Nextflow: Might scalable and reproducible workflows be useful for more than science?

Minnebar 15

Tuesday, October 6, 2020

Michael L Heuer
UC Berkeley

https://github.com/heuermh

MINNEBAR 15

In this talk

- Repeatable, reproducible, and scalable
- An example Nextflow workflow that can address these concerns
  - https://github.com/heuermh/minnebar15-workflow
- A real-world Nextflow workflow in production analyzing SARS-CoV-2 clinical samples
  - https://github.com/nf-core/viralrecon

Repeatable, reproducible, and scalable

- Repeatable - If you had to do it again, on the same hardware and with the same data, could you?
- Reproducible - Could a colleague or collaborator reproduce your results, on their own hardware and with their own data?
- Scalable - If it works on one sample or a small cohort, will it scale up to handle many?

OPEN ACCESS

EDITORIAL

Ten Simple Rules for Reproducible Computational Research

Geir Kjetil Sandve, Anton Nekrutenko, James Taylor, Eivind Hovig

Published: October 24, 2013 • https://doi.org/10.1371/journal.pcbi.1003285

Citation: Sandve GK, Nekrutenko A, Taylor J, Hovig E (2013) Ten Simple Rules for Reproducible Computational Research. PLoS Comput Biol 9(10): e1003285. https://doi.org/10.1371/journal.pcbi.1003285

Editor: Philip E. Bourne, University of California San Diego, United States of America

Published: October 24, 2013

Good habits, not just for science!

"We want to emphasize that reproducibility is not only a moral responsibility with respect to the scientific field, but that a lack of reproducibility can also be a burden for you as an individual researcher. As an example, a good practice of reproducibility is necessary in order to allow previously developed methodology to be effectively applied on new data, or to allow reuse of code and results for new projects. **In other words, good habits of reproducibility may actually turn out to be a time-saver in the longer run.**"

