Building Pipelines in DolphinNext

Onur Yukselen, Alper Kucukural

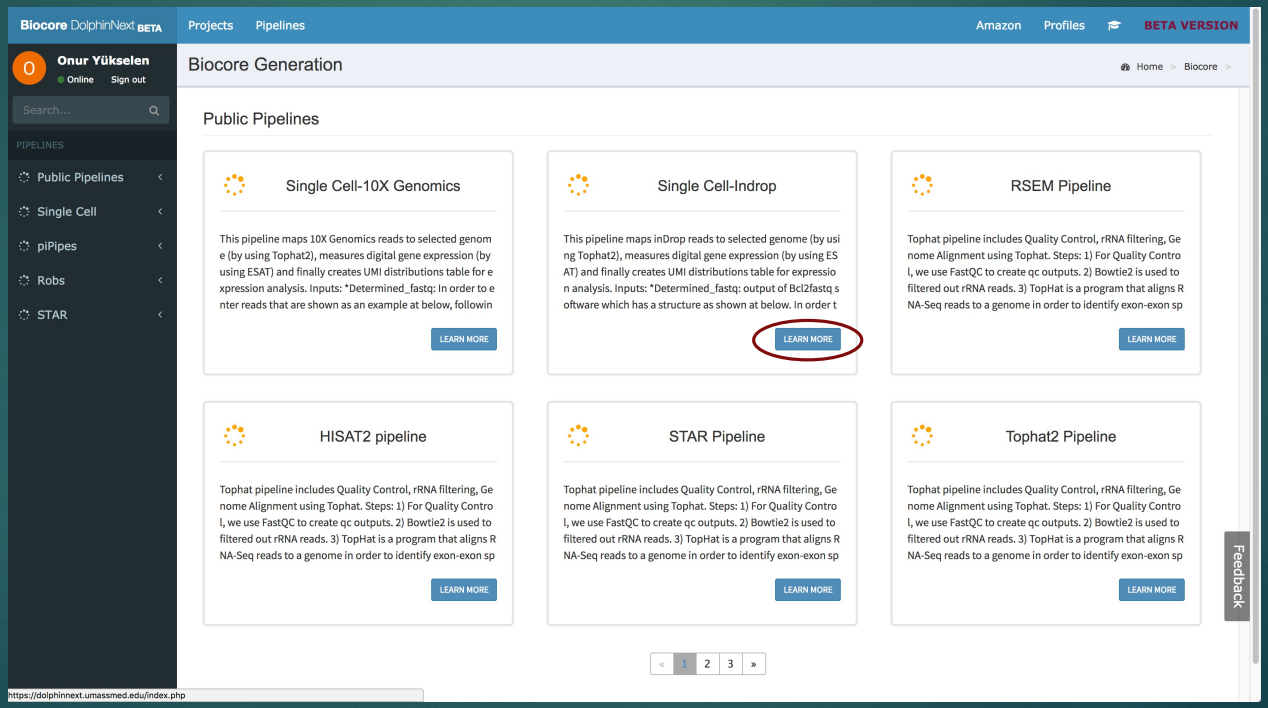
Bootcamp, May 05, 2021

University of Massachusetts Medical School
UMASS

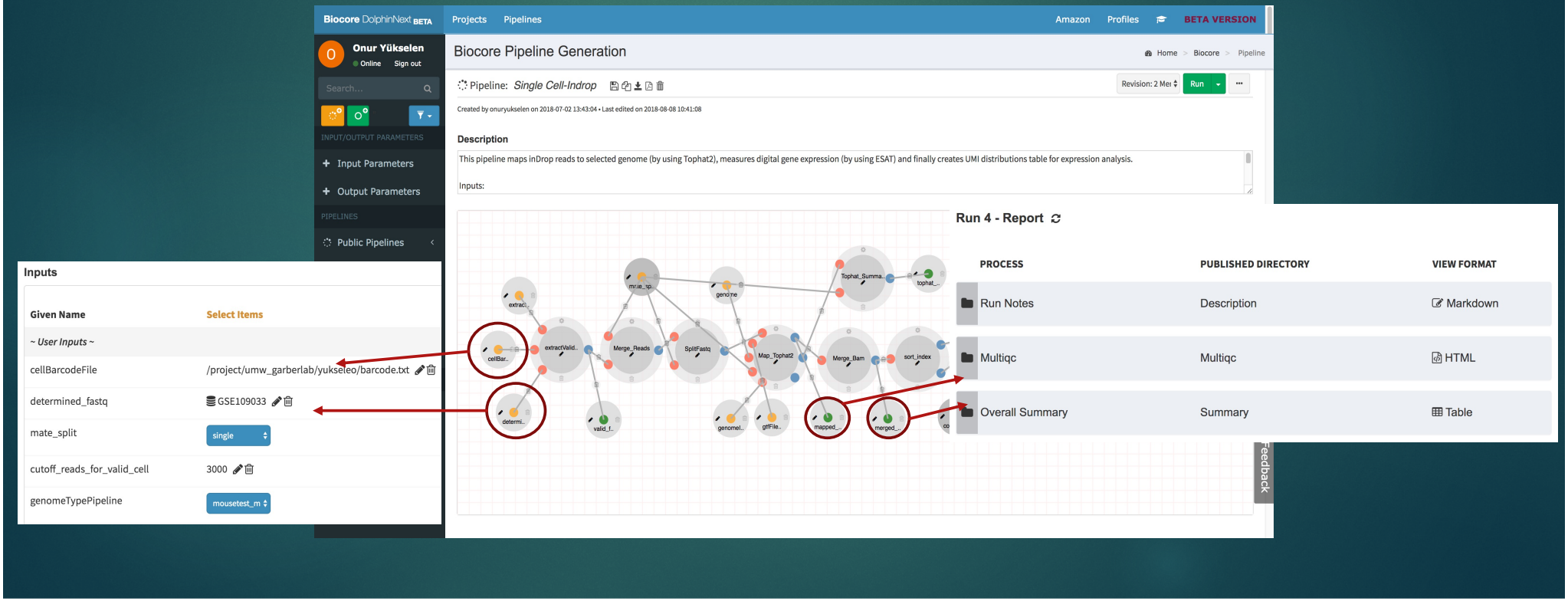
Content

- Pipeline Basics
  - Process, Inputs, Outputs, Script
- Building Blocks
  - Val, File, Set
- How it Works?

