Nextflow Workshop

November 2021

Seamless data pipelines at scale

Schedule

Day	Time	Topics
Day 1 – Nov 16, 2021	**11:30 – 2:00 ET**	Introduction to Nextflow
		Getting started with Nextflow
		A simple RNA-Seq pipeline
		Managing dependencies & containers
Day 2 – Nov 17, 2021	**11:30 - 1:30 ET**	Nextflow Channels
		Nextflow Processes
		Nextflow Operators
		Groovy basic structures and idioms
		DSL2 and modules
Day 3 – Nov 18, 2021	**11:30 - 1:30 ET**	Nextflow configuration
		Deployment scenarios
		Execution cache and resume
		Error handling & troubleshooting
		Getting started with Nextflow Tower

nextflow

Seamless data pipelines at scale.

Big data analysis can get complicated.
Cluster & cloud configuration is difficult.

Enable your team to build, deploy and manage pipelines collaboratively on any infrastructure.

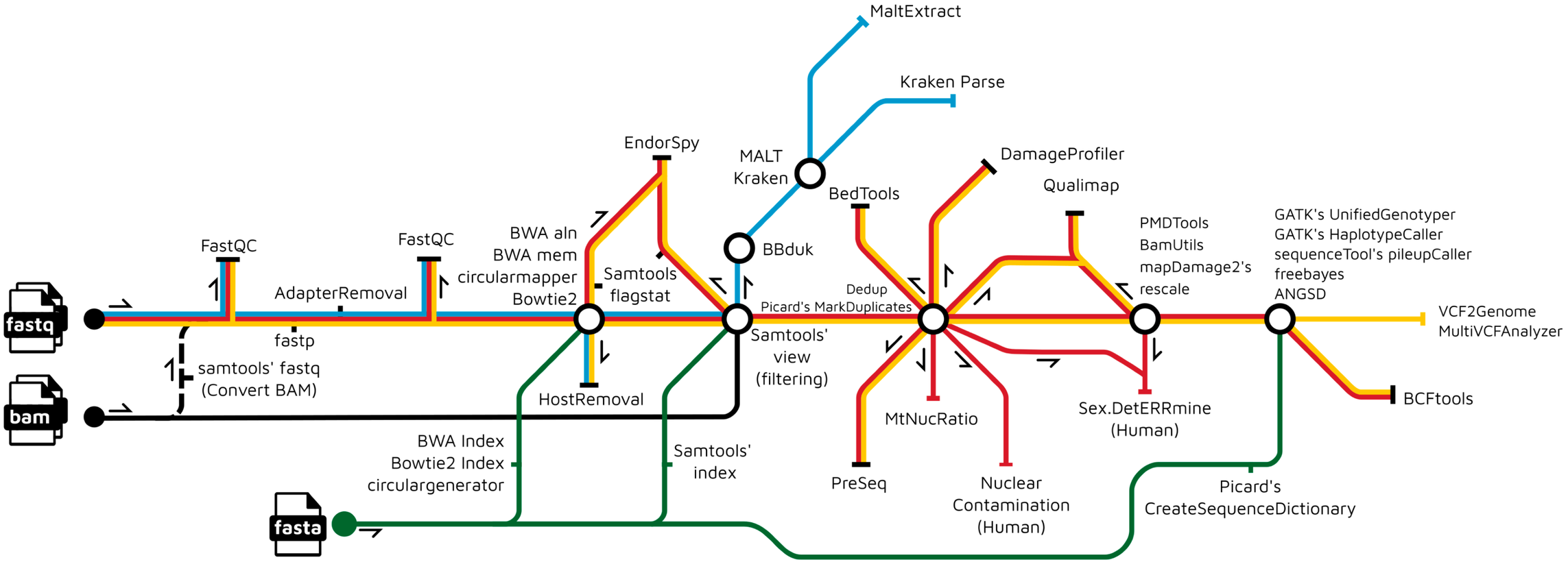
Build better pipelines, reduce costs and improve time to results.

Evan Floden

CEO & Co-founder Seqera Labs

evan@seqera.io

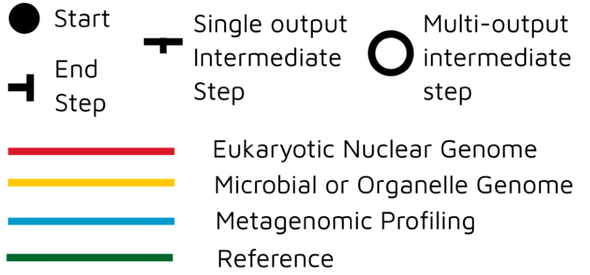
Writing modern workflows is complex



nf-core/eager v2.4

Example analysis pathways

Legend

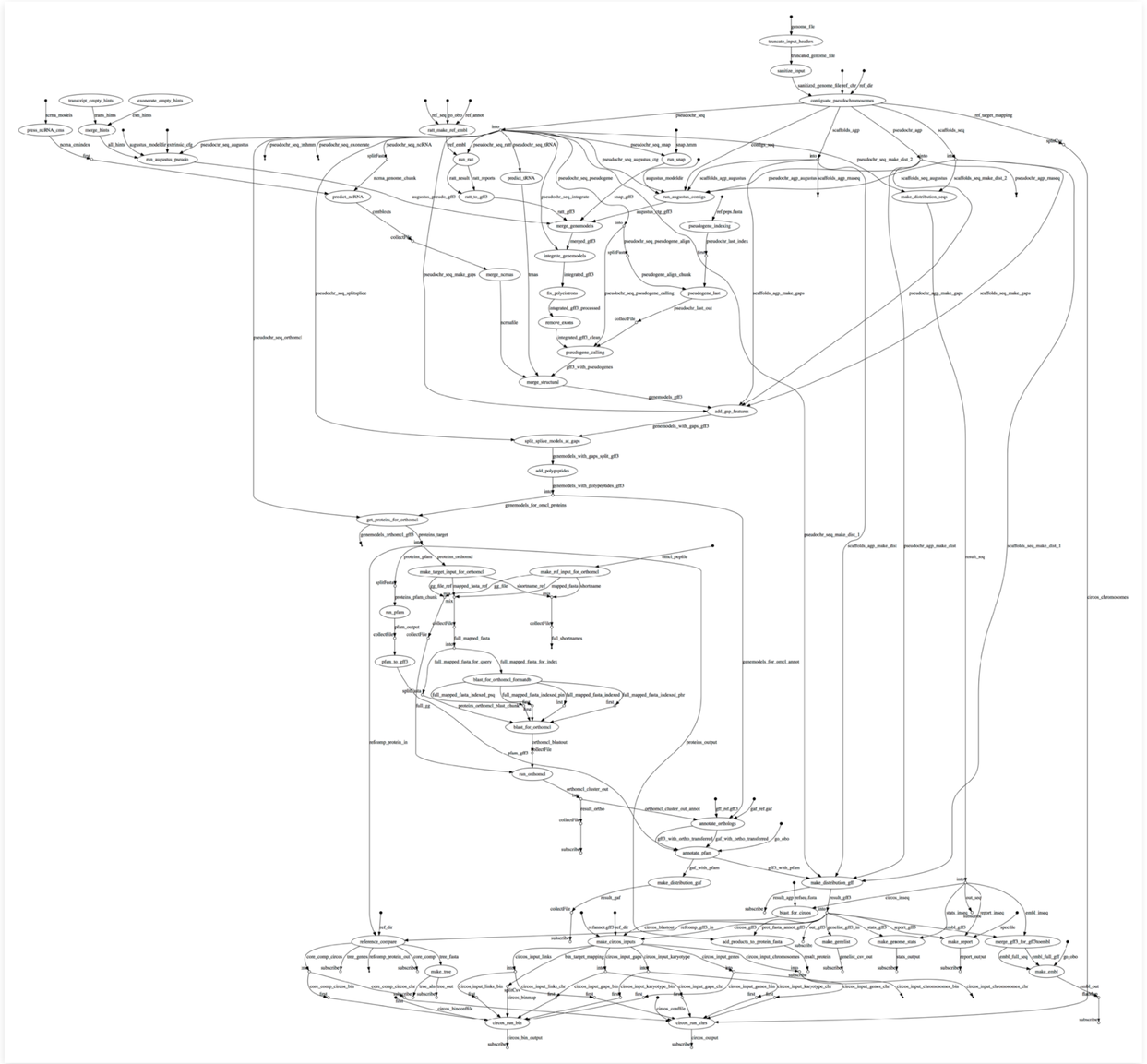


Data analysis workflows

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

A lot of moving parts!

