Swiss Institute of Bioinformatics

Nextflow in Action:

Build Smarter, Faster, Reproducible Pipelines

PhDc Jeferyd Yepes-García
University of Fribourg

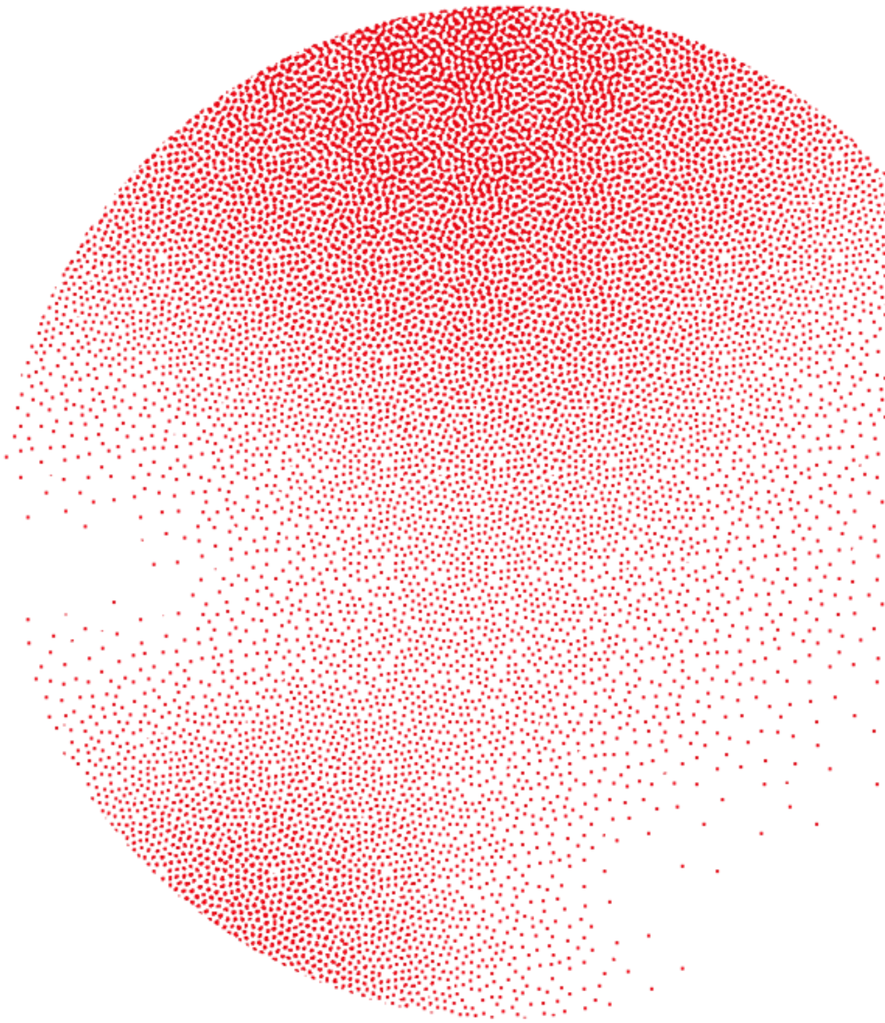
BUGFri

March 17th, 2026

elixir SWITZERLAND

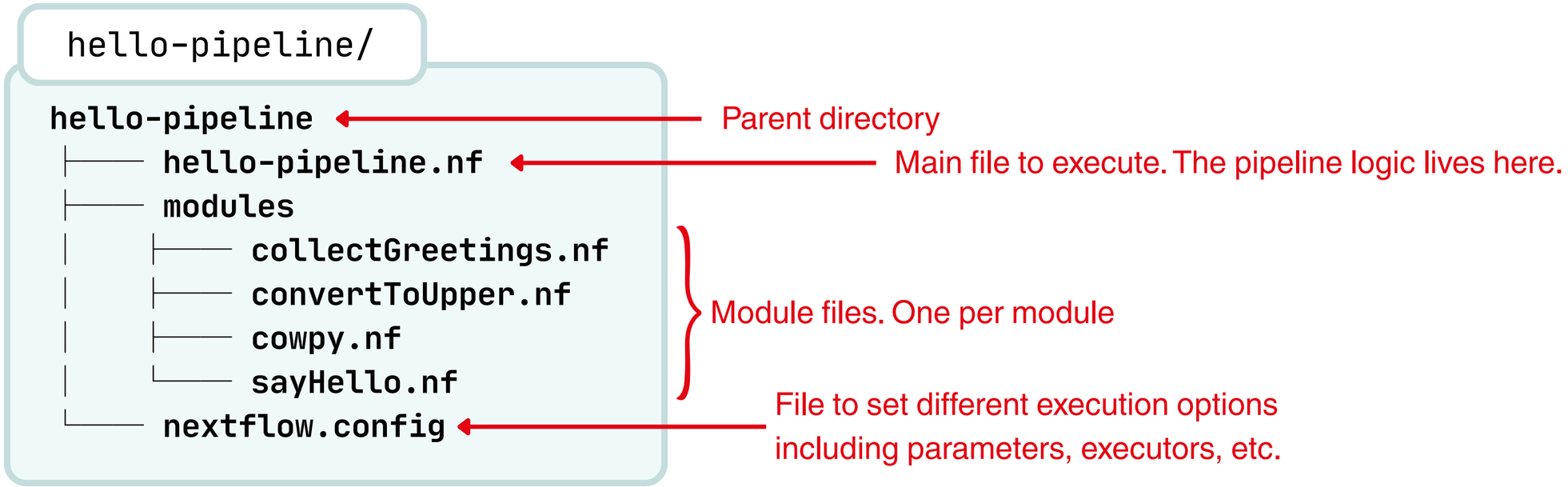
This presentation

- Hello-Nextflow pipeline
- Channel concept
- Channel factories
- Parameters
- Process and output directives
- Modules
- Variable handling

