nextflow

Workflow caching and checkpointing

Pre-hackathon training — Kemri-Wellcome Trust

March 17th 2025

Cris Tuñí-i-Domínguez
c.tuni.dominguez@gmail.com

A bit about me

Cris Tuñí i Domínguez

ctuni.dev

- PhD in Biomedicine student
- Bioinformatics scientist at Flomics Biotech S.L.
- From Barcelona
- Nextflow Ambassador
- Cat lover

Introduction

Objectives of today’s session

Workflow caching and checkpointing

Questions and objectives

- How can I restart a Nextflow workflow after an error?
- How can I add new data to a workflow without starting from the beginning?
- Where can I find intermediate data and results?

- Resume a Nextflow workflow using the -resume option.
- Restart a Nextflow workflow using new data.

Pre-requisites

For both reporting and config

You should run the examples on the same directory/environment that you used to carry out the previous days of the training.

Remember! We are learning how to:

- Re-run failed Nextflow runs

Documentation can be found here:
https://carpentries-incubator.github.io/workflows-nextflow/instructor/10-workflow_checkpoint_caching.html

Let’s begin!

Caching and checkpointing

Caching

A key feature of workflow management systems

re-entrancy, which is the ability to restart a pipeline after an error from the last successfully executed process. Re-entrancy enables time-consuming successfully completed steps, such as index creation, to be skipped when adding more data to a pipeline. This in turn leads to faster prototyping and development of workflows, and faster analyses of additional data.

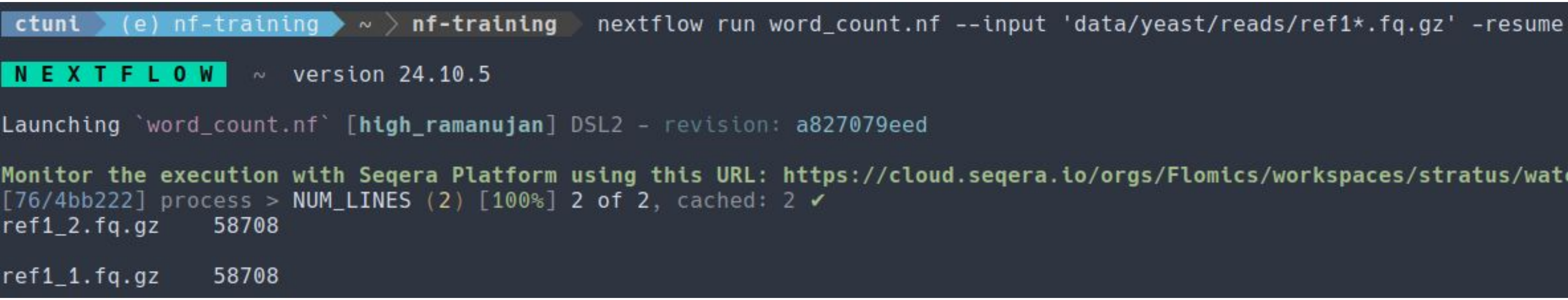
Nextflow achieves re-entrancy by automatically keeping track of all the processes executed in your pipeline via caching and checkpointing.

-resume

The magic argument

To restart from the last successfully executed process we add the command line option -resume to the Nextflow command.

For example, the command below would resume the word_count.nf script from the last successful process.

