Workflows and Operators in nextflow

Pre-training
nf-core hackathon 2025

March 10th 2025

Antoine Buetti-Dinh
antoine.buetti@gmail.com

ZDS ZURICH DATA SCIENTISTS

nextflow AMBASSADOR

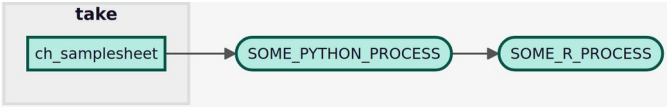
Highlights of Today

1. Workflows in nextflow
   - Workflows (pipelines) of multiple steps that perform analyses in data science
   - Example of a mix of programming languages
     - Python, R, Bash in same script
     - Transportable & reproducible via containers docker CONDA wave ...
     - Easy deployment on different infrastructure (servers, cloud, clusters) aws Google Cloud Azure ...
     - Refs:
       https://carpentries-incubator.github.io/workflows-nextflow/06-workflow.html
       https://training.nextflow.io/latest/basic_training/intro/#nextflow-code
       https://training.nextflow.io/latest/basic_training/rnaseq_pipeline/

2. Operators in nextflow
   - Simple manipulation tools
     - Filter, gather, split, sort,...
   - Refs:
     https://carpentries-incubator.github.io/workflows-nextflow/07-operators.html
     https://training.nextflow.io/latest/basic_training/operators/

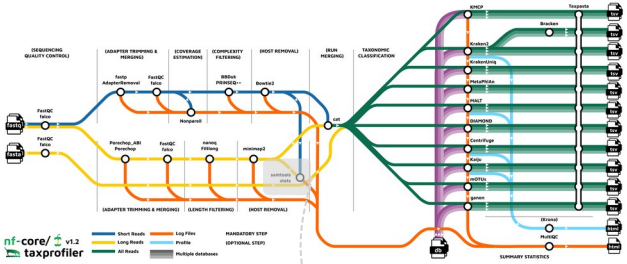
Simple workflow

We will try a cocktail of R & Python



More complicated workflow

input



output

Include old software!

(by creating right environment)

Workflow Example

Want a more bioinformatics example?

→ https://carpentries-incubator.github.io/workflows-nextflow/06-workflow.html

Workflow Example

Want a more bioinformatics example?

→ https://carpentries-incubator.github.io/workflows-nextflow/06-workflow.html

Just a file with a number in it

```
workflow {

    input_ch = Channel.fromPath('input.csv')

    SOME_PYTHON_PROCESS(input_ch)

    SOME_R_PROCESS(SOME_PYTHON_PROCESS.out)
}
```


```
process SOME_PYTHON_PROCESS {

    input:
    path input_file

    output:
    path '*_processed.txt'

    script:
    """
    #!/usr/bin/env python3

    import sys
    import random

    with open("${input_file}", "r") as f:
        number = int(f.read().strip())

    result = [random.randint(5, 15) for i in range(0, number)]

    with open("${input_file.baseName}_processed.txt", "w") as f:
        for item in result:
            f.write(f"{item}\\n")
    """
}
```

Generates random numbers

11
6
15
5
15
14
.
.
.

```
process SOME_R_PROCESS {

    input:
    path input_file

    output:
    path '*_analyzed.txt'

    script:
    """
    #!/usr/bin/env Rscript

    input_data <- read.table("${input_file}", header=FALSE,
    col.names=c("value"))
    summary_stats <- summary(input_data\$value)

    sink("${input_file.baseName}_analyzed.txt")
    cat("Summary Statistics:\\n")
    print(summary_stats)

    sink()
    """
}
```

Calculates Summary statistics

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
5.00    6.25   11.50   10.14   13.00   15.00
```

Workflow Example

Want a more bioinformatics example?

→ https://carpentries-incubator.github.io/workflows-nextflow/06-workflow.html

Just a file with a number in it

```
workflow {

    input_ch = Channel.fromPath('input.csv')

    SOME_PYTHON_PROCESS(input_ch)

    SOME_R_PROCESS(SOME_PYTHON_PROCESS.out)
}
```


```
process SOME_PYTHON_PROCESS {
    container 'python:3.9'
    input:
    path input_file

    output:
    path '*_processed.txt'

    script:
    """
    #!/usr/bin/env python3

    import sys
    import random

    with open("${input_file}", "r") as f:
        number = int(f.read().strip())

    result = [random.randint(5, 15) for i in range(0, number)]

    with open("${input_file.baseName}_processed.txt", "w") as f:
        for item in result:
            f.write(f"{item}\\n")
    """
}
```

You can ask to use particular versions/packages via containers (e.g. Docker or Conda)

```
process SOME_R_PROCESS {
    conda 'r-base=4.1.0'
    input:
    path input_file

    output:
    path '*_analyzed.txt'

    script:
    """
    #!/usr/bin/env Rscript

    input_data <- read.table("${input_file}", header=FALSE,
    col.names=c("value"))
    summary_stats <- summary(input_data\$value)

    sink("${input_file.baseName}_analyzed.txt")
    cat("Summary Statistics:\\n")
    print(summary_stats)

    sink()
    """
}
```

Generates random numbers

```
11
6
15
5
15
14
.
.
.
```

Calculates Summary statistics

```
Min. 1st Qu. Median  Mean 3rd Qu.  Max.
5.00    6.25  11.50  10.14   13.00 15.00
```

Let’s see some more cool things one can do!

Another Example...

Another Example...

Also something quite simpler...

```
.
├── parallel.nf
├── sample1.fastq.gz
├── sample2.fastq.gz
├── sample3.fastq.gz
├── sample4.fastq.gz
├── sample5.fastq.gz
└── sample6.fastq.gz
```

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

process myProcess {
    input:
    path myInp

    output:
    path '*.fastq'

    script:
    """
    gunzip -c $myInp > "${myInp.baseName}.fastq"
    """
}

workflow {
    myChannel = Channel.fromPath("./*.fastq.gz")
    myProcess(myChannel)
}
```

My simple process to decompress 6 zip files

Simple call from main

Another Example...

~ Concept of "channel"

Also something quite simpler...

```
.
├── parallel.nf
├── sample1.fastq.gz
├── sample2.fastq.gz
├── sample3.fastq.gz
├── sample4.fastq.gz
├── sample5.fastq.gz
└── sample6.fastq.gz
```

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

process myProcess {
    input:
    path myInp

    output:
    path '*.fastq'

    script:
    """
    gunzip -c $myInp > "${myInp.baseName}.fastq"
    """
}

workflow {
    myChannel = Channel.fromPath("./*.fastq.gz")
    myProcess(myChannel)
}
```

Run:

6 processes to run are spotted!

```
executor >  local (6)
[4b/3a266b] myProcess (6) | 6 of 6 ✔
```

```
.
├── parallel.nf
├── sample1.fastq.gz
├── sample2.fastq.gz
├── sample3.fastq.gz
├── sample4.fastq.gz
├── sample5.fastq.gz
├── sample6.fastq.gz
└── work
    ├── 08
    │   └── b6ae829662ebf47836fbf7223f08c2
    │       ├── sample5.fastq.fastq
    │       └── sample5.fastq.gz -> /home/antoinebuetti/Desktop/tmp/parallel/sample5.fastq.gz
    ├── 22
    │   └── 893f66cfb7c7af3deedccc8271382f
    │       ├── sample4.fastq.fastq
    │       └── sample4.fastq.gz -> /home/antoinebuetti/Desktop/tmp/parallel/sample4.fastq.gz
    ├── 2d
    │   └── 740a9ce945adacb0dd012c544c310d
    │       ├── sample3.fastq.fastq
    │       └── sample3.fastq.gz -> /home/antoinebuetti/Desktop/tmp/parallel/sample3.fastq.gz
    ├── 4b
    │   └── 3a266ba49b13d36645494c0717376b
    │       ├── sample2.fastq.fastq
    │       └── sample2.fastq.gz -> /home/antoinebuetti/Desktop/tmp/parallel/sample2.fastq.gz
    ├── b0
    │   └── 58e4c8308156332c41e83c01e8432e
    │       ├── sample1.fastq.fastq
    │       └── sample1.fastq.gz -> /home/antoinebuetti/Desktop/tmp/parallel/sample1.fastq.gz
    └── d7
        └── 0c9ac81adaf588748b42234660b6ff
            ├── sample6.fastq.fastq
            └── sample6.fastq.gz -> /home/antoinebuetti/Desktop/tmp/parallel/sample6.fastq.gz
```

Another Example...

Also something quite simpler...

