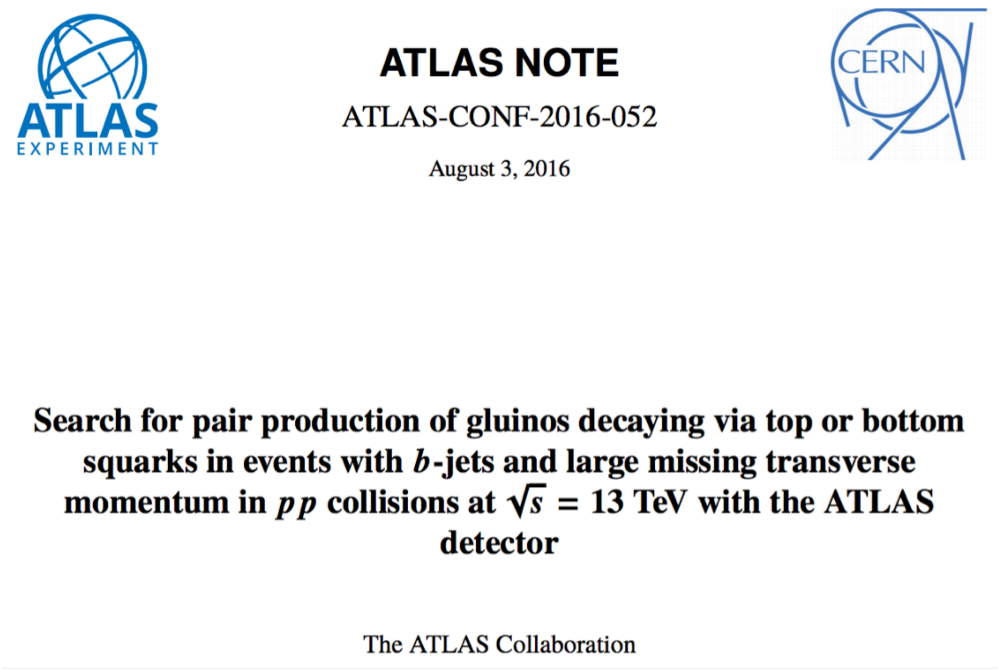
diana-hep/packtivity

packtivity

build passing pypi v0.5.7

Case Study: ATLAS Run-2 Analysis Event Selection Code (Multi-B-jet analysis, publication ATLAS-CONF-2016-052

- Code developed in GitLab repository
- using own C++ framework on top of ATLAS analysis releases taken from CVMFS
- Tested using GitLab CI continuous integration within Docker environment
- build Docker image (including custom code, excluding cvmfs), same build scripts as are used in CI builds..
- Runs Event Selection on ATLAS centrally produced DxAOD derivation files, writes out HistFitter ROOT files



ATLAS EXPERIMENT

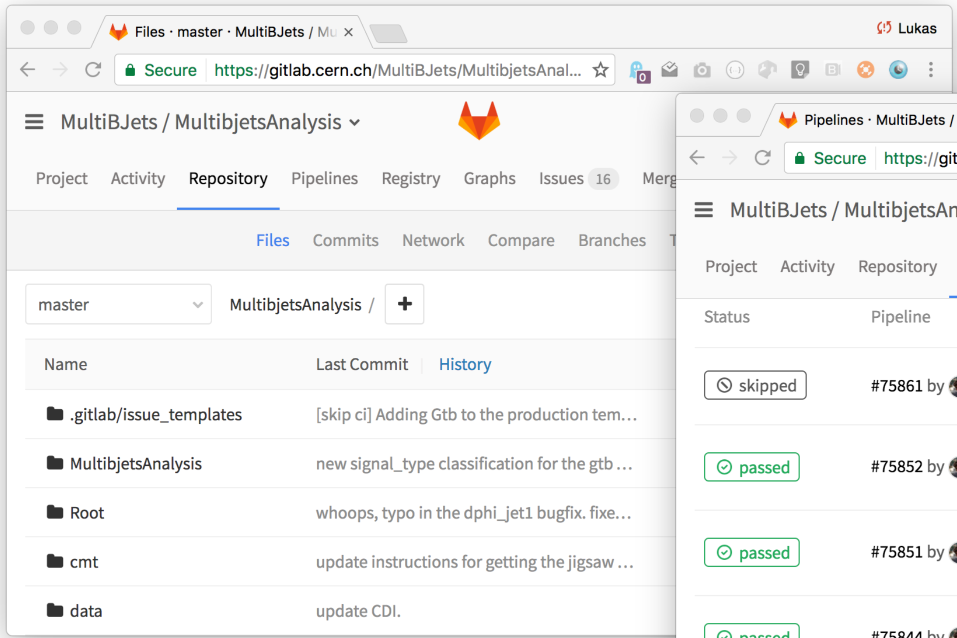
ATLAS NOTE

ATLAS-CONF-2016-052

August 3, 2016

Search for pair production of gluinos decaying via top or bottom squarks in events with b-jets and large missing transverse momentum in pp collisions at $\sqrt{s}$ = 13 TeV with the ATLAS detector

The ATLAS Collaboration



Files · master · MultiBJets / Mu

Secure https://gitlab.cern.ch/MultiBJets/MultibjetsAnal...

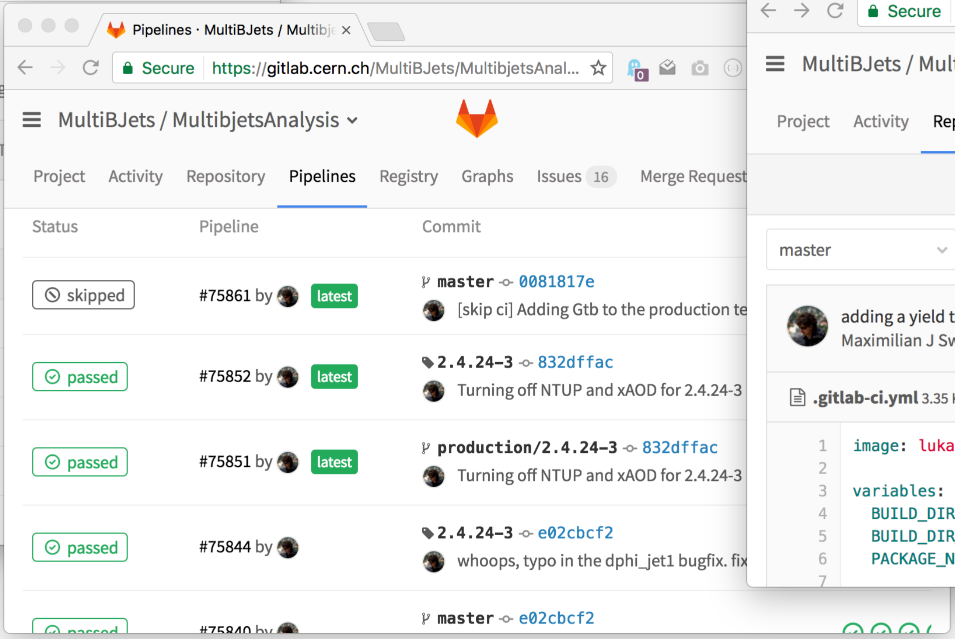
MultiBJets / MultibjetsAnalysis

Project Activity Repository Pipelines Registry Graphs Issues 16 Merg

Files Commits Network Compare Branches T

master MultibjetsAnalysis /

Name	Last Commit | History
.gitlab/issue_templates	[skip ci] Adding Gtb to the production tem…
MultibjetsAnalysis	new signal_type classification for the gtb …
Root	whoops, typo in the dphi_jet1 bugfix. fixe…
cmt	update instructions for getting the jigsaw …
data	update CDI.



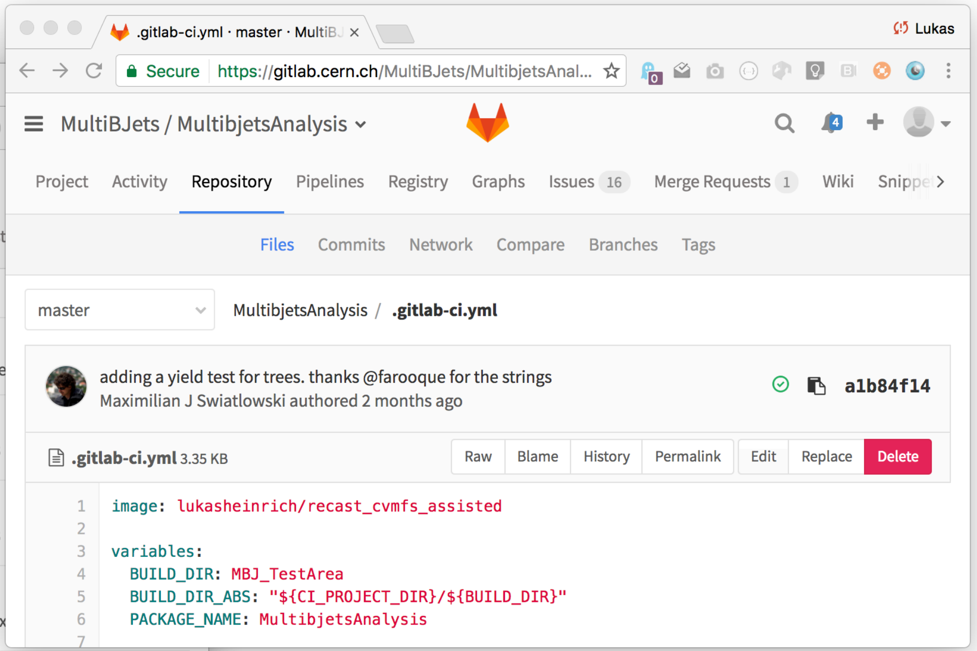
Pipelines · MultiBJets / Multibj

Secure https://gitlab.cern.ch/MultiBJets/MultibjetsAnal...

