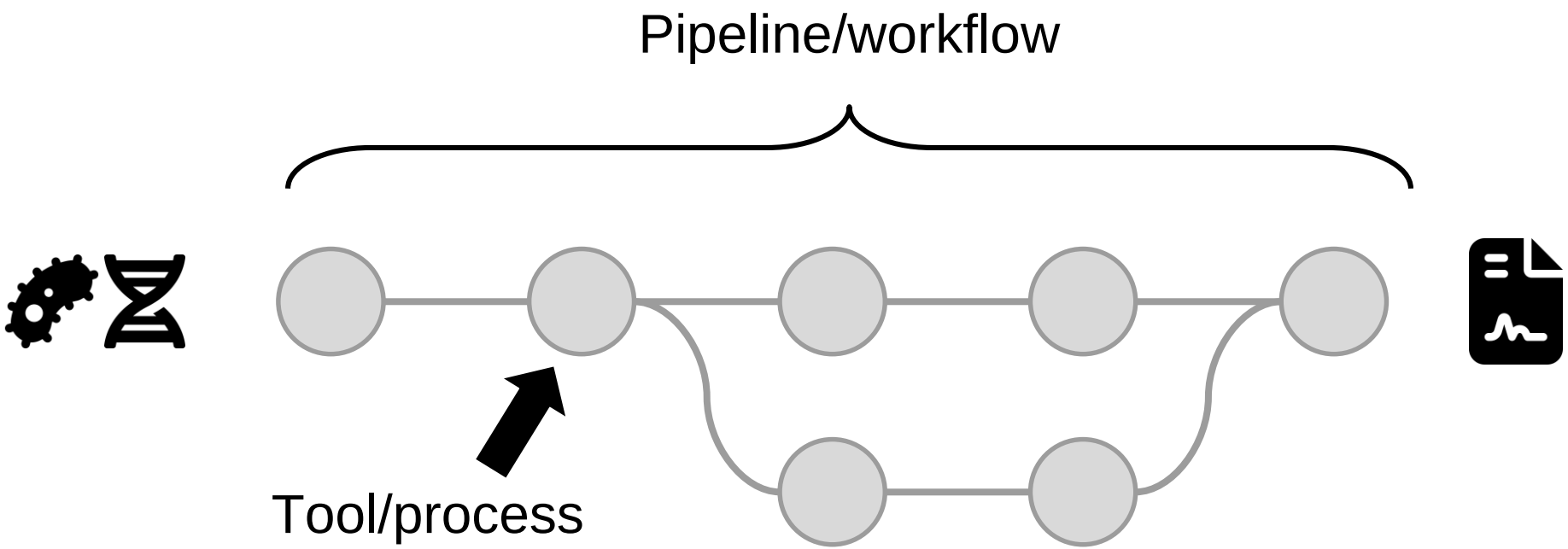
Public Health Agency of Canada
Agence de la santé publique du Canada

Canada

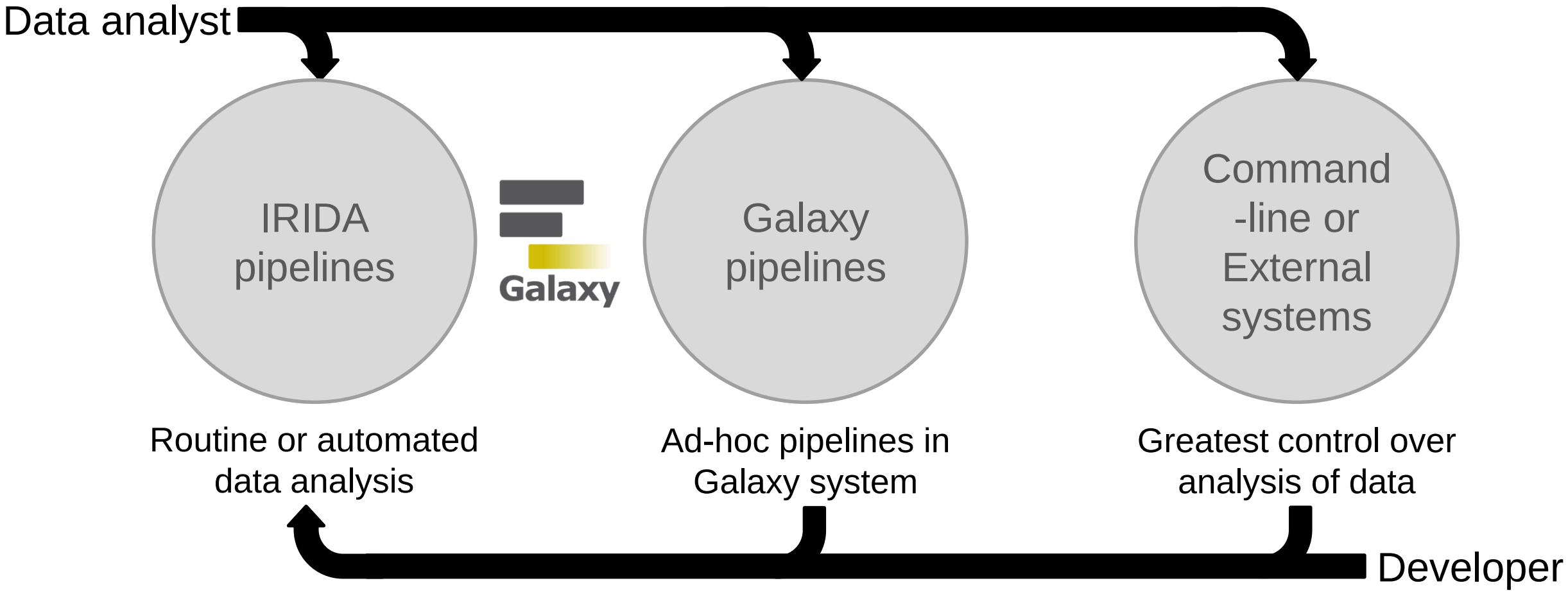
Pipeline development with Nextflow and nf-core

