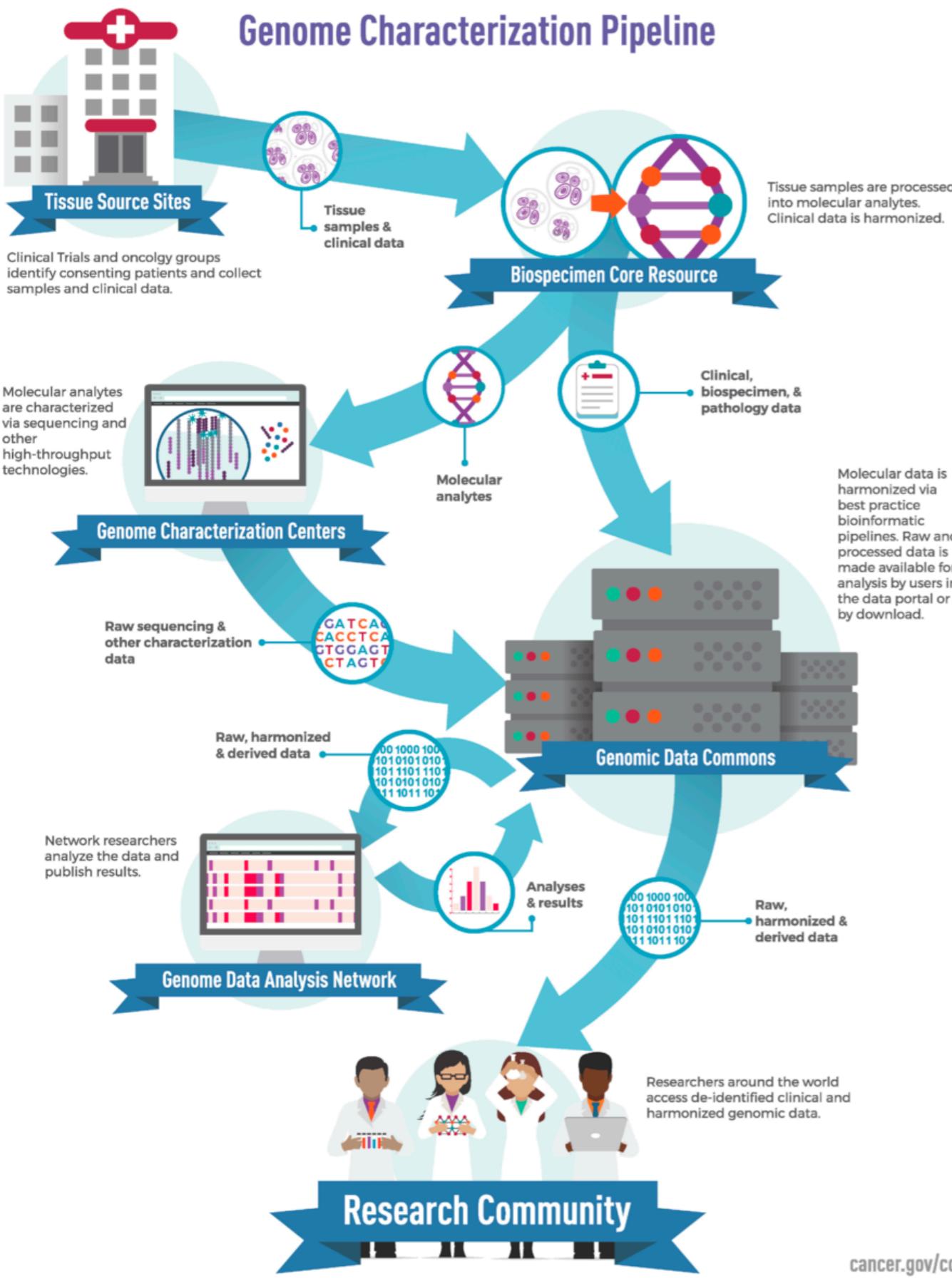


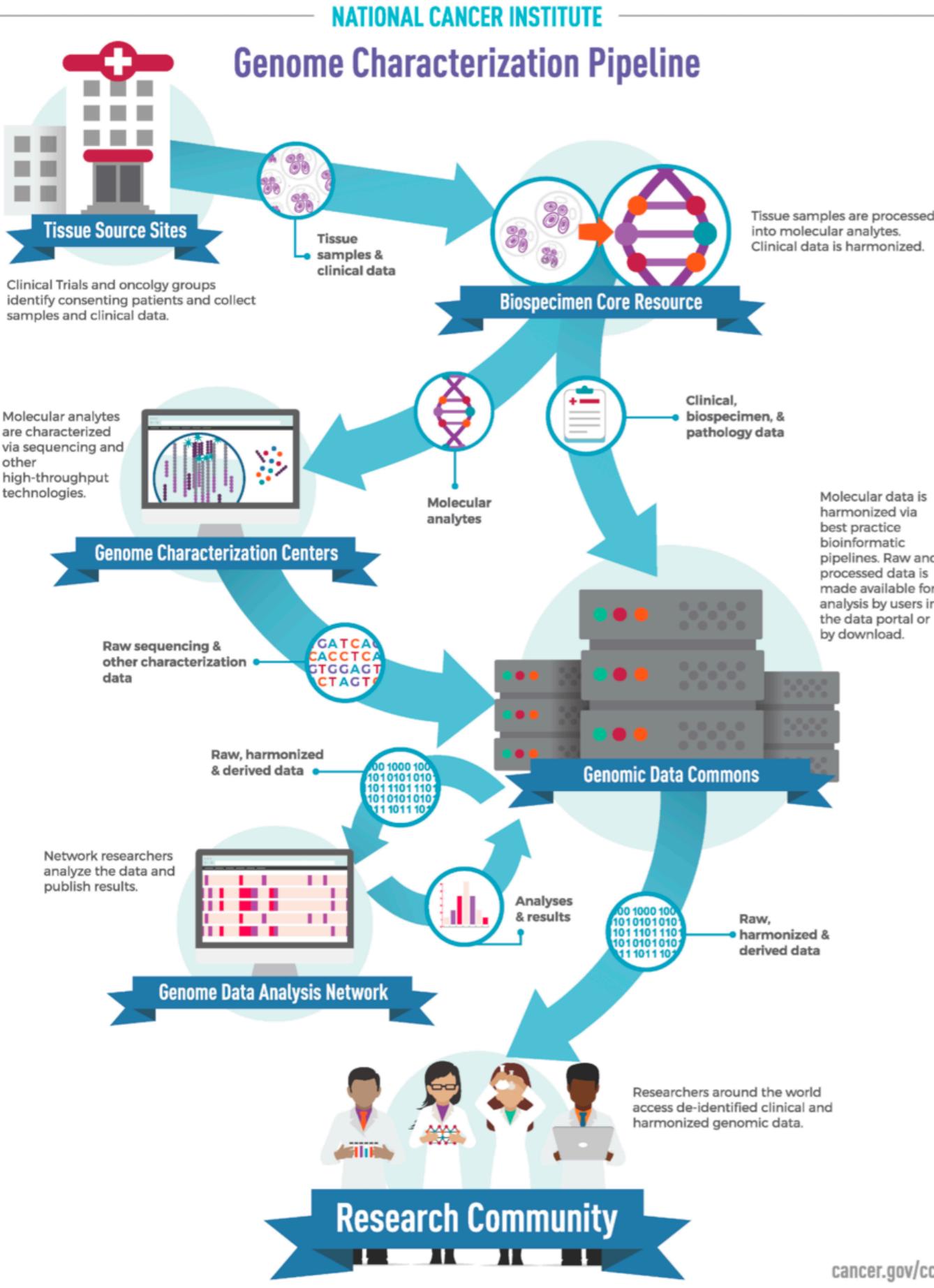
Nextflow: Distribuible escalable, y reproducible computación científica.

Alex Di Genova



Nextflow

Genómica



Nextflow

Genómica

Requisitos de pipelines para proyectos genómicos a gran escala

- **Reproducible:** los resultados genómicos deben ser totalmente reproducibles
- **Escalable:** Facil ejecución en cluster HPC o cloud.
- **Portátil:** puede ejecutarse en varias infraestructuras (diferente sistema operativo, cloud)
- **Manejar la heterogeneidad:** funciona con dependencias de software en conflicto y varios requisitos de recursos.