Pipeline Basics



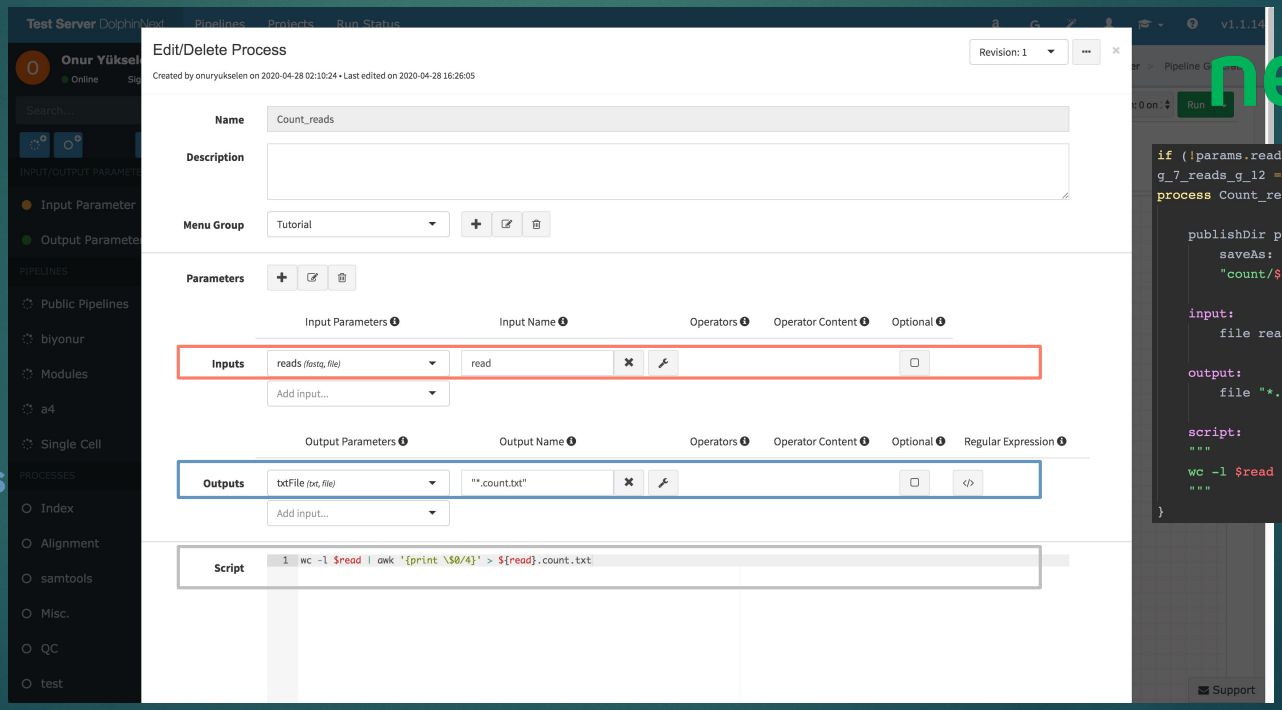
Pipeline Basics – Input/Output Parameter



