Introduction_to_nextflow

2024, Jan 9th

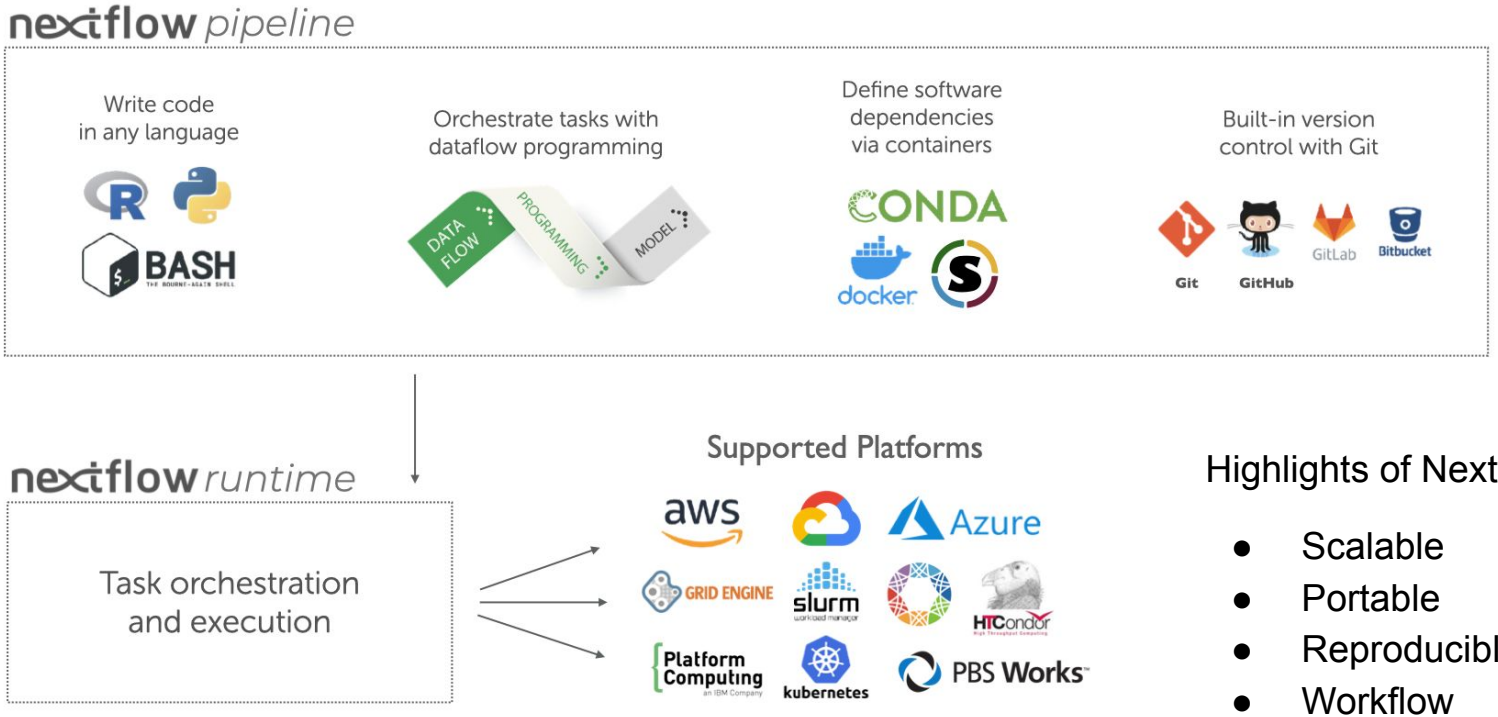
What is Nextflow?

WHAT'S NEXTFLOW?

nextflow

- Workflow Management Software (WMS)
  - Based on Groovy (Java)
  - Nextflow is free, open-source
  - Comes with lots of nice-to-have-features
    - Automates pipelines
    - Easy access to parallelization
    - Checkpointing
    - Submits jobs to schedulers (SLURM, MOAB, etc)
    - Compute resource management on non-HPC jobs
    - Docker/Singularity containerization