Pipelines para datos genómicos

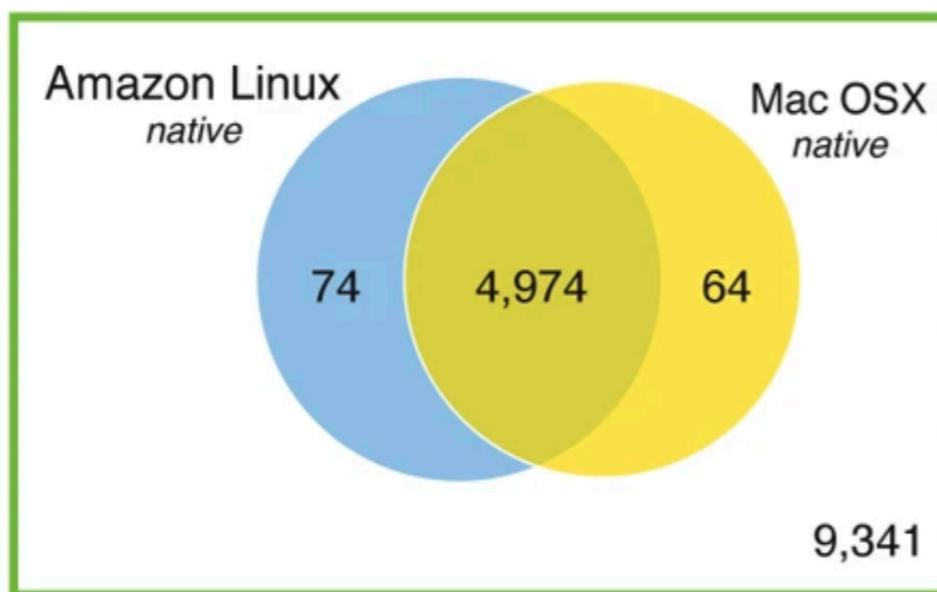
Solución utilizar un lenguaje de dominio específico (DSL)

Un lenguaje específico de dominio (DSL) es un lenguaje de programación especializado en un dominio de aplicación particular (pipelines).

- **Nextflow:** Basado en Groovy (java); Desarrollado en el Centro de regulacion del genoma (Barcelona, España).

C

Transcript quantification and differential expression with Kallisto and Sleuth



nature biotechnology

Explore content ▾ About the journal ▾ Publish with us ▾

[nature](#) > [nature biotechnology](#) > [correspondence](#) > [article](#)

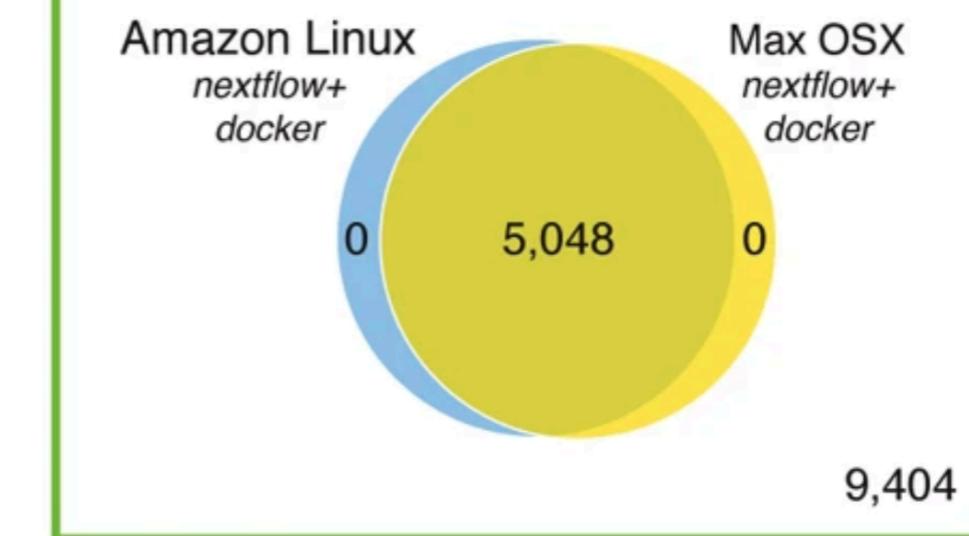
[Published: 11 April 2017](#)

Nextflow enables reproducible computational workflows

[Paolo Di Tommaso](#), [Maria Chatzou](#), [Evan W Floden](#), [Pablo Prieto Barja](#), [Emilio Palumbo](#) & [Cedric Notredame](#)✉

[Nature Biotechnology](#) 35, 316–319 (2017) | [Cite this article](#)

14k Accesses | 590 Citations | 122 Altmetric | [Metrics](#)



Nextflow

Diseño y características

1. Write code in any language

```
trim_galore --paired --fastqc --gzip \  
--basename sample1 -j 2 pair_1.fq pair_2.fq
```

2. Define succession of processes using dataflow programming

Trimming | Mapping | Quantification

3. Define software dependencies

Conda, Docker, Singularity

3. Define configuration

Executor (local, HPC scheduler),
error management, etc

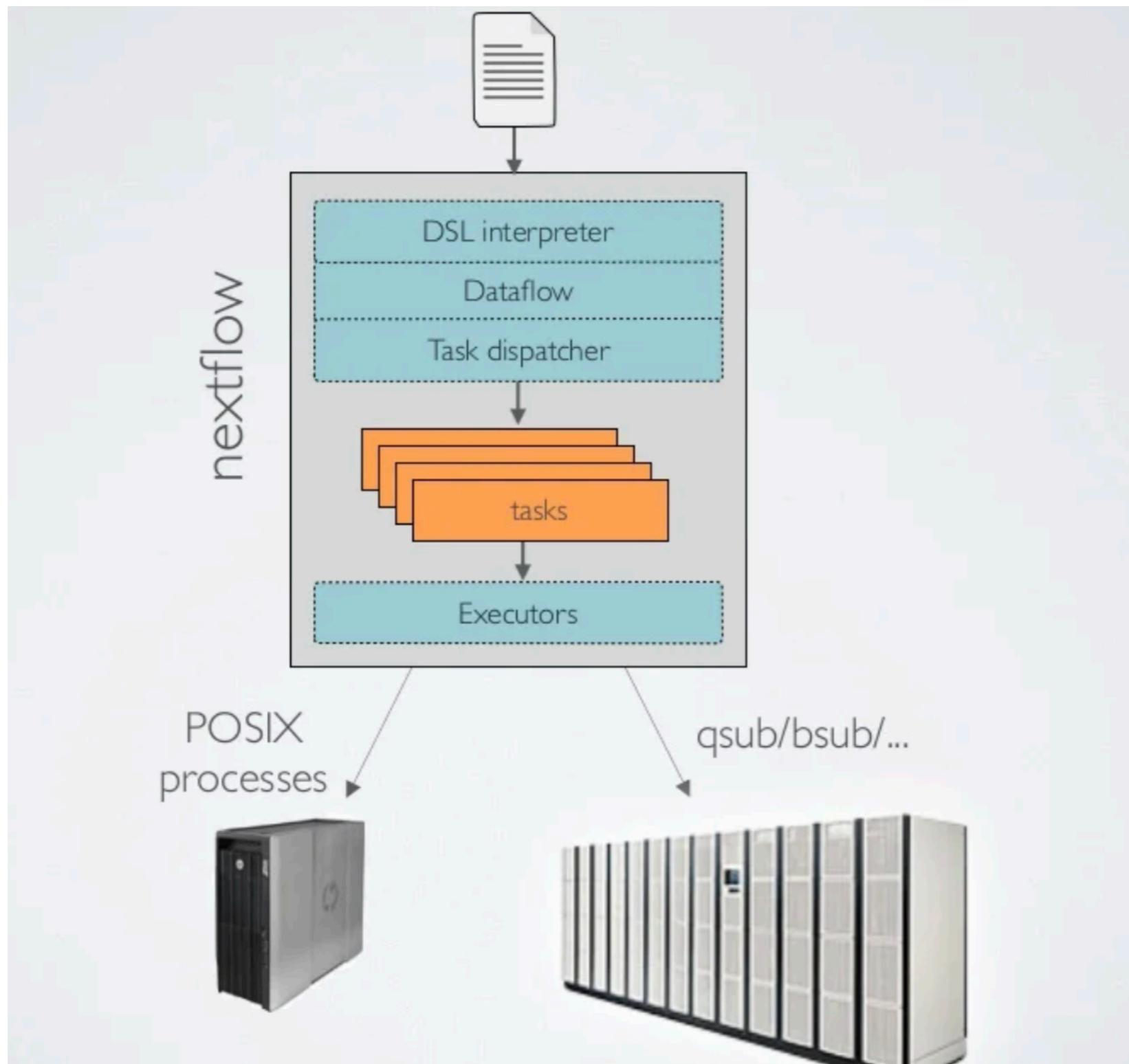
4. Run

Lenguaje de pipeline con paralelización implícita y programación de tareas:

- El mismo conjunto de tareas se aplica a todos los archivos de entrada
- Las tareas se ejecutan automáticamente en el orden correcto dadas sus entradas y salidas
- Maneja automáticamente el envío de tareas a distintos sistemas administradores de recursos.

