

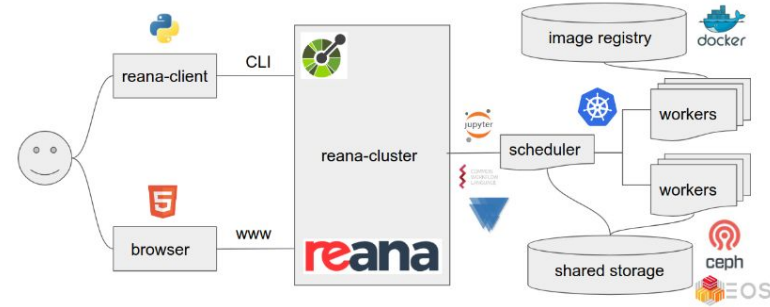
Analysis Grand Challenge at REANA

Andrii Povsten

Mentors: Alex Held, Matthew Feickert (University of Wisconsin- Madison),
Oksana Shadura (University Nebraska-Lincoln), Tibor Simko (CERN)

REANA - Reusable analysis

- Allow **complex multi-stage physics analysis to be executed with a single command, using REANA service**
- Enable to **submit parameterized computational workflows to run on remote compute clouds** or using other backends
- REANA **uses container technologies** to provide exact runtime environment necessary for various analysis steps
- Supports several **different container technologies** (Docker, Singularity), **compute clouds** (Kubernetes/OpenShift,), shared **storage systems** (Ceph, EOS) and **structured workflow specifications** (CWL, Yadage, Snakemake)



Your workflows Refreshed at 12:44:01 UTC

Search...

Status Show deleted runs Latest first

✔ snakemake-multicascading #8	finished in 20 min 45 sec
<small>Finished an hour ago</small>	<small>step 785/785</small>
✔ snakemake-multicascading #7	finished in 8 min 1 sec
<small>Finished 2 hours ago</small>	<small>step 404/404</small>
✔ test2 #1	finished in 2 min 42 sec
<small>Finished 5 hours ago</small>	<small>step 18/18</small>
✔ snakemake-multicascading #5	finished in 10 min 59 sec
<small>47.88 MiB</small>	<small>step 504/504</small>
<small>Finished a day ago</small>	
✔ snakemake-multicascading #3	finished in 20 min 14 sec
<small>56.02 MiB</small>	<small>step 604/604</small>
<small>Finished a day ago</small>	

REANA instance at CERN - <https://reana.cern.ch>

reana

Your REANA token

In order to use your token, make sure you have reana-client installed and run:

```
$ export REANA_SERVER_URL=https://reana.cern.ch
$ export REANA_ACCESS_TOKEN=
```

Your GitLab projects

Connect to GitLab
In order to integrate your GitLab projects with REANA you need to grant permissions. [Connect](#)

Your quota

CPU 0s used

Disk 0 Bytes out of 300 GiB [used 0%](#)

[Privacy notice](#) [Docs](#) [Forum](#) [Chat](#) [Cluster health](#)

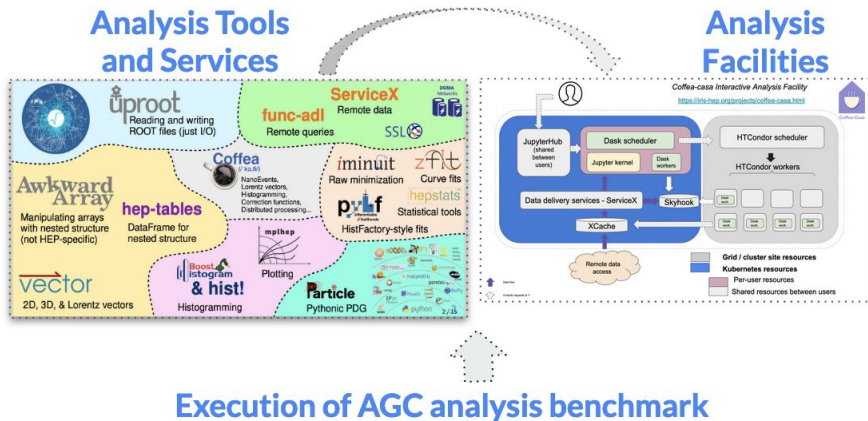
- Using custom user tokens
- CERN gitlab native integration - allows to integrate REANA into your GitLab pipelines.
- Excellent documentation <https://docs.reana.io/>
- User support: <https://forum.reana.io/>

For setting REANA client:
`pip install reana-client`

```
... success: sha256:md5:tree-sitter: has been queued
(reana) andrewpovsten@pucomphep03 cms-open-data-ttbar % reana-client ping
REANA server: https://reana.cern.ch
REANA server version: 0.9.2
REANA client version: 0.9.1
Authenticated as: Andrii Povsten <andrii.povsten@cern.ch>
Status: Connected
(reana) andrewpovsten@pucomphep03 cms-open-data-ttbar %
```