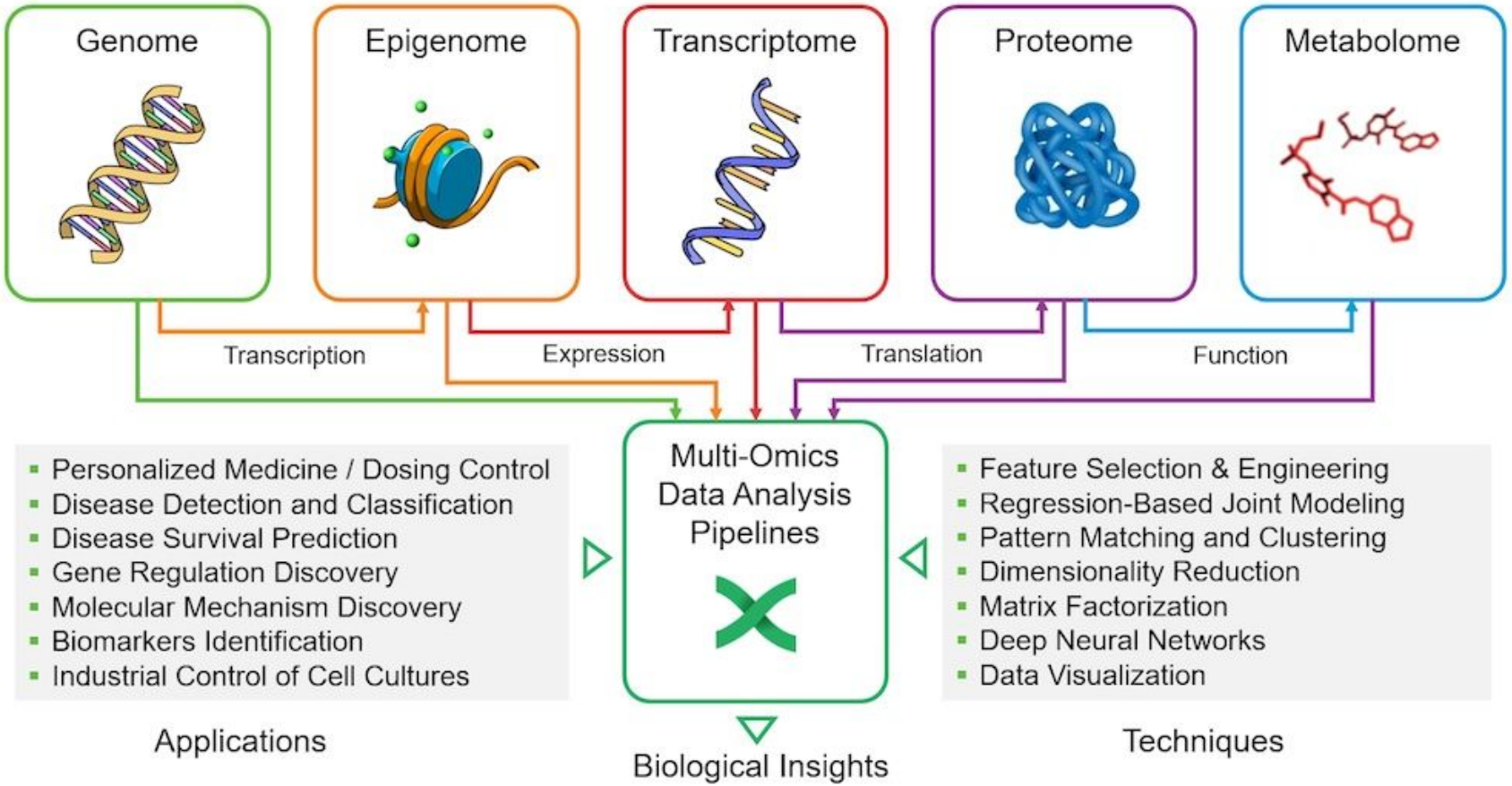
Nextflow introduction



Highlights of Nextflow:

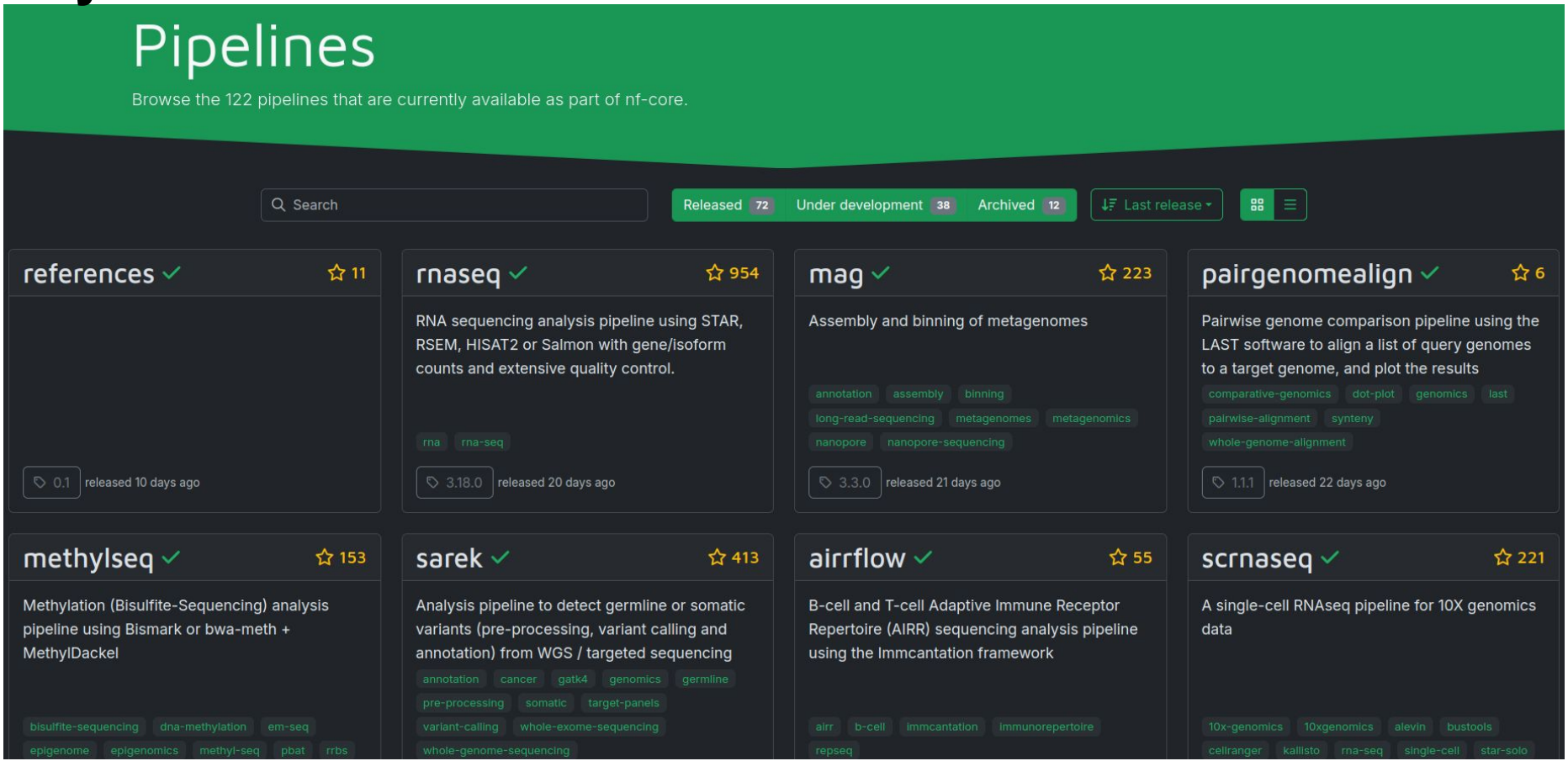
- Scalable
- Portable
- Reproducible
- Workflow orchestration

Why Nextflow?