Nextflow

Independiente de la plataforma

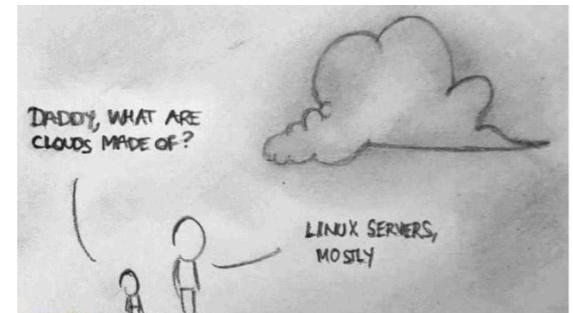


Nextflow

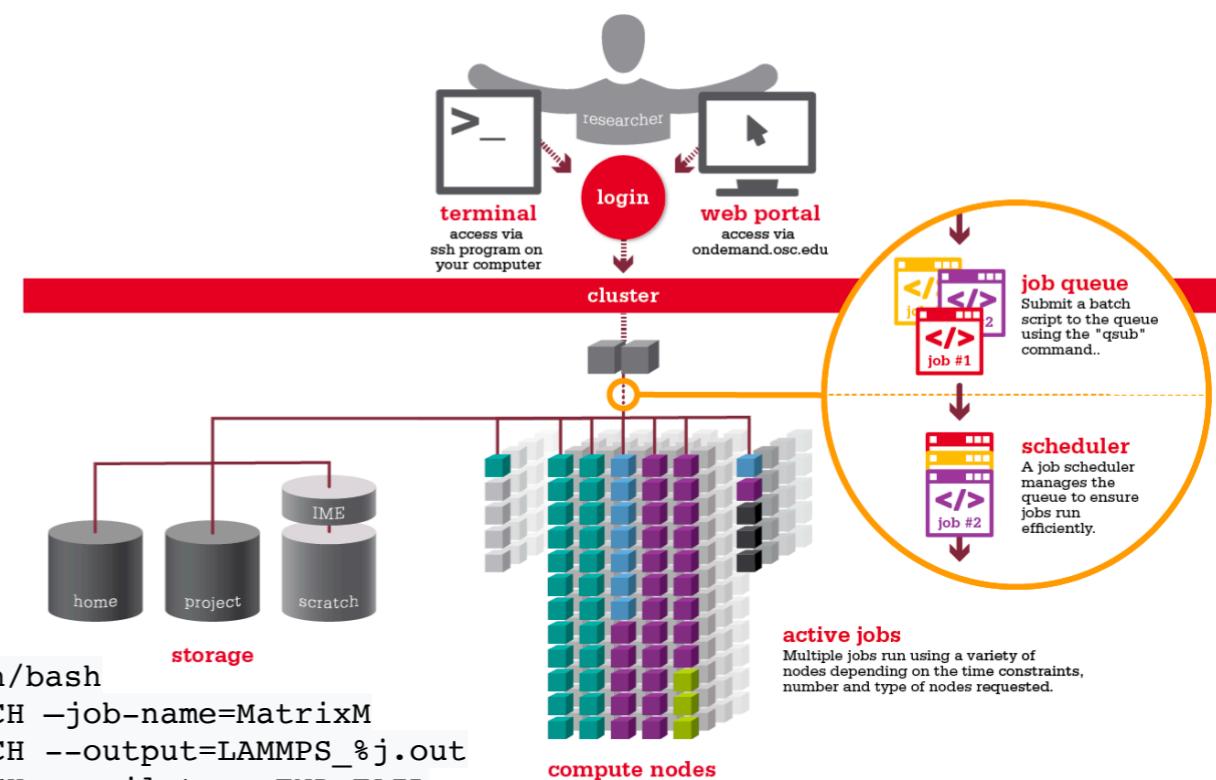
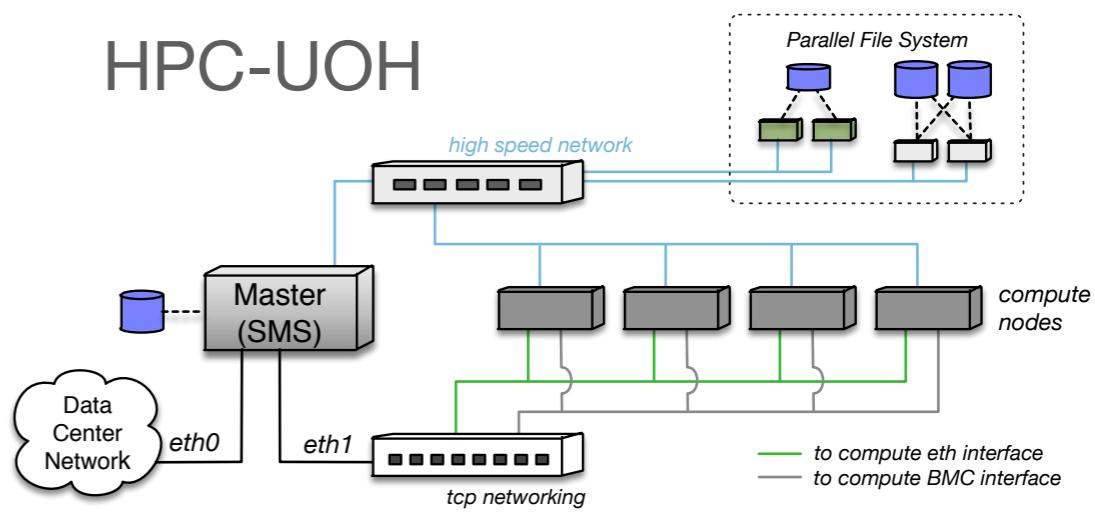
Local, cluster, cloud

- Manejadores de recursos:

- Local** (Ejecuta los procesos en la computadora local)
- SLURM** (<https://slurm.schedmd.com/documentation.html>)
- LSF** (https://en.wikipedia.org/wiki/IBM_Spectrum_LSF)
- Moab** (https://en.wikipedia.org/wiki/Moab_Cluster_Suite)
- OAR** (<https://oar.imag.fr/>)
- PBS/Torque** (http://en.wikipedia.org/wiki/Portable_Batch_System)
- SGE** (<http://gridscheduler.sourceforge.net/>)
- Clouds:**
 - AWS Batch** (<https://aws.amazon.com/batch/>)
 - Azure Batch** (<https://azure.microsoft.com/en-us/services/batch/>)
 - Google Cloud Batch** (<https://cloud.google.com/batch>)
 - Google Life Sciences** (<https://cloud.google.com/life-sciences>)