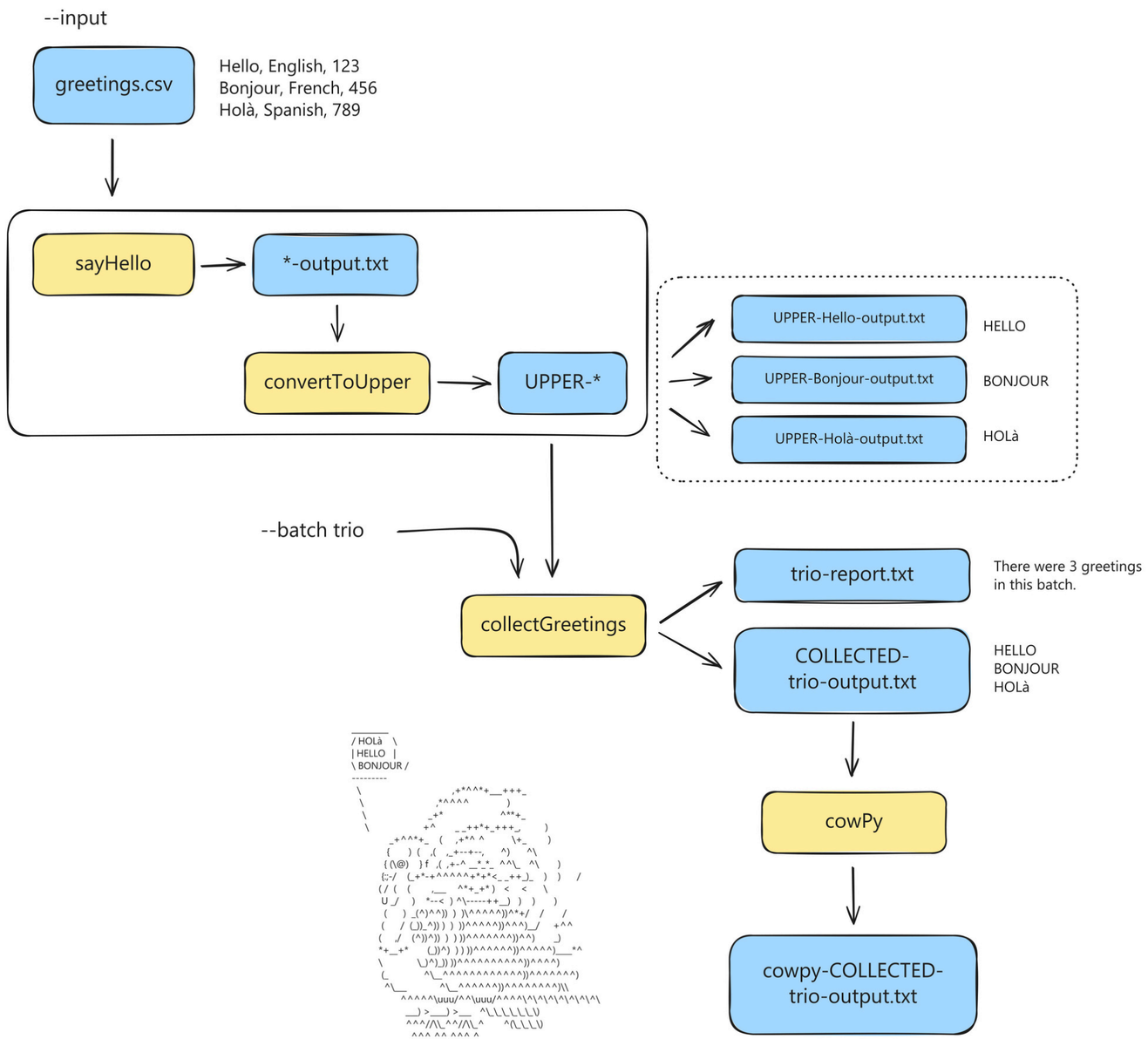


The pipeline