- 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

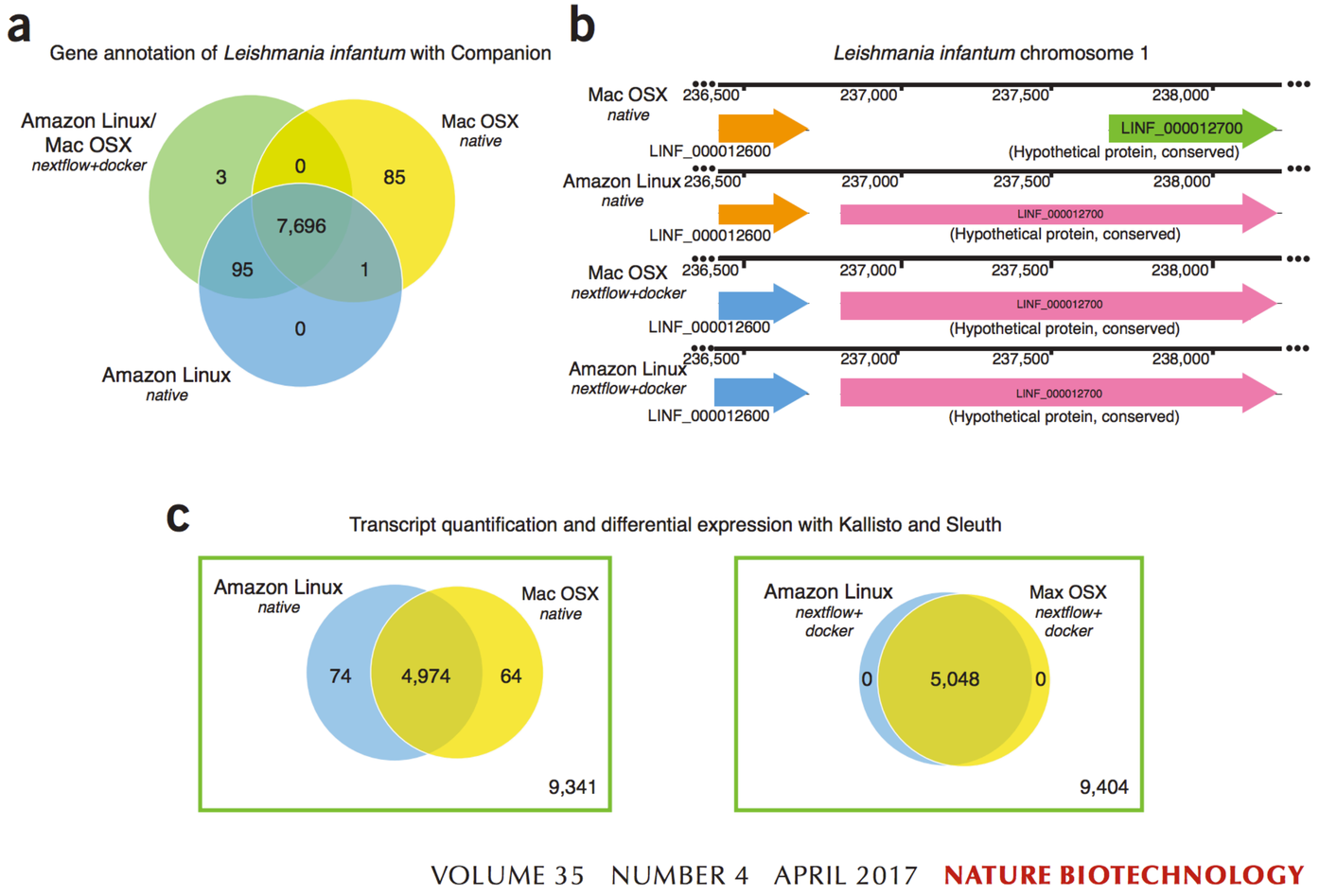
Daniel Garijo[1], Sarah Kinnings[2], Li Xie[3], Lei Xie[4], Yinliang Zhang[5], Philip E. Bourne[3]*, Yolanda Gil[6]*

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

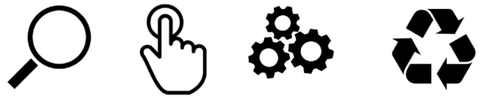


VOLUME 35 NUMBER 4 APRIL 2017 NATURE BIOTECHNOLOGY

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

What is our goal?

Findable Accessible Interoperable Reusable + Equitable Scalable Tested



Streamlining data analysis to make it:

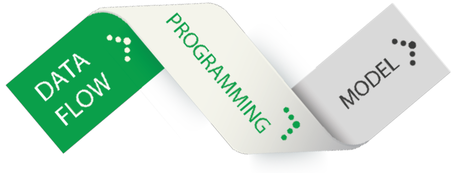
- Findable ⇒ openly available and searchable through repositories
- Accessible ⇒ can be used by everyone
- Interoperable ⇒ portable to any cluster & cloud
- Reusable ⇒ in a reproducible way
- Equitable ⇒ free of bias
- Scalable ⇒ from laptop to supercomputer
- Tested ⇒ validated using modern software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

seqeralabs

What is Nextflow?

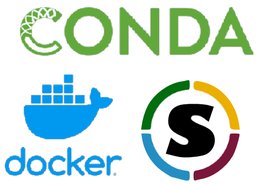
nextflow *script*

Write code
in any language



Define orchestration with
dataflow programming

Define software
dependencies
with containers



Version control

nextflow *runtime*

Orchestration of tasks to
deploy anywhere with ease

Supports all major platforms

Nextflow is a language, a runtime and a community

How does it work?

- **Fast prototyping** ⇒ custom DSL that enables task composition, simplifies most use cases + general purpose programming language for corner cases
- **Easy parallelisation** ⇒ declarative reactive programming model based on dataflow paradigm, implicit portable parallelism
- **Self-contained** ⇒ functional approach, a task execution is idempotent ie. 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
    """

}
```

Tasks 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
    """
}
```

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
    """
}
```

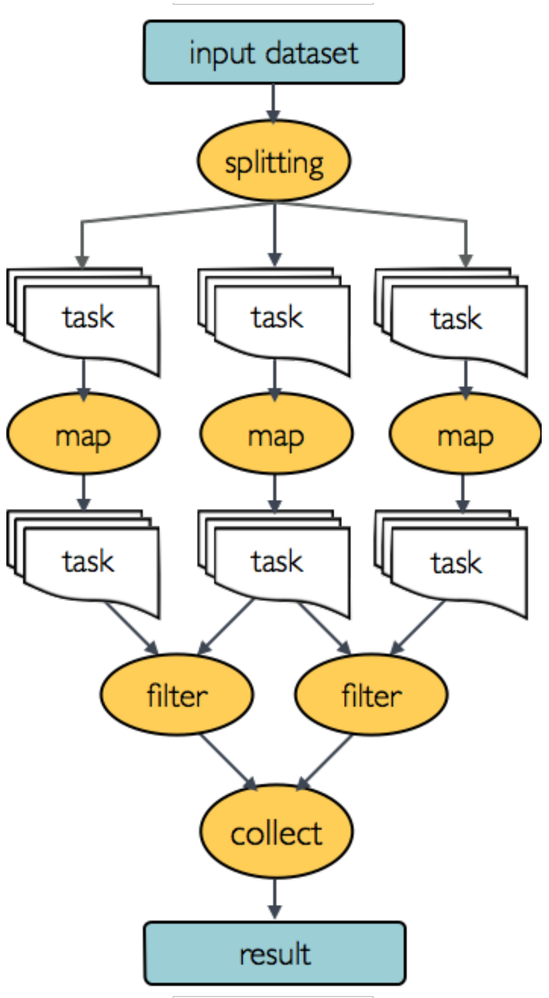
workflow

