ZS Discovery

Elucidate. Innovate. Accelerate.

Nextflow fundamentals training

DTU – Biosustain

Albert and Felipe - 2025 Nov, 12

This document is solely for the use of ZS personnel. No part of it may be circulated, quoted or reproduced for distribution outside of ZS without prior written approval of ZS.

Evanston +1 847 492 3600

What will we cover?

- Who am I?
- What is Nextflow?
- Understanding the tool
  - Basic concepts
  - Core features
- Behind Nextflow
  - Why is a community needed?
  - The community efforts
- Use of the tool in Industry

Why are we here today?

Big Data

Experiments and datasets only get bigger and bigger

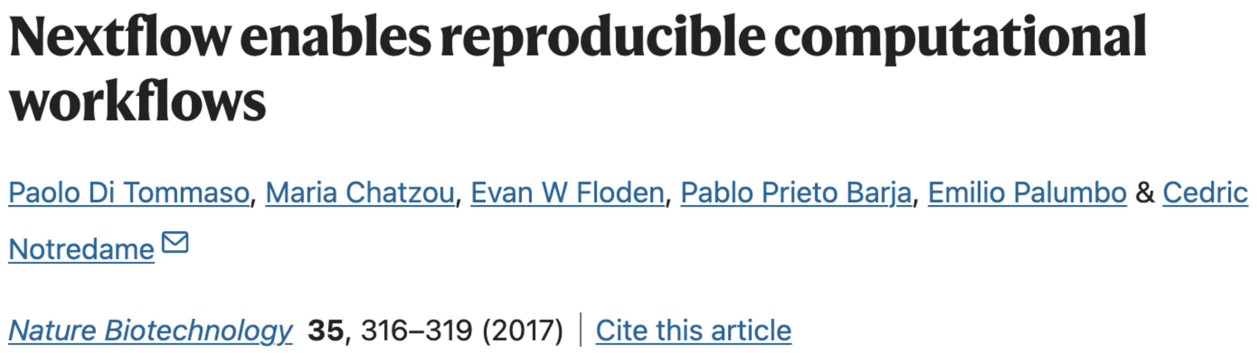
Scalability

With this, scalability becomes a bottleneck

Reproducibility

At the end, analyses must still be reproducible anywhere

That is where Nextflow comes



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

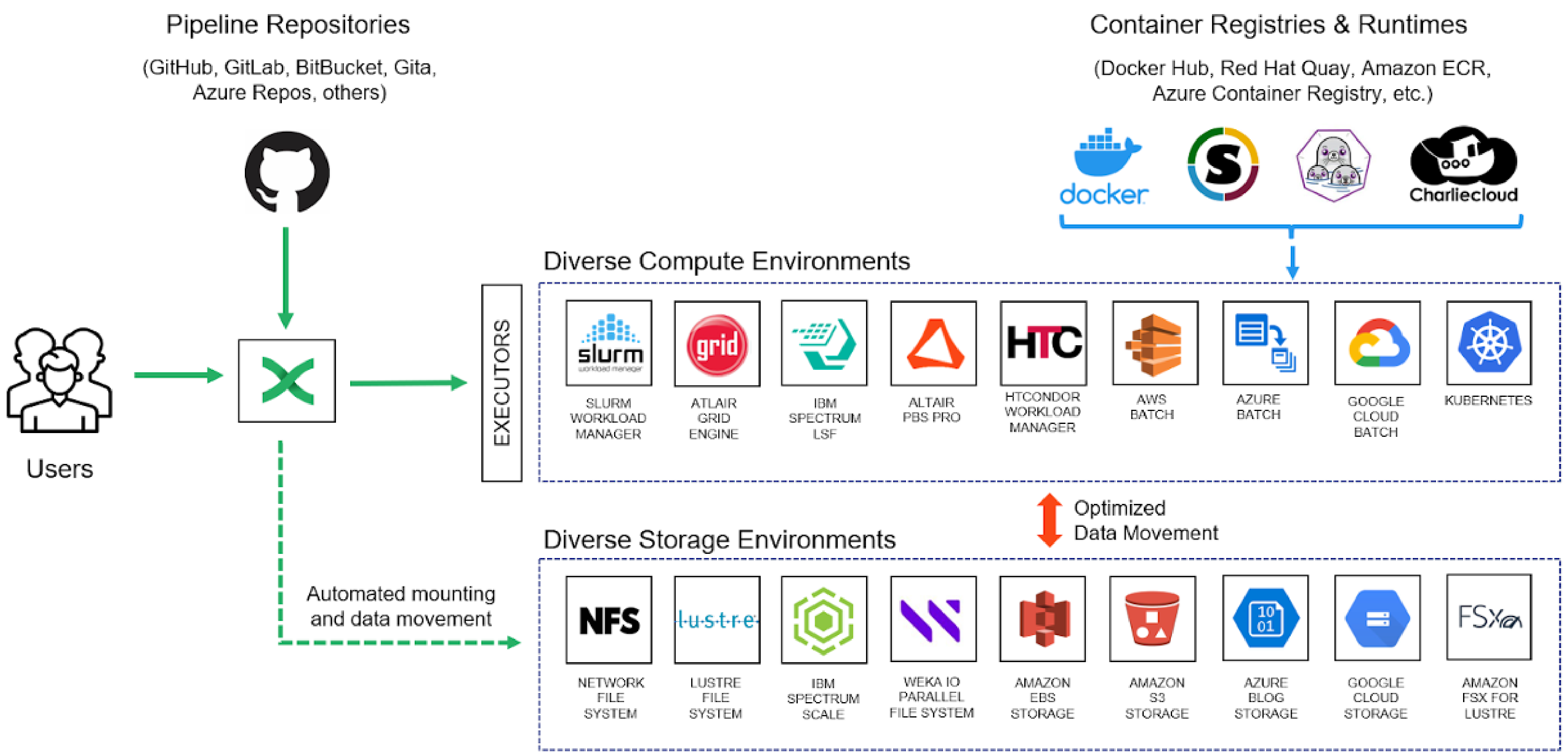
Nextflow addresses these challenges, allowing us to build efficient workflows coupled with strong supportive community behind.

What is Nextflow?

In summary

- Nextflow is a **software** – Workflow orchestrator engine
- It is a Domain-specific language (DSL) built on top of Groovy
- Workflow orchestrator engine:
  - It “wraps-up” tasks as submission scripts for different computing environments and queuing systems
  - Allows use of preferred programing languages for tasks

What does it mean in practice?



Pipeline Repositories

(GitHub, GitLab, BitBucket, Gita, Azure Repos, others)

Container Registries & Runtimes

(Docker Hub, Red Hat Quay, Amazon ECR, Azure Container Registry, etc.)