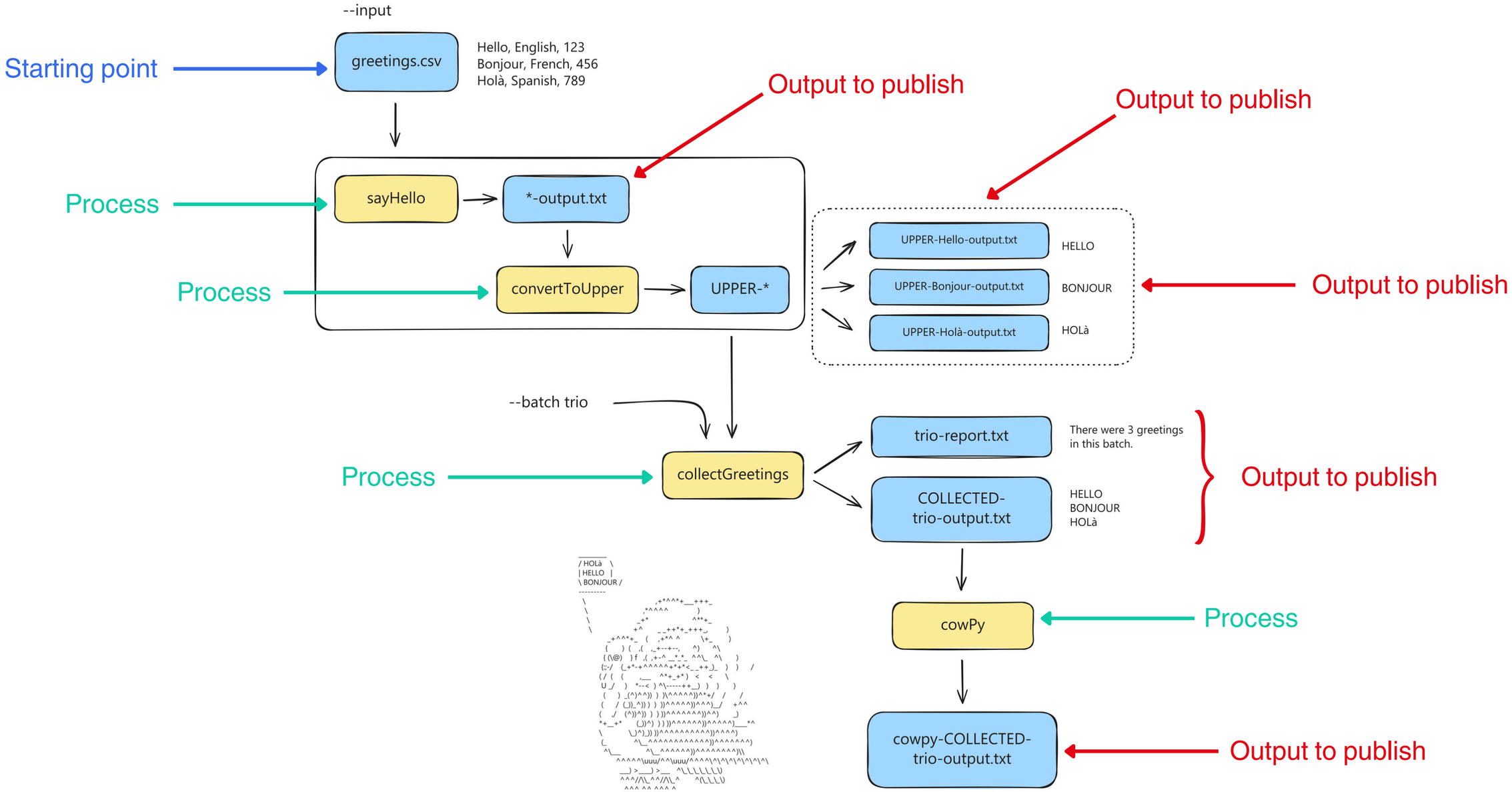
Typical pipeline folder structure:



The pipeline



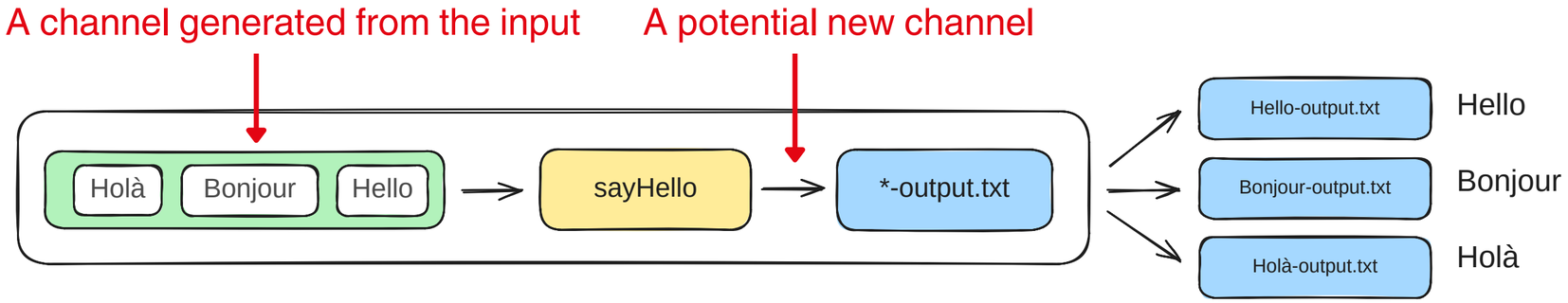
The pipeline



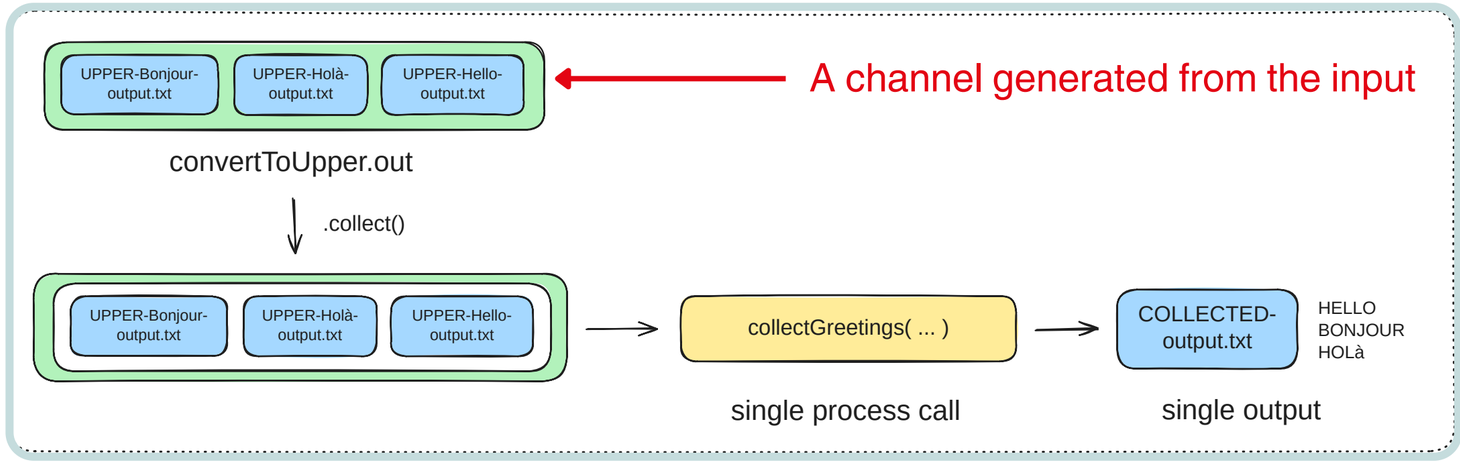
Inside a channel

A channel is a data stream that emits items **one by one** (asynchronous), where each item can be a value, file, or list.

1



2



Channel factories: The starting point

- channel.of:

```
ch = channel.of( 1, 3, 5, 7 )
ch.view { v -> "value: $v" }
```

- channel.fromPath

```
myFileChannel = channel.fromPath( '/data/some/bigfile.txt' )
```

- channel.fromFilePairs

```
channel
    .fromFilePairs('/my/data/SRR*_{1,2}.fastq')
    .view()
```

- channel.value

```
v1 = channel.value( 'Hello there' )
v2 = channel.value( [1,2,3,4,5] )
```

- channel.empty

Not a channel!

Working with files:

- file()

```
myFile = file('some/path/to/my_file.file')
```

Parameters

nextflow.config

```
/*
* Pipeline parameters
*/
params {
    input = 'data/greetings.csv'
    batch = 'batch'
    character = 'turkey'
}
```

Parameters can be used across **workflows** and **processes.**

hello-pipeline.nf

```
// emit a greeting
sayHello(greeting_ch)
// convert the greeting to uppercase
convertToUpper(sayHello.out)
// collect all the greetings into one file
collectGreetings(convertToUpper.out.collect(), params.batch)
// generate ASCII art of the greetings with cowpy
cowpy(collectGreetings.out.outfile, params.character)
```

main.nf

```
    // Primary input
    input: Path

    // Reference genome archive