HPC-UOH



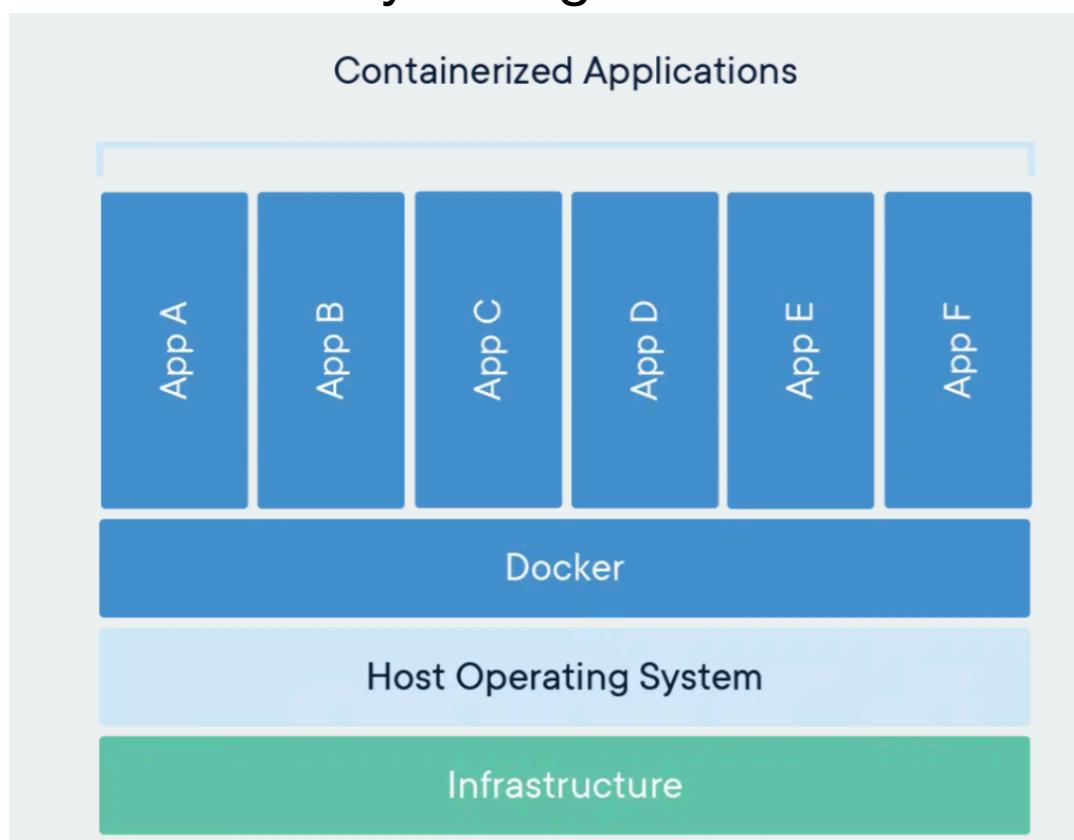
```
#!/bin/bash
#SBATCH --job-name=MatrixM
#SBATCH --output=LAMMPS_%j.out
#SBATCH --mail-type=END,FAIL
#SBATCH --mail-user=alex.digenova@uoh.cl
#SBATCH --nodes=4           # Number of nodes
#SBATCH --ntasks=8          # Number of MPI ranks
#SBATCH --ntasks-per-node=2 # Number of MPI ranks per node
#SBATCH --ntasks-per-socket=1# Number of tasks per processor socket
on the node
#SBATCH --cpus-per-task=8   # Number of OpenMP threads for each
MPI process/rank
#SBATCH --mem-per-cpu=2000mb # Per processor memory request
#SBATCH --time=4-00:00:00    # Walltime in hh:mm:ss or d-hh:mm:ss
```

```
## bash code
date;hostname;pwd
module load intel/2018 openmpi/3.1.0
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK
srun --mpi=pmi_v1 /path/to/app/matrix_multiplication
date
```

Nexflow

Contenedores (Docker/singularity)

- Un contenedor es una unidad estándar de software que empaqueta el código y todas sus dependencias, por lo que la aplicación se ejecuta de forma rápida y fiable de un entorno informático a otro.
- Una imagen de contenedor de Docker es un paquete de software ligero, independiente y ejecutable que incluye todo lo necesario para ejecutar una aplicación: código, binarios, herramientas del sistema, bibliotecas del sistema y configuración.



```
#####
# BASE IMAGE #####
FROM mambaorg/micromamba:0.15.3
##site to test docker configuration files
#####
# METADATA #####
LABEL base_image="mambaorg/micromamba"
LABEL version="0.15.3"
LABEL software="k-count-nf"
LABEL software.version="1.1"
LABEL about.summary="Container image containing all requirements for k-count-nf"- bioconda
LABEL about.home="https://github.com/digenoma-lab/k-count-nf"
LABEL about.documentation="https://github.com/digenoma-lab/k-count-nf/README.md"
LABEL about.license_file="https://github.com/digenoma-lab/k-count-nf/LICENSE.txt" defaults
LABEL about.license="MIT"

#####
# MAINTAINER #####
MAINTAINER Alex Di Genova <digenova@gmail.com>
#####
# INSTALLATION #####
USER root
#the next command install the ps command needed by nexflow to collect run metrics
RUN apt-get update && apt-get install -y procps
USER micromamba
COPY --chown=micromamba:micromamba environment.yml /tmp/environment.yml
RUN micromamba create -y -n k-count -f /tmp/environment.yml && \
    micromamba clean --all --yes
ENV PATH /opt/conda/envs/k-count/bin:$PATH
```

<https://hub.docker.com/>
<https://biocontainers.pro/>



```
singularity pull biocontainers/fastqc:v0.11.9_cv8
singularity shell fastqc_v0.11.9_cv8.sif
```

name: k-count

channels:

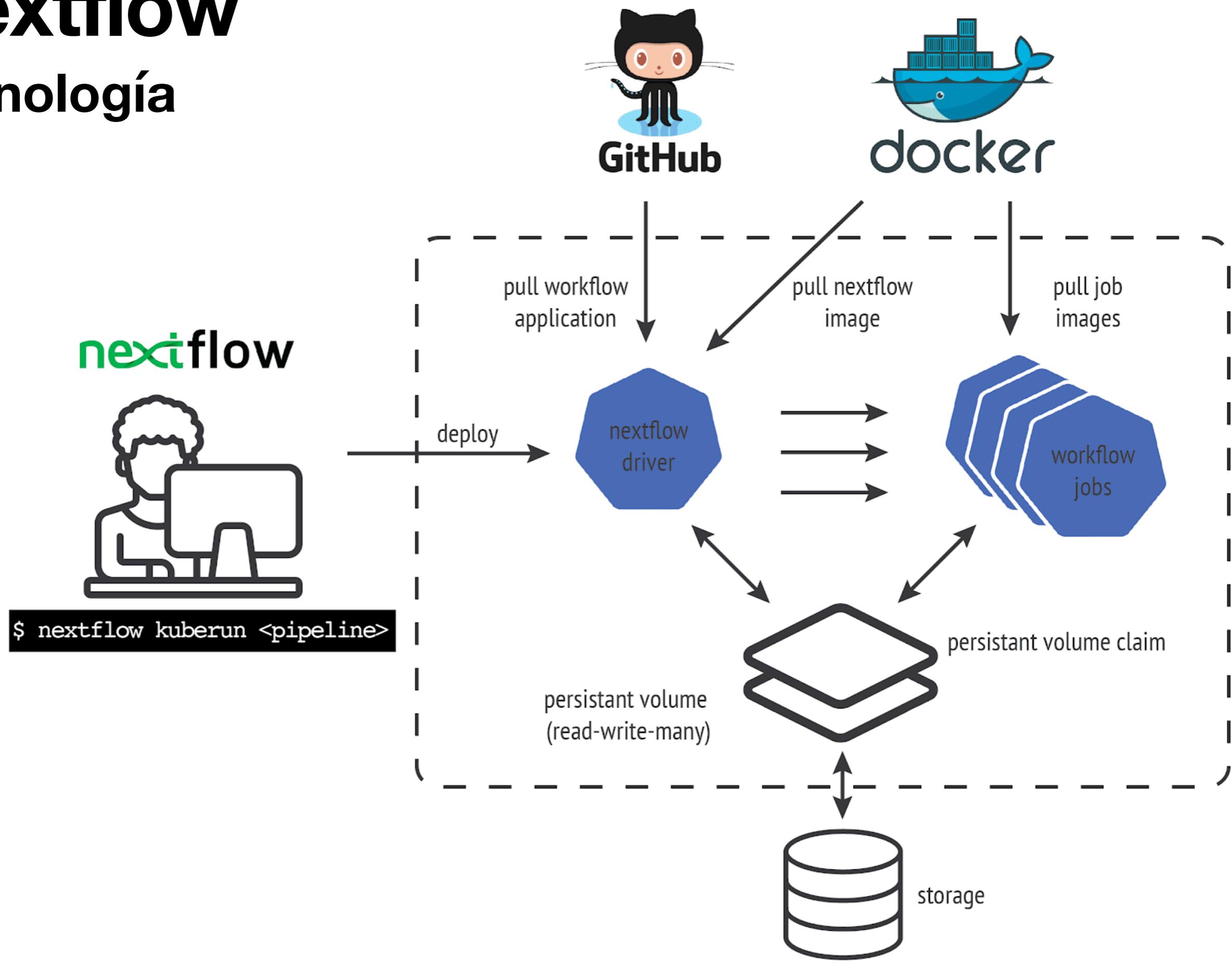
- conda-forge

dependencies:

- kmc=3.1.1rc1

Nextflow

Tecnología



GitHub

Nextflow pipeline

The screenshot shows a GitHub repository page for 'digenoma-lab/longreadstats'. The repository is public and contains 1 branch and 1 tag. The main commit is by 'c-valenzuelac' titled 'Update base.config' with a timestamp of '1b9077b 15 days ago' and 8 commits. The repository has no description, website, or topics provided. It has 0 stars, 1 watching, and 0 forks. There is one release, 'longreadstats v1.0dev', which is the latest version and was published 15 days ago. No packages have been published.

digenoma-lab / longreadstats Public

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main · 1 branch · 1 tag

Go to file Add file Code

About

No description, website, or topics provided.