MultiBJets / MultibjetsAnalysis

Project Activity Repository Pipelines Registry Graphs Issues 16 Merge Request

Status	Pipeline	Commit
skipped	#75861 by latest	master 0081817e [skip ci] Adding Gtb to the production te
passed	#75852 by latest	2.4.24-3 832dffac Turning off NTUP and xAOD for 2.4.24-3
passed	#75851 by latest	production/2.4.24-3 832dffac Turning off NTUP and xAOD for 2.4.24-3
passed	#75844 by	2.4.24-3 e02cbcf2 whoops, typo in the dphi_jet1 bugfix. fix
passed	#75840 by	master e02cbcf2



.gitlab-ci.yml · master · MultiBJ

Secure https://gitlab.cern.ch/MultiBJets/MultibjetsAnal...

MultiBJets / MultibjetsAnalysis

Project Activity Repository Pipelines Registry Graphs Issues 16 Merge Requests 1 Wiki Snippe

Files Commits Network Compare Branches Tags

master MultibjetsAnalysis / .gitlab-ci.yml

adding a yield test for trees. thanks @farooque for the strings
Maximilian J Swiatlowski authored 2 months ago a1b84f14

.gitlab-ci.yml 3.35 KB Raw Blame History Permalink Edit Replace Delete

```
image: lukasheinrich/recast_cvmfs_assisted

variables:
  BUILD_DIR: MBJ_TestArea
  BUILD_DIR_ABS: "${CI_PROJECT_DIR}/${BUILD_DIR}"
  PACKAGE_NAME: MultibjetsAnalysis
```

diana-hep/packtivity

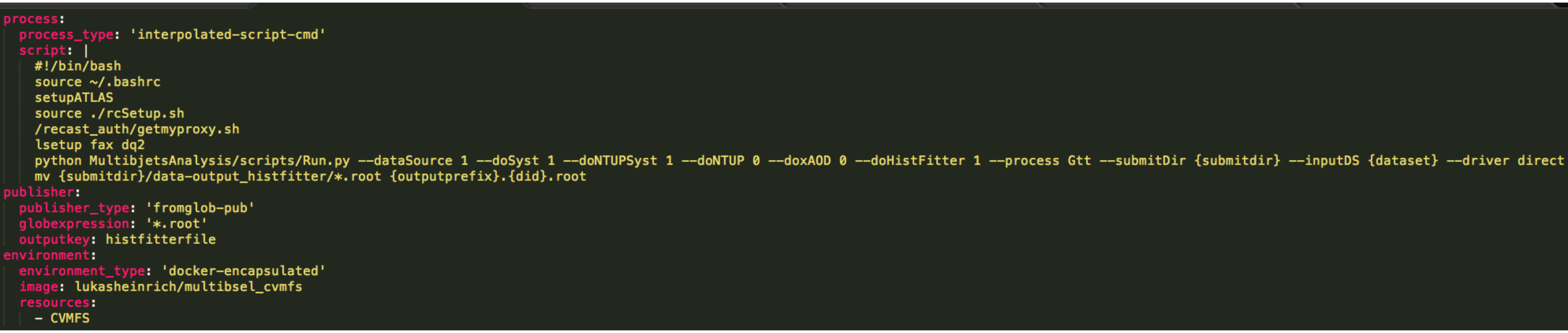
packtivity

build passing pypi v0.5.7

A lot of the work is already done **during the development of the analysis,** adapts as analysis team adapts (framwork switches, code version etc…)

- Code Capture in Docker images
- Build scripts
- etc..

To capture semantically, only need small amount of information how to use the code on a given input file (in this case a grid dataset streamed via XrootD). Can be prepared/tested during development, fixed at publication time.

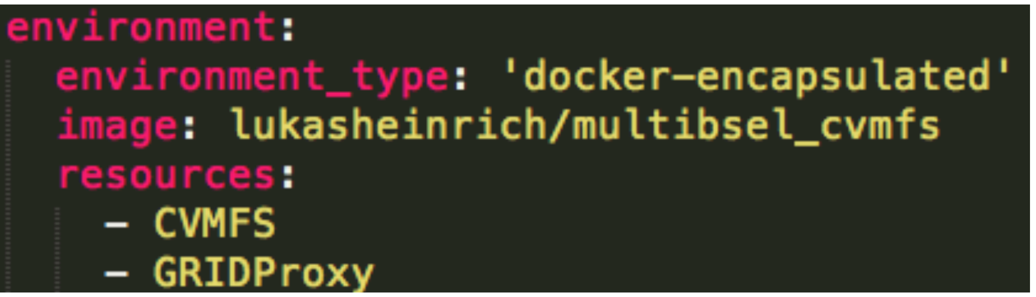


```
process:
  process_type: 'interpolated-script-cmd'
  script: |
    #!/bin/bash
    source ~/.bashrc
    setupATLAS
    source ./rcSetup.sh
    /recast_auth/getmyproxy.sh
    lsetup fax dq2
    python MultibjetsAnalysis/scripts/Run.py --dataSource 1 --doSyst 1 --doNTUPSyst 1 --doNTUP 0 --doxAOD 0 --doHistFitter 1 --process Gtt --submitDir {submitdir} --inputDS {dataset} --driver direct
    mv {submitdir}/data-output_histfitter/*.root {outputprefix}.{did}.root
publisher:
  publisher_type: 'fromglob-pub'
  globexpression: '*.root'
  outputkey: histfitterfile
environment:
  environment_type: 'docker-encapsulated'
  image: lukasheinrich/multibsel_cvmfs
  resources:
    - CVMFS
```

diana-hep/packtivity

packtivity

build passing pypi v0.5.7

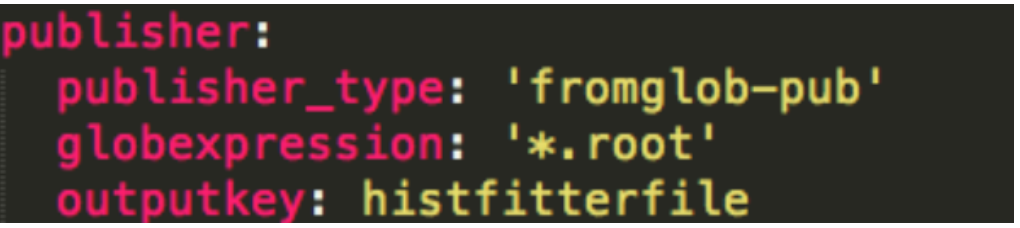


```
environment:
  environment_type: 'docker-encapsulated'
  image: lukasheinrich/multibsel_cvmfs
  resources:
    - CVMFS
    - GRIDProxy
```

←

uniquely specify code base
list external requirements: CVMFS, X509
authentication to get grid access

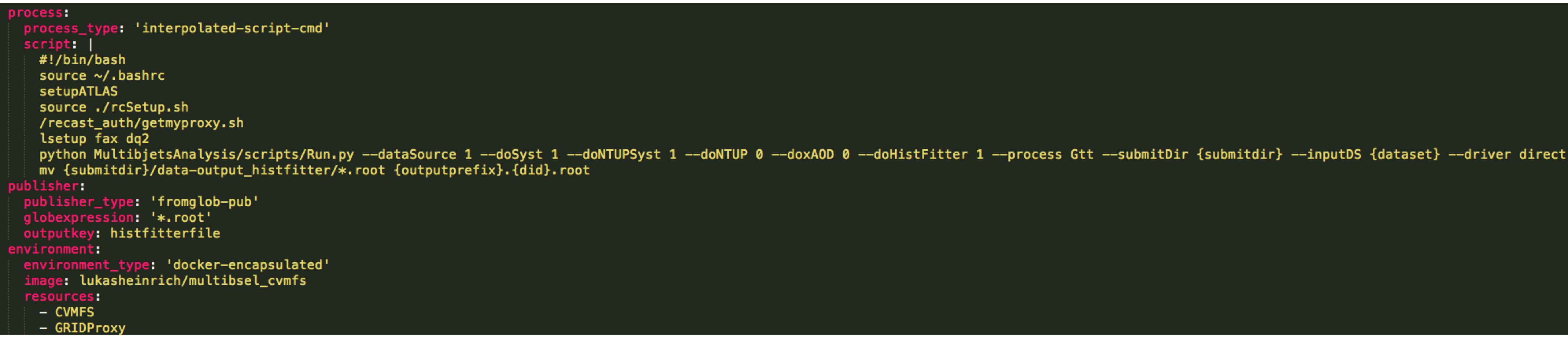
←



```
publisher:
  publisher_type: 'fromglob-pub'
  globexpression: '*.root'
  outputkey: histfitterfile
```

semantic output of this step is: all root files
in work directory after event selection has run

```
{
   "histfitterfile": ["dynamicfilename.root"]
}
```



```
process:
  process_type: 'interpolated-script-cmd'
  script: |
    #!/bin/bash
    source ~/.bashrc
    setupATLAS
    source ./rcSetup.sh
    /recast_auth/getmyproxy.sh
    lsetup fax dq2
    python MultibjetsAnalysis/scripts/Run.py --dataSource 1 --doSyst 1 --doNTUPSyst 1 --doNTUP 0 --doxAOD 0 --doHistFitter 1 --process Gtt --submitDir {submitdir} --inputDS {dataset} --driver direct
    mv {submitdir}/data-output_histfitter/*.root {outputprefix}.{did}.root
publisher:
  publisher_type: 'fromglob-pub'
  globexpression: '*.root'
  outputkey: histfitterfile
environment:
  environment_type: 'docker-encapsulated'
  image: lukasheinrich/multibsel_cvmfs
  resources:
    - CVMFS
    - GRIDProxy
```

