EURO-LABS BOOTCAMP
Snakemake Introduction

A. Matta, V. Pestel,
LPC Caen, CNRS/IN2P3

GANIL, 2th-6th February 2026

EUROPEAN LABORATORIES FOR ACCELERATOR BASED SCIENCES

Financé par l'Union européenne

Workflow

What is it?

- All the steps to produce a results
  1. run_i.dat + cal.txt → calibrated_i.root
  2. calibrated_i.root + param.txt → histogram_i.root
  3. sum all histogram_i.root → final_histogram.root

Workflow

What is it?

- All the steps to produce a results
  1. run_i.dat + cal.txt → calibrated_i.root
  2. calibrated_i.root + param.txt → histogram_i.root
  3. sum all histogram_i.root → final_histogram.root

Automation

- "Code" your workflow:
  - → Re-run easily
  - → Error free
  - → Distribute

Workflow management options

README

- ✓ widely available
- ✓ easy to read & write
- ✗ hard to re-run
- ✗ hard to maintain

Workflow management options

README

- ✓ widely available
- ✓ easy to read & write
- ✗ hard to re-run
- ✗ hard to maintain

bash scripts

- ✓ widely available
- ✗ dependencies is hard to code
- ✗ hard to read &code
- ✗ does not scale
- ✗ hard to port

Workflow management options

README

- ✓ widely available
- ✓ easy to read & write
- ✗ hard to re-run
- ✗ hard to maintain

make

- ✓ widely available
- ✓ dependencies are taken into account
- ✗ hard to read & code
- ✗ does not scale

bash scripts

- ✓ widely available
- ✗ dependencies is hard to code
- ✗ hard to read &code
- ✗ does not scale
- ✗ hard to port

Workflow management options

README

- ✓ widely available
- ✓ easy to read & write
- ✗ hard to re-run
- ✗ hard to maintain

make

- ✓ widely available
- ✓ dependencies are taken into account
- ✗ hard to read & code
- ✗ does not scale

bash scripts

- ✓ widely available
- ✗ dependencies is hard to code
- ✗ hard to read &code
- ✗ does not scale
- ✗ hard to port

snakemake

- ✗ need a specific install
- ✓ dependencies are taken into account
- ✓ easy to read & code : python based
- ✓ scale easily : run locally or on clusters via slurm
- ✓ env & container friendly

To Push or to Pull? That is the question!

Push model

Start from inputs:

- for every run file, do the calibration, then
- for every calibrated file, do histograms, then
- sum all histograms.

→ Typically what is done in bash.

To Push or to Pull? That is the question!

Push model

Start from inputs:

- for every run file, do the calibration, then
- for every calibrated file, do histograms, then
- sum all histograms.

→ Typically what is done in bash.

Pull model

Start from outputs:

- final_histogram.root needs histogram_i.root, does it exist?

→ No: histogram_i.root needs calibrated_i.root, does it exist?

→ No: calibrated_i.root needs run_i.root, does it exist?

→ Yes: produce calibrated_i.root, then histogram_i.root, then final_histogram.root

→ What is done in make and snakemake

Snakemake

snakemake.readthedocs.io

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.

Snakemake

snakemake.readthedocs.io

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.

Anatomy of a Snakefile

- set of rule to produce files
- pattern to deduce **inputs filename based on outputs filename**
- file are produced on a need basis, a file need to be an input of a rule to be produced
  - → a phony rule with no outputs is used to **trigger production**

a Snakefile Example

```
rule phony:
    input:
        "histograms/hist_1.root",
        "histograms/hist_2.root",
        "histograms/hist_3.root",
        "histograms/hist_4.root",
        "histograms/hist_5.root"

rule simulation:
    output:
        "simulation/run_{run}.root"
    shell:
        """
        simulator --output {output}
        """

rule make_histograms:
    output:
        "histograms/hist_{run}.root"
    input:
        "simulation/run_{run}.root"
    shell:
        """
        histogrammer --input {intpu} --output {output}
        """
```

Snakemake

snakemake.readthedocs.io

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.

Anatomy of a Snakefile

- set of rule to produce files
- pattern to deduce **inputs filename based on outputs filename**
- file are produced on a need basis, a file need to be an input of a rule to be produced
  - → a phony rule with no outputs is used to **trigger production**

Hands on tutorial

Lets try with a few exercice!

a Snakefile Example

```
rule phony:
    input:
        "histograms/hist_1.root",
        "histograms/hist_2.root",
        "histograms/hist_3.root",
        "histograms/hist_4.root",
        "histograms/hist_5.root"

rule simulation:
    output:
        "simulation/run_{run}.root"
    shell:
        """
        simulator --output {output}
        """

rule make_histograms:
    output:
        "histograms/hist_{run}.root"
    input:
        "simulation/run_{run}.root"
    shell:
        """
        histogrammer --input {intpu} --output {output}
        """
```