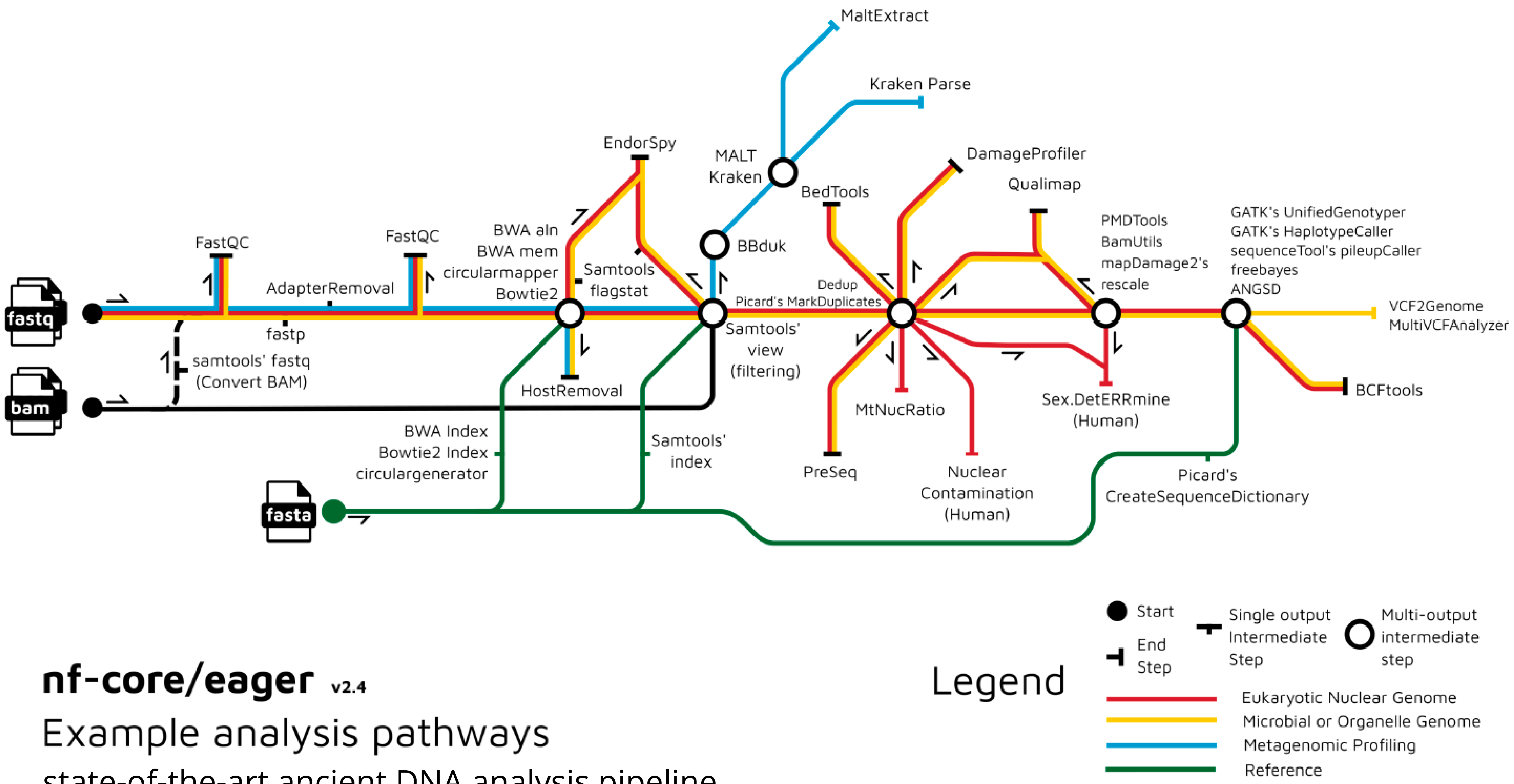
nextflow

Reproducible, Scalable, and Shareable Data Analysis Workflows

Writing modern workflows is complex



nf-core/eager v2.4

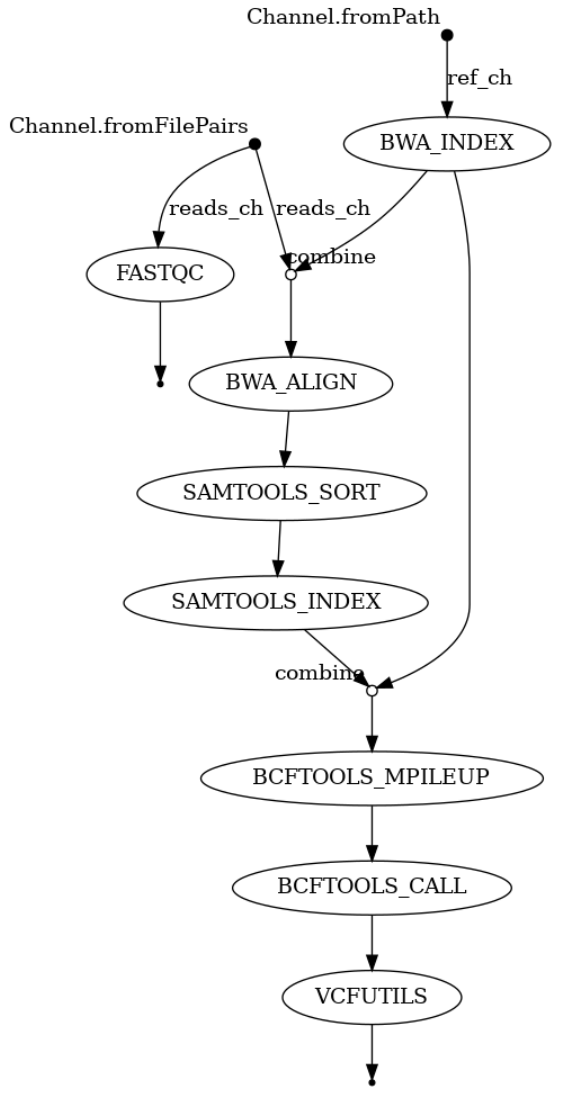
Example analysis pathways

state-of-the-art ancient DNA analysis pipeline

Data analysis workflows

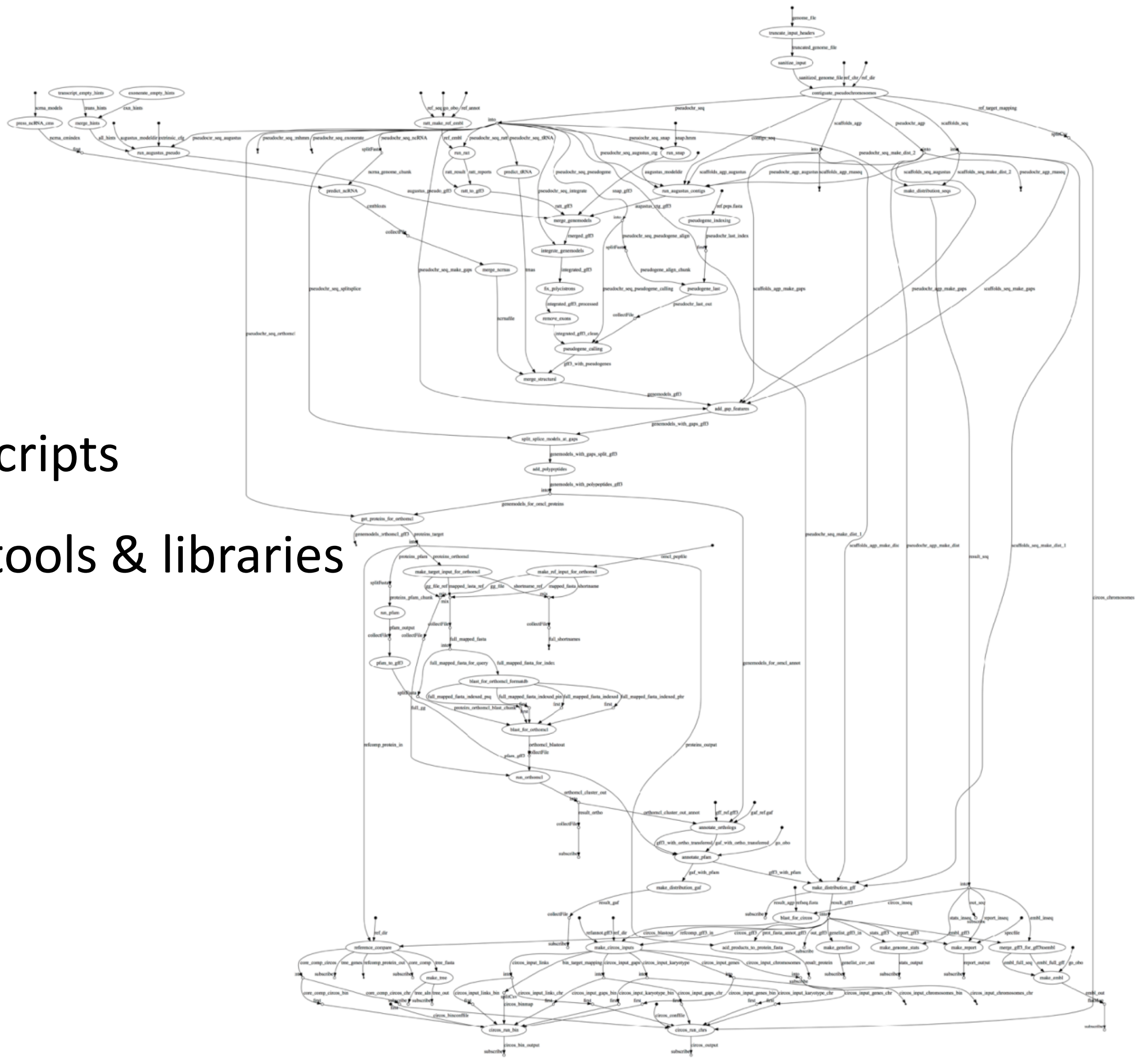
- Data analysis applications perform computation to generate information from (large) datasets
- Embarrassingly parallel! → can spawn 100s-100k jobs over distributed cluster
- Mash-up of many different tools and scripts (dependencies!)
- Complex dependency trees and configuration → very fragile ecosystem!

A lot of moving parts!



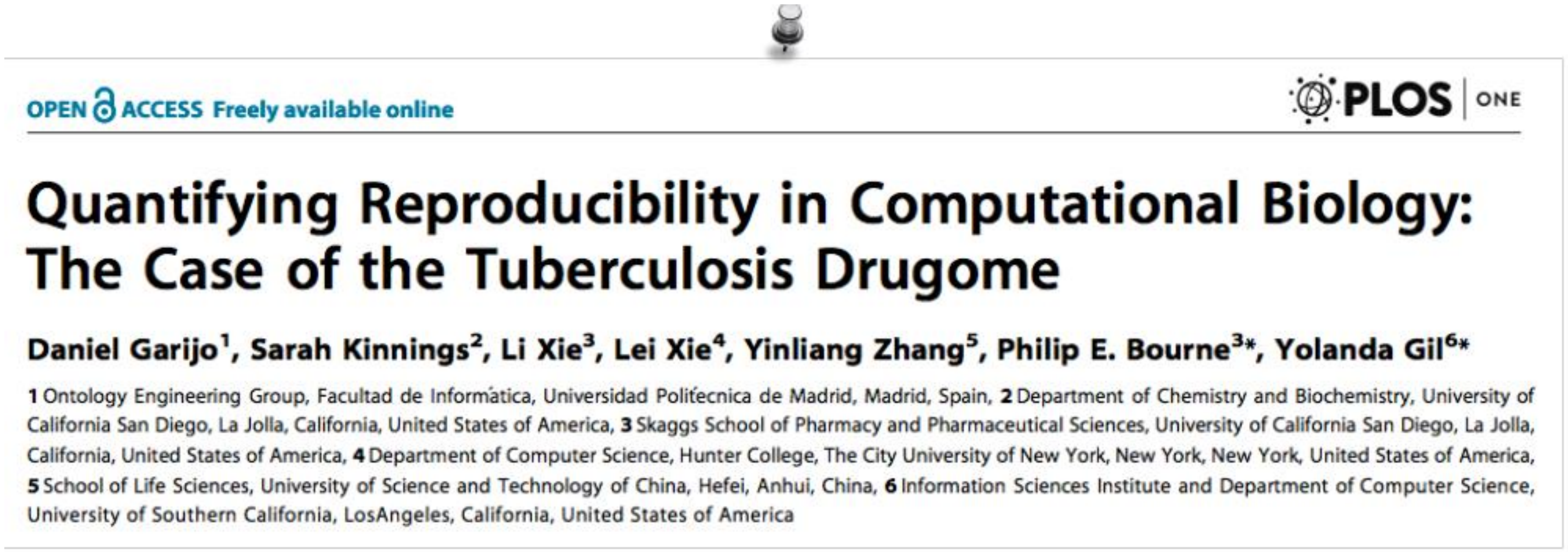
Variant-Calling pipeline

- 70 processes
- 55 external scripts
- 39 software tools & libraries



Steinbiss et al., Companion parassite genome annotation pipeline, DOI: 10.1093/nar/gkw292

To reproduce the result of a typical computational biology experiment requires 280 hours.


OPEN ACCESS Freely available online

PLOS | ONE

Quantifying Reproducibility in Computational Biology: The Case of the Tuberculosis Drugome

Daniel Garijo[1], Sarah Kinnings[2], Li Xie[3], Lei Xie[4], Yinliang Zhang[5], Philip E. Bourne[3]*, Yolanda Gil[6]*

1 Ontology Engineering Group, Facultad de Informática, Universidad Politécnica de Madrid, Madrid, Spain, 2 Department of Chemistry and Biochemistry, University of California San Diego, La Jolla, California, United States of America, 3 Skaggs School of Pharmacy and Pharmaceutical Sciences, University of California San Diego, La Jolla, California, United States of America, 4 Department of Computer Science, Hunter College, The City University of New York, New York, New York, United States of America, 5 School of Life Sciences, University of Science and Technology of China, Hefei, Anhui, China, 6 Information Sciences Institute and Department of Computer Science, University of Southern California, LosAngeles, California, United States of America

≈1.7 months!