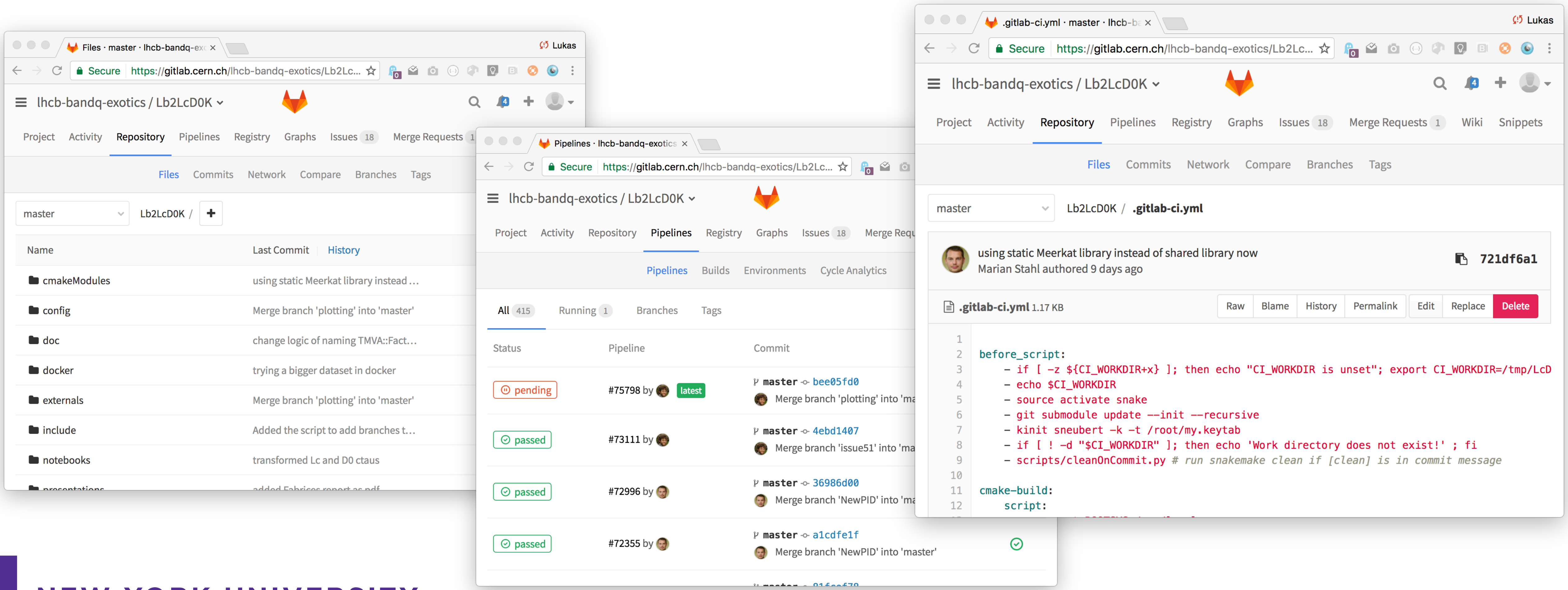
diana-hep/packtivity

packtivity

build passing pypi v0.5.7

Case Study: LHCb Lb2LcD0K analysis

- Code developed in GitLab repository
- I literally don't know anything about the code base :) ask Sebastian
- Tested using GitLab CI continuous integration, does not need CVMFS
- lots of custom software requirements, from user packages to PyPI packages
- Runs complex data pipeline using external workflow tool: snakemake



Files · master · lhcb-bandq-exotics
Secure https://gitlab.cern.ch/lhcb-bandq-exotics/Lb2Lc...
Lukas
lhcb-bandq-exotics / Lb2LcD0K
Project Activity Repository Pipelines Registry Graphs Issues 18 Merge Requests 1
Files Commits Network Compare Branches Tags
master Lb2LcD0K /

Name	Last Commit History
cmakeModules	using static Meerkat library instead …
config	Merge branch 'plotting' into 'master'
doc	change logic of naming TMVA::Fact…
docker	trying a bigger dataset in docker
externals	Merge branch 'plotting' into 'master'
include	Added the script to add branches t…
notebooks	transformed Lc and D0 ctaus
presentations	added Fabrices report as pdf

Pipelines · lhcb-bandq-exotics
Secure https://gitlab.cern.ch/lhcb-bandq-exotics/Lb2Lc...
lhcb-bandq-exotics / Lb2LcD0K
Project Activity Repository Pipelines Registry Graphs Issues 18 Merge Requ
Pipelines Builds Environments Cycle Analytics
All 415 Running 1 Branches Tags

Status	Pipeline	Commit
pending	#75798 by latest	master bee05fd0 Merge branch 'plotting' into 'ma
passed	#73111 by	master 4ebd1407 Merge branch 'issue51' into 'ma
passed	#72996 by	master 36986d00 Merge branch 'NewPID' into 'ma
passed	#72355 by	master a1cdfe1f Merge branch 'NewPID' into 'master'

.gitlab-ci.yml · master · lhcb-b
Secure https://gitlab.cern.ch/lhcb-bandq-exotics/Lb2Lc...
Lukas
lhcb-bandq-exotics / Lb2LcD0K
Project Activity Repository Pipelines Registry Graphs Issues 18 Merge Requests 1 Wiki Snippets
Files Commits Network Compare Branches Tags
master Lb2LcD0K / .gitlab-ci.yml
Raw Blame History Permalink Edit Replace Delete

using static Meerkat library instead of shared library now
Marian Stahl authored 9 days ago 721df6a1

.gitlab-ci.yml 1.17 KB

```

before_script:
    - if [ -z ${CI_WORKDIR+x} ]; then echo "CI_WORKDIR is unset"; export CI_WORKDIR=/tmp/LcD
    - echo $CI_WORKDIR
    - source activate snake
    - git submodule update --init --recursive
    - kinit sneubert -k -t /root/my.keytab
    - if [ ! -d "$CI_WORKDIR" ]; then echo 'Work directory does not exist!' ; fi
    - scripts/cleanOnCommit.py # run snakemake clean if [clean] is in commit message

cmake-build:
    script:
```

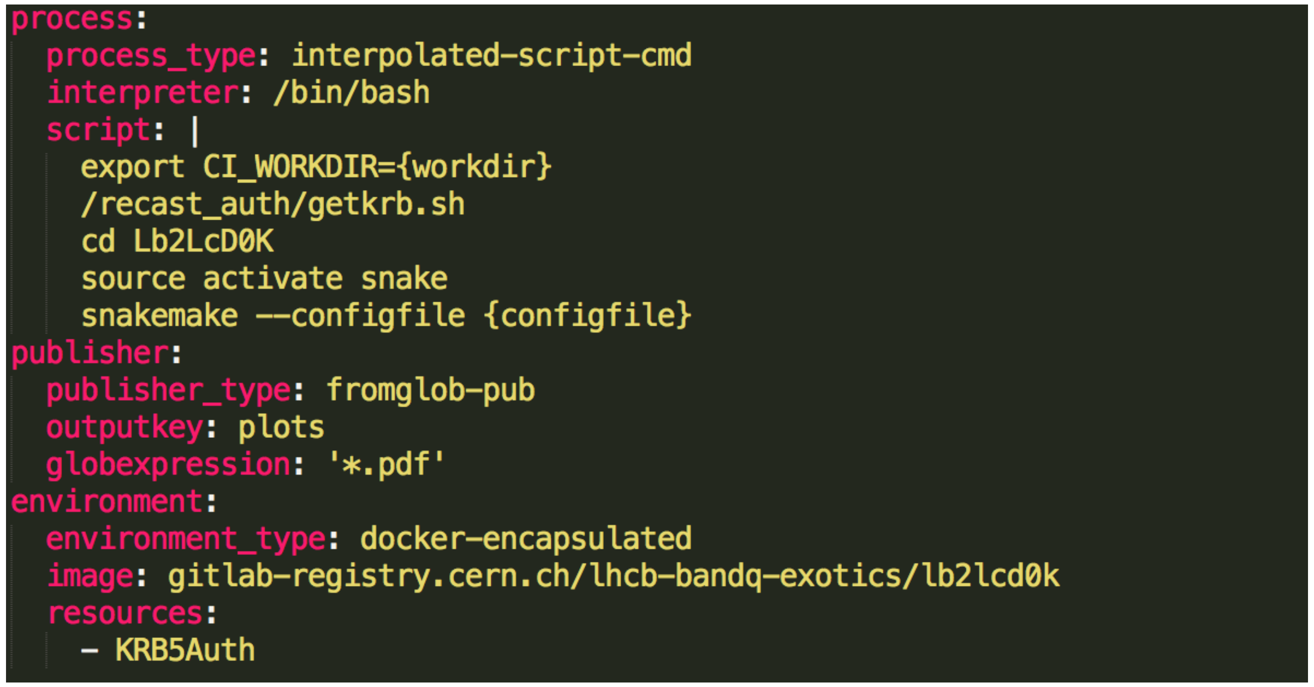
diana-hep/packtivity

packtivity

build passing pypi v0.5.7

Case Study: LHCb Lb2LcD0K analysis

- To re-run it, just need Docker image, and Kerberos Access to LHCb Data to read in centrally produces data
- (re-)produces publication quality plots
- same scripts as in CI builds, so developed **during analysis**, not additional work at end of analysis lifecycle



```yaml
process:
  process_type: interpolated-script-cmd
  interpreter: /bin/bash
  script: |
    export CI_WORKDIR={workdir}
    /recast_auth/getkrb.sh
    cd Lb2LcD0K
    source activate snake
    snakemake --configfile {configfile}
publisher:
  publisher_type: fromglob-pub
  outputkey: plots
  globexpression: '*.pdf'
environment:
  environment_type: docker-encapsulated
  image: gitlab-registry.cern.ch/lhcb-bandq-exotics/lb2lcd0k
  resources:
    - KRB5Auth
```

Semantic Step Output: List of Plots in PDF format

```
{
   “plots”: [“long”,”list”,”of”,”pdf”,”files”]
}
```

