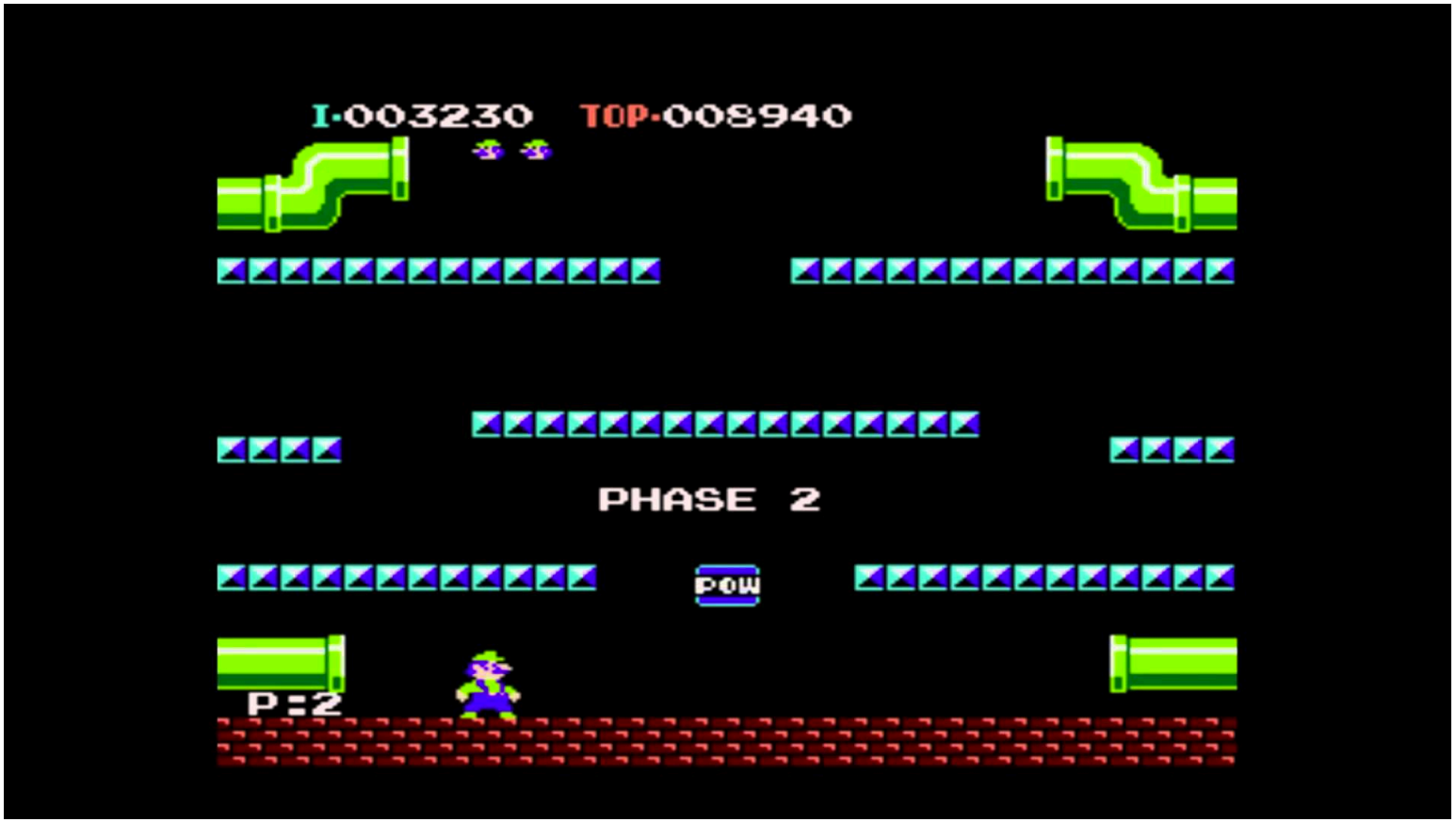
Pipelines and Workflows

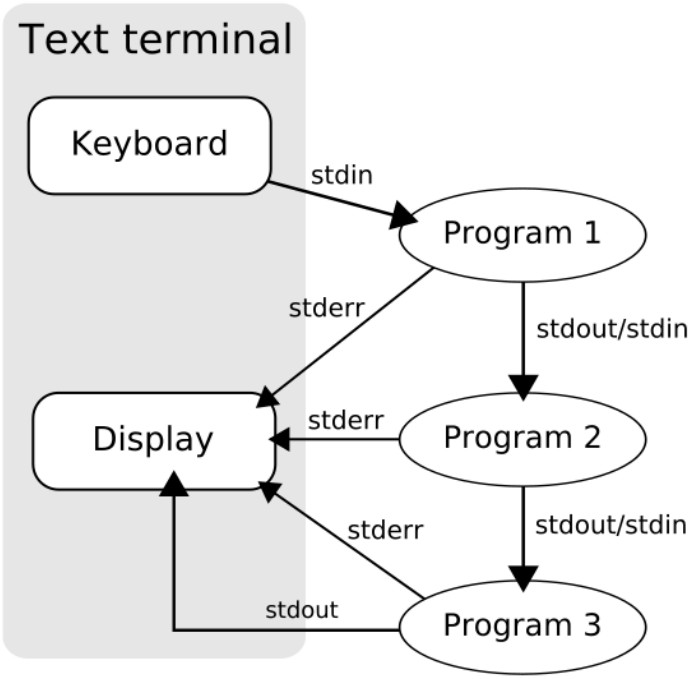


Luca Cozzuto

Bioinformatics Core Facility

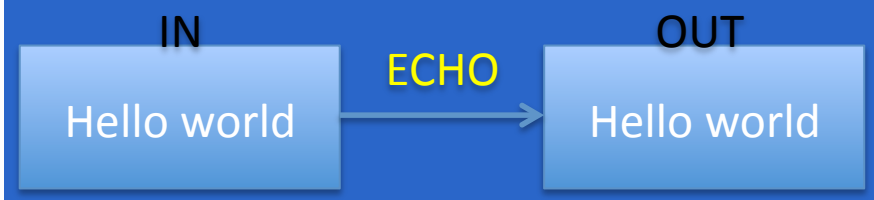
Pipelines and Workflows

What is a pipeline?



Pipelines and Workflows

```
$ echo "Hello World"
Hello World
```

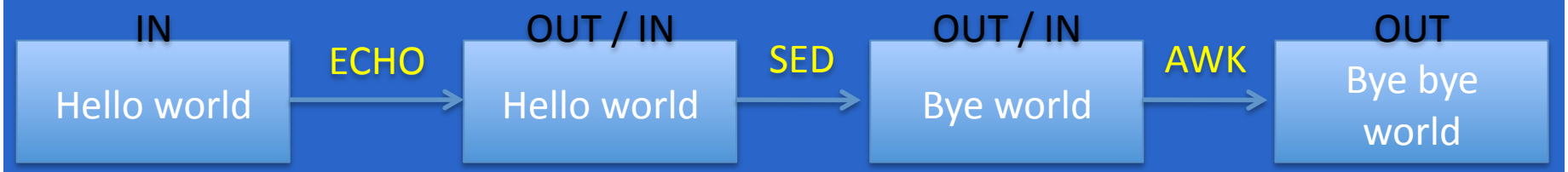


Pipelines and Workflows

```
$ echo "Hello World"
Hello World

$ echo "Hello World" | sed s/Hello/Bye/g
Bye World
```

Pipelines and Workflows

```
$ echo "Hello World"
Hello World

$ echo "Hello World" | sed s/Hello/Bye/g
Bye World

$ echo "Hello World" | sed s/Hello/Bye/g | awk '{print $1" "$1" "$2}'
Bye Bye World
```