The **same application**

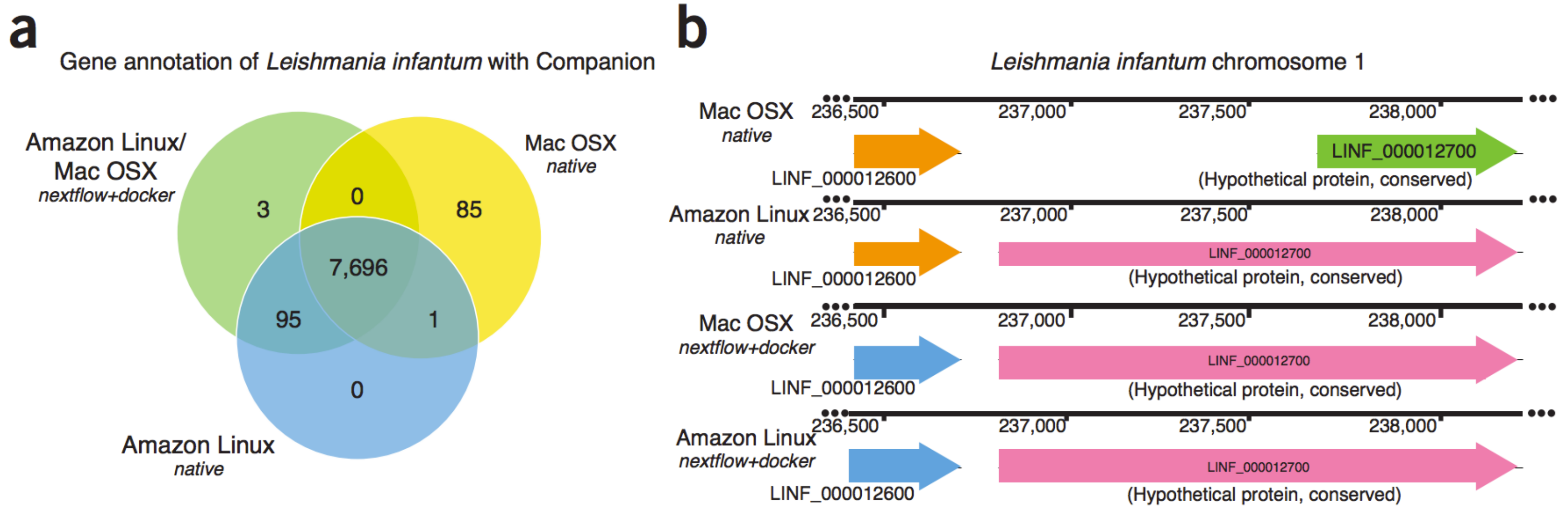
deployed in

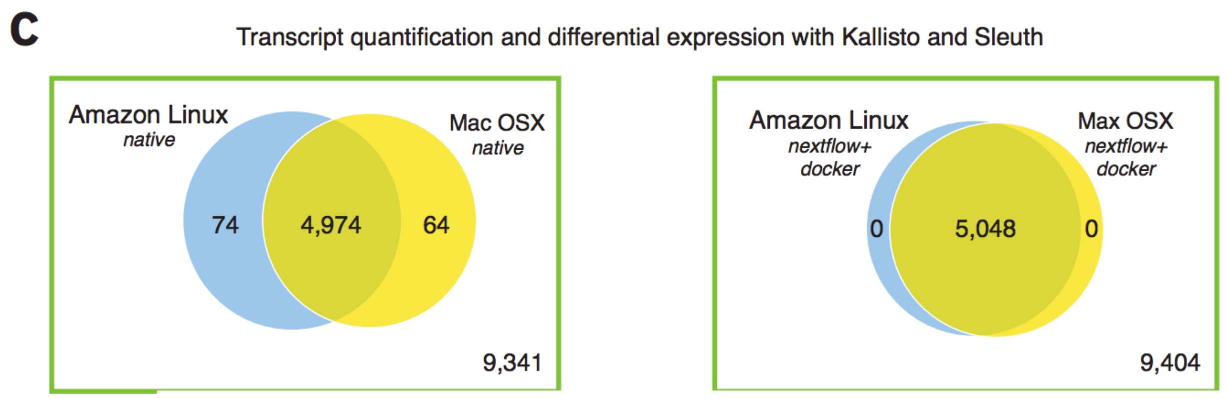
different environments

Produces

different results

Differences in genome annotations & expressed genes





VOLUME 35 NUMBER 4 APRIL 2017 NATURE BIOTECHNOLOGY

* Di Tommaso P, et al., *Nextflow enables computational reproducibility*, Nature Biotech, 2017

Why do Reproducible Research?

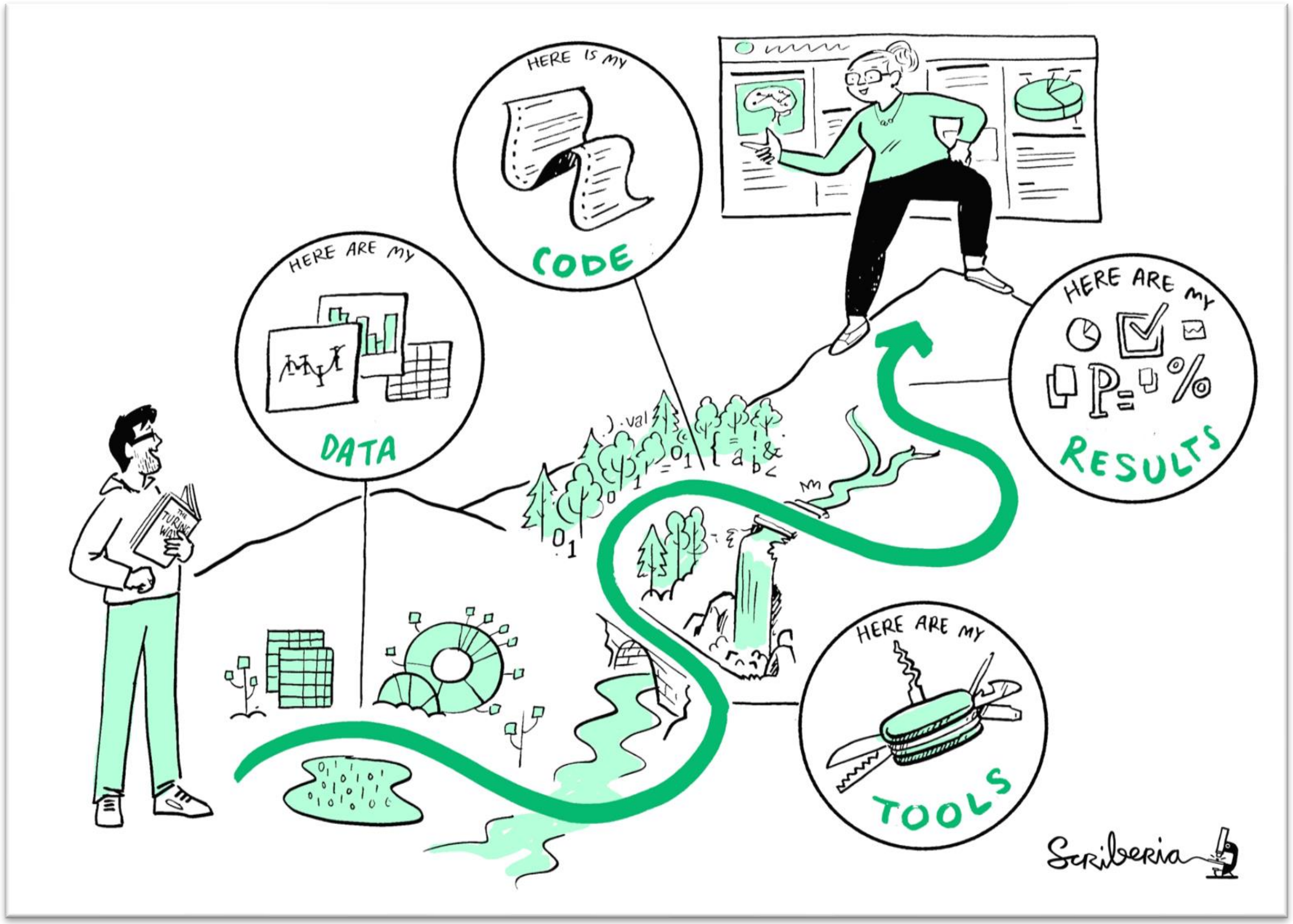
An article about computational results is advertising, not scholarship.

The actual scholarship is the full software environment, code and data, that produced the result

- ***Claerbout & Karrenbach***

https://library.seg.org/doi/abs/10.1190/1.1822162, doi:https://doi.org/10.1190/1.1822162

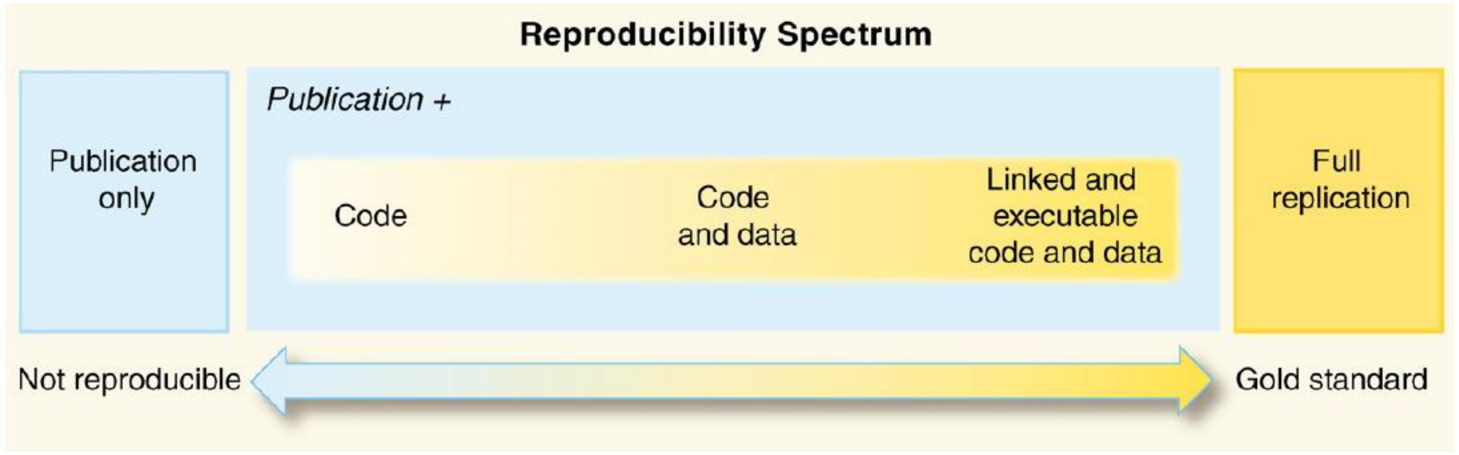
Authors **need to** provide all the necessary data and the computer codes to run the analysis again, re-creating the results.



The Turing Way project illustration by Scriberia. Used under a CC-BY 4.0 licence. DOI: 10.5281/zenodo.3332807.
https://the-turing-way.netlify.app/reproducible-research/

covers topics related to skills, tools and best practices for research reproducibility.

What does this actually look like?



https://science.sciencemag.org/content/334/6060/1226

Reproducibility Checklist

- ❑ Uses Code – instead of pointing and clicking
- ❑ Data archived and available
- ❑ Open-Source – Code transparency is key to reproducibility
- ❑ Version tracking – using version-control software as Git and Github/Gitlab
- ❑ Replicate your environment – using containers or conda
- ❑ Analyses documented – Jupyter Notebooks or Rmarkdown
- ❑ Automated – workflow management tool (Snakemake, CWL, Nextflow)

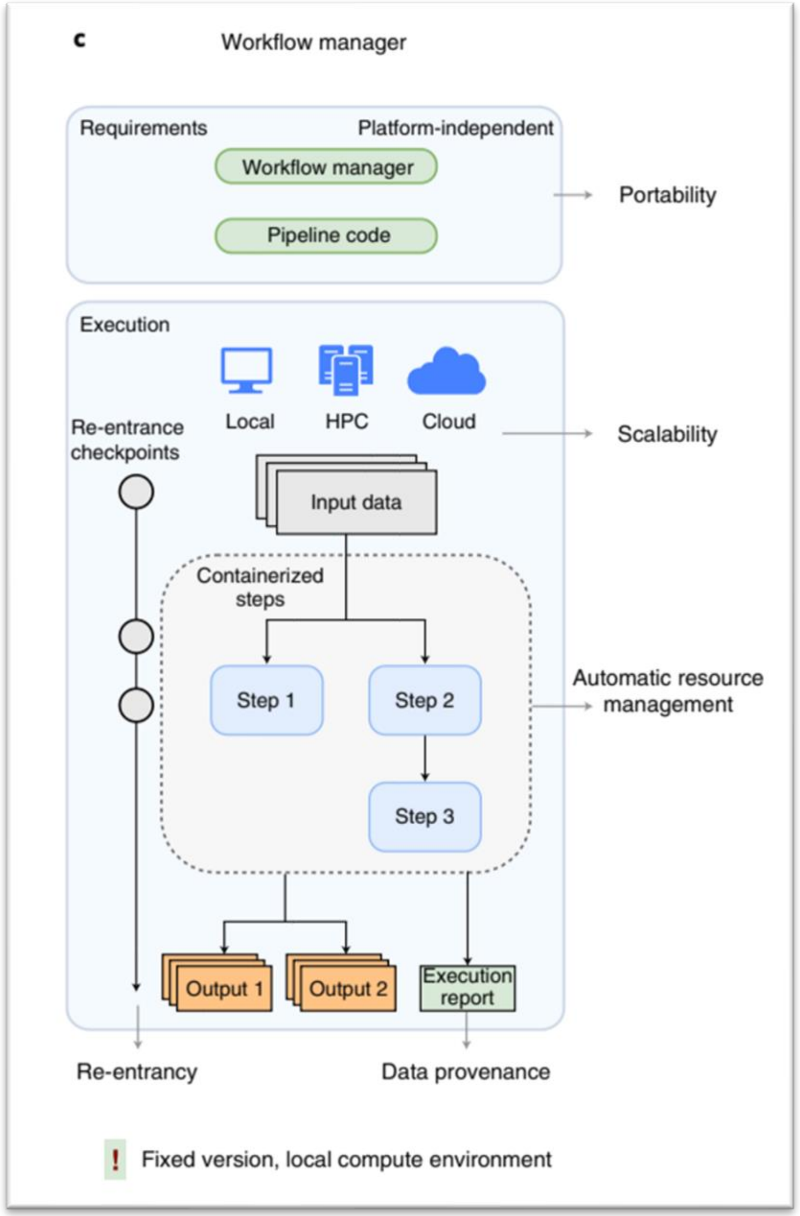
What should be our goal?

Findable Accessible Interoperable Reusable + Equitable Scalable Tested

Streamlining data analysis to make it:

- Findable ⇒ openly available and searchable through repositories
- Accessible ⇒ can be used by everyone
- Interoperable ⇒ portable to any cluster & cloud
- Reusable ⇒ in a reproducible way
- Equitable ⇒ free of bias
- Scalable ⇒ from laptop to supercomputer
- Tested ⇒ validated using modern software practices

Scientific Workflow Management Systems



(WfMSs) automate computational analyses by stringing together individual data processing tasks into cohesive pipelines.

They abstract away the issues of orchestrating data movement and processing, managing task dependencies, and allocating resources within the compute infrastructure.

Wratten, L., Wilm, A. & Göke, J. Reproducible, scalable, and shareable analysis pipelines with bioinformatics workflow managers. Nat Methods 18, 1161–1168 (2021). https://doi.org/10.1038/s41592-021-01254-9

Aspects of WfMSs relevant to bioinformatics

- **Modularity of the pipeline to enable checkpointing**
- **Scalability with respect to the number of tasks in the pipeline**
- **Robustness against failures due to data issues, resource unavailability, or aborted execution**
- **Reproducibility via logs recording data provenance and task execution**
- **Portability across compute environments**
- **Ease of development by users with a range of experience and computational knowledge.**

What is Nextflow?

nextflow *script*

Write code
in any language

R BASH

Define orchestration with
dataflow programming

Define software
dependencies
with containers

CONDA
docker

Git GitHub GitLab Bitbucket

Version control

nextflow *runtime*

Orchestration of tasks to
deploy anywhere with ease

Supports all major platforms

aws Azure
GRID ENGINE slurm workload manager HTCondor
Platform Computing an IBM Company kubernetes PBS Works

**Nextflow is
a language,
a runtime and
a community**

https://nextflow.io/

How does it work?