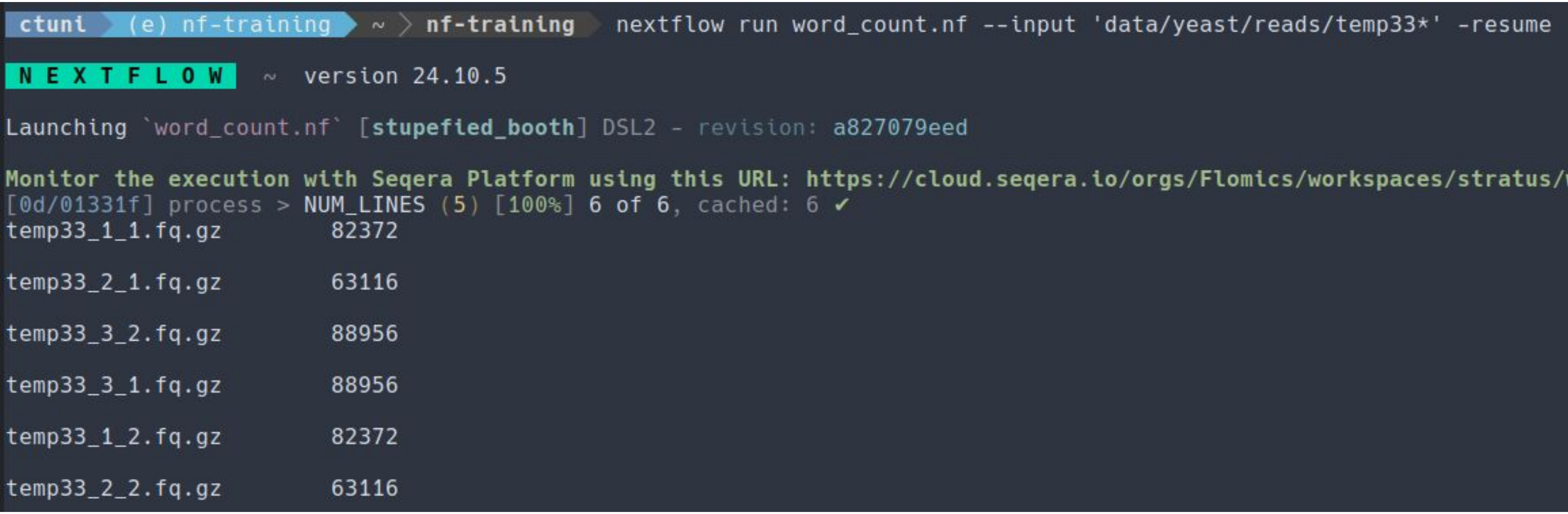


```
ctuni  (e) nf-training  ~  nf-training  nextflow run word_count.nf --input 'data/yeast/reads/ref1*.fq.gz' -resume

 N E X T F L O W   ~  version 24.10.5

Launching `word_count.nf` [high_ramanujan] DSL2 - revision: a827079eed

Monitor the execution with Seqera Platform using this URL: https://cloud.seqera.io/orgs/Flomics/workspaces/stratus/wat
[76/4bb222] process > NUM_LINES (2) [100%] 2 of 2, cached: 2 ✔
ref1_2.fq.gz    58708

ref1_1.fq.gz    58708
```

`-resume`

The magic argument

We can change the input of the same script and use `-resume` as well



```
ctuni  (e) nf-training  ~  nf-training  nextflow run word_count.nf --input 'data/yeast/reads/temp33*' -resume

 N E X T F L O W   ~  version 24.10.5

Launching `word_count.nf` [stupefied_booth] DSL2 - revision: a827079eed

Monitor the execution with Seqera Platform using this URL: https://cloud.seqera.io/orgs/Flomics/workspaces/stratus/
[0d/01331f] process > NUM_LINES (5) [100%] 6 of 6, cached: 6 ✔
temp33_1_1.fq.gz        82372

temp33_2_1.fq.gz        63116

temp33_3_2.fq.gz        88956

temp33_3_1.fq.gz        88956

temp33_1_2.fq.gz        82372

temp33_2_2.fq.gz        63116
```

`-resume`

How does it work?

Nextflow stores all intermediate files and task results during the execution of a workflow is `work` directory. It acts as a scratch space where **all the temporary data required for the workflow's execution is kept**. Within the work directory, Nextflow creates subdirectories named with unique hashes (e.g., work/ab/cd1234...). Each of these subdirectories corresponds to a specific process or task in the pipeline. The hashed directory names ensure that each task's outputs are isolated and uniquely identified.

The mechanism works by assigning a unique ID to each task. This unique ID is used to create a separate execution directory, within the work directory, where the tasks are executed and the results stored. **A task's unique ID is generated as a 128-bit hash number obtained from a composition of the task's:**

- Inputs values
- Input files
- Command line string
- Container ID
- Conda environment
- Environment modules
- Any executed scripts in the bin directory

`-resume`

How does it work?

When we resume a workflow Nextflow uses this unique ID to check if:

- The working directory exists
- It contains a valid command exit status
- It contains the expected output files.

If these conditions are satisfied, the task execution is skipped and the previously computed outputs are applied. When a task requires recomputation, ie. the conditions above are not fulfilled, the downstream tasks are automatically invalidated.

Therefore, if you modify some parts of your script, or alter the input data using -resume, will only execute the processes that are actually changed.

The execution of the processes that are not changed will be skipped and the cached result used instead.