Aaron Petkau
Bioinformatics Lab Meeting
July 18, 2023

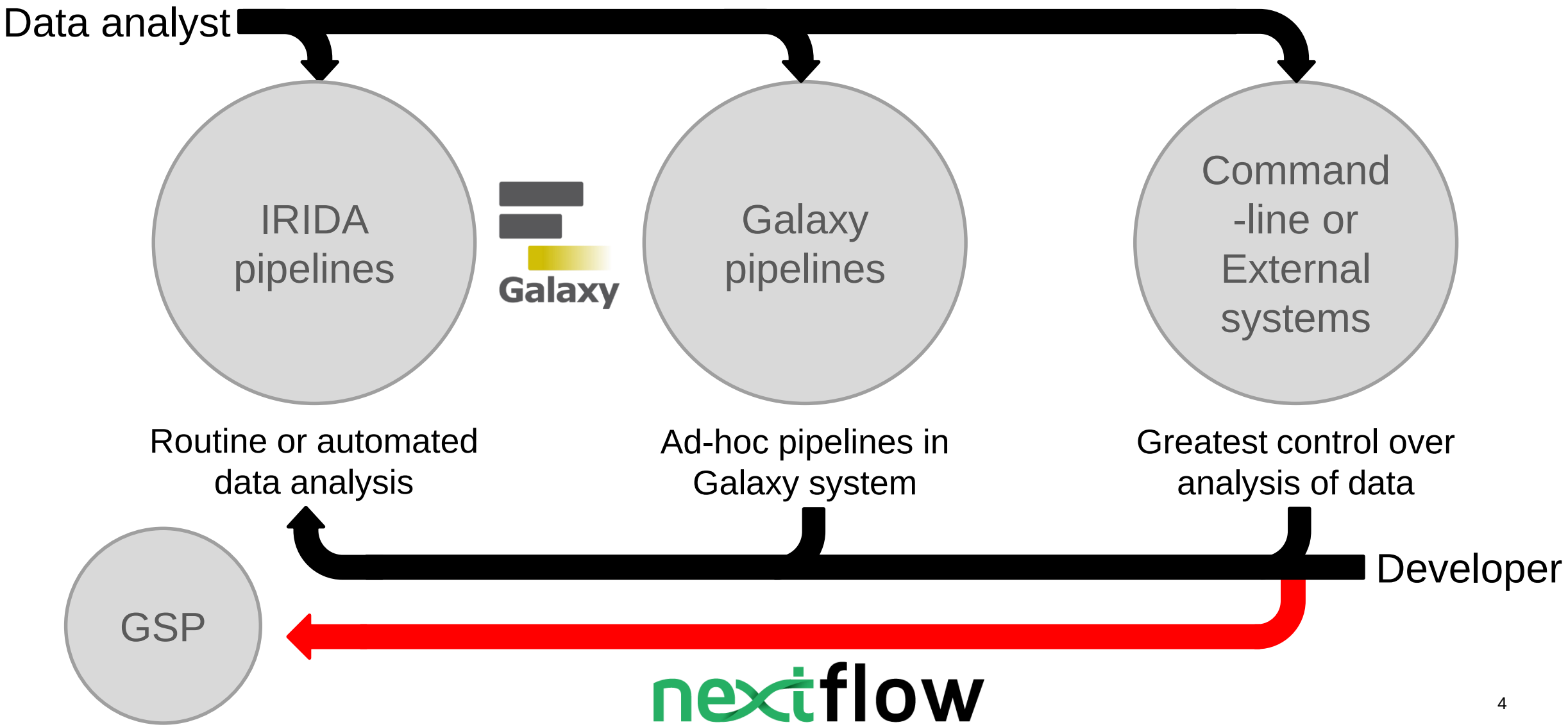
Analysis pipelines



Pipeline development with IRIDA



Pipeline development with IRIDA



Why Nextflow?

Table 1 | Overview of workflow managers for bioinformatics (top, editable version; bottom, image version)

Tool	Class	Ease of use[a]	Expressiveness[b]	Portability[c]	Scalability[d]	Learning resources[e]	Pipeline initiatives[f]
Galaxy	Graphical	●●●	●○○	●●●	●●●	●●●	●●○
KNIME	Graphical	●●●	●○○	○○○	●●◐	●●●	●●○
Nextflow	DSL	●●○	●●●	●●●	●●●	●●●	●●●
Snakemake	DSL	●●○	●●●	●●◐	●●●	●●○	●●●
GenPipes	DSL	●●○	●●●	●●○	●●○	●●○	●●○
bPipe	DSL	●●○	●●●	●●○	●●◐	●●○	●○○
Pachyderm	DSL	●●○	●●●	●○○	●●○	●●●	○○○
SciPipe	Library	●●○	●●●	○○○	○○○	●●○	○○○
Luigi	Library	●●○	●●●	●○○	●●◐	●●○	○○○
Cromwell + WDL	Execution + workflow specification	●○○	●●○	●●●	●●◐	●●○	●●○
cwltool + CWL	Execution + workflow specification	●○○	●●○	●●◐	○○○	●●●	●●○
Toil + CWL/ WDL/Python	Execution + workflow specification	●○○	●●●	●◐○	●●●	●●○	●●○