https://doi.org/10.1371/journal.pcbi.1003285

Lack of reproducibility can be a burden

- A colleague leaves, monthly log error report stops working

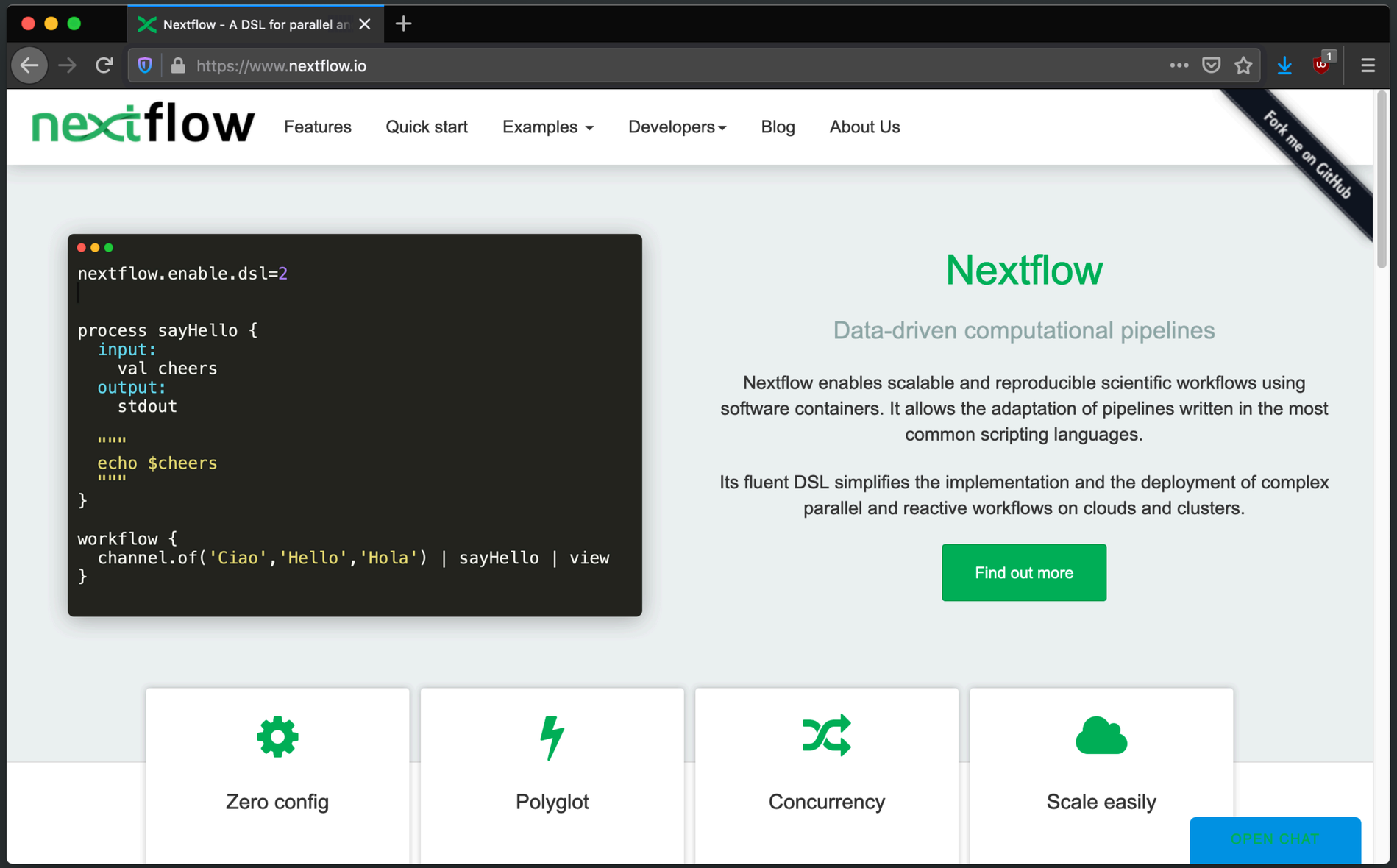
Lack of reproducibility can be a burden

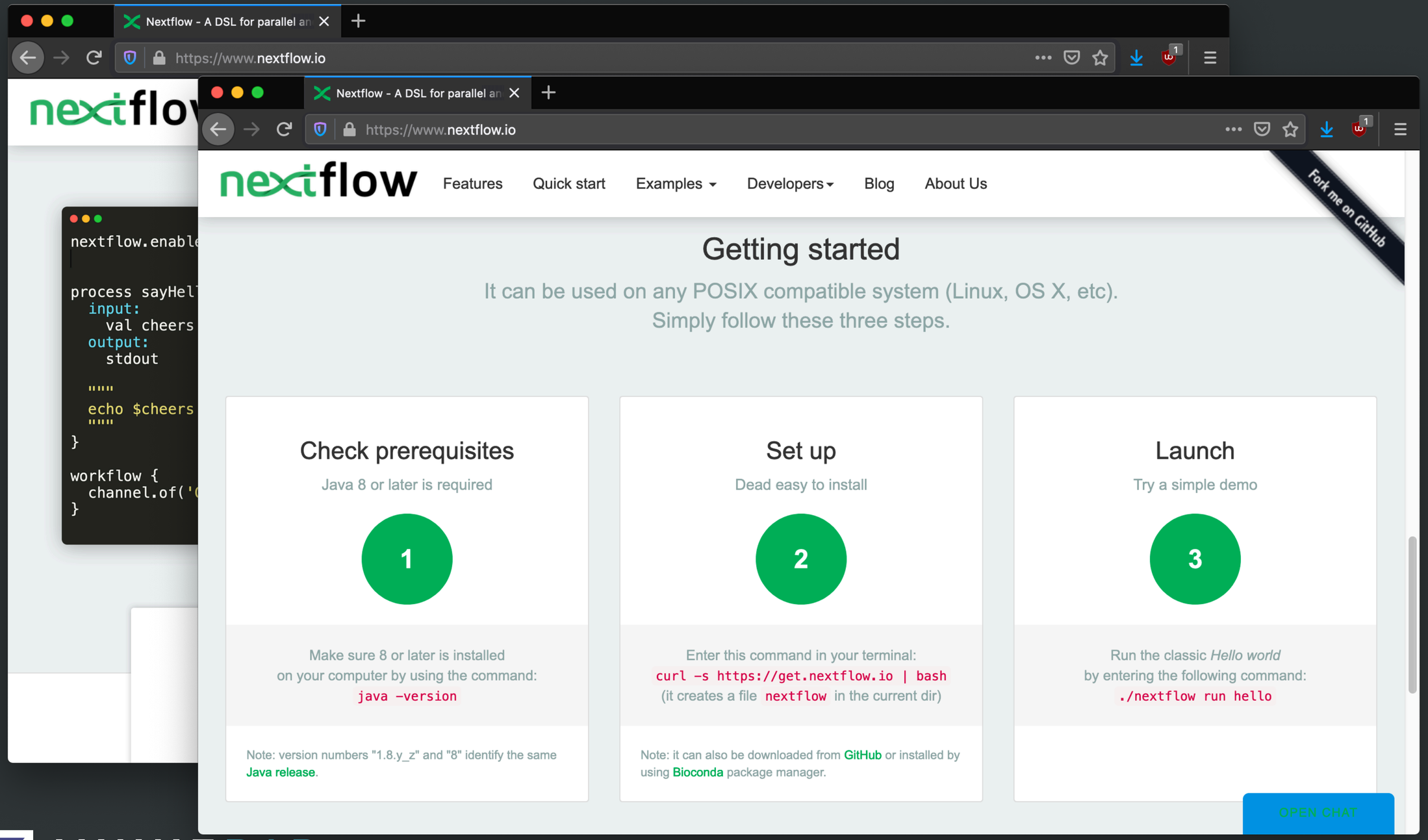
- A colleague leaves, monthly log error report stops working