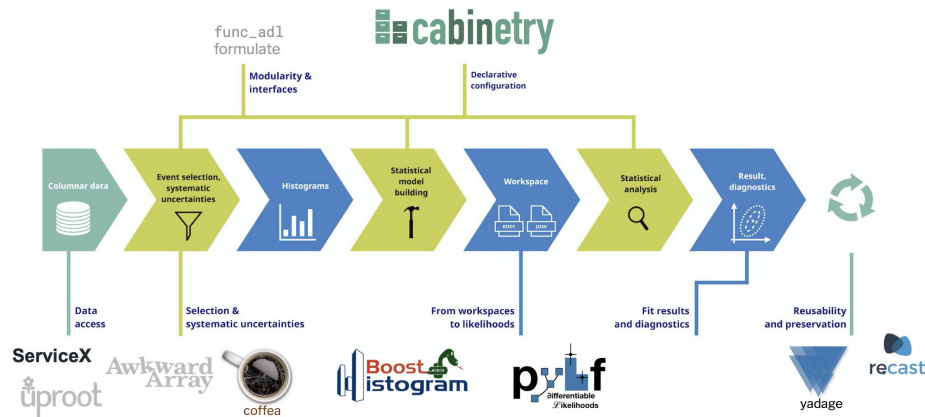
Analysis Grand Challenge (AGC)

- **Main AGC analysis task:** *t*tbar cross-section measurement in single lepton channel
 - Includes single top reconstruction
- **AGC goal** to allow coping with HL-LHC data-size by rethinking data pipeline (e.g. increasingly complex analysis benchmark)
 - As well to provide flexible, easy-to-use, low latency analysis facilities



Analysis Grand Challenge

- Columnar data extraction from large dataset
 - Processing of that data (event filtering, construction of observables, evaluation of systematic uncertainties) into histograms
 - Statistical model construction and statistical inference
 - Relevant visualisation for this steps
- + **Adding analysis preservation step to AGC pipeline**



Next step: REANA specification file

The REANA reproducible analysis platform requires to have `reana.yaml` file present in your analysis source code (REANA specification file).

Its purpose is to answer the **Four Questions**:

1. What is your input data? (e.g. dataset samples)

2. What is your analysis code? (e.g. python notebook, compiled executable, script)

3. What is your computing environment? (e.g. docker image)

4. Which computational steps do you take to arrive at results? (e.g. data processing or statistical model construction and statistical inference)

Next step: Workflow management engine choice

- Our choice was to use **Snakemake workflow management system (integrated in REANA)**
 - Help to keep a record of used scripts and their input dependencies
 - Run multiple steps in sequence, parallelising where possible
 - Automatically detect if something changes and then reprocess data if needed
- **Snakemake key feature is a “rule” description, which enables the parallelisation within REANA**, running each rule in a separate virtual node.
- **Snakemake allows you to create a set of rules, each one defining a “step” of your analysis.**
- The rules need to be written in a file called Snakefile:
 - *The input*: Data files, scripts, executables or any other files.
 - *The expected output*: It's not required to list all possible outputs. Just those that you want to monitor or that are used by a subsequent step as inputs.
 - *Shell*: A command to run to process the input and create the output.

Example of Snakemake rule:

```
rule ttbar_nominal_sample:

    input:

        "samples.ipynb"

    output:

        "histograms_ttbar_nominal.root"

    params:

        key_to_extract = 'ttbar',

        variation = 'nominal'

    container:


        "povstenandrii/papermill:20231102a"

    shell:

        "/bin/bash -l && source fix-env.sh &&
        papermill {input} sample1_out.ipynb -p
        key_to_extract {params.key_to_extract} -p
        variation {params.variation} -k python3"
```

REANA specification file


Version without parallelisation



```
workflow:
  type: serial
  specification:
    steps:
      - name: demoanalyzer
        environment:
          'hub.opensciencegrid.org/iris-hep/analysis-systems-base:latest'
        commands:
          - /bin/bash -l && source fix-env.sh && python
            ttbar_analysis_pipeline.py
  outputs:
    files:
      - merged_histograms.root
```

Extremely long execution time since AGC was written in mind without support for other workflow languages

Version with parallelisation



```
inputs:
  files:
    - ttbar_analysis_pipeline.py
    - ttbar_analysis_pipeline.ipynb
    - nanaod_inputs.json
    - nanaod_branch_ratios.json
    - corrections.json
    - Snakefile
    - merged.ipynb
  directories:
    - histograms
    - models
    - utils
    - files
workflow:
  type: snakemake
  file: Snakefile
outputs:
  files:
    - merged_histograms.root
```


AGC notebooks modification:

- Rerun the same notebook n-times but with different parameters => instead of processing all files, samples we process one sample with one file
- Firstly we parallelized each sample from fileset:

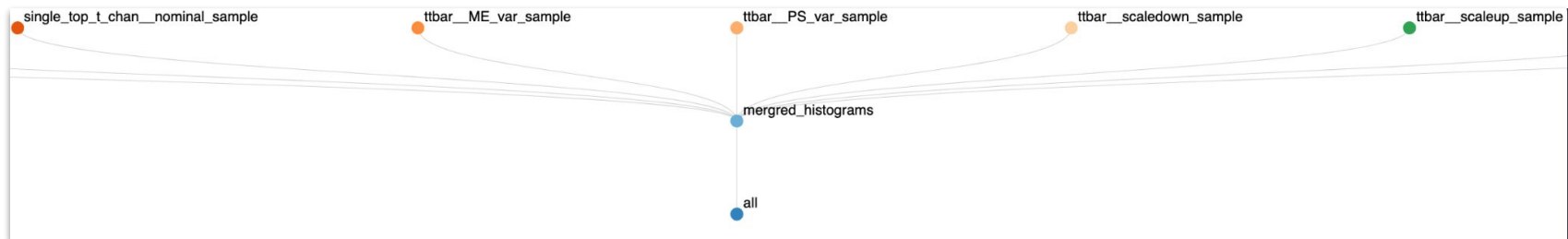
```
original_dict = fileset
selected_file = original_dict[sample_name]['files']
new_dict = {sample_name: {'files': [filename], 'metadata': original_dict[sample_name]['metadata']}}
```

- Second, parallelize each file for each sample:

```
all_histograms, metrics = run(
    fileset={sample_name: new_dict[sample_name]},
    treename=treename,
    processor_instance=TtbarAnalysis(USE_INFERENCE, USE_TRITON)
)
```

The main idea is to see the whole picture of your analysis what steps suppose to be after another and modify it on the early stages to have a separate pieces which could be count as 1 job.

Analysis Grand Challenge pipeline: Adapting to Snakemake



Each rule REANA sends to the Kubernetes cluster as separate node

analyse file_ttbar_01 file_ttbar_02 file ... file_wjets_01 file_wjets_02 file_wjets_03 ...

\ | /

\ | /

merge sample ttbar_nominal

merge sample wjets_nominal

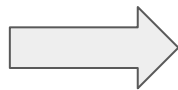
\

/

merge all samples

|

Plot

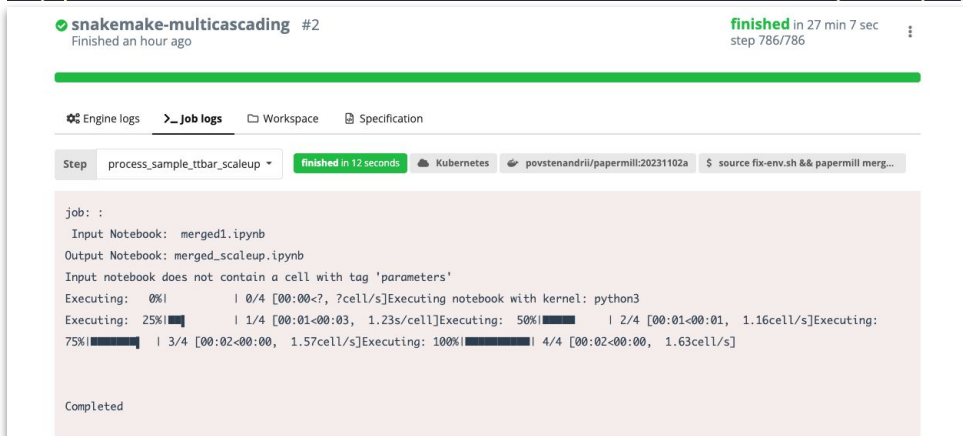


Snakemake checks the inputs and outputs in the rules to see the dependencies and order of execution

Next step: Execute REANA workflow

To run the workflow use the: `reana-client run -f reana.yaml -w your-workflow`

```
(reana) andrewpovsten@pucomphep03 cms-open-data-ttbar % reana-client run -f reana-snakemake.yaml -w snakemake-multicascading  
==> Creating a workflow...  
Job stats:
```



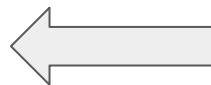
snakemake-multicascading #2
Finished an hour ago finished in 27 min 7 sec
step 786/786