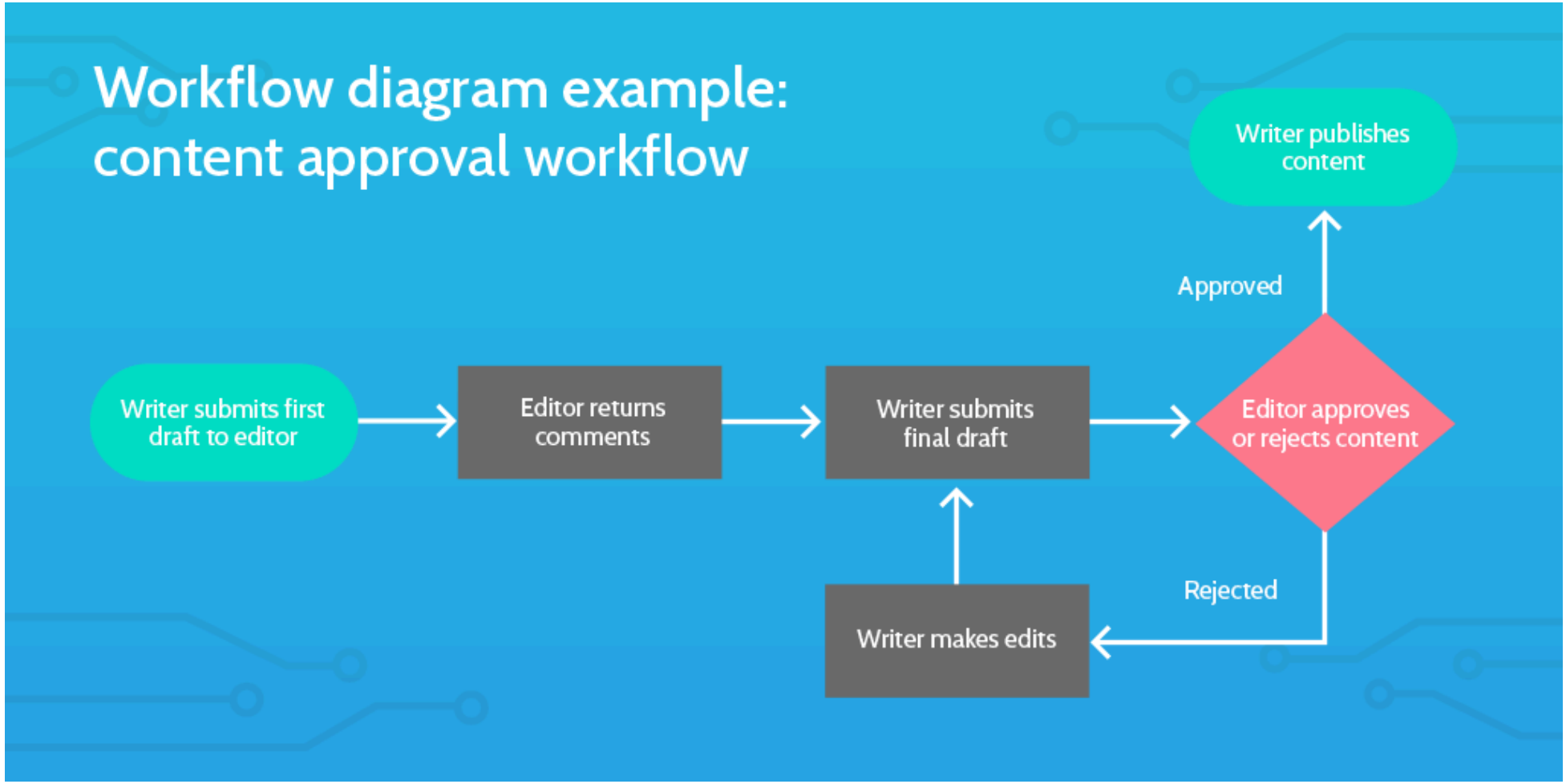


What is a workflow?

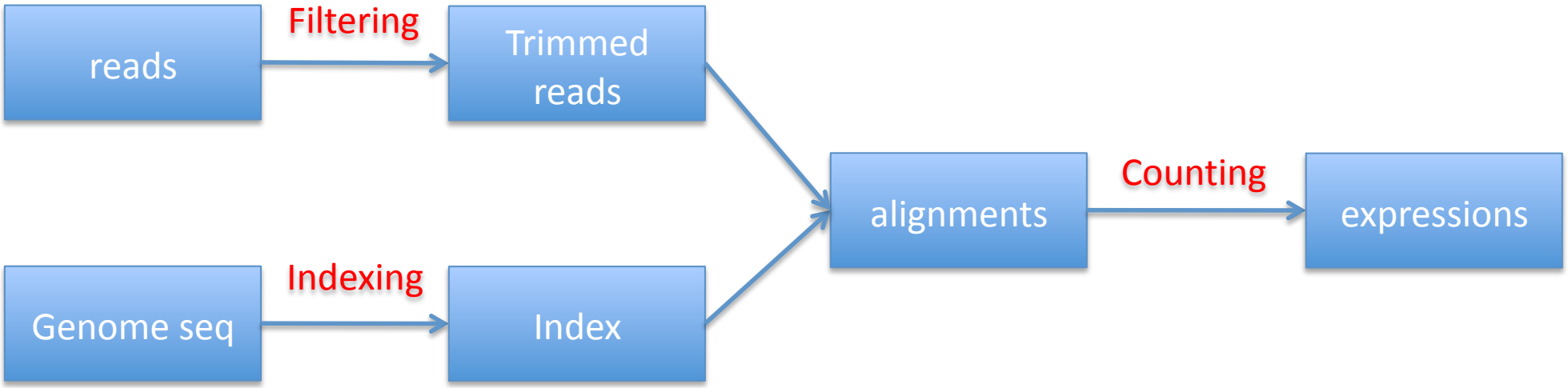
"A workflow consists of an orchestrated and repeatable pattern of business activity enabled by the systematic organization of resources into processes that transform materials, provide services, or process information It can be depicted as a sequence of operations, declared as work of a person or group, an organization of staff, or one or more simple or complex mechanisms"
Wikipedia