Pipeline Basics – Inputs/Outputs/Script

inputs outputs

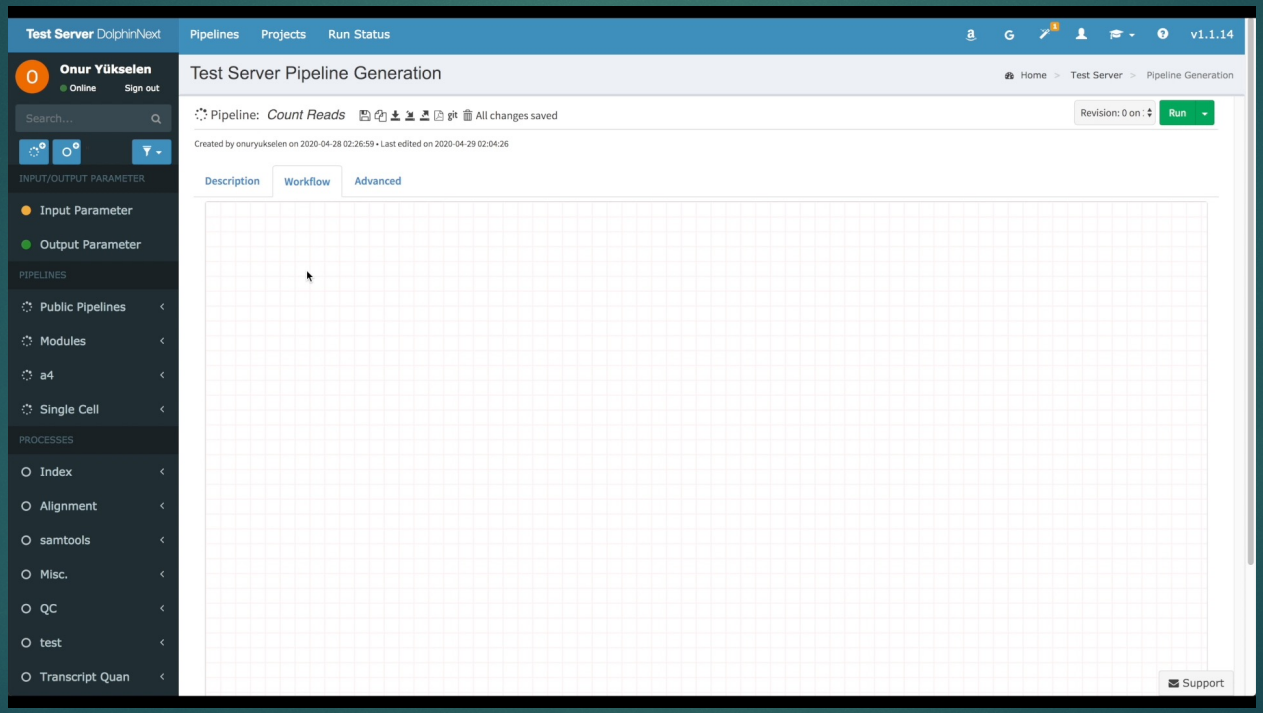
script



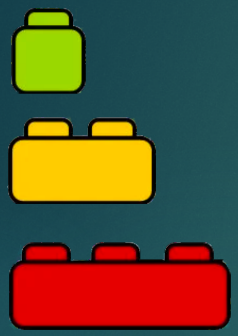
nextflow

```
if (!params.reads){params.reads = ""}
g_7_reads_g_12 = file(params.reads, type: 'any')
process Count_reads {

    publishDir params.outdir, overwrite: true, mode: 'copy',
        saveAs: {filename -> if (filename =~ /.*.count.txt$/)
        "count/$filename"}

    input:
        file read from g_7_reads_g_12

    output:
        file "*.count.txt"  into g_12_txtFile

    script:
    """
    wc -l $read | awk '{print \$0/4}' > ${read}.count.txt
    """
}
```