    hisat2_index_zip: Path
}
```

Dynamic vs Static typing

Process and output directives

nextflow.config

```
/*
* Process settings
*/
process {
    memory = 1.GB
    withName: 'cowpy' {
        memory = 2.GB
        cpus = 2
    }
}
```

Resources to allocate to a specific process.

Resources (memory) to allocate per process **(global)**

Where to store the output

nextflow.config

```
/*
* Output settings
*/
outputDir = "results/${params.batch}"
workflow.output.mode = 'copy'
```

How to store the output

Modules: better organization for a plug-n-play ecosystem

Processes can’t be executed in the workflow without declaring them first.

sayHello.nf

```
process sayHello {

    input:
    val greeting

    output:
    path "${greeting}-output.txt"

    script:
    """
    echo '${greeting}' > '${greeting}-output.txt'
    """
}
```

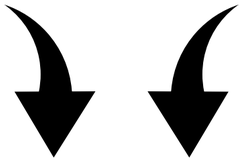
sayHello.nf

```
process sayHello {

    input:
    val greeting

    output:
    path "${greeting}-output.txt"

    script:
    """
    echo '${greeting}' > '${greeting}-output.txt'
    """
}
```



hello-pipeline.nf

```
// Include modules
include { sayHello          }          from './modules/sayHello.nf'
include { convertToUpper    }          from './modules/convertToUpper.nf'
include { collectGreetings  }          from './modules/collectGreetings.nf'
include { cowpy             }          from './modules/cowpy.nf'
```

