

# NextFlow

CMSE 890-402

# NextFlow

- Written in Groovy, a Python-like Java variant
  - Requires use of { } to define some code blocks!
- Uses “Channels” and “Processes”
- Handles conda and containers well
- Runs processes on local or remote systems as requested
  - Note: the main NextFlow process has to be able to run for the full time of the workflow!

[https://training.nextflow.io/basic\\_training/intro/](https://training.nextflow.io/basic_training/intro/)

# NextFlow Channels and Processes

## Channels

- Transfer information between processes
- Can be explicitly or implicitly created
- Can operate on channels
- Two types:
  - Queue- First in, first out asynchronous and unidirectional. Items are removed once used by a process.
  - Value- has a single value and can be read unlimited times.

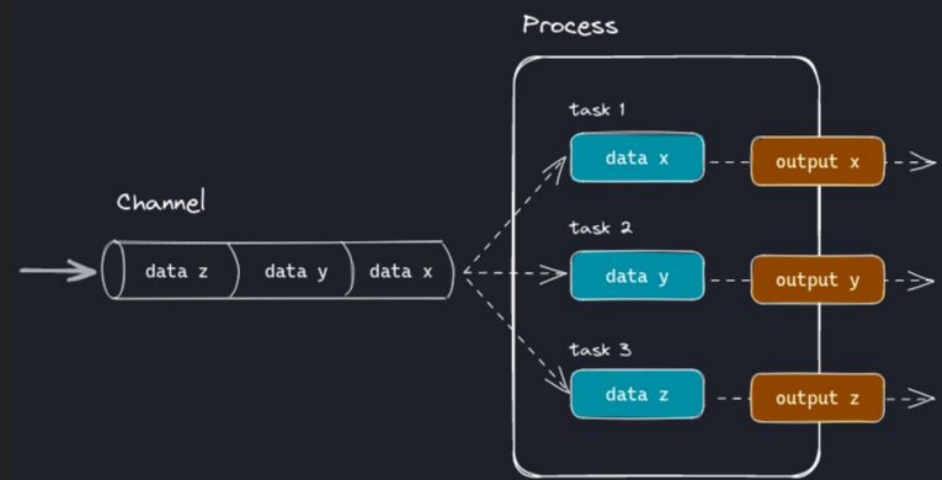
Can be thought of as the data flow in a DFD

[https://training.nextflow.io/basic\\_training/channels/](https://training.nextflow.io/basic_training/channels/)

## Processes

- Perform external operations on channels
- Runs scripts, software, etc

[https://training.nextflow.io/basic\\_training/processes/](https://training.nextflow.io/basic_training/processes/)



# NextFlow syntax

## *Queue Channel*

```
ch1 = Channel.of(1, 2, 3)
```

## *Value Channel*

Often implicitly created from variables

Explicitly:

```
ch1 = Channel.value()
ch2 = Channel.value('Hello there')
ch3 = Channel.value([1, 2, 3, 4, 5])
```

## *Process*

```
process < name > {
    [ directives ]

    input:
    < process inputs >

    output:
    < process outputs >

    when:
    < condition >

    [script|shell|exec]:
    """
    < user script to be executed >
    """
}
```

# Operators

- Perform some transformation on a Channel
- View its contents (print to screen), `view()`
- Apply a function to all elements of the Channel, `map()`
- Combine Channel elements together, `mix()`
- Join elements by a key, `join()`
- Split channel elements using boolean logic, `branch()`

[https://training.nextflow.io/basic\\_training/operators/](https://training.nextflow.io/basic_training/operators/)

# Modular NextFlow

- NextFlow scripts can be loaded with the `include { }` statement to bring in processes from other scripts
- Entire workflows can be defined in `workflow` blocks and called in the final `workflow` block of the script
- Workflows can take specific inputs and outputs to connect them to other workflows or processes

[https://training.nextflow.io/basic\\_training/modules](https://training.nextflow.io/basic_training/modules)

# Best practices

- Use the `-resume` option to use existing results and avoid re-running processes
- Use containers for processes (singularity on most HPC, Docker for cloud)
- Use an unnamed `process { }` block to define resource requirements for all processes

```
process {  
    executor = 'slurm'  
    queue = 'short'  
    memory = '10 GB'  
    time = '30 min'  
    cpus = 4  
}
```

# In-class assignment

We will go through the Nextflow “Hello world” workshop together:

[https://training.nextflow.io/basic\\_training/](https://training.nextflow.io/basic_training/)

Click “Open in GitPod”

Sign up with your GitHub account, or just watch along as I go through



# Homework

Work on your semester project!