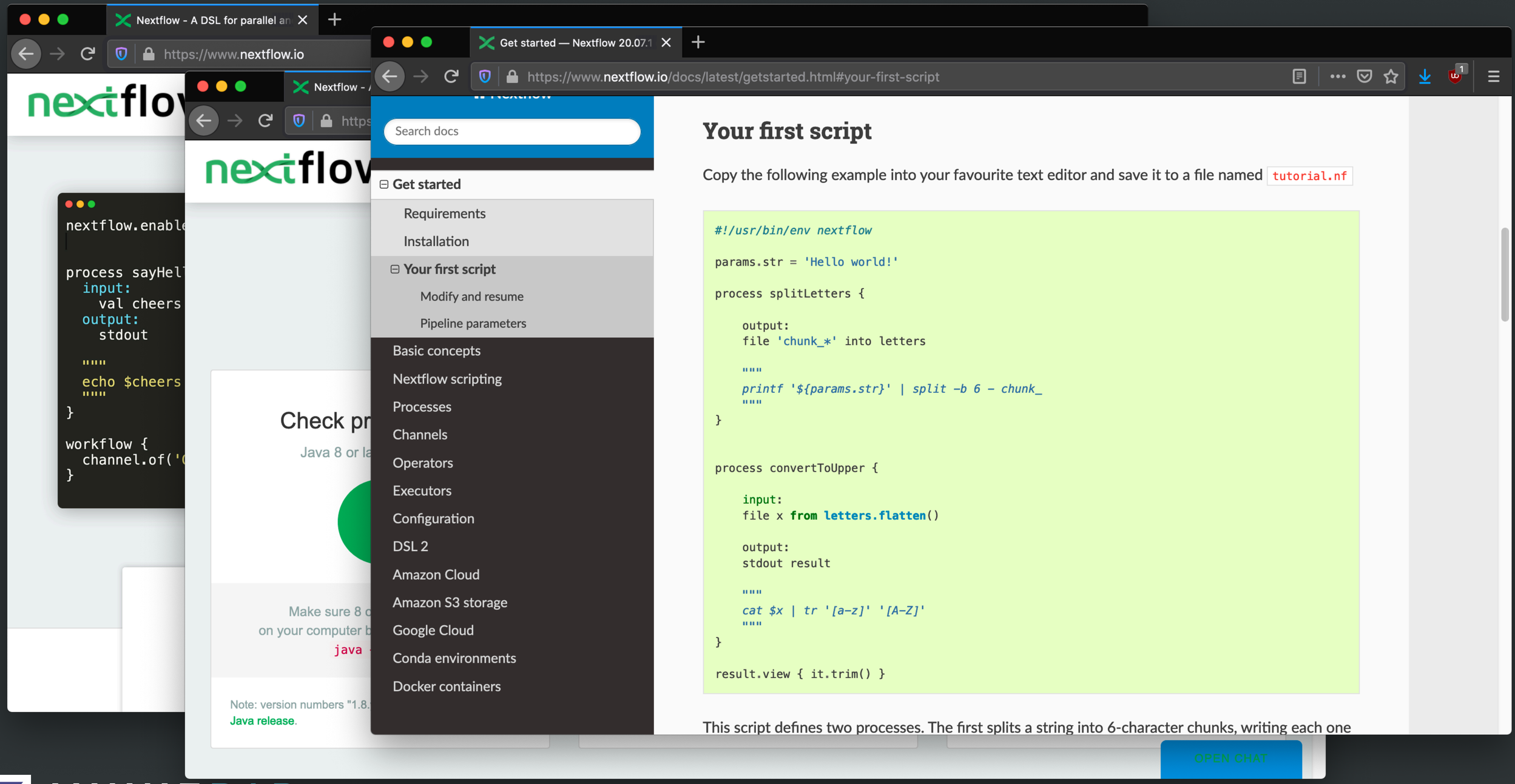
```
$ ls /home/former_colleague/log_stuff | cat
logErrors.pl
log_errors.py.2.bak
log_errors.py.old
log_errors(1).py
log_errors.py
```

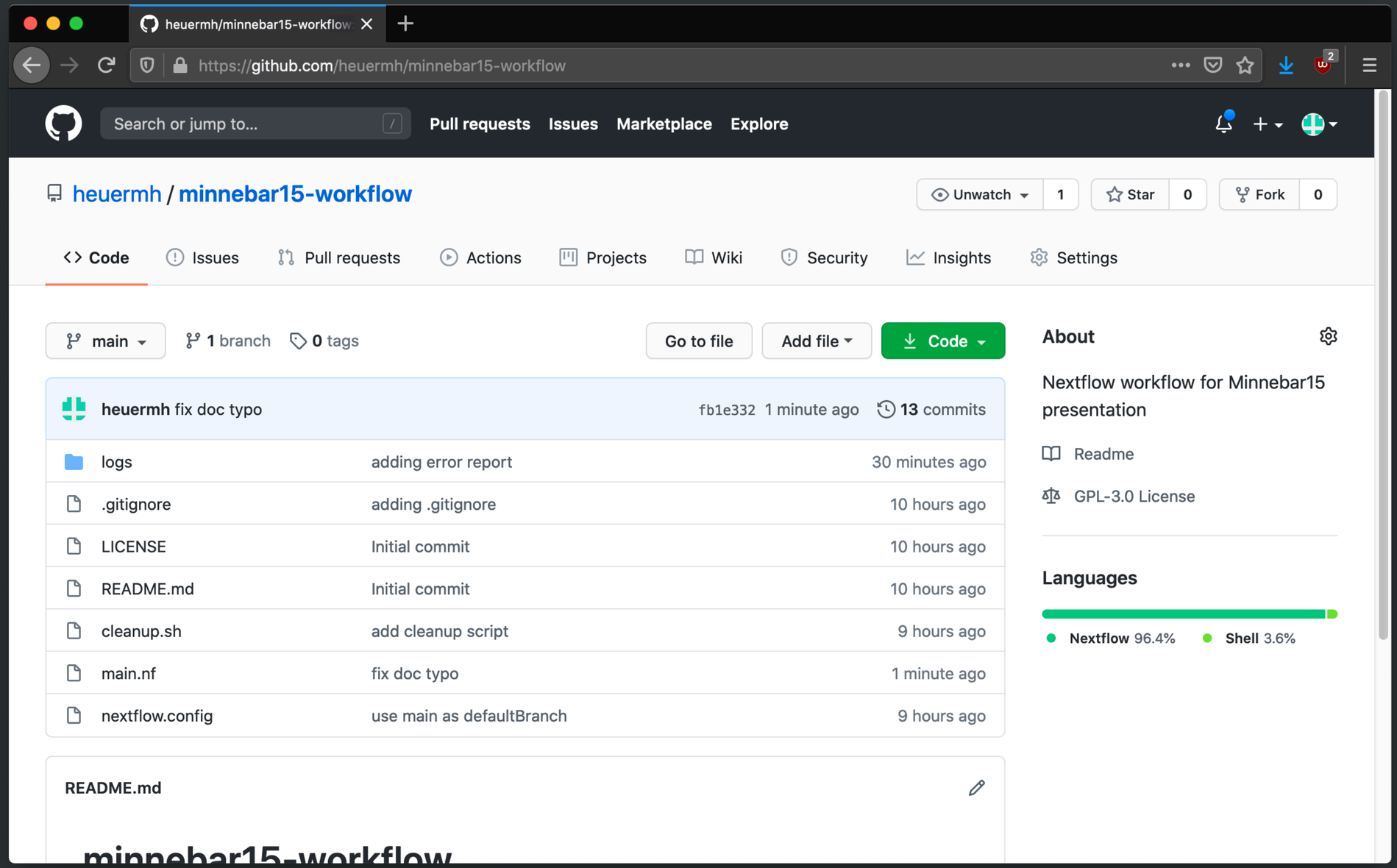
An example workflow

```
// find all the log files, gzip them to archive directory
$ find /my/logs -name "*.log" \
    -exec sh -c 'gzip -c {} > archive/{}.gz' \;

// find all the log files, grep for lines containing
// ERROR, and gzip them to the errors directory
$ find /my/logs -name "*.log" \
    -exec sh -c 'grep ERROR {} | gzip -c > errors/{}.gz' \;

// send an error report via email/twitter/slack
$ send_error_report.py errors/
```


