# INTRODUCTION TO NEXTFLOW

UE REPROHACKTHON

Frédéric Lemoine / Thomas Cokelaer
Institut Pasteur

2020/10/01

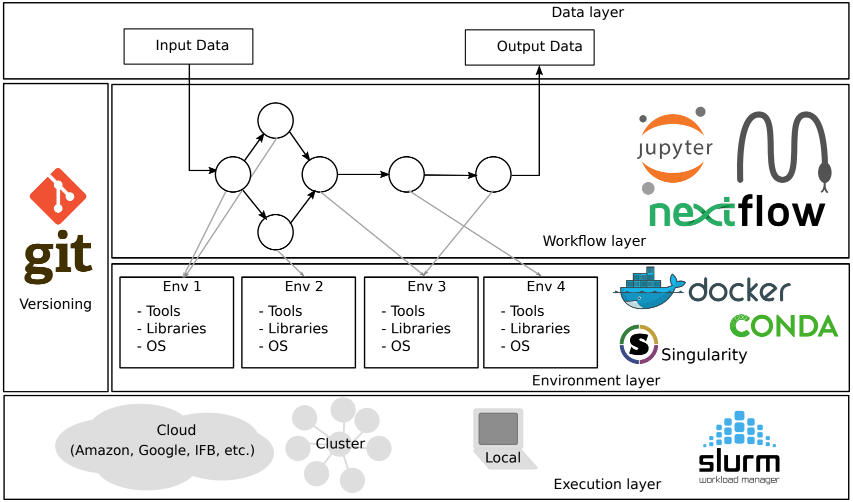
## PART 1

Introduction

Institut Pasteur

# 1.1 Analysis stack

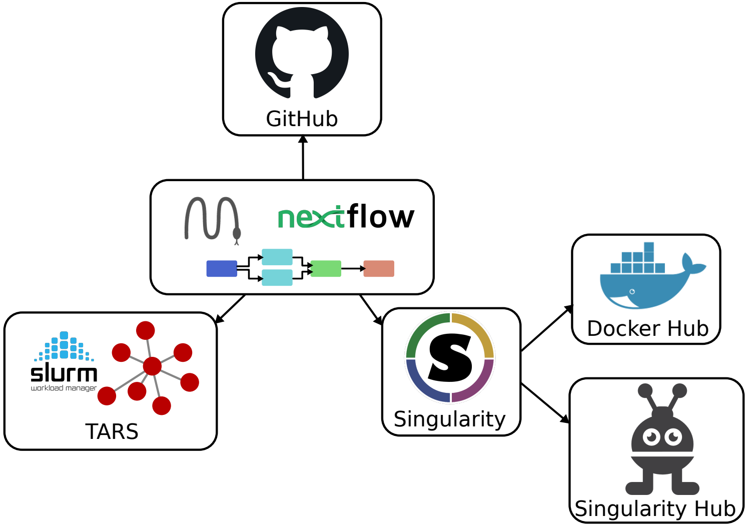
## Analysis stack: reminder



Institut Pasteur

# 1.2 Everything working together

## Analysis development: Workflow systems



Institut Pasteur

## 1.3 Workflow

### Definition

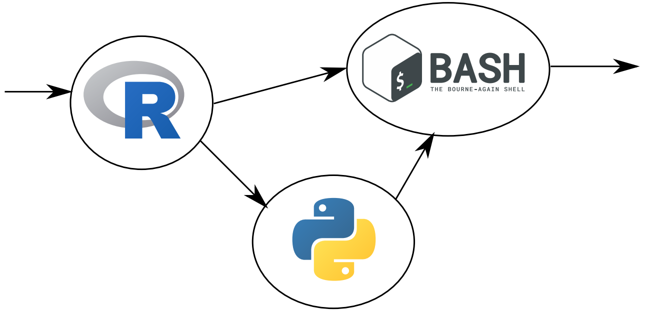
”Computational workflows describe the complex multi-step methods that are used for data collection, data preparation, analytics, predictive modelling, and simulation that lead to new data products”.