```
.
├── parallel.nf
├── sample1.fastq.gz
├── sample2.fastq.gz
├── sample3.fastq.gz
├── sample4.fastq.gz
├── sample5.fastq.gz
└── sample6.fastq.gz
```

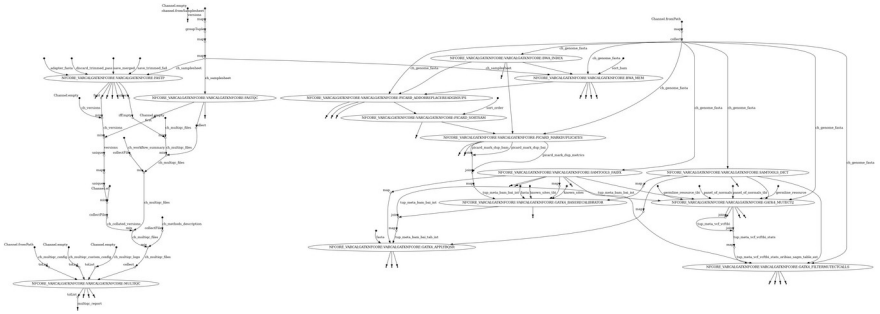
```nextflow
#!/usr/bin/env nextflow

process myProcess {
    input:
    path myInp

    output:
    path '*.fastq'

    script:
    """
    gunzip -c $myInp > "${myInp.baseName}.fastq"
    """
}

workflow {
    myChannel = Channel.fromPath("./*.fastq.gz")
    myProcess(myChannel)
}
```

Run:

6 processes to run are spotted!

```
executor >  local (6)
[4b/3a266b] myProcess (6) | 6 of 6 ✔
```

Asynchronous programming



Branched DAGs allow to establish priorities

- **Things to compute get propagated through the process structure**
- **According to available resources**

Another Example...

Also something quite simpler...

```
.
├── parallel.nf
├── sample1.fastq.gz
├── sample2.fastq.gz
├── sample3.fastq.gz
├── sample4.fastq.gz
├── sample5.fastq.gz
└── sample6.fastq.gz
```

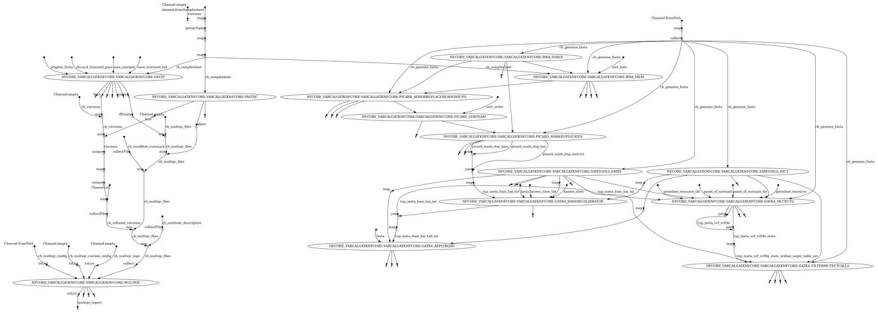
```nextflow
#!/usr/bin/env nextflow

process myProcess {
    input:
    path myInp

    output:
    path '*.fastq'

    script:
    """
    gunzip -c $myInp > "${myInp.baseName}.fastq"
    """
}

workflow {
    myChannel = Channel.fromPath("./*.fastq.gz")
    myProcess(myChannel)
}
```

Run:

6 processes to run are spotted!

```
executor >  local (6)
[4b/3a266b] myProcess (6) | 6 of 6 ✔
```

Asynchronous programming



Some other sequential code in R/Python/...
→ gets also handled the parallel way!

Branched DAGs allow to establish priorities

- **Things to compute get propagated through the process structure**
- **According to available resources**

That’s why nextflow is a good choice also for simple things 🙂

Operators

Overview on some operators

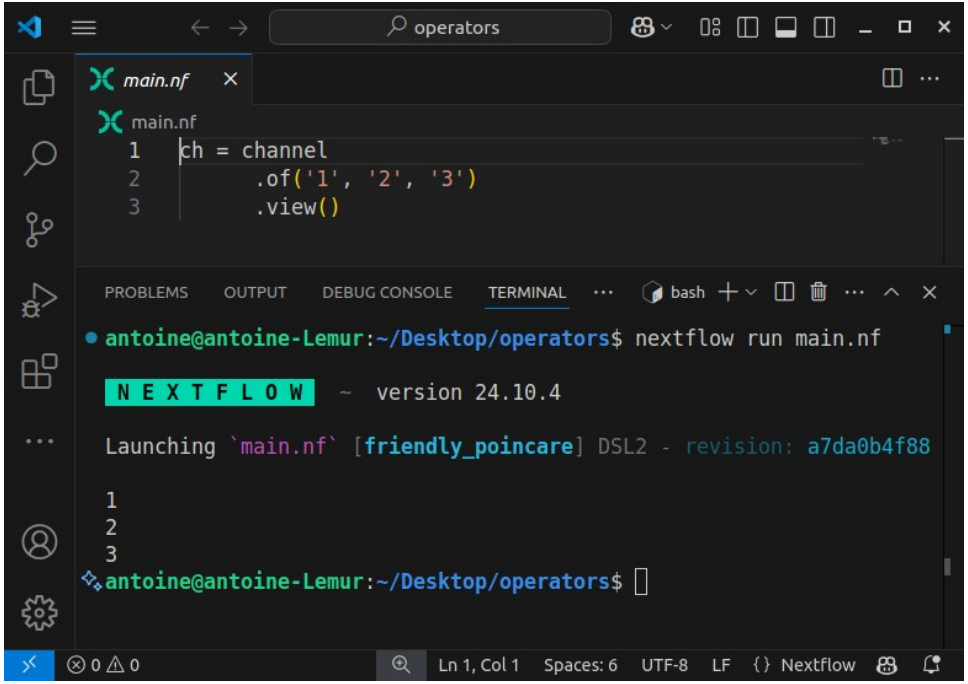
→ https://carpentries-incubator.github.io/workflows-nextflow/07-operators.html

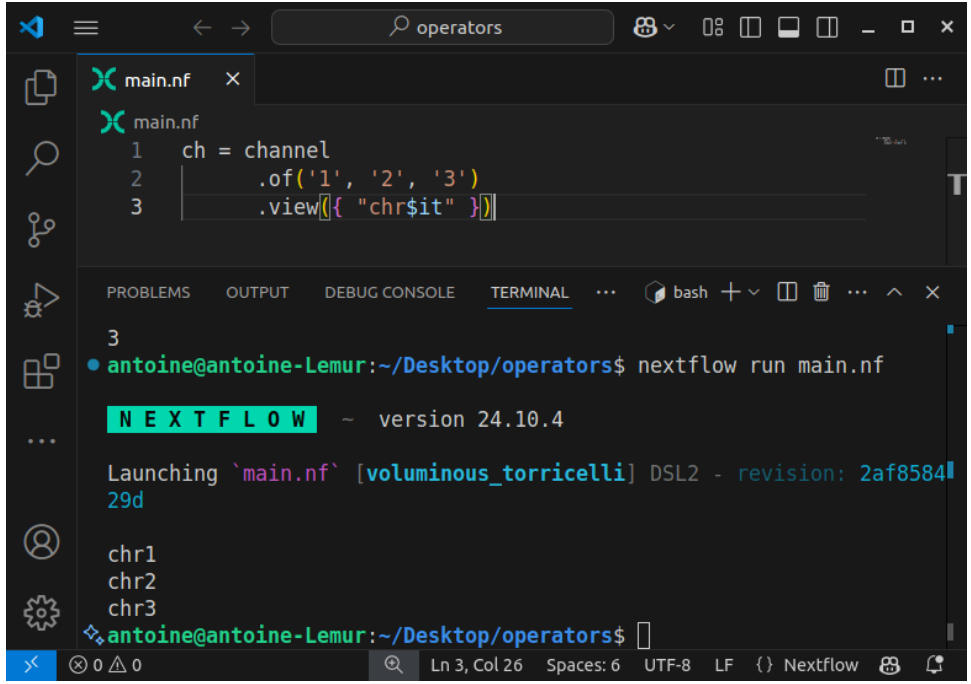
That's why nextflow is a good choice also for simple things 🙂

Operators

Overview on some operators

→ https://carpentries-incubator.github.io/workflows-nextflow/07-operators.html





Operators

Overview on some operators

→ https://carpentries-incubator.github.io/workflows-nextflow/07-operators.html