NEW YORK UNIVERSITY

diana-hep/packtivity

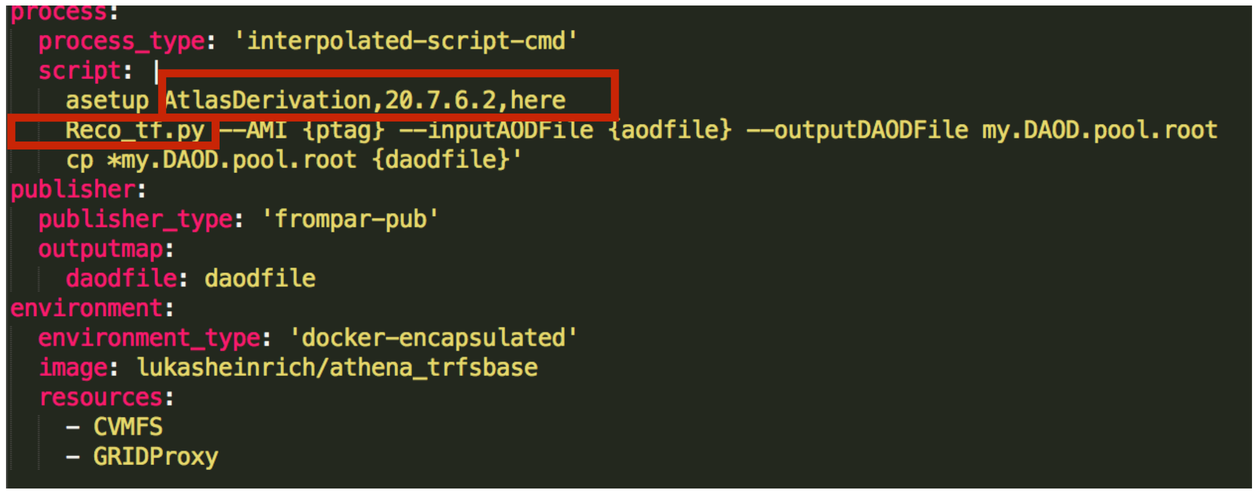
packtivity

build passing pypi v0.5.7

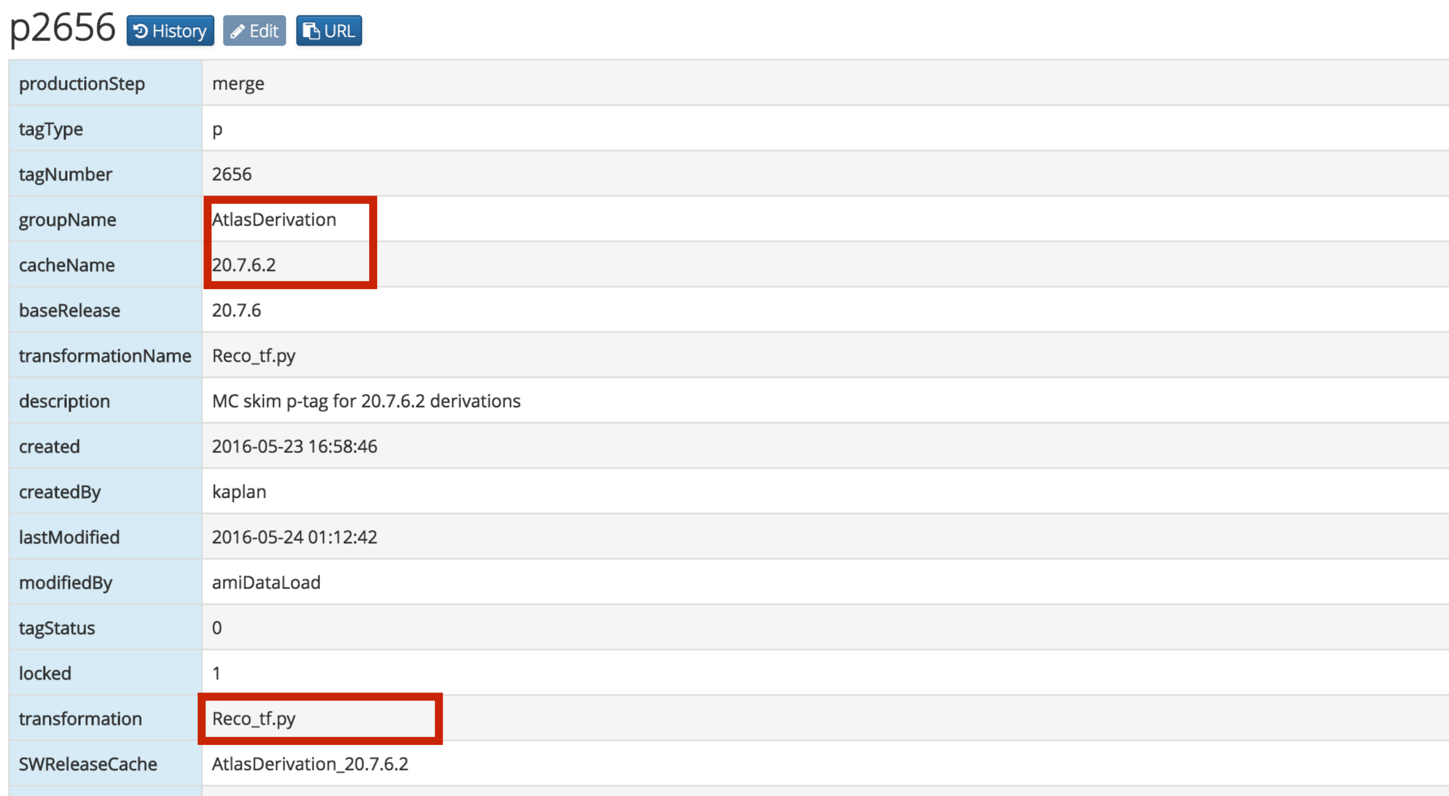
Comparison to AMI

- Very similar kind of information stored
- AMI makes some implicit assumption on software environment (not a complete VM/container image but understanding that release is setup on a machine that
- parameters of transforms not completely specified, since they can be used in many use-cases
- possible scenario: generate a given packtivity YAML/JSON based on AMI data on the fly for specific transform that is needed

- custom “data prod/AMI” environment_type and process_types (i.e. not Docker based, not scripts) from which a specific transform can be re-run when requested (e.g. even custom machines with legacy hardwar



```
process:
  process_type: 'interpolated-script-cmd'
  script: |
    asetup AtlasDerivation,20.7.6.2,here
    Reco_tf.py --AMI {ptag} --inputAODFile {aodfile} --outputDAODFile my.DAOD.pool.root
    cp *my.DAOD.pool.root {daodfile}'
publisher:
  publisher_type: 'frompar-pub'
  outputmap:
    daodfile: daodfile
environment:
  environment_type: 'docker-encapsulated'
  image: lukasheinrich/athena_trfsbase
  resources:
    - CVMFS
    - GRIDProxy
```



p2656 History Edit URL

productionStep	merge
tagType	p
tagNumber	2656
groupName	AtlasDerivation
cacheName	20.7.6.2
baseRelease	20.7.6
transformationName	Reco_tf.py
description	MC skim p-tag for 20.7.6.2 derivations
created	2016-05-23 16:58:46
createdBy	kaplan
lastModified	2016-05-24 01:12:42
modifiedBy	amiDataLoad
tagStatus	0
locked	1
transformation	Reco_tf.py
SWReleaseCache	AtlasDerivation_20.7.6.2

diana-hep/yadage

yadage - yaml based adage

pypi package 0.10.5 | build passing | health 92% | docs latest | 403.2MB 8 layers

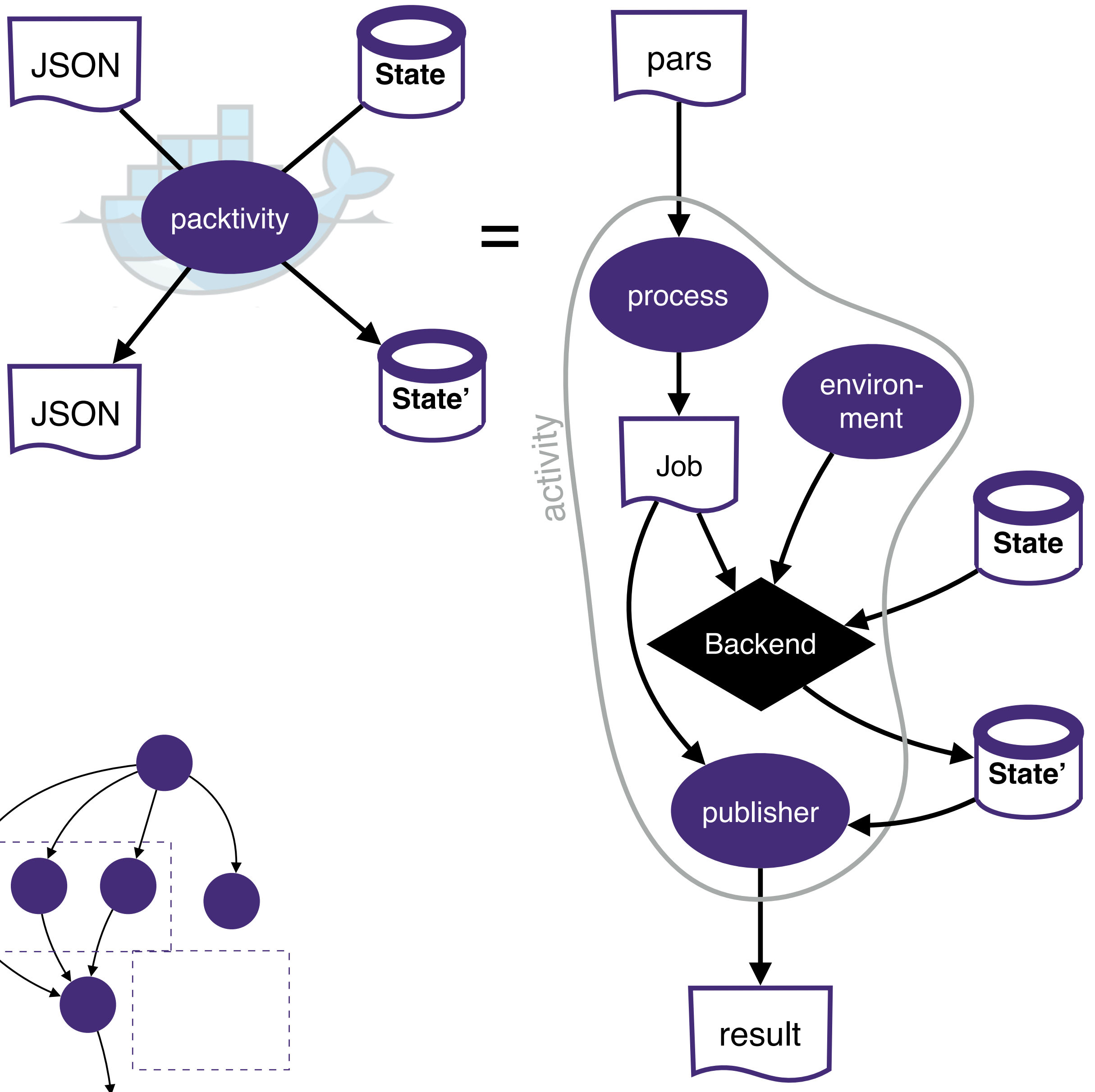
- Packtivities are like simple transforms that take JSON → JSON with side-effects on external storage
- **Workflow:** a prescription how to chain these JSON APIs, link outputs to inputs etc…