Users

EXECUTORS

Diverse Compute Environments

SLURM WORKLOAD MANAGER

ATLAIR GRID ENGINE

IBM SPECTRUM LSF

ALTAIR PBS PRO

HTCONDOR WORKLOAD MANAGER

AWS BATCH

AZURE BATCH

GOOGLE CLOUD BATCH

KUBERNETES

Optimized Data Movement

Diverse Storage Environments

Automated mounting and data movement

NETWORK FILE SYSTEM

LUSTRE FILE SYSTEM

IBM SPECTRUM SCALE

WEKA IO PARALLEL FILE SYSTEM

AMAZON EBS STORAGE

AMAZON S3 STORAGE

AZURE BLOG STORAGE

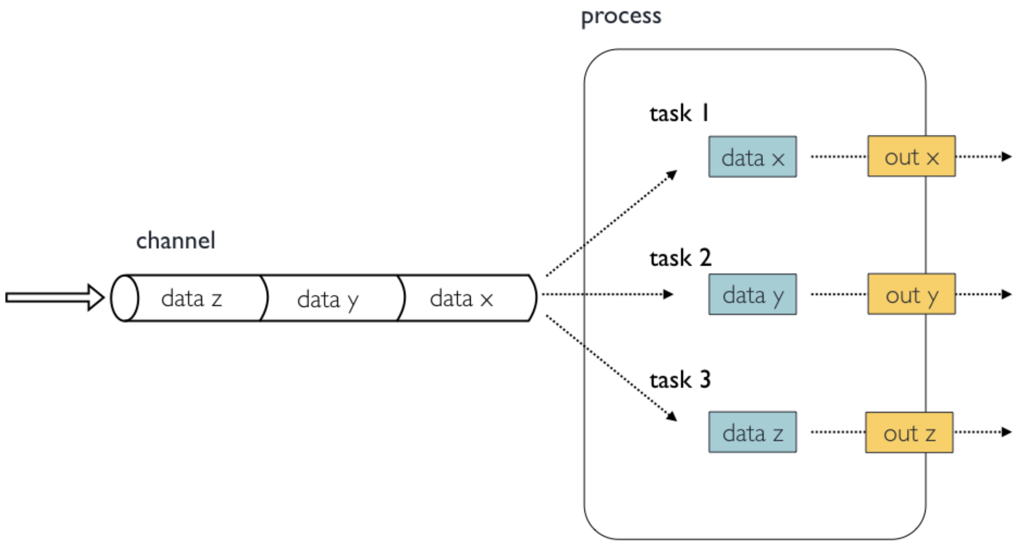
GOOGLE CLOUD STORAGE

AMAZON FSX FOR LUSTRE

A bit about the tool before the community and its use

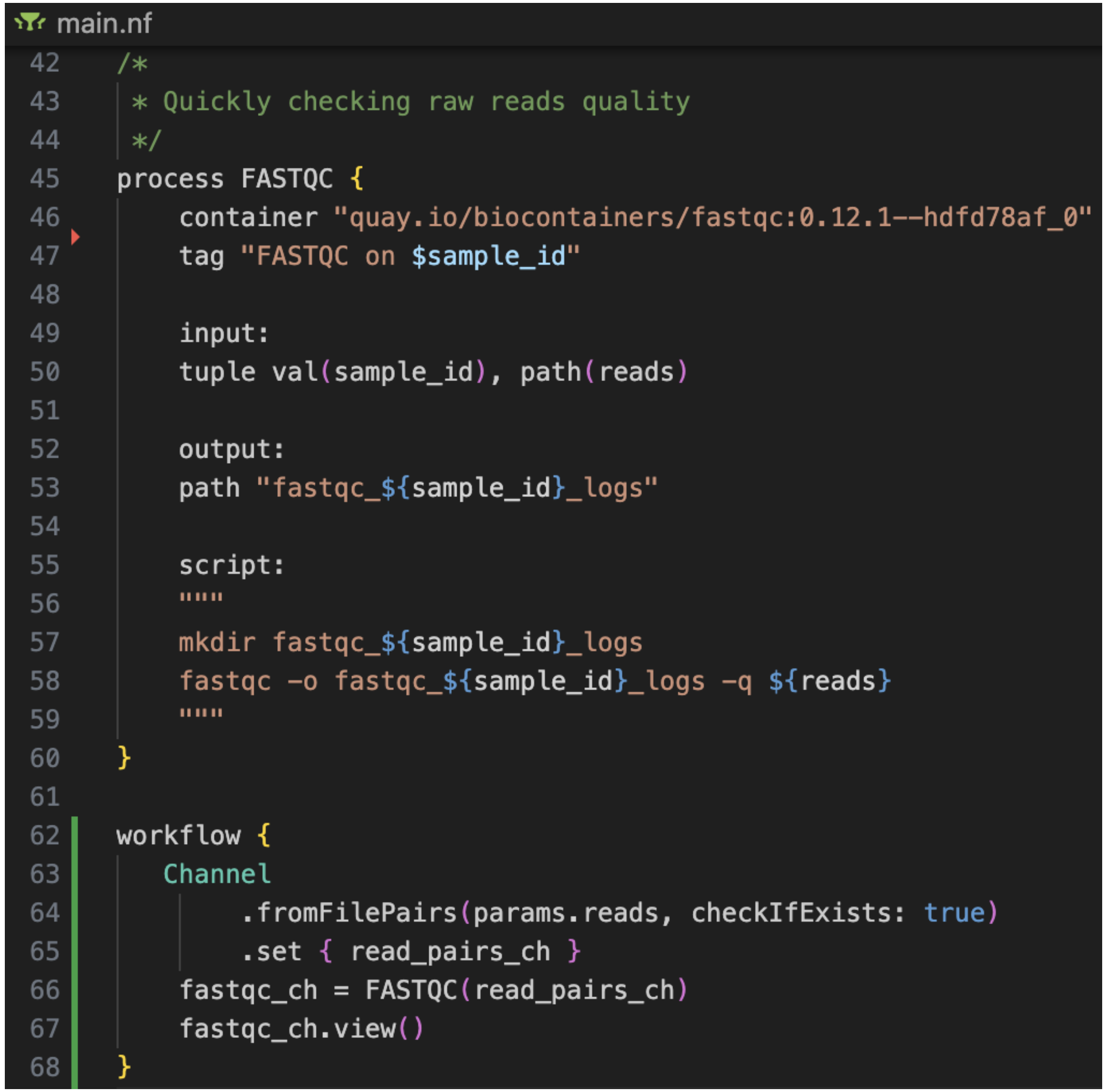
Nextflow is based on few primitives

- **Process:** is every step in your pipeline, and they are executed independently isolated from each other
- **Channels:** Control the data use in processes, and allows the connection of inputs and outputs between processes to make a dependency rule
- **Workflow:** Is the final pipeline itself. If all processes and channels connected, setting the full dependency graph that sets the order of your pipeline execution



A Nextflow script

The .nf files are workflow scripts



main.nf

```
/*
 * Quickly checking raw reads quality
 */
process FASTQC {
    container "quay.io/biocontainers/fastqc:0.12.1--hdfd78af_0"
    tag "FASTQC on $sample_id"

    input:
    tuple val(sample_id), path(reads)

    output:
    path "fastqc_${sample_id}_logs"

    script:
    """
    mkdir fastqc_${sample_id}_logs
    fastqc -o fastqc_${sample_id}_logs -q ${reads}
    """
}

workflow {
   Channel
        .fromFilePairs(params.reads, checkIfExists: true)
        .set { read_pairs_ch }
    fastqc_ch = FASTQC(read_pairs_ch)
    fastqc_ch.view()
}
```