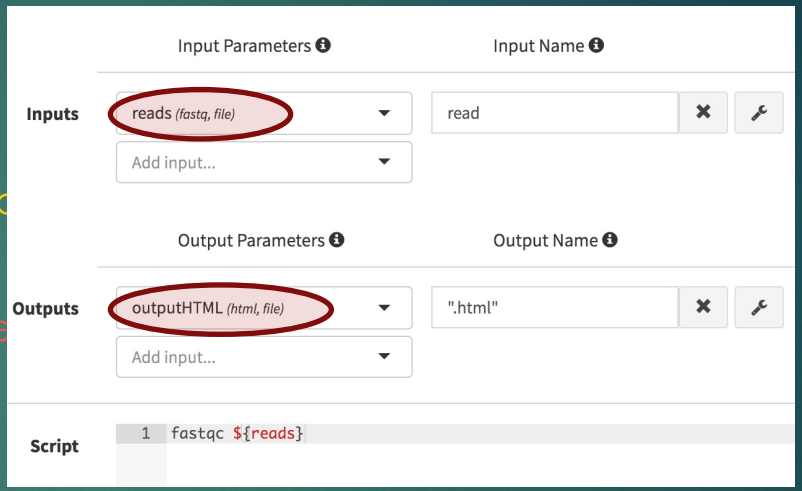
Pipeline Basics – Drag&Drop



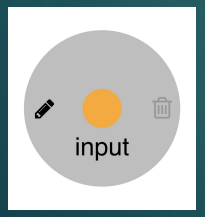
Building Blocks

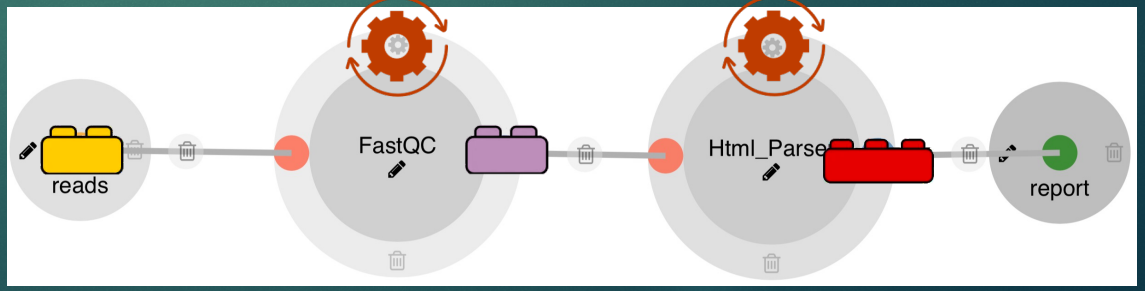


Qualifier	Examples
Val	"mouse"
File	exp_rep1.fasta
Set	["exp_rep1", e

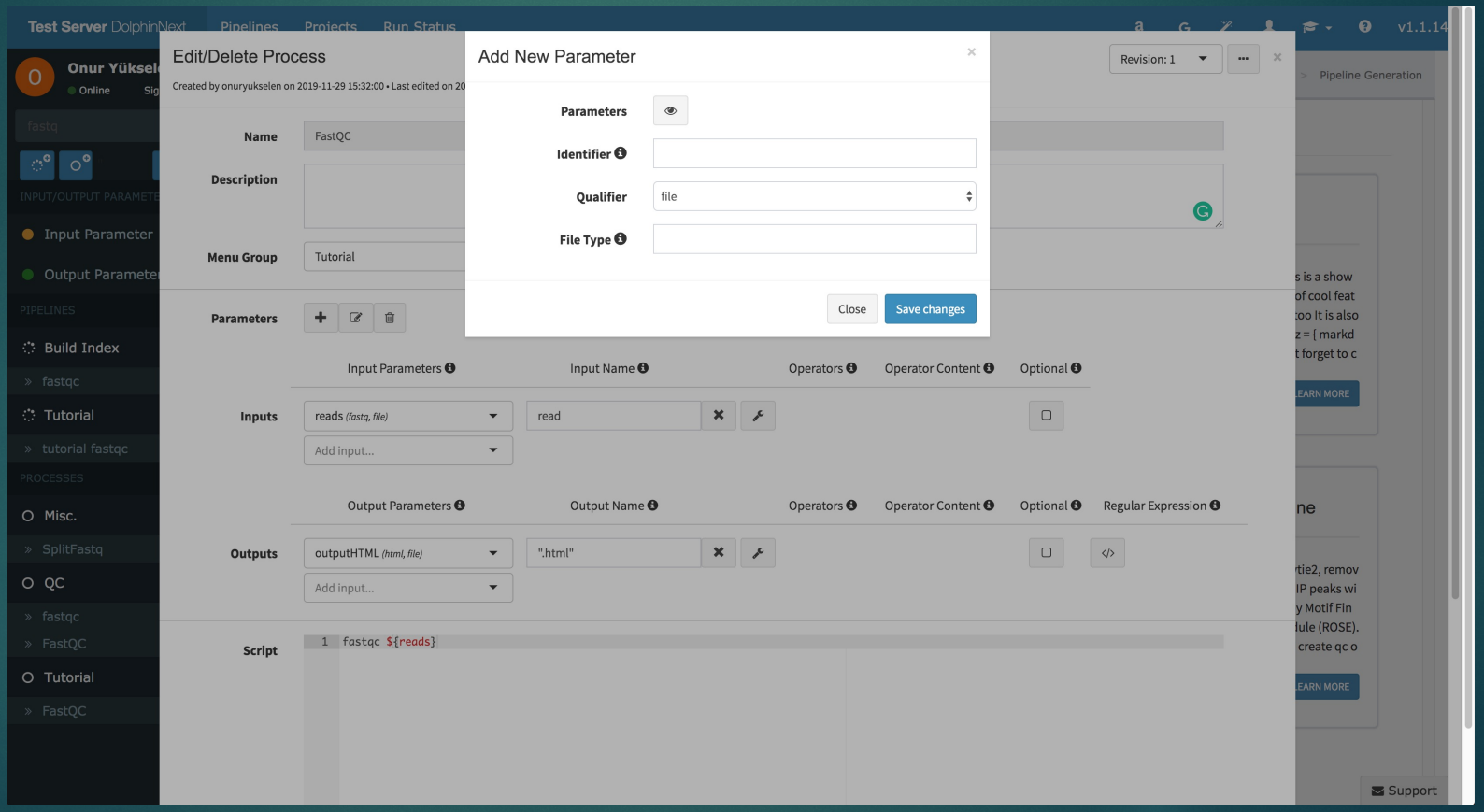


Input Parameters





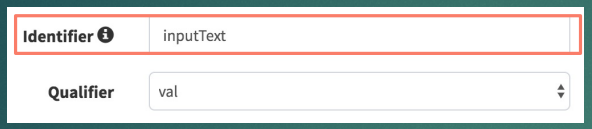
Adding New Parameter

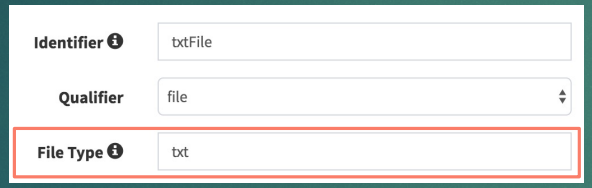


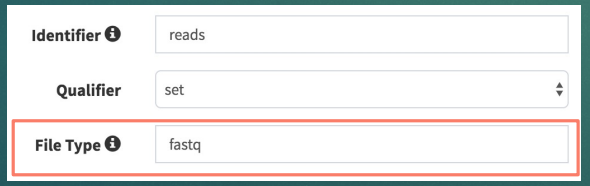
Adding New Parameter

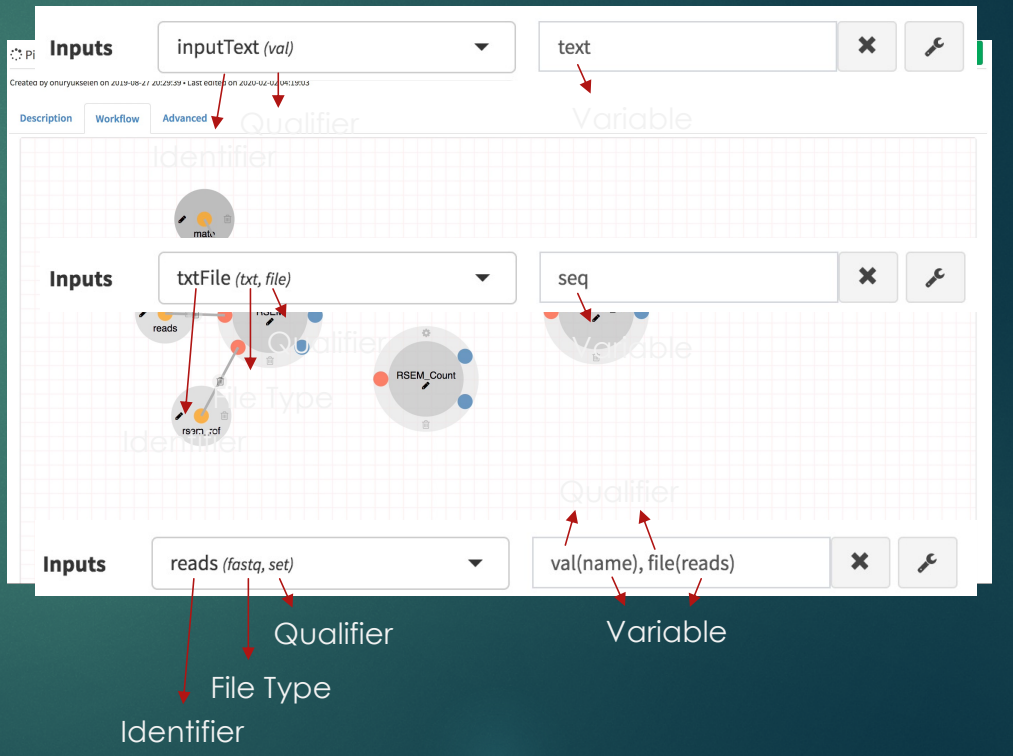
Adding New Parameter

Using Parameter in the Process









How it Works? (1) – Using Val

"Hello World"

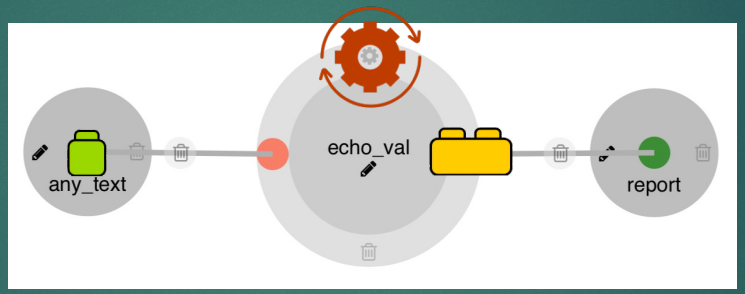
example.txt

report

User Inputs

Inputs

Given Name	Select Items
~ User Inputs ~	
any_text	Hello World!



Run Logs

```
N E X T F L O W ~ version 20.01.0
Launching `/home/oy28w/tutorials/run1/run422/nextflow.nf`
[84/2dfda8] Submitted process > echo_val
```

Inputs inputText *(val)* | text

Script

```
1  echo $text > example.txt
```

```
$ echo Hello World! > example.txt
```

Outputs txtFile *(txt, file)* | "example.txt"

Run Reports

Echo Val

example.txt

```
Hello World!
```

How it Works? (2) – Using File

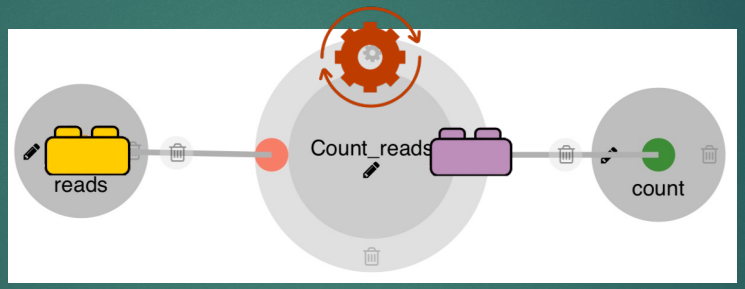
control_rep1.fastq