Nextflow - A DSL for parallel an
https://www.nextflow.io
nextflow
Features
Quick start
Examples
Developers
Blog
About Us
Fork me on GitHub
nextflow.enable.dsl=2
process sayHello {
input:
val cheers
output:
stdout
"""
echo $cheers
"""
}
workflow {
channel.of('Ciao','Hello','Hola') | sayHello | view
}
Nextflow
Data-driven computational pipelines
Nextflow enables scalable and reproducible scientific workflows using software containers. It allows the adaptation of pipelines written in the most common scripting languages.
Its fluent DSL simplifies the implementation and the deployment of complex parallel and reactive workflows on clouds and clusters.
Find out more
Zero config
Polyglot
Concurrency
Scale easily
OPEN CHAT


Nextflow - A DSL for parallel an
https://www.nextflow.io
nextflow
Features
Quick start
Examples
Developers
Blog
About Us
Fork me on GitHub
Getting started
It can be used on any POSIX compatible system (Linux, OS X, etc).
Simply follow these three steps.
Check prerequisites
Java 8 or later is required
1
Make sure 8 or later is installed
on your computer by using the command:
java -version
Note: version numbers "1.8.y_z" and "8" identify the same Java release.
Set up
Dead easy to install
2
Enter this command in your terminal:
curl -s https://get.nextflow.io | bash
(it creates a file nextflow in the current dir)
Note: it can also be downloaded from GitHub or installed by using Bioconda package manager.
Launch
Try a simple demo
3
Run the classic Hello world
by entering the following command:
./nextflow run hello
OPEN CHAT
nextflow.enable
process sayHel
input:
val cheers
output:
stdout
"""
echo $cheers
"""
}
workflow {
channel.of('
}


Nextflow - A DSL for parallel an
https://www.nextflow.io
nextflow.enable
process sayHel
input:
val cheers
output:
stdout
"""
echo $cheers
"""
}
workflow {
channel.of('
}
Get started — Nextflow 20.07.1
https://www.nextflow.io/docs/latest/getstarted.html#your-first-script
Search docs
Get started
Requirements
Installation
Your first script
Modify and resume
Pipeline parameters
Basic concepts
Nextflow scripting
Processes
Channels
Operators
Executors
Configuration
DSL 2
Amazon Cloud
Amazon S3 storage
Google Cloud
Conda environments
Docker containers
Your first script
Copy the following example into your favourite text editor and save it to a file named tutorial.nf
#!/usr/bin/env nextflow
params.str = 'Hello world!'
process splitLetters {
output:
file 'chunk_*' into letters
"""
printf '${params.str}' | split -b 6 - chunk_
"""
}
process convertToUpper {
input:
file x from letters.flatten()
output:
stdout result
"""
cat $x | tr '[a-z]' '[A-Z]'
"""
}
result.view { it.trim() }
This script defines two processes. The first splits a string into 6-character chunks, writing each one
Check pr
Java 8 or la
Make sure 8 c
on your computer b
java
Note: version numbers "1.8.
Java release.
OPEN CHAT


heuermh/minnebar15-workflow
https://github.com/heuermh/minnebar15-workflow
Search or jump to...
Pull requests
Issues
Marketplace
Explore
heuermh / minnebar15-workflow
Unwatch 1
Star 0
Fork 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
main
1 branch
0 tags
Go to file
Add file
Code
heuermh fix doc typo
fb1e332 1 minute ago
13 commits
logs adding error report 30 minutes ago
.gitignore adding .gitignore 10 hours ago
LICENSE Initial commit 10 hours ago
README.md Initial commit 10 hours ago
cleanup.sh add cleanup script 9 hours ago
main.nf fix doc typo 1 minute ago
nextflow.config use main as defaultBranch 9 hours ago
README.md
minnebar15-workflow
About
Nextflow workflow for Minnebar15 presentation
Readme
GPL-3.0 License
Languages
Nextflow 96.4%
Shell 3.6%

MINNEBAR 15

78 lines (64 sloc) | 1.28 KB

Raw | Blame

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

//
// Provide default values for input parameters
//
params.logs = "logs"
params.archive = "archive"
params.errors = "errors"

//
// Find all log files in the log file directory
//
logFiles = Channel.fromPath("${params.logs}/**.log")

//
// Copy found log files to two channels (similar to `tee`)
//
(toArchive, toFilter) = logFiles.into(2)

//
// Gzip input files and publish them via file copy to the archive directory
```

```
//
(toArchive, toFilter) = logFiles.into(2)

//
// Gzip input files and publish them via file copy to the archive directory
//
process gzipToArchive {
  publishDir "${params.archive}", mode: 'copy'

  input:
    file f from toArchive
  output:
    file "${f}.gz"

  """
  gzip -c $f > ${f}.gz
  """
}

//
// Grep input files for lines containing ERROR into output files
//
process grepErrors {
```

