Processes - Directives

Adjust context and configuration

```
process align {

    cpus 1
    memory "200MB"
    // executor "local"
    executor "pbs"
    // queue "diag"
    beforeScript "source file.sh"
    tag "example ${id}"
    label "example_dir1"
    echo true
    conda "bioconda::bwa=1.19"
    // conda "bwa" // do not!

    input:  [...]
    output: [...]
    script: [...]
}
```

Organize your results

```
process foo {
    publishDir './data/chunks' , mode: 'copy', overwrite: false

    output:
    path 'chunk_*'

    """
    printf 'Hola' | split -b 1 - chunk_
    """
```

example7.nf

Skeleton

```
workflow [ name ] {

    take:
    < workflow inputs >

    main:
    < dataflow statements >

    emit:
    < workflow outputs >

}
```

Workflows are composition of processes

Workflows access any variables from the global scope (main.nf)

```
workflow my_pipeline {
    foo()
    bar( foo.out.collect() )
}

workflow {
    my_pipeline()
}
```

```
workflow my_pipeline {
    take:
    data1
    data2

    main:
    foo(data1, data2)
    bar(foo.out)

    emit:
    bar.out
}
```

Configuration

```
## DSL2

assets/
bin/
conf/
docs/
recipes/
test/
nf-modules/common/process
nf-modules/common/subworkflow
nf-modules/local/process
nf-modules/local/subworkflow
main.nf
nextflow.config
```

cluster.config
base.config
process.config

```
process {
  executor = 'slurm'
  queue = params.queue ?: null
}
```

```
includeConfig 'conf/base.config'

// Profiles
profiles {
  cluster {
    includeConfig 'conf/cluster.config'
  }
}
```

Configuration

Configuration is related to scope

main.nf

```
process toto {

label "someLabel"

when:
go = params.flag ?: false

script:
    """
    bash script.sh $myvar

    """
}
```

nextflow.config

```
env.myvar = "${baseDir}/somePath/boum.txt"
params.flag = true
process {
    executor = "pbs"
    withLabel: "someLabel" {
        cpus = 1
        memory = "200MB"
    }
}
```

Configuration – Execution order

1. main.nf
2. $HOME/.nextflow/config
3. nextflow.config (project directory)
4. nextflow.config (launch directory)
5. -c <config-file>
6. -params-file
7. --something value

Configuration - Profile

nextflow run <your script> -profile standard,cloud

```
profiles {

    standard {
        process.executor = 'local'
    }

    cluster {
        process.executor = 'sge'
        process.queue = 'long'
        process.memory = '10GB'
    }

    cloud {
        process.executor = 'cirrus'
        process.container = 'cbcrg/imagex'
        docker.enabled = true
    }
}
```

Error Management

```
process example {

    errorStrategy "terminate"    // kill all submitted jobs [default]
    errorStrategy "finish"       // let spawn job finished
    errorStrategy "retry"        // requeue the job
    errorStrategy "ignore"       //go to next process
    // with closure
    errorStrategy { sleep(Math.power(2, task.attempt) * 30 as long); return retry }

    maxRetries 5 //when errorStrategy = retry, single process
    maxErrors 2  //total number of errors (all process instances)
    validExitStatus 0,1

    memory { 2.GB * task.attempt }

}
```

retry should not be a strategy.

Debugging

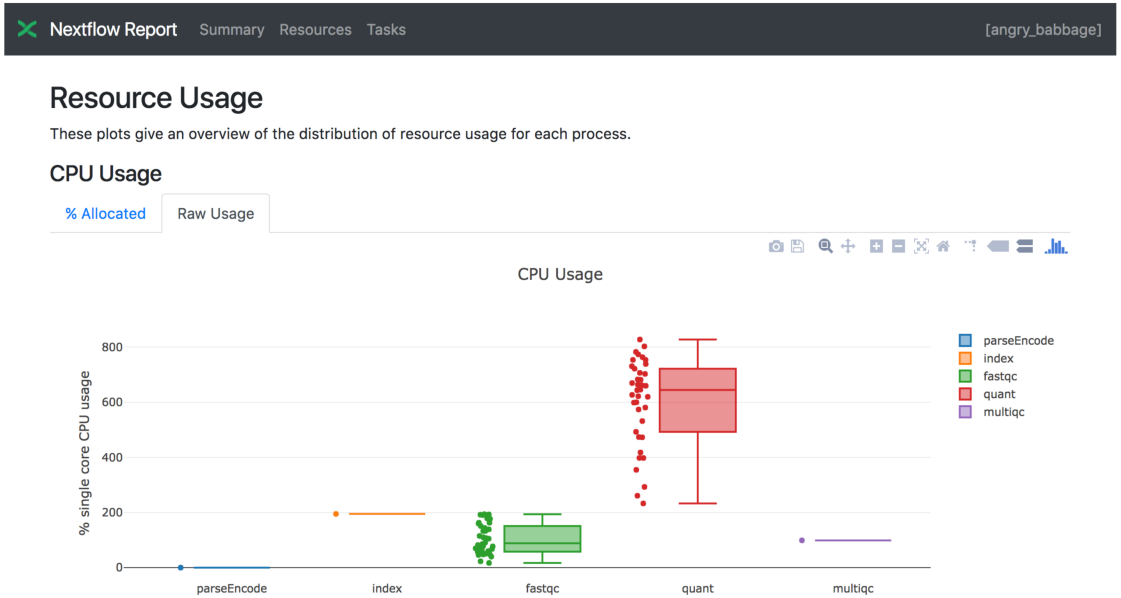
```
work
`-- a3
    `-- 5efc6e8b5c1779f668fee0a17a07e2
        |-- .command.begin
        |-- .command.err
        |-- .command.log
        |-- .command.out
        |-- .command.run
        |-- .command.sh
        |-- .command.trace
        |-- .exitcode
        |-- D1601.bam
        |-- D1601.bam.bai
        `-- 231116_A00514_1601_AHNYYJDSX7 -> /mnt/beegfs/DATA/231116_A00514_1601_AHNYY
```

