Swiss Institute of Bioinformatics

Nextflow in Action:

Build Smarter, Faster, Reproducible Pipelines

PhDc Jeferyd Yepes-García
University of Fribourg

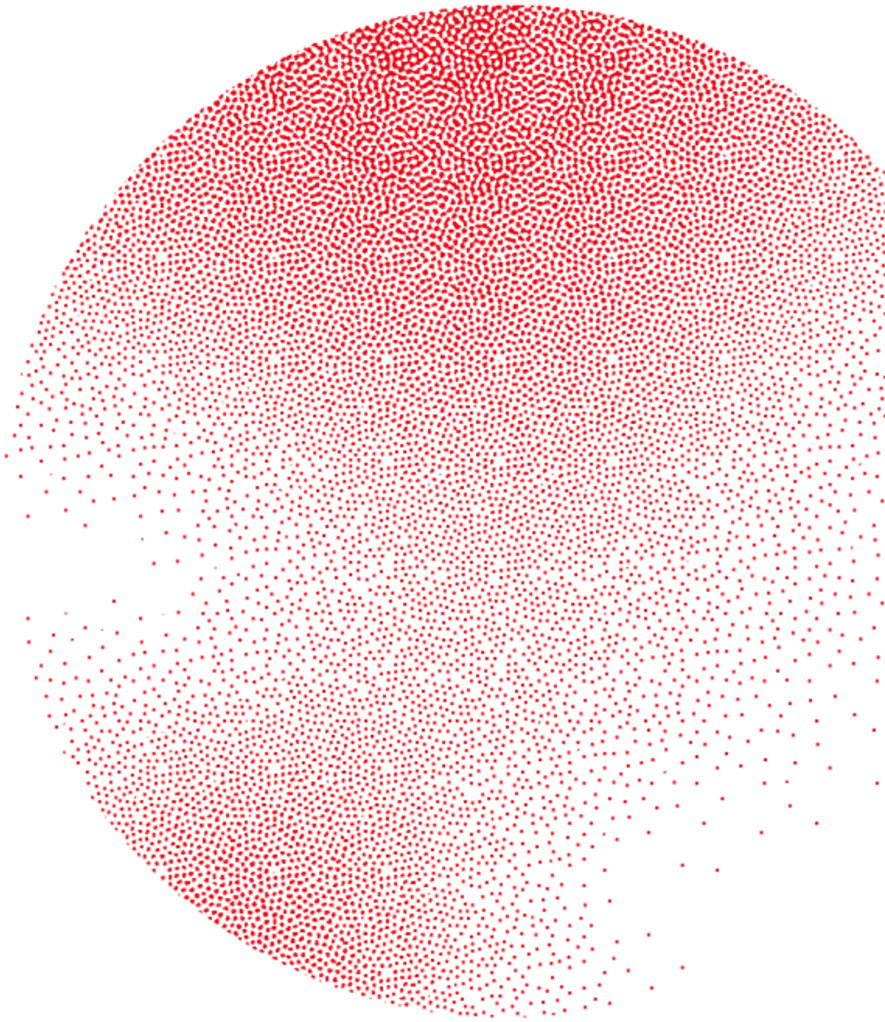
BUGFri

March 17th, 2026

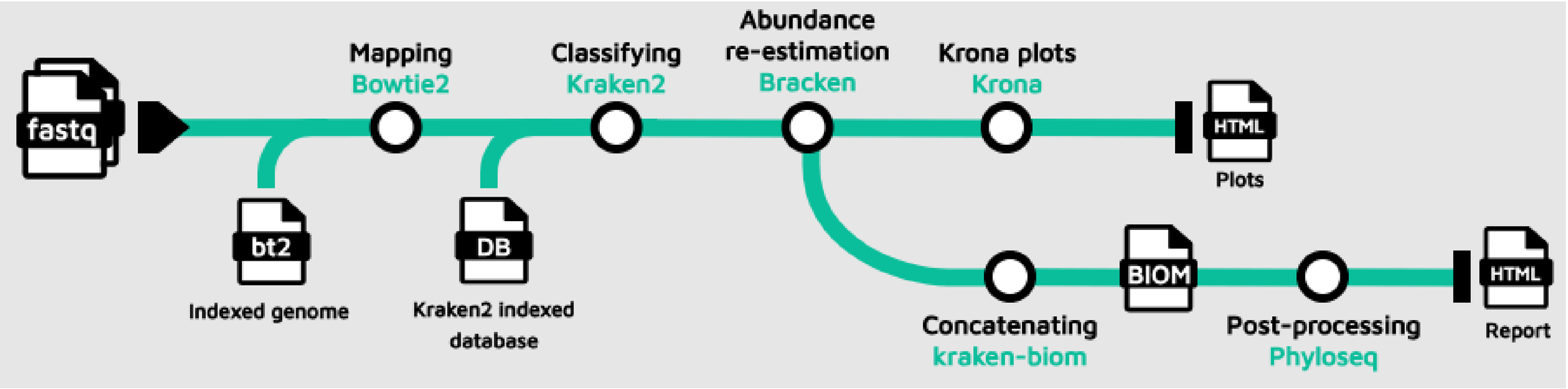
elixir SWITZERLAND

This presentation

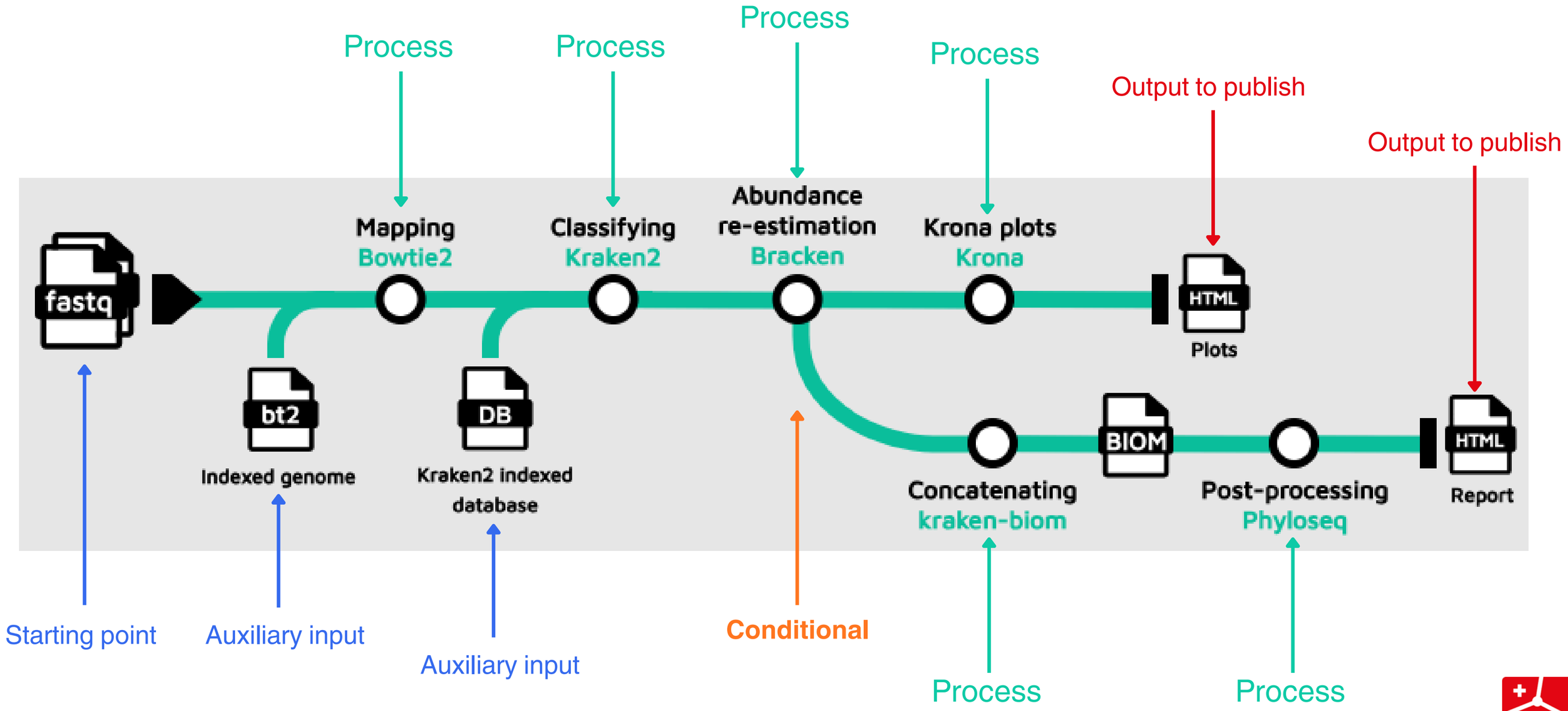
- Metagenomics pipeline
- Subworkflows
- Conditionals
- Scripts *à la carte*
- Report
- Resume pipelines



The pipeline



The pipeline



Subworkflows

- It is possible to define sub-routines or subworkflows to maintain the pipeline clean and structured.

workflow.nf

```
workflow TaxoFlow {
    // required inputs
    take:
        bowtie2_index
        kraken2_db
        reads_ch
    // workflow implementation
    main:
        BOWTIE2(reads_ch, bowtie2_index)
        KRAKEN2(BOWTIE2.out, kraken2_db)
        BRACKEN(KRAKEN2.out, kraken2_db)
        K_REPORT_TO_KRONA(BRACKEN.out)
        KT_IMPORT_TEXT(K_REPORT_TO_KRONA.out)
        if(params.sheet_csv){
            KRAKEN_BIOM(BRACKEN.out.collect())
            KNIT_PHYLOSEQ(KRAKEN_BIOM.out)
        }

    emit:
        bowtie_unali = BOWTIE2.out
        kraken_class = KRAKEN2.out
        bracken_class = BRACKEN.out
        k_report = K_REPORT_TO_KRONA.out
        biom = KRAKEN_BIOM.out
}
```

It is named.