```
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
}
```

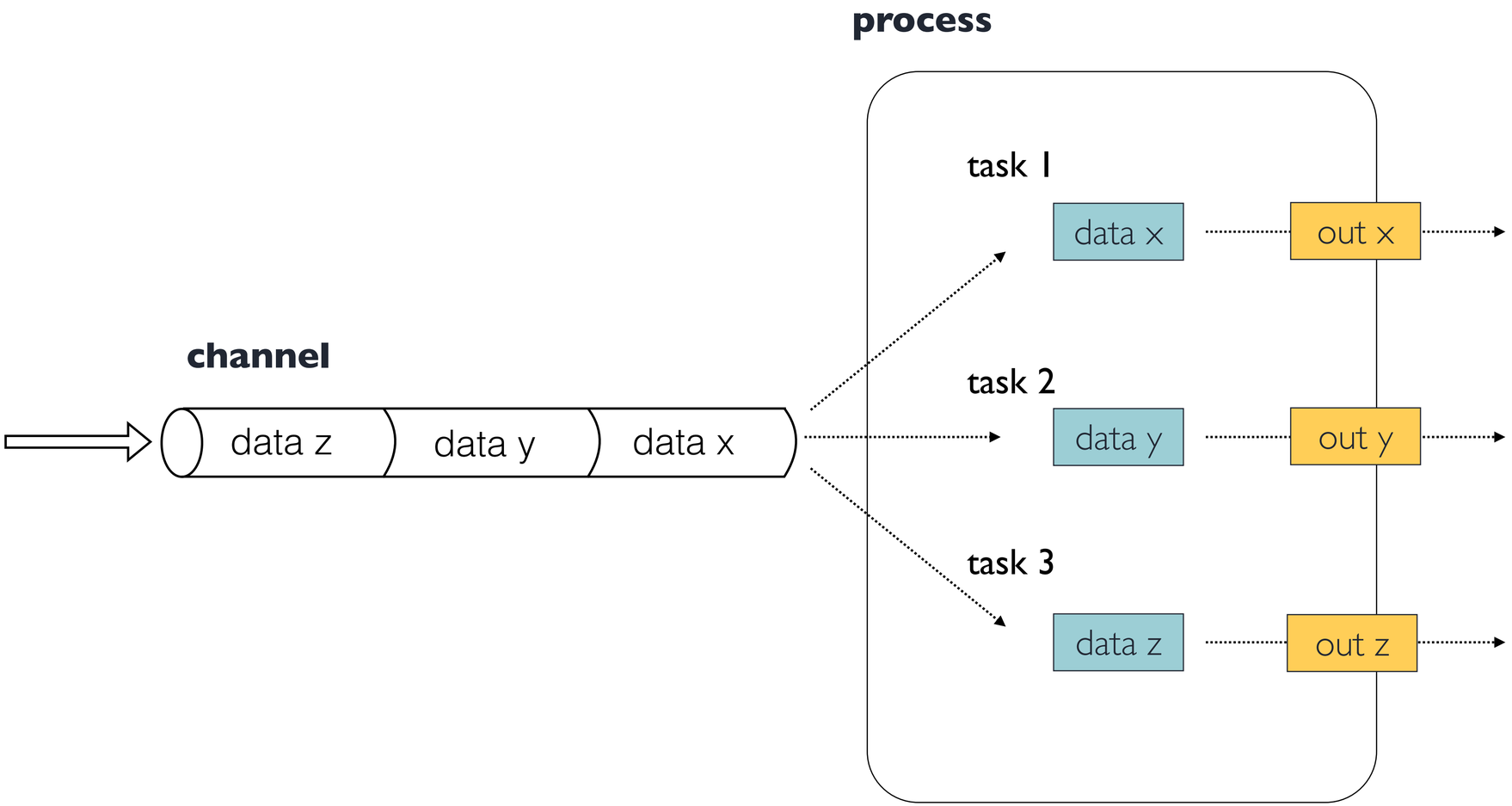
Scientist and engineers can now write complex, distributed and parallel data pipelines without requiring a degree in computer science.

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
- Parallelisation and tasks dependencies are implicitly defined by process in/out declarations



How does parallelisation work?



How does parallelisation work?

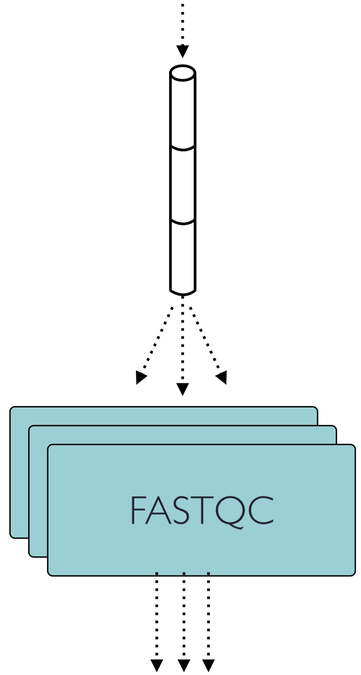
```
samples_ch = Channel.fromPath('data/sample.fastq')


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}
    """
}
```

How does parallelisation work?