- **Fast prototyping** ⇒ custom DSL that enables task composition, simplifies most use cases + general purpose programming language for corner cases

- **Easy parallelization** ⇒ declarative reactive programming model based on dataflow paradigm, implicit portable parallelism

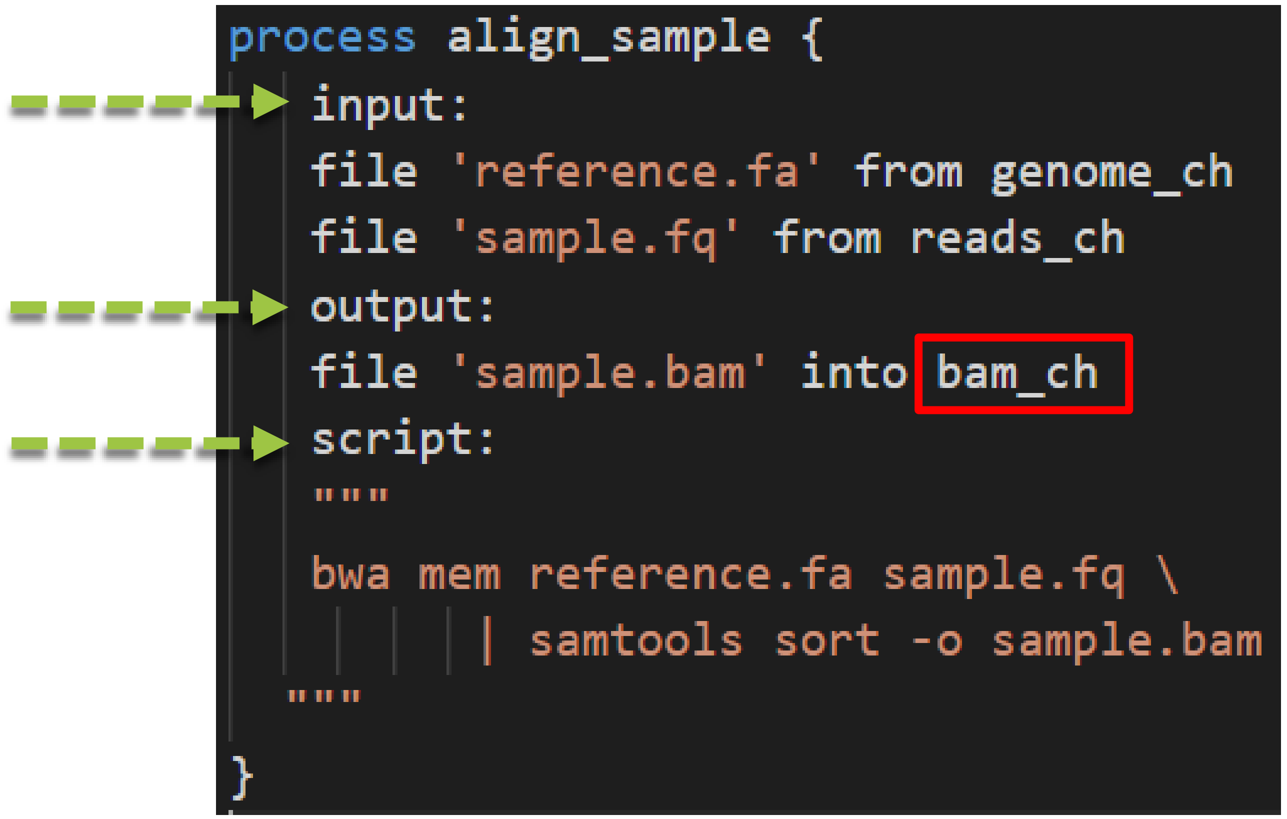
- **Self-contained** ⇒ functional approach, a task execution is idempotent i.e., cannot modify the state of other tasks + isolate dependencies with containers

- **Portable deployments** ⇒ executor abstraction layer + deployment configuration from implementation logic

Task example

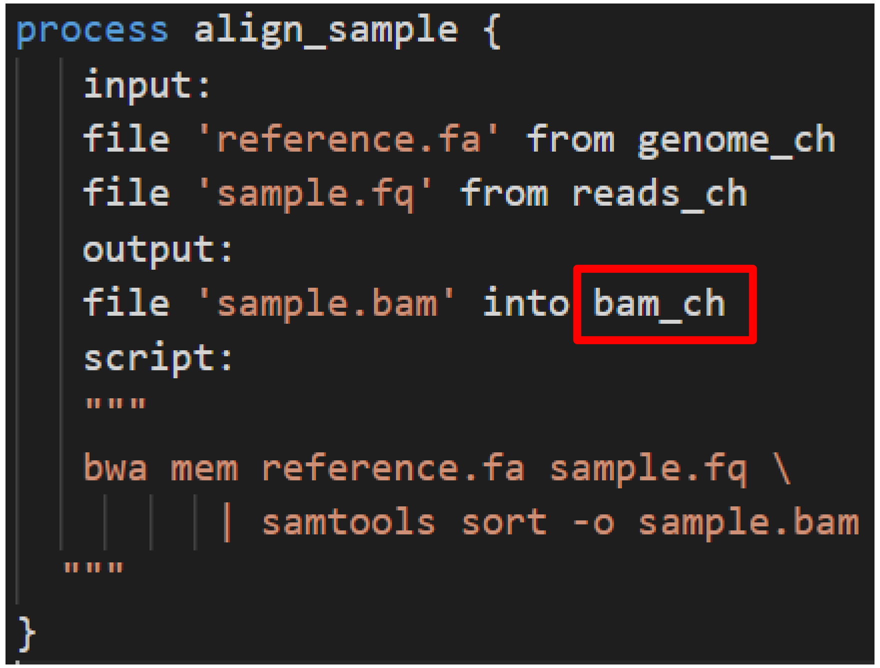
```
bwa mem reference.fa sample.fq \
| samtools sort -o sample.bam
```

Task example

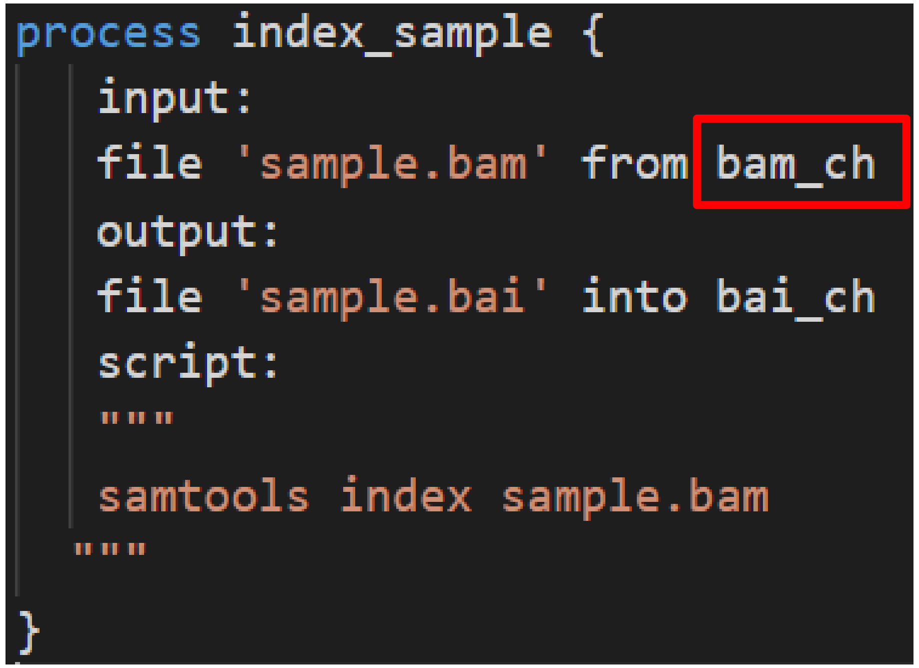


```
process align_sample {
  input:
  file 'reference.fa' from genome_ch
  file 'sample.fq' from reads_ch
  output:
  file 'sample.bam' into bam_ch
  script:
  """
  bwa mem reference.fa sample.fq \
        | samtools sort -o sample.bam
  """
}
```

Task Composition

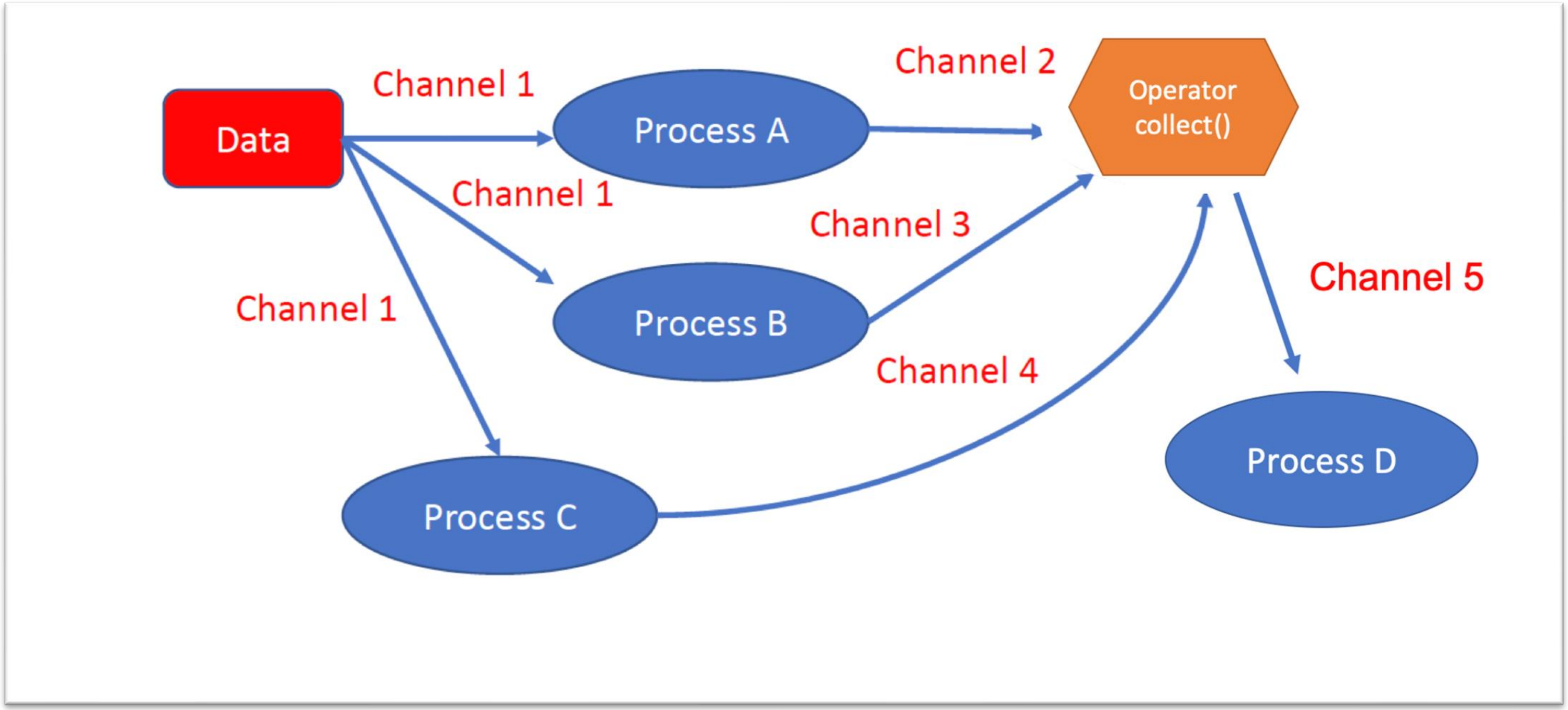


```
process align_sample {
    input:
    file 'reference.fa' from genome_ch
    file 'sample.fq' from reads_ch
    output:
    file 'sample.bam' into bam_ch
    script:
    """
    bwa mem reference.fa sample.fq \
            | samtools sort -o sample.bam
   """
}
```



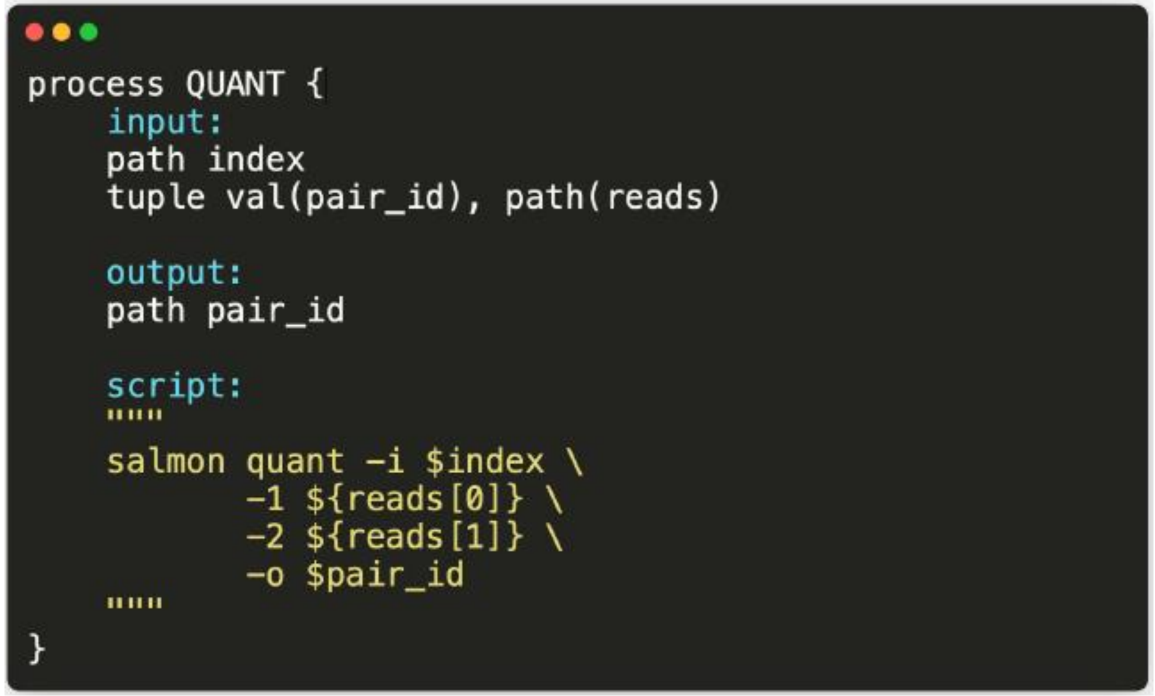
```
process index_sample {
    input:
    file 'sample.bam' from bam_ch
    output:
    file 'sample.bai' into bai_ch
    script:
    """
    samtools index sample.bam
   """
}
```

Workflow


Data
Channel 1
Process A
Channel 2
Operator collect()
Channel 1
Channel 3
Process B
Channel 1
Channel 4
Channel 5
Process C
Process D

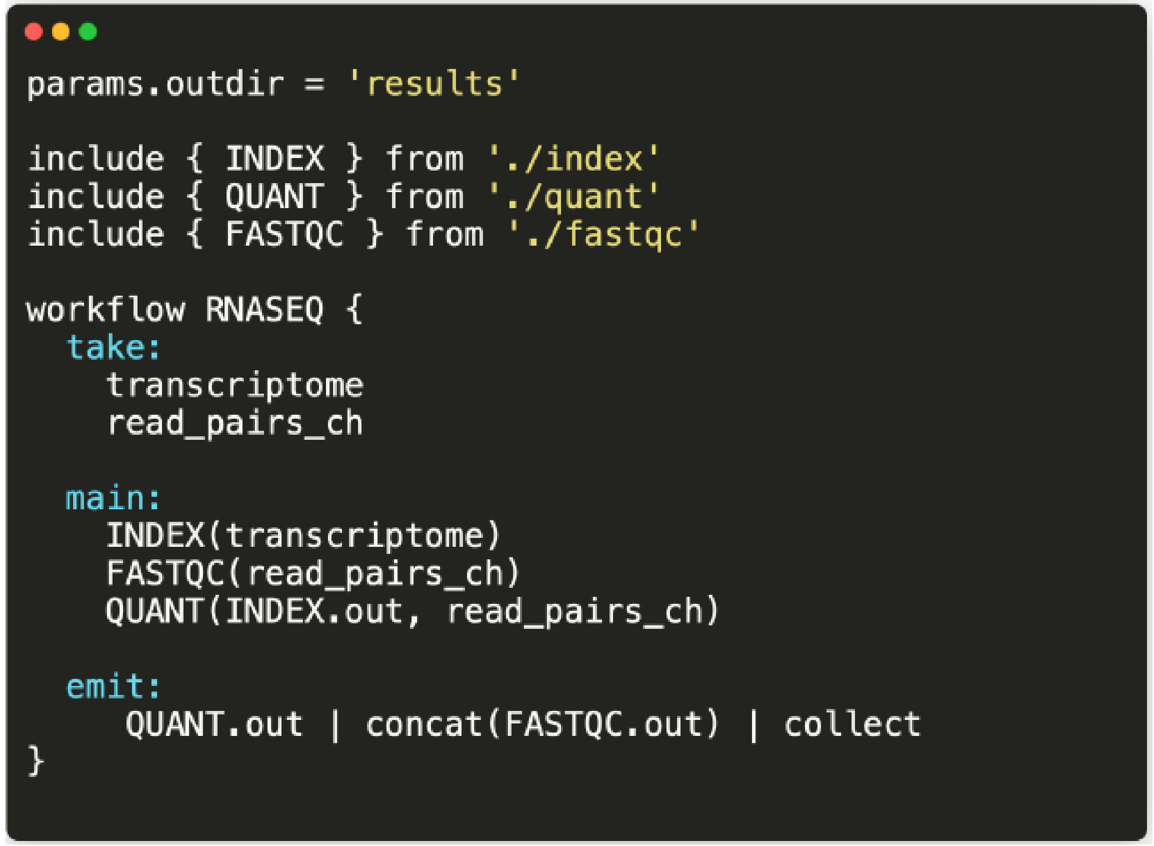
Nextflow Syntax

task



```
process QUANT {
    input:
    path index
    tuple val(pair_id), path(reads)

    output:
    path pair_id

    script:
    """
    salmon quant -i $index \
           -1 ${reads[0]} \
           -2 ${reads[1]} \
           -o $pair_id
    """
}
```