Goble, C., Cohen-Boulakia, S., Soiland-Reyes, S., Garijo, D., Gil, Y., Crusoe, M. R., ... & Schober, D. (2020). FAIR computational workflows. Data Intelligence, 2(1-2), 108-121.

## 1.4 Workflow management system

### Definition

”A bioinformatics workflow management system is a specialized form of workflow management system designed specifically to compose and execute a series of computational or data manipulation steps, or a workflow, that relate to bioinformatics.”
*Wikipedia*



Institut Pasteur

## 1.5 Workflow management system

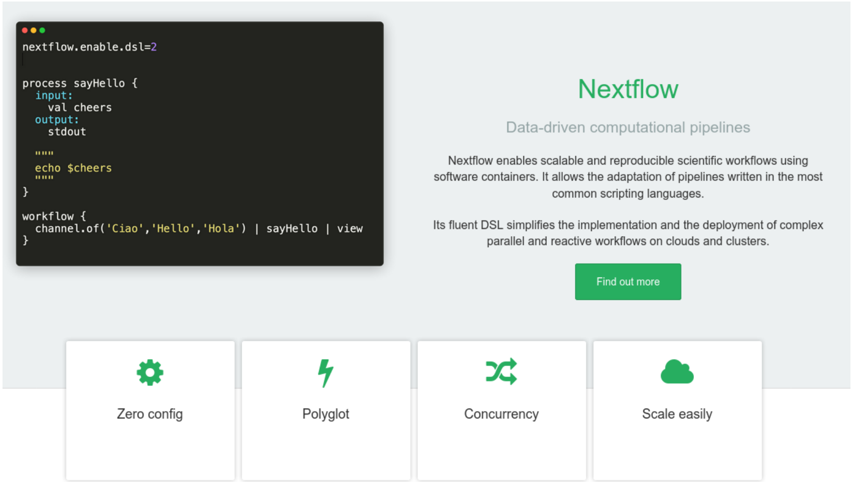
### Examples

- Nextflow (https://www.nextflow.io/)
- Snakemake
- Knime
- Galaxy
- Taverna

https://taverna.incubator.apache.org/

# 1.6 Nextflow

## About



```
nextflow.enable.dsl=2

process sayHello {
  input:
    val cheers
  output:
    stdout

  """
  echo $cheers
  """
}

workflow {
  channel.of('Ciao','Hello','Hola') | sayHello | view
}
```

Nextflow

Data-driven computational pipelines

Nextflow enables scalable and reproducible scientific workflows using software containers. It allows the adaptation of pipelines written in the most common scripting languages.

Its fluent DSL simplifies the implementation and the deployment of complex parallel and reactive workflows on clouds and clusters.

# 1.6 Nextflow

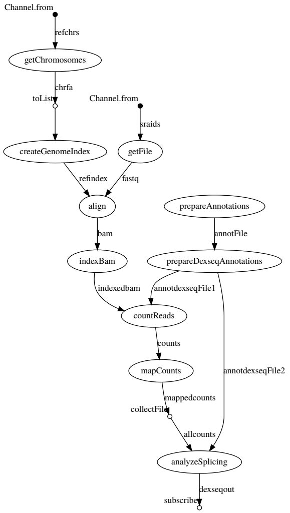
## Nextflow workflow

A nextflow workflow is made of:

- Processes (tasks)
- Channels (flux of data)
- Operators (channel manipulations)

It can :

- Execute ANY script (python, R, bash, binary, etc.)
- Manage environments (Docker/Singularity Containers)
- Execute on several execution environments (Clusters, local, cloud, etc.)



Institut Pasteur

## 1.6 Nextflow

### Processes

”In Nextflow, a process is the basic processing primitive to execute a user script.”

```
process < name > {

  [ directives ]

  input:
  < process inputs >

  output:
  < process outputs >

  when:
  < condition >

  [script|shell|exec]:
  < user script to be
      executed >
}
```

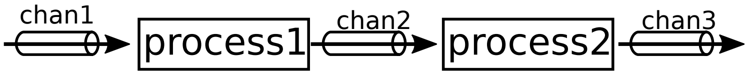
```
process reformatFasta {
  input:
  file sequences from fastaFiles

  output:
  file "*.phylip" into
      phylipFiles

  script:
  """
  goalign reformat phylip -i ${
      sequences} -o ${sequences.
      baseName}.phylip
  """
}
```

Institut Pasteur

## 1.6 Nextflow

### Channels

”Nextflow is based on the Dataflow programming model in which processes communicate through channels.”