This helps a lot when **testing or modifying** part of your pipeline without having to re-execute it from scratch.

-resume

Let's see it in action

We'll alter the timestamp on file `temp33_3_2.fq.gz` to trick Nextflow into thinking that it's a new file, and we'll re-run the word_count.nf script with resume.

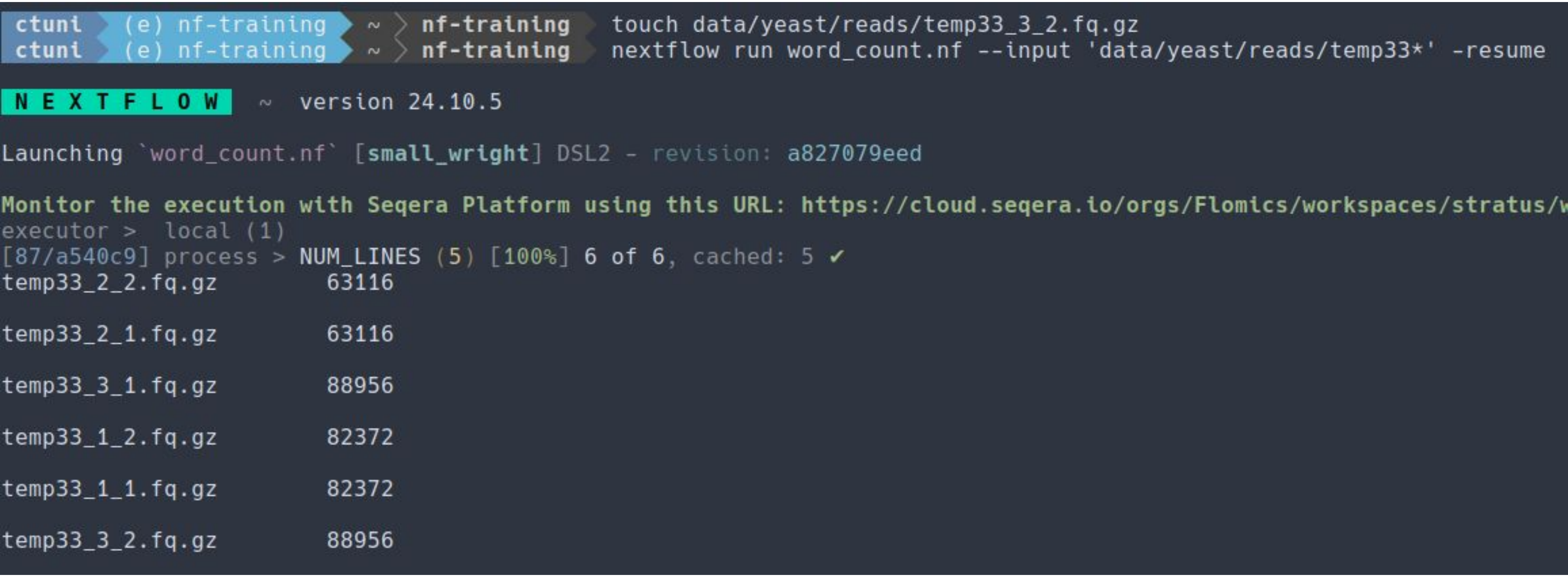
How many processes are going to get cached and how many will run anew?

```
touch
data/yeast/reads/temp33_3_2.fq.gz
```

```
nextflow run word_count.nf --input
'data/yeast/reads/temp33*' -resume
```