A workflow is a "series of operations" that has to be completed to achieve a goal.

Pipelines and Workflows



Bioinformatics workflow diagram



Workflow orchestration

”Orchestration is the automated arrangement, coordination, and management of computer systems, middleware, and services”
Wikipedia

What is NextFlow?

It is both a domain specific language and a workflow orchestration tool.

https://www.nextflow.io/

Introduction to NextFlow

What is Nextflow for?

It is for making pipelines without caring about parallelization, dependencies, file names, data structure, resuming processes etc.

Was it published?

nature biotechnology

Altmetric: 117 Citations: 7 More detail »

Correspondence

Nextflow enables reproducible computational workflows

Paolo Di Tommaso, Maria Chatzou, Evan W Floden, Pablo Prieto Barja, Emilio Palumbo & Cedric Notredame

Why using NextFlow:

- Fast prototyping
- Polyglot
- Highly scalable and portable
- Reproducible (native support of containers)
- Continuous checkpoints for resuming / expanding pipelines.

Introduction to NextFlow

Schedulers

Sun Grid Engine (SGE)

LSF

Platform Load Sharing Facility

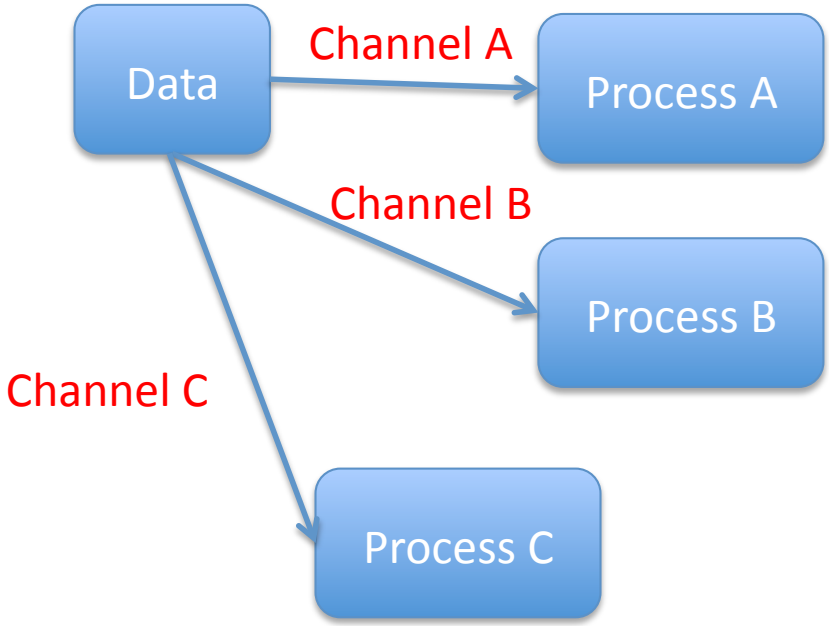
Portable Batch System

Cloud platforms

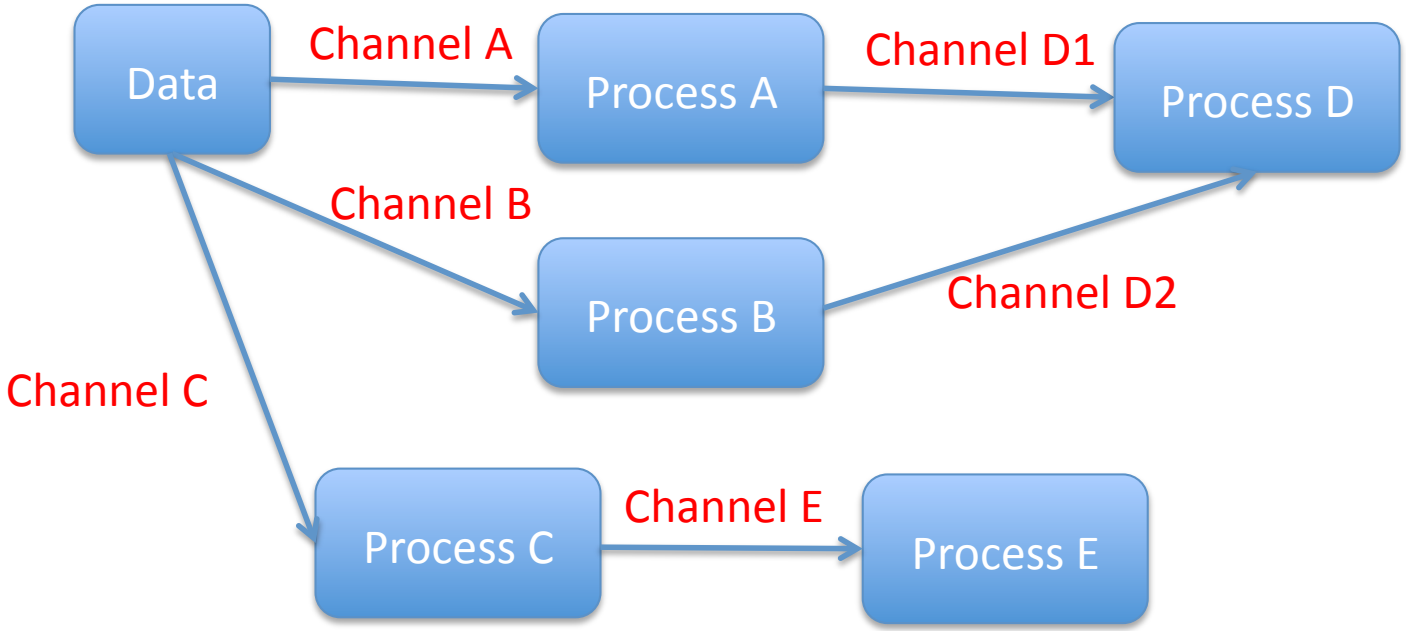
Who is using NextFlow?