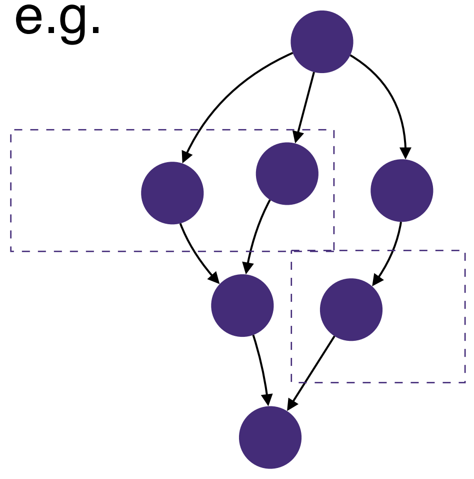
Natural Data Model: *directed acyclic graphs (DAGs)*

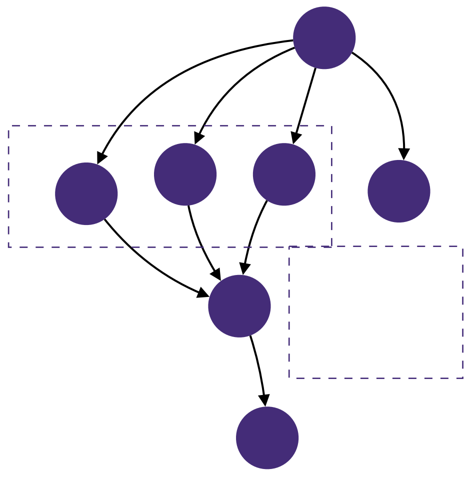
- **nodes**: individual steps
- **edges**: dependency relations

Two place where parametrization enter:

1. individual steps parametrized: "packtivities"
2. Parametrized Graph Topology:
   - variable number of created files during execution,
   - conditional choices (if/else)/flags do enable/disable steps, e.g. run systematics / not




Par. Set 1

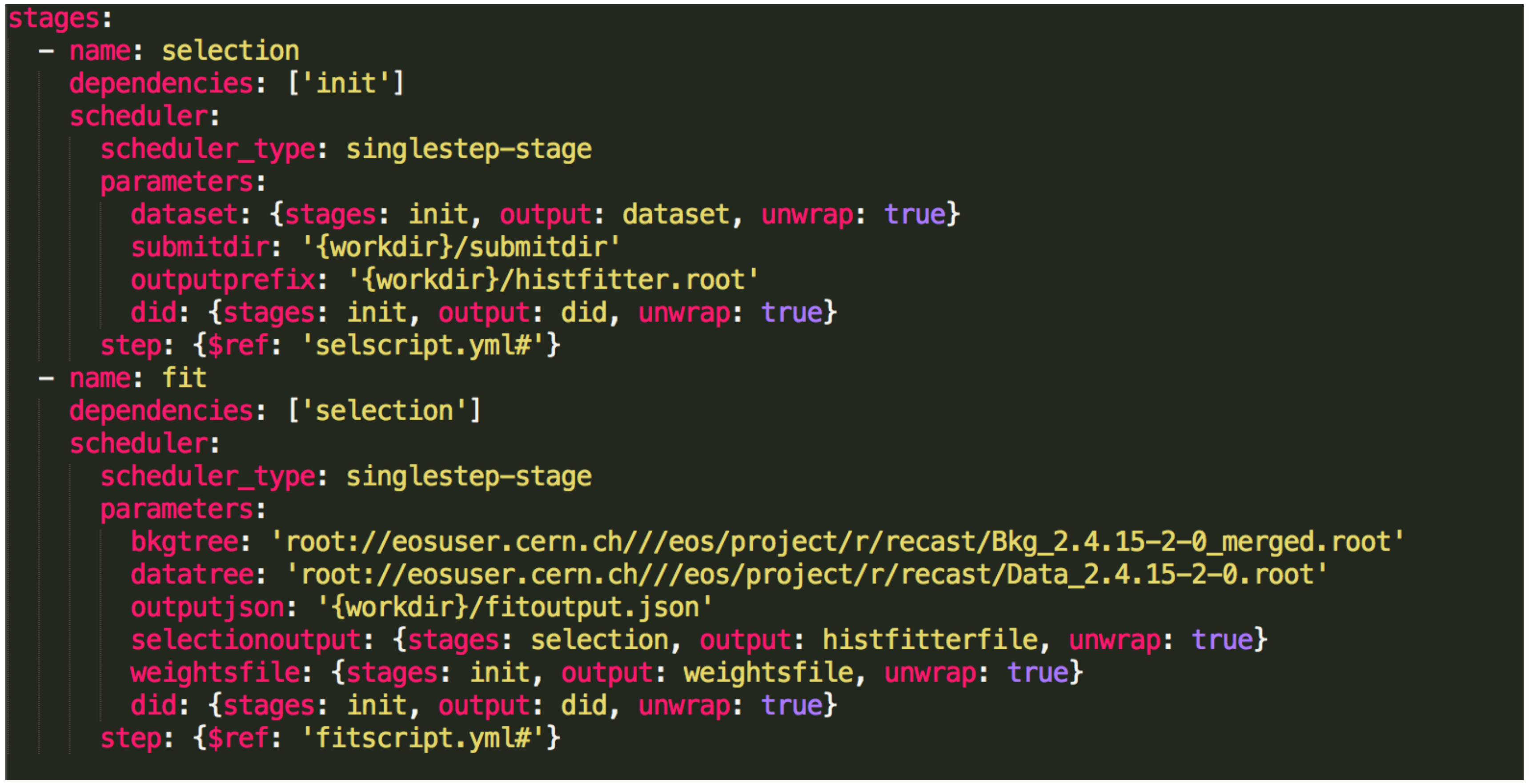

Par. Set 2

diana-hep/yadage

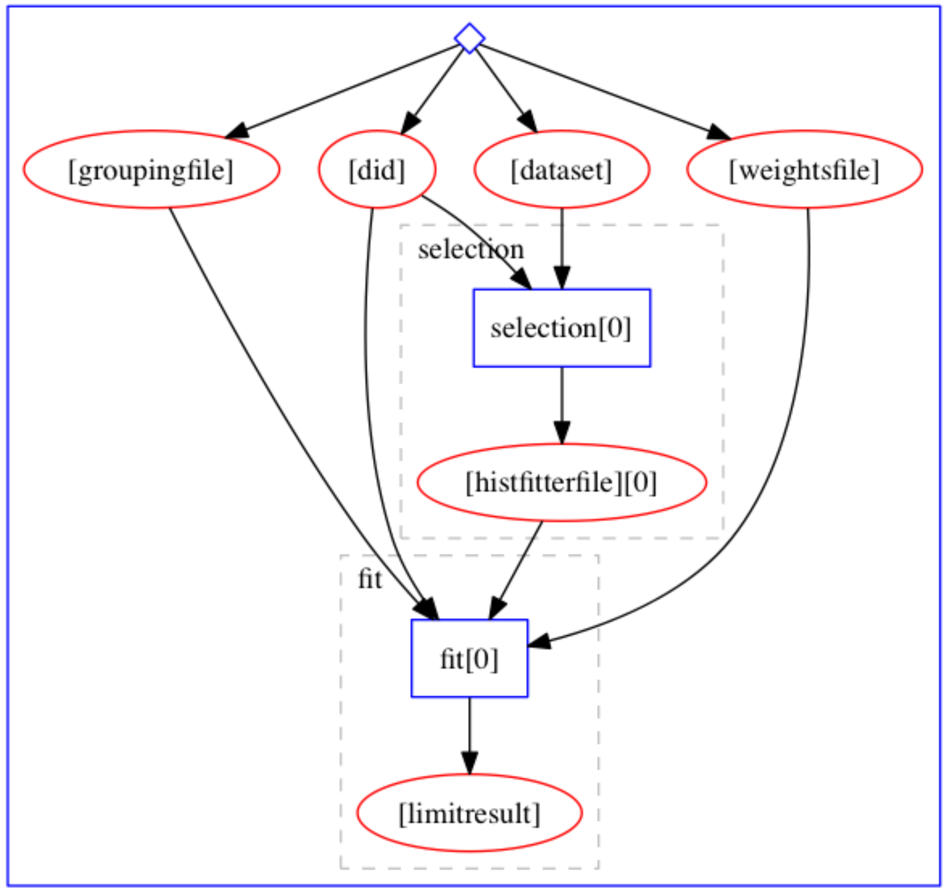
yadage - yaml based adage

pypi package 0.10.5 | build passing | health 92% | docs latest | 403.2MB 8 layers

- Short YAML File to describe workflow template
- References earlier, separately developed packtivity step YAML files
- Fit Step in this case reads Background and Data files from fixed EOS location
  can be adapted to point to ingested, long-term archived EOS location managed by CAP



```yaml
stages:
  - name: selection
    dependencies: ['init']
    scheduler:
      scheduler_type: singlestep-stage
      parameters:
        dataset: {stages: init, output: dataset, unwrap: true}
        submitdir: '{workdir}/submitdir'
        outputprefix: '{workdir}/histfitter.root'
        did: {stages: init, output: did, unwrap: true}
      step: {$ref: 'selscript.yml#'}
  - name: fit
    dependencies: ['selection']
    scheduler:
      scheduler_type: singlestep-stage
      parameters:
        bkgtree: 'root://eosuser.cern.ch///eos/project/r/recast/Bkg_2.4.15-2-0_merged.root'
        datatree: 'root://eosuser.cern.ch///eos/project/r/recast/Data_2.4.15-2-0.root'
        outputjson: '{workdir}/fitoutput.json'
        selectionoutput: {stages: selection, output: histfitterfile, unwrap: true}
        weightsfile: {stages: init, output: weightsfile, unwrap: true}
        did: {stages: init, output: did, unwrap: true}
      step: {$ref: 'fitscript.yml#'}
```