Wratten L, et al. (2021) "Reproducible, scalable, and shareable analysis pipelines with bioinformatics workflow managers". doi: 10.1038/s41592-021-01254-9

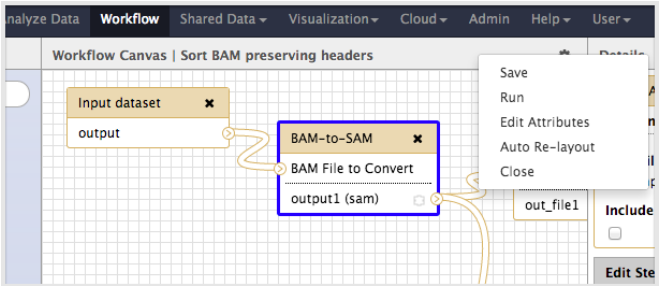
Why Nextflow?

Table 1 | Overview of workflow managers for bioinformatics (top, editable version; bottom, image version)

Tool	Class	Ease of use[a]	Expressiveness[b]	Portability[c]	Scalability[d]	Learning resources[e]	Pipeline initiatives[f]
Galaxy	Graphical	●●●	●○○	●●●	●●●	●●●	●●○
Nextflow	DSL	●●○	●●●	●●●	●●●	●●●	●●●

Ease of use

Galaxy > Nextflow



Expressiveness & Pipeline initiatives

Nextflow > Galaxy

```
if {
 // something
} else {
 // something else
}
```

Wratten L, et al. (2021) "Reproducible, scalable, and shareable analysis pipelines with bioinformatics workflow managers". doi: 10.1038/s41592-021-01254-9

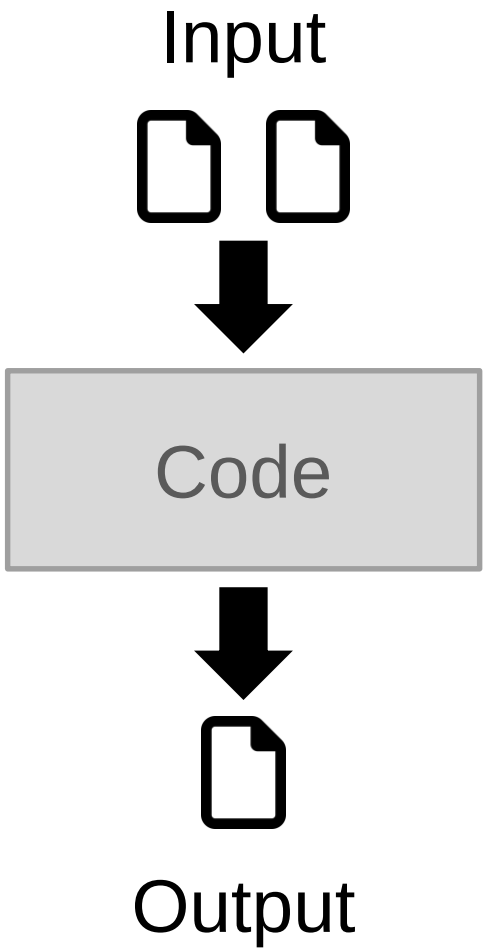
Nextflow

- A workflow development language
- An execution engine
  - Local, cluster, cloud
- Dependency management
- Community-maintained pipelines and modules
  - Nf-core

```
process sayHello {
  input:
    val cheers
  output:
    stdout

  """
  echo $cheers
  """
}

workflow {
  channel.of('Ciao','Hello','Hola') | sayHello | view
}
```