Readme MIT license Cite this repository

0 stars 1 watching 0 forks

Releases 1

longreadstats v1.0dev Latest
15 days ago

Packages

No packages published Publish your first package

File	Commit Message	Time Ago
base.config	c-valenzuelac Update base.config	15 days ago
assets	adding nanoplot pipeline	17 days ago
bin	adding nanoplot pipeline	17 days ago
conf	Update base.config	15 days ago
docs	adding nanoplot pipeline	17 days ago
lib	adding nanoplot pipeline	17 days ago
modules	adding nanoplot pipeline	17 days ago
reads	adding nanoplot pipeline	17 days ago
subworkflows/local	adding nanoplot pipeline	17 days ago
workflows	adding nanoplot pipeline	17 days ago
CHANGELOG.md	adding nanoplot pipeline	17 days ago
CITATION.cff	adding nanoplot pipeline	17 days ago
CITATIONS.md	adding nanoplot pipeline	17 days ago
LICENSE	adding nanoplot pipeline	17 days ago

<https://github.com/digenoma-lab/longreadstats>

Nextflow

Corriendo un pipeline

```
nextflow run digenoma-lab/longreadstats -profile <singularity> -input samplesheet.csv
```

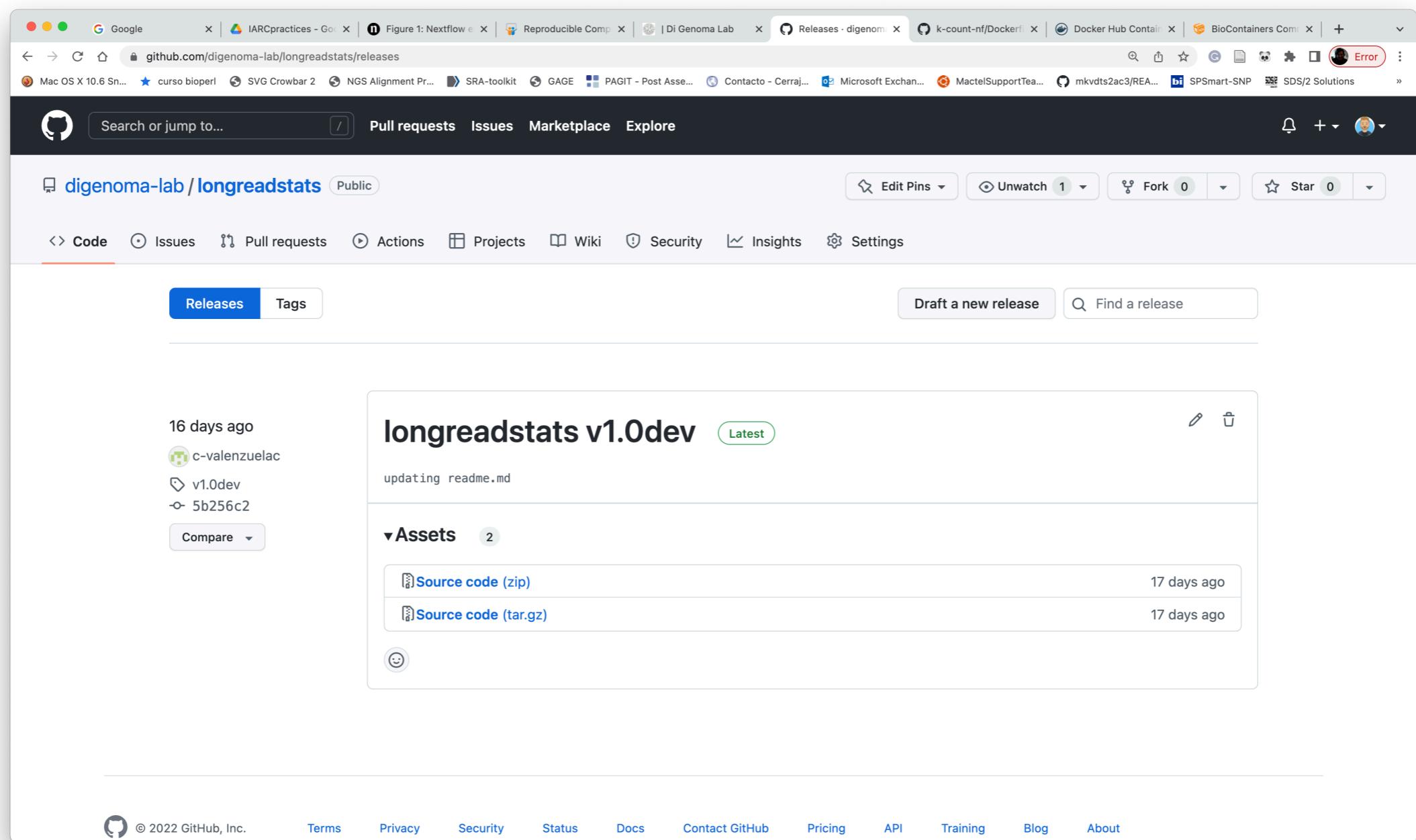
Nombre del pipeline: ruta a un script de *nextflow script*, una ruta a un directorio con un archivo de configuración de *nextflow* que indica un script *nextflow*, o un repositorio en *github* (por ejemplo, <https://github.com/digenoma-lab/longreadstats>)

Opciones de nextflow (“-”): por ejemplo, perfiles definiendo el uso de contenedores o el software local.

sample,fastq_1,fastq_2
A01,/home/adigenova/digenoma-lab-longreadstats/reads/tmp1.fastq.gz,
A02,/home/adigenova/digenoma-lab-longreadstats/reads/tmp2.fastq.gz,
A03,/home/adigenova/digenoma-lab-longreadstats/reads/tmp3.fastq.gz,

Github releases

```
nextflow run digenoma-lab/longreadstats -r v1.0dev --profile <singularity> --input samplesheet.csv
```



Nextflow

Convirtiendo la pesadilla de las dependencias en...

Para correr un pipeline usualmente debemos instalar muchos programas:

gatk4, fastqc, trim-galore, samtools, star, rseqc, ldc, sambamba, samblaster, multiqc, htseq, R with ggplot2, gsalib, reshape package....

..... Un gran sueño.

```
nextflow run digenoma-lab/longreadstats -r v1.0dev --profile <singularity> --input samplesheet.csv
```

Nextflow

Reportes

Figure 1: Nextflow Report

The screenshot shows a web browser window displaying a Nextflow report. The URL is `/Users/adigenova/tmp/report.html`. The browser has many tabs open, including Google, IARCPRACTICES, Figure 1: Nextflow, Reproducible Co..., Di Genoma Lab, digenoma-lab/loc..., k-count-nf/Dock..., Docker Hub Com..., BioContainers, and [festering_banach].

The Nextflow Report page has a dark header with tabs: Nextflow Report, Summary, Resources, Tasks, and [festering_banach].

Nextflow command:

```
nextflow run UPHL-BioNGS/Cecret -profile singularity -c lefraru.nextflow.conf --cleaner fastp --relatedness true --samtools_ampliconstats false --samtools_plot_ampliconstats false --reads merge_reads -resume
```

Configuration details:

CPU-Hours	319.1 (99.9% cached, 0.1% failed)
Launch directory	/home/adigenova/DiGenomaLab/projects/COVID/analysis/adg/CECRET
Work directory	/home/adigenova/DiGenomaLab/projects/COVID/analysis/adg/CECRET/work
Project directory	/home/adigenova/.nextflow/assets/UPHL-BioNGS/Cecret
Script name	main.nf
Script ID	63ca81f81fdc1375962ae1eb234e9dfd
Workflow session	afb6e729-6a87-4e82-ae6a-b61ae0cb8c12
Workflow repository	https://github.com/UPHL-BioNGS/Cecret , revision master (commit hash 0fdad331df4f28f36bcc1c27a752306185f4e7e9)
Workflow profile	singularity
Nextflow version	version 22.09.5.edge, build 5802 (24-09-2022 23:38 UTC)

Resource Usage:

These plots give an overview of the distribution of resource usage for each process.