```
chr_ch = channel.of( 1..22, 'X', 'Y' )
autosomes_ch =chr_ch.filter( Number )
autosomes_ch.view()
```

```
Launching `main.nf` [magical_ptolemy] DSL2 - revision: 1e43205639

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
antoine@antoine-Lemur:~/Desktop/operators$
```

```
chr_ch = channel.of( 1..22, 'X', 'Y' )
autosomes_ch =chr_ch.filter( String )
autosomes_ch.view()
```

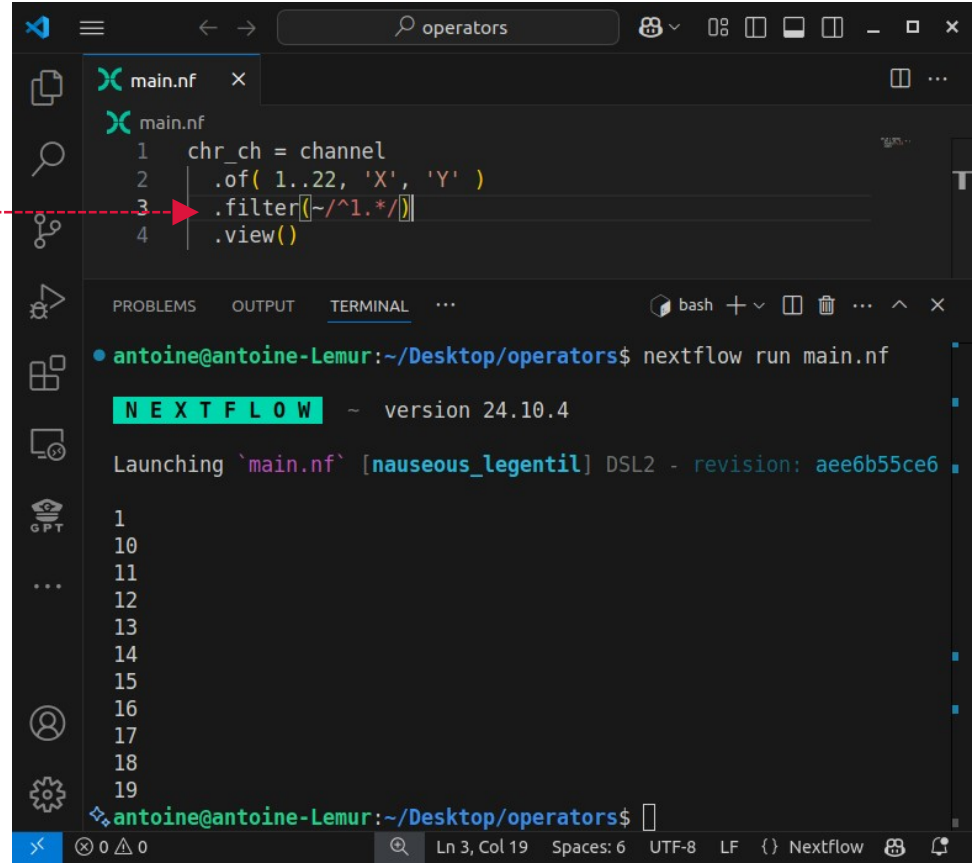
```
antoine@antoine-Lemur:~/Desktop/operators$ nextflow run main.nf