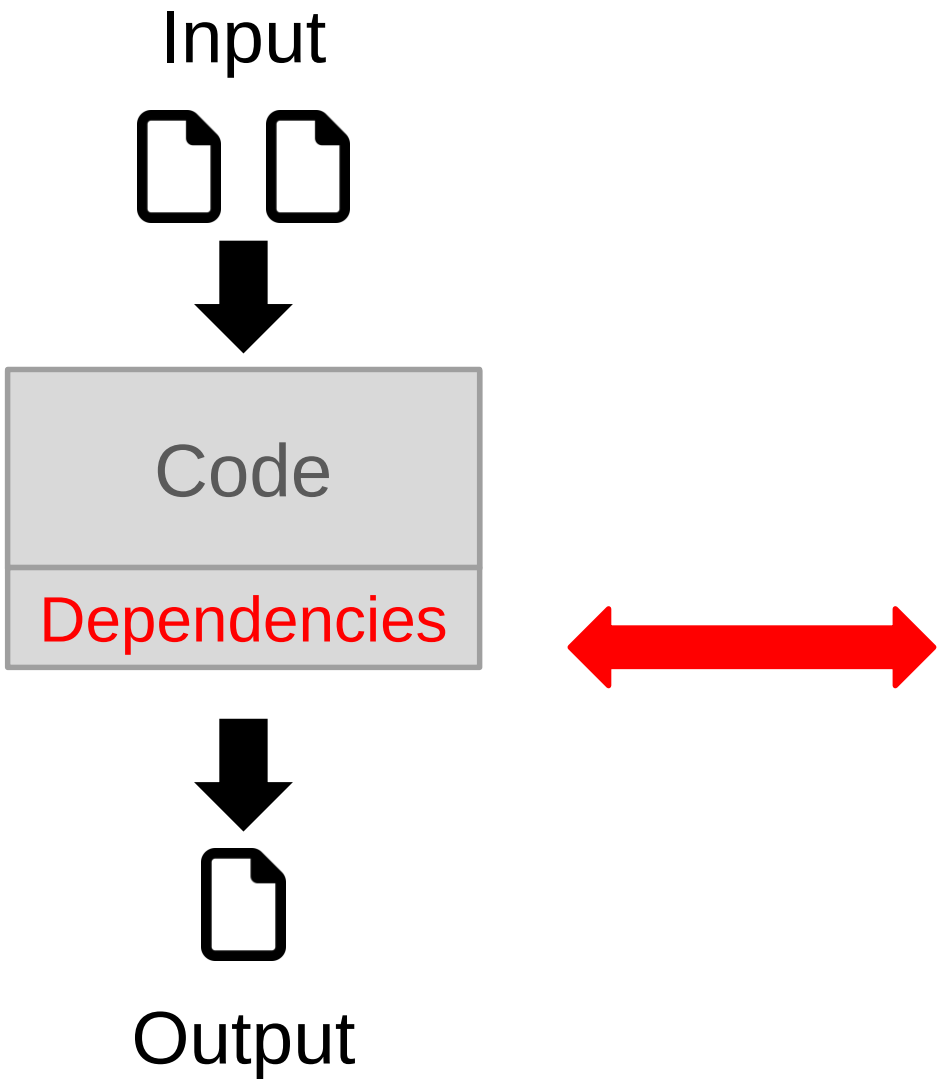
nextflow

Process




```
process MEGAHIT {
  input:
  tuple val(meta), path(fq)

  script:
  """
  megahit -1 ${fq[0]} -2 ${fq[1]}
  """

  output:
  path("${meta.id}.fasta")
}
```

Process




```
process MEGAHIT {
  input:
  tuple val(meta), path(fq)

  script:
  """
  megahit -1 ${fq[0]} -2 ${fq[1]}
  """

  container "megahit:1.2.9"

  output:
  path("${meta.id}.fasta")
}
```

Process

A **process** in Nextflow is similar to a **tool** in Galaxy

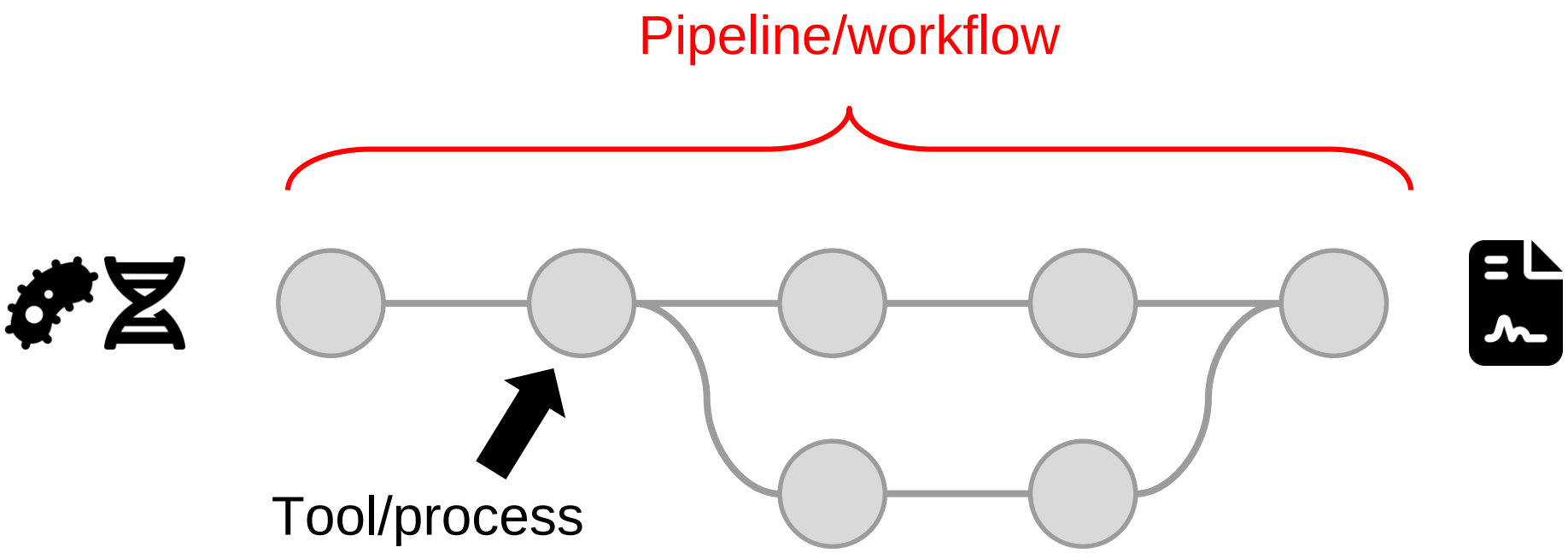
```
<tool id="megahit">
  <inputs>
    <param name="fq1"/>
    <param name="fq2"/>
  </inputs>

  <command>
  megahit -1 '${fq1}' -2 '${fq2}'
  </command>

  <requirements>…</requirements>

  <output>
    <data format="fasta"/>
  </output>
</tool>
```

```
process MEGAHIT {
  input:
  tuple val(meta), path(fq)

  script:
  """
  megahit -1 ${fq[0]} -2 ${fq[1]}
  """

  container "megahit:1.2.9"

  output:
  path("${meta.id}.fasta")
}
```