CPU:

Raw Usage % Allocated

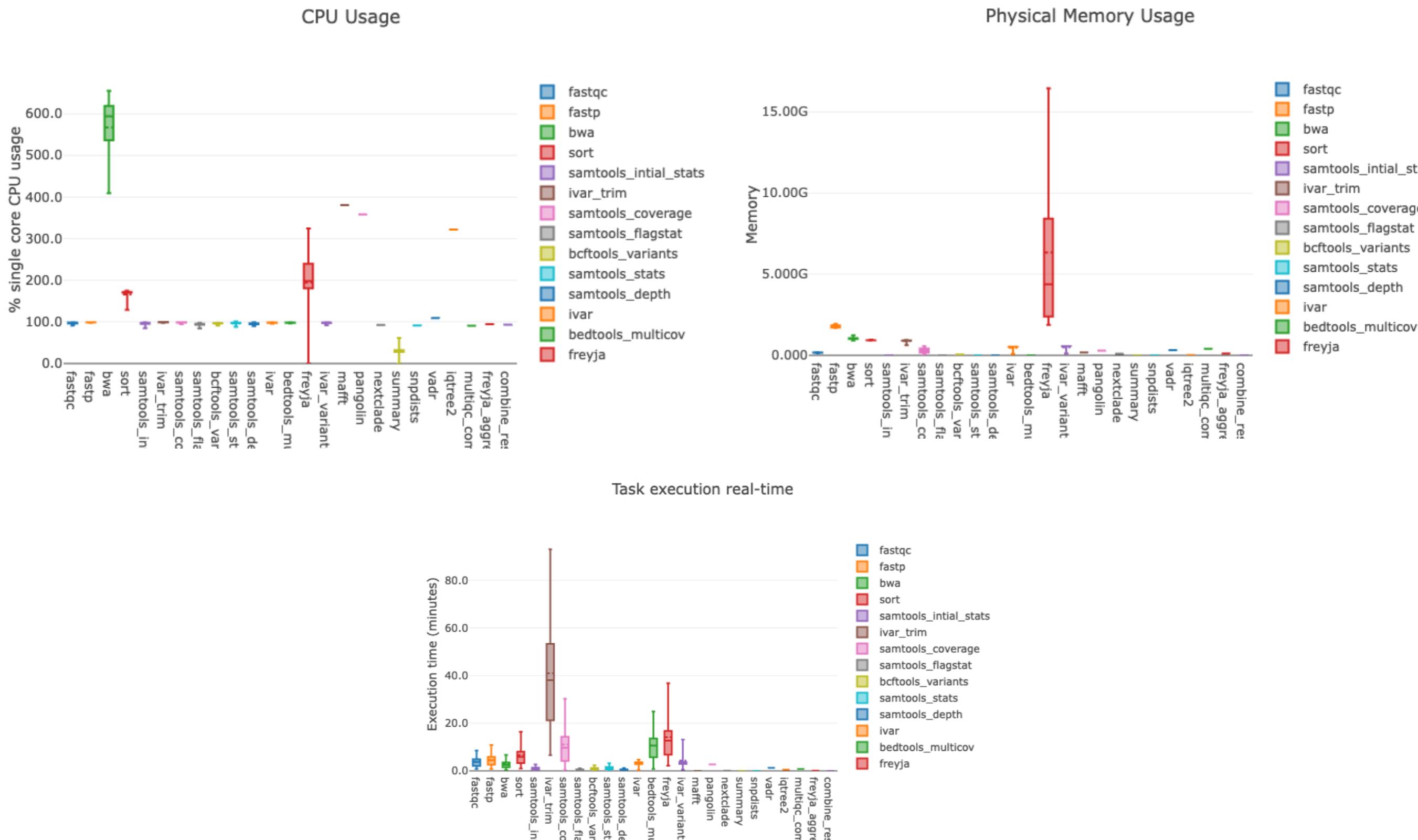
CPU Usage

Legend:

- fastqc
- fastp
- bwa
- sort
- samtools_initial_stats

Nextflow

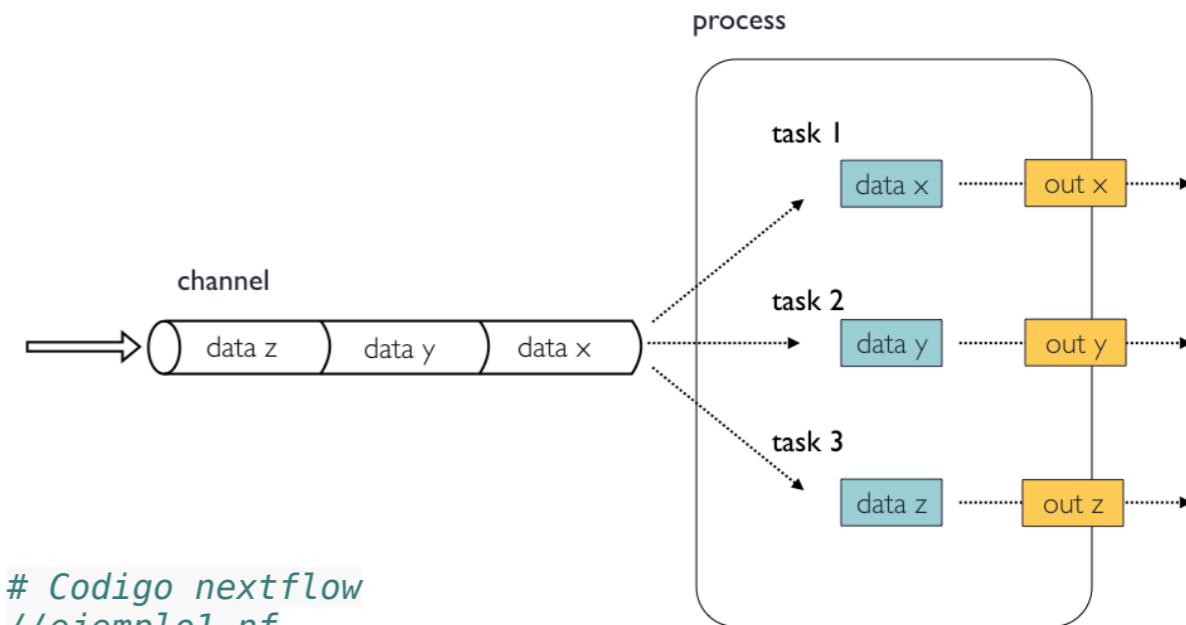
Reportes



Nextflow

Bases de programación

- Un script de Nextflow se crea uniendo diferentes **procesos**. Cada proceso se puede escribir en cualquier lenguaje de comandos que pueda ejecutar la plataforma Linux (Bash, Perl, Ruby, Python, etc.).
- Los **procesos** se ejecutan de forma independiente y están aislados entre sí, es decir, no comparten un estado común (de escritura). La única forma de comunicación es a través de colas FIFO asíncronas, llamadas **canales** en Nextflow.
- Cualquier proceso puede definir uno o más **canales** como entrada y salida. La interacción entre estos procesos y, en última instancia, el propio flujo de ejecución del pipeline, se define implícitamente mediante estas declaraciones de entrada y salida.



```
# Código nextflow
//ejemplo1.nf
nextflow.enable.dsl=2

process NUM_LINEAS {
    input:
        path read

    script:
        """
        printf '${read}'
        gunzip -c ${read} | wc -l
        """
}

reads_ch = Channel.fromPath( 'reads/ref*.fq.gz' )

workflow {
    NUMLINES(reads_ch)
}

# Bash
nextflow run ejemplo1.nf -process.echo

# output
[cd/77af6d] process > NUMLINES (1) [100%] 6 of 6 ✓
ref1_1.fq.gz 58708
ref3_2.fq.gz 52592
ref2_2.fq.gz 81720
ref2_1.fq.gz 81720
ref3_1.fq.gz 52592
ref1_2.fq.gz 58708
```

Nextflow

Primer pipeline

Procesos

```
//pipeline_01.nf
nextflow.enable.dsl=2

process INDEX {
    input:
        path transcriptome
    output:
        path 'index'
    script:
        """
            salmon index -t $transcriptome -i index
        """
}
process QUANT {
    input:
        each path(index)
        tuple(val(pair_id), path(reads))
    output:
        path pair_id
    script:
        """
            salmon quant --threads $task.cpus --libType=U \
                -I $index \
                -1 ${reads[0]} -2 ${reads[1]} -o $pair_id
        """
}
//Definición del pipeline
workflow {
    //canales de entrada
    transcriptome_ch =
        channel.fromPath('transcriptome/*.fa.gz',checkIfExists: true)
    read_pairs_ch =
        channel.fromFilePairs('reads/*_{1,2}.fq.gz',checkIfExists: true)
    //el proceso INDEX toma el canal transcriptome_ch
    index_ch = INDEX(transcriptome_ch)
    //el proceso QUANT toma dos canales como argumentos: Indice y las lecturas
    QUANT( index_ch, read_pairs_ch ).view()
}
```

Channels

Create a channel

```
Channel.of( "A" )
```



Channel content

A



Use the channel in the process

```
input: val(x)      """ echo $x """
```

- `Channel.of()` or `Channel.fromList()` will not alter anything

```
Channel.fromPath( "A" )
```



/path/A

```
Channel.fromPath( "/path/A" )
```



```
input: path(x)      """ cat $x """
```

or

```
input: path("x.txt")      """ cat x.txt """
```

- If the full file path is absent, `.fromPath()` will prefix current folder as path
- So the resulting channels always carry a full path