N E X T F L O W  ~  version 24.10.4

Launching `main.nf` [infallible_kirch] DSL2 - revision: 9beda2a34b

X
Y
antoine@antoine-Lemur:~/Desktop/operators$
```

Operators

Overview on some operators

→ https://carpentries-incubator.github.io/workflows-nextflow/07-operators.html

Regular expressions !

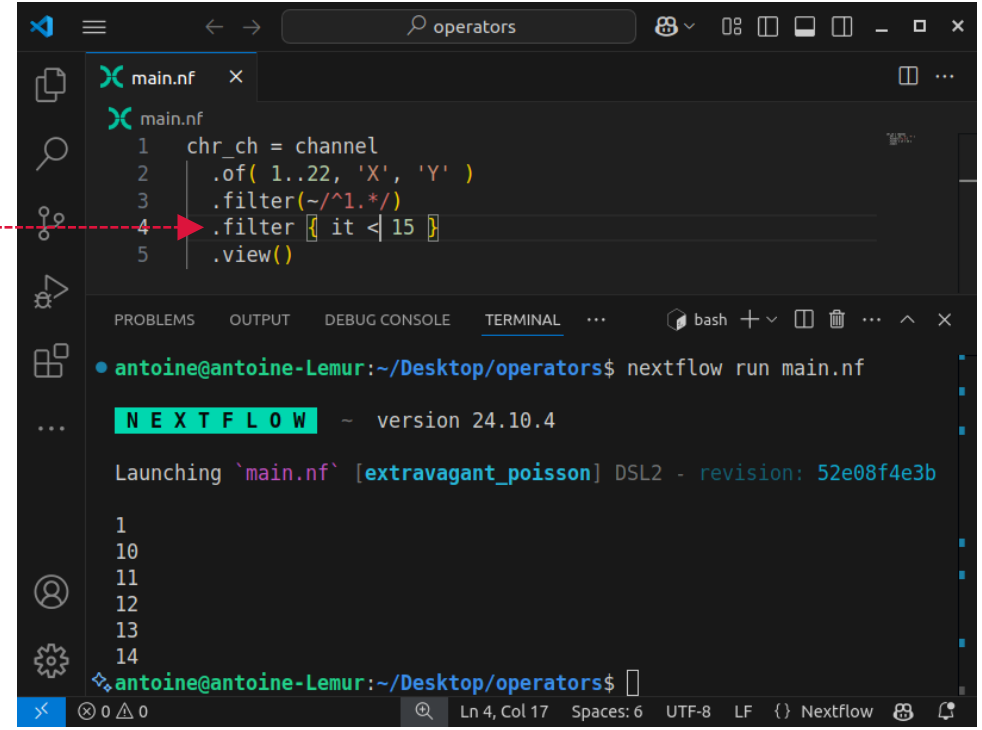


Operators

Overview on some operators

→ https://carpentries-incubator.github.io/workflows-nextflow/07-operators.html

Filtering !

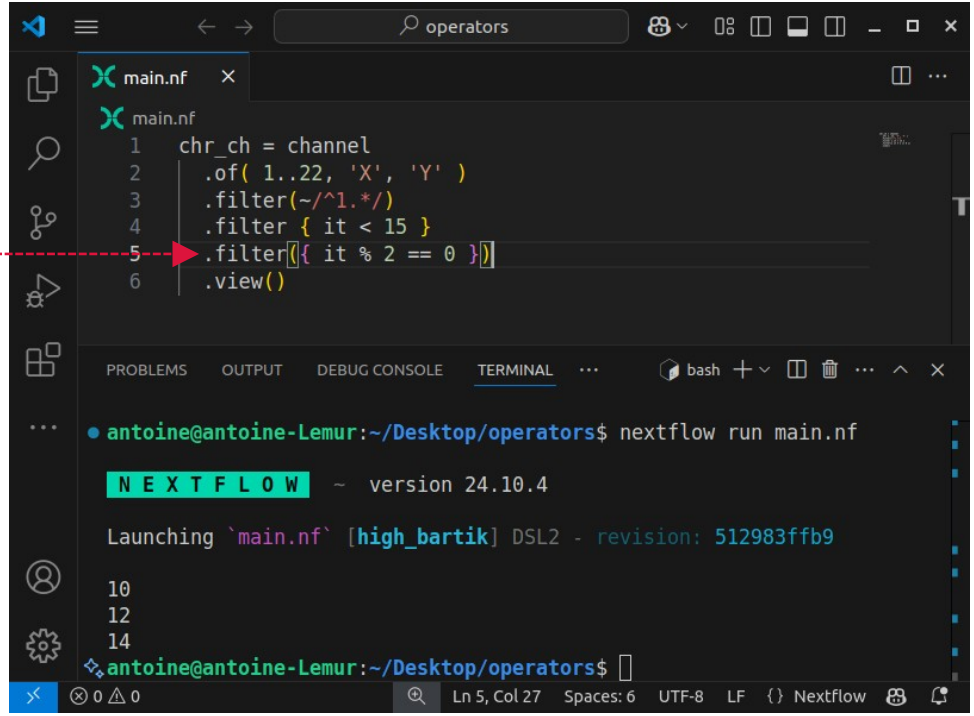


Operators

Overview on some operators

→ https://carpentries-incubator.github.io/workflows-nextflow/07-operators.html

Filtering !

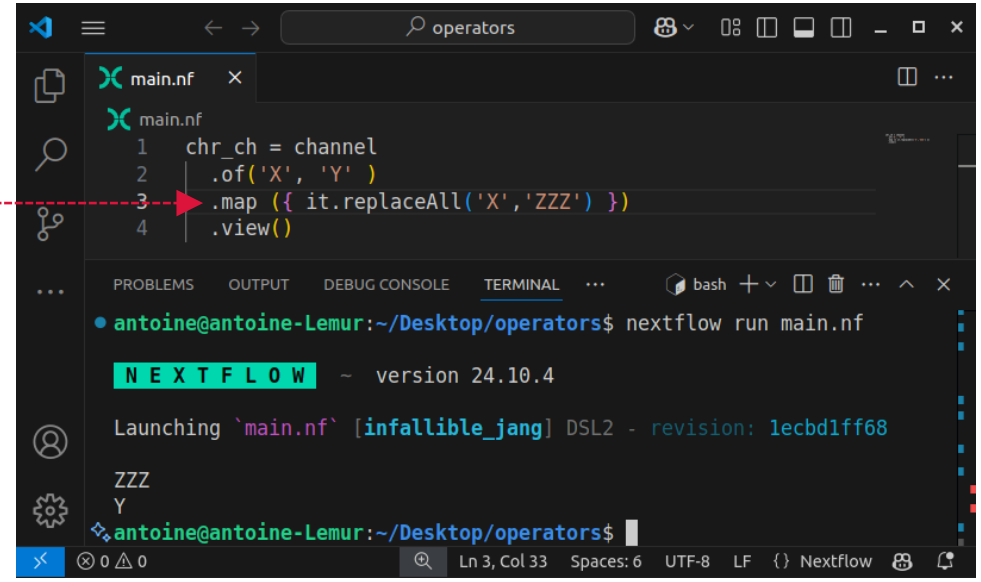


Operators

Overview on some operators