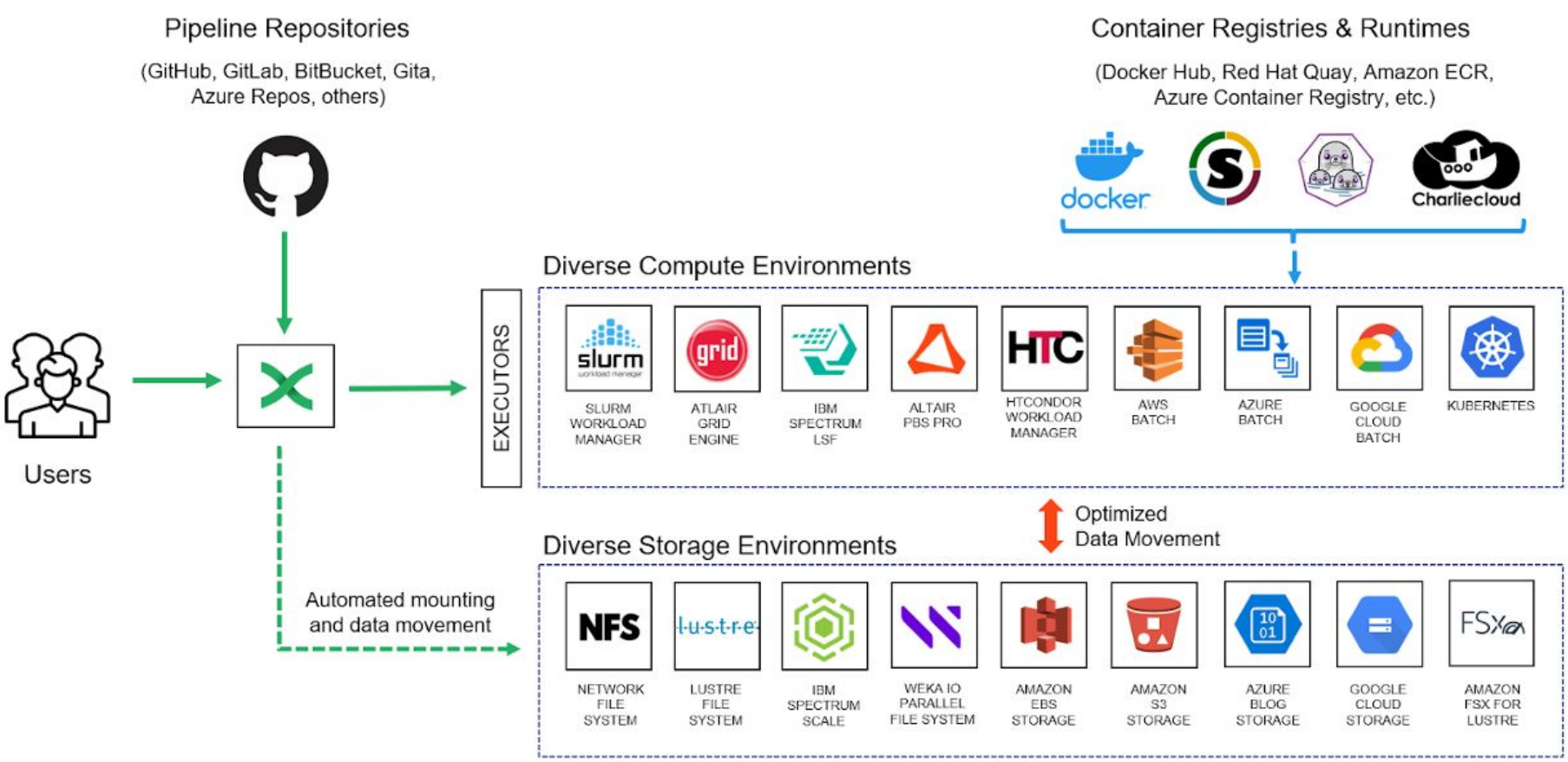
https://seqera.io/blog/nextflowomics-untangling-biology-with-data-analysis-pipelines/ https://nf-co.re/