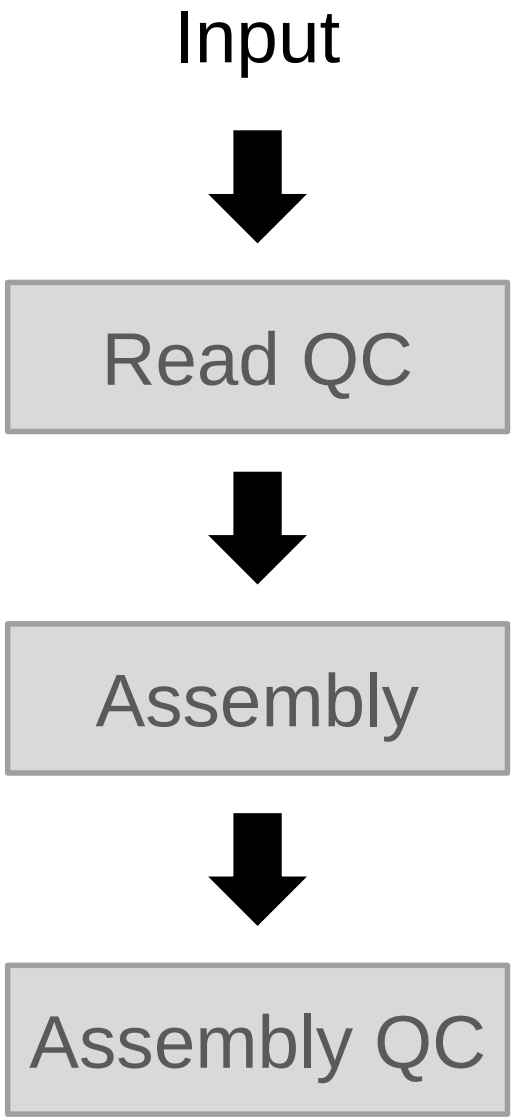
Galaxy

nextflow

Workflow



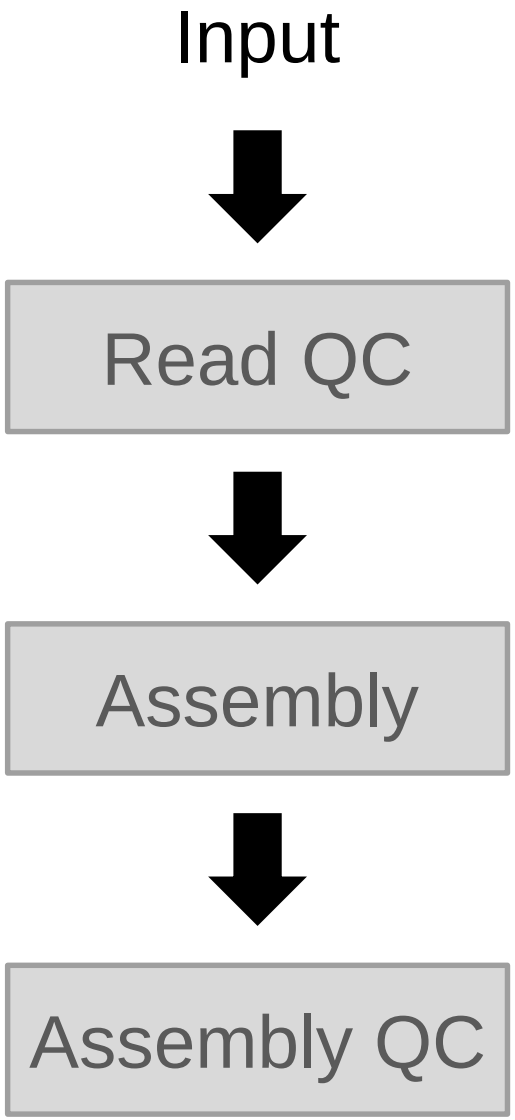
Workflow




```
workflow GENOME_ASSEMBLY {
  FASTP   ( reads )

  MEGAHIT( FASTP.out.reads)

  QUAST   ( MEGAHIT.out.contigs)
}
```

A workflow combines inputs and outputs of many tools together.