count.txt

Run Inputs

Inputs

Given Name	Select Items
~ User Inputs ~	
reads	/share/data/umw_biocore/dnext_data/tutorial/fastq_data/single/control_rep1.fastq



Inputs reads *(fastq, file)* read

Script

```
1  wc -l $read | awk '{print \$0/4}' > count.txt
```

```
$ wc -l control_rep1.fastq | awk '{print $0/4}' > count.txt
```

Outputs txtFile *(txt, file)* "count.txt"

Run Logs

```
N E X T F L O W  ~  version 20.01.0
Launching `/home/oy28w/tutorials/run2/run423/nextflow.nf
[8b/62795a] Submitted process > Count_reads
```

Run Reports

Count Reads

count.txt 24288

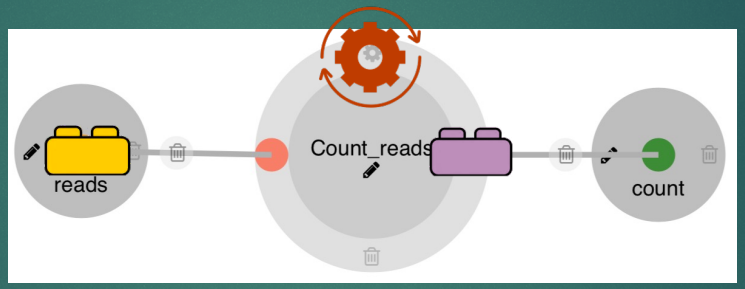
How it Works? (3) – Using File – Multiple Input Problem?

control_rep1.fastq count.txt
control_rep2.fastq count.txt
control_rep3.fastq count.txt

Run Inputs

Inputs

Given Name	Select Items
~ User Inputs ~	
reads	3 single-end reads



Run Logs

```
N E X T F L O W ~ version 20.01.0
Launching `/home/oy28w/tutorials/run2/run423/nextflow.nf`
[5f/19a1f0] Submitted process > Count_reads (1)
[56/5a17f6] Submitted process > Count_reads (2)
[d9/7557e2] Submitted process > Count_reads (3)
```

Inputs reads *(fastq, file)* read

Script

```
1  wc -l $read | awk '{print \$0/4}' > count.txt
```

Outputs txtFile *(txt, file)* "count.txt"