```
32    gzip -c $f > ${f}.gz
33    """
34  }
35
36  //
37  // Grep input files for lines containing ERROR into output files
38  //
39  process grepErrors {
40
41    input:
42      file f from toFilter
43    output:
44      file "${f}.errors" into errors
45
46    """
47    grep ERROR $f > ${f}.errors || true
48    """
49  }
50
51  //
52  // Gzip input files and publish them via file copy to the errors directory
53  //
54  process gzipToErrors {
55    publishDir "${params.errors}", mode: 'copy'
56
```

```
    """
    grep ERROR $f > ${f}.errors || true
    """
}

//
// Gzip input files and publish them via file copy to the errors directory
//
process gzipToErrors {
    publishDir "${params.errors}", mode: 'copy'

    input:
      file f from errors
    output:
      file "${f}.gz" into toReport

    """
    gzip -c $f > ${f}.gz
    """
}

//
// Send an error report via email/twitter/slack
//
process errorReport {
```

```
59    output:
60      file "${f}.gz" into toReport
61
62    """
63    gzip -c $f > ${f}.gz
64    """
65  }
66
67  //
68  // Send an error report via email/twitter/slack
69  //
70  process errorReport {
71    input:
72    file f from toReport
73
74    // replace with actual script
75    """
76    [ -s $f ] || echo "Found errors in $f"
77    """
78  }
```

```
minnebar15-workflow main
$
```

```
minnebar15-workflow main
$ nextflow run main.nf
N E X T F L O W  ~  version 20.07.1
Launching `main.nf` [agitated_euler] - revision: 38d3e76045
executor >  local (16)
[23/86835d] process > gzipToArchive (3) [100%] 4 of 4 ✓
[72/56a729] process > grepErrors (2)    [100%] 4 of 4 ✓
[a1/fc5d70] process > gzipToErrors (4)  [100%] 4 of 4 ✓
[7d/5b7865] process > errorReport (4)   [100%] 4 of 4 ✓

minnebar15-workflow main
$
```

```
minnebar15-workflow main
$ nextflow run main.nf
N E X T F L O W  ~  version 20.07.1
Launching `main.nf` [agitated_euler] - revision: 38d3e76045
executor >  local (16)
[23/86835d] process > gzipToArchive (3) [100%] 4 of 4 ✓
[72/56a729] process > grepErrors (2)    [100%] 4 of 4 ✓
[a1/fc5d70] process > gzipToErrors (4)  [100%] 4 of 4 ✓
[7d/5b7865] process > errorReport (4)   [100%] 4 of 4 ✓

minnebar15-workflow main
$ ls archive/
bar.log.gz baz.log.gz foo.log.gz qux.log.gz
minnebar15-workflow main
$ ls errors/
bar.log.errors.gz baz.log.errors.gz foo.log.errors.gz qux.log.errors.gz
minnebar15-workflow main
$
```

```
[7d/5b7865] process > errorReport (4)   [100%] 4 of 4 ✓

minnebar15-workflow main
$ ls archive/
bar.log.gz baz.log.gz foo.log.gz qux.log.gz
minnebar15-workflow main
$ ls errors/
bar.log.errors.gz baz.log.errors.gz foo.log.errors.gz qux.log.errors.gz
minnebar15-workflow main
$ ./cleanup.sh
minnebar15-workflow main
$ nextflow run main.nf
N E X T F L O W  ~  version 20.07.1
Launching `main.nf` [loving_hypatia] - revision: 38d3e76045
executor >  local (16)
[09/8628c1] process > gzipToArchive (4) [100%] 4 of 4 ✓
[6a/fb64c3] process > grepErrors (2)    [100%] 4 of 4 ✓
[69/197fb6] process > gzipToErrors (4)  [100%] 4 of 4 ✓
[9b/94aaf3] process > errorReport (4)   [100%] 4 of 4 ✓

minnebar15-workflow main
$
```

```
$ ls errors/
bar.log.errors.gz baz.log.errors.gz foo.log.errors.gz qux.log.errors.gz
minnebar15-workflow main
$ ./cleanup.sh
minnebar15-workflow main
$ nextflow run main.nf
N E X T F L O W  ~  version 20.07.1
Launching `main.nf` [loving_hypatia] - revision: 38d3e76045
executor >  local (16)
[09/8628c1] process > gzipToArchive (4) [100%] 4 of 4 ✓
[6a/fb64c3] process > grepErrors (2)    [100%] 4 of 4 ✓
[69/197fb6] process > gzipToErrors (4)  [100%] 4 of 4 ✓
[9b/94aaf3] process > errorReport (4)   [100%] 4 of 4 ✓

minnebar15-workflow main
$ ls archive/
bar.log.gz baz.log.gz foo.log.gz qux.log.gz
minnebar15-workflow main
$ ls errors/
bar.log.errors.gz baz.log.errors.gz foo.log.errors.gz qux.log.errors.gz
minnebar15-workflow main
$
```

Repeatable, reproducible, and scalable

- Repeatable - If you had to do it again, on the same hardware and with the same data, could you?
- Reproducible - Could a colleague or collaborator reproduce your results, on their own hardware and with their own data?
- Scalable - If it works on one sample or a small cohort, will it scale up to handle many?

