Snakemake : What? Why?

- Snakemake is lightweight
- Snakemake is flexible
- Snakemake use both classical *Bash* and *Python* syntax
- Your analysis process don't have to be distorded
- Snakemake has a vast community
- Looks like *Make* but it's not : make the most of *Make* and *Python*
- Snakemake is actually usefull beyond reproducibility
- Usable on cluster with minimum effort

Snakemake : Philosophy

- Looks like a Python script
- The workflow is a set of entangled rules (inputs/outputs/shell)

The dependency between rules is based on I/O

- Computed by Snakemake based on
- Snakefile

```
rule rule_1:
    input:
        input_1 = 'file_1.txt'
    output:
        output_1 = 'file_1.out'
    shell:
        """
        So some shell command here
        I.e. echo "Hello World" > file_1.out
        """
```

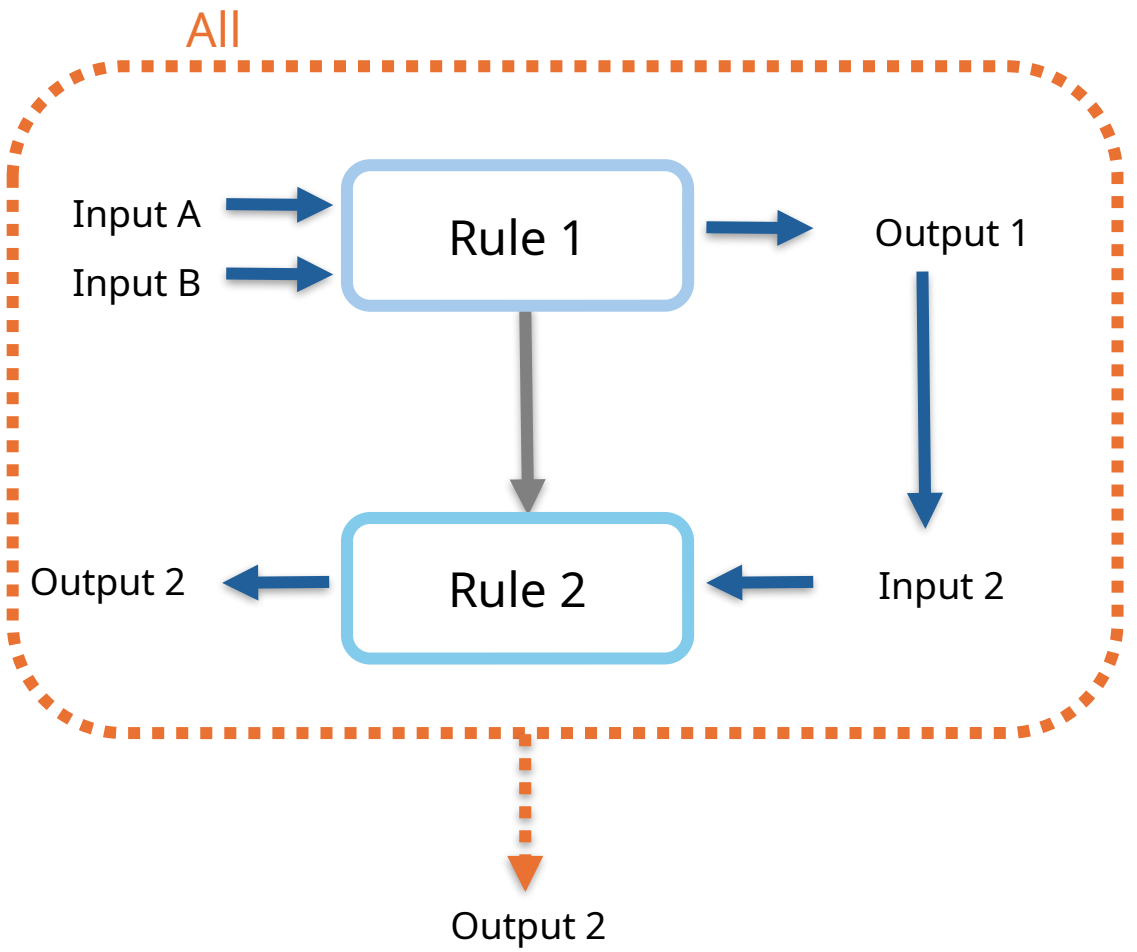
- Snakemake rules have to be defined in *Snakefile*
- Launch pipeline : *snakemake -c 1*
- Launch pipeline in dry mode : *snakemake -n*
- Plot workflow steps : *snakemake --dag | dot -T pdf > dag.pdf*
- Re-run all jobs the output of which is recognized as incomplete : *snakemake -c 1 --&erun-incomplete*