Why Nextflow?



https://nf-co.re/pipelines/

How Nextflow Works



Domain-Specific Language (DSL)

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

// Define the input text file
params.input = "input.txt"

// Define a process for counting words
process countWords {
    input:
    path file

    output:
    path "word_count.txt"

    script:
    """
    wc -w $file > word_count.txt
    """
}
```

```
// Define a process for displaying the result
process printResult {
    input:
    path countFile

    script:
    """
    cat $countFile
    """
}

// Workflow definition
workflow {
    textFile = file(params.input)
    wordCountFile = countWords(textFile)
    printResult(wordCountFile)
}
```

Core Concepts - Processes

Definition: A **process** in Nextflow is a fundamental building block of a workflow. It defines an **independent computation step**, specifying what needs to be done (e.g., running a script or command). Processes are executed in isolation and can be run in parallel if resources allow.

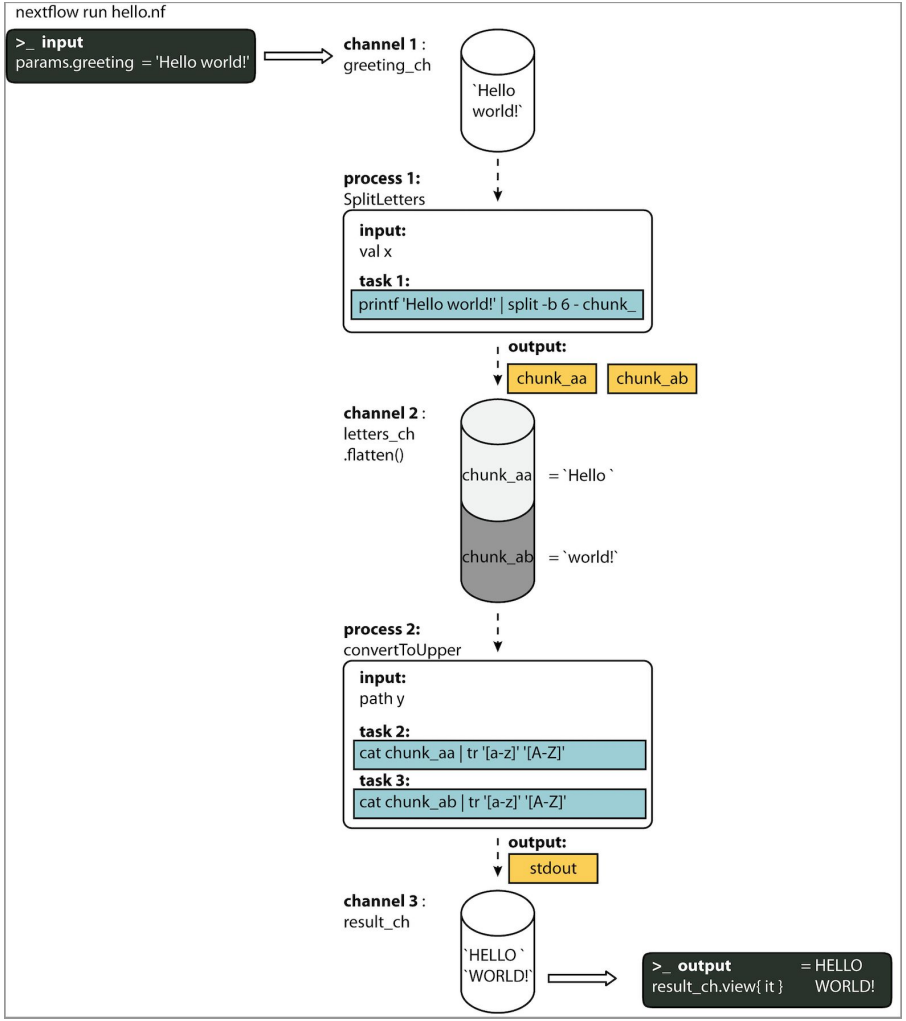
Key Features:

- Encapsulates computation logic.
- Can define input, output, and scripts.
- Supports execution on various platforms (e.g., local, HPC, cloud).