- `input: path("x.txt")` will create a **symlink** to `/path/A` in the working directory with the name "x.txt"

Channels

Input text file content

A

B

C

Create a channel

```
Channel.fromPath( "file.txt" )
    .splitText { it.strip() }
```

- `.splitText` will read in the `file.txt`
- `it.strip()` will remove the blank items

```
Channel.fromPath( "file.txt" )
    .splitText { it.strip() }
    .map { it -> file(it) }
```

- `file()` adds current folder as path unless there is already full path in the item

A	/path/A.bam	/path/A.bam.bai
B	/path/B.bam	/path/B.bam.bai
C	/path/C.bam	/path/C.bam.bai

Channel content

A →
B →
C →

Use the channel in the process

input: val(x)

- 1 channel, 3 items
- 3 parallel executions of process

input: path(x)

/path/A →
/path/B →
/path/C →

- 1 channel, 3 items
- 3 parallel executions of process

```
Channel.fromPath( "file.tsv" )
```

```
    .splitCsv( sep: "\t" )
```

```
    .map { row -> [ row[0], file(row[1]), file(row[2]) ] }
```

- `.map{ }` is very useful to select columns and specify channel structure. Here it converts the row to a tuple

[A, /path/A.bam, /path/A.bam.bai] →
[B, /path/B.bam, /path/B.bam.bai] →
[C, /path/C.bam, /path/C.bam.bai] →

input: tuple val(x), path(bam), path(bai)

strain	bam
A	A.bam
B	B.bam
C	C.bam

```
Channel.fromPath( "file.tsv" )
```

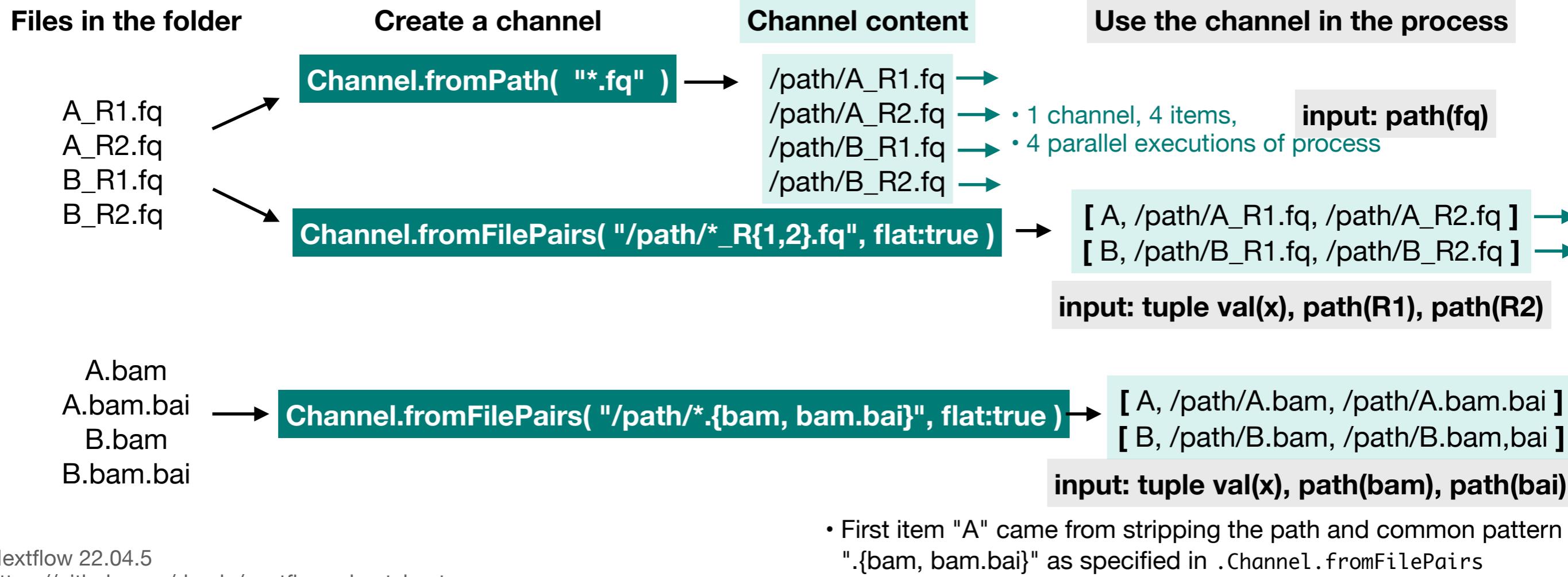
```
    .splitCsv( header:true, sep: "\t" )
```

```
    .map { row ->
```

```
        if ( params.bam_path != "" ) {
            row.bam = "${params.bam_path}/${row.bam}"
        }
    [ row.strain, file("${row.bam}"), file("${row.bam}.bai") ] }
```

- A path is added with `params.bam_path`
- If "`params.bam_path`" doesn't add a full path, `file()` will.
- `[x, y, z]` is the same as `tuple(x, y, z)`

Channels



Channels

examples of 1 channel:

```
Channel.of( "A.fa", "B.fa", "C.fa" )
```

A.fa
B.fa
C.fa

- **Each item (row) has 1 execution of the process**
- Here is 1 channel with 3 items
- 3 parallel executions of the process

```
Channel.of( [ "A.fa", "B.fa", "C.fa" ] )
```

[A.fa, B.fa, C.fa]

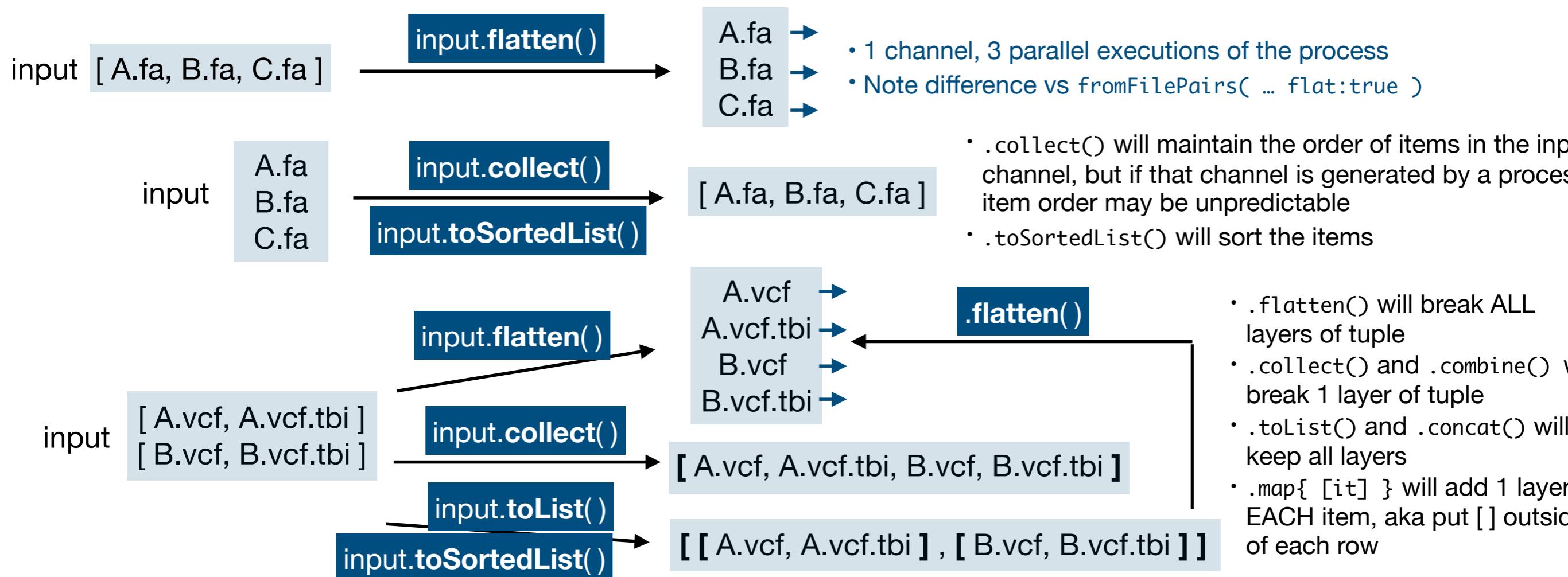
- 1 channel, 1 item, 1 execution

```
Channel.of( [ "I.vcf", "I.vcf.tbi" ] ,  
            [ "II.vcf", "II.vcf.tbi" ] ,  
            [ "III.vcf", "III.vcf.tbi" ] )
```