```
tmp
$ ls -R logs/
foo.log
tmp
$
```

```
tmp
$ ls -R logs/
foo.log
tmp
$ nextflow run heuermh/minnebar15-workflow --logs logs/
N E X T F L O W  ~  version 20.07.1
Launching `heuermh/minnebar15-workflow` [lethal_ritchie] - revision: fb1e332409 [main]
executor >  local (4)
[18/c81a13] process > gzipToArchive (1) [100%] 1 of 1 ✓
[ad/cd627d] process > grepErrors (1)    [100%] 1 of 1 ✓
[b3/4b8e9b] process > gzipToErrors (1)  [100%] 1 of 1 ✓
[c9/fb5591] process > errorReport (1)   [100%] 1 of 1 ✓

tmp
$
```

```
tmp
$ ls -R logs/
foo.log
tmp
$ nextflow run heuermh/minnebar15-workflow --logs logs/
N E X T F L O W  ~  version 20.07.1
Launching `heuermh/minnebar15-workflow` [lethal_ritchie] - revision: fb1e332409 [main]
executor >  local (4)
[18/c81a13] process > gzipToArchive (1) [100%] 1 of 1 ✓
[ad/cd627d] process > grepErrors (1)    [100%] 1 of 1 ✓
[b3/4b8e9b] process > gzipToErrors (1)  [100%] 1 of 1 ✓
[c9/fb5591] process > errorReport (1)   [100%] 1 of 1 ✓

tmp
$ ls archive/
foo.log.gz
tmp
$ ls errors/
foo.log.errors.gz
tmp
$
```

```
tmp
$ ls -R logs/
foo.log
tmp
$
```

```
tmp
$ ls -R logs/
foo.log
tmp
$ nextflow run heuermh/minnebar15-workflow --logs logs/ -with-docker ubuntu:18.04
N E X T F L O W  ~  version 20.07.1
Launching `heuermh/minnebar15-workflow` [maniac_woese] - revision: fb1e332409 [main]
executor >  local (4)
[3a/fab030] process > gzipToArchive (1) [100%] 1 of 1 ✓
[36/a4feb1] process > grepErrors (1)    [100%] 1 of 1 ✓
[7d/a11324] process > gzipToErrors (1)  [100%] 1 of 1 ✓
[0d/a31f02] process > errorReport (1)   [100%] 1 of 1 ✓

tmp
$
```

```
tmp
$ ls -R logs/
foo.log
tmp
$ nextflow run heuermh/minnebar15-workflow --logs logs/ -with-docker ubuntu:18.04
N E X T F L O W  ~  version 20.07.1
Launching `heuermh/minnebar15-workflow` [maniac_woese] - revision: fb1e332409 [main]
executor >  local (4)
[3a/fab030] process > gzipToArchive (1) [100%] 1 of 1 ✓
[36/a4feb1] process > grepErrors (1)    [100%] 1 of 1 ✓
[7d/a11324] process > gzipToErrors (1)  [100%] 1 of 1 ✓
[0d/a31f02] process > errorReport (1)   [100%] 1 of 1 ✓