workflow

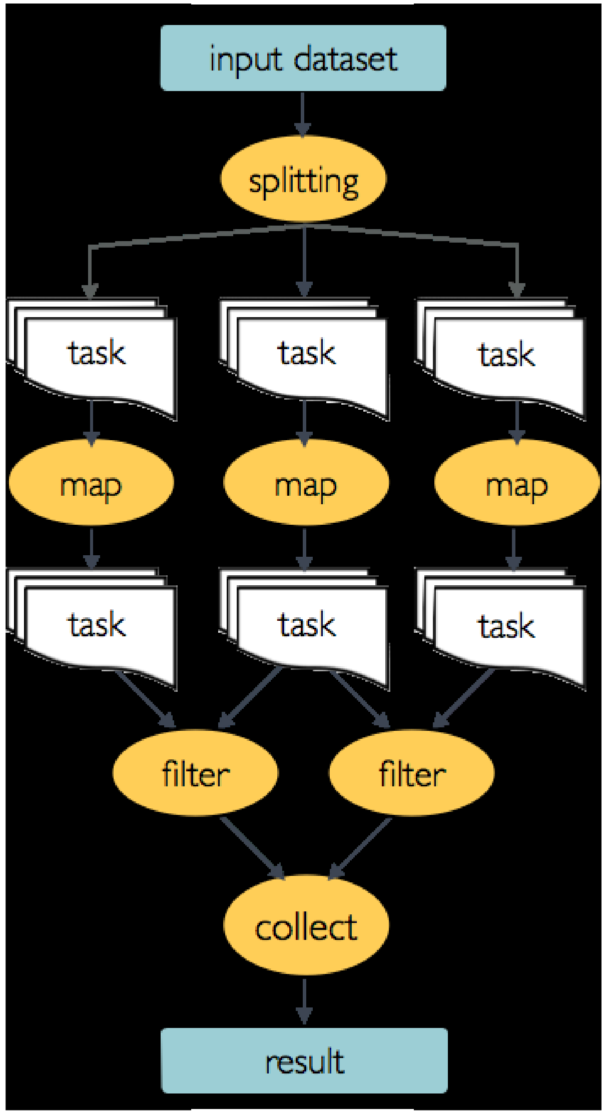


```
params.outdir = 'results'

include { INDEX } from './index'
include { QUANT } from './quant'
include { FASTQC } from './fastqc'

workflow RNASEQ {
  take:
    transcriptome
    read_pairs_ch

  main:
    INDEX(transcriptome)
    FASTQC(read_pairs_ch)
    QUANT(INDEX.out, read_pairs_ch)

  emit:
     QUANT.out | concat(FASTQC.out) | collect
}
```

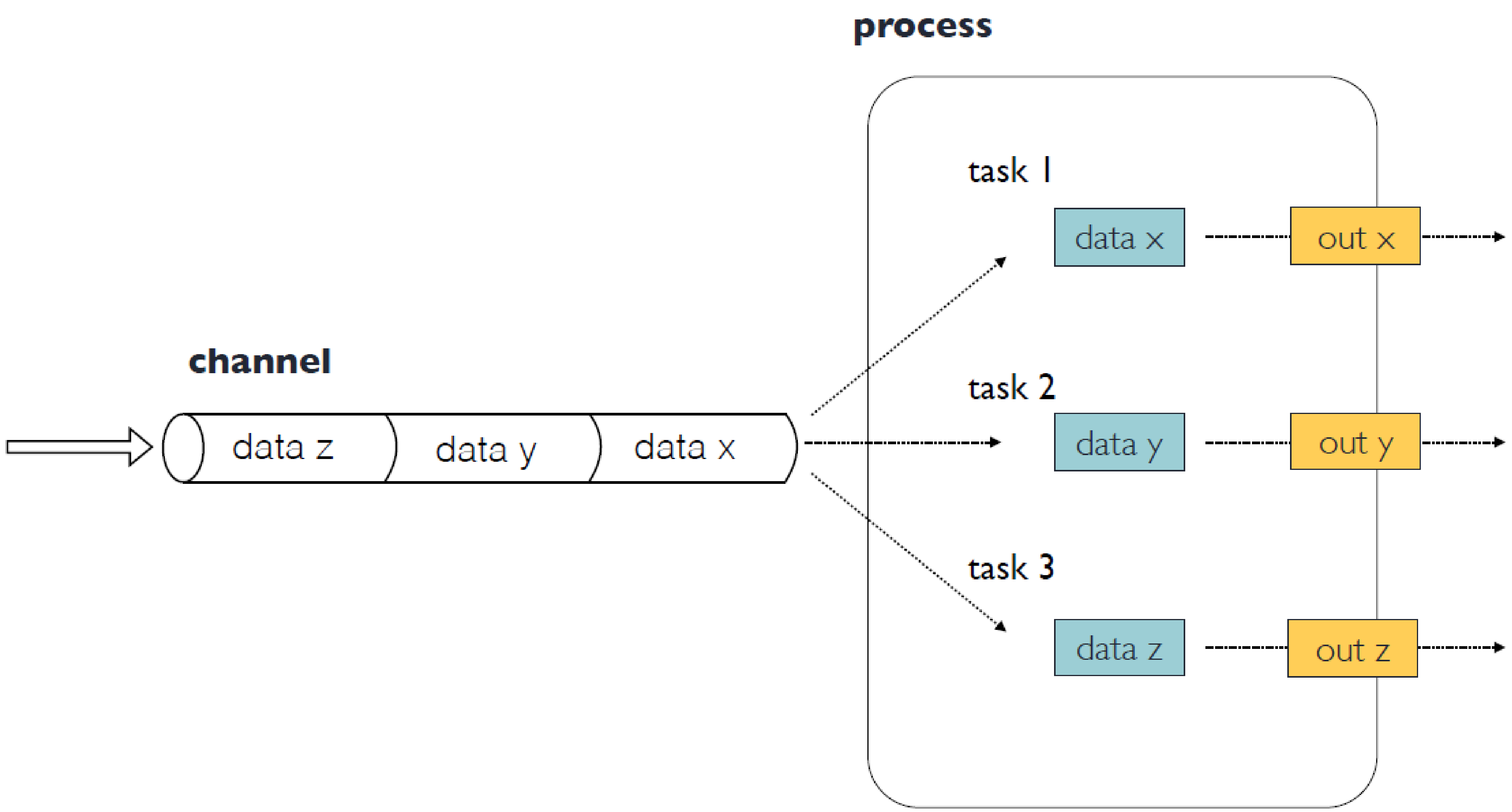
Scientists and engineers can now write complex, distributed and parallel data pipelines without requiring a degree in computer science.

Dataflow concepts

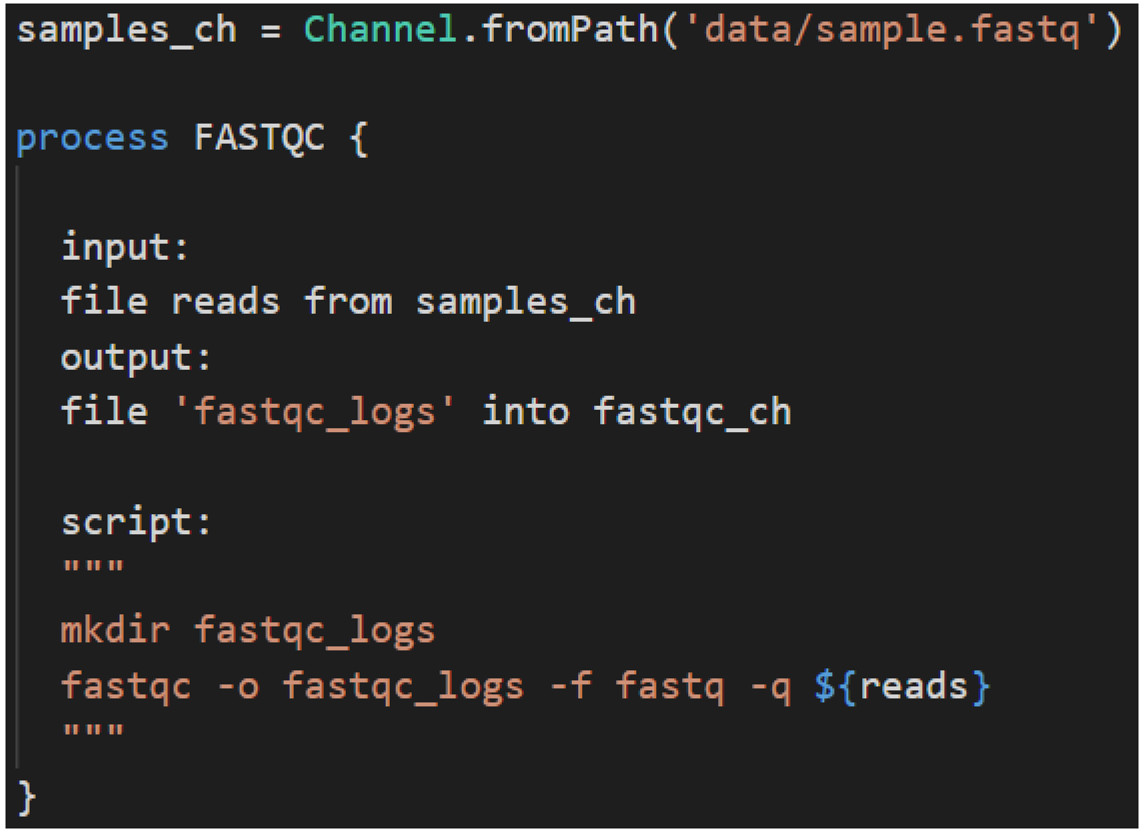
- **Declarative computational model for parallel process executions**
- **Processes wait for data, when an input set is ready the process is executed**
- **They communicate by using dataflow variables i.e., async FIFO queues called channels**
- **Parallelization and tasks dependencies are implicitly defined by process in/out declarations**



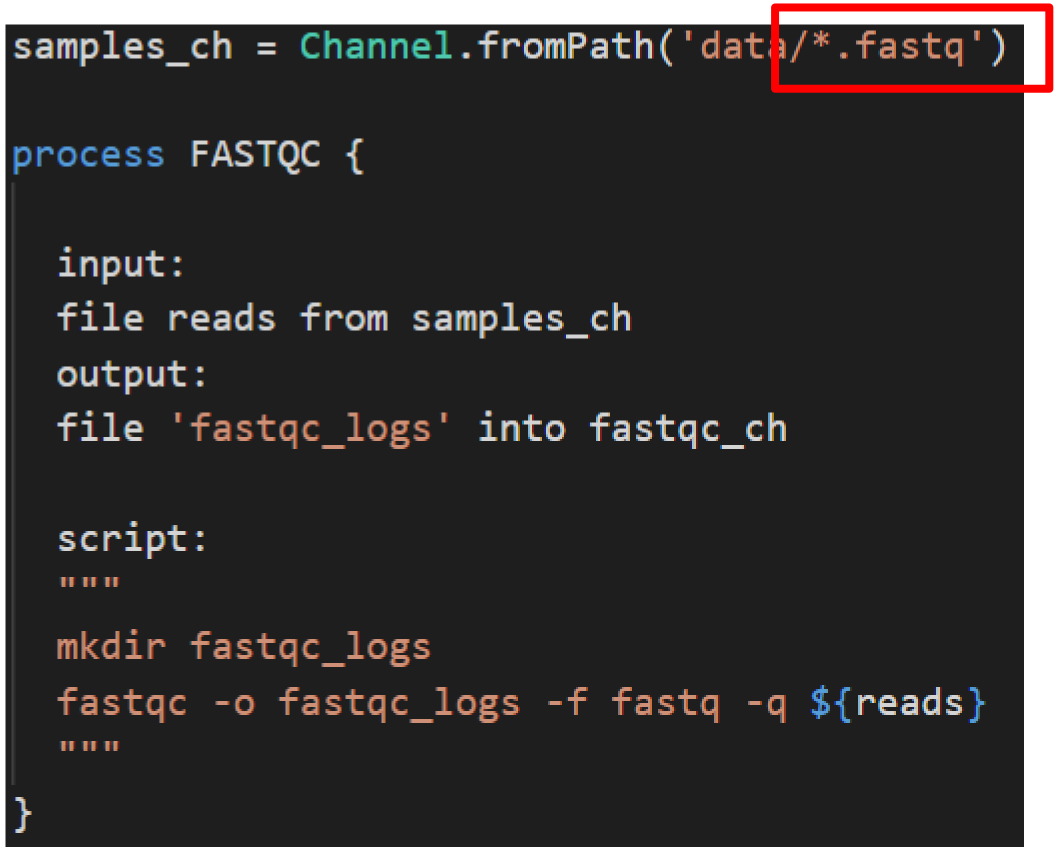
How does parallelization work?



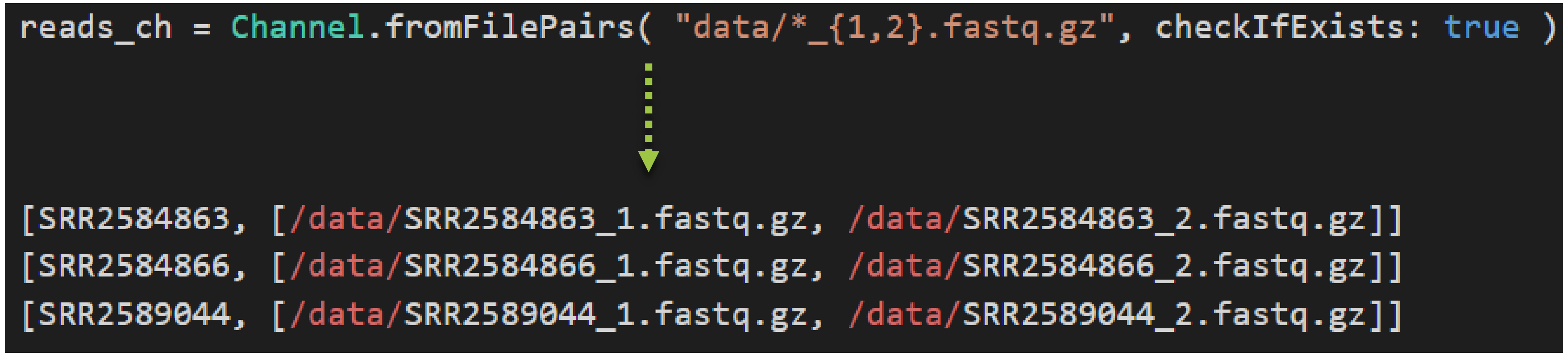
How does parallelization work?



```
samples_ch = Channel.fromPath('data/sample.fastq')

process FASTQC {

  input:
  file reads from samples_ch
  output:
  file 'fastqc_logs' into fastqc_ch

  script:
  """
  mkdir fastqc_logs
  fastqc -o fastqc_logs -f fastq -q ${reads}
  """
}
```