→ https://carpentries-incubator.github.io/workflows-nextflow/07-operators.html

Replace patterns !

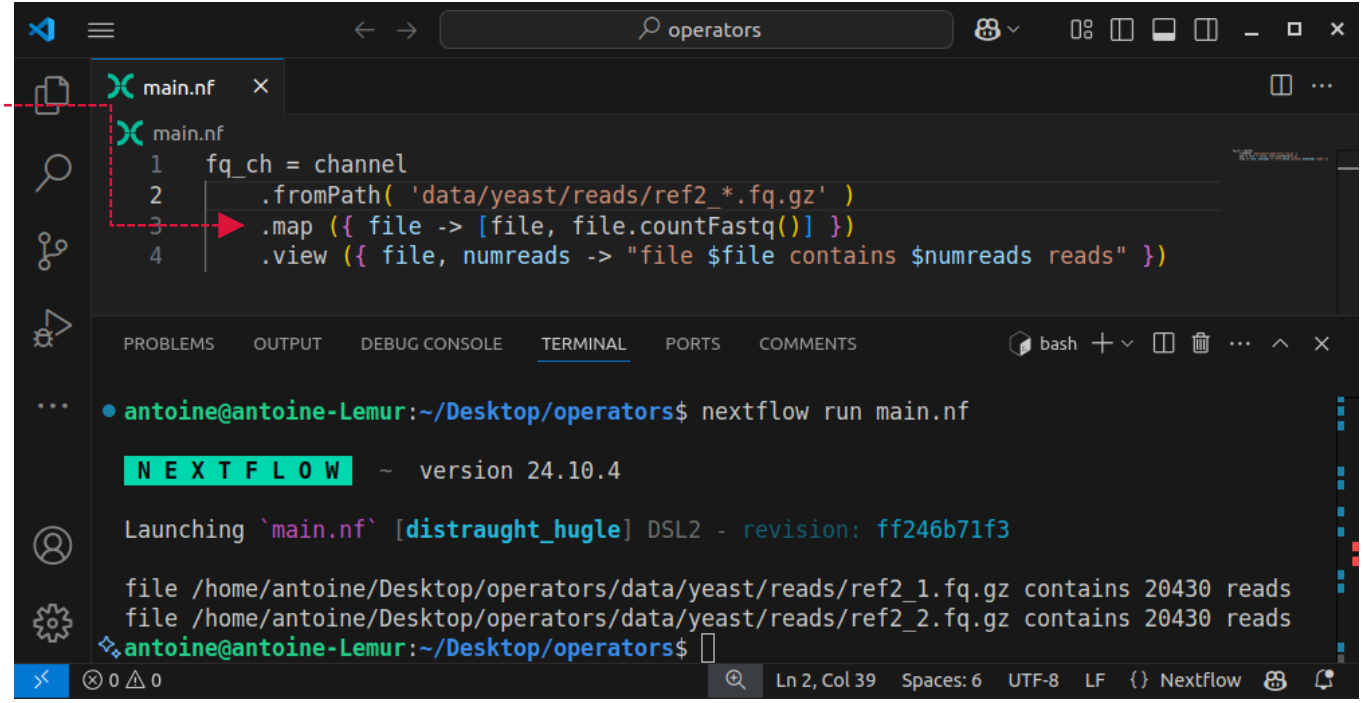


Operators

Overview on some operators

→ https://carpentries-incubator.github.io/workflows-nextflow/07-operators.html

Somewhat specific tools (fastq stuff) can be integrated

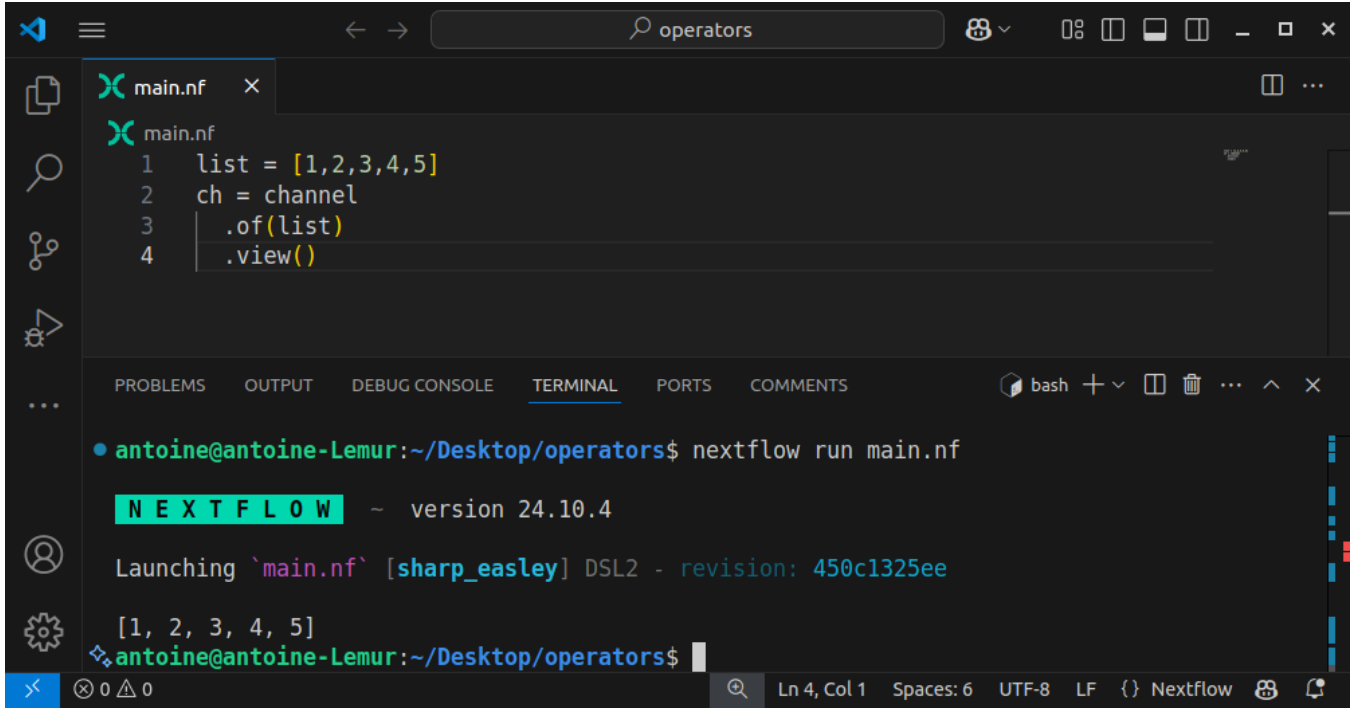


Operators

Overview on some operators

→ https://carpentries-incubator.github.io/workflows-nextflow/07-operators.html

Values *vs.* lists
Flatten *vs.* collect

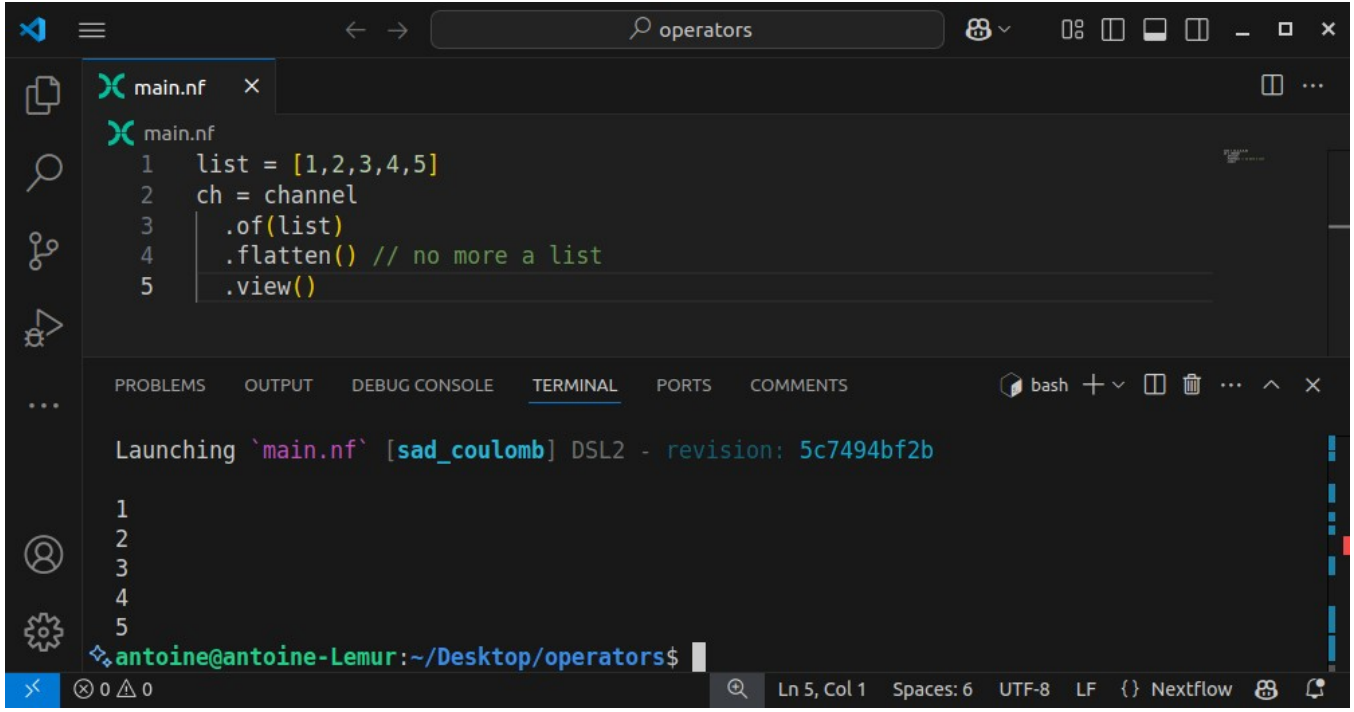


Operators

Overview on some operators

→ https://carpentries-incubator.github.io/workflows-nextflow/07-operators.html

Values *vs.* lists
Flatten *vs.* collect