tmp
$ ls archive/
foo.log.gz
tmp
$ ls errors/
foo.log.errors.gz
tmp
$
```

Repeatable, reproducible, and scalable

- Repeatable - If you had to do it again, on the same hardware and with the same data, could you?
- Reproducible - Could a colleague or collaborator reproduce your results, on their own hardware and with their own data?
- Scalable - If it works on one sample or a small cohort, will it scale up to handle many?

Executors

In the Nextflow framework architecture, the *executor* is the component that determines the system where a pipeline process is run and supervises its execution.

The *executor* provides an abstraction between the pipeline processes and the underlying execution system. This allows you to write the pipeline functional logic independently from the actual processing platform.

In other words you can write your pipeline script once and have it running on your computer, a cluster resource manager or the cloud by simply changing the executor definition in the Nextflow configuration file.

Local

The *local* executor is used by default. It runs the pipeline processes in the computer where Nextflow is launched. The processes are parallelised by spawning multiple *threads* and by taking advantage of multi-cores architecture provided by the CPU.

In a common usage scenario, the *local* executor can be useful to develop and test your pipeline script in your computer, switching to a cluster facility when you need to run it on production data.

SGE

SLURM

The *SLURM* executor allows you to run your pipeline script by using the SLURM resource manager.

Nextflow manages each process as a separate job that is submitted to the cluster by using the `sbatch` command.

Being so, the pipeline must be launched from a node where the `sbatch` command is available, that is, in a common usage scenario, the cluster *head* node.

To enable the SLURM executor simply set to `process.executor` property to `slurm` value in the `nextflow.config` file.

The amount of resources requested by each job submission is defined by the following process directives:

- cpus
- queue
- time
- memory
- clusterOptions

Note

SLURM *partitions* can be considered jobs queues. Nextflow allows to set partitions by using the above `queue` directive.

AWS Batch

Nextflow supports AWS Batch service which allows submitting jobs in the cloud without having to spin out and manage a cluster of virtual machines. AWS Batch uses Docker containers to run tasks, which makes deploying pipelines much simpler.

The pipeline processes must specify the Docker image to use by defining the `container` directive, either in the pipeline script or the `nextflow.config` file.

To enable this executor set the property `process.executor = 'awsbatch'` in the `nextflow.config` file.

The pipeline can be launched either in a local computer or a EC2 instance. The latter is suggested for heavy or long running workloads. Moreover a S3 bucket must be used as pipeline work directory.

See the AWS Batch page for further configuration details.

Google Life Sciences

Google Cloud Life Sciences is a managed computing service that allows the execution of containerized workloads in the Google Cloud Platform infrastructure.

Nextflow provides built-in support for Life Sciences API which allows the seamless deployment of a Nextflow pipeline in the cloud, offloading the process executions as pipelines (it requires Nextflow 20.01.0 or later).

nf-core/viralrecon

Assembly and intrahost/low-frequency variant calling for viral samples

- Virtual COVID-19 Biohackathon, April 5 - 11, 2020
  - https://github.com/virtual-biohackathons/covid-19-bh20
- International collaboration (UK, Germany, Scotland, Canada, Russia, Spain, Sweden, US)
- Part of the nf-core community
  - https://nf-co.re

Nextflow 70.7% Python 20.1%
R 7.3% HTML 1.5%
Dockerfile 0.4%

README.md

Introduction

nfcore/viralrecon is a bioinformatics analysis pipeline used to perform assembly and intra-host/low-frequency variant calling for viral samples. The pipeline supports short-read Illumina sequencing data from both shotgun (e.g. sequencing directly from clinical samples) and enrichment-based library preparation methods (e.g. amplicon-based: ARTIC SARS-CoV-2 enrichment protocol; or probe-capture-based).

Pipeline summary

1. Download samples via SRA, ENA or GEO ids (`ENA FTP`, `parallel-fastq-dump`; *if required*)
2. Merge re-sequenced FastQ files (`cat`; *if required*)
3. Read QC (`FastQC`)
4. Adapter trimming (`fastp`)
5. Variant calling
    i. Read alignment (`Bowtie 2`)
    ii. Sort and index alignments (`SAMtools`)
    iii. Primer sequence removal (`iVar`; *amplicon data only*)
    iv. Duplicate read marking (`picard`; *removal optional*)
    v. Alignment-level QC (`picard`, `SAMtools`)
    vi. Genome-wide and amplicon coverage QC plots (`mosdepth`)
    vii. Choice of multiple variant calling and consensus sequence generation routes (`VarScan 2`, `BCFTools`, `BEDTools` // `iVar variants and consensus` // `BCFTools`, `BEDTools`)
        - Variant annotation (`SnpEff`, `SnpSift`)
        - Consensus assessment report (`QUAST`)
    viii. Intersect variants across callers (`BCFTools`)
6. *De novo* assembly
    i. Primer trimming (`Cutadapt`; *amplicon data only*)
    ii. Removal of host reads (`Kraken 2`)
    iii. Choice of multiple assembly tools (`SPAdes` // `metaSPAdes` // `Unicycler` // `minia`)
        - Blast to reference genome (`blastn`)
        - Contiguate assembly (`ABACAS`)
        - Assembly report (`PlasmidID`)

- Variant annotation (`SnpEff` , `SnpSift`)
- Consensus assessment report (`QUAST`)

viii. Intersect variants across callers (`BCFTools`)

6. *De novo* assembly
   i. Primer trimming (`Cutadapt` ; *amplicon data only*)
   ii. Removal of host reads (`Kraken 2`)
   iii. Choice of multiple assembly tools (`SPAdes` // `metaSPAdes` // `Unicycler` // `minia`)
   - Blast to reference genome (`blastn`)
   - Contiguate assembly (`ABACAS`)
   - Assembly report (`PlasmidID`)
   - Assembly assessment report (`QUAST`)
   - Call variants relative to reference (`Minimap2` , `seqwish` , `vg` , `Bandage`)
   - Variant annotation (`SnpEff` , `SnpSift`)
7. Present QC and visualisation for raw read, alignment, assembly and variant calling results (`MultiQC`)

> **NB:** The pipeline has a number of options to allow you to run only specific aspects of the workflow if you so wish. For example, you can skip all of the assembly steps with the `--skip_assembly` parameter. See the usage docs for all of the available options when running the pipeline.

Pipeline reporting

Numerous QC and reporting steps are included in the pipeline in order to collate a full summary of the analysis within a single MultiQC report. You can see an example MultiQC report here, generated using the parameters defined in this configuration file. The pipeline was run with these samples, prepared from the ncov-2019 ARTIC Network V1 amplicon set and sequenced on the Illumina MiSeq platform in 301bp paired-end format.

Quick Start

1. Install `nextflow`
2. Install either `Docker` or `Singularity` for full pipeline reproducibility *(please only use `Conda` as a last resort; see docs)*
3. Download the pipeline and test it on a minimal dataset with a single command:

```
nextflow run nf-core/viralrecon -profile test,<docker/singularity/conda/institute>
```

> Please check nf-core/configs to see if a custom config file to run nf-core pipelines already exists for your Institute. If so, you can simply use `-profile <institute>` in your command. This will enable either `docker` or `singularity` and set the appropriate execution settings for your local compute environment.

4. Start running your own analysis!
    - Typical command for shotgun analysis:

```
nextflow run nf-core/viralrecon \
    --input samplesheet.csv \
    --genome 'MN908947.3' \
    -profile <docker/singularity/conda/institute>
```

    - Typical command for amplicon analysis:

```
nextflow run nf-core/viralrecon \
```

```
tmp
$ nextflow run nf-core/viralrecon -profile test,docker
```

```
tmp
$ nextflow run nf-core/viralrecon -profile test,docker
N E X T F L O W  ~  version 20.07.1
Pulling nf-core/viralrecon ...
downloaded from https://github.com/nf-core/viralrecon.git
Launching `nf-core/viralrecon` [drunk_franklin] - revision: 75a2f9763e [master]
------------------------------------------------------
                                          ,--.  ,-.
        ___     __   __   __   ___     /,-._.--~'
  |\ | |__  __ /  ` /  \ |__) |__         } {
  | \| |       \__, \__/ |  \ |___     \`-._,-`-,
                                          `-._,-'

  nf-core/viralrecon v1.1.0
------------------------------------------------------
Pipeline Release      : master
Run Name              : drunk_franklin
Samplesheet           : https://raw.githubusercontent.com/nf-core/test-datasets/viralrecon/samplesheet/samplesheet_test_amplicon.csv
Protocol              : amplicon
Amplicon Fasta File   : https://raw.githubusercontent.com/nf-core/test-datasets/viralrecon/genome/NC_045512.2/amplicon/nCoV-2019.artic.V1.primer.fasta
Amplicon BED File     : https://raw.githubusercontent.com/nf-core/test-datasets/viralrecon/genome/NC_045512.2/amplicon/nCoV-2019.artic.V1.bed
Amplicon Left Suffix  : _LEFT
Amplicon Right Suffix : _RIGHT
```