Directives

Code block

Process

Channel

Workflow
(or sub-workflow)

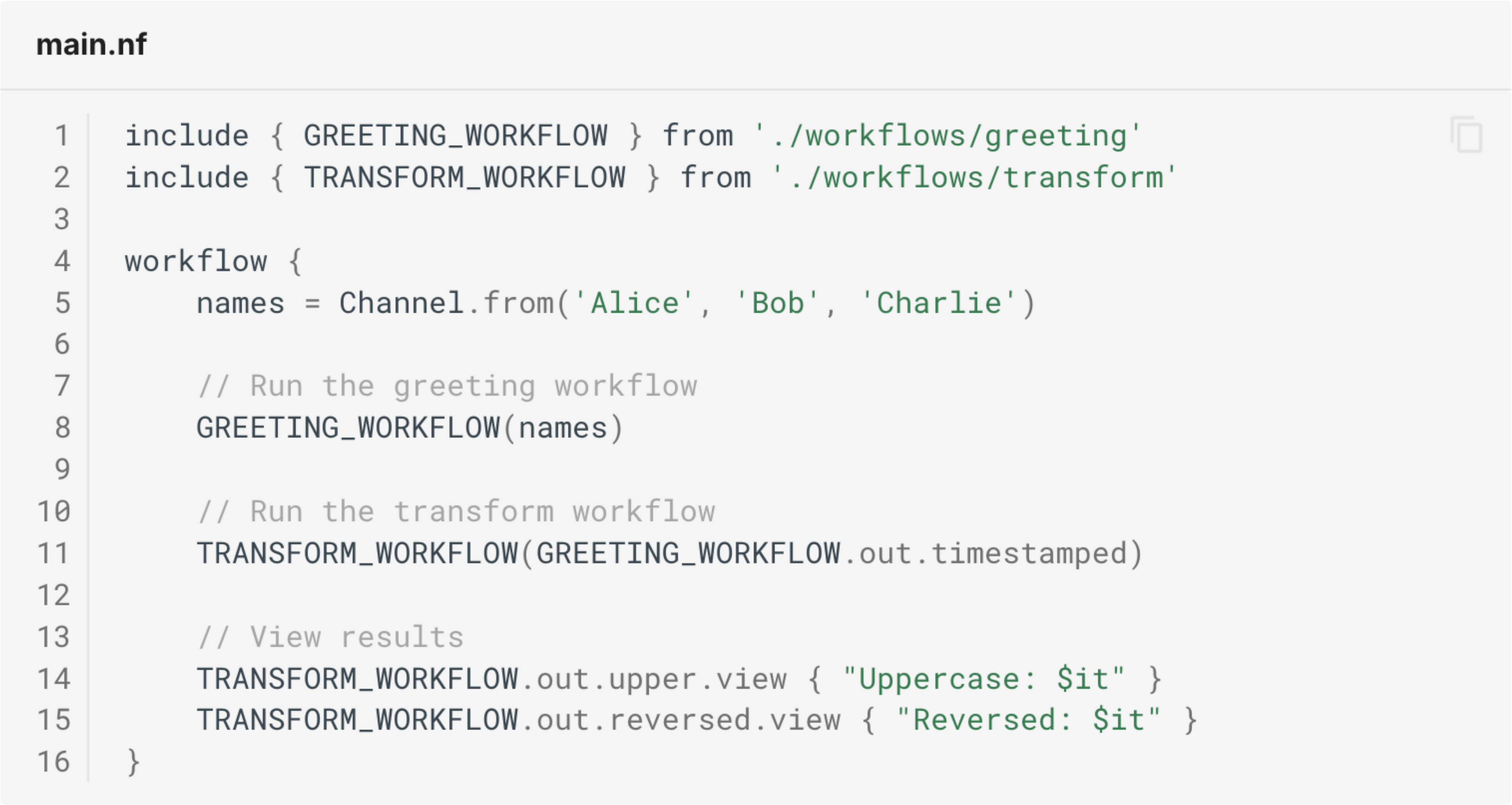
One can also organize separate blocks of pre-defined workflows that can be “glued” together

--- the “sub-workflows”

An extended example

One can see:

- Sub-workflows being called in
- Definition of input-output dependency



main.nf

```
include { GREETING_WORKFLOW } from './workflows/greeting'
include { TRANSFORM_WORKFLOW } from './workflows/transform'

workflow {
    names = Channel.from('Alice', 'Bob', 'Charlie')

    // Run the greeting workflow
    GREETING_WORKFLOW(names)

    // Run the transform workflow
    TRANSFORM_WORKFLOW(GREETING_WORKFLOW.out.timestamped)

    // View results
    TRANSFORM_WORKFLOW.out.upper.view { "Uppercase: $it" }
    TRANSFORM_WORKFLOW.out.reversed.view { "Reversed: $it" }
}
```

In other words …

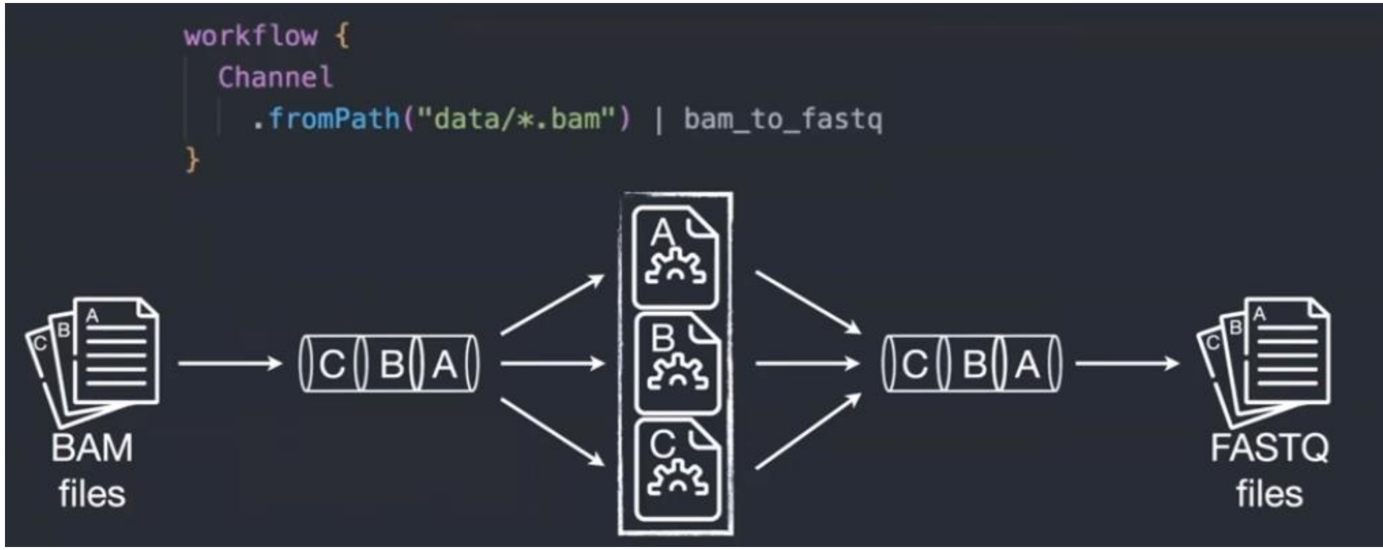
Processes / Modules → Sub-workflows → Pipeline / workflow