Subworkflows can't publish, only emit.

main.nf

```
include {TaxoFlow} from './workflow.nf'

workflow {

    main:

    if(params.reads){
            reads_ch = channel.fromFilePairs(params.reads, checkIfExists:true)
        } else {
            reads_ch = channel.fromPath(params.sheet_csv)
                              .splitCsv(header:true)
                              .map {row -> tuple(row.sample_id, [file(row.fastq_1), file(row.fastq_2)])}
        }

    TaxoFlow(params.bowtie2_index, params.kraken2_db, reads_ch)

    // publish files
    publish:

    bowtie_unali = TaxoFlow.out.bowtie_unali
    kraken_class = TaxoFlow.out.kraken_class
    bracken_class = TaxoFlow.out.bracken_class
    k_report = TaxoFlow.out.k_report
    biom = TaxoFlow.out.biom

}
```

Import the subworkflow by its name.

Conditional execution

- Define conditions that will guide the execution.

config.nextflow

```
params {
    reads                                   = null
    outdir                                  = "${projectDir}/results"
    bowtie2_index                           = "${projectDir}/data/genome/TAIR10/TAIR10"
    kraken2_db                              = "${projectDir}/data/krakendb"
    sheet_csv                               = null
    report                                  = "${projectDir}/bin/report.Rmd"
}
```

Two options for input data to trigger the execution

main.nf

```
if(params.reads){
        reads_ch = channel.fromFilePairs(params.reads, checkIfExists:true)
    } else {
        reads_ch = channel.fromPath(params.sheet_csv)
                          .splitCsv(header:true)
                          .map {row -> tuple(row.sample_id, [file(row.fastq_1), file(row.fastq_2)])
    }
```

Conditional execution

- Define conditions that will guide the execution.

main.nf

```
main:
    BOWTIE2(reads_ch, bowtie2_index)
    KRAKEN2(BOWTIE2.out, kraken2_db)
    BRACKEN(KRAKEN2.out, kraken2_db)
    K_REPORT_TO_KRONA(BRACKEN.out)
    KT_IMPORT_TEXT(K_REPORT_TO_KRONA.out)
    if(params.sheet_csv){
        KRAKEN_BIOM(BRACKEN.out.collect())
        KNIT_PHYLOSEQ(KRAKEN_BIOM.out)
    }
```

Two processes controlled by the type of input

General form

```
if (condition) {
do this
} else {
do this
}
```

What if in the CSV there is only one sample?

Conditional execution

- Define conditions that will guide the execution.

main.nf

```
main:
    BOWTIE2(reads_ch, bowtie2_index)
    KRAKEN2(BOWTIE2.out, kraken2_db)
    BRACKEN(KRAKEN2.out, kraken2_db)
    K_REPORT_TO_KRONA(BRACKEN.out)
    KT_IMPORT_TEXT(K_REPORT_TO_KRONA.out)
    if(params.sheet_csv){
        KRAKEN_BIOM(BRACKEN.out.collect())
        KNIT_PHYLOSEQ(KRAKEN_BIOM.out)
    }
```

General form

```
if (condition) {
do this
} else {
do this
}
```

Two processes controlled by the type of input

What if in the CSV there is only one sample?

Conditional execution

- Deciding which container to pull.

General form

```
condition to evaluate
? Do this
: Else, do this
```

Groovy syntax

bigmag.nf

```
container "${workflow.containerEngine == 'singularity' && !task.ext.singularity_pull_docker_container
    ? 'https://depot.galaxyproject.org/singularity/pandas:1.4.3'
    : 'biocontainers/pandas:1.4.3'}"
```

If

Do this

Else, do this

Scripts *à la carte*

You can use your favourite scripting language (Perl, Python, R, etc), or even mix them in the same pipeline.

Any script

```
process perl_task {
    """
    #!/usr/bin/perl

    print 'Hi there!' . '\n';
    """
}

process python_task {
    """
    #!/usr/bin/python

    x = 'Hello'
    y = 'world!'
    print "%s - %s" % (x,y)
    """
}

workflow {
    perl_task()
    python_task()
}
```

Make sure of having the interpreter available or even better provide the correct environment.