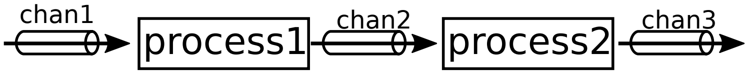
```
process process1 {
  input:
  file v from chan1

  output:
  file "*_out.txt" into chan2

  script:
  """
  command -i $v > ${v}_out.txt
  """
}
```

```
process process2 {
  input:
  file v from chan2

  output:
  file "*_out2.txt" into chan3

  script:
  """
  command2 -i $v > ${v}_out2.txt
  """
}
```

## 1.6 Nextflow

### Channels

”Nextflow is based on the Dataflow programming model in which processes communicate through channels.”

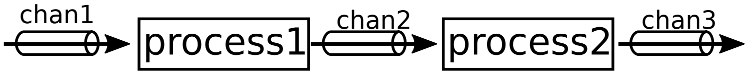


```
ch = Channel.of( 1, 3, 5, 7 )
process writeFiles {
  input:
  val odd from ch

  output:
  file "*.txt" into file

  script:
  """
  echo "Value = ${odd}" > file_${odd}.txt
  """
}
```

Institut Pasteur

## 1.6 Nextflow

### Channels



Example:

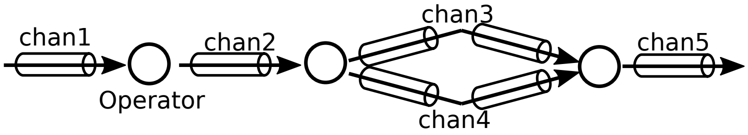
```
fastaFiles = Channel.fromPath("/home/user/fasta/*.fasta")
process reformatFasta {
  input:
  file sequences from fastaFiles

  output:
  file "*.phylip" into phylipFiles

  script:
  """
  goalign reformat phylip -i ${sequences} -o ${sequences.baseName}.phylip
  """
}
```

Institut Pasteur

## 1.6 Nextflow

### Operators

”Nextflow operators are methods that allow you to connect channels to each other or to transform values emitted by a channel applying some user provided rules.”

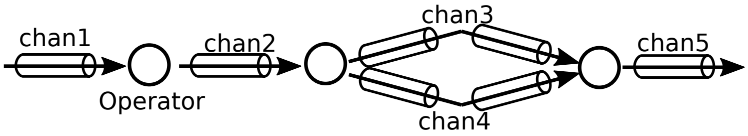


- Filtering operators
- Transforming operators
- Splitting operators
- Combining operators
- Forking operators
- Maths operators
- Other operators

## 1.6 Nextflow

### Operators

”Nextflow operators are methods that allow you to connect channels to each other or to transform values emitted by a channel applying some user provided rules.”

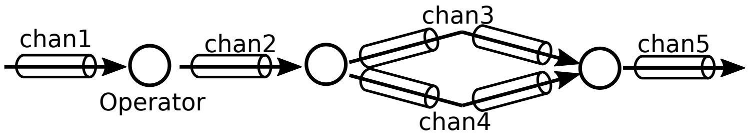


```
chan2 = chan1.randomSample( 10 )
chan2.into{chan3; chan4}
chan5 = chan3.combine(chan4, by: 0)
```

## 1.6 Nextflow

### Operators

”Nextflow operators are methods that allow you to connect channels to each other or to transform values emitted by a channel applying some user provided rules.”



Example:

```
fastaFiles = Channel.fromPath("/home/user/fasta/*.fasta")
randSample = fastaFiles.randomSample( 10 )
```

## 1.6 Nextflow

### Executors

"In the Nextflow framework architecture, the executor is the component that determines the system where a pipeline process is run and supervises its execution."