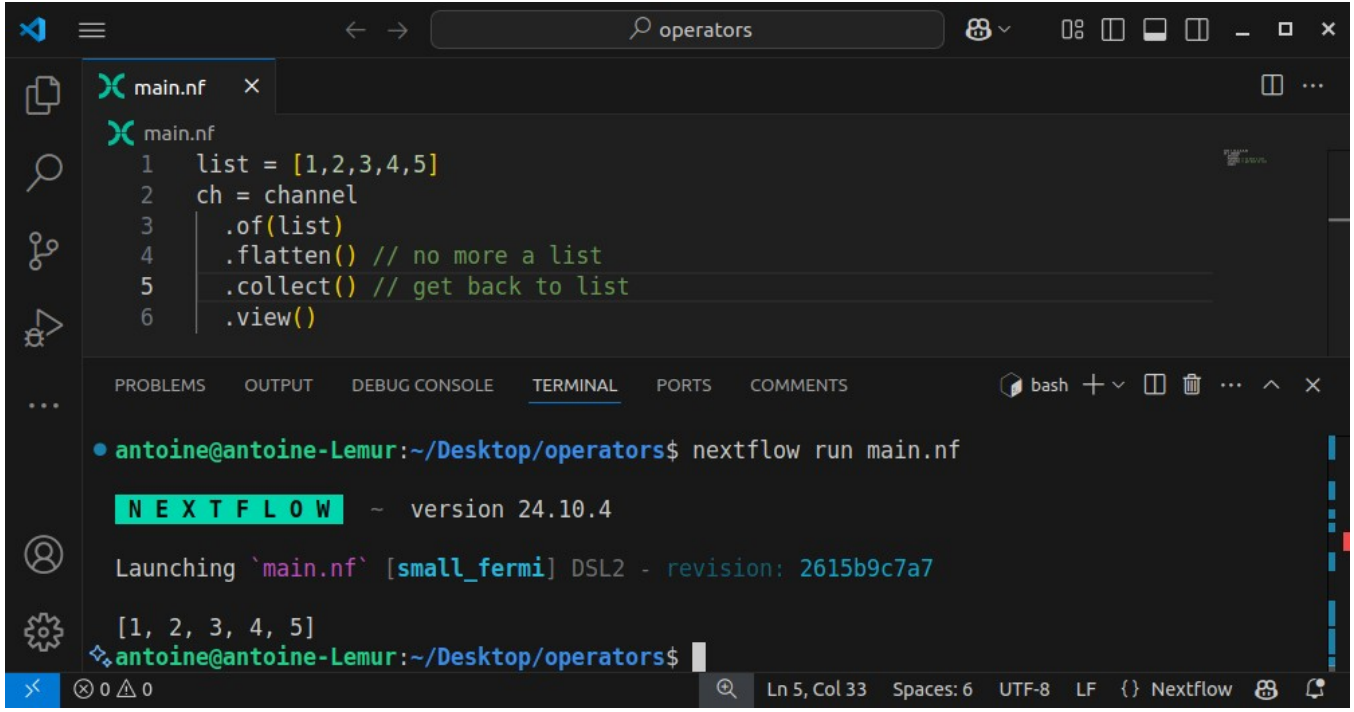


Operators

Overview on some operators

→ https://carpentries-incubator.github.io/workflows-nextflow/07-operators.html

Values *vs.* lists
Flatten *vs.* collect

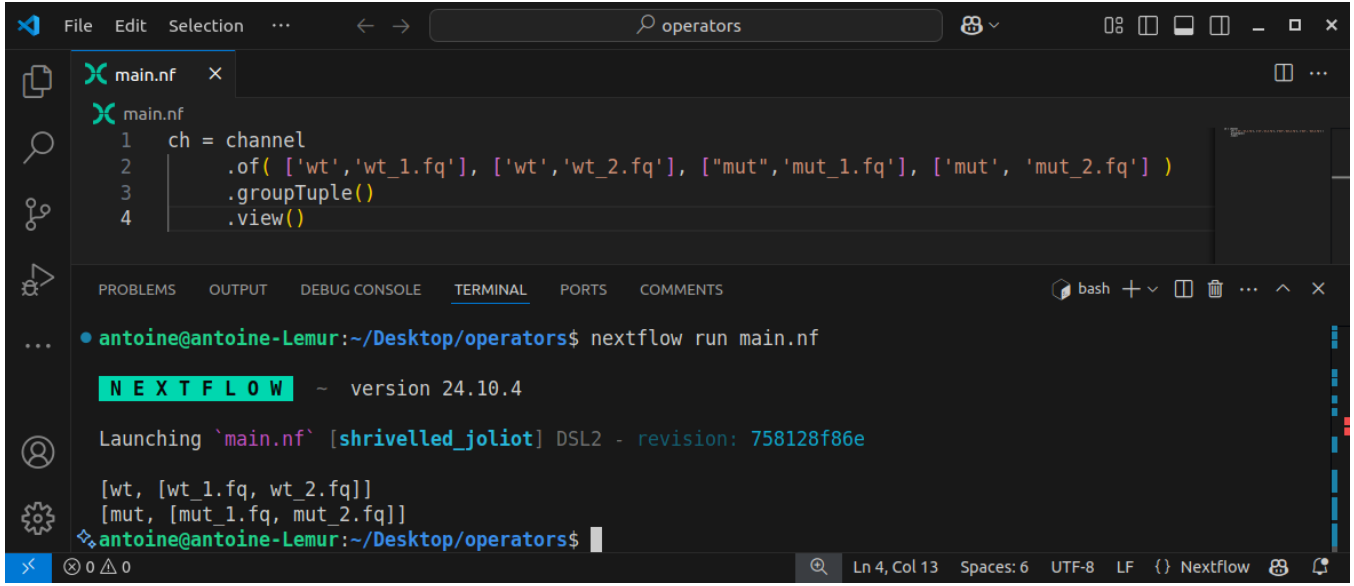


Operators

Overview on some operators

→ https://carpentries-incubator.github.io/workflows-nextflow/07-operators.html

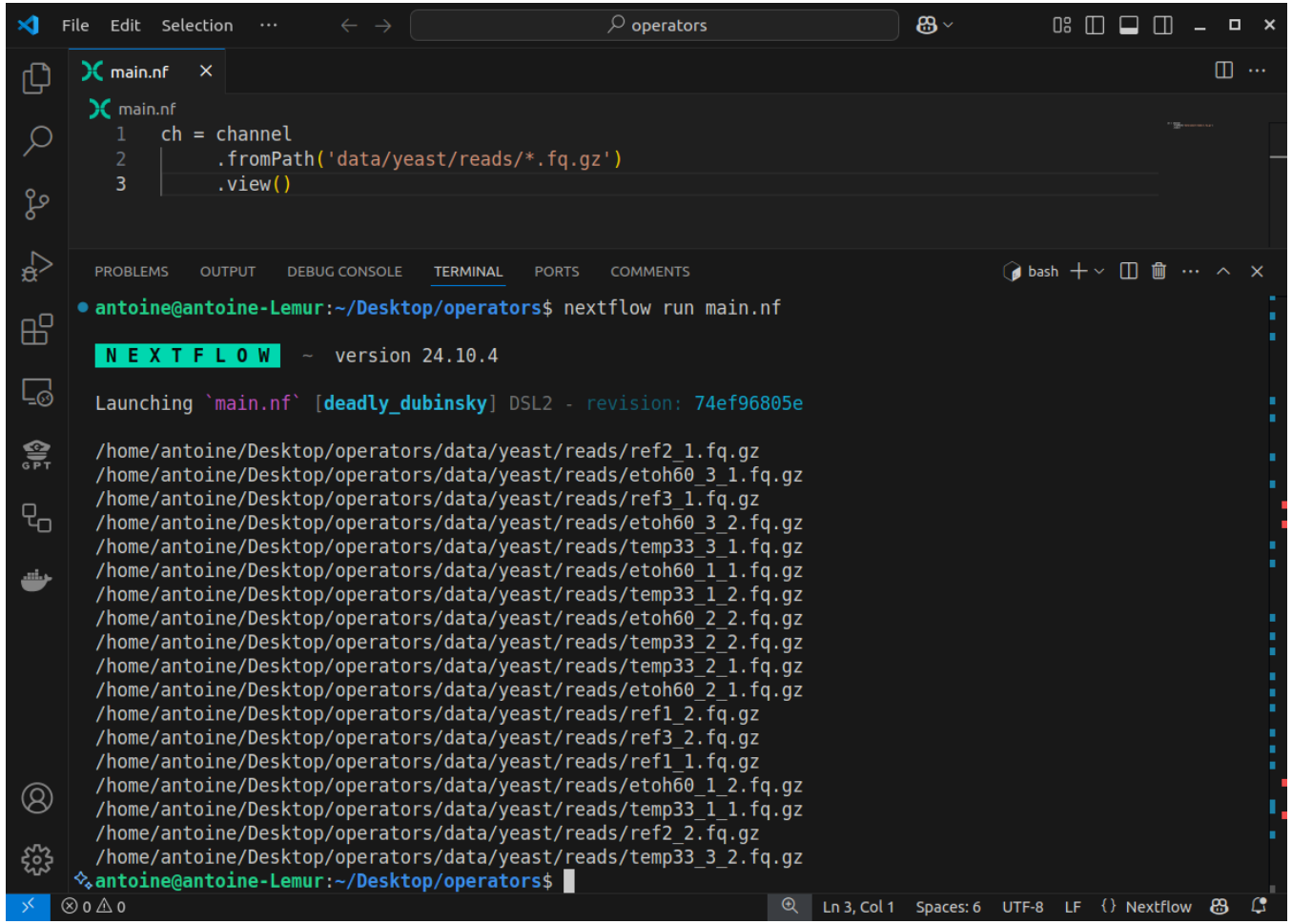
Group stuff together!
Such as read pairs



Operators

Overview on some operators

→ https://carpentries-incubator.github.io/workflows-nextflow/07-operators.html



Operators

Overview on some operators

→ https://carpentries-incubator.github.io/workflows-nextflow/07-operators.html

String manipulation