nextflow run <your script> -resume

Metrics

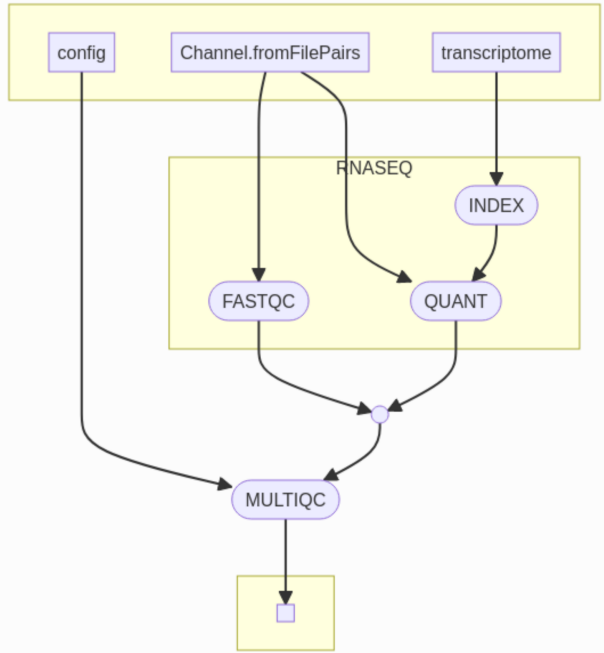
nextflow run <pipeline> -with-report [file name]
nextflow run <pipeline> -with-timeline
nextflow run <pipeline> -with-trace

	duration	walltime	%cpu	rss	vmem	rchar	wchar
88	1m	5s	0.0%	29.8 MB	354 MB	33.3 MB	0
11	30s	10s	35.7%	152.8 MB	428.1 MB	192.7 MB	1 MB
18	29s	20s	4.5%	289.5 MB	381.6 MB	33.3 MB	0
59	30s	9s	6.0%	122.8 MB	353.4 MB	33.3 MB	0
07	30s	19s	5.0%	195 MB	395.8 MB	65.3 MB	121 KB
53	30s	12s	43.6%	140.7 MB	432.2 MB	192.7 MB	182.7 MB
57	1m 30s	1m 11s	94.3%	611.6 MB	693.8 MB	961.2 GB	6.1 GB
02	1m 1s	38s	36.6%	115.8 MB	167.8 MB	364 GB	5.1 GB
25	30s	12s	59.6%	696 MB	734.6 MB	354.3 GB	420.4 MB
46	3m 1s	2m 6s	130.1%	703.3 MB	760.9 MB	1.1 TB	28.6 GB
18	3m 1s	2m 43s	116.6%	682.1 MB	743.6 MB	868.5 GB	42 GB
61	10m 2s	9m 16s	95.5%	706.2 MB	764 MB	1.6 TB	172.4 GB
43	30s	352ms	0.0%	35.6 MB	58.3 MB	199.3 MB	7.9 MB
55	30s	488ms	0.0%	108.2 MB	158 MB	317.1 MB	9.8 MB
70	30s	238ms	0.0%	6.7 MB	29.6 MB	190 MB	91.2 MB
08	30s	442ms	0.0%	108.1 MB	158 MB	832 MB	565.6 MB
73	30s	6s	0.0%	112.7 MB	162.8 MB	4.9 GB	3.9 GB
48	30s	616ms	0.0%	10.4 MB	34.6 MB	238 MB	8.4 MB
9	30s	1s	0.0%	4.8 MB	42 MB	240.6 MB	79 KB



Metrics

nextflow run <pipeline> -with-dag flowchart.png



```
process sayHello {

  input: val(x)
  output: stdout
  """
  echo -n $x
  """
}

workflow {
  def ch1 = channel.of('Hello','Ciao','Hola','Boujour')
  def ch2 = channel.of('world!','mondo!','mundo!','le monde!')

  ch1 | sayHello | merge(ch2) | view((x,y) → "$x $y")
}
```

example8.nf

How to force execution of two process without relation

Add a dependency with a dedicated channel

```
Process foo{
    output:
    val(true), emit: doneCh

    script:
    """
    your command here
    """
}
Process bar{
    input:
    val(flag)

    script ...
}
```

How to select the active channel

```
process bar {
    output:
    path("file.txt"), emit: barCh

    when:
    params.flag

    script:
    """
    echo "bar" > file.txt
    """
}
```

```
process foo {
    output:
    path("file.txt"), emit: fooCh

    when:
    !params.flag

    script:
    """
    echo "foo" > file.txt
    """
}
```

```
params.flag=false

workflow{
    foo()
    bar()
    joe(foo.out.fooCh.mix(bar.out.barCh))
}
```

```
process joe {

    publishDir '.' , mode: 'copy', overwrite: false

    input:
    path(x)

    output:
    path(x)

    script:
    """

    """
}
```

example9.nf