[I.vcf, I.vcf.tbi]
[II.vcf, II.vcf.tbi]
[III.vcf, III.vcf.tbi]

- 1 channel, 3 items
- 3 parallel executions of process
- [] indicates a “set” “tuple” “ArrayList”

Channels



Channels

vcf [l.vcf]
index [l.vcf.tbi]

`vcf.combine(index)`

[l.vcf, l.vcf.tbi]

- Note how `.combine()` works when there are multiple items in each input channels (see bottom section)

vcf1 [l.vcf, l.vcf.tbi]

`vcf1.combine(vcf2)`

[l.vcf, l.vcf.tbi, ll.vcf, ll.vcf.tbi]

vcf2 [ll.vcf, ll.vcf.tbi]

`vcf1.concat(vcf2)`

[l.vcf, l.vcf.tbi]
[ll.vcf, ll.vcf.tbi]

`.collect()`

- `.concat()` and then `.collect()` sometimes gives the same result as `.combine()`, but they are much more versatile for diverse input types

vcf [l.vcf, l.vcf.tbi]
[ll.vcf, ll.vcf.tbi]

contig contigs.txt

`vcf.concat(contig)`

[l.vcf, l.vcf.tbi]
[ll.vcf, ll.vcf.tbi]
contigs.txt

`.collect()`

[l.vcf, l.vcf.tbi, ll.vcf, ll.vcf.tbi, contigs.txt]

- Can use `input: path("*")` to pass all files to the process

vcf1 [l.vcf, l.vcf.tbi]

`vcf1.map { [it] }`

[[l.vcf, l.vcf.tbi]]

`.combine(contig)`

[[l.vcf, l.vcf.tbi], contigs.txt]

- This is especially useful when number of items in vcf1 could vary. In process: `input: tuple path("*"). path(contig)`

vcf
A.vcf
B.vcf

`vcf.combine(contig)`

[A.vcf, chr1]
[A.vcf, chr2]
[B.vcf, chr1]
[B.vcf, chr2]

- All combinations

input1

[A, 1]
[B, 2]

- counting starts from 0

`input1.combine(input2, by:0)`

[A, 1, 1]
[B, 2, m]
[B, 2, n]

input2

[A, 1]
[B, m]
[B, n]

`input1.join(input2)`

[A, 1, 1]
[B, 2, m]

[l, m, n]

`input2.collect { it[1] }`

- "it" represents an item, which is

- `join()` will keep only the first match

Nextflow patterns

```
process foo {  
    debug true  
    input:  
        path x  
  
    script:  
        """  
        echo your_command --input $x  
        """  
}  
  
workflow {  
    foo("$baseDir/data/reads/*_1.fq.gz")  
}  
    Process per file path  
  
process foo {  
    debug true  
  
    input:  
        tuple val(sampleId), file(reads)  
  
    script:  
        """  
        echo your_command --sample $sampleId --reads $reads  
        """  
}  
  
    Process per file pairs  
  
workflow {  
    Channel.fromFilePairs("$baseDir/data/reads/*_{1,2}.fq.gz", checkIfExists:true) \  
        | foo  
}
```

Nextflow patterns

Table 1

sampleId	read1	read2
FC816RLABXX	110101_I315_FC816RLABXX_L1_HUMrutRGXDIAAPE_1	110101_I315_FC816RLABXX_L1_HUMrutRGXDIAAPE_2
C812MWABXX	110105_I186_FC812MWABXX_L8_HUMrutRGVDIABPE_1	110105_I186_FC812MWABXX_L8_HUMrutRGVDIABPE_2
FC81DE8ABXX	110121_I288_FC81DE8ABXX_L3_HUMrutRGXDIAAPE_1	110121_I288_FC81DE8ABXX_L3_HUMrutRGXDIAAPE_2
FC81DB5ABXX	110122_I329_FC81DB5ABXX_L6_HUMrutRGVDIAAPE_1	110122_I329_FC81DB5ABXX_L6_HUMrutRGVDIAAPE_2
FC819P0ABXX	110128_I481_FC819P0ABXX_L5_HUMrutRGWDIAAPE_1	110128_I481_FC819P0ABXX_L5_HUMrutRGWDIAAPE_2

params.index = "\$baseDir/data/index.csv"

```
process foo {
    debug true
    input:
        tuple val(sampleId), file(read1), file(read2)

    script:
    """
    echo your_command --sample $sampleId --reads $read1 $read2
    """

    workflow {
        Channel.fromPath(params.index) \
            | splitCsv(header:true) \
            | map { row-> tuple(row.sampleId, file(row.read1), file(row.read2)) } \
            | foo
    }
}
```

Nextflow patterns

```
process foo {
    input:
    path x
    output:
    path 'file.fq'
    script:
    """
        < $x zcat > file.fq
    """
}

process bar {
    debug true
    input:
    path '*.fq'
    script:
    """
        cat *.fq | head -n 50
    """
}

workflow {
    Channel.fromPath("$baseDir/data/reads/*_1.fq.gz", checkIfExists: true) \
        | foo \
        | collect \
        | bar
}
```

Process all outputs altogether

Real example

Nextflow

- BRCA pipeline
- <https://github.com/digenoma-lab/BRCA/blob/main/>

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

nextflow.enable.dsl = 2
//check some variables before execution

//loading scripts from modules
include {FASTQC} from './modules/fastqc'
include {BWAMEM} from './modules/bwamem'
include {MERGEB} from './modules/mergeb'
include {ELPREP} from './modules/elprep'
include {SAMTOOLS} from './modules/samtools'
include {QUALIMAP} from './modules/qualimap'
include {B2C} from './modules/b2c'
include {STRELKA_ONESAMPLE} from './modules/strelka'
include {STRELKA_POOL} from './modules/strelka'
include {BCFTOOLS_FILTER; BCFTOOLS_FILTER as BF} from './modules/bcftools'
include {ANNOVAR} from './modules/annovar'
```

Nextflow

Groovy

- Apache Groovy es un lenguaje de programación orientado a objetos compatible con la sintaxis de Java para la plataforma Java.

The screenshot shows a web browser window with the URL groovy-lang.org/syntax.html. The page is titled "Syntax" and is part of the "Apache Groovy" documentation. The left sidebar contains a "Table of contents" with sections such as Comments, Keywords, Identifiers, Strings, and Special characters. The main content area starts with a brief introduction about the Groovy grammar's derivation from Java. It then details "Comments", including "Single-line comment" (using `//`) and "Multiline comment" (using `/* */`). It also covers "Groovydoc comments", which are multiline comments starting with `/**` and ending with `*/`, and how they are associated with Java code. Below this, it discusses "String interpolation" and "Special case of string interpolation". At the bottom, there is a note about Groovydoc comments being compatible with Java annotations and enums. The browser interface includes a toolbar at the top with various icons and tabs, and a tab bar at the bottom showing other open files like "newplot (Z).png" and "newplot (I).png".

- <http://groovy-lang.org/documentation.html>
- https://github.com/chlazaris/Nextflow_training/blob/main/nextflow_cheatsheet.md
- <https://www.nextflow.io/docs/latest/script.html>

Nextflow

Creando pipelines con el estandar nf-core

```
nf-core create -n long-read-stats -d "pipeline for computing long-read statistics" -a "Alex Di Genova"
```

```
nf-core modules list remote
```

```
nf-core modules install nanoplot
```

The screenshot shows a GitHub repository page for 'digenoma-lab/longreadstats'. The repository has 8 commits and 1 branch. The code tab is selected, showing files like base.config, assets, bin, conf, docs, lib, modules, reads, subworkflows/local, workflows, CHANGELOG.md, CITATION.cff, CITATIONS.md, and LICENSE. The About section indicates no description, website, or topics provided. A 'Readme' link is present. The Releases section shows 'longreadstats v1.0dev' (Latest) from 15 days ago. The Packages section indicates no packages published.

```
[adigenova@leftraru1 ~]$ nf-core modules list remote
```



nf-core/tools version 2.5.1 – <https://nf-co.re>
There is a new version of nf-core/tools available! (2.6)

INFO Modules available from nf-core/modules (master):

Module Name
nf-core/abacas
nf-core/abricate/run
nf-core/abricate/summary
nf-core/adapterremoval
nf-core/adapterremovalfixprefix
nf-core/agrvate
nf-core/allelecounter
nf-core/ampir
nf-core/amplify/predict
nf-core/amps
nf-core/amrfinderplus/run
nf-core/amrfinderplus/update
nf-core/angsd/docounts
nf-core/antismash/antismashlite
nf-core/antismash/antismashliteownloadadbases
nf-core/aria2
nf-core/ariba/getref
nf-core/ariba/run

Preguntas?