```
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}
    """
}
```

Implicit parallelism

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



CWL vs. Nextflow

- Language specification
- Declarative meta-language (YAML/JSON)
- Verbose
- Committee driven
- Many vendors/implementations (and specification version)

- Language + app. runtime
- DSL on top of a general purpose programming lang.
- Concise, fluent
- Community driven
- Single implementation, quick iterations

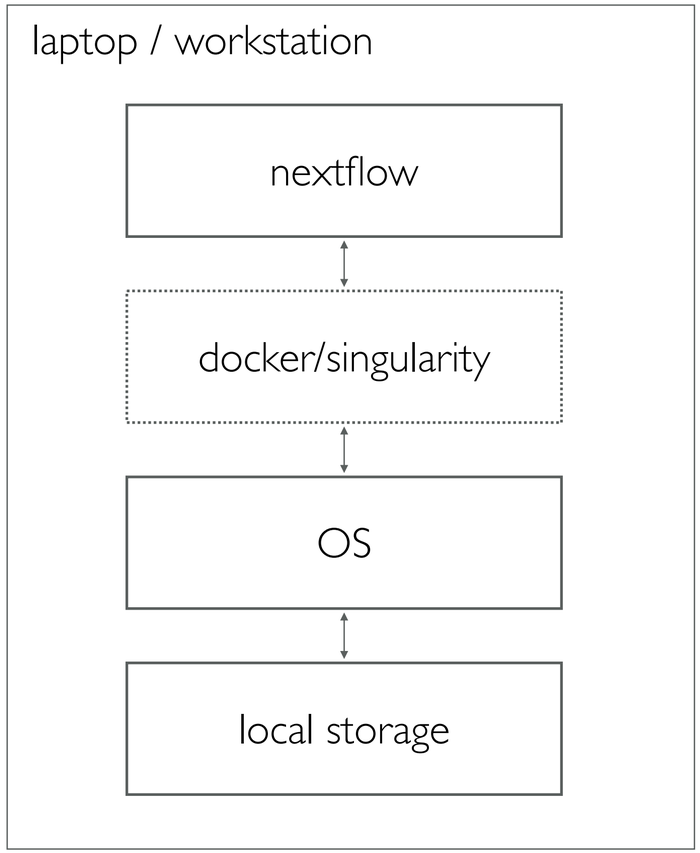
Snakemake vs. Nextflow

- 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

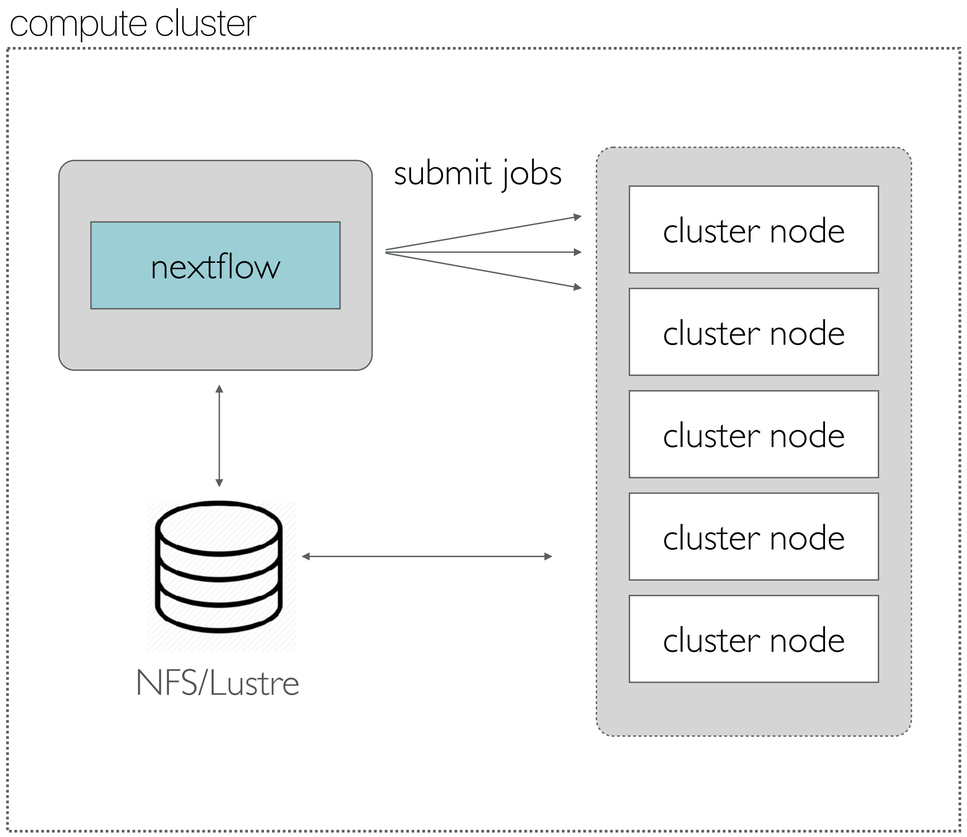
Deployment Scenarios

Local execution



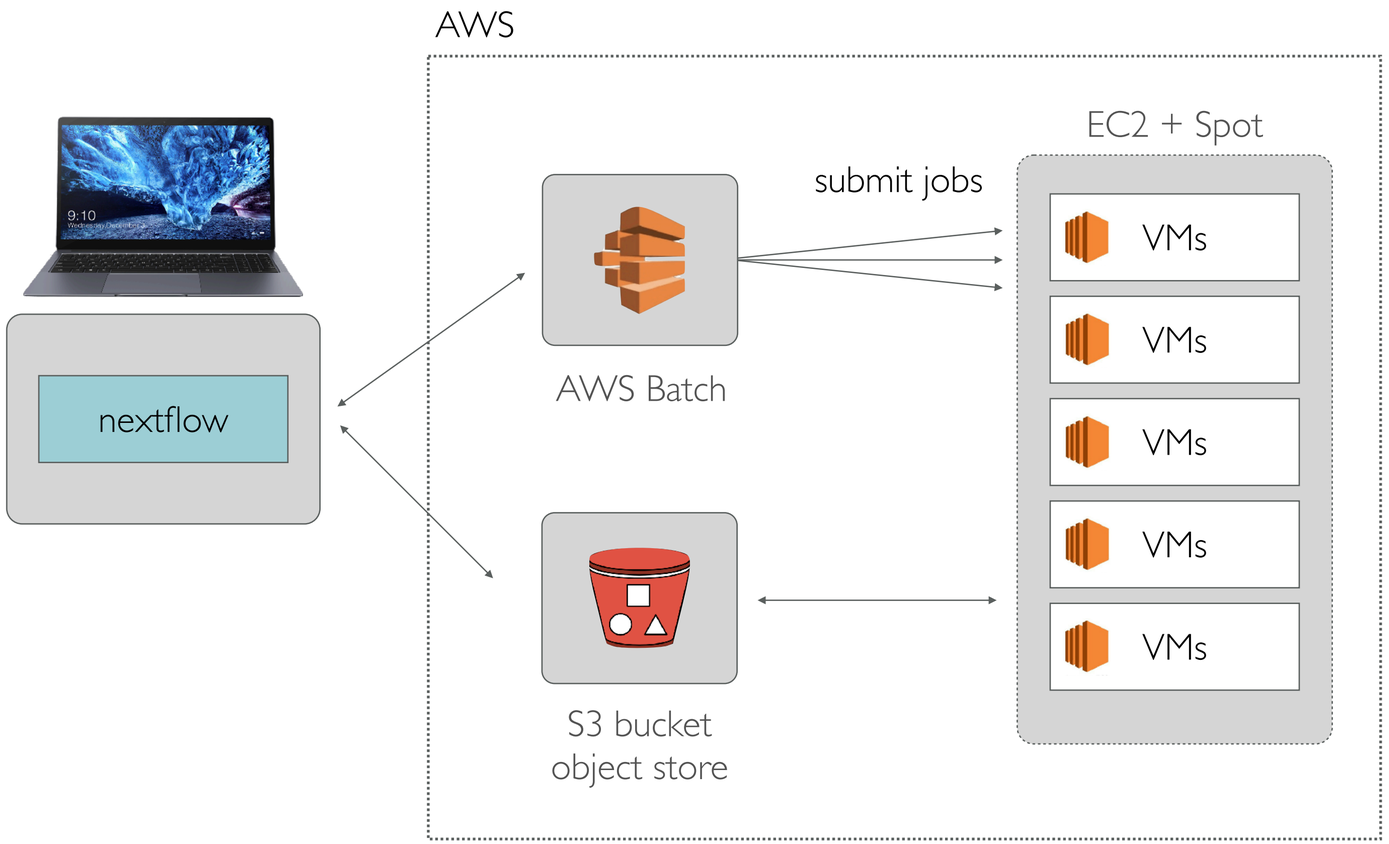
- Common development scenario
- Dependencies can be managed using a container runtime
- Parallelisations is managed spawning posix processes
- Can scale vertically using fat server / shared mem. machine

Centralised cluster orchestration



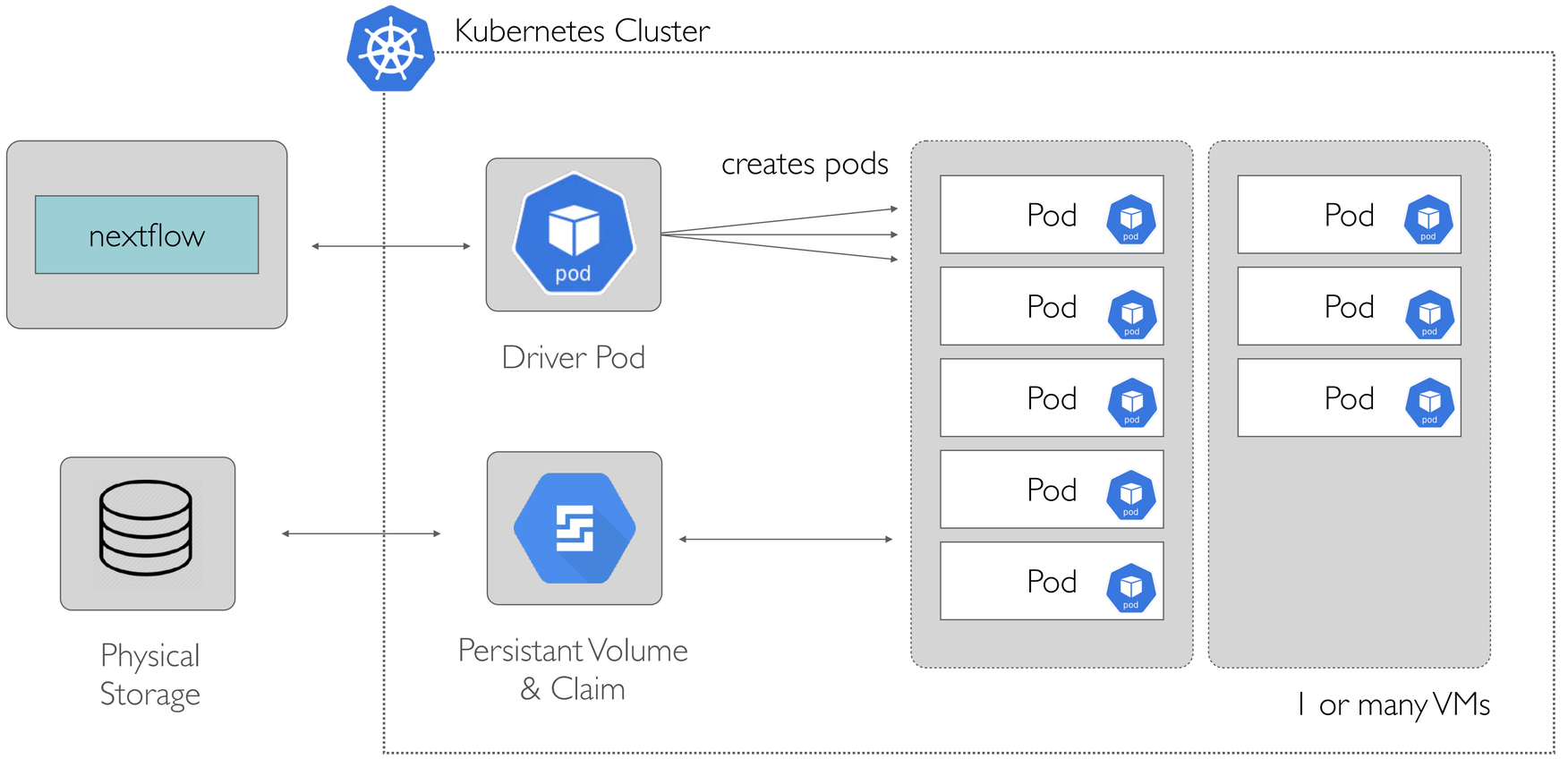
- Nextflow orchestrates workflow execution submitting jobs to a compute schedular.
- Can run in the head node or a compute node.
- Requires a shared storage to exchange data between tasks.
- Ideal for corse-grained parallelism.

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.

Cloud native orchestration



- Nextflow orchestrates workflow execution
- Each task is executed as a kubernetes pod.
- Transfer of data between tasks via a persistent volume claim.

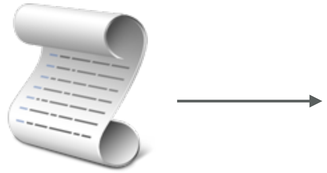
kubernetes

Portability

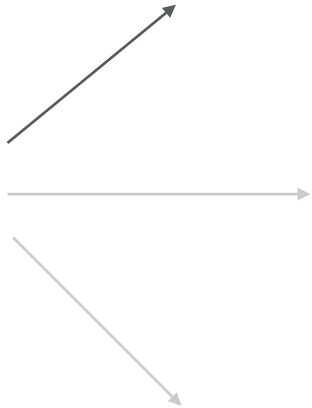
Portability

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

Portability



nextflow



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

slurm
workload manager

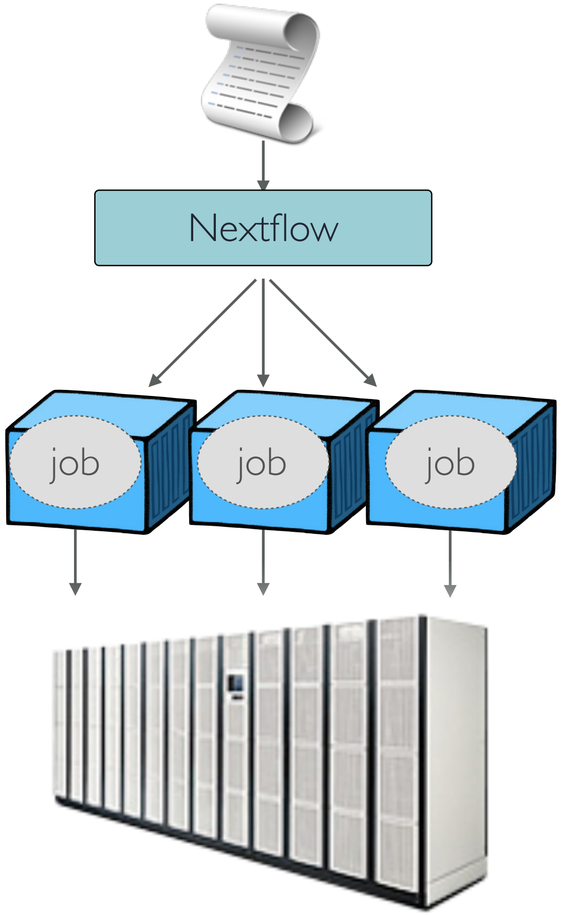