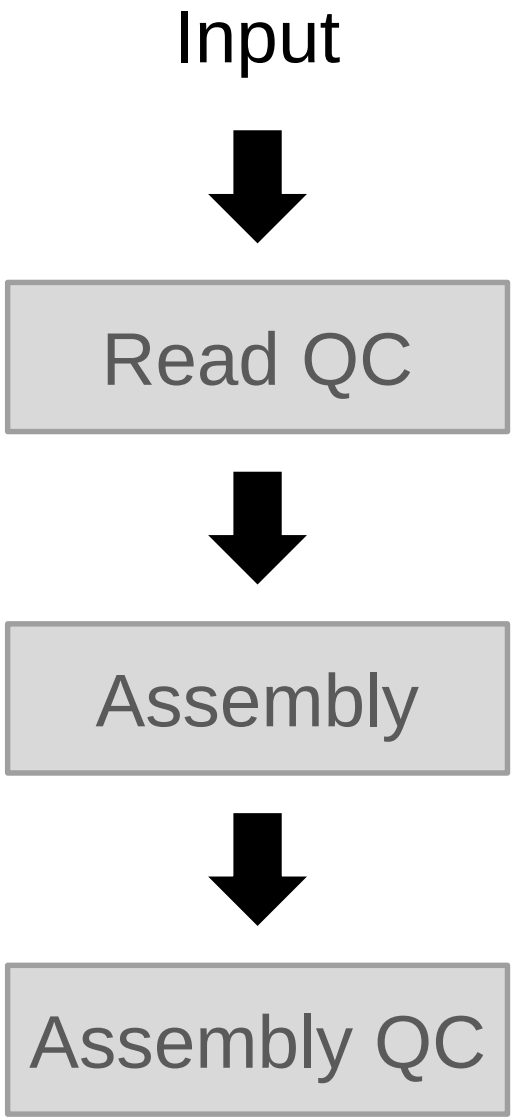
Workflow




```
workflow GENOME_ASSEMBLY {
  reads | FASTP | MEGAHIT | QUAST
}
```

An alternative syntax allows one to use a more familiar pipe “|” similar to commands on Unix/Linux.

Workflow




```
workflow GENOME_ASSEMBLY {
  reads = Channel.fromFilePairs(…)
  reads | FASTP | MEGAHIT | QUAST
}
```

"reads" is a **channel**, used to communicate messages (in this case, pairs of fastq files).

Configuration

```
process {
  executor = 'local'
  cpus = 4
}
```

Configuration defined in "`nextflow.config`" file by default.

Lets you control aspects of execution, like running on a local computer…

Configuration

```
process {
  executor = 'slurm'
  cpus = 4
}
```

Configuration defined in "`nextflow.config`" file by default.

Lets you control aspects of execution, like running on a local computer…

…or on a cluster (slurm).

Configuration

```
process {
  executor = 'local'
  cpus = 4
}

profiles {
  singularity {
    singularity.enabled = true
  }

  docker {
    docker.enabled = true
  }
}
```

Profiles let you group different configurations and enable/disable on command-line.

Running a pipeline

```
$ nextflow run main.nf -profile singularity
```

```
N E X T F L O W  ~  version 23.04.1
Launching `./main.nf` [festering_hoover] DSL2 -
revision: aa73df2297
executor >  local (9)
[e2/67f58b] process > FASTP (3)    [100%] 3 of 3 ✓
[3a/22e96a] process > MEGAHIT (3) [100%] 3 of 3 ✓
[89/fcfbed] process > QUAST (2)    [100%] 3 of 3 ✓
```

github.com/apetkau/assembly-nf

Running a pipeline

- Singularity images stored in "`work`" directory (by default)
- Each process gets a separate directory
- Input is linked (or copied) into process directory
- "`.command.sh`" contains the command
- "`.command.run`" contains setup code for running command
- Running "`bash .command.run`" can be used to debug a process
- The "`results`" output directory of published files.
- Running "`nextflow … -resume`" will re-use existing output files

Nf-core

nf-core

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

Pipelines

Modules (tools)

Templates

Creating a new pipeline with nf-core

```
$ nf-core create
  --name assemblyexample
  --description "Example assembly pipeline"
  --plain
  --author "Aaron Petkau"
```

Creates a template directory for nf-core pipelines.

We will update this to incorporate our previous pipeline.

Code in github.com/apetkau/nf-core-assemblyexample

Step 1: Run template pipeline

```
$ nextflow run main.nf
  --input samplesheet.csv --outdir results
  -profile singularity
  --genome hg38
```

sample	fastq_1	fastq_2
08-5578-small	example-data/reads/08-5578-small_1.fastq.gz	example-data/reads/08-5578-small_2.fastq.gz
08-5923-small	example-data/reads/08-5923-small_1.fastq.gz	example-data/reads/08-5923-small_2.fastq.gz
hcc23-small	example-data/reads/hcc23-small_1.fastq.gz	example-data/reads/hcc23-small_2.fastq.gz

Requires a “samplesheet.csv” file listing samples and fastqs.