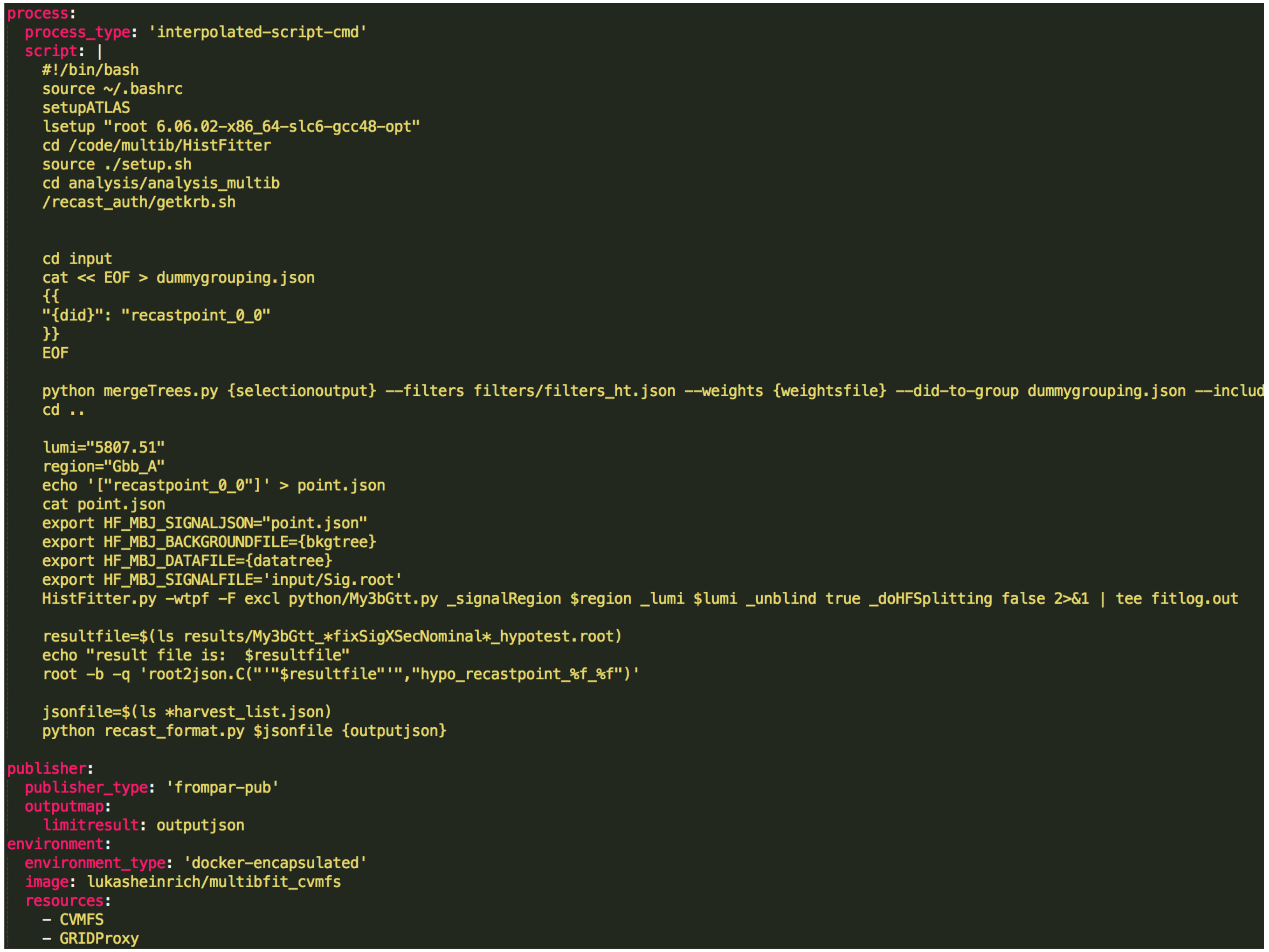


diana-hep/yadage

yadage - yaml based adage

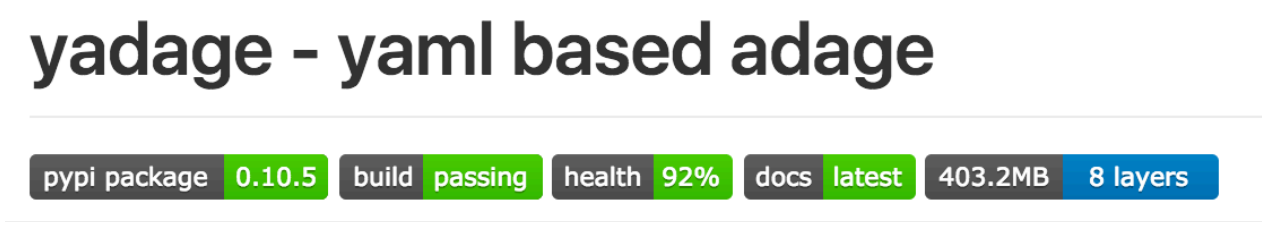
pypi package 0.10.5 build passing health 92% docs latest 403.2MB 8 layers

Preserving the Mess:
this is the fit script, more or less copy, pasted from E-Mail sent to me describing how to run the Fit (same Run-II analysis as Event Selection
but completely different team, other ROOT version, etc…)



```
process:
  process_type: 'interpolated-script-cmd'
  script: |
    #!/bin/bash
    source ~/.bashrc
    setupATLAS
    lsetup "root 6.06.02-x86_64-slc6-gcc48-opt"
    cd /code/multib/HistFitter
    source ./setup.sh
    cd analysis/analysis_multib
    /recast_auth/getkrb.sh


    cd input
    cat << EOF > dummygrouping.json
    {{
    "{did}": "recastpoint_0_0"
    }}
    EOF

    python mergeTrees.py {selectionoutput} --filters filters/filters_ht.json --weights {weightsfile} --did-to-group dummygrouping.json --includ
    cd ..

    lumi="5807.51"
    region="Gbb_A"
    echo '["recastpoint_0_0"]' > point.json
    cat point.json
    export HF_MBJ_SIGNALJSON="point.json"
    export HF_MBJ_BACKGROUNDFILE={bkgtree}
    export HF_MBJ_DATAFILE={datatree}
    export HF_MBJ_SIGNALFILE='input/Sig.root'
    HistFitter.py -wtpf -F excl python/My3bGtt.py _signalRegion $region _lumi $lumi _unblind true _doHFSplitting false 2>&1 | tee fitlog.out

    resultfile=$(ls results/My3bGtt_*fixSigXSecNominal*_hypotest.root)
    echo "result file is:  $resultfile"
    root -b -q 'root2json.C("'"$resultfile"'","hypo_recastpoint_%f_%f")'

    jsonfile=$(ls *harvest_list.json)
    python recast_format.py $jsonfile {outputjson}

publisher:
  publisher_type: 'frompar-pub'
  outputmap:
    limitresult: outputjson
environment:
  environment_type: 'docker-encapsulated'
  image: lukasheinrich/multibfit_cvmfs
  resources:
    - CVMFS
    - GRIDProxy
```

diana-hep/yadage


yadage - yaml based adage

pypi package 0.10.5 | build passing | health 92% | docs latest | 403.2MB 8 layers

yadage features

- **dynamic, parametrized** (i.e. not a fixed pipeline)
- **multiple backends** (depending on environment types and external resources), Docker on laptop, on Kubernetes on Google Container Engine, Amazon, Rackspace, etc.…
- **independently composable**(workflows can reference other workflows, but no coordination necessary (target name collisions etc…) each workflow sandboxed. Can build e.g.
- **no DSL**, pure JSON schemas, standard JSON technologies (JSON references, JSON pointers, jq query, JSONPath…)
  - i.e. indexable and searchable
  - machine readable *and writable* (i.e. automated compositions)
  - natively fits into CAP
- **distributed computing** (many hosts writing to distributed storage, just JSON message passing)

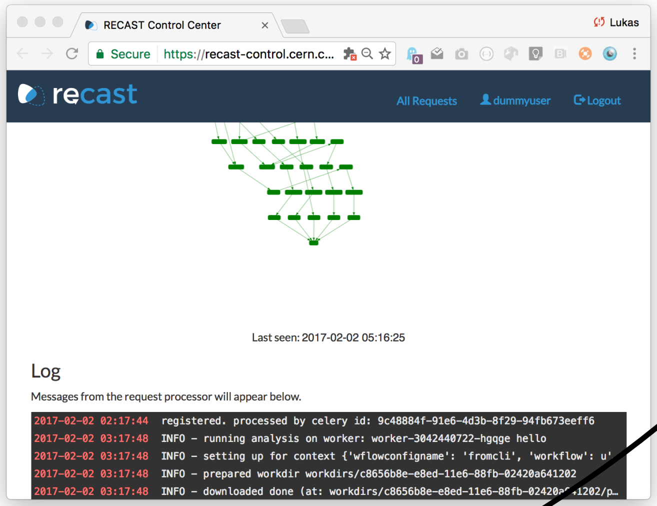
RECAST usage of CAP / yadage

Current Infrastructure

RECAST Control Center

https://recast-control.cern.ch

- Interface for Collaboration members
- Review (accept/reject) Requests
- Submit Workflow Processing



Last seen: 2017-02-02 05:16:25

Log

Messages from the request processor will appear below.

```
2017-02-02 02:17:44  registered. processed by celery id: 9c48884f-91e6-4d3b-8f29-94fb673eeff6
2017-02-02 03:17:48  INFO - running analysis on worker: worker-3042440722-hgqge hello
2017-02-02 03:17:48  INFO - setting up for context {'wflowconfigname': 'fromcli', 'workflow': u'
2017-02-02 03:17:48  INFO - prepared workdir workdirs/c8656b8e-e8ed-11e6-88fb-02420a641202
2017-02-02 03:17:48  INFO - downloaded done (at: workdirs/c8656b8e-e8ed-11e6-88fb-02420a641202/p…
```