Configuration decoupling
is the key to
portable deployments


Containerisation

Container vs. VM

- Lighter: MB vs GB
- Faster startup: ms/secs vs minutes
- Virtualise a process/application instead of a 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.

Containerisation



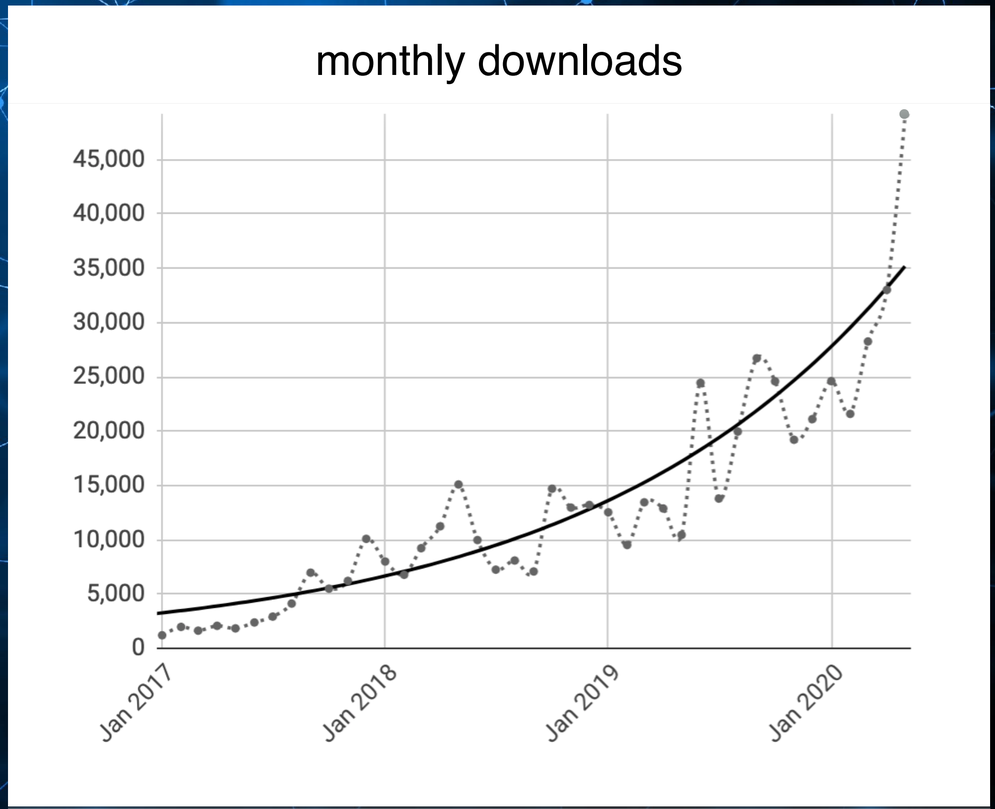
- Nextflow envisioned the use of software containers to fix computational reproducibility
- Mar 2014 (ver 0.7), support for Docker
- Dec 2016 (ver 0.23), support for Singularity
- 2020 support for Podman

When use containers?

Always!

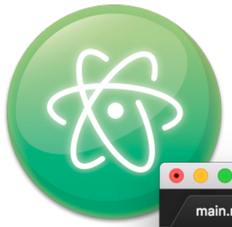
Open Source Community

55K+ monthly downloads	10,000+ active developers /month	35 international Workshops
150K+ lines of code	1.6K+ stars on github	100+ contributors

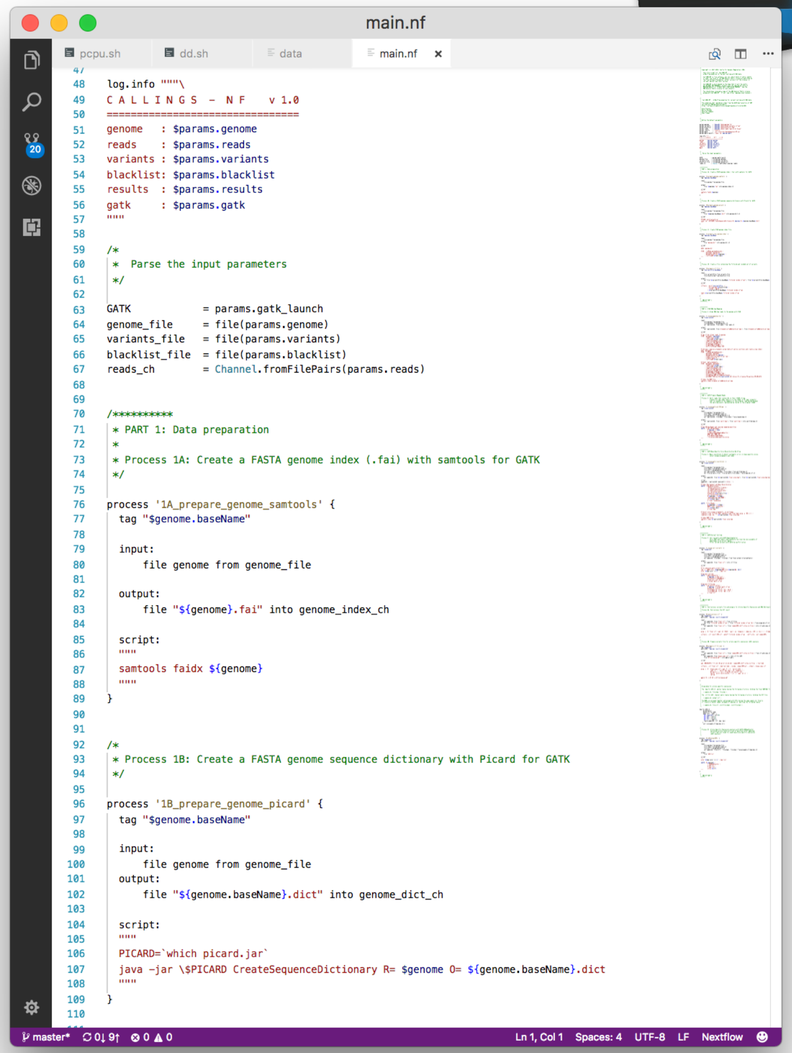


Over 2 million downloads

Editors



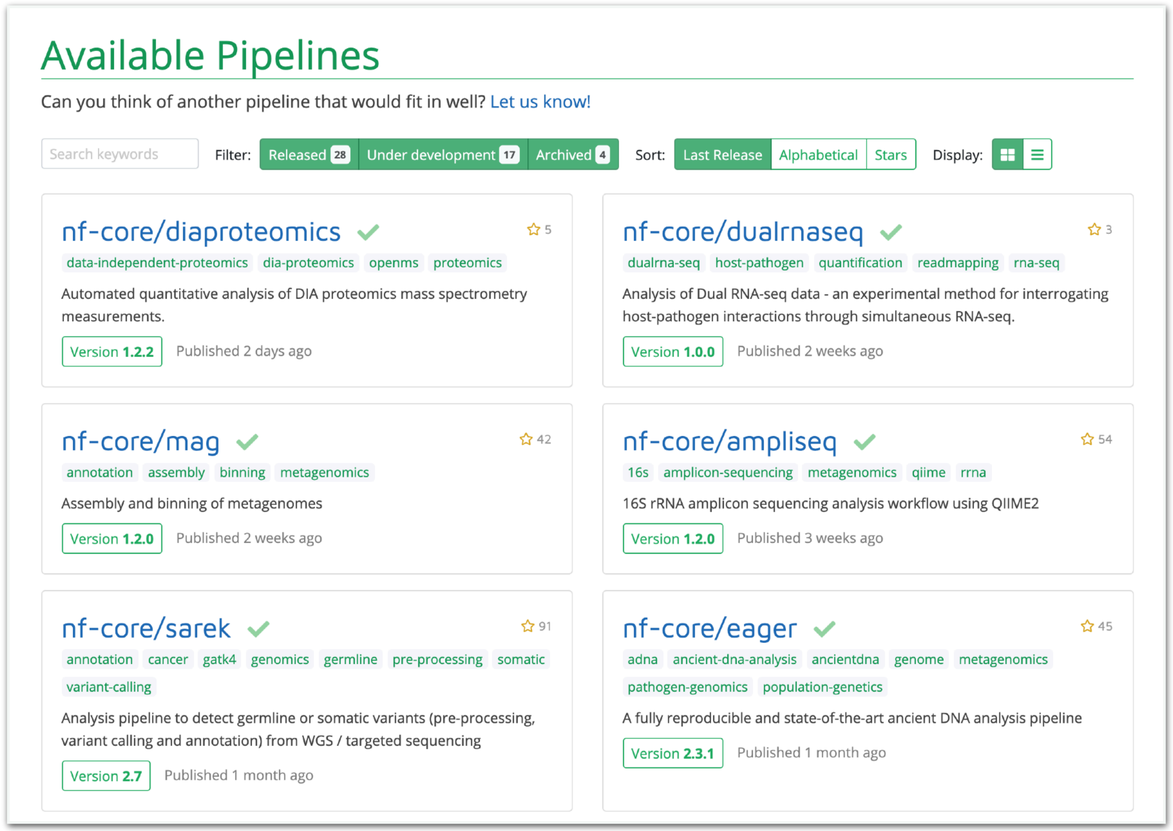

main.nf — ~/projects/callings-nf
main.nf
log.info """\
C A L L I N G S - N F v 1.0
genome : $params.genome
reads : $params.reads
variants : $params.variants
blacklist: $params.blacklist
results : $params.results
gatk : $params.gatk
"""
/*
* Parse the input parameters
*/
GATK = params.gatk_launch
genome_file = file(params.genome)
variants_file = file(params.variants)
blacklist_file = file(params.blacklist)
reads_ch = Channel.fromFilePairs(params.reads)
/**********
* PART 1: Data preparation
*
* Process 1A: Create a FASTA genome index (.fai) with samtools for GATK
*/
process '1A_prepare_genome_samtools' {
tag "$genome.baseName"
input:
file genome from genome_file
output:
file "${genome}.fai" into genome_index_ch
script:
"""
samtools faidx ${genome}
"""
}
/*
* Process 1B: Create a FASTA genome sequence dictionary with Picard for GATK
*/
process '1B_prepare_genome_picard' {