```
process countWords {
    input:
    path file  // Input file

    output:
    path "word_count.txt"  // Output file

    script:
    """
    wc -w $file > word_count.txt
    """
}
```



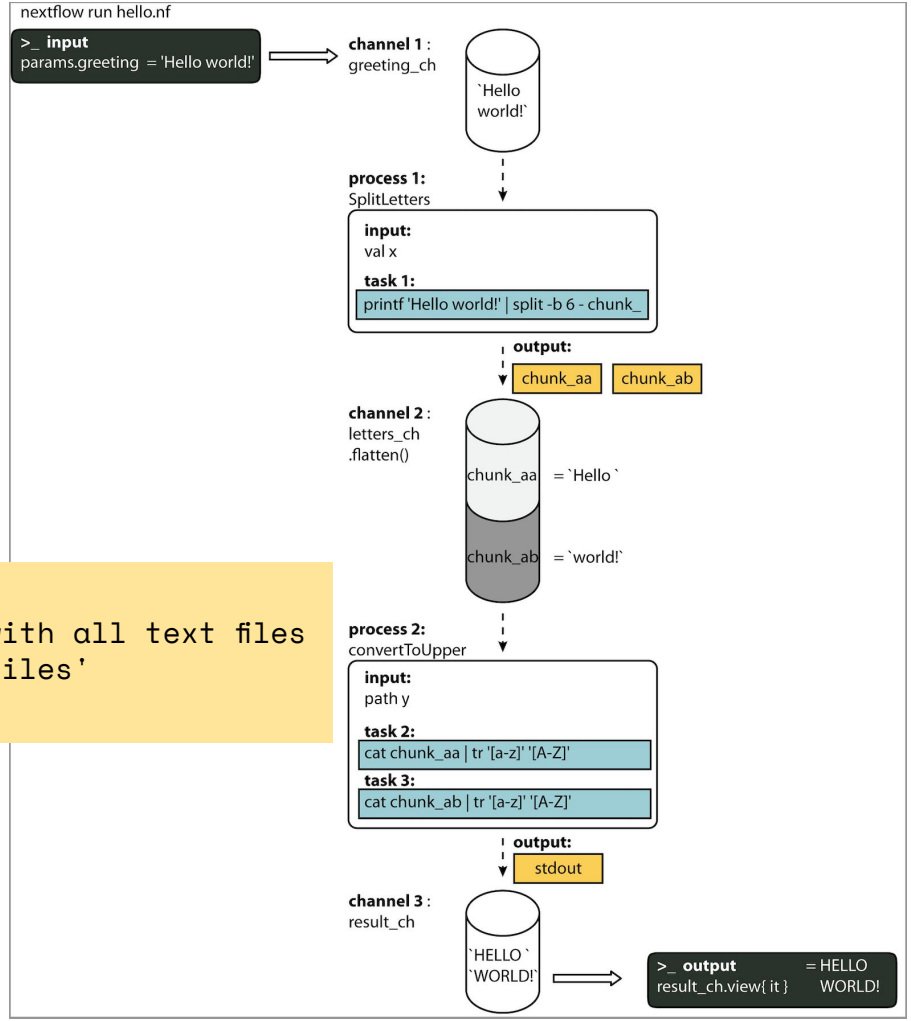
Core Concepts - Channels

Definition: A **channel** in Nextflow acts as a **data connector** between processes. It defines the **flow of data** from one process to another. Channels can carry files, strings, or other data objects, and they can be static or dynamic.

Types of Channels:

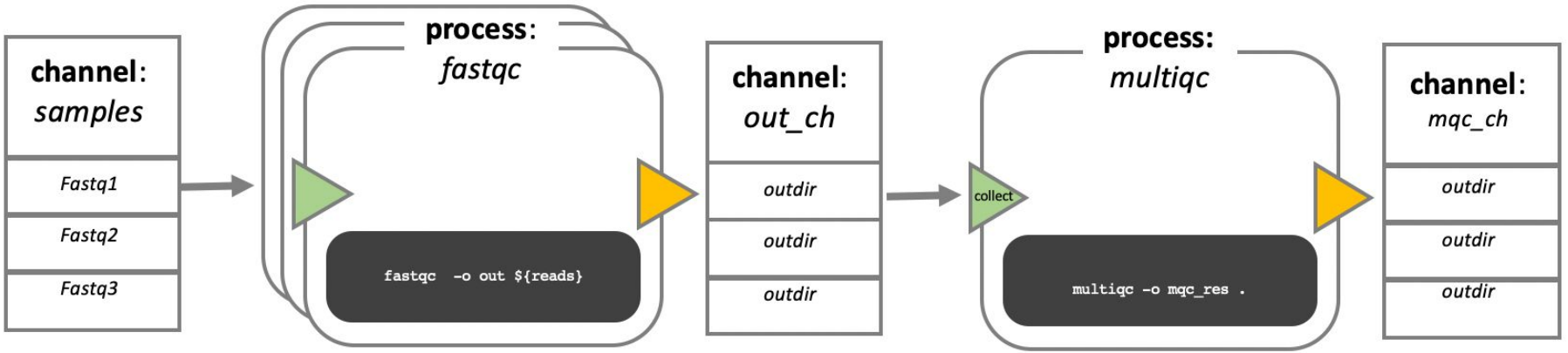
- **Emitter Channel:** Produces data (e.g., `fromPath`, `fromFile`).
- **Operator Channel:** Transforms data (e.g., `map`, `filter`).
- **Consumer Channel:** Passes data to a process.