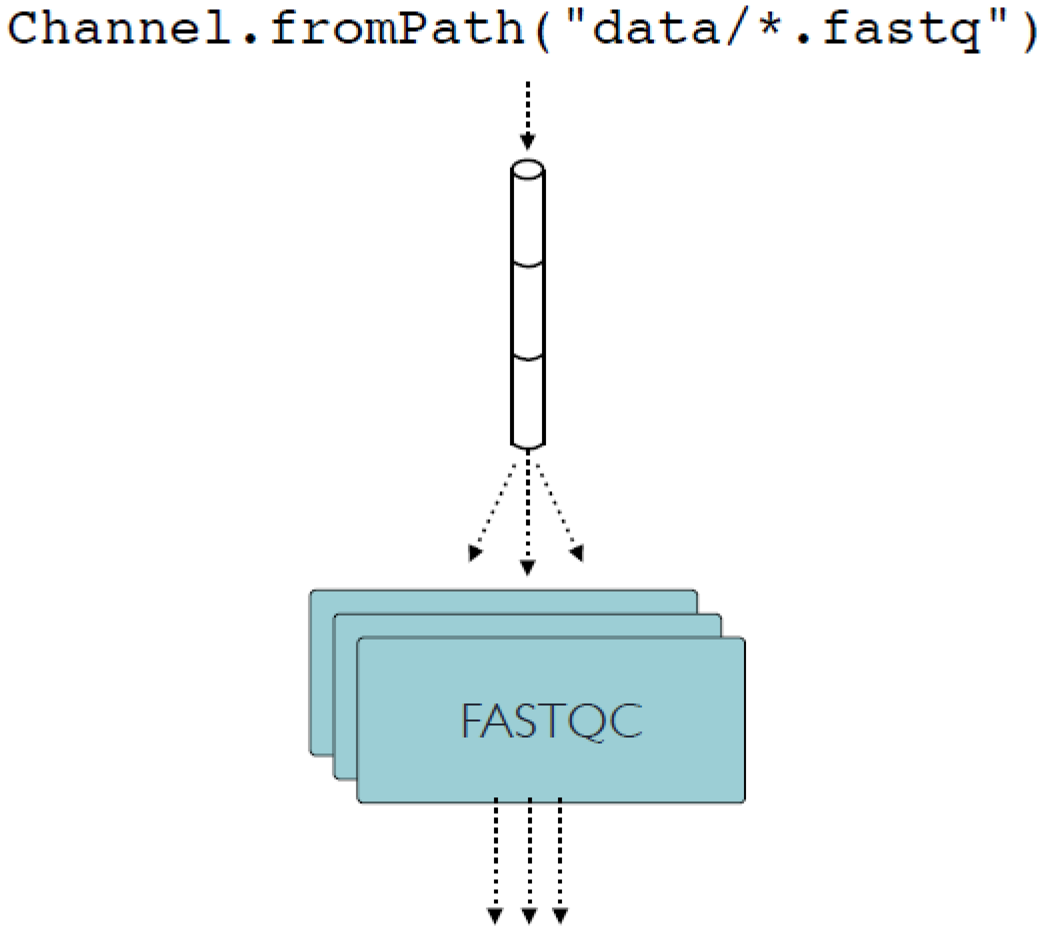


```
samples_ch = Channel.fromPath('data/*.fastq')

process FASTQC {

  input:
  file reads from samples_ch
  output:
  file 'fastqc_logs' into fastqc_ch

  script:
  """
  mkdir fastqc_logs
  fastqc -o fastqc_logs -f fastq -q ${reads}
  """
}
```



```
reads_ch = Channel.fromFilePairs( "data/*_{1,2}.fastq.gz", checkIfExists: true )

[SRR2584863, [/data/SRR2584863_1.fastq.gz, /data/SRR2584863_2.fastq.gz]]
[SRR2584866, [/data/SRR2584866_1.fastq.gz, /data/SRR2584866_2.fastq.gz]]
[SRR2589044, [/data/SRR2589044_1.fastq.gz, /data/SRR2589044_2.fastq.gz]]
```

Implicit parallelism


`Channel.fromPath("data/*.fastq")`

CWL vs Nextflow

• Language specification	• Language + app. runtime
• Declarative meta-language (YAML/JSON)	• DSL on top of a general purpose programming lang.
• Verbose	• Concise, fluent
• Committee driven	• Community driven
• Many vendors/implementations (and specification version)	• Single implementation, quick iterations

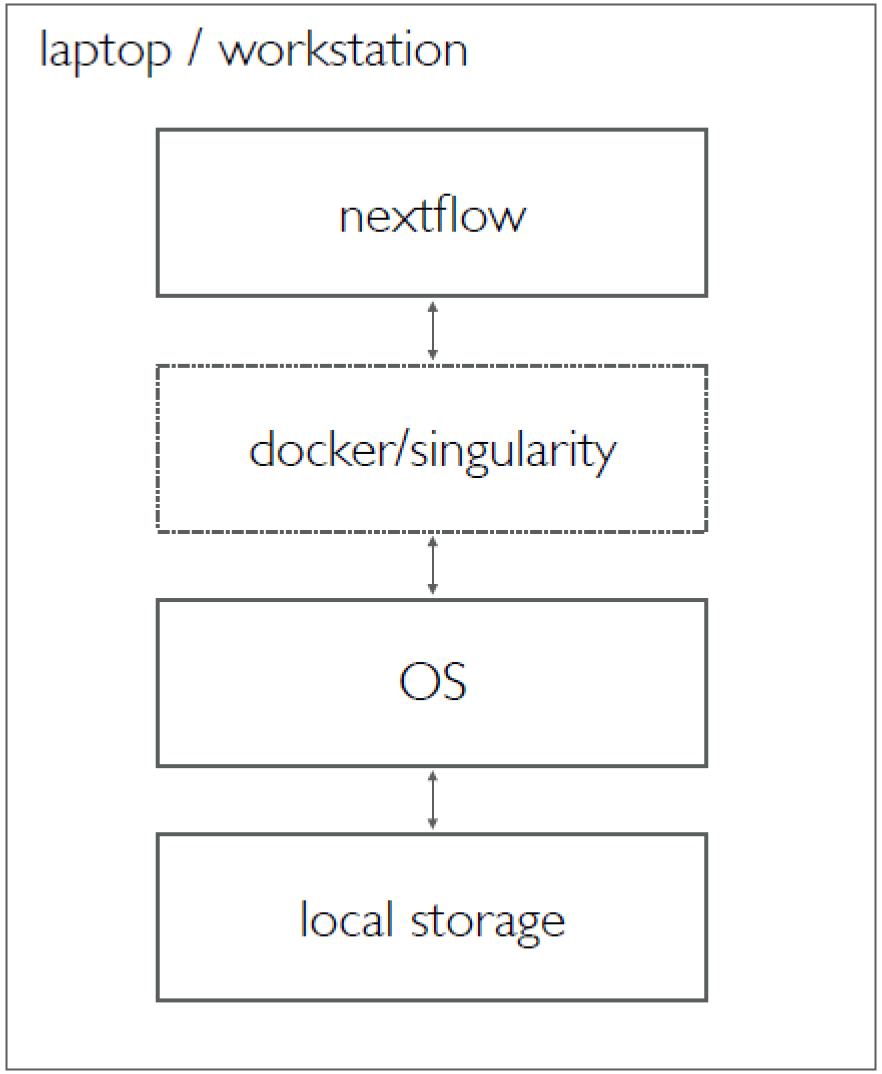
Snakemake vs Nextflow

- Command line oriented tool
- Pull model
- Rules defined using file name patterns
- Compute DAG ahead
- Built-in support for Singularity/Docker
- Custom scripts for cluster deployments
- No support for source code management system
- Python based

- Command line oriented tool
- Push model
- Can manage any data structure
- Compute DAG at runtime
- All major container runtimes
- Built-in support for clusters and cloud
- Built-in support for Git/GitHub, etc., manage pipeline revisions
- Groovy/JVM based

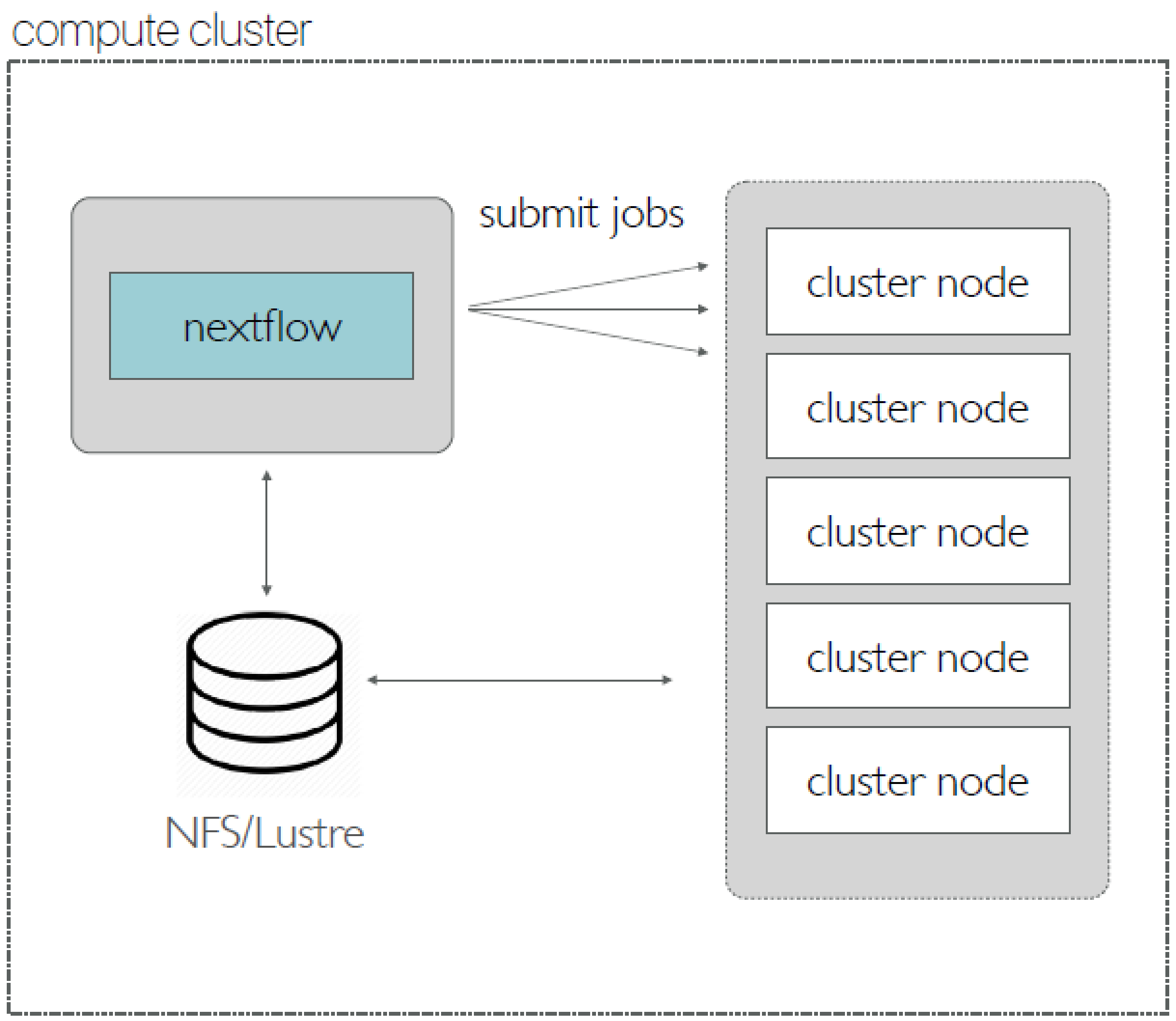
Deployment Scenarios

Local execution



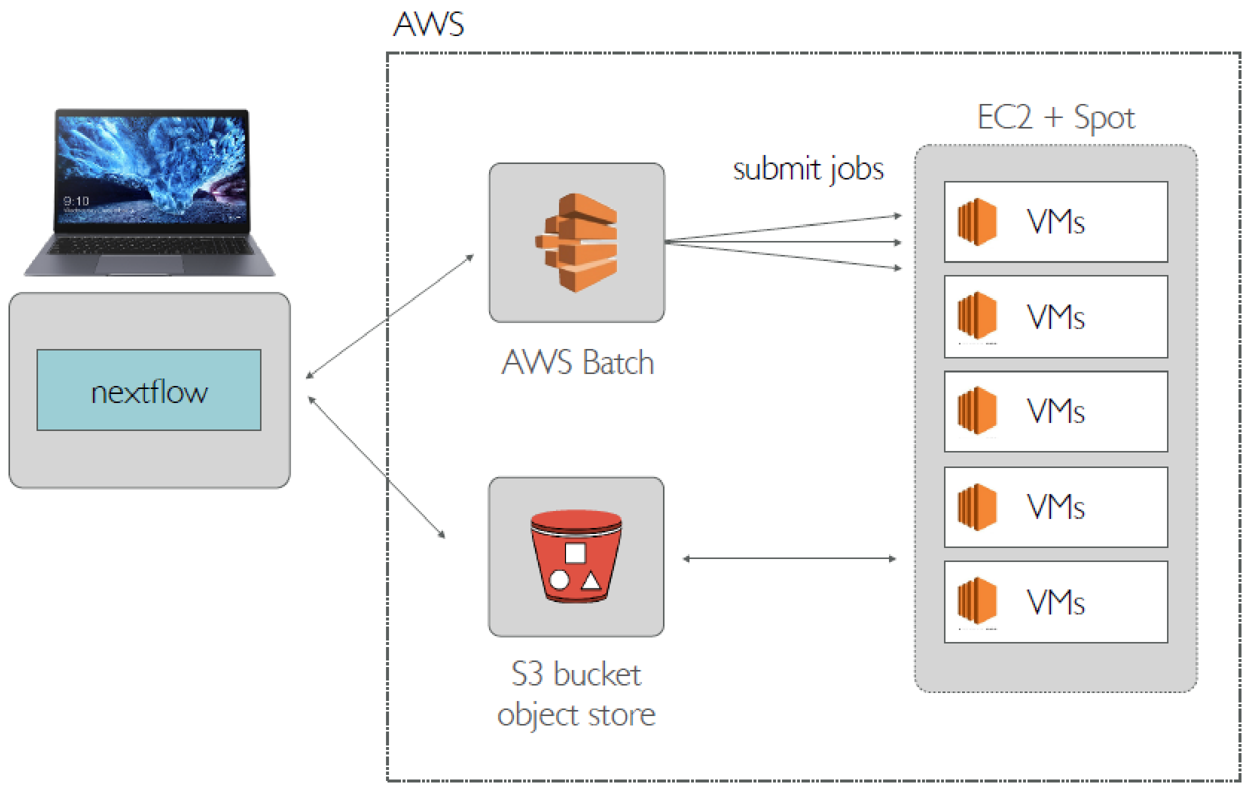
- **Common development scenario**
- **Dependencies can be managed using a container runtime**
- **Parallelization is managed by spawning posix processes**
- **Can scale vertically using fat server / shared mem. machine**

Centralized cluster orchestration



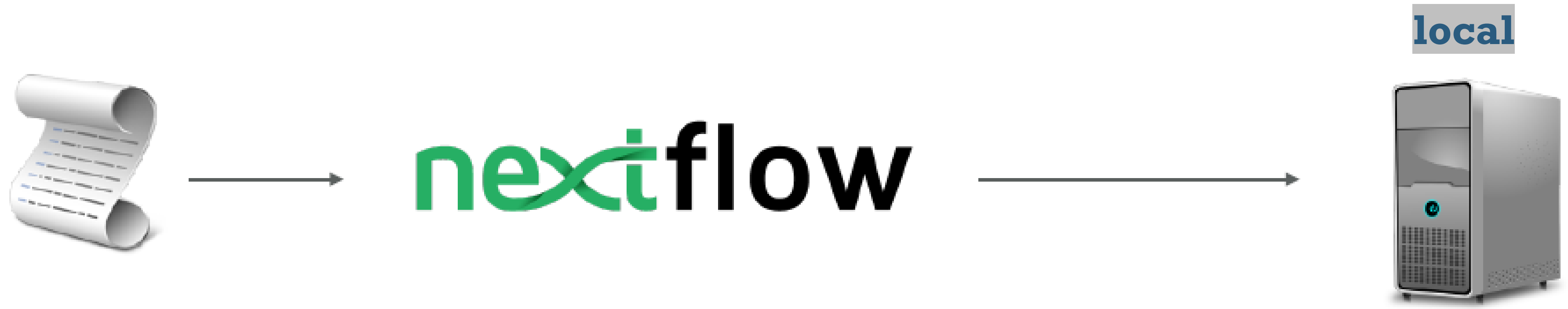
- **Nextflow orchestrates workflow execution submitting jobs to a compute scheduler.**
- **Can run in the head node or a compute node.**
- **Requires a shared storage to exchange data between tasks.**
- **Ideal for coarse-grained parallelism.**