Snakemake : My first workflow

```
rule rule_1:
    input:
        input_A = 'file_a.txt'
        input_B = 'file_b.txt'
    output:
        output_1 = 'file_1.out'
    shell:
        """
        So some shell command here
        I.e. cat {input.input_A} {input.input_B} > file_1.out
        """

rule rule_2:
    input:
        input_2 = 'file_1.out'
    output:
        output_2 = 'words_total.out'
    shell:
        """
        So some shell command here
        I.e. wc -w {input.input_2} > words_total.out
        """

rule all:
    input:
        'words_total.out'
```

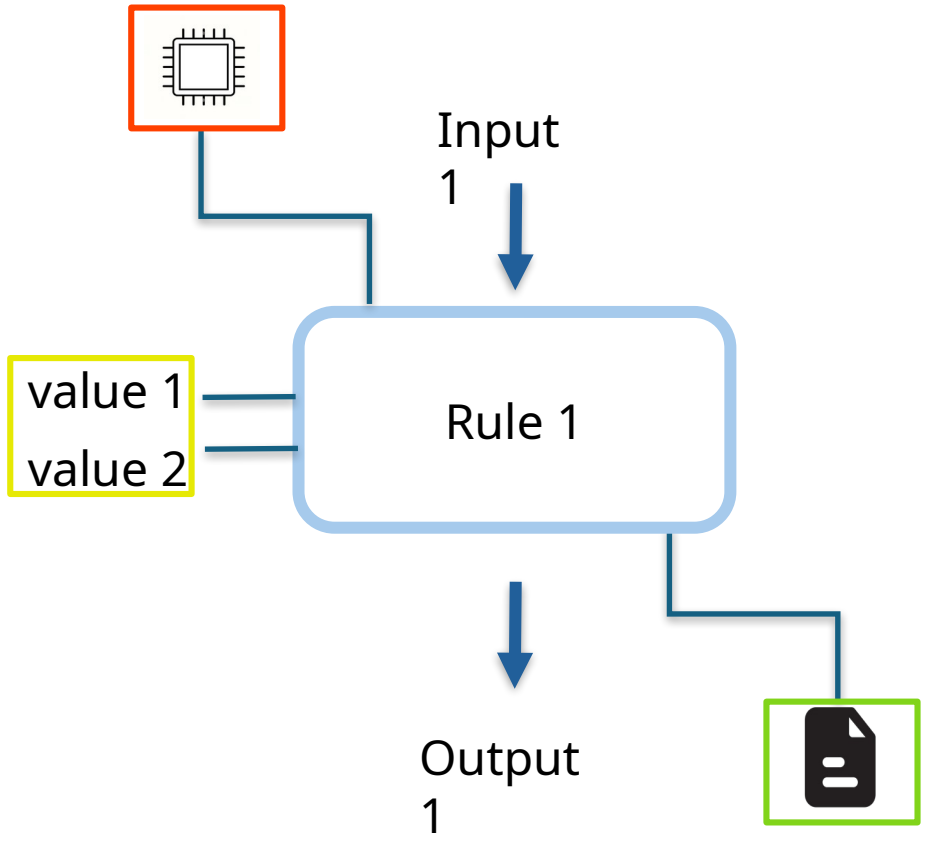


Launch pipeline : `snakemake --cores 1`

Snakemake : Pimp my rule

```
rule generic_rule:
    input:
        input_file = 'file_a.txt'
    output:
        output_file = 'file_a.out'
    threads: 2
    ressources:
        mem_mb=100
    params:
        param1 = 'value1'
        param2 = 'value2'
    conda : 'envs/myenv.yaml'
    message: 'This is a generic rule using {params.param1} and {params.param2}'
    log: 'logs/{input.inputfile}.log'

    shell:
        """
        So some shell command here
        I.e. command --arg1 {params.param1} --args2 {params.param2} --threads {threads} \
        {input.input_file} > {output.output_file}
        """
```