Run Reports

Count Reads

count.txt	17568

How it Works? (3) – Using File - Problem Fixed

control_rep1.fastq
control_rep2.fastq
control_rep3.fastq

control_rep1.fastq.count.txt
control_rep2.fastq.count.txt
control_rep3.fastq.count.txt

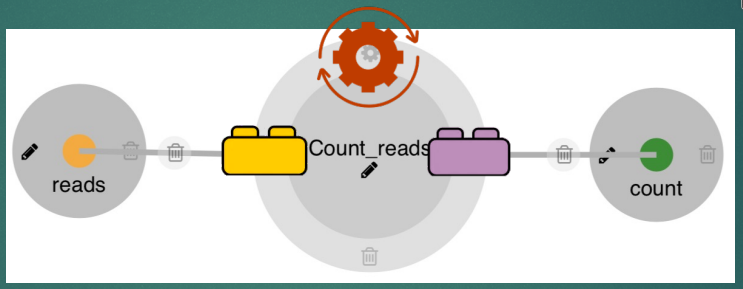
Run Logs

```
N E X T F L O W ~ version 20.01.0
Launching `/home/oy28w/tutorials/run2/run423/nextflow.nf`
[5f/19a1f0] Submitted process > Count_reads (1)
[56/5a17f6] Submitted process > Count_reads (2)
[d9/7557e2] Submitted process > Count_reads (3)
```

Run Inputs

Inputs

Given Name	Select Items
~ User Inputs ~	
reads	3 single-end reads



Inputs reads *(fastq, file)* read

Script

```
1  wc -l $read | awk '{print \$0/4}' > ${read}.count.txt
```

$read == ${read}

```
$ wc -l control_rep1.fastq | awk '{print $0/4}' > control_rep1.fastq.count.txt
```

Outputs txtFile *(txt, file)* "*.count.txt"

Run Reports

Count Reads

control_rep1.fastq.count.txt	24288
control_rep2.fastq.count.txt	
control_rep3.fastq.count.txt	

How it Works? (4) – Using Set

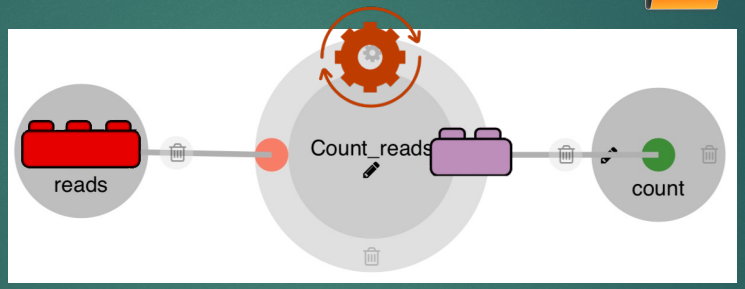
["control_rep1", control_rep1.fastq]
["control_rep2", control_rep2.fastq]
["control_rep3", control_rep3.fastq]

control_rep1.txt
control_rep2.txt
control_rep3.txt

Run Inputs

Inputs

Given Name	Select Items
~ User Inputs ~	
reads	3 single-end reads



Run Logs

```
N E X T F L O W  ~  version 20.01.0
Launching `/home/oy28w/tutorials/run2/run423/nextflow.nf`
[5f/19a1f0] Submitted process > Count_reads (1)
[56/5a17f6] Submitted process > Count_reads (2)
[d9/7557e2] Submitted process > Count_reads (3)
```

Inputs reads *(fastq, set)* | val(name), file(read)

Script

```
1  wc -l $read | awk '{print \$0/4}' > ${name}.txt
```

Outputs txtFile *(txt, file)* | "${name}.txt"

Run Reports

Count Reads

control_rep1.txt	24288
control_rep2.txt	
control_rep3.txt	

How it Works? (5) – Using Set for Paired Reads

["control_rep1", control_rep1.R1.fastq, control_rep1.R2.fastq]
["control_rep2", control_rep2.R1.fastq, control_rep2.R2.fastq]
["control_rep3", control_rep3.R1.fastq, control_rep3.R2.fastq]

"pair"

control_rep1.txt
control_rep2.txt
control_rep3.txt

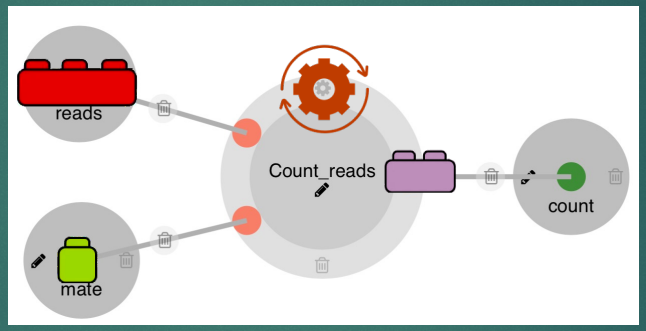
Run Logs

```
N E X T F L O W  ~  version 20.01.0
Launching `/home/oy28w/tutorials/run4/run425/nextflow.nf`
[a0/bfffa5] Submitted process > Count_reads (2)
[8e/6c6a0e] Submitted process > Count_reads (1)
[a8/b5f2d6] Submitted process > Count_reads (3)
```

Run Inputs

~ User Inputs ~

mate	pair
reads	3 paired-end reads



Run Reports

Count Reads

control_rep1.txt	24288
control_rep2.txt	24288
control_rep3.txt	48576

Inputs: reads *(fastq, set)* | val(name), file(read)

Script:

```
1  wc -l $read | awk '{print \$0/4}' > ${nam
```

```
$ wc -l control_rep1.R1.fastq control_rep1.R2.fastq | awk '{print $0/4}' >
```

```
$ wc -l control_rep1.R1.fastq control_rep1.R2.fastq
  97152 control_rep1.R1.fastq
  97152 control_rep1.R2.fastq
 194304 total
$ wc -l control_rep1.R1.fastq control_rep1.R2.fastq | awk '{print $0/4}'
  24288
  24288
  48576
```