```
>0 LN:i:3807 KC:i:1322833 km:f:350.234
TTCACTCTTCATTTCCAAAAAGCTTGAAACAAGTTTGTCATAGAGATTTTTATCAAAGACAGCTAAGTAGACATTTGTGCGAACAGTATCTACACAAACTC
TTAAAGAATGTATAGGGTCAGCACCAAAAATACCAGCTGATAATAATGGTGCAAGTAGAACTTCGTGCTGATTAAAATTTTCATAAGCACTCTTAAGAAGT
TGAATGTCTTCACCTTTGTTAACATTTGGGCCGACAACATGAAGACAGTGTTTAGCAAGATTGTGTCCGCTTAAAACACAACTACCACCCACTTTAAGTGG
TCCATTAGTAGCTATGTAATCATCAGATTCAACTTGCATGGCATTGTTAGTAGCCTTATTTAAGGCTCCTGCAACACCTCCTCCATGTTTAAGGTAAACAT
TGGCTGCATTAACAACCACTGTTGGTTTTACCTTTTTAGCTTCTTCCACAATGTCTGCATTTTTAATGTATACATTGTCAGTAAGTTTTAAATAACCACTA
AAACTATTCACTTCAATAGTCTGAACAACTGGTGTAAGTTCCATCTCTAATTGAGGTTGAACCTCAACAATTGTTTGAATAGTAGTTGTCTGATTGTCCTC
ACTGCCGTCTTGTTGACCAACAGTTTGTTGACTATCATCATCTAACCAATCTTCTTCTTGCTCTTCTTCAGGTTGAAGAGCAGCAGAAGTGGCACCAAATT
CCAAAGGTTTACCTTGGTAATCATCTTCAGTACCATACTCATATTGAGTTGATGGCTCAAACTCTTCTTCTTCACAATCACCTTCTTCTTCATCCTCATCT
GGAGGGTAAAAAGAACAATACATATGTGAAGCCAATTTAAACTCACCAGACTCATCAAATAAGTAGTATGTAGCCATACTCCACTCATCTAAATCAATGCC
CAGTGGTGTAAGTAATTCAGATACTGGTTGCAAAGTTTTTATGACAGCATCTGCCACAACACAGGCGAACTCATTTACTTCTGTACCGAGTTCAACTGTAT
AGGCAGAGCACTTCTCATTAAGTACTTTATCAATCCTTTCATCAAGTTCAAAAGTGATATTCACACTCTTGTAACCTTGCACTTCTATCACAGTGTCATCA
CCAAAAGTAACCTTTGTTGGTGCACCGCCTTTGAGTGTGAAGGTATTGTTTGTTACCATCATATTAGGTGCAAGGGCACAGTACTTTTCTGTGTCTTTGAT
TTCGAGCAACATAAGCCCGTTAATACAAACTGGTGTACCAACCAATGGAGCTTCAACAGCTTCACTAGTAGGTTGTTCTAATGGTTGTAAATCACCAGTTT
TCAAGACAACTTCCTCTGTTAACACTTCTGTGGGAAGTGTTTCTCCCTCTAAGAAGATAATTTCTTTTGGGGCTTTTAGAGGCATGAGTAGGCCAGTTTCT
TCTCTGGATTTAACACACTTTCTGTACAATCCCTTTGAGTGCGTGACAAATGTTTCACCTAAATTCAAGGCTTTAAGTTTAGCTCCACCAATAATGATAGA
GTCAGCACACAAAGCCAAAAATTTATTTACAAGCTTAAAGAATGTCTGAACACTCTCCTTAATTTCCTTTGCACAGGTGACAATTTGTCCACCGACAATTT
CACAAGCACAGGTTGAGATAAATTTAACAATTTCCCAACCGTCTCTAAGAAACTCTACACCTTCCTTAAACTTCTCTTCAAGCCAATCAAGGACGGGTTTG
AGTTTTTCATAAACAGTGCCAAAGATGTTAGTTAGCCACTGCGAAGTCAACTGAACAACACCACCTGTAATGTAGGCCATTACAACTAGATTGTTAGTAGC
CAAATCAGATGTGAACATCATAGCATCAATGAGTCTCAGTGAATACTGTGAAATTCCATCTAGTATTGTTATAGCGGCCTTCTGTAAAACACGCACAGAAT
TTTGAGCAGTTTCAAGAGTGCGGGAGAAAATTGATCGTACAACACGAGCAGCCTCTGATGCAAATGCATAAAGAGGACTCAGTATTGATTTCTGTTCACCA
ATATTCCAGGCACCTTTTTTAGCTTTTCCTTTTGTAACTTTAAAATTACCACAGGATTCAACAATTTGTTTGAATGCTTTATAATCCAAACCTTTCACAGT
TTCCACAAAAGCACTTGTGGAAGCAGAAAAAGATGCCAAAATAATGGCGATCTCTTCATTAAGTTTAAAGTCACCAACAATATTGATGTTGACTTTCTCTT
TTTGGAGTATTTCAAGAAGGTTGTCATTAAGACCTTCGGAACCTTCTCCAACAACACCTGTATGGTTACAACCTATGTTAGCGCTAGCACGTGGAACCCAA
results/assembly/minia/31/SAMPLE1_PE.k31.scaffolds.fa
```

Resources

- Nextflow
  - https://www.nextflow.io
- Example workflow
  - https://github.com/heuermh/minnebar15-workflow
- nf-core community
  - https://nf-co.re
- nf-core/viralrecon workflow
  - https://github.com/nf-core/viralrecon

Resources

- Nextflow
  - https://www.nextflow.io
- Example workflow
  - https://github.com/heuermh/minnebar15-workflow
- nf-core community
  - https://nf-co.re
- nf-core/viralrecon workflow
  - https://github.com/nf-core/viralrecon

Thank you!