Nextflow – Interesting features

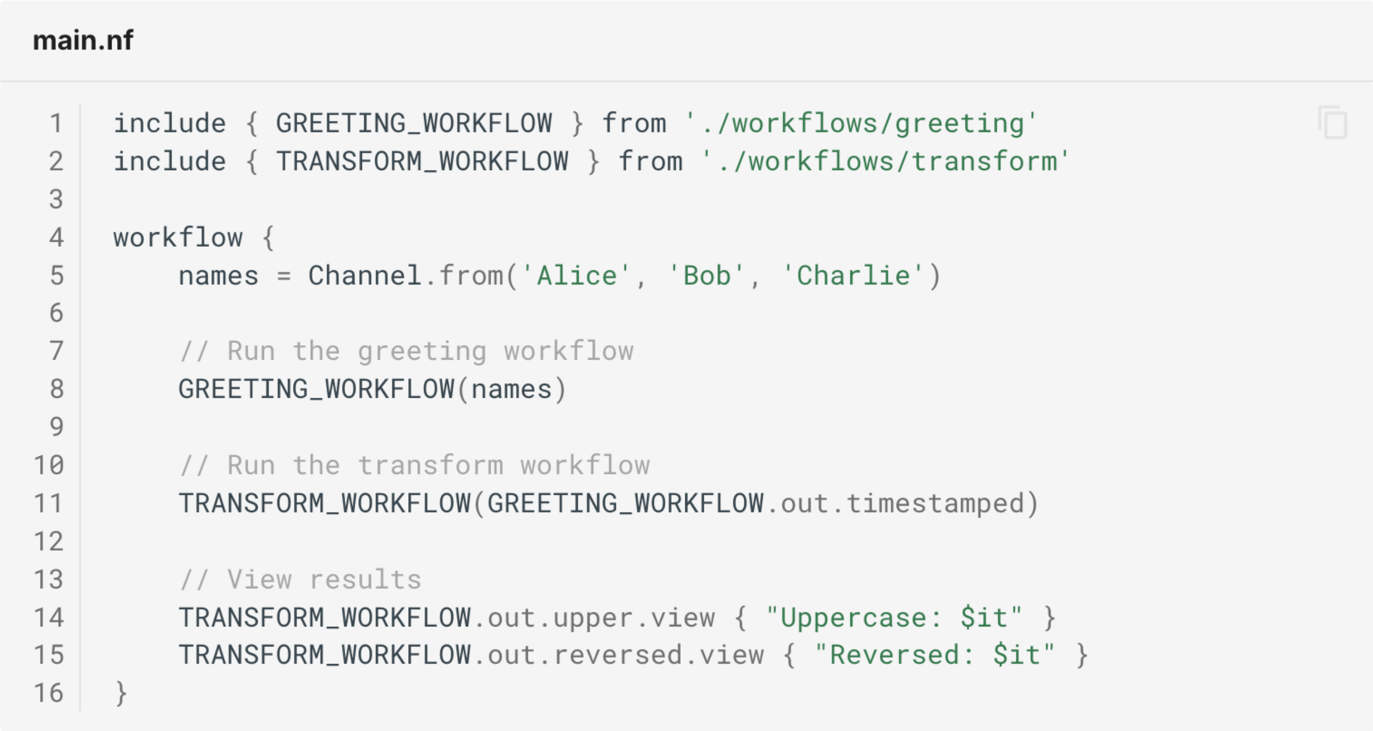
- Implicit **parallelism** (tasks in a process are run by default in parallel)
    - Every module is a “startable” task as long as its input channel exists

- **Re-entrancy** (resume partial runs, do not need to rerun the entire pipeline when something went wrong, it starts from wherever it stopped)

- **Reusability** (reuse different modules, subworkflows, written and containerized by the Nextflow community)



```
workflow {
  Channel
    .fromPath("data/*.bam") | bam_to_fastq
}
```



main.nf

```
include { GREETING_WORKFLOW } from './workflows/greeting'
include { TRANSFORM_WORKFLOW } from './workflows/transform'

workflow {
    names = Channel.from('Alice', 'Bob', 'Charlie')

    // Run the greeting workflow
    GREETING_WORKFLOW(names)

    // Run the transform workflow
    TRANSFORM_WORKFLOW(GREETING_WORKFLOW.out.timestamped)

    // View results
    TRANSFORM_WORKFLOW.out.upper.view { "Uppercase: $it" }
    TRANSFORM_WORKFLOW.out.reversed.view { "Reversed: $it" }
}
```

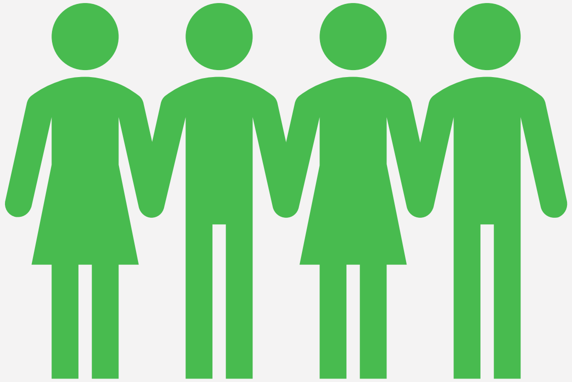
How Nextflow address the challenges of today

- **Reproducible** between runs
  - integration with code management tools
  - all packages downloaded, organized in containers, and control over computing environment
- **Portable** between systems
  - you can write the code in your laptop and can run everywhere (HPC, cloud)
  - works with most of computing environments
- **Scalable**
  - it can be run for 10 on your laptop or thousands of samples in an HPC or the cloud
- **Integration** of existing tools, systems, and industry standards

Break

Behind Nextflow

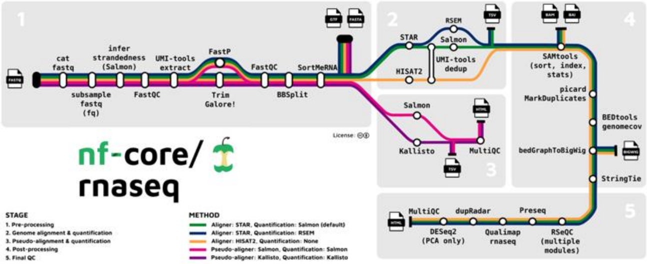
The community itself is what makes Nextflow great