NextFlow's pipeline

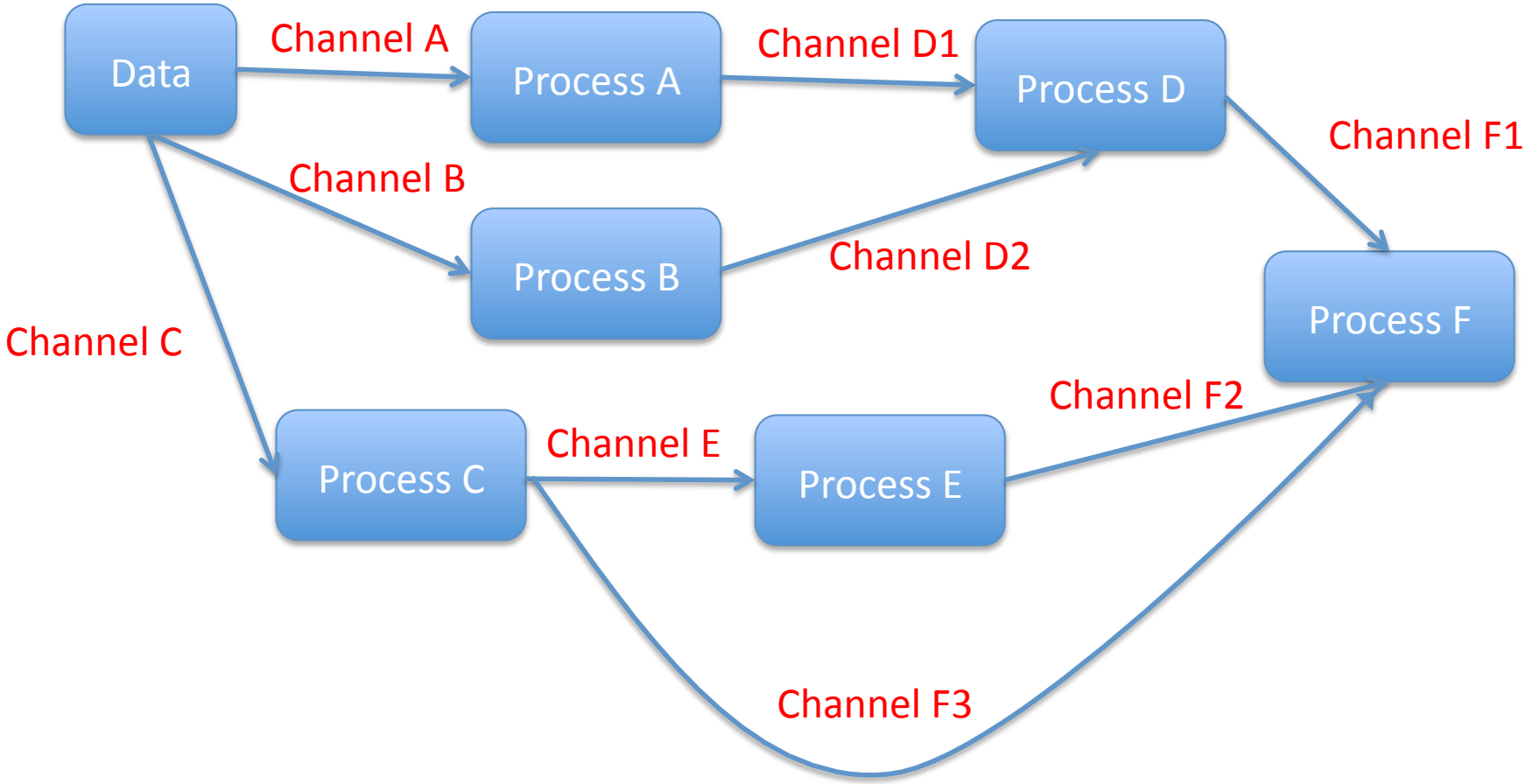
NextFlow's pipeline



NextFlow's pipeline



NextFlow's pipeline



Scripting

```
#!/usr/bin/env nextflow

greetings = Channel.from ("Bonjour", "Ciao", "Hello", "Hola")
```

Scripting

```
#!/usr/bin/env nextflow

greetings = Channel.from ("Bonjour", "Ciao", "Hello", "Hola")

process sayHello {

    input:
    val greeting from greetings


   script:
   """
       echo ${greeting}
   """
}
```

Scripting

```
#!/usr/bin/env nextflow

greetings = Channel.from ("Bonjour", "Ciao", "Hello", "Hola")

process sayHello {

  input:
  val greeting from greetings

  script:
  """
      echo ${greeting}
  """
}
```

$ nextflow run sayHello.nf

```
N E X T F L O W  ~  version 0.28.0
Launching `sayHello.nf` [stoic_stone] - revision: fd500d940d
[warm up] executor > local
[19/3d7ecb] Submitted process > sayHello (1)
[3c/41b1b8] Submitted process > sayHello (3)
[4f/963272] Submitted process > sayHello (2)
[37/b2f225] Submitted process > sayHello (4)
```

$ nextflow run sayHello.nf

```
N E X T F L O W  ~  version 0.28.0
Launching `sayHello.nf` [stoic_stone] - revision: fd500d940d
[warm up] executor > local
[19/3d7ecb] Submitted process > sayHello (1)
[3c/41b1b8] Submitted process > sayHello (3)
[4f/963272] Submitted process > sayHello (2)
[37/b2f225] Submitted process > sayHello (4)
```

Temporary files

For each process Nextflow create a folder structure (indicated in the log) where are stored:

- Links to input files (if any)
- Output files (if any)
- A number of hidden files with stderr, stdout, log etc.
- In particular the .command.sh contains the script executed

```
cat work/19/3d7ecb7c683c93b161c7254af36a72/.command.sh
#!/bin/bash -ue
echo Bonjour
```