-resume

Let's see it in action



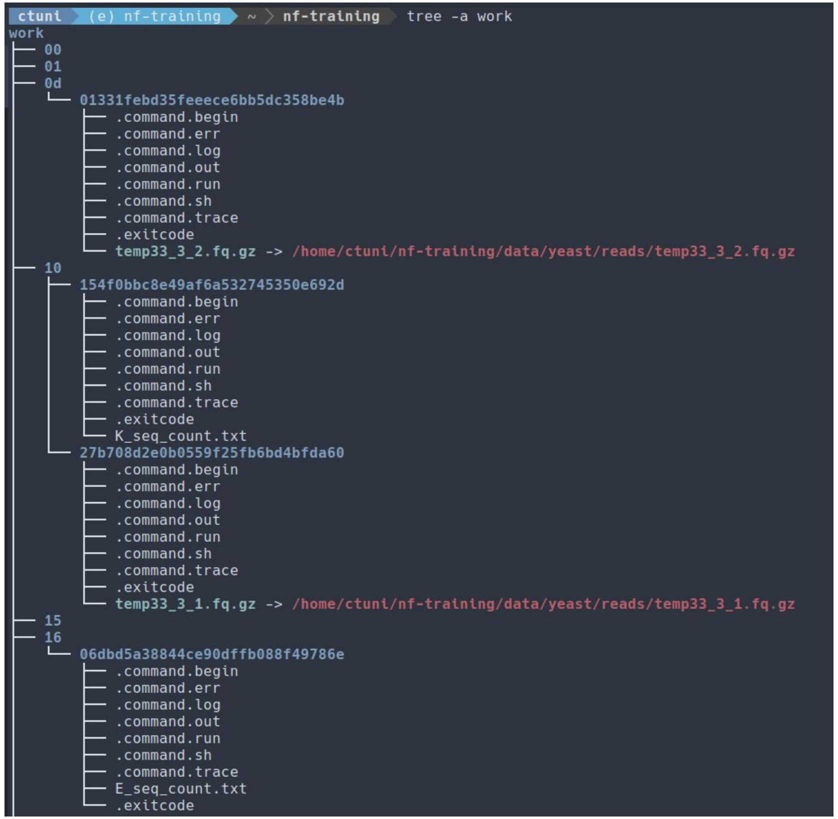
```
ctuni  (e) nf-training  ~  nf-training  touch data/yeast/reads/temp33_3_2.fq.gz
ctuni  (e) nf-training  ~  nf-training  nextflow run word_count.nf --input 'data/yeast/reads/temp33*' -resume

 N E X T F L O W   ~  version 24.10.5

Launching `word_count.nf` [small_wright] DSL2 - revision: a827079eed

Monitor the execution with Seqera Platform using this URL: https://cloud.seqera.io/orgs/Flomics/workspaces/stratus/w
executor >  local (1)
[87/a540c9] process > NUM_LINES (5) [100%] 6 of 6, cached: 5 ✔
temp33_2_2.fq.gz        63116

temp33_2_1.fq.gz        63116

temp33_3_1.fq.gz        88956

temp33_1_2.fq.gz        82372

temp33_1_1.fq.gz        82372

temp33_3_2.fq.gz        88956
```

The `work` directory

Where magic happens

By default the pipeline results are cached in the directory `work` where the pipeline is launched.

We can use the Bash `tree` command to list the contents of the work directory. Note: By default tree does not print hidden files (those beginning with a dot .). Use the -a to view all files.



```
 ctuni  (e) nf-training  ~  nf-training   tree -a work
work
├── 00
├── 01
├── 0d
│   └── 01331febd35feeece6bb5dc358be4b
│       ├── .command.begin
│       ├── .command.err
│       ├── .command.log
│       ├── .command.out
│       ├── .command.run
│       ├── .command.sh
│       ├── .command.trace
│       ├── .exitcode
│       └── temp33_3_2.fq.gz -> /home/ctuni/nf-training/data/yeast/reads/temp33_3_2.fq.gz
├── 10
│   ├── 154f0bbc8e49af6a532745350e692d
│   │   ├── .command.begin
│   │   ├── .command.err
│   │   ├── .command.log
│   │   ├── .command.out
│   │   ├── .command.run
│   │   ├── .command.sh
│   │   ├── .command.trace
│   │   ├── .exitcode
│   │   └── K_seq_count.txt
│   └── 27b708d2e0b0559f25fb6bd4bfda60
│       ├── .command.begin
│       ├── .command.err
│       ├── .command.log
│       ├── .command.out
│       ├── .command.run
│       ├── .command.sh
│       ├── .command.trace
│       ├── .exitcode
│       └── temp33_3_1.fq.gz -> /home/ctuni/nf-training/data/yeast/reads/temp33_3_1.fq.gz
├── 15
├── 16
│   └── 06dbd5a38844ce90dffb088f49786e
│       ├── .command.begin
│       ├── .command.err
│       ├── .command.log
│       ├── .command.out
│       ├── .command.run
│       ├── .command.sh
│       ├── .command.trace
│       ├── E_seq_count.txt
│       └── .exitcode
```

Task execution directory

Deep dive into the `work` directory

Within the `work` directory there are multiple task execution directories. There is one directory for each time a process is executed. These task directories are identified by the process execution hash. For example the task directory `fa/cd3e49b63eadd6248aa357083763c1` would be location for the process identified by the hash `fa/cd3e49` .