- Driven by #OpenScience and #OpenSource
- Driven by the need to know software, params, options, method used, being sure of what is done
- Nextflow advocates for an open community with multiple channels to interact and connect people, forums, Slack channels, conferences, hackathons regularly organize
- **Nf-core**

nf-core

A global community effort to collect a curated set of open-source analysis pipelines built using Nextflow.
https://nf-co.re/

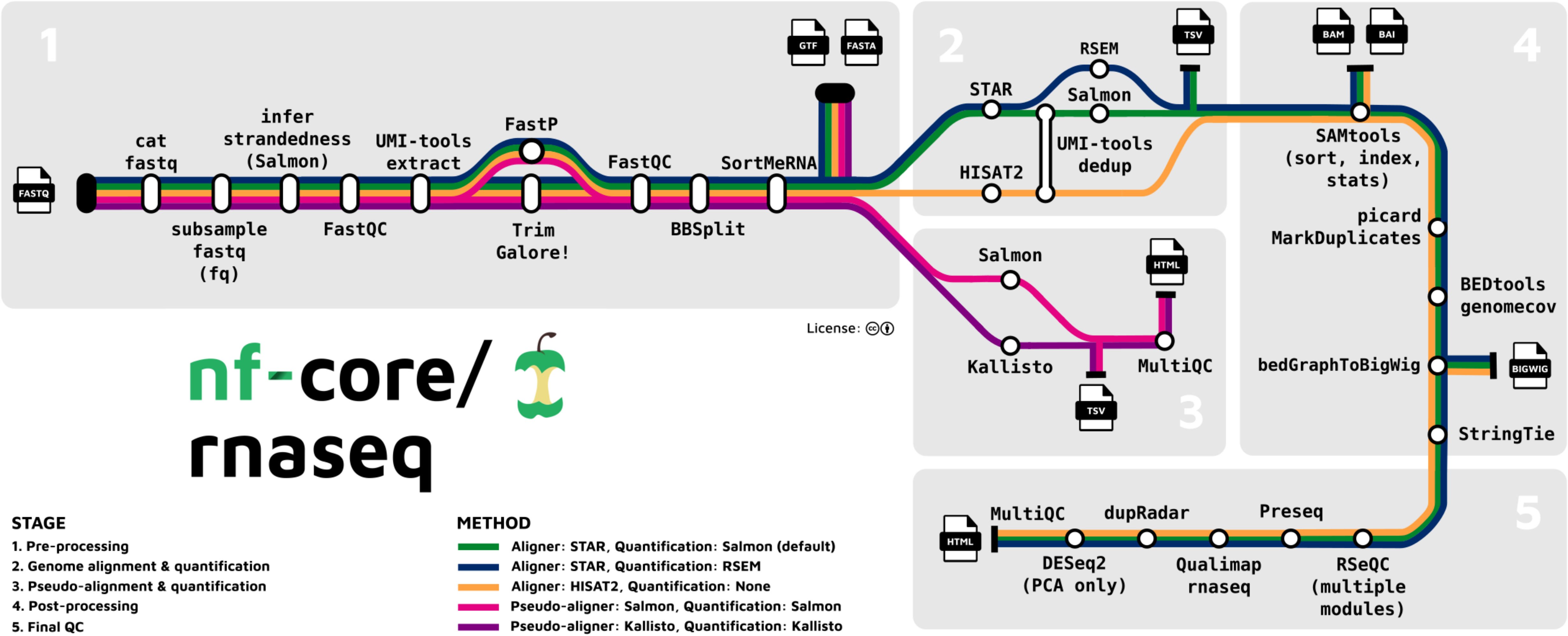


>100 **pipelines** available to process and analyse many different data types

Modules: >1,000 reusable components that can be integrated into pipelines

Subworkflows: >70 pre-assembled combinations of modules aimed at streamlining commonly used workflows

A nextflow pipeline in metro map



nf-core/ rnaseq

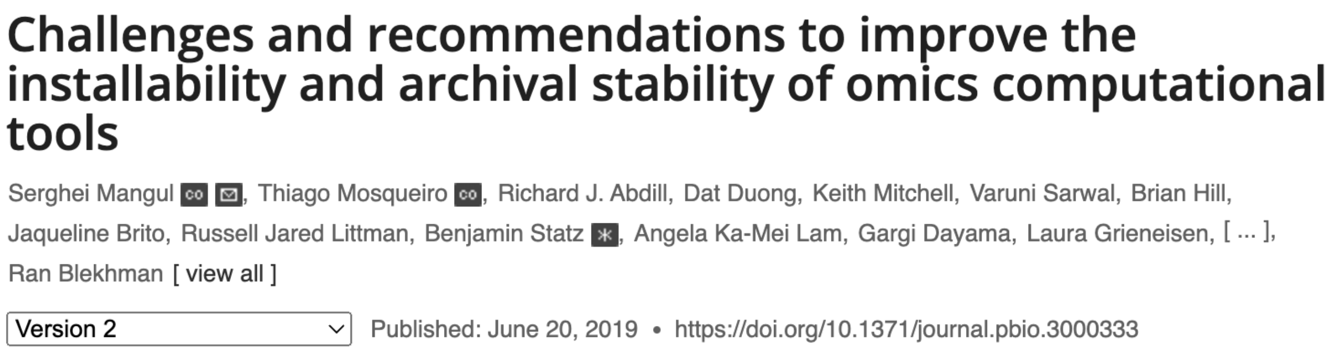
STAGE

1. Pre-processing
2. Genome alignment & quantification
3. Pseudo-alignment & quantification
4. Post-processing
5. Final QC

METHOD

- 🟩 Aligner: STAR, Quantification: Salmon (default)
- 🟦 Aligner: STAR, Quantification: RSEM
- 🟧 Aligner: HISAT2, Quantification: None
- 🟥 Pseudo-aligner: Salmon, Quantification: Salmon
- 🟪 Pseudo-aligner: Kallisto, Quantification: Kallisto

Why do we need a Community effort?