```
Channel
    .fromPath('*.txt') // Emitter: produces a channel with all text files
    .set { textFiles } // Assigns the channel to 'textFiles'
```



Workflow - Process and Channel Work Together

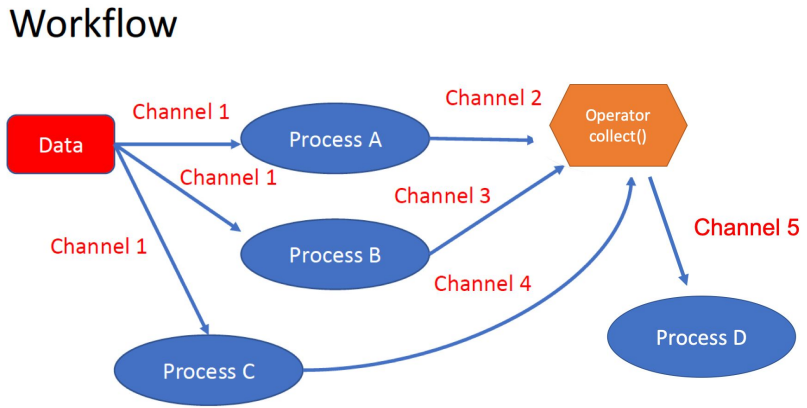
Processes operate on data received from channels, and their outputs are sent to channels, enabling downstream processes to consume the results



Workflow

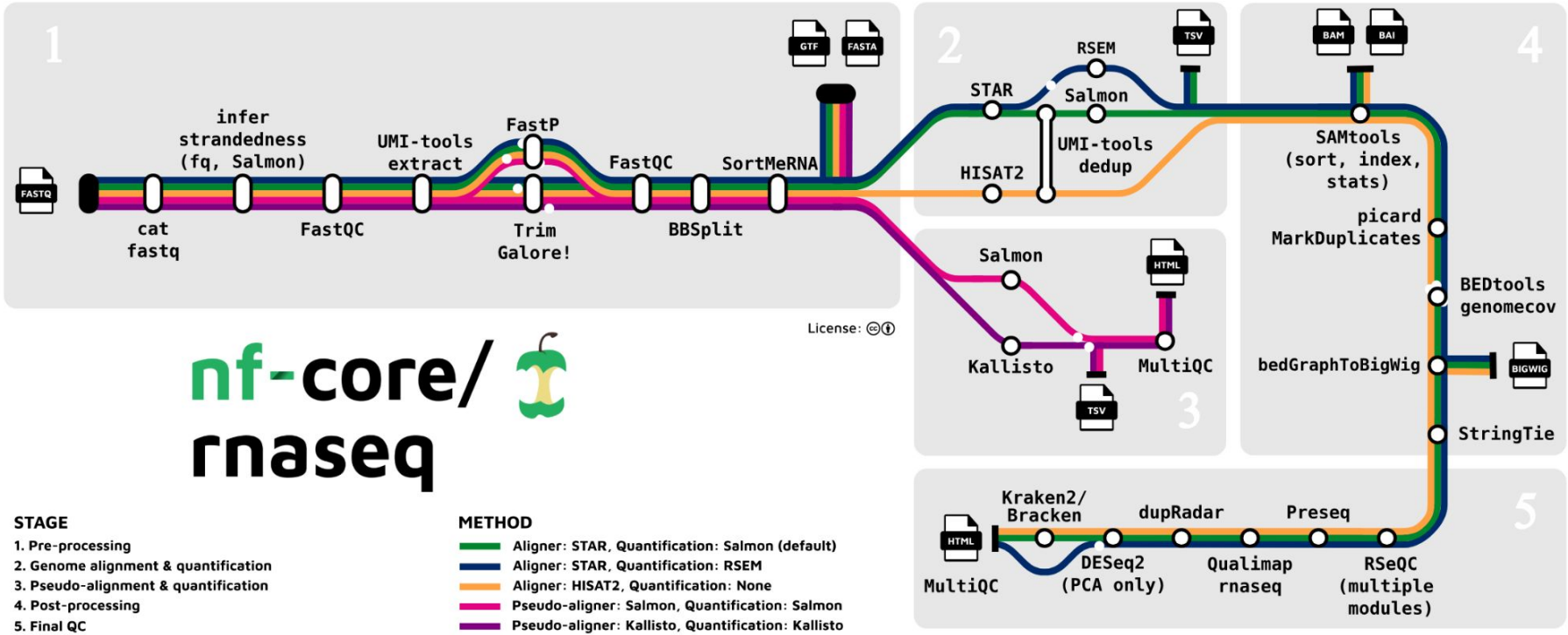
Definition: A **workflow** in Nextflow defines the overall **pipeline logic** by connecting **processes** and **channels**. It orchestrates the flow of data and tasks, specifying how processes should be executed in sequence or in parallel.
Key Features:

- Controls the execution of multiple processes.
- Encapsulates data flow using channels.
- Modular and can integrate subworkflows or other pipelines.



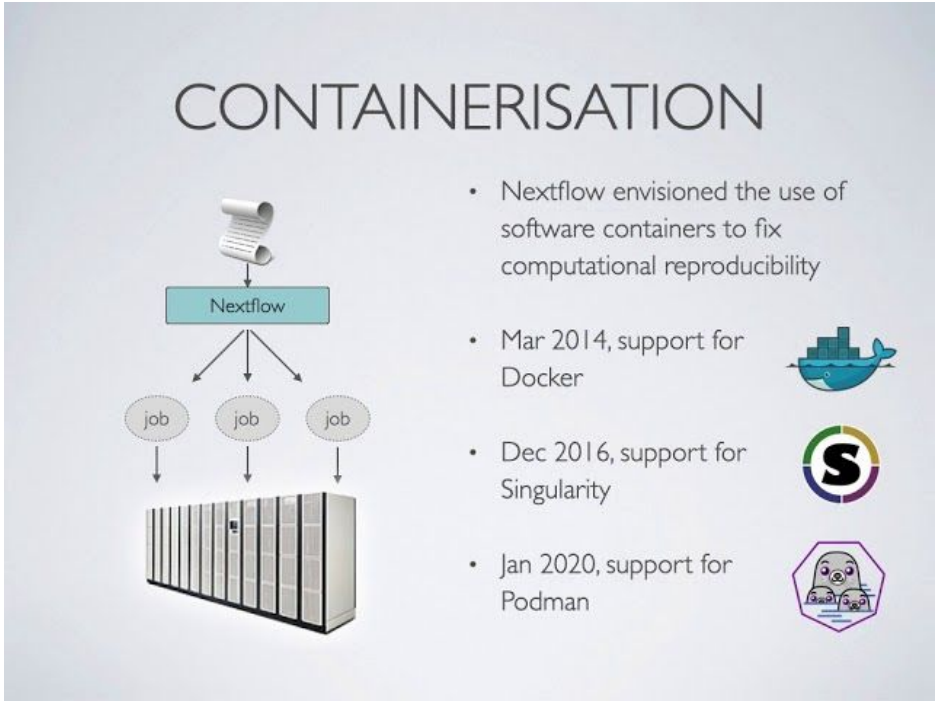