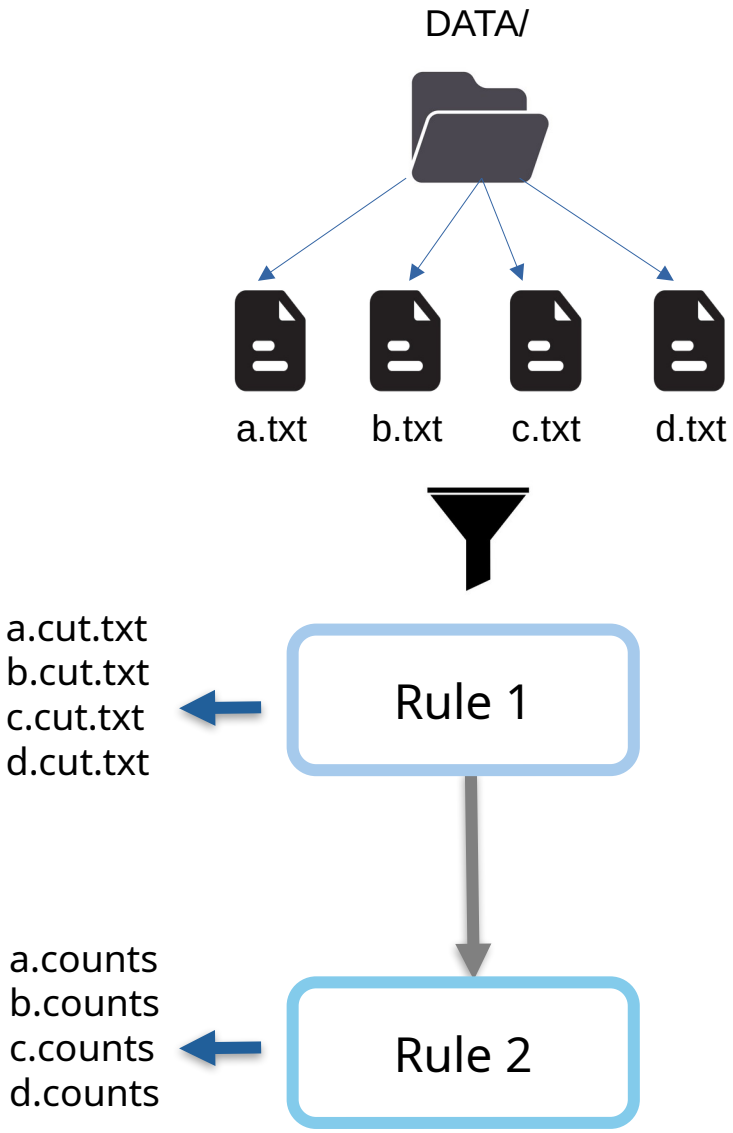
Snakemake : Make my workflow generic

```python
from os import join
DATA_FOLDER = "DATA/""
RESULTS_FOLDER = "RESULTS/"
SAMPLES = ["A", "B", "C", "D"]

rule rule_1:
    input:
        my_input = expand(join(DATA_FOLDER, "{sample}.txt"), sample = SAMPLES)
    output:
        my_output = expand(join(RESULTS_FOLDER, "{sample}.cut.txt"), sample = SAMPLES)
    shell:
        """
        cut -f 1 {input.my_input} > {outpout.my_output}
        """

rule rule_2:
    input:
        my_input_bis = expand(join(RESULTS_FOLDER, "{sample}.cut.txt"), sample = SAMPLES)
    output:
        my_last_output = expand(join(RESULTS_FOLDER, "{sample}.counts"), sample = SAMPLES)
    shell:
        """
        wc -l {input.my_input_bis} > {output.my_output_bis}
        """

rule all:
    input:
        expand(join(DATA_FOLDER, "{sample}.counts"), sample = SAMPLES)
```



Use wildcards to make rules adaptable to various file names