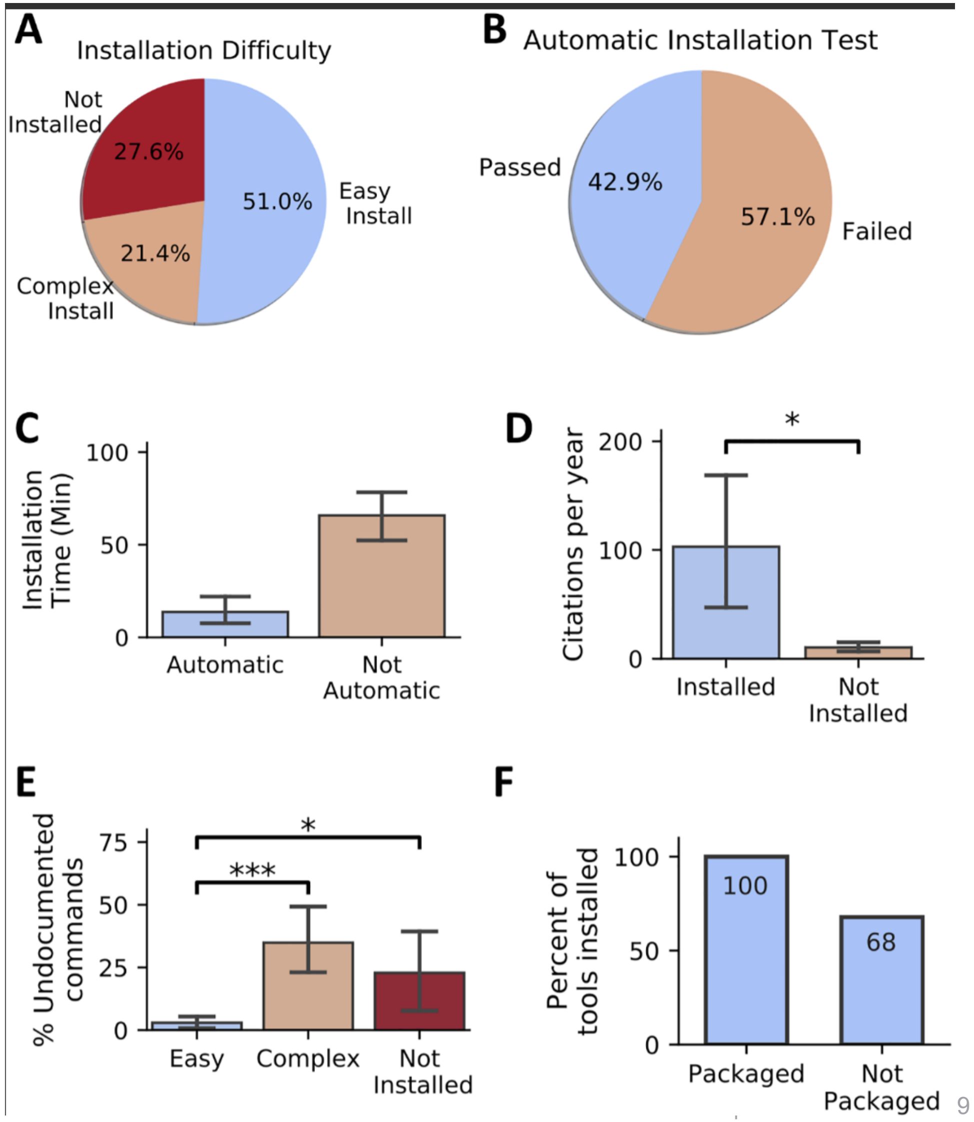


Challenges and recommendations to improve the installability and archival stability of omics computational tools

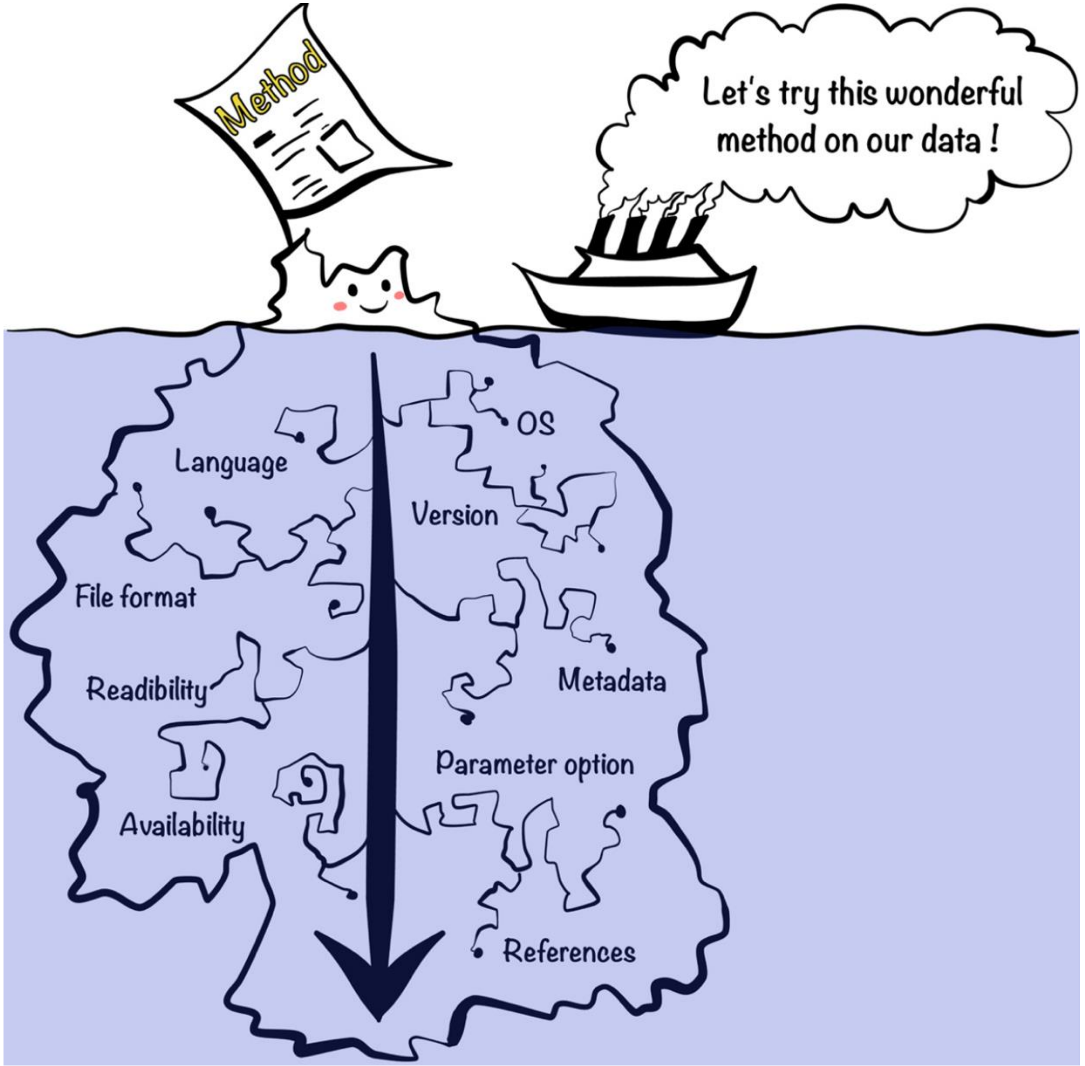
Serghei Mangul, Thiago Mosqueiro, Richard J. Abdill, Dat Duong, Keith Mitchell, Varuni Sarwal, Brian Hill, Jaqueline Brito, Russell Jared Littman, Benjamin Statz, Angela Ka-Mei Lam, Gargi Dayama, Laura Grieneisen, […], Ran Blekhman [view all]