Step 2: Add processes

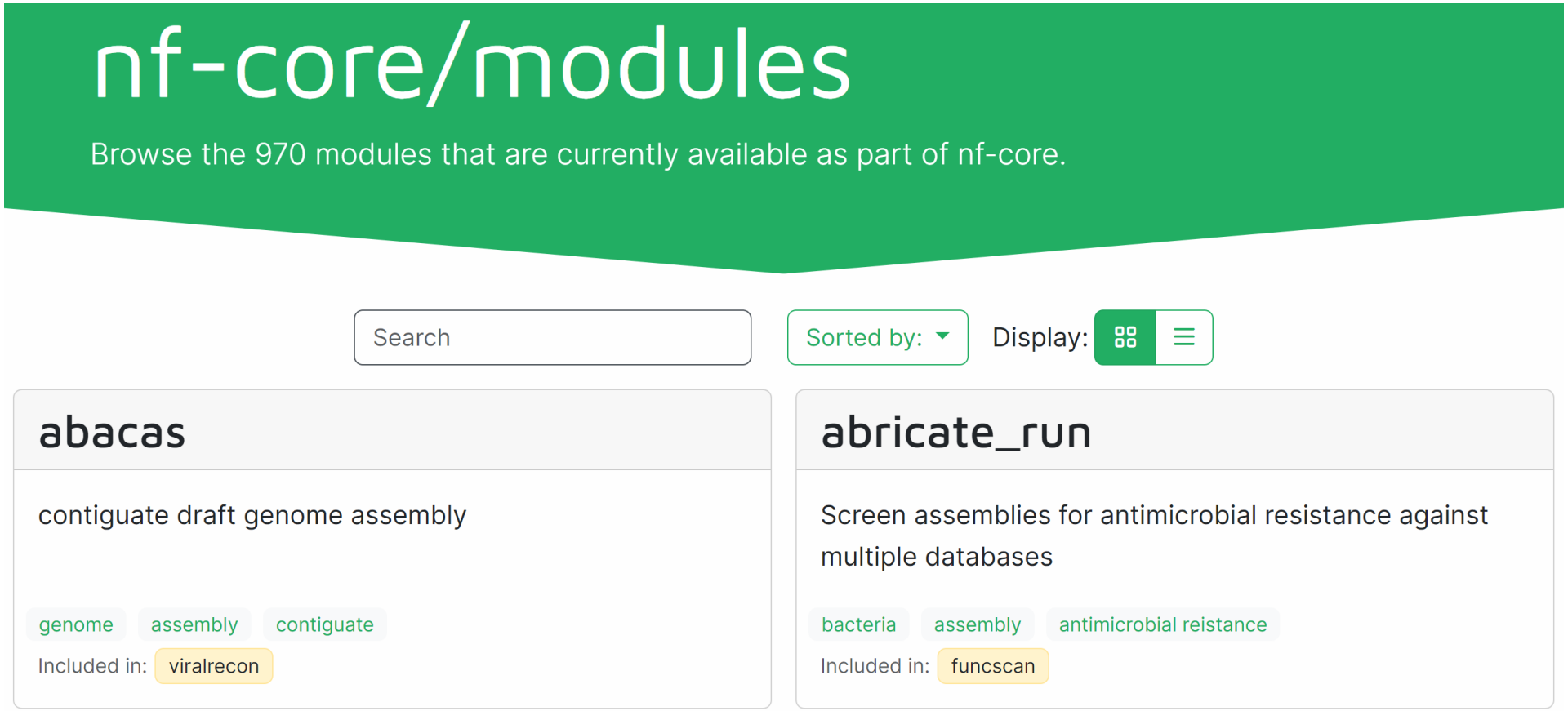
- Processes are added in the "`modules/local/`" directory
  - One per file (e.g., "`fastp.nf`", "`megahit.nf`", "`quast.nf`")
- Import and add to workflow in "`workflows/assemblyexample.nf`"

Name
..
fastp.nf
megahit.nf
quast.nf
samplesheet_check.nf

```diff
@@ -82,6 +85,19 @@ workflow ASSEMBLYEXAMPLE {
 82  85      )
 83  86      ch_versions = ch_versions.mix(FASTQC.out.versions.first())
 84  87
     88  +   // Adding assembly workflow steps here
     89  +   FASTP (
     90  +       INPUT_CHECK.out.reads
     91  +   )
     92  +
     93  +   MEGAHIT (
     94  +      FASTP.out.reads
     95  +   )
     96  +
     97  +   QUAST (
     98  +       MEGAHIT.out.contigs
     99  +   )
    100  +
 85  101     CUSTOM_DUMPSOFTWAREVERSIONS (
 86  102         ch_versions.unique().collectFile(name: 'collated_versions.yml')
 87  103     )
```

Step 3: Switch to community-maintained modules

- The nf-core community maintains a collection of reusable modules
- These can be incorporated into your workflow

