Snakemake

Valentin Pestel,

KM3NeT's Grid tutorial & workshop

25/02/2025

KM3NeT

LPC CAEN

Why a workflow management system

To achieve your goal, you need complexe workflows:

- Getting the data, pre-processing, extracting information, all that for each runs, then merging
- You can use bash script, but…

In the ideal world, you'd like your worfklow:

- Clear and understandable
  - everything is explicit, I understand and can explain what happen in which order
- Flexible
  - easy re-usability of already existing steps
- Performant and scalable
  - parallelize task that can be, offer solution to monitor the resources needed
- Portable and reproducible
  - Easy to port on another system, easy to reproduce by somebody else

Snakemake

The Snakemake workflow management system is a tool to create reproducible and scalable data analyses. Workflows are described via a human readable, Python based language. They can be seamlessly scaled to server, cluster, grid and cloud environments, without the need to modify the workflow definition. Finally, Snakemake workflows can entail a description of required software, which will be automatically deployed to any execution environment.

From Snakemake documention

Core principles

Snakemake language is built on Python:

- Snakemake workflow can integrate python code

Workflow expressed as individual “rule”:

- Minimal implementation is `input`, `output`, and `shell`

To build a specific target, Snakemake construct the list of step to be executed:

- Works like Makefile
- Construction of a Directed Acyclic Graph of actions, of DAG

```
rule my_step:
    output:
        "something_new.txt"
    input:
        "something_old.txt"
    shell:
        """
        # some bash code
        cat {input} > {output}
        """
```

```
rule my_step:
    output:
        "old_{label}.txt"
    input:
        "new_{label}.txt"
    …
```

Snakemake session

Session splitted in 3 parts:

- Snakemake basics
  - Walk through basic features with 5 incrementally complex examples
- Snakemake x Rucio
  - Illustration of high-level usage with Rucio as data provider
  - Exercice -> you'll be the one working
- Overview of snakemake workflows in KM3NeT

Snakemake walk-through

Clone the git repository:

- `git clone git@git.km3net.de:vpestel/snakemake_basics.git`

Source the micromamba environment

- `source /cvmfs/km3net.egi.eu/micromamba/micromamba_x86.sh`
- `micromamba activate /sps/km3net/users/vpestel/education/KM3NeT_GRID_workshop_2025/envs/snakemake_basics`

Alternatively, you can build the environment yourself:

- micromamba install -f environment.yaml -p /path/you/want

Ready to level-up ?

Snakemake X Rucio : exercice

Goal:

- Building a workflow that takes a dataset as input, and perform actions on all the files it contains
- Actions -> histograms if you feel like it

Snakemake Rucio exercice

Clone the git repository:

- `git clone git@git.km3net.de:vpestel/snakemake_rucio_example.git`

Source the micromamba environment

- `source /cvmfs/km3net.egi.eu/micromamba/micromamba_x86.sh`
- `micromamba activate /sps/km3net/users/vpestel/education/KM3NeT_GRID_workshop_2025/envs/snakemake_rucio_example/`

Alternatively, you can build the environment yourself:

- micromamba install -f environment.yaml -p /path/you/want

Ready to level-up ?

KM3NeT Snakemake workflows

A handful of examples already in use in the collaboration:

- Run-based data processing
- Calibration workflows

Moving to snakemake v8

Some changes on snakemake side :

- executor-plugins

Some new features

Declare explicitly true detector (i.e. use for MC generation):

- Previously used the offline (reconstruction) detector file

Calibration “menus” for the reconstructions