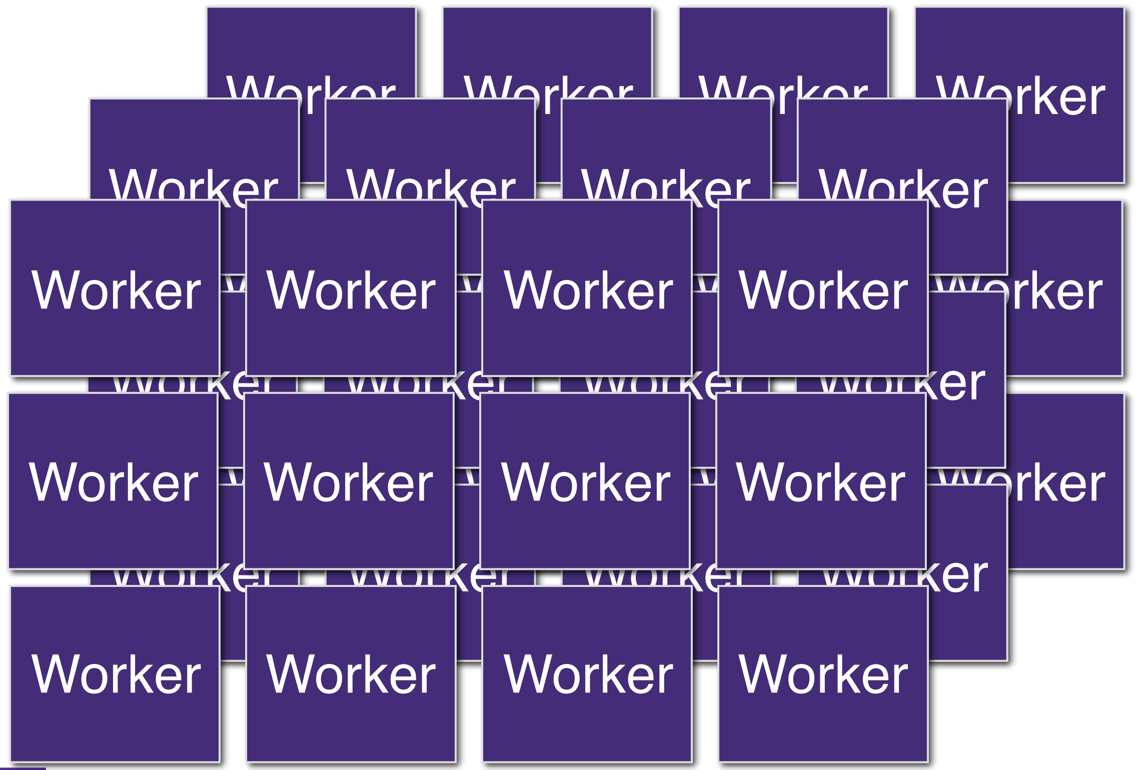
when approved & reviewed upload results to frontend

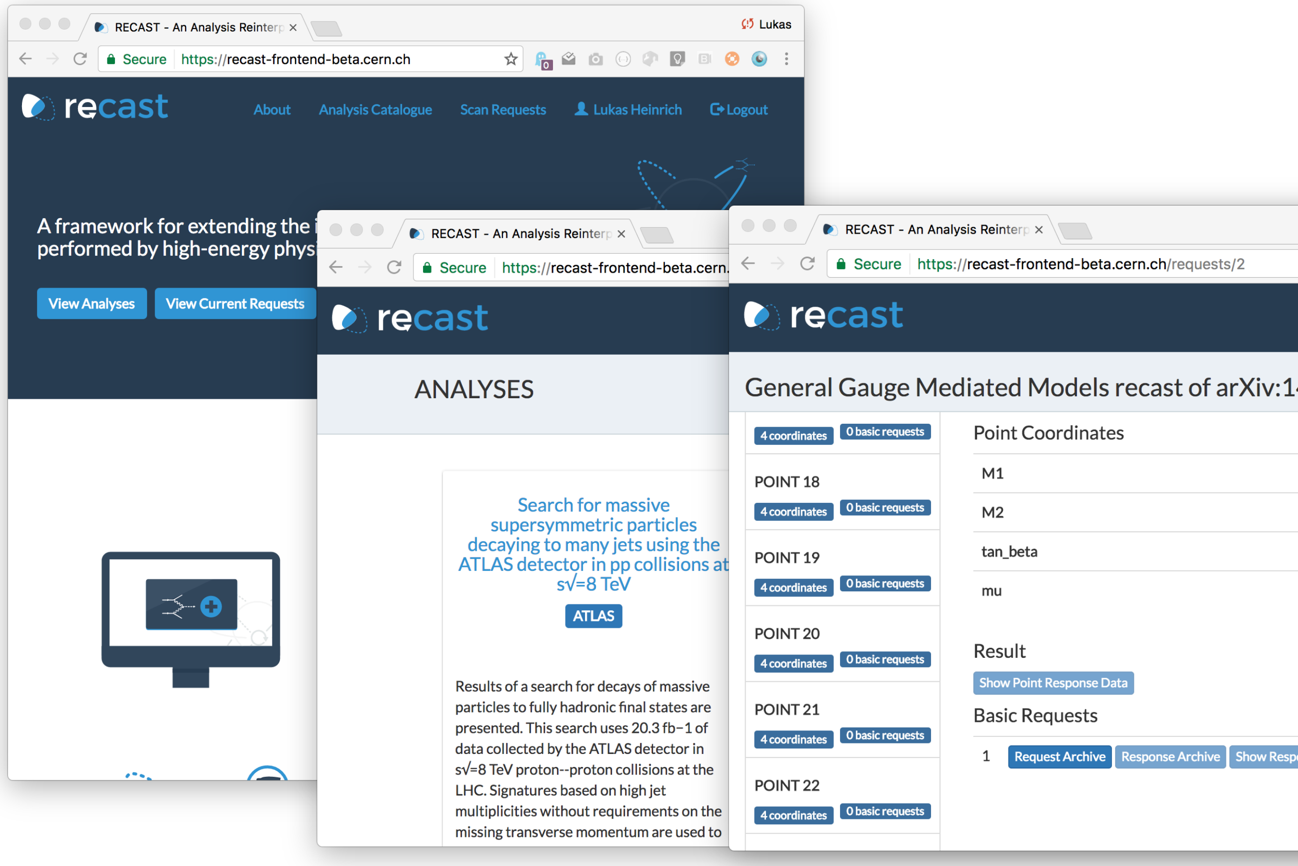
RECAST Frontend

https://recast-frontend-beta.cern.ch

- Interface for Theorists, i.e. ppl outside of collaborations to submit requests
- Describe Request, upload necessary data fragments (Model files, parameter files, pre-generated events in LHE format)
- Authentication via ORCID

- Cluster that can runs workflows of container workloads





A framework for extending the
performed by high-energy phys

ANALYSES

Search for massive supersymmetric particles decaying to many jets using the ATLAS detector in pp collisions at s√=8 TeV

Results of a search for decays of massive particles to fully hadronic final states are presented. This search uses 20.3 fb−1 of data collected by the ATLAS detector in s√=8 TeV proton--proton collisions at the LHC. Signatures based on high jet multiplicities without requirements on the missing transverse momentum are used to