```
ch = channel
    .fromPath('data/yeast/reads/*.fq.gz')
    .map { file -> [ "Hello!! 😄 ", file ] }
    .view()
```

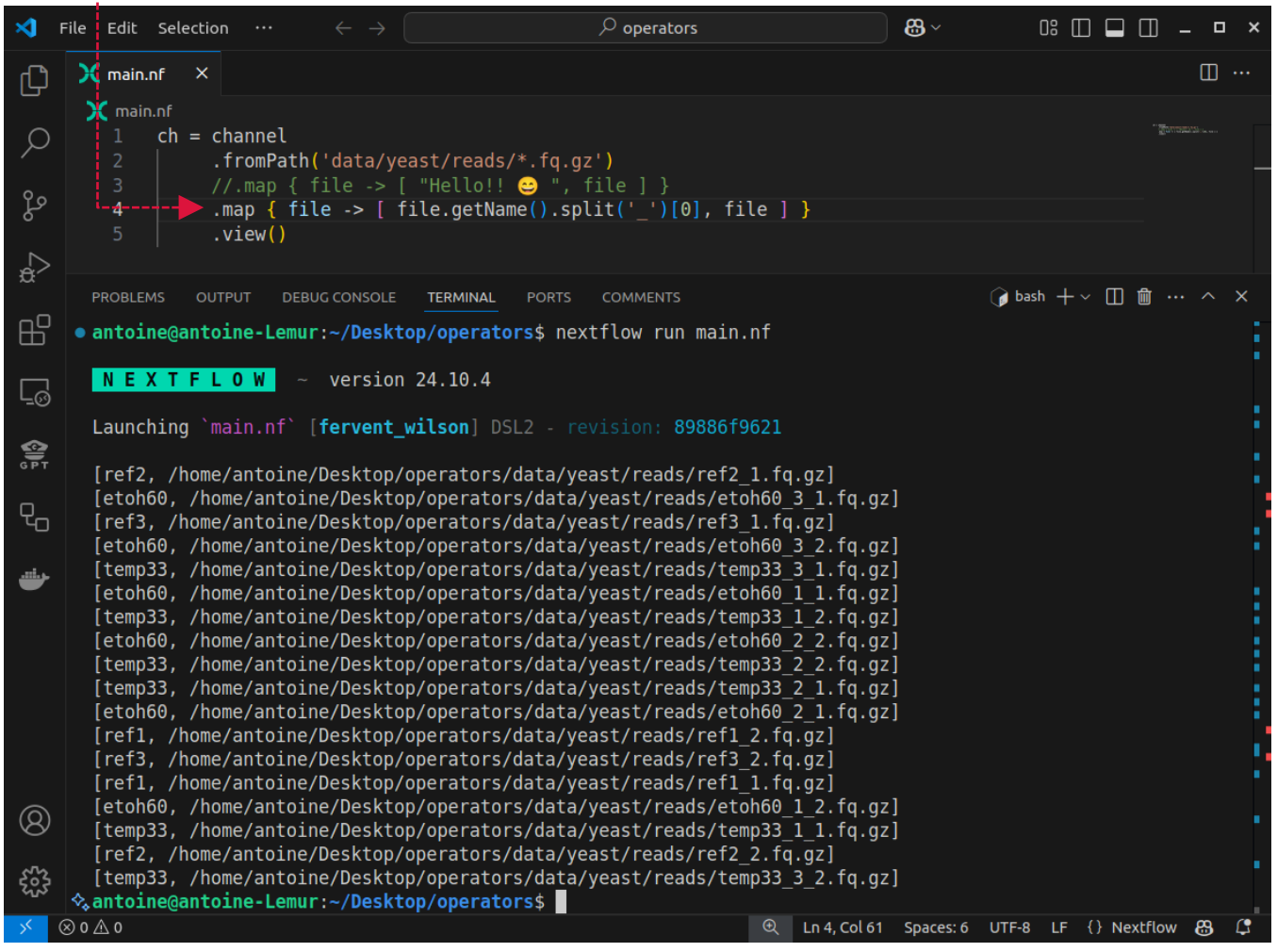
```
antoine@antoine-Lemur:~/Desktop/operators$ nextflow run main.nf