Engine logs > Job logs Workspace Specification

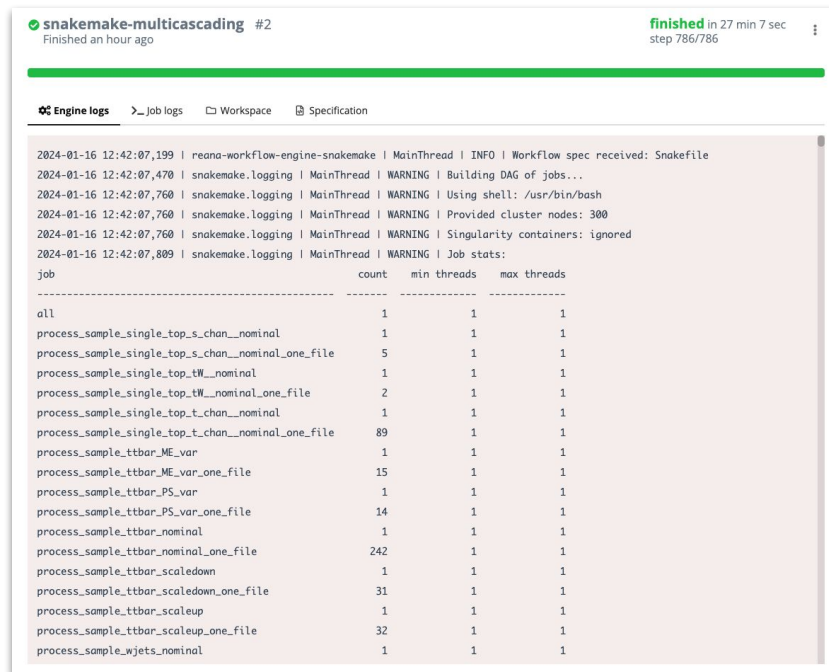
Step process_sample_ttbar_scaleup finished in 12 seconds Kubernetes povstenandrii/papermill:20231102a \$ source fix-env.sh && papermill merg...

```
job: :  
Input Notebook: merged1.ipynb  
Output Notebook: merged_scaleup.ipynb  
Input notebook does not contain a cell with tag 'parameters'  
Executing: 0% | 0/4 [00:00<?, ?cell/s]Executing notebook with kernel: python3  
Executing: 25% | 1/4 [00:01<00:03, 1.23s/cell]Executing: 50% | 2/4 [00:01<00:01, 1.16cell/s]Executing:  
75% | 3/4 [00:02<00:00, 1.57cell/s]Executing: 100% | 4/4 [00:02<00:00, 1.63cell/s]
```

Completed



Job logs shows the process and full workflow for each step(rule)

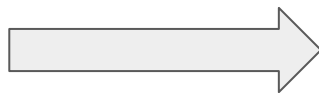


snakemake-multicascading #2
Finished an hour ago finished in 27 min 7 sec
step 786/786

Engine logs > Job logs Workspace Specification

```
2024-01-16 12:42:07,199 | reana-workflow-engine-snakemake | MainThread | INFO | Workflow spec received: Snakefile  
2024-01-16 12:42:07,470 | snakemake.logging | MainThread | WARNING | Building DAG of jobs...  
2024-01-16 12:42:07,760 | snakemake.logging | MainThread | WARNING | Using shell: /usr/bin/bash  
2024-01-16 12:42:07,760 | snakemake.logging | MainThread | WARNING | Provided cluster nodes: 300  
2024-01-16 12:42:07,760 | snakemake.logging | MainThread | WARNING | Singularity containers: ignored  
2024-01-16 12:42:07,809 | snakemake.logging | MainThread | WARNING | Job stats:
```

job	count	min threads	max threads
all	1	1	1
process_sample_single_top_s_chan_nominal	1	1	1
process_sample_single_top_s_chan_nominal_one_file	5	1	1
process_sample_single_top_tW_nominal	1	1	1
process_sample_single_top_tW_nominal_one_file	2	1	1
process_sample_single_top_t_chan_nominal	1	1	1
process_sample_single_top_t_chan_nominal_one_file	89	1	1
process_sample_ttbar_ME_var	1	1	1
process_sample_ttbar_ME_var_one_file	15	1	1
process_sample_ttbar_PS_var	1	1	1
process_sample_ttbar_PS_var_one_file	14	1	1
process_sample_ttbar_nominal	1	1	1
process_sample_ttbar_nominal_one_file	242	1	1
process_sample_ttbar_scaledown	1	1	1
process_sample_ttbar_scaledown_one_file	31	1	1
process_sample_ttbar_scaleup	1	1	1
process_sample_ttbar_scaleup_one_file	32	1	1
process_sample_wjets_nominal	1	1	1



Engine logs: shows information about steps scheduling.

Conclusion

- Successfully implement the AGC at REANA
- Get parameterized AGC notebook and execute it with papermill tool

Future Tasks

- Identify the possible ways of improving AGC adoption for REANA workflow and RECAST
- Demonstrate HEPData submission of AGC artefacts for reusability
- Further optimisation of AGC processing time in REANA
- Testing on larger clusters
- Testing AGC ServiceX and machine learning pipelines in REANA