Importing processes

Variable expansion and dynamic naming

File traceability:

collectGreetings.nf

```
process collectGreetings {

    input:
    path input_files
    val batch_name

    output:
    path "COLLECTED-${batch_name}-output.txt", emit: outfile
    path "${batch_name}-report.txt", emit: report

    script:
    count_greetings = input_files.size()
    """
    cat ${input_files} > 'COLLECTED-${batch_name}-output.txt'
    echo 'There were ${count_greetings} greetings in this batch.' > '${batch_name}-report.txt'
    """
}
```

Variable expansion and dynamic naming

File traceability:

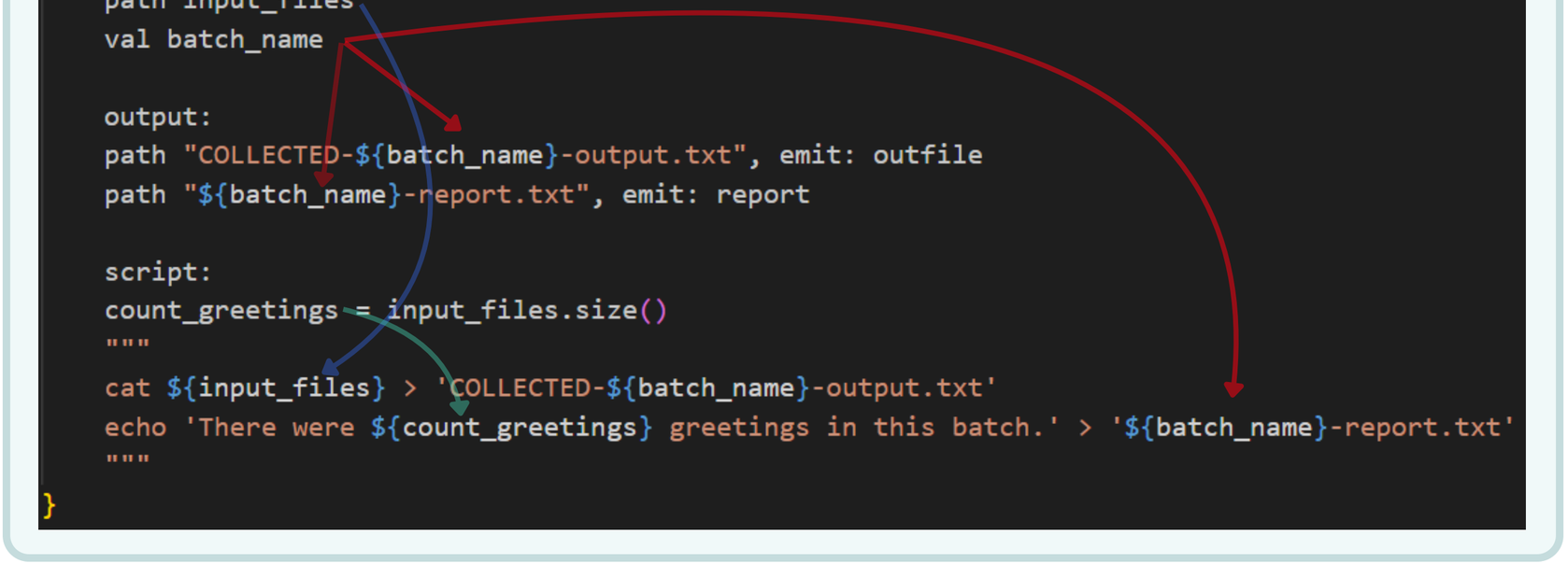
collectGreetings.nf

```
process collectGreetings {

    input:
    path input_files
    val batch_name

    output:
    path "COLLECTED-${batch_name}-output.txt", emit: outfile
    path "${batch_name}-report.txt", emit: report

    script:
    count_greetings = input_files.size()
    """
    cat ${input_files} > 'COLLECTED-${batch_name}-output.txt'
    echo 'There were ${count_greetings} greetings in this batch.' > '${batch_name}-report.txt'
    """
}
```



How many inputs are being received by this process?

It’s your time!