N E X T F L O W  ~  version 24.10.4

Launching `main.nf` [ridiculous_edison] DSL2 - revision: 67c7f5567e

[Hello!! 😄 , /home/antoine/Desktop/operators/data/yeast/reads/ref2_1.fq.gz]
[Hello!! 😄 , /home/antoine/Desktop/operators/data/yeast/reads/etoh60_3_1.fq.gz]
[Hello!! 😄 , /home/antoine/Desktop/operators/data/yeast/reads/ref3_1.fq.gz]
[Hello!! 😄 , /home/antoine/Desktop/operators/data/yeast/reads/etoh60_3_2.fq.gz]
[Hello!! 😄 , /home/antoine/Desktop/operators/data/yeast/reads/temp33_3_1.fq.gz]
[Hello!! 😄 , /home/antoine/Desktop/operators/data/yeast/reads/etoh60_1_1.fq.gz]
[Hello!! 😄 , /home/antoine/Desktop/operators/data/yeast/reads/temp33_1_2.fq.gz]
[Hello!! 😄 , /home/antoine/Desktop/operators/data/yeast/reads/etoh60_2_2.fq.gz]
[Hello!! 😄 , /home/antoine/Desktop/operators/data/yeast/reads/temp33_2_2.fq.gz]
[Hello!! 😄 , /home/antoine/Desktop/operators/data/yeast/reads/temp33_2_1.fq.gz]
[Hello!! 😄 , /home/antoine/Desktop/operators/data/yeast/reads/etoh60_2_1.fq.gz]
[Hello!! 😄 , /home/antoine/Desktop/operators/data/yeast/reads/ref1_2.fq.gz]
[Hello!! 😄 , /home/antoine/Desktop/operators/data/yeast/reads/ref3_2.fq.gz]
[Hello!! 😄 , /home/antoine/Desktop/operators/data/yeast/reads/ref1_1.fq.gz]
[Hello!! 😄 , /home/antoine/Desktop/operators/data/yeast/reads/etoh60_1_2.fq.gz]
[Hello!! 😄 , /home/antoine/Desktop/operators/data/yeast/reads/temp33_1_1.fq.gz]
[Hello!! 😄 , /home/antoine/Desktop/operators/data/yeast/reads/ref2_2.fq.gz]
[Hello!! 😄 , /home/antoine/Desktop/operators/data/yeast/reads/temp33_3_2.fq.gz]
antoine@antoine-Lemur:~/Desktop/operators$
```

Operators

Overview on some operators

→ https://carpentries-incubator.github.io/workflows-nextflow/07-operators.html

String manipulation



Operators

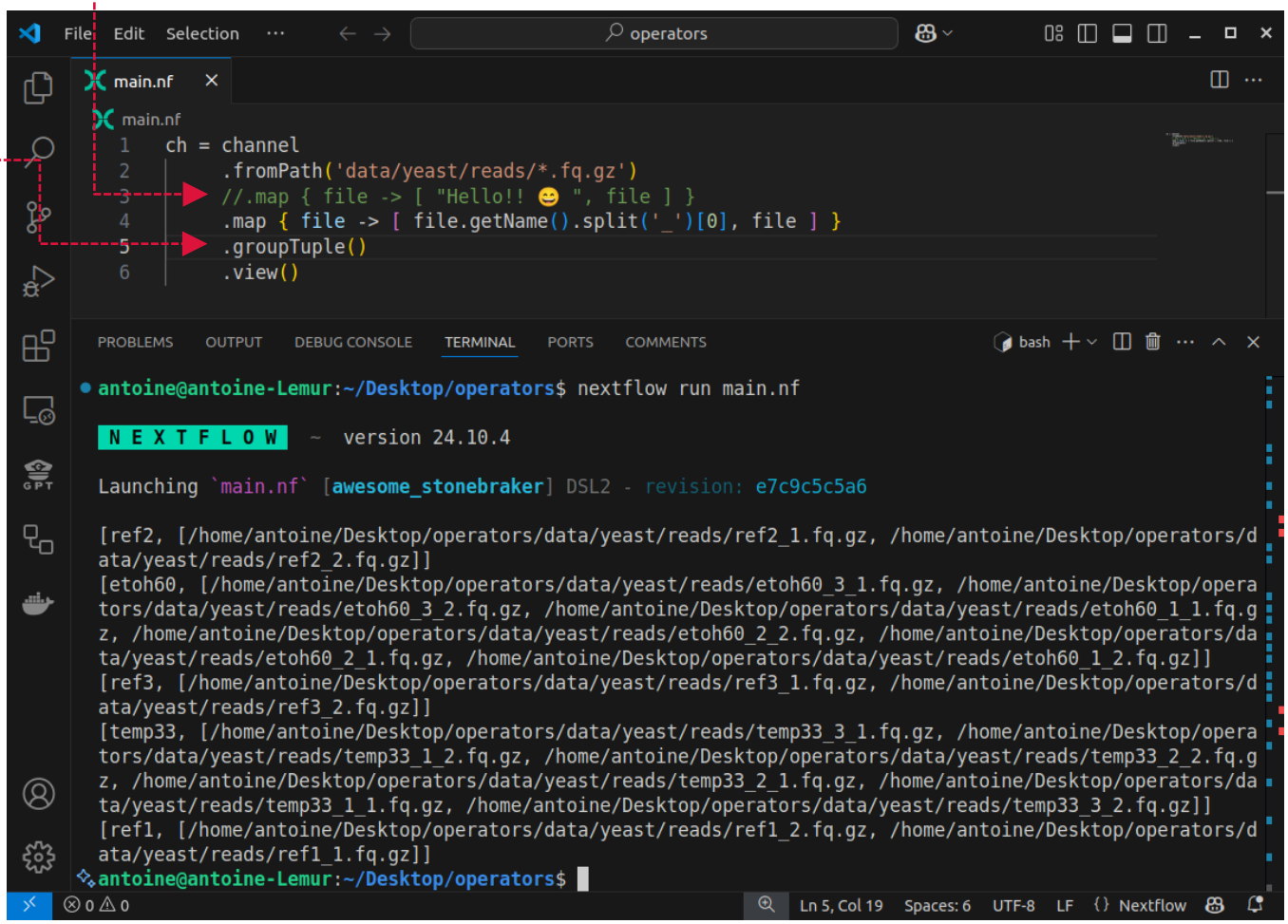
Overview on some operators

→ https://carpentries-incubator.github.io/workflows-nextflow/07-operators.html

String manipulation

+

Group stuff together!
Such as read pairs

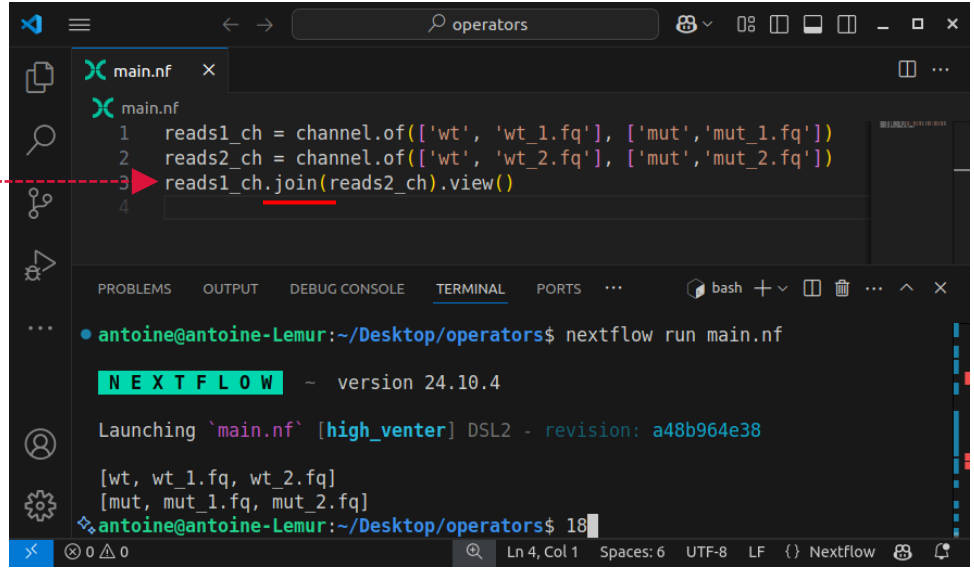


Operators

Overview on some operators

→ https://carpentries-incubator.github.io/workflows-nextflow/07-operators.html

Join lists !
(also mix)

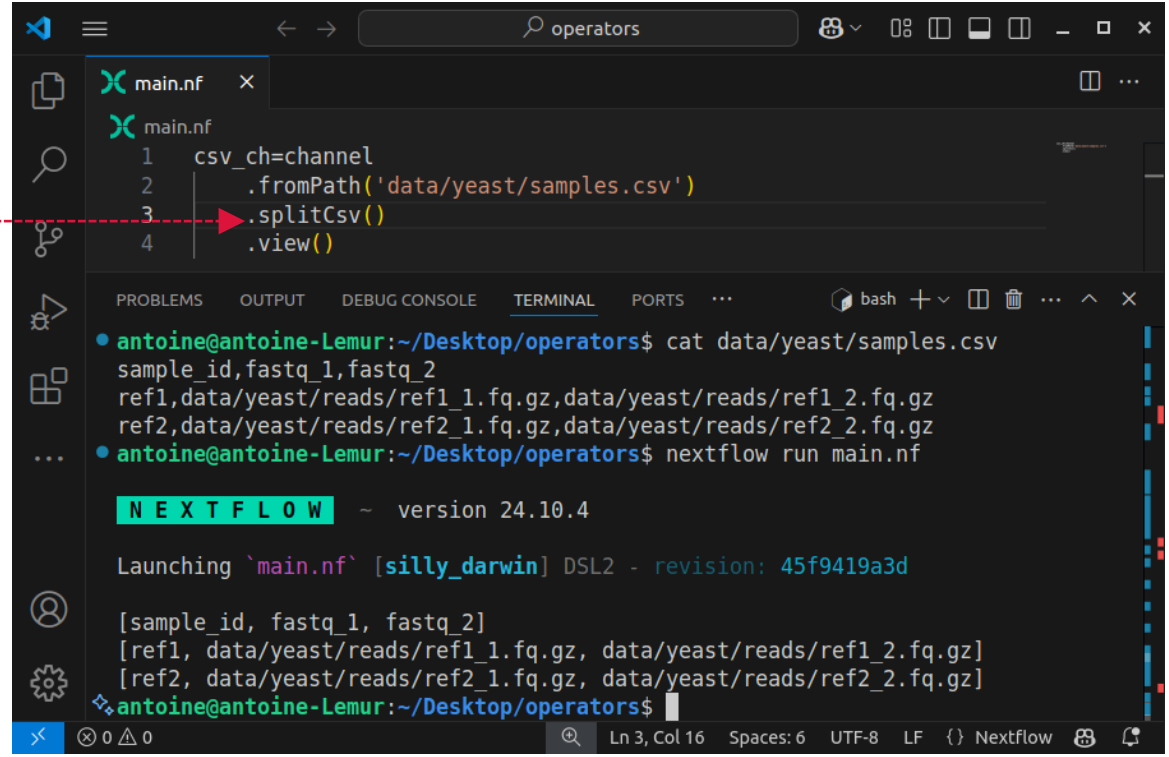


Operators

Overview on some operators

→ https://carpentries-incubator.github.io/workflows-nextflow/07-operators.html

Extract file content



Operators

Overview on some operators

→ https://carpentries-incubator.github.io/workflows-nextflow/07-operators.html

Extract file content
+ further manipulation

```
main.nf
csv_ch=channel
    .fromPath('data/yeast/samples.csv')
    .splitCsv()
    .view( {it[0]} )
```

```
antoine@antoine-Lemur:~/Desktop/operators$ cat data/yeast/samples.csv
sample_id,fastq_1,fastq_2
ref1,data/yeast/reads/ref1_1.fq.gz,data/yeast/reads/ref1_2.fq.gz
ref2,data/yeast/reads/ref2_1.fq.gz,data/yeast/reads/ref2_2.fq.gz
antoine@antoine-Lemur:~/Desktop/operators$ nextflow run main.nf

N E X T F L O W  ~  version 24.10.4

Launching `main.nf` [silly_darwin] DSL2 - revision: 45f9419a3d

[sample_id, fastq_1, fastq_2]
[ref1, data/yeast/reads/ref1_1.fq.gz, data/yeast/reads/ref1_2.fq.gz]
[ref2, data/yeast/reads/ref2_1.fq.gz, data/yeast/reads/ref2_2.fq.gz]
antoine@antoine-Lemur:~/Desktop/operators$ nextflow run main.nf

N E X T F L O W  ~  version 24.10.4

Launching `main.nf` [friendly_mahavira] DSL2 - revision: 3ae9ac7948

sample_id
ref1
ref2
antoine@antoine-Lemur:~/Desktop/operators$
```

Operators

Overview on some operators

→ https://carpentries-incubator.github.io/workflows-nextflow/07-operators.html

More useful examples

→ https://training.nextflow.io/latest/basic_training/operators/

Thank you for your attention!

Want to go further?
Ask us to organize Nextflow events, hackathons, ...

zurich-data-scientists.ch

nextflow.io/our_ambassadors.html

More info

https://nf-co.re

github.com/nf-core

nfcore.slack.com

youtube.com/nf-core

@nf_core@mstdn.science

@nf_core