Cloud batch orchestration

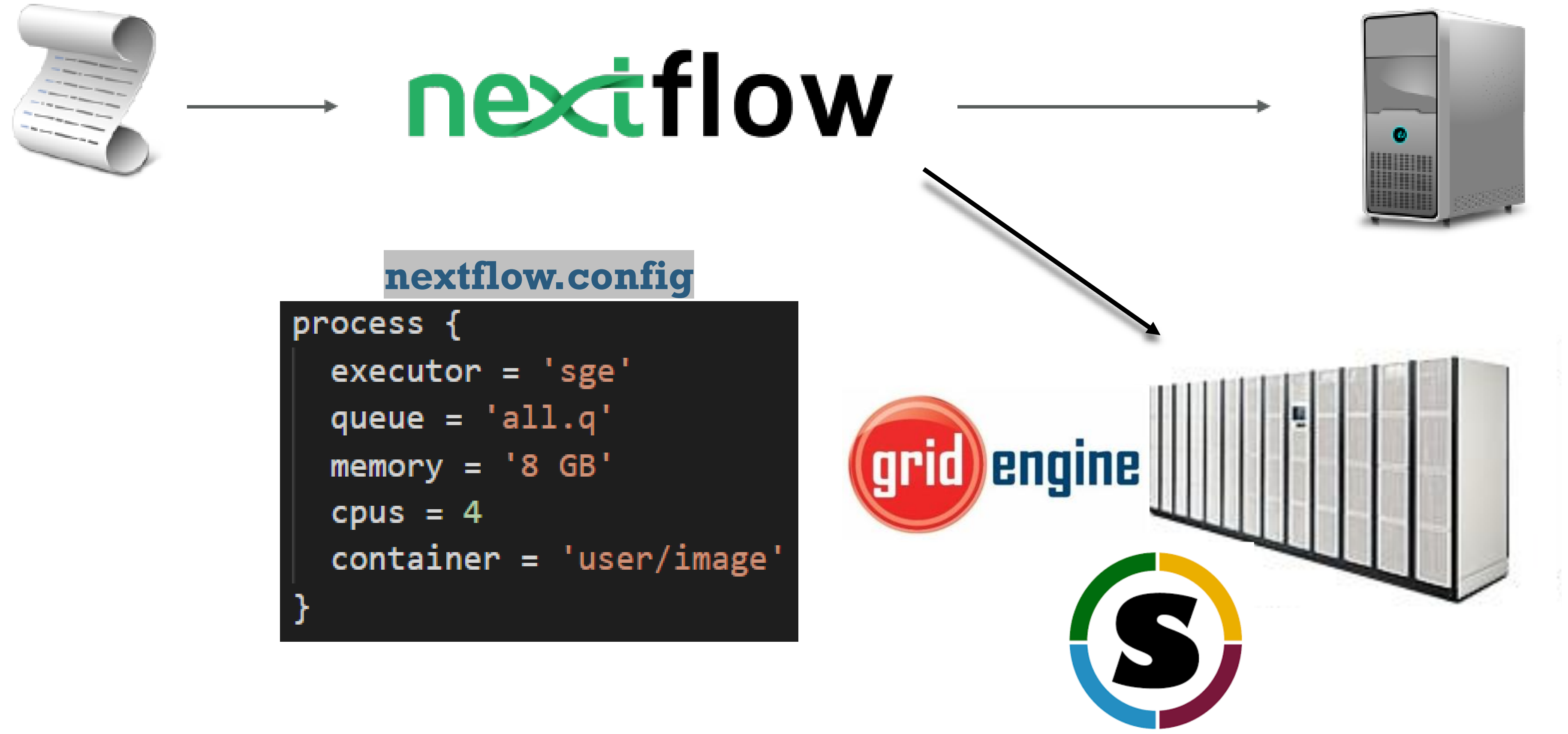


- **Nextflow orchestrates workflow execution via AWS Batch**
- **Launch workflow from anywhere into the cloud**
- **Transfer of data between local environment and cloud storage**
- **Requires a shared object storage to exchange data between VMs.**

Portability



Portability



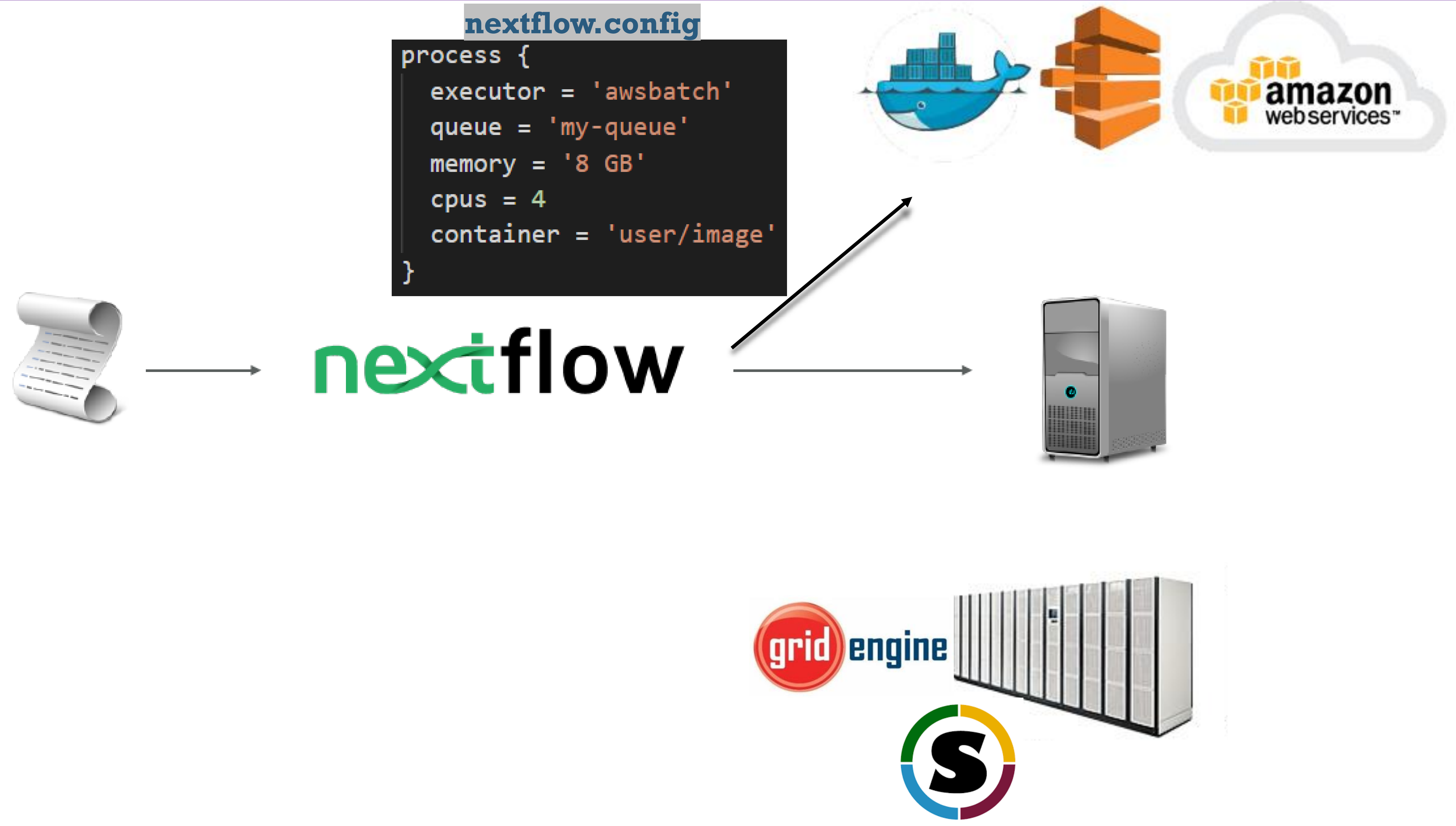
nextflow.config

```
process {
  executor = 'sge'
  queue = 'all.q'
  memory = '8 GB'
  cpus = 4
  container = 'user/image'
}
```

Portability

nextflow.config

```
process {
  executor = 'awsbatch'
  queue = 'my-queue'
  memory = '8 GB'
  cpus = 4
  container = 'user/image'
}
```



Configuration Decoupling
is the key to
portable deployments

Widespread enterprise adoption

AstraZeneca

Roche

FRED HUTCH
CURES START HERE™

Johnson & Johnson

Boehringer
Ingelheim

THE FRANCIS CRICK INSTITUTE

EMBL-EBI

wellcome sanger institute

WuXiNextCODE
Leading Genomics

Institut Pasteur

IQVIA™

UCSF
University of California
San Francisco

FABRIC GENOMICS™

illumina®

SciLifeLab

CHAN ZUCKERBERG
BIOHUB

enterome
bioscience

BASECLEAR

Karolinska
Institutet

CANCER
RESEARCH
UK

International Agency
Research on Cancer
World Health
Organization

IMP
Research Institute of Molecular Pathology

Agency for
Science, Technology
and Research
SINGAPORE

HARVARD
UNIVERSITY

gmi genomics medicine
ireland

SOPHiA™

JGI
JOINT GENOME INSTITUTE
DEPARTMENT OF ENERGY

BENAROYA
RESEARCH INSTITUTE
Virginia Mason

Weill Cornell
Medicine

ACHILLES
THERAPEUTICS

PAWSEY
supercomputing centre

VIR

SYNTHETIC GENOMICS

institutCurie

QBiC
Quantitative
Biology
Center

Berkeley
UNIVERSITY OF CALIFORNIA

Kintai
THERAPEUTICS

gritstone
ONCOLOGY

UTSouthwestern
Medical Center

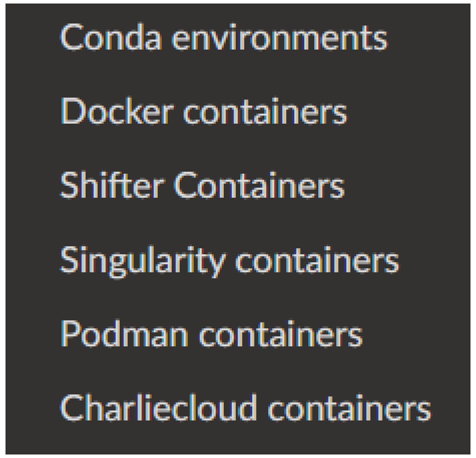
University of Colorado
Boulder

READCOOR

Container vs. VM

- Lighter: MB vs GB
- Faster startup: ms/secs vs minutes
- Virtualize a process/application instead of OS/Hardware
- Immutable: don't change over time, thus guarantee replicability over executions.
- Composable: the output of one container is directly consumable as input by another container.
- Transparent: they are created with a well-defined automated procedure.

Nextflow Supports:



Conda environments

Docker containers

Shifter Containers

Singularity containers

Podman containers

Charliecloud containers

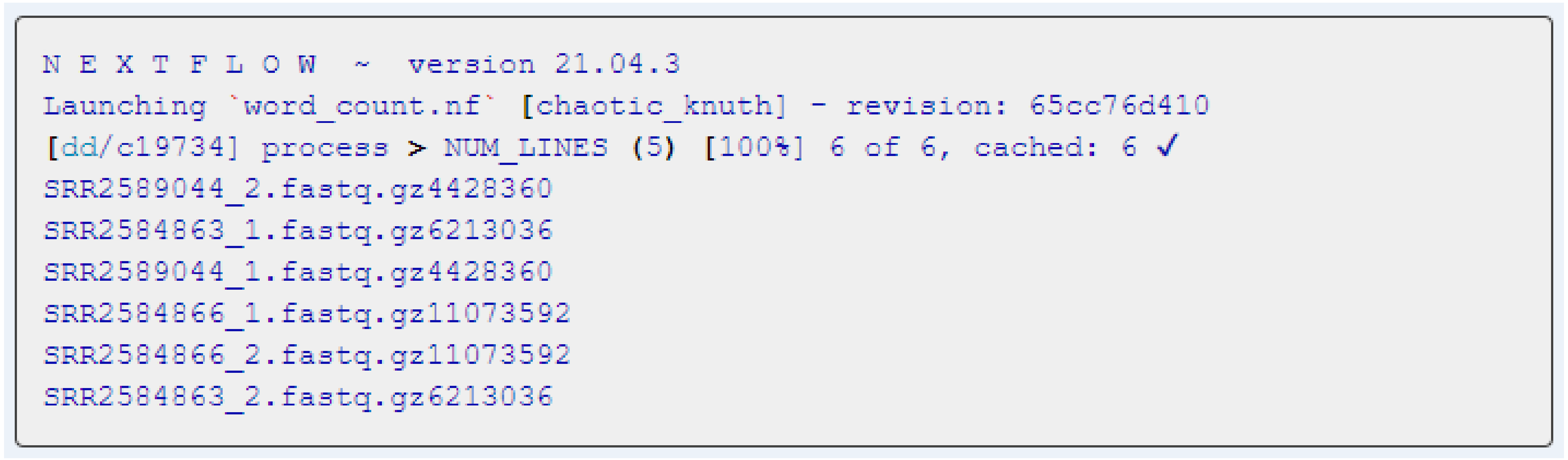
When use containers?

Always!

Caching and Checkpointing

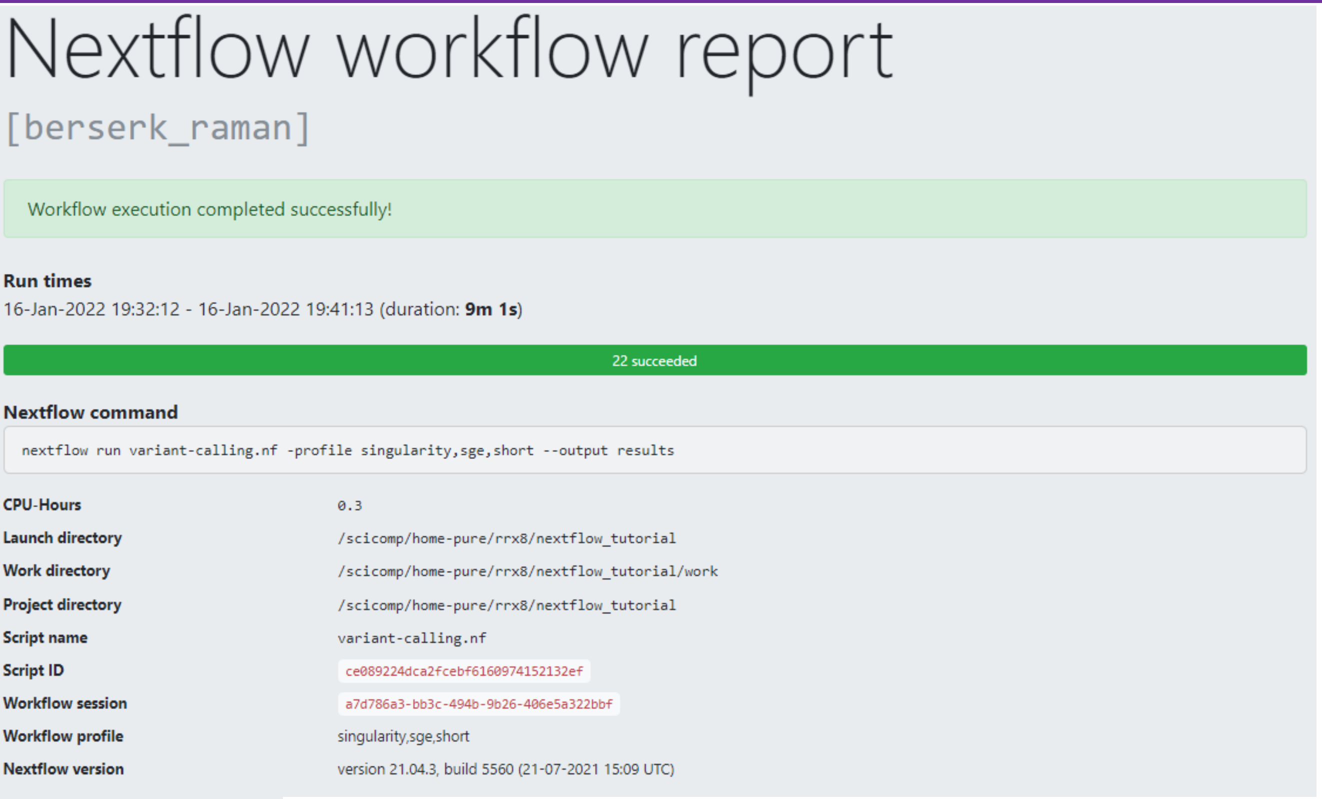
A key feature of Nextflow is **re-entrancy** which is the ability to restart a pipeline after an error from the last successful process.

```
nextflow run word_count.nf --input 'data/untrimmed_fastq/*.fastq.gz' -resume
```



```
N E X T F L O W  ~  version 21.04.3
Launching `word_count.nf` [chaotic_knuth] - revision: 65cc76d410
[dd/c19734] process > NUM_LINES (5) [100%] 6 of 6, cached: 6 ✓
SRR2589044_2.fastq.gz4428360
SRR2584863_1.fastq.gz6213036
SRR2589044_1.fastq.gz4428360
SRR2584866_1.fastq.gz11073592
SRR2584866_2.fastq.gz11073592
SRR2584863_2.fastq.gz6213036
```

Reporting



Nextflow workflow report

[berserk_raman]

Workflow execution completed successfully!

