

# nextflow

Luca Cozzuto  
Bioinformatics Core Facility  
Centre for Genomics Regulation  
[luca.cozzuto@crg.eu](mailto:luca.cozzuto@crg.eu)

## What is NextFlow?

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



**nextflow**

<https://www.nextflow.io/>

## 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

Access provided by Universitat Pompeu Fabra




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.

# Introductory course to NextFlow

---

## Schedulers



---

## Cloud platforms



# Who is using NextFlow?



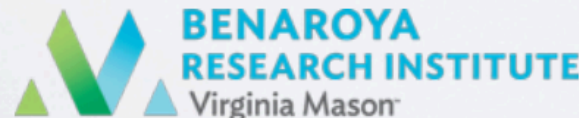
UiO: University of Oslo



Weill Cornell Medical College



Institut Pasteur

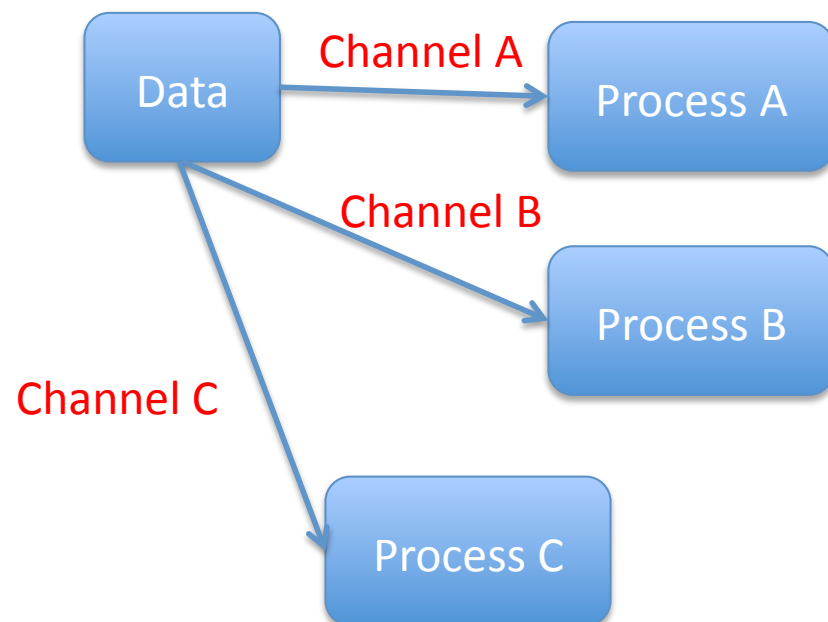


# NextFlow's pipeline

---

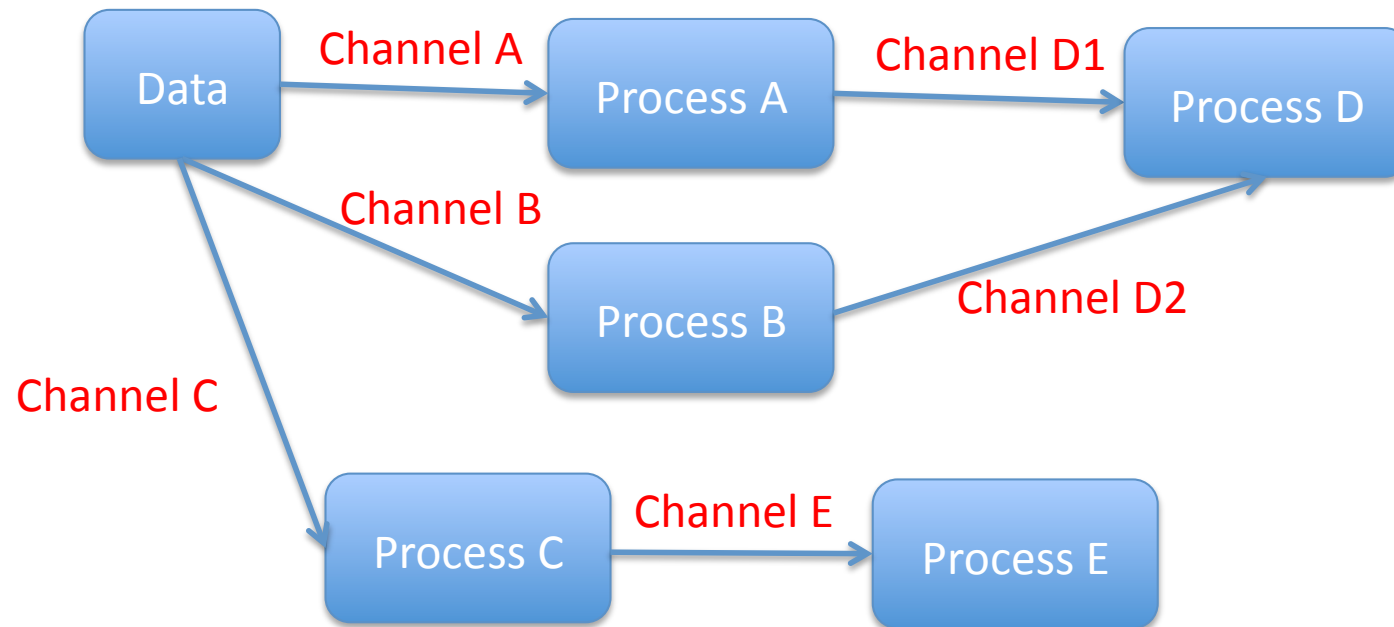
Data

# NextFlow's pipeline

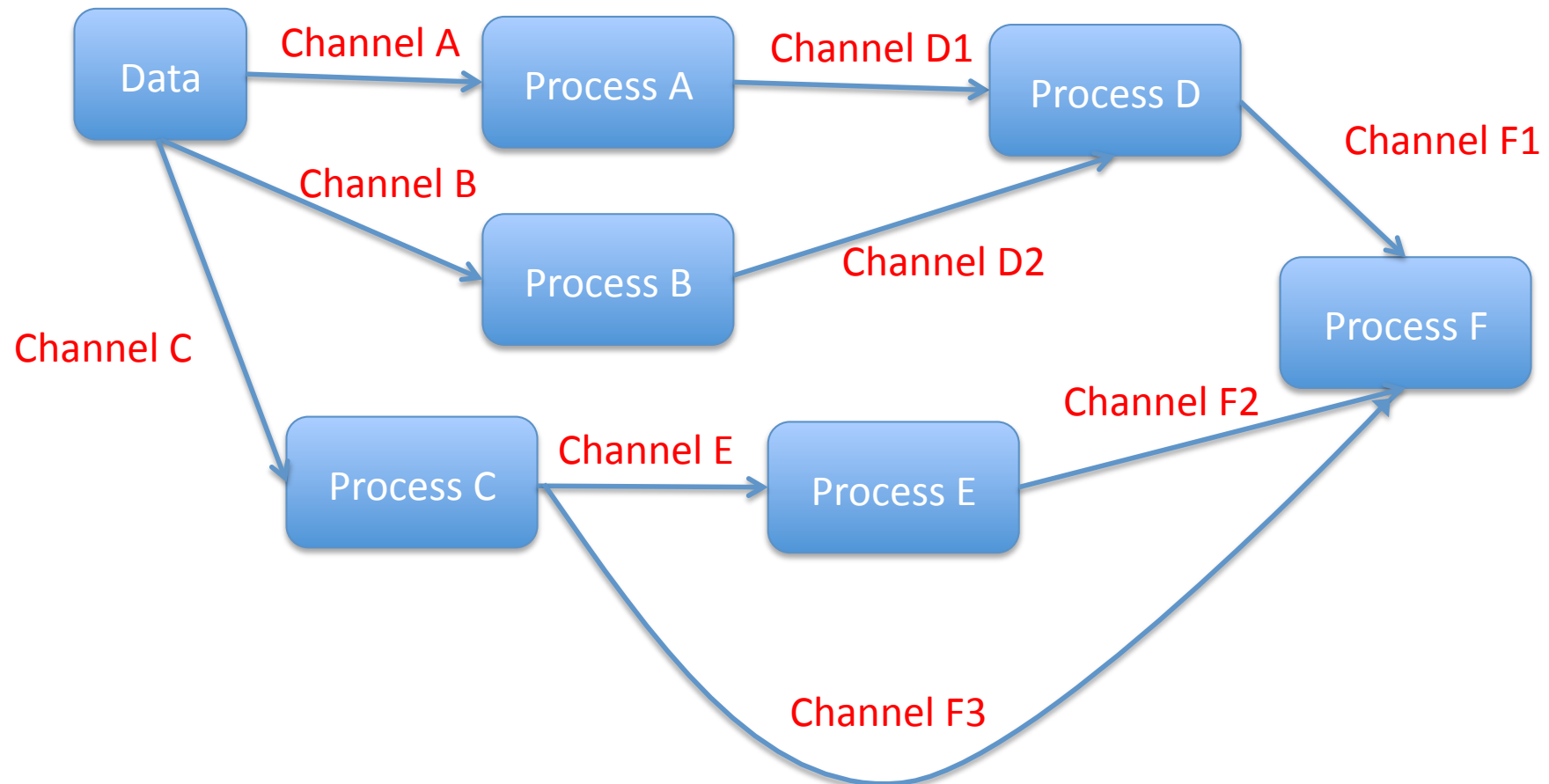




# NextFlow's pipeline



# NextFlow's pipeline



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

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

```
process sayHello {
```

```
    input:
```

```
    val greeting from greetings
```

```
    script:
```

```
    """
```

```
        echo ${greeting}
```

```
    """
```

```
}
```

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

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

```
process sayHello {  
  
    input:  
    val greeting from greetings  
  
    script:  
    """  
        echo ${greeting}  
    """  
}
```

```
#!/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'
```

```
// directive
```

```
    input:
```

```
    val greeting from greetings
```

```
    output:
```

```
    file "${greeting}.txt" into greetingFiles
```

```
    script:
```

```
    """
```

```
        echo $greeting > ${greeting}.txt
```

```
    """
```

```
}
```

ls results/

Bonjour.txt Ciao.txt Hello.txt Hola.txt

cat results/\*

Bonjour

Ciao

Hello

Hola

Let's do something practical!!!

Git clone

<https://github.com/biocorecrg/C4LWG-2018.git>

NextFlow documentation

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

Awesome pipelines

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