Scripting

```
#!/usr/bin/env nextflow

greetings = Channel.from ("Bonjour", "Ciao", "Hello", "Hola")

process sayHello {
   publishDir 'results'

   input:
   val greeting from greetings

 output:
   file "${greeting}.txt" into greetingFiles

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

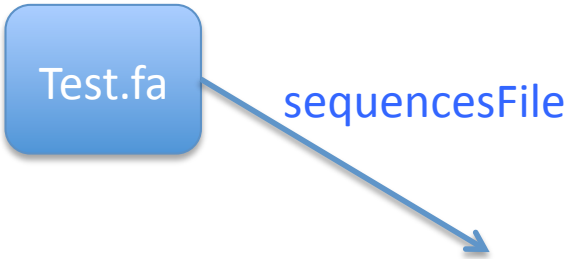
Scripting

```
#!/usr/bin/env nextflow

greetings = Channel.from ("Bonjour", "Ciao", "Hello", "Hola")

process sayHello {
    publishDir 'results'                    // directive

    input:
    val greeting from greetings

    output:
    file "${greeting}.txt" into greetingFiles

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

Publishing results

```
ls results/
Bonjour.txt   Ciao.txt   Hello.txt  Hola.txt

cat results/*
Bonjour
Ciao
Hello
Hola
```

More processes



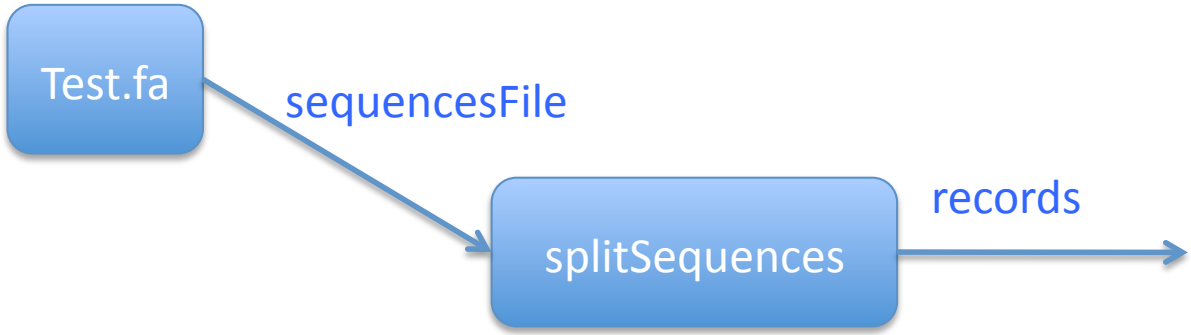
More processes

```
#!/usr/bin/env nextflow

params.inputFile = "$baseDir/../testdata/test.fa”

sequencesFile = file(params.inputFile)
```

Test.fa

```
>1
ATGTGACACGTCACGTACGTACCCA
>2
ATCTCACTCATGGTCAACATGTGGGAGAC
>3
ATTGTGGCACAACATTGCAAACC
```

More processes



More processes

```
#!/usr/bin/env nextflow

params.inputFile = "$baseDir/../testdata/test.fa”

sequencesFile = file(params.inputFile)

 /*
 * split a fasta file in multiple files
 */

process splitSequences {

   input:
   file 'input.fa' from sequencesFile

   output:
   file ('seq_*') into records

   """
   awk '/^>/{f="seq_"++d} {print > f}' < input.fa
   """
}
```

More processes

```
#!/usr/bin/env nextflow

params.inputFile = "$baseDir/../testdata/test.fa”

sequencesFile = file(params.inputFile)

 /*
 * split a fasta file in multiple files
 */

process splitSequences {

   input:
   file 'input.fa' from sequencesFile

   output:
   file ('seq_*') into records

   """
   awk '/^>/{f="seq_"++d} {print > f}' < input.fa
   """
}
```