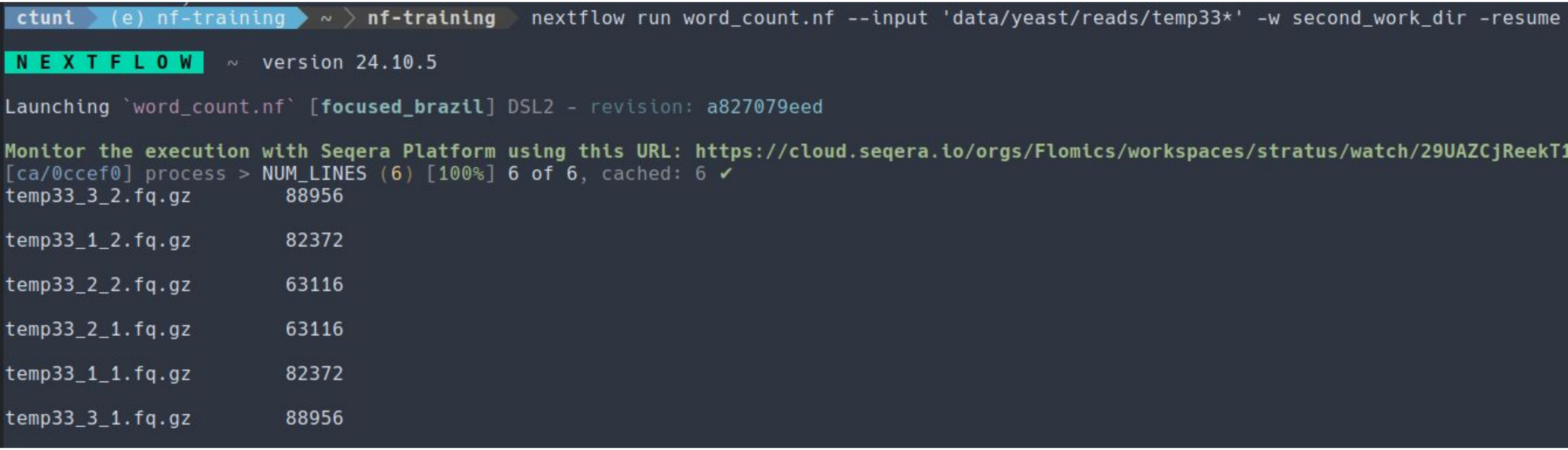
The task execution directory contains:

- `.command.sh`: The command script. The `.command.sh` file includes the specific instructions you've written to process your data or perform computations.
- `.command.run`: A Bash script generated by Nextflow to manage the execution environment of the `.command.sh` script. This script acts as a wrapper around `.command.sh`. It performs several tasks like setting up the task's environment variables, handling the task's pre and post execution (like moving inputs and outputs to correct locations, logging start and end times, handling errors, and ensuring resource limits are respected
- `.command.out`: The complete job standard output.
- `.command.err`: The complete job standard error.
- `.command.log`: The wrapper execution output.
- `.command.begin`: A file created as soon as the job is launched.
- `.exitcode`: A file containing the task exit code. This file is used to capture and store the exit status of the process that was run by the `.command.sh` script.
- Any task input files (symlinks)
- Any task output files

Specifying another work directory

For better control

Depending on your script, this work folder can take a lot of disk space. You can specify another work directory using the command line option `-w`. Note: **Using a different work directory will mean that any jobs will need to re-run from the beginning.**



```
ctuni  (e) nf-training  ~ > nf-training  nextflow run word_count.nf --input 'data/yeast/reads/temp33*' -w second_work_dir -resume

 N E X T F L O W   ~  version 24.10.5

Launching `word_count.nf` [focused_brazil] DSL2 - revision: a827079eed

Monitor the execution with Seqera Platform using this URL: https://cloud.seqera.io/orgs/Flomics/workspaces/stratus/watch/29UAZCjReekT1
[ca/0ccef0] process > NUM_LINES (6) [100%] 6 of 6, cached: 6 ✔
temp33_3_2.fq.gz        88956

temp33_1_2.fq.gz        82372

temp33_2_2.fq.gz        63116

temp33_2_1.fq.gz        63116

temp33_1_1.fq.gz        82372

temp33_3_1.fq.gz        88956
```

Clean the work directory

Saving some space

If you are sure you won't resume your pipeline execution, clean this folder periodically using the command `nextflow clean`.

```
nextflow clean [run_name|session_id] [options]
```

Supply the option `-n` to print names of files to be removed without deleting them, or `-f` to force the removal of the files. If you only want to remove files from a run but retain execution log entries and metadata, add the option `-k`. Multiple runs can be cleaned with the options, `-before`, `-after` or `-but` before the run name. For example, the command below would remove all the temporary files and log entries for runs before the run gigantic_minsky.