Outputs: txtFile *(txt, file)* | "${name}.txt"

How it Works? (6) – Using Set for Single & Paired Reads

Run Inputs

~ *User Inputs* ~

mate	pair
reads	3 paired-end reads

Run Logs

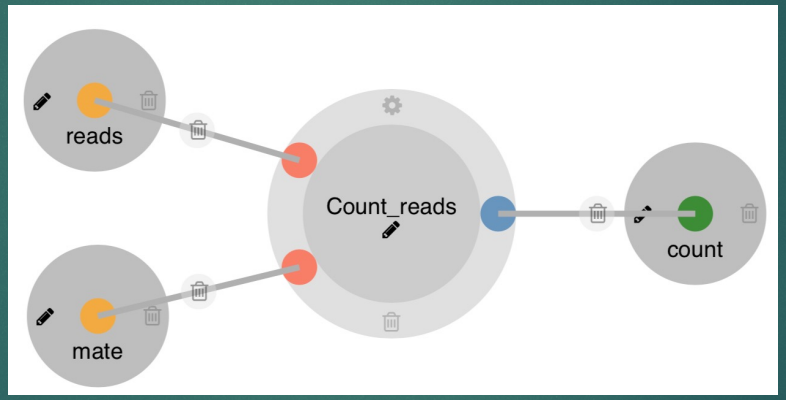
```
N E X T F L O W  ~  version 20.01.0
Launching `/home/oy28w/tutorials/run4/run425/nextflow.nf`
[a0/bfffa5] Submitted process > Count_reads (2)
[8e/6c6a0e] Submitted process > Count_reads (1)
[a8/b5f2d6] Submitted process > Count_reads (3)
```

Run Reports

Count Reads

- control_rep1.txt
- control_rep2.txt
- control_rep3.txt

```
control_rep1.R1.fastq   24288
control_rep1.R2.fastq   24288
```



Inputs reads *(fastq, set)* | val(name), file(read)

Script

```
1  if [ "${mate}" == "pair" ]; then
2      wc -l $read |  awk '{print \$2 "\\t" \$1/4}' | head -2 > ${name}.txt
3  else
4      wc -l $read |  awk '{print \$2 "\\t" \$1/4}' > ${name}.txt
5  fi
```

Outputs txtFile *(txt, file)* | "${name}.txt"

Run Inputs

~ *User Inputs* ~

mate	single
reads	3 single-end reads

Run Logs

```
N E X T F L O W  ~  version 20.01.0
Launching `/home/oy28w/tutorials/run4/run425/nextflow.nf`
[a0/bfffa5] Submitted process > Count_reads (2)
[8e/6c6a0e] Submitted process > Count_reads (1)
[a8/b5f2d6] Submitted process > Count_reads (3)
```

Run Reports

Count Reads

- control_rep1.txt
- control_rep2.txt
- control_rep3.txt

```
control_rep1.fastq      24288
```

How it Works? (7) – Merging tables

["control_rep1", control_rep1.R1.fastq, control_rep1.R2.fastq]
["control_rep2", control_rep2.R1.fastq, control_rep2.R2.fastq]
["control_rep3", control_rep3.R1.fastq, control_rep3.R2.fastq]

"pair"

control_rep1.txt
control_rep2.txt
control_rep3.txt

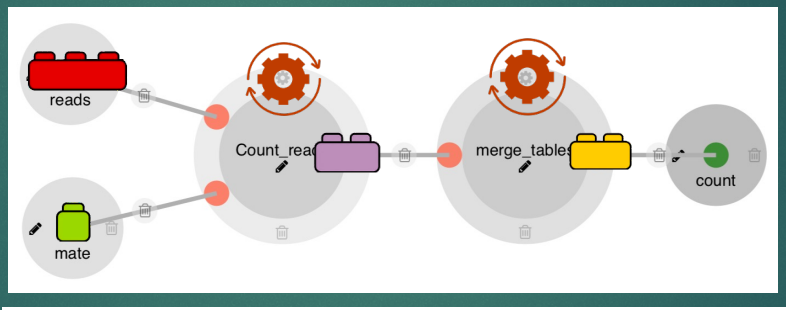
counts.tsv

Run Logs

```
N E X T F L O W ~ version 20.01.0
Launching `/home/oy28w/tutorials/run7/run428/nextflow.nf`
[db/ef1138] Submitted process > Count_reads (1)
[3c/6ae923] Submitted process > Count_reads (2)
[d8/cbb4b6] Submitted process > Count_reads (3)
[83/8a6743] Submitted process > merge_tables
```

Run Inputs

~ User Inputs ~	
mate	pair
reads	3 paired-end reads



Inputs txtFile *(txt, file)* | txtFiles | collect

Script

```
1  echo -e 'filename\tcount' > counts.tsv
2  cat $txtFiles >> counts.tsv
```

```
$ cat control_rep1.txt control_rep2.txt control_rep3.txt >> counts.tsv
```

Outputs outFileTSV *(tsv, file)* | "counts.tsv"