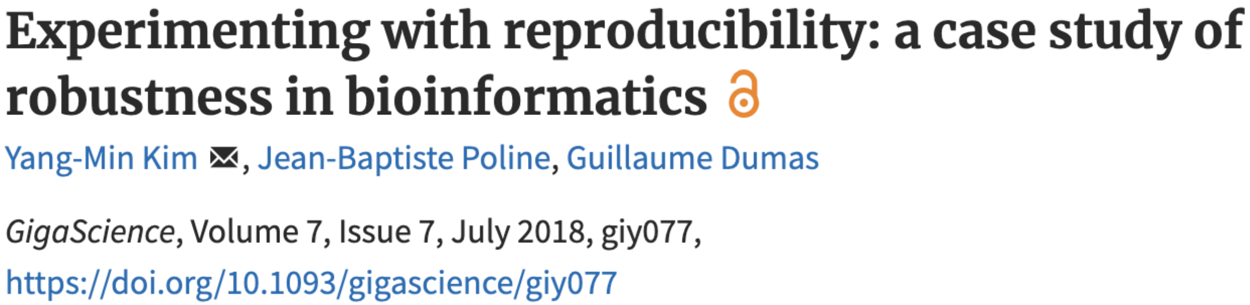
Version 2 | Published: June 20, 2019 • https://doi.org/10.1371/journal.pbio.3000333

- 28% of all omics software resources were not accessible through URLs published
- Among the tools found, 49% were difficult to install or could not be installed at all!



Why do we need a Community effort?





Experimenting with reproducibility: a case study of robustness in bioinformatics

Yang-Min Kim, Jean-Baptiste Poline, Guillaume Dumas

GigaScience, Volume 7, Issue 7, July 2018, giy077,

https://doi.org/10.1093/gigascience/giy077

- "First we tried to rerun the analysis with the code and the data provided by the authors. Second we reimplemented the whole method in a python package…"

Nf-core helps addressing some of the challenges

Modules are open source and maintained by the community --- **shared ownership**

All modules are containerized with public containers --- **portable and repeatable**

All steps have automated CI/CD testing to ensure results are reproduced even after changes --- **reproducibility**

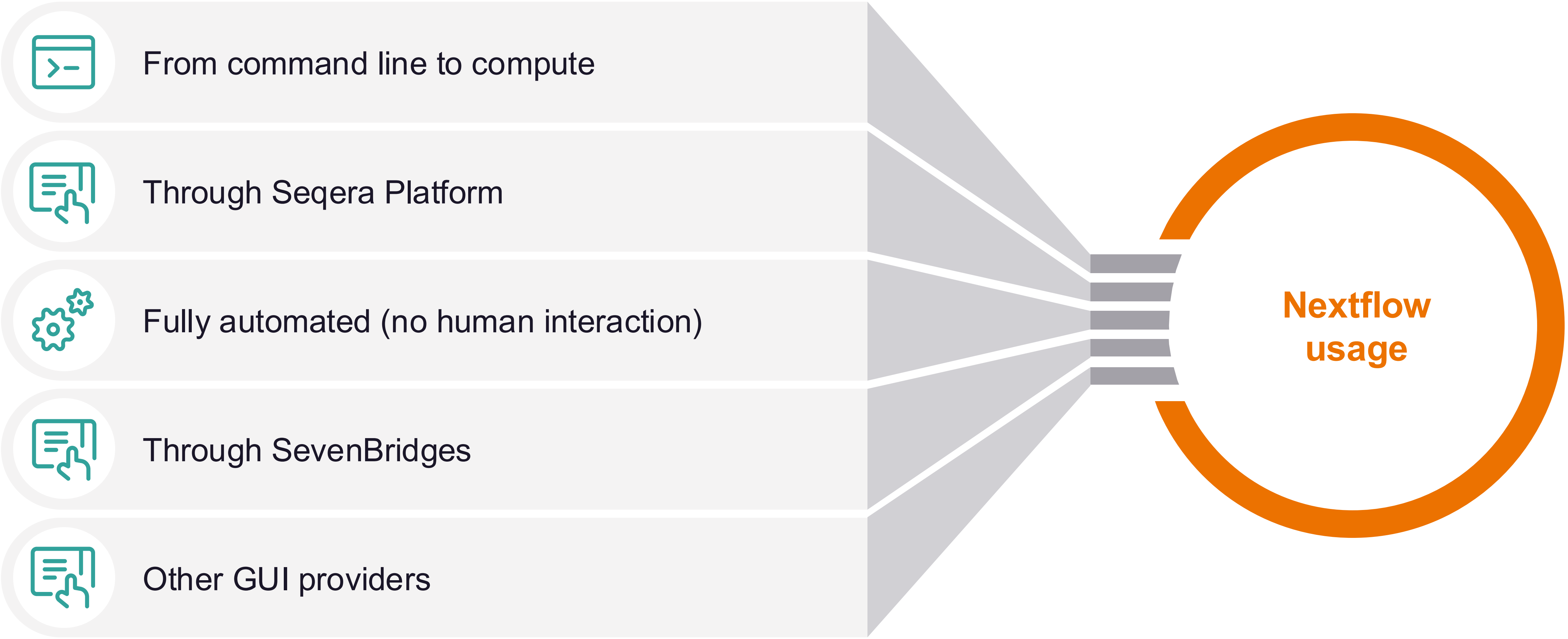
Not only modules, but the same applies to shared sub-workflows and even pipelines --- **reproducibility**

Even when very careful, reproducibility is challenging.

Together, via open science and shared ownership, the community ensures all pieces are tested and reproducible all together, while being extremely portable.

Nextflow in Industry

Widely adopted, but in many different ways



From command line to compute

Through Seqera Platform

Fully automated (no human interaction)

Through SevenBridges

Other GUI providers

Nextflow usage

Anyways …

The community guidelines, templates and code are widely adopted!