Scripts *à la carte*

knit_phyloseq.nf

```
process KNIT_PHYLOSEQ {
    tag "knit_phyloseq"
    container "community.wave.seqera.io/library/bioconductor-phyloseq_knit_r-base_r-ggplot2_r-rmdformats:6efceb52eb05eb44"

    input:
    path merged

    output:
    stdout

    script:
    def report = params.report
    def outdir = params.outdir
    """
    biom_path=\$(realpath ${merged})
    outreport=\$(realpath ${outdir})
    Rscript -e "rmarkdown::render('${report}', params=list(args='\${biom_path}'),output_file='\${outreport}/report.html')"
    """
}
```

Specific container to execute the script

In-house script

Performance report

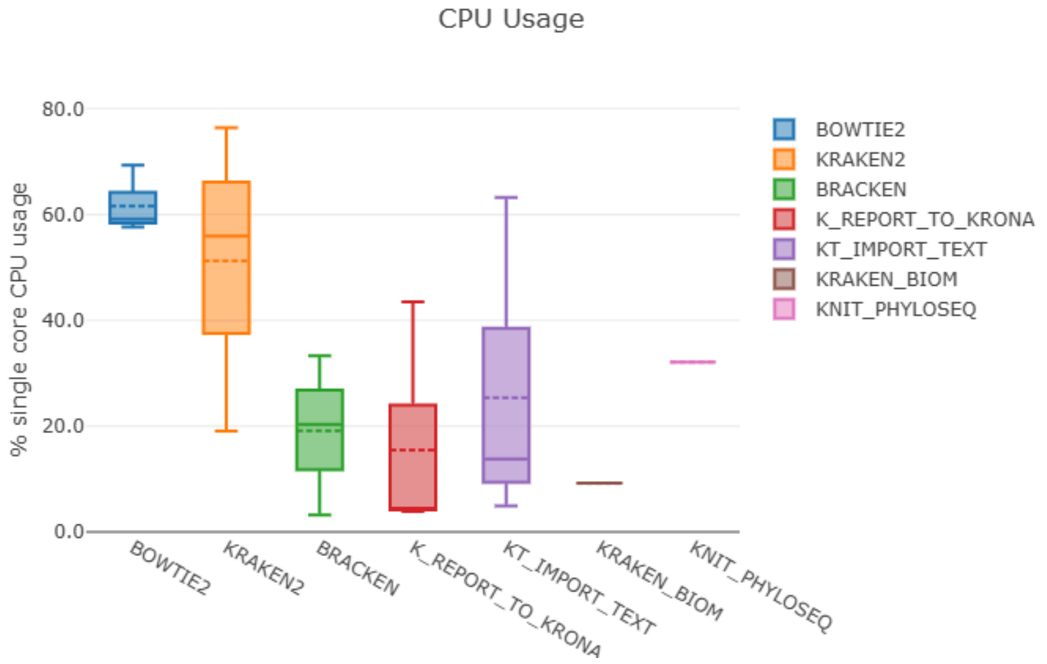
The execution report is an HTML report that includes metrics about a pipeline run. The report is organized into three sections: Summary, Resources, and Tasks.

```
nextflow run main.nf --sheet_csv data/samplesheet.csv -with-report report.html
```

CLI

nextflow.config

```
report {
    enabled = true
    file = '${projectDir}/report.html'
}
```



Demystifying Nextflow -resume

```
(base) ubuntu@ip-10-200-0-234:~/nextflow$ nextflow run calculateSimilarity.nf -profile ci -resume
N E X T F L O W  ~  version 20.10.0
Launching `calculateSimilarity.nf` [reverent_raman] - revision: 91805ee1f5
executor >  local (29)
[4b/735b7b] process > mmseqs2_filtering:getFastas (IGHV4-28) [100%] 4 of 4, cached: 4 ✔
[af/36b8d0] process > mmseqs2_filtering:search (IGHV1-45)    [100%] 2935 of 2935, cached: 2935 ✔
[7d/688cb0] process > align_h3:parasail (IGHV1-45)           [  8%] 105 of 1399, cached: 105
[e5/991e68] process > align_hcdr:parasail (IGHV1-45)         [  7%] 95 of 1387, cached: 95
```

Hash mechanism

Task executed previously

- Pipelines should not modify input files.
- Non-deterministic input channels (a process receiving from different channels).
- Process modifying global variables.

Demystifying Nextflow -resume

The mechanism works by assigning a unique ID to each task. This unique ID is used to create a separate execution directory, called the working directory (*work*), where the tasks are executed and the results stored. A task's unique ID is generated as a 128-bit hash number obtained from a composition of the task's:

→ Inputs values
→ Input files
→ Command line string
→ Container ID
→ Conda environment
→ Environment modules
→ Any executed scripts in the bin directory

It’s your time!