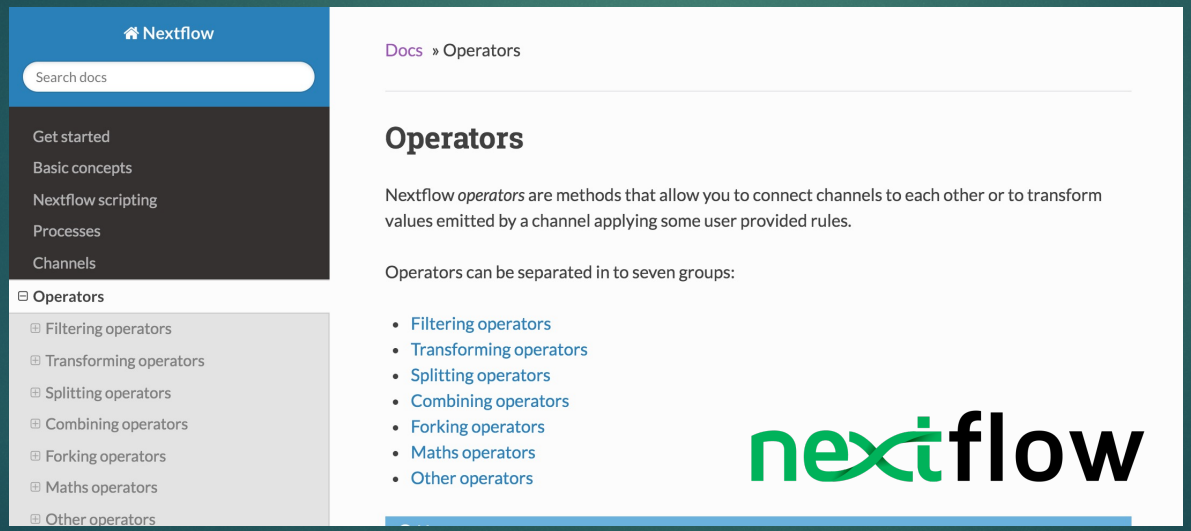
Run Reports

counts.tsv

filename	count
control_rep1.R1.fastq	24288
control_rep1.R2.fastq	24288
control_rep2.R1.fastq	15693
control_rep2.R2.fastq	15693
control_rep3.R1.fastq	17008
control_rep3.R2.fastq	17008

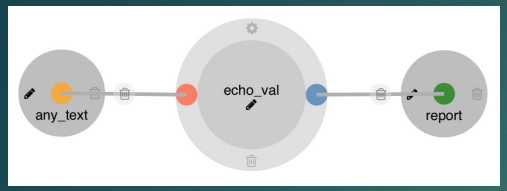
Operators

- Reference Documentation for Operators:
  - https://www.nextflow.io/docs/latest/index.html



Summary

(1) Val



Inputs: inputText *(val)* | text

Outputs: txtFile *(txt, file)* | "example.txt"

Script:

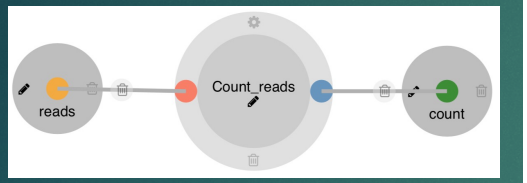
```
1  echo $text > example.txt
```

Echo Val

example.txt

```
Hello World!
```

(3) File



Inputs: reads *(fastq, file)* | read

Outputs: txtFile *(txt, file)* | "*.count.txt"

Script:

```
1  wc -l $read | awk '{print \$0/4}' > ${read}.count.txt
```

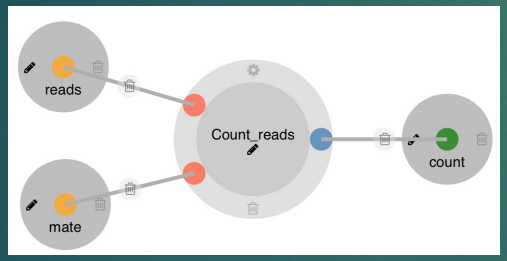
Count Reads

control_rep1.fastq.count.txt 24288

control_rep2.fastq.count.txt

control_rep3.fastq.count.txt

(6) Set



Inputs: reads *(fastq, set)* | val(name), file(read)

Outputs: txtFile *(txt, file)* | "${name}.txt"

Script:

```
1  if [ "${mate}" == "pair" ]; then
2      wc -l $read |  awk '{print \$2 "\\t" \$1/4}' | head -2 > ${name}.txt
3  else
4      wc -l $read |  awk '{print \$2 "\\t" \$1/4}' > ${name}.txt
5  fi
```

Count Reads

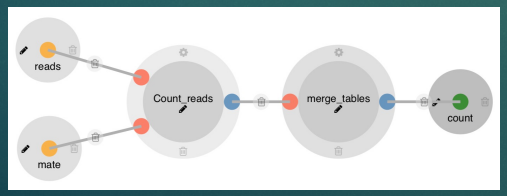
control_rep1.txt

control_rep2.txt

control_rep3.txt

```
control_rep1.R1.fastq   24288
control_rep1.R2.fastq   24288
```

(7) Operator



Inputs: txtFile *(txt, file)* | txtFiles | collect

Outputs: outFileTSV *(tsv, file)* | "counts.tsv"

Script:

```
1  echo -e 'filename\tcount' > counts.tsv
2  cat $txtFiles >> counts.tsv
```

counts.tsv

filename	count
control_rep1.R1.fastq	24288
control_rep1.R2.fastq	24288
control_rep2.R1.fastq	15693
control_rep2.R2.fastq	15693
control_rep3.R1.fastq	17008
control_rep3.R2.fastq	17008

Next Step: Building Pipeline

Overview

- Before you start
- Getting Started
- Exercise 1: Creating processes
  - FastQC process
  - Hisat2 process
  - RSeQC process
- Exercise 2: Building a pipeline
- Exercise 3: Running a pipeline
- Exercise 4 (optional): Supporting single and paired end reads

If you prefer, you can click on the video links to follow the tutorial in a video.