Run times

16-Jan-2022 19:32:12 - 16-Jan-2022 19:41:13 (duration: **9m 1s**)

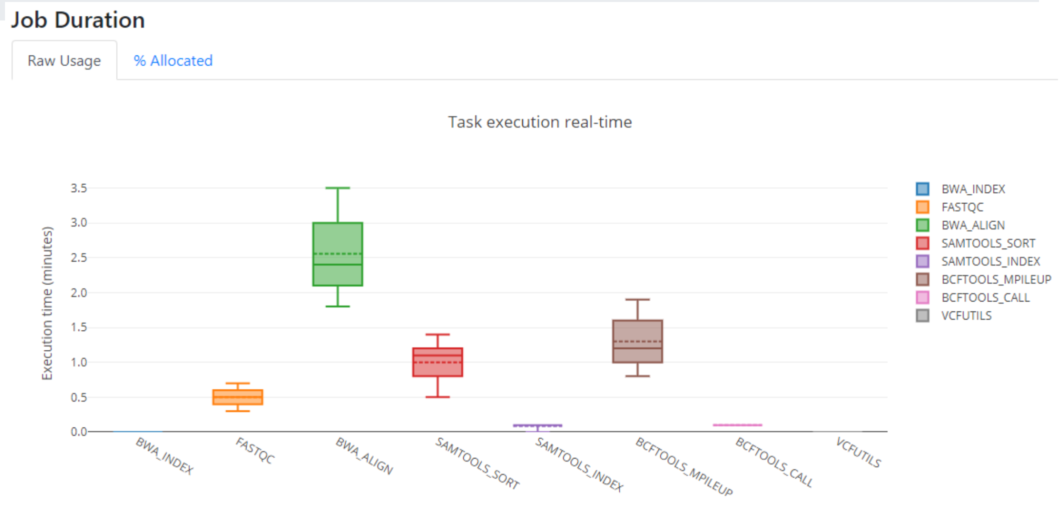
22 succeeded

Nextflow command

```
nextflow run variant-calling.nf -profile singularity,sge,short --output results
```

CPU-Hours	0.3
Launch directory	/scicomp/home-pure/rrx8/nextflow_tutorial
Work directory	/scicomp/home-pure/rrx8/nextflow_tutorial/work
Project directory	/scicomp/home-pure/rrx8/nextflow_tutorial
Script name	variant-calling.nf
Script ID	ce089224dca2fcebf6160974152132ef
Workflow session	a7d786a3-bb3c-494b-9b26-406e5a322bbf
Workflow profile	singularity,sge,short
Nextflow version	version 21.04.3, build 5560 (21-07-2021 15:09 UTC)

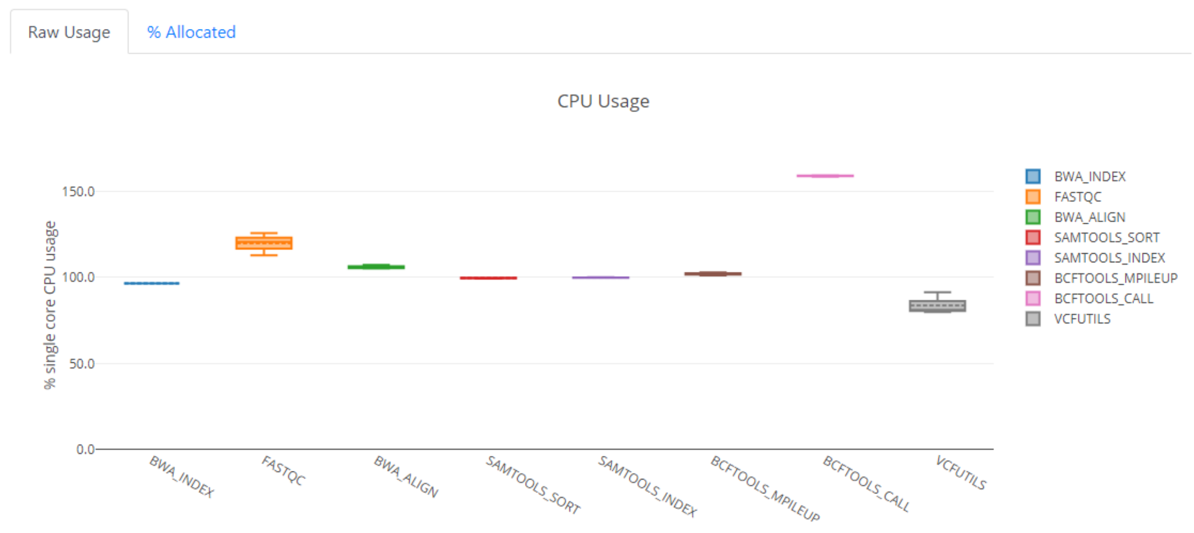
Job Duration



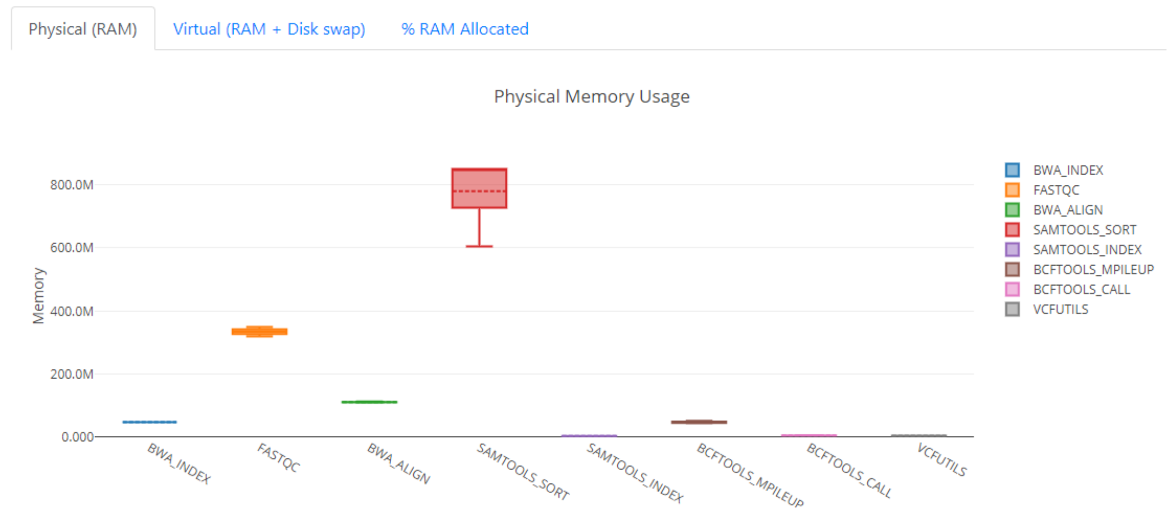
Resource Usage

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

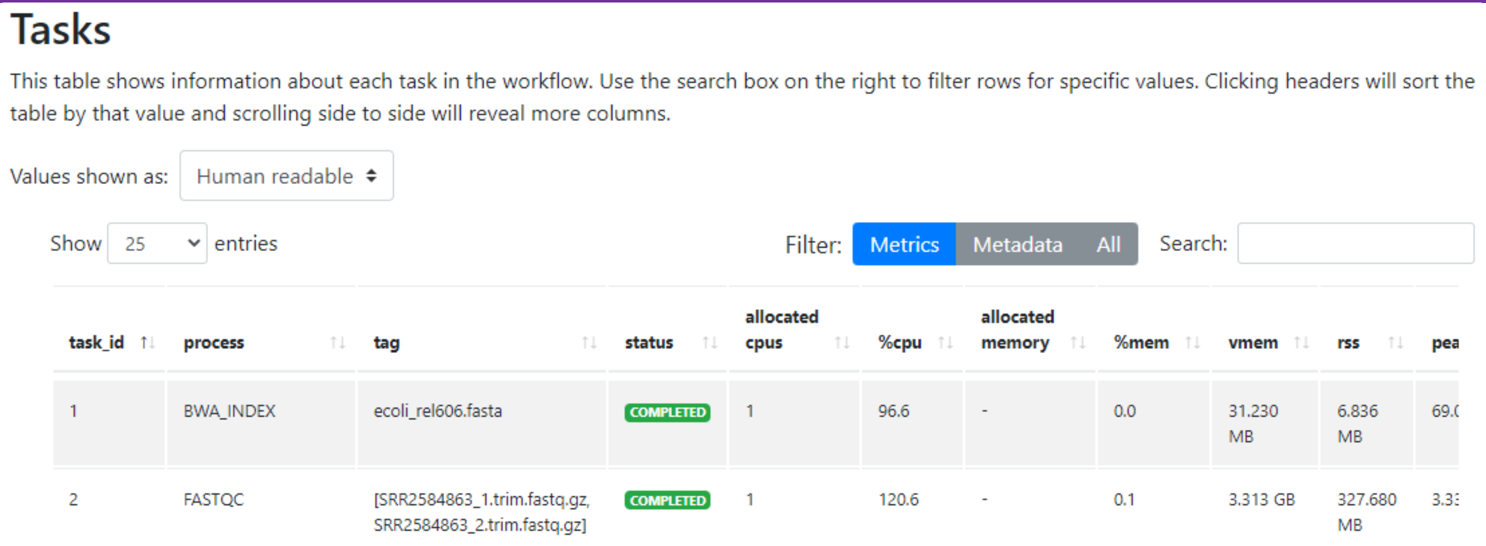
CPU



Memory



Reporting



Tasks

This table shows information about each task in the workflow. Use the search box on the right to filter rows for specific values. Clicking headers will sort the table by that value and scrolling side to side will reveal more columns.

Values shown as: Human readable

Show 25 entries

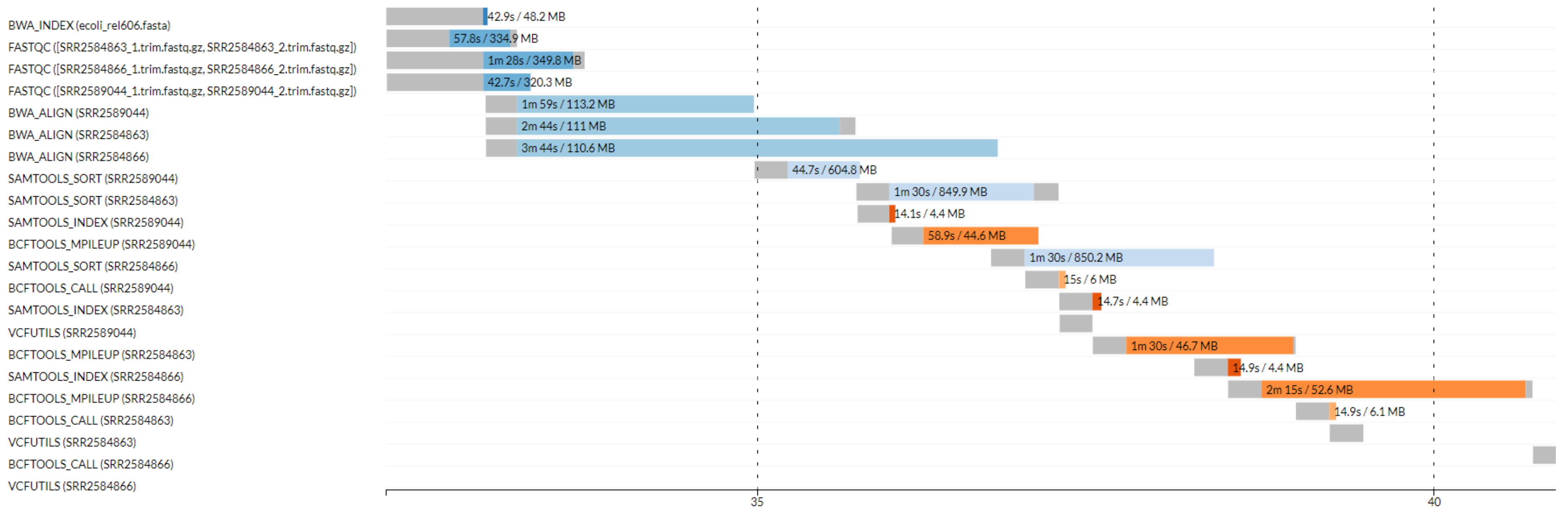
Filter: Metrics Metadata All

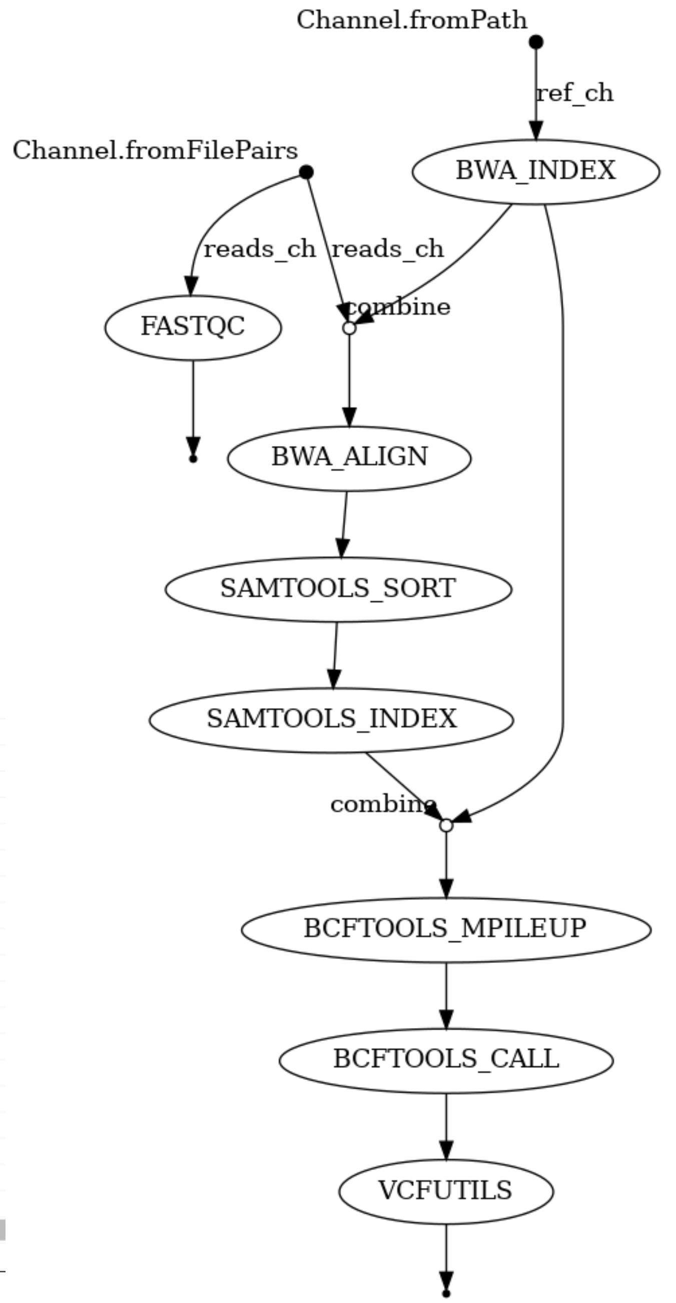
Search:

task_id	process	tag	status	allocated cpus	%cpu	allocated memory	%mem	vmem	rss	pea
1	BWA_INDEX	ecoli_rel606.fasta	COMPLETED	1	96.6	-	0.0	31.230 MB	6.836 MB	69.(
2	FASTQC	[SRR2584863_1.trim.fastq.gz, SRR2584863_2.trim.fastq.gz]	COMPLETED	1	120.6	-	0.1	3.313 GB	327.680 MB	3.3:

Processes execution timeline

Launch time: 16 Jan 2022 19:32
Elapsed time: 9m 2s
Legend: job wall time / memory usage (RAM)





https://nf-co.re/

nf-core

A community effort to collect a curated set of analysis pipelines built using Nextflow.