Seq_1

```
>1
ATGTGACACGTCACGTACGTACCCA
```

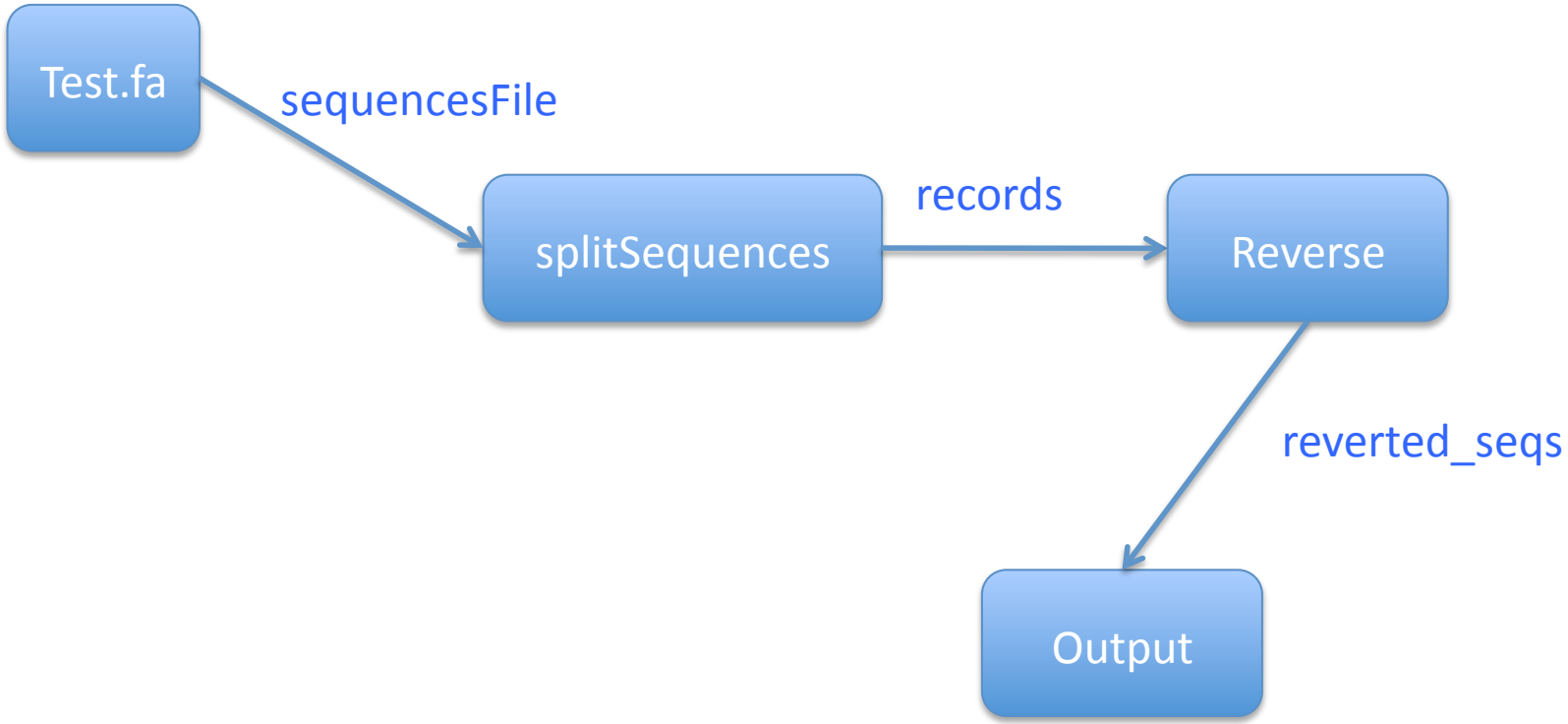
Seq_2

```
>2
ATCTCACTCATGGTCAACATGTGGGAGAC
```

Seq_3

```
>3
ATTGTGGCACAACATTGCAAACC
```

More processes



More processes

```
/*
 * Simple reverse the sequences
 */

process reverse {

    publishDir "output"

    input:
    file seq from records.flatten()

    output:
    file "*.rev" into reverted_seqs



    """
    cat $seq | awk '{if (\$1~">") {print \$0} else system("echo " \$0 " |rev")}' > $seq".rev"
    """
}
```

More processes

```
/*
 * Simple reverse the sequences
 */

process reverse {

    publishDir "output"

    input:
    file seq from records.flatten()

    output:
    file "*.rev" into reverted_seqs



    """
    cat $seq | awk '{if (\$1~">") {print \$0} else system("echo " \$0 " |rev")}' > $seq".rev"
    """
}
```

Seq_1

```
>1
ACCCATGCATGCACTGCACAGTGTA
```

Seq_2

```
>2
CAGAGGGTGTACAACTGGTACTCACTCTA
```

Seq_3

```
>3
CCAAACGTTACAACACGGTGTTA
```

NextFlow documentation

https://www.nextflow.io/docs/latest/index.html

Awesome pipelines

https://github.com/nextflow-io/awesome-nextflow