tag "$genome.baseName"
input:
file genome from genome_file
output:
file "${genome.baseName}.dict" into genome_dict_ch
script:
"""
PICARD=`which picard.jar`
java -jar \$PICARD CreateSequenceDictionary R= $genome O= ${genome.baseName}.dict
"""
}
main.nf* 93:1
LF UTF-8 Nextflow master 4 files


main.nf
pcpu.sh
dd.sh
data
main.nf
log.info """\
C A L L I N G S - N F v 1.0
genome : $params.genome
reads : $params.reads
variants : $params.variants
blacklist: $params.blacklist
results : $params.results
gatk : $params.gatk
"""
/*
* Parse the input parameters
*/
GATK = params.gatk_launch
genome_file = file(params.genome)
variants_file = file(params.variants)
blacklist_file = file(params.blacklist)
reads_ch = Channel.fromFilePairs(params.reads)
/**********
* PART 1: Data preparation
*
* Process 1A: Create a FASTA genome index (.fai) with samtools for GATK
*/
process '1A_prepare_genome_samtools' {
tag "$genome.baseName"
input:
file genome from genome_file
output:
file "${genome}.fai" into genome_index_ch
script:
"""
samtools faidx ${genome}
"""
}
/*
* Process 1B: Create a FASTA genome sequence dictionary with Picard for GATK
*/
process '1B_prepare_genome_picard' {
tag "$genome.baseName"
input:
file genome from genome_file
output:
file "${genome.baseName}.dict" into genome_dict_ch
script:
"""
PICARD=`which picard.jar`
java -jar \$PICARD CreateSequenceDictionary R= $genome O= ${genome.baseName}.dict
"""
}
master* 0↓ 9↑ 0 0
Ln 1, Col 1 Spaces: 4 UTF-8 LF Nextflow

Widespread enterprise adoption

Growing community content

Production ready analysis pipelines built with Nextflow.



Chan Zuckerberg Initiative

nf-core

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

VIEW PIPELINES

Deploy

- Stable pipelines
- Centralized configs
- List and update pipelines