Clean the work directory

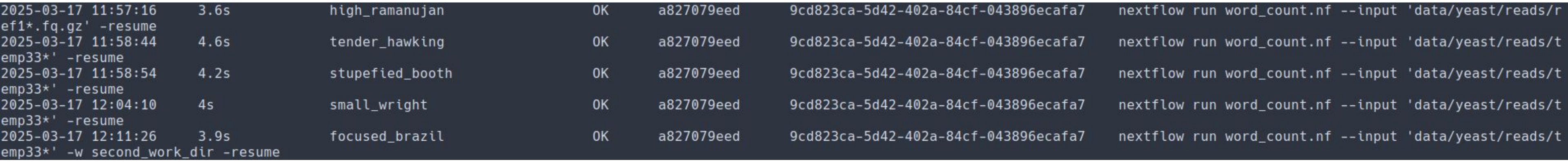
Saving some space

If you are sure you won't resume your pipeline execution, clean this folder periodically using the command `nextflow clean`.

```
nextflow clean [run_name|session_id]
[options]
```

For example, the command below would remove all the temporary files and log entries for runs before the run `small_wright`.

```
nextflow clean -f -before
small_wright
```



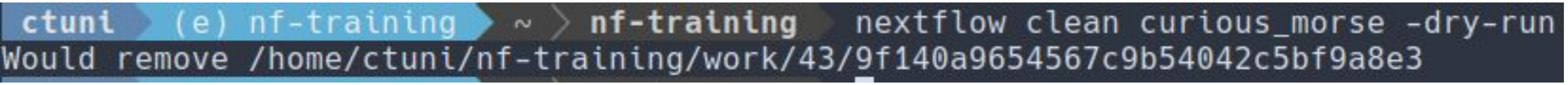
```
2025-03-17 11:57:16     3.6s            high_ramanujan          OK      a827079eed      9cd823ca-5d42-402a-84cf-043896ecafa7    nextflow run word_count.nf --input 'data/yeast/reads/r
ef1*.fq.gz' -resume
2025-03-17 11:58:44     4.6s            tender_hawking          OK      a827079eed      9cd823ca-5d42-402a-84cf-043896ecafa7    nextflow run word_count.nf --input 'data/yeast/reads/t
emp33*' -resume
2025-03-17 11:58:54     4.2s            stupefied_booth         OK      a827079eed      9cd823ca-5d42-402a-84cf-043896ecafa7    nextflow run word_count.nf --input 'data/yeast/reads/t
emp33*' -resume
2025-03-17 12:04:10     4s              small_wright            OK      a827079eed      9cd823ca-5d42-402a-84cf-043896ecafa7    nextflow run word_count.nf --input 'data/yeast/reads/t
emp33*' -resume
2025-03-17 12:11:26     3.9s            focused_brazil          OK      a827079eed      9cd823ca-5d42-402a-84cf-043896ecafa7    nextflow run word_count.nf --input 'data/yeast/reads/t
emp33*' -w second_work_dir -resume
```

Clean the work directory

Saving some space

With the `-dry-run` option, we can see which files we would delete

```
nextflow clean small_wright -dry-run
```



```
ctuni  (e) nf-training  ~  nf-training   nextflow clean curious_morse -dry-run
Would remove /home/ctuni/nf-training/work/43/9f140a9654567c9b54042c5bf9a8e3
```

Conclusion

Summary, key points, and utility of caching

- Nextflow automatically keeps track of all the processes executed in your pipeline via checkpointing.
- Nextflow caches intermediate data in task directories within the work directory.
- Nextflow caching and checkpointing allows re-entrancy into a workflow after a pipeline error or using new data, skipping steps that have been successfully executed.
- Re-entrancy is enabled using the `-resume` option.
- This is useful for debugging, developing, and testing purposes
- Don't forget to clean the work environment from time to time!

nextflow

Asante sana!

Hope to see you again!

Thank you