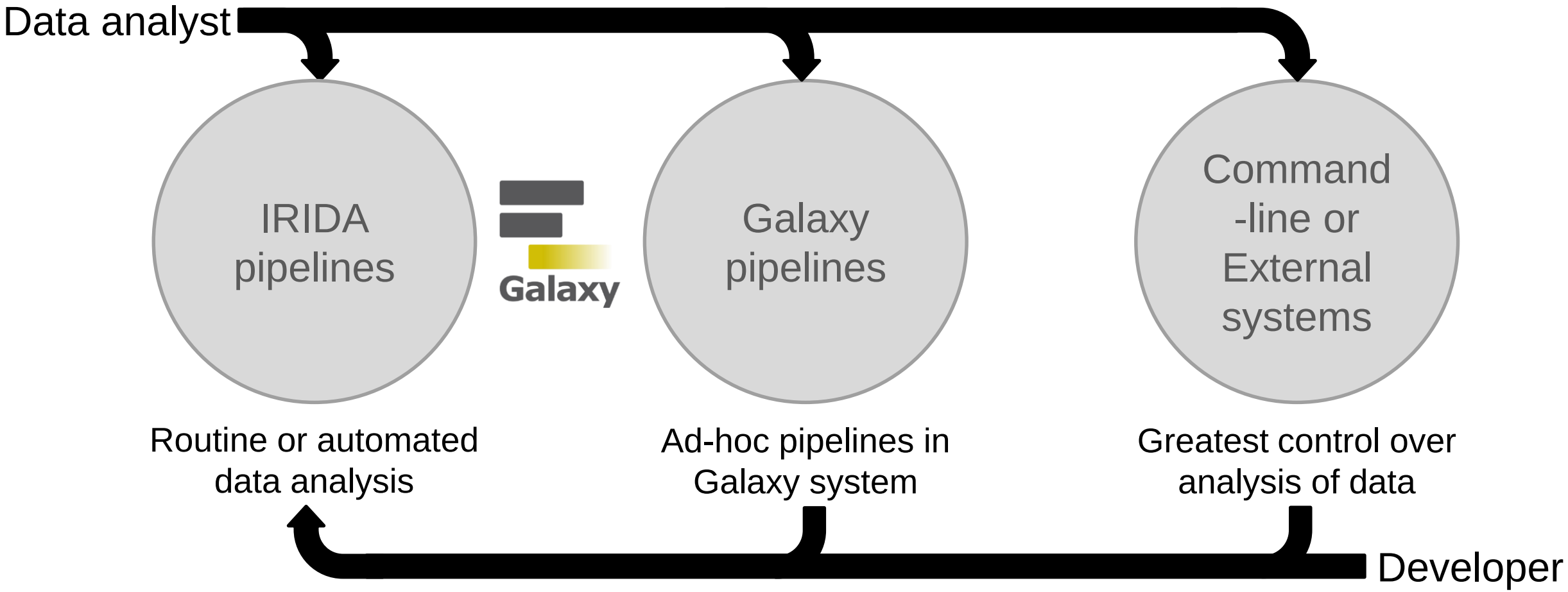


Step 4: Adjusting parameters

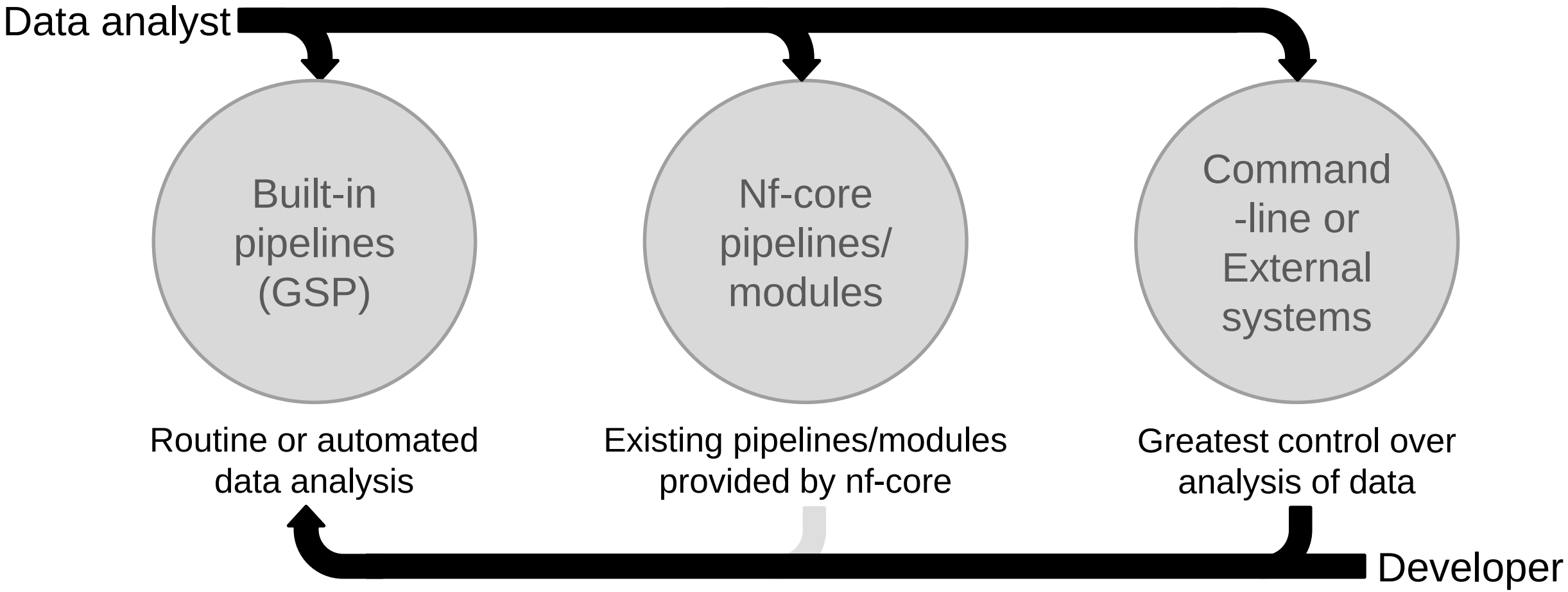
- Parameter adjustments are made in “`nextflow.config`”

```
// Global default params, used in configs
params {

    // TODO nf-core: Specify your pipeline's command line flags
    // Input options
    input                      = null
    // References
    genome                     = 'hg38'
    igenomes_base              = 's3://ngi-igenomes/igenomes'
    igenomes_ignore            = false


    // MultiQC options
    multiqc_config             = null
    multiqc_title              = null
```

Pipeline development with IRIDA



Future pipeline development



Using Nextflow, pipelines or modules could be added to nf-core, but don't have to be.

Conclusion

- **Nextflow** is a workflow language and execution environment
- **Nf-core** provides a community for developing and sharing pipelines
- Positives
  - Easy command-line interface for running pipelines
  - Flexible language for pipelines and configuration
- Negatives
  - A bit confusing on how to adapt nf-core template to my use cases