- Download for offline use

Participate

- Documentation
- Slack workspace
- Twitter updates
- Hackathons

Develop

- Starter template
- Code guidelines
- CI code linting and tests
- Helper tools

nextflow

data pipelines at scale

massively scalable pipelines across cluster & cloud.

The world's leading workflow software
for genomics, biopharma and life sciences.

features.

Powerful HPC execution engines

Deploy across cloud & clusters effortlessly

Out-of-the-box support for AWS and GCP plus schedulers including SLURM, LSF & Grid Engine.

Portable & reproducible

Containers without the hassle

Docker and Singularity integrations encapsulate all pipeline dependencies across environments.

Language agnostic

Write pipelines in your language

The flexibility to develop, port and reuse code allows you to do things your way.

nextflow *tower*

delivering discovery.

Manage, optimize and launch data analysis pipelines from a secure command-post.

Real time monitoring, anywhere

Keep up-to-date with progress.

Track workloads and retrieve a full history of executions across mobile and desktop browsers.

Launch workflows

Deploy on local, cluster, hybrid, cloud & serverless.

Execution of workflows via the GUI launchpad or API and submit to user-defined environments.

Team & organization management

Collaborate and share securely.

User management via enterprise providers such as LDAP and 3rd party authentication methods.

Advanced cloud-native storage

Reduce data transfer times.

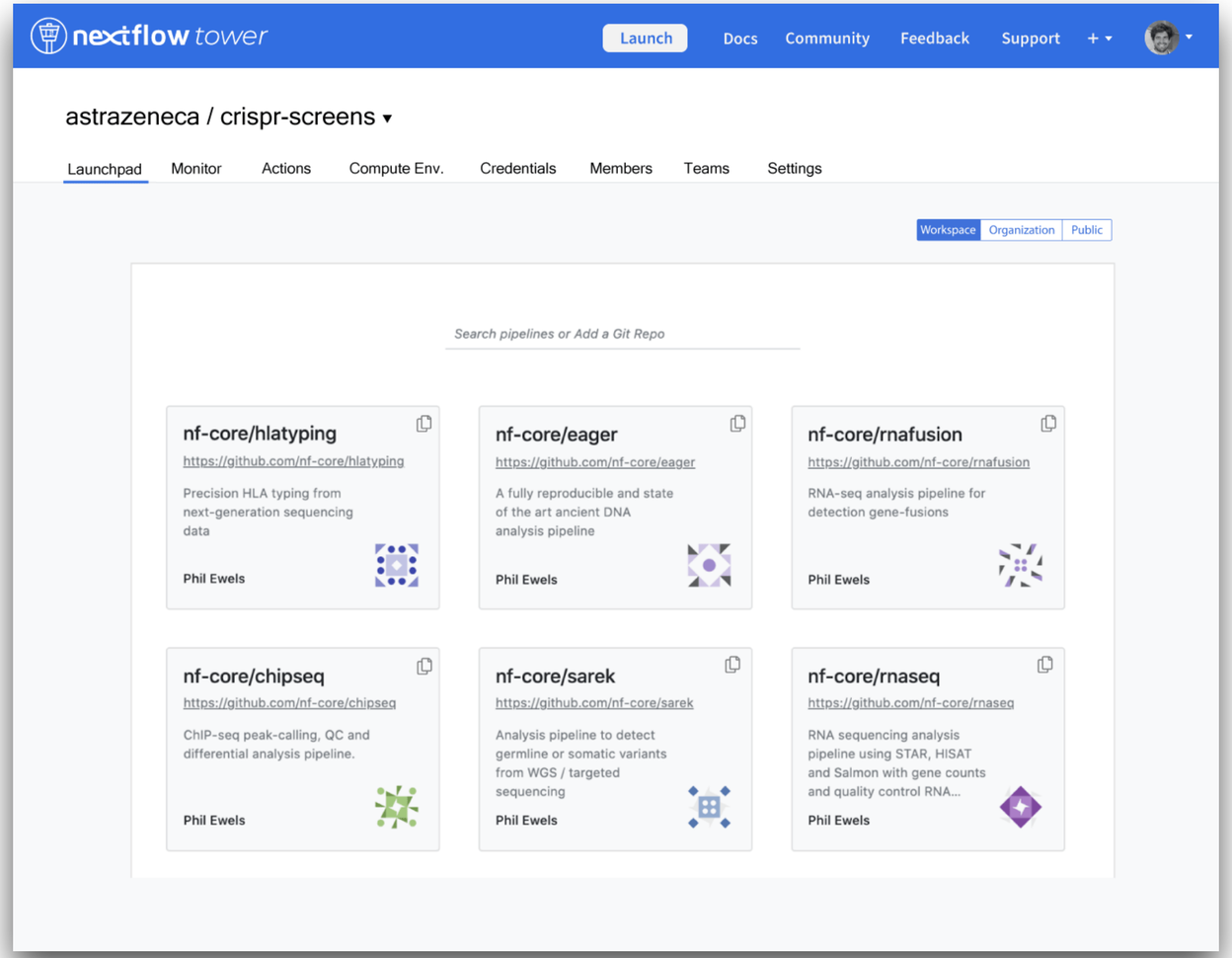
Significantly speed up workloads through the use of vendor-specific systems such as Lustre FSx.

Accounting & budgeting

Understand and control project costs.

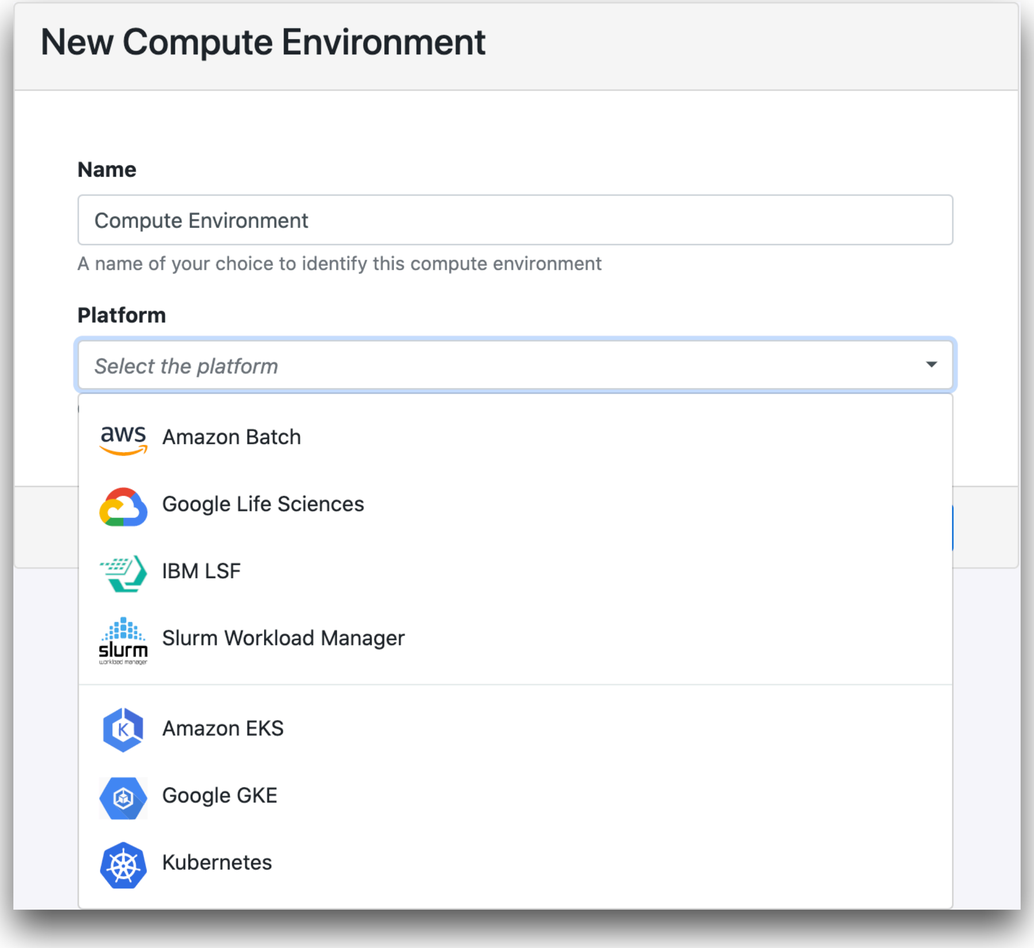