General Gauge Mediated Models recast of arXiv:1

POINT 18
POINT 19
POINT 20
POINT 21
POINT 22

Point Coordinates

M1
M2
tan_beta
mu

Result

Basic Requests

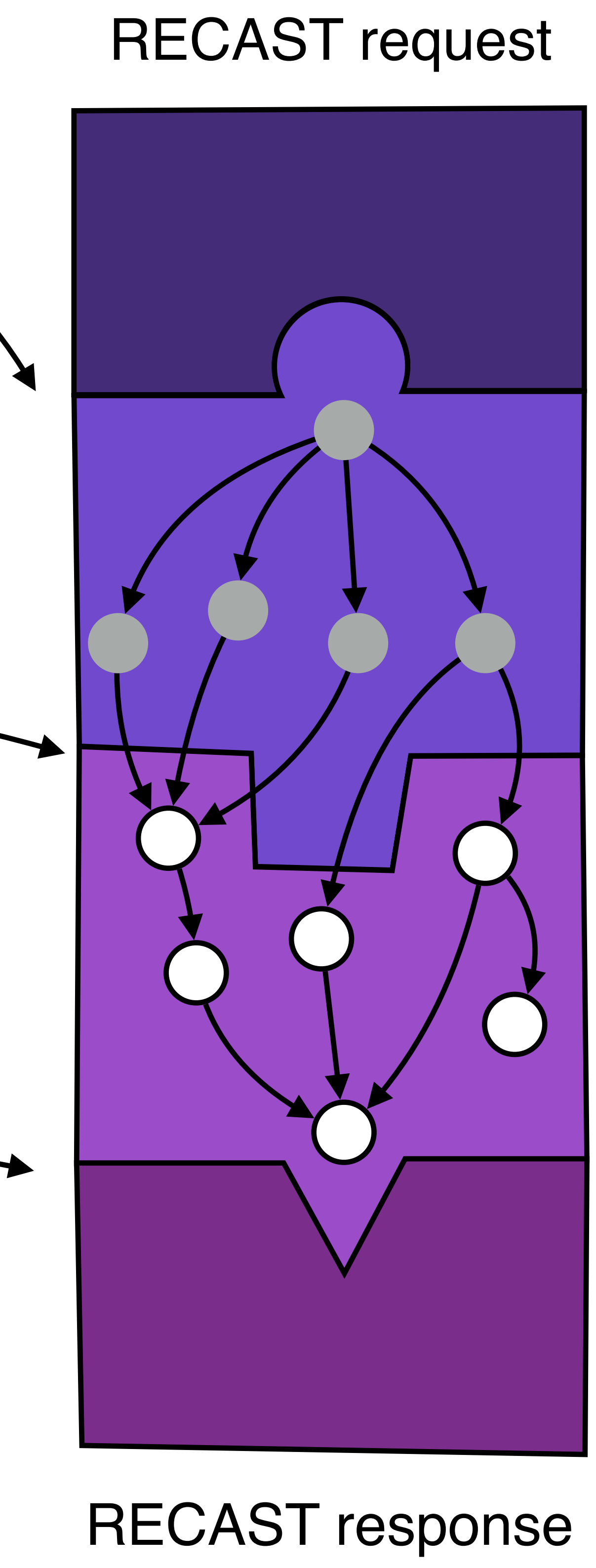
1

Satisfying Multiple Interfaces

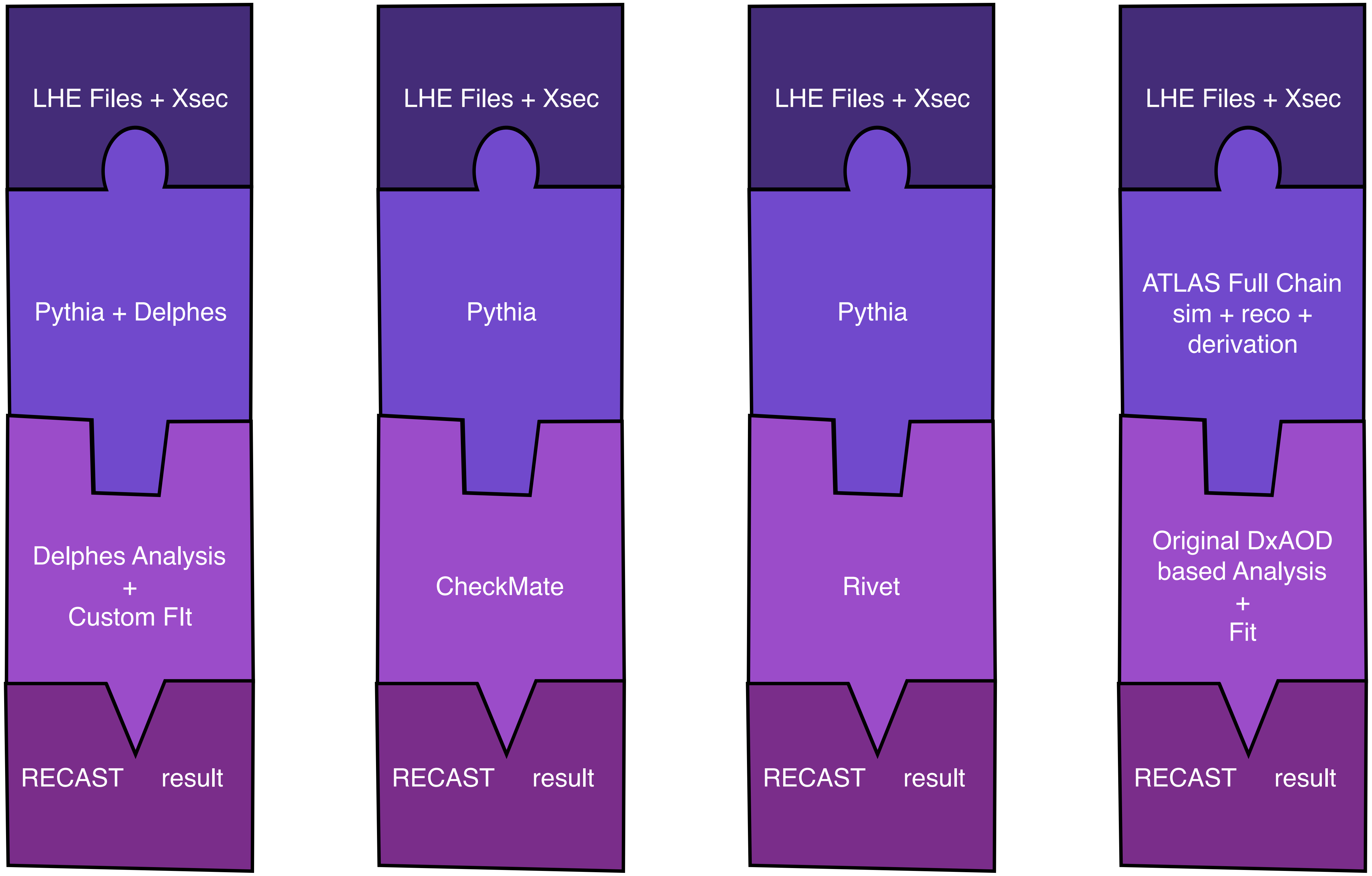
1. **RECAST Request <-> Event Generation Interface**
   - **multiple options:** LHE files, HepMC files, SLHA Parameter Cards, etc, may depend on downstream, (internally e..g ATLAS JobOptions, EVNT files)

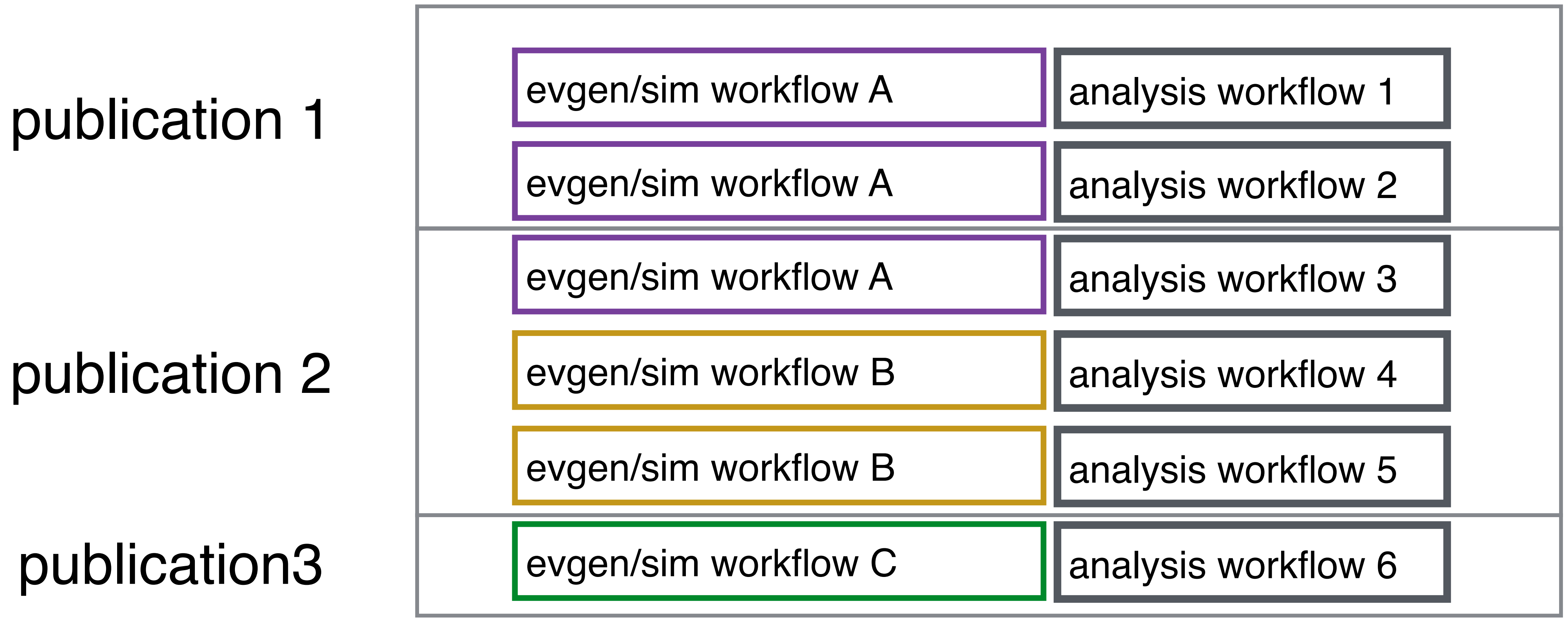
2. **EvGen / Simulation <-> Analysis Interface**
   - link between central production and analysis

3. **Analysis Interface <-> RECAST Results**
   - how to extract standard limit results (CLs values, p-values, etc..)



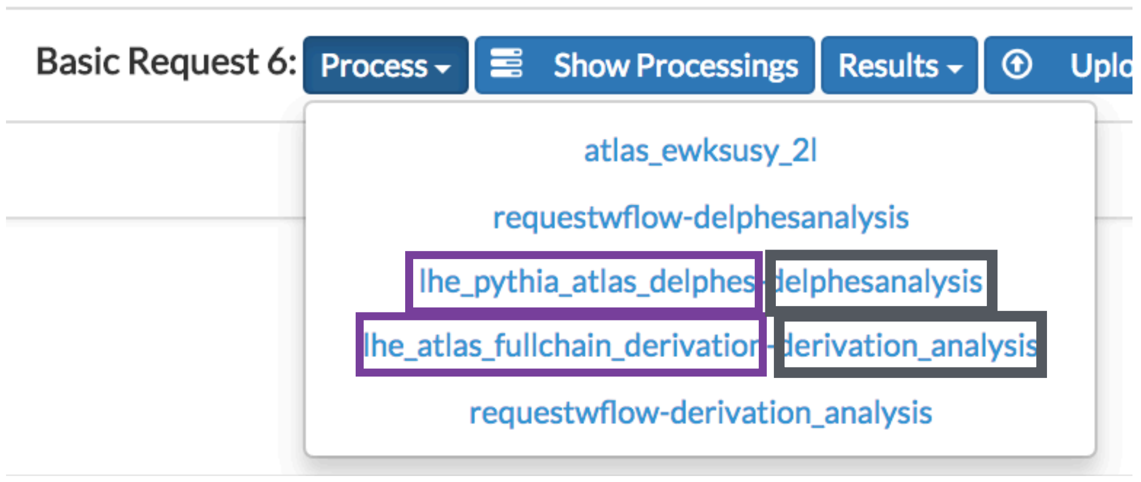
Satisfying Multiple Interfaces





- RECAST keeps a library of analysis workflows and their required upstream interfaces (what's the required input to which we attach)
- Can generate dynamically which combinations are possible

- A Publication can have multiple analysis workflows
- Example 1: many SM analyses publish independent Rivet implementation on top of original analysis code
- Example 2: RECAST specific workflow that does not re-run background or data sample, just new signal


Basic Request 6:
Process
Show Processings
Results
Uplo
atlas_ewksusy_2l
requestwflow-delphesanalysis
lhe_pythia_atlas_delphes-delphesanalysis
lhe_atlas_fullchain_derivation-derivation_analysis
requestwflow-derivation_analysis