- Local: default
- SGE
- slurm
- PBS
- Kubernetes
- AWS Batch
- Google
- ...

Institut Pasteur

## 1.6 Nextflow

### Executors

”In the Nextflow framework architecture, the executor is the component that determines the system where a pipeline process is run and supervises its execution.”
Example:

```
fastaFiles = Channel.fromPath("/home/user/fasta/*.fasta")

process reformatFasta {
  executor="local"

  input:
  file sequences from fastaFiles

  output:
  file "*.phylip" into phylipFiles

  script:
  """
  goalign reformat phylip -i ${sequences} -o ${sequences.baseName}.phylip
  """
}
```

Institut Pasteur

# 1.6 Nextflow

## Environment: Containers

”Nextflow integration with Docker containers technology allows you to write self-contained and truly reproducible computational pipelines.”
Example:

```
fastaFiles = Channel.fromPath("/home/user/fasta/*.fasta")

process reformatFasta {
  executor="local"
  container="evolbioinfo/goalign:v0.3.2"

  input:
  file sequences from fastaFiles

  output:
  file "*.phylip" into phylipFiles

  script:
  """
  goalign reformat phylip -i ${sequences} -o ${sequences.baseName}.phylip
  """
}
```

Institut Pasteur

## 1.6 Nextflow

### Process settings

Example:

```
fastaFiles = Channel.fromPath("/home/user/fasta/*.fasta")

process reformatFasta {
  executor "local"
  container "evolbioinfo/goalign:v0.3.2"
  cpus 12
  memory "10G"
  time "30m"
  ...
}
```

Institut Pasteur

## 1.6 Nextflow

### Run nextflow

Nextflow script may be written in `main.nf` script, then:

```
nextflow run main.nf -with-docker --param1=10 --param2=file
```

Then in the script:

```
params.param1="default"
params.param2="default"
```

## 1.6 Nextflow

### Resume a nextflow run

Restart a analysis:

```
nextflow run main.nf -resume
```

It is useful if:

- The previous run stopped with error (hardware, code, etc.)
- The workflow has been modified
- Some data changed

In that case Nextflow resumes the analysis were it stopped, and reexecutes only the tasks that need to be restarted.

## 1.6 Nextflow

### Technical aspects of a Nextflow run

EACH Nextflow job is run in a dedicated folder inside `work` directory. Example:

- Task: AlignHmmProfile (705)
- Work dir: `work/1c/8b95199cc174e6590288faf07f3658`
- Dir content:
  - .command.err : Content of STDERR
  - .command.log : Log File (STDOUT+STDERR)
  - .command.out : Content of STDOUT
  - .command.run : Nextflow generated commands to run the script (cloud, etc.)
  - .command.sh : Script of the task (“script:“ in process definition)
  - .command.trace : Memory/CPU/Disk footprint of the task
  - .exitcode : Exit status of the task
  - Input files + Output files of the script

## Nextflow configuration

Instead of configuring processes in their implementation: Configuration file!

`nextflow.config`

```
executor {
  name = 'slurm'
  queueSize = 2000
}

report {
       enabled = true
       file = 'reports/report.html'
}

trace {
    enabled = true
    file = 'reports/trace.txt'
}

timeline {
    enabled = true
    file = 'reports/timeline.html'
}

dag {
    enabled = true
    file = 'reports/dag.dot'
}
```

```
singularity {
        enabled = true
        autoMounts = true
        runOptions = '--home $HOME:/home/
             $USER --bind /pasteur'
}
process {
    executor='slurm'
    queue = 'bioevo'
    clusterOptions = "--qos=bioevo"
    scratch=false
    maxRetries=30
    errorStrategy='ignore'

    withName: samtools {
        container="evolbioinfo/samtools:
             v1.10"
    }
    withName: nanoplot {
        container="evolbioinfo/nanoplot:
             v1.29.1"
       cpus=20
    }
    ...
```

Institut Pasteur

**Institut Pasteur**
25-28, rue du Docteur Roux
75724 Paris Cedex 15 - France
**www.pasteur.fr**