Open-Source Community

55K+

monthly downloads

10,000+

active developers /month

35

international Workshops

150K+

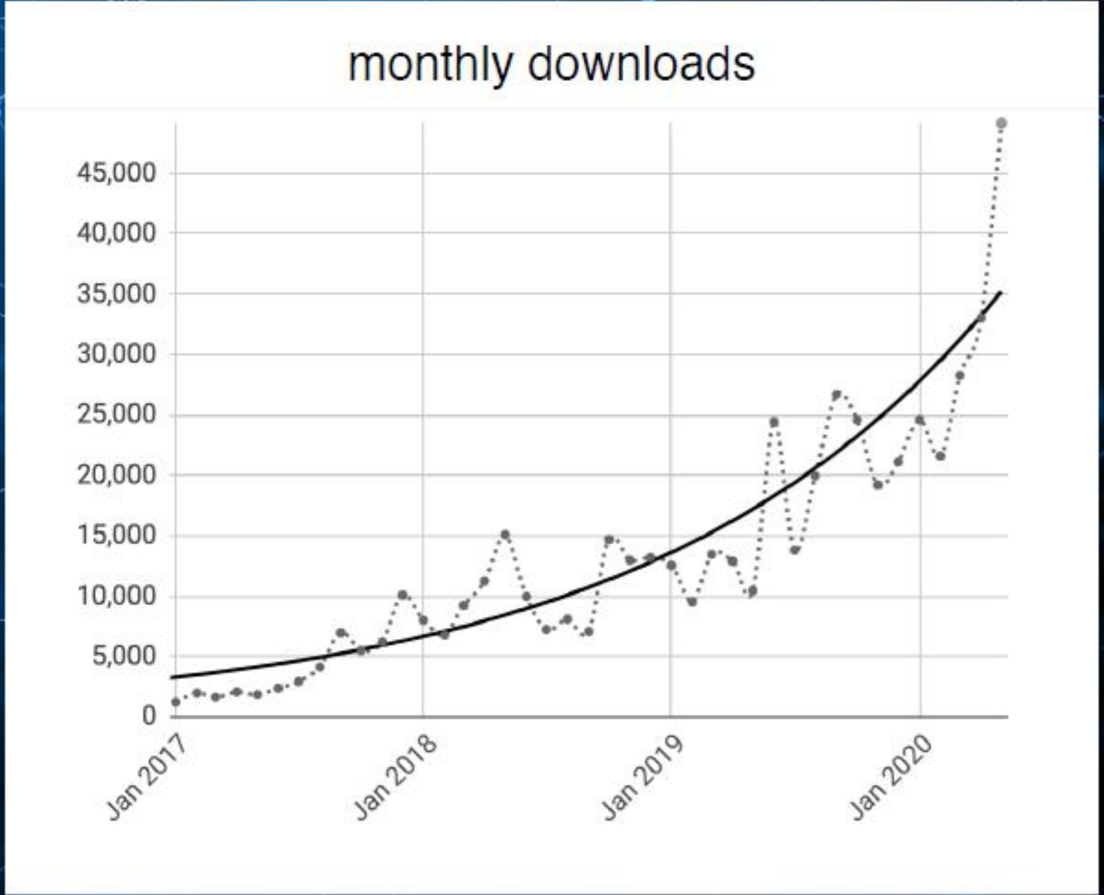
lines of code

1.6K+

stars on github

100+

contributors

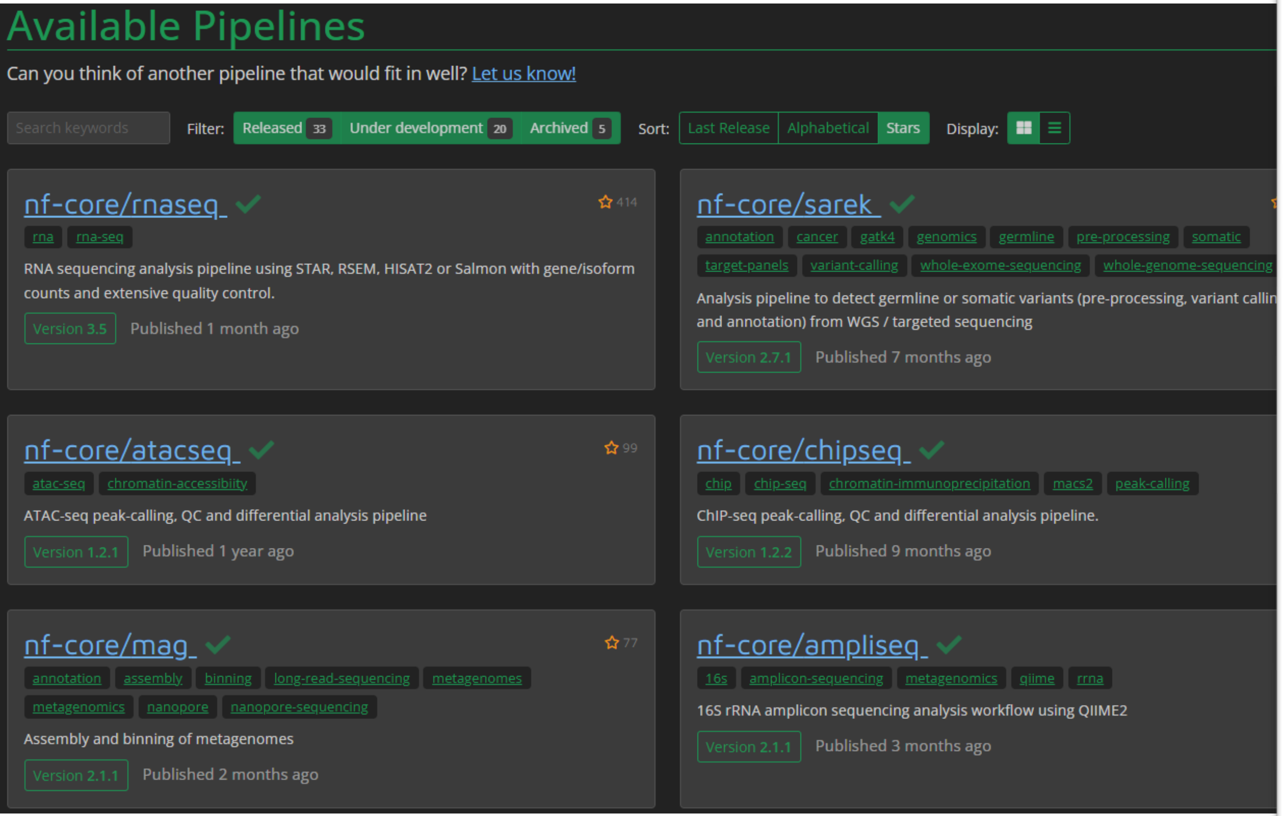


Over 2 million downloads

nf-core

Growing community content

Production ready analysis pipelines built with Nextflow.



Available Pipelines

Can you think of another pipeline that would fit in well? Let us know!

Search keywords | Filter: Released 33 | Under development 20 | Archived 5 | Sort: Last Release | Alphabetical | Stars | Display:

nf-core/rnaseq ✓ ☆ 414
rna · rna-seq
RNA sequencing analysis pipeline using STAR, RSEM, HISAT2 or Salmon with gene/isoform counts and extensive quality control.
Version 3.5 · Published 1 month ago

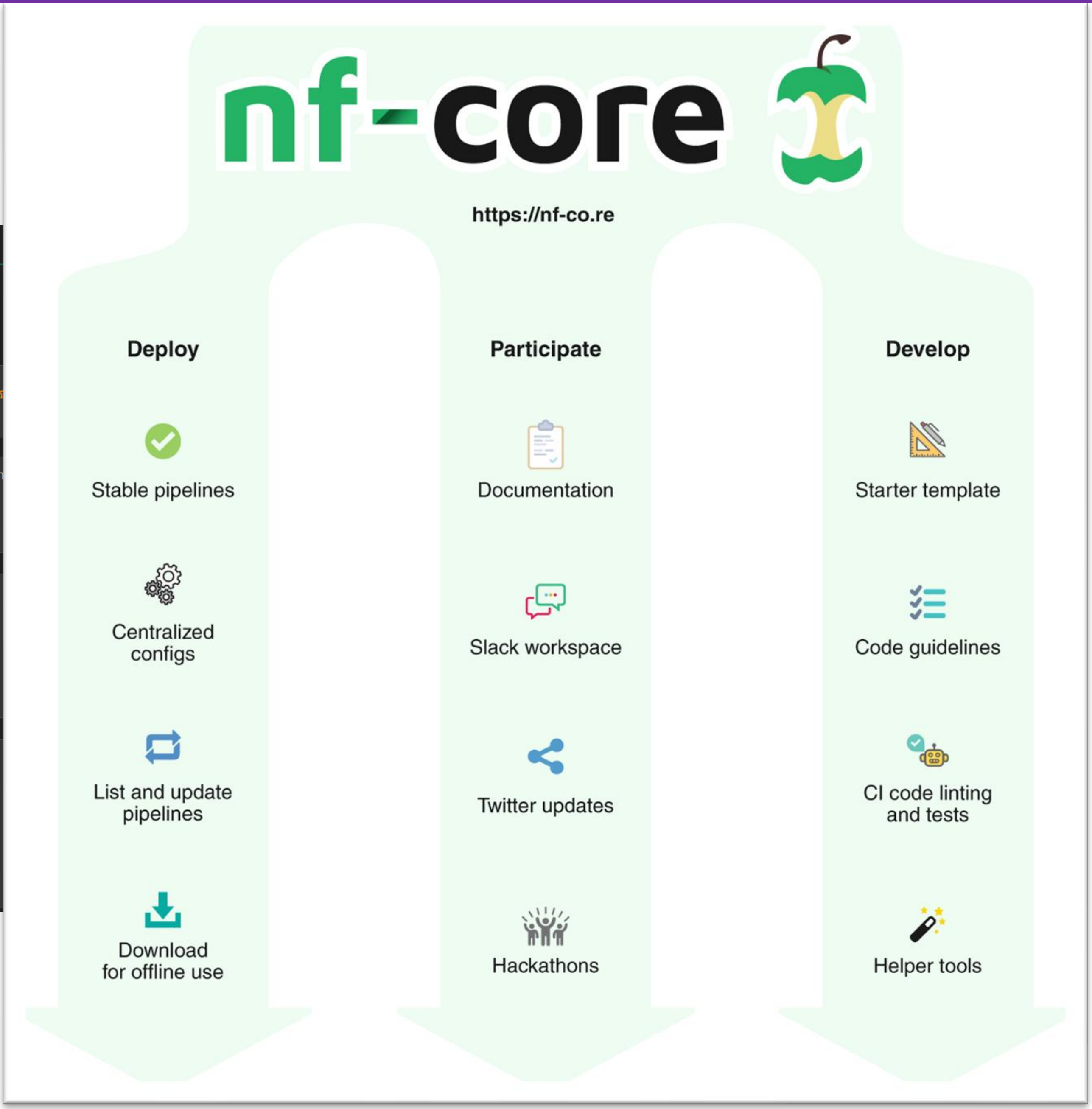
nf-core/sarek ✓
annotation · cancer · gatk4 · genomics · germline · pre-processing · somatic · target-panels · variant-calling · whole-exome-sequencing · whole-genome-sequencing
Analysis pipeline to detect germline or somatic variants (pre-processing, variant callin and annotation) from WGS / targeted sequencing
Version 2.7.1 · Published 7 months ago

nf-core/atacseq ✓ ☆ 99
atac-seq · chromatin-accessibility
ATAC-seq peak-calling, QC and differential analysis pipeline
Version 1.2.1 · Published 1 year ago

nf-core/chipseq ✓
chip · chip-seq · chromatin-immunoprecipitation · macs2 · peak-calling
ChIP-seq peak-calling, QC and differential analysis pipeline.
Version 1.2.2 · Published 9 months ago

nf-core/mag ✓ ☆ 77
annotation · assembly · binning · long-read-sequencing · metagenomes · metagenomics · nanopore · nanopore-sequencing
Assembly and binning of metagenomes
Version 2.1.1 · Published 2 months ago

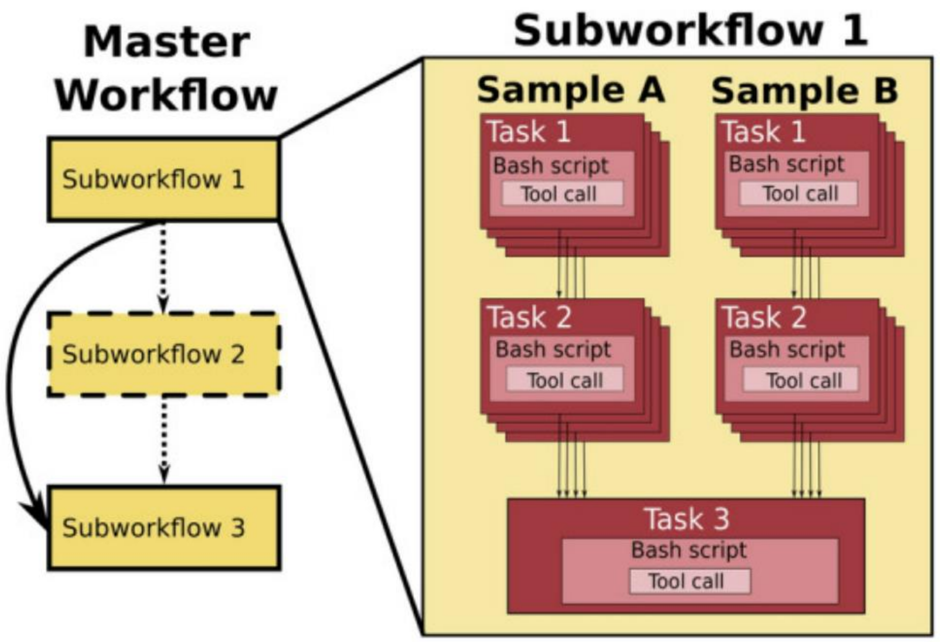
nf-core/ampliseq ✓
16s · amplicon-sequencing · metagenomics · qiime · rrna
16S rRNA amplicon sequencing analysis workflow using QIIME2
Version 2.1.1 · Published 3 months ago

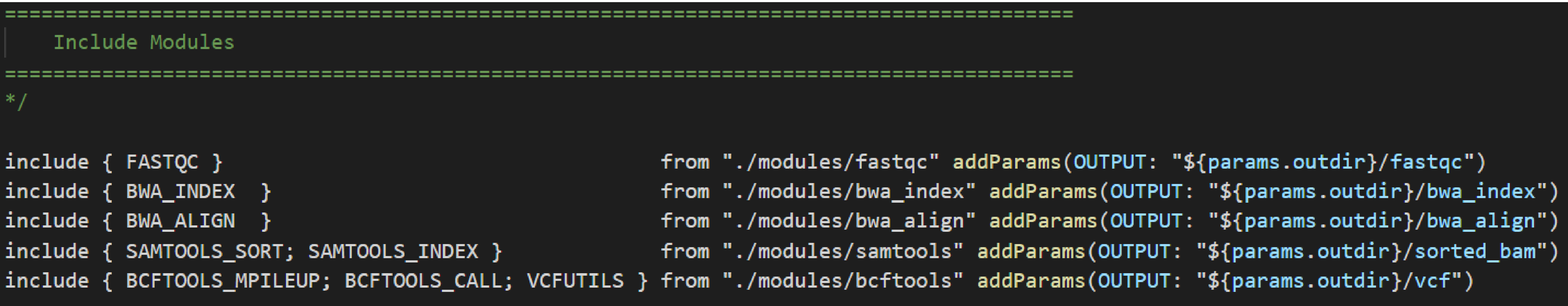


DSL2 Modules & Sub-Workflows = Re-Usability!

- Modularity is a very important design principle for production bioinformatics workflows.
- The core idea is to build a library of reusable modules (tasks or sub-workflows) and assemble them into various master workflows.

This enables -
(1) performing different analyses without having to refactor the entire workflow
(2) check-pointing and restart of a workflow run from a task in the middle of analysis if needed
(3) customizing runtime environments and compute resources which may vary between analysis stages.





```
========================================================================================
    Include Modules
========================================================================================
*/

include { FASTQC }                                     from "./modules/fastqc" addParams(OUTPUT: "${params.outdir}/fastqc")
include { BWA_INDEX  }                                 from "./modules/bwa_index" addParams(OUTPUT: "${params.outdir}/bwa_index")
include { BWA_ALIGN  }                                 from "./modules/bwa_align" addParams(OUTPUT: "${params.outdir}/bwa_align")
include { SAMTOOLS_SORT; SAMTOOLS_INDEX }              from "./modules/samtools" addParams(OUTPUT: "${params.outdir}/sorted_bam")
include { BCFTOOLS_MPILEUP; BCFTOOLS_CALL; VCFUTILS } from "./modules/bcftools" addParams(OUTPUT: "${params.outdir}/vcf")
```