Estimates of compute costs and granular permissioning of resources across teams.

Build pipelines that run out-of-the-box. Anywhere.



Preconfigured pipelines with input data and forms based on JSON schema.

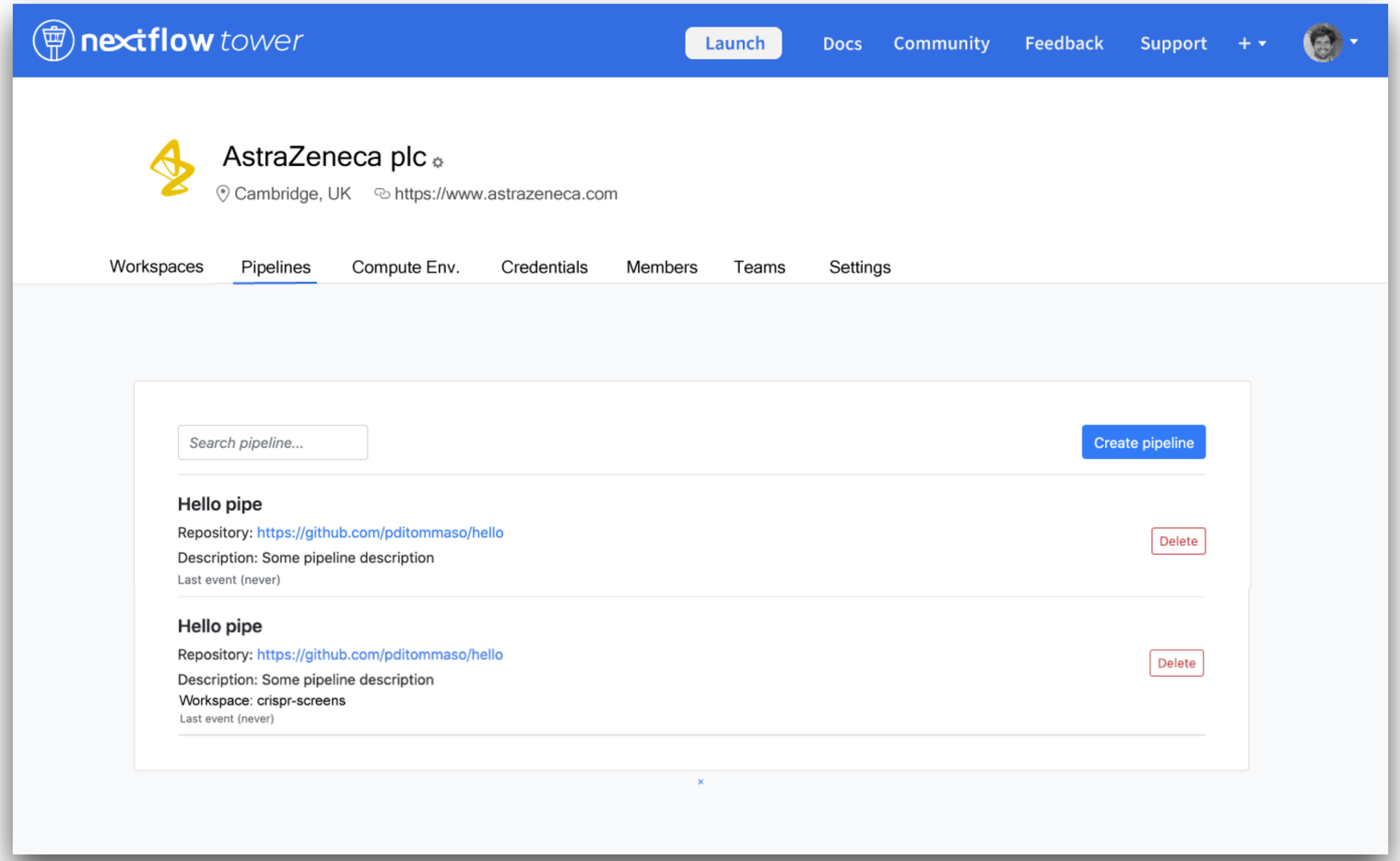
Manage cloud and on-prem environments in one place.



Deploy in any infrastructure with native support for all major cloud, Kubernetes & HPC platforms.

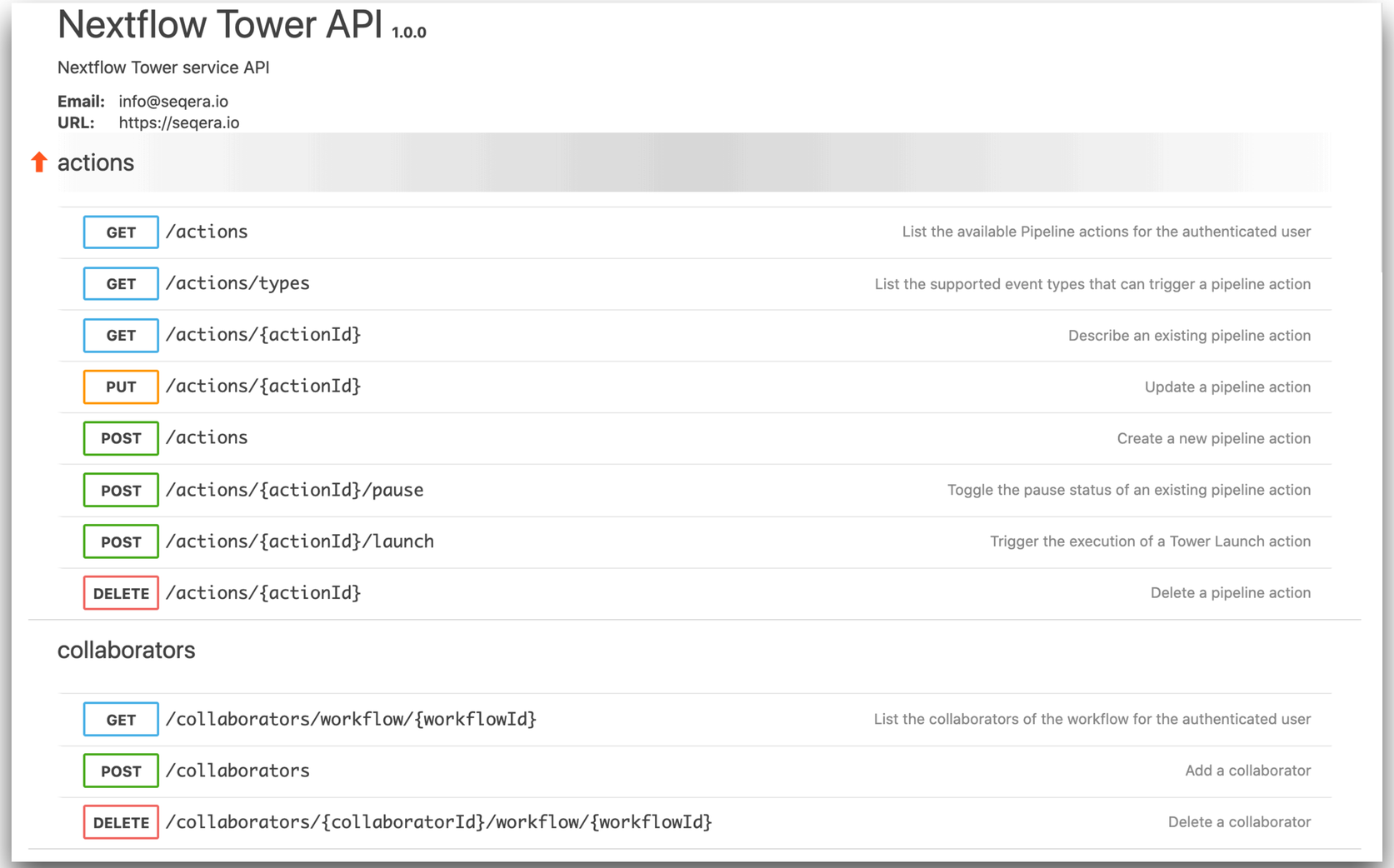
seqeralabs

Provide teams with the resources they need.



Role-based access for resource management across teams and organizations, plus security, compliance and cost control best practices.

Powerful API for integrations & automations



Nextflow Tower API 1.0.0

Nextflow Tower service API

Email: info@seqera.io
URL: https://seqera.io

actions

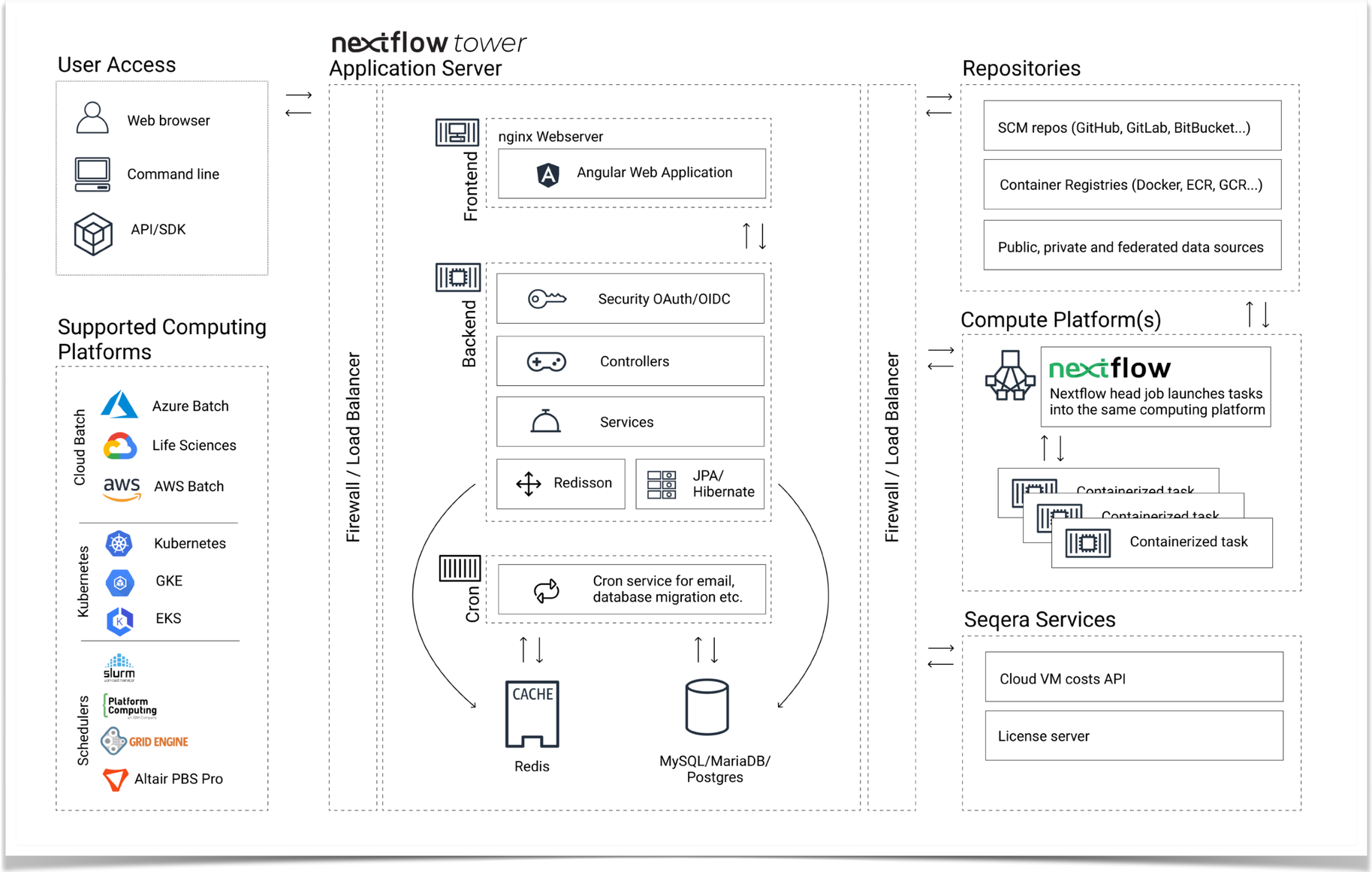
GET	/actions	List the available Pipeline actions for the authenticated user
GET	/actions/types	List the supported event types that can trigger a pipeline action
GET	/actions/{actionId}	Describe an existing pipeline action
PUT	/actions/{actionId}	Update a pipeline action
POST	/actions	Create a new pipeline action
POST	/actions/{actionId}/pause	Toggle the pause status of an existing pipeline action
POST	/actions/{actionId}/launch	Trigger the execution of a Tower Launch action
DELETE	/actions/{actionId}	Delete a pipeline action

collaborators

GET	/collaborators/workflow/{workflowId}	List the collaborators of the workflow for the authenticated user
POST	/collaborators	Add a collaborator
DELETE	/collaborators/{collaboratorId}/workflow/{workflowId}	Delete a collaborator

Tower is built with Micronaut - a modern, API-first microservices architecture.

Full stack solution for data pipelines at scale



Let's get started with the workshop!

https://www.seqera.io/training