```
workflow {
    textFiles = Channel.fromPath('*.txt')  // Channel for input files
    wordCountResults = countWords(textFiles)  // Process execution
    printResult(wordCountResults)  // Downstream process
}
```

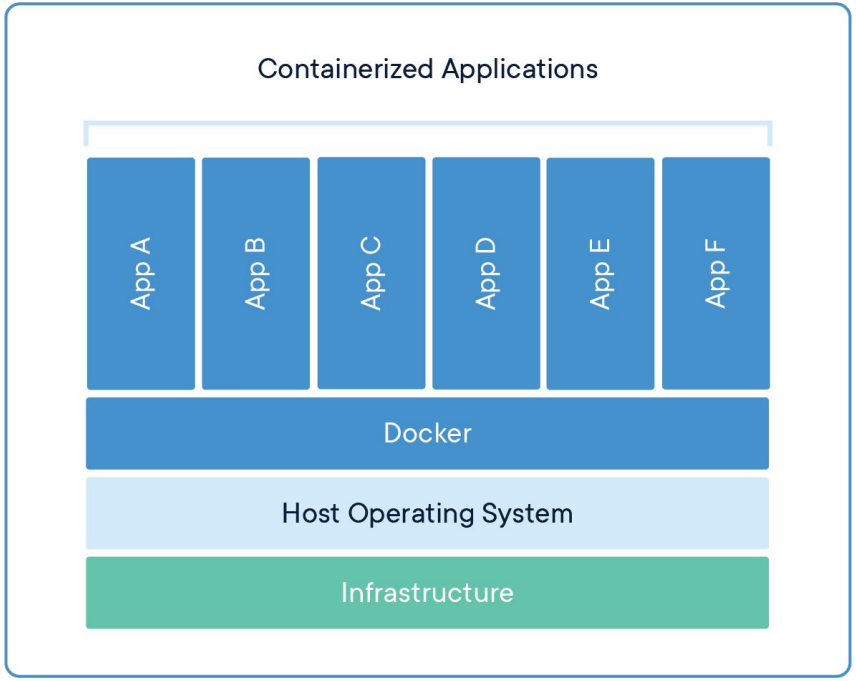
Workflow


1
FASTQ
cat fastq
infer strandedness (fq, Salmon)
FastQC
UMI-tools extract
FastP
Trim Galore!
FastQC
SortMeRNA
BBSplit
GTF
FASTA
2
STAR
RSEM
Salmon
TSV
HISAT2
UMI-tools dedup
3
Salmon
HTML
Kallisto
MultiQC
TSV
4
BAM
BAI
SAMtools (sort, index, stats)
picard MarkDuplicates
BEDtools genomecov
bedGraphToBigWig
BIGWIG
StringTie
5
Kraken2/ Bracken
dupRadar
Preseq
HTML
MultiQC
DESeq2 (PCA only)
Qualimap rnaseq
RSeQC (multiple modules)
License:
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

Containerization

Definition: Use of Docker/Singularity for reproducible environments.





https://hub.docker.com/

Learn Docker

Get started with Docker

- https://docs.docker.com/get-started/

Video for intuitive learning:

- https://www.youtube.com/watch?v=3c-iBn73dDE

Docker use case in bioinformatics:

- https://www.melbournebioinformatics.org.au/tutorials/tutorials/docker/media/#1

Nextflow Advantages

Reproducibility: Ensures workflows can be rerun with the same results

Portability: Nextflow workflows can run on different computational environments without modification

Scalability: Nextflow scales from local machines to cloud-based clusters

Version Control: Built-in support for Git integration to version workflows

Workflow Composition: Combining multiple workflows into modular pipelines

Skill Requirements

- Nextflow pipeline components
- Groovy
- Container (Docker, singularity)
- Git and Github
- Slurm
- HPC environment

Guideline for learning Nextflow

1. Watch these videos:

https://www.youtube.com/playlist?list=PL3xpfTVZLcNgLBGLAiY6Rl9fizsz-DTCT

https://nf-co.re/events/2024/training-foundational-march

2. Read this document:

https://www.nextflow.io/docs/latest/overview.html

3. Make a slide summarizing important Nextflow concepts

Exercise

1. **Install Nextflow and Docker**
   - Set up the necessary tools for executing Nextflow workflows and managing containers with Docker.
2. **Practice with Nextflow Training Session 1 and Review**
   - Follow the instructions in Session 1 of the Nextflow training and review the material to reinforce your understanding.
3. **Develop a Nextflow Process for the FastQC Tool**
   - Include the following components:
     - **Input/Output Channels**: Define the channels for data input and output.
     - **Process**: Write the script that specifies how the FastQC tool will execute within the Nextflow framework.
     - **Workflow**: Integrate the process into a workflow that orchestrates data processing.
     - **PublishDir**: Configure where the output files will be published.
     - **Nextflow Config**: Set up the configuration file with the necessary parameters and resources.
     - **Container**: Use a Docker or Singularity container to encapsulate the FastQC tool.