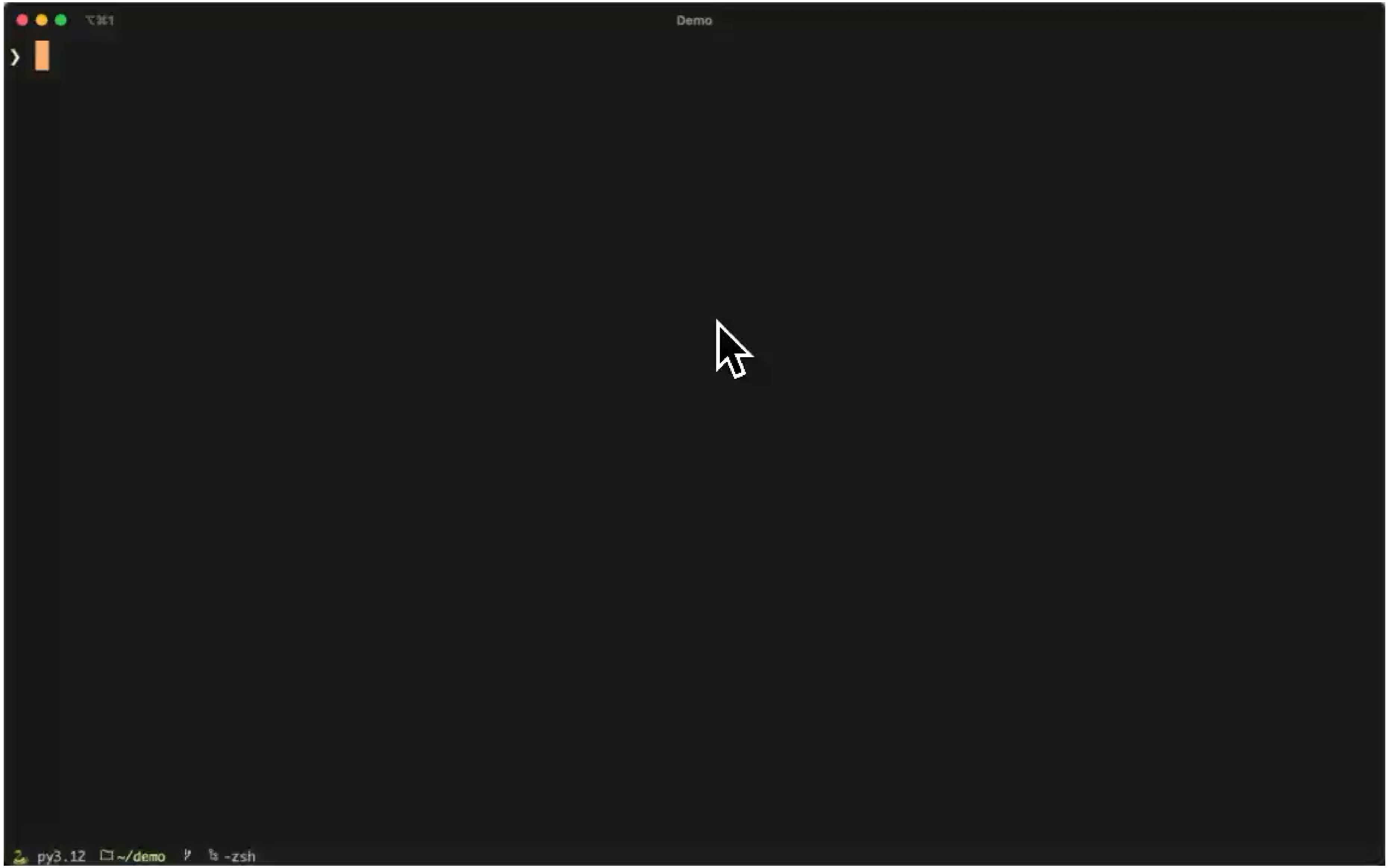
- Some use as is, directly from nf-core git and contribute there
- Adding modules, pipelines, sub-workflows, etc. – Being part of it

nf-core

- Some have internal private services so they “wrap-up extra modules around
- Using the shared modules to benefit from the thorough community testing and validation

Some examples of deployment and industry case study

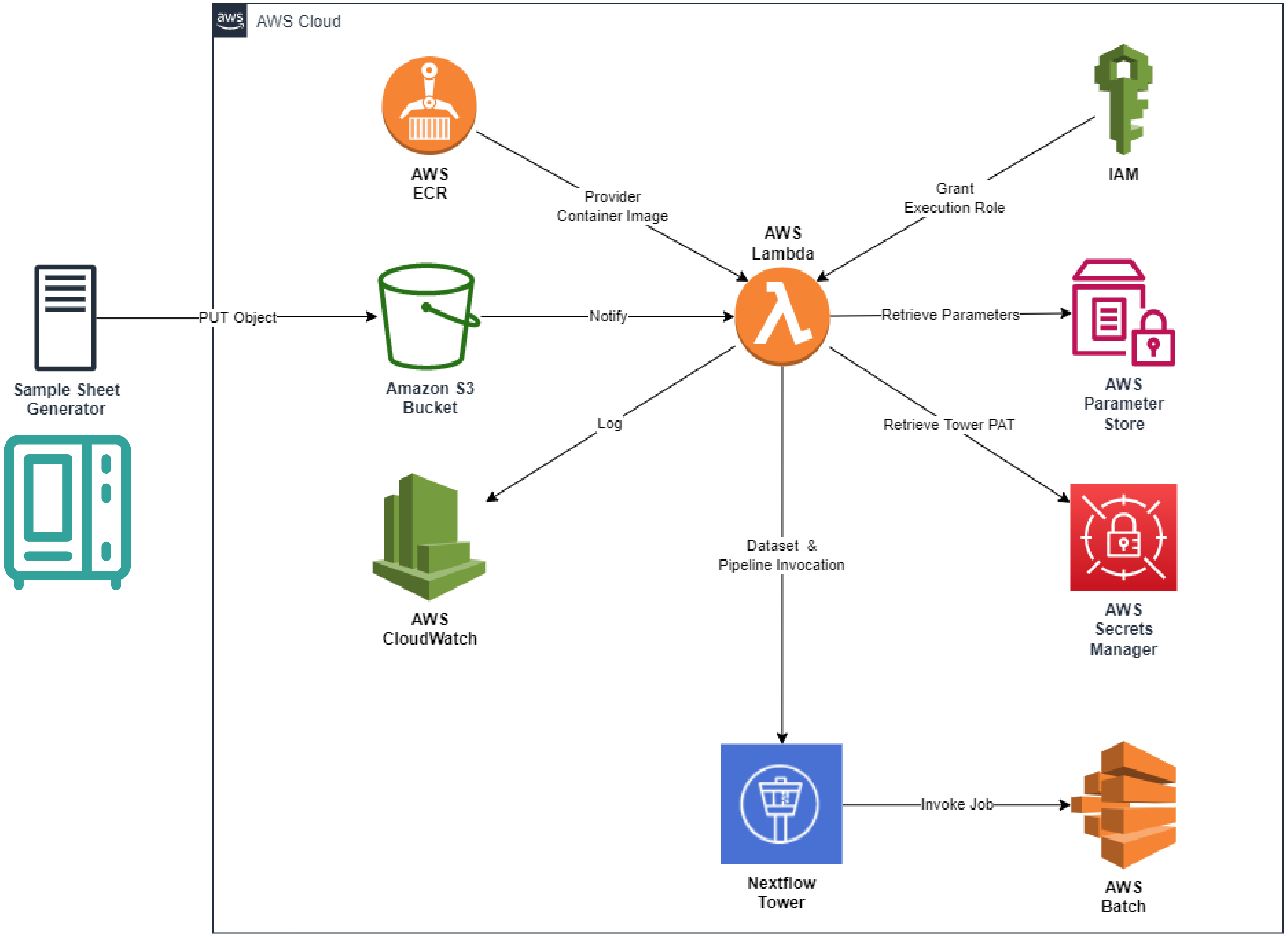
Via Seqera Platform



```
❯ 
```

```
py3.12  ~/demo    -zsh
```

Via automated settings



Machines upload data that triggers the engine

ZS Discovery

Elucidate. Innovate. Accelerate.

Thank you!

Impact where it matters.