Virtual

Foundations in Genomic Analyses

Getting Started with nf-core Nextflow Pipelines

Northwestern | INFORMATION TECHNOLOGY
RESEARCH COMPUTING AND DATA SERVICES

Setup...

1. slides are at
   https://github.com/nuitrcs/nextflow_nfcore_intro
2. log onto Quest via terminal (or Quest OnDemand)

   ```
   ssh <netid>@quest.northwestern.edu # enter your netid
   password
   ```
3. move to our classroom folder

   ```
   cd /projects/e32680
   ```
4. make your own subfolder if you haven't already

   ```
   mkdir <folder_name>
   ```

Nextflow: a DSL for parallel and scalable computational pipelines

- DSL = domain specific language
- Parallel = tasks that are not dependent can run at the same time
- Scalable = can be run on one or many samples with the same format

Nextflow is a programming language, specifically for building computational pipelines. If you'd like to build your own pipelines, you will have to learn their syntax. But, there are many prebuilt pipelines!

The benefits of Nextflow pipelines are their **scalability**, and their **reproducibility**.

Pipelines are series of tasks with dependent inputs and outputs

Raw sequencing data

- ↳ Quality control
- ↳ Trim adaptors
- ↳ Remove contaminants
- ↳ Align to reference genome
- ↳ Quantification

Pipelines are series of tasks with dependent inputs and outputs

Raw sequencing data

- ↳ Quality control
- ↳ Trim adaptors
- ↳ Remove contaminants
- ↳ Align to reference genome
- ↳ Quantification

- You could work through this step by step, sample by sample.
- You could work through each step with all samples with a job array.
- You could work through all steps with all samples with dependent job arrays.

Pipelines are series of tasks with dependent inputs and outputs

Raw sequencing data

- ⮡ Quality control
- ⮡ Trim adaptors
- ⮡ Remove contaminants
- ⮡ Align to reference genome
- ⮡ Quantification

So why nextflow?

- some pipelines are pre-developed, less time troubleshooting your own scripts
- containerized software for reproducibility and portability
- many people using the same pipelines and working on their development

Pipelines are series of tasks with dependent inputs and outputs

Raw sequencing data

- Quality control
- Trim adaptors
- Remove contaminants
- Align to reference genome
- Quantification

But there are cons!!

- blackbox - little understanding of what’s happening under the hood
- hard to troubleshoot when you do encounter errors
- working in a high-throughput might not notice issues with individual samples
- can remove tailoring parameters to best fit your own particular data

Defining a workflow

- Nextflow refers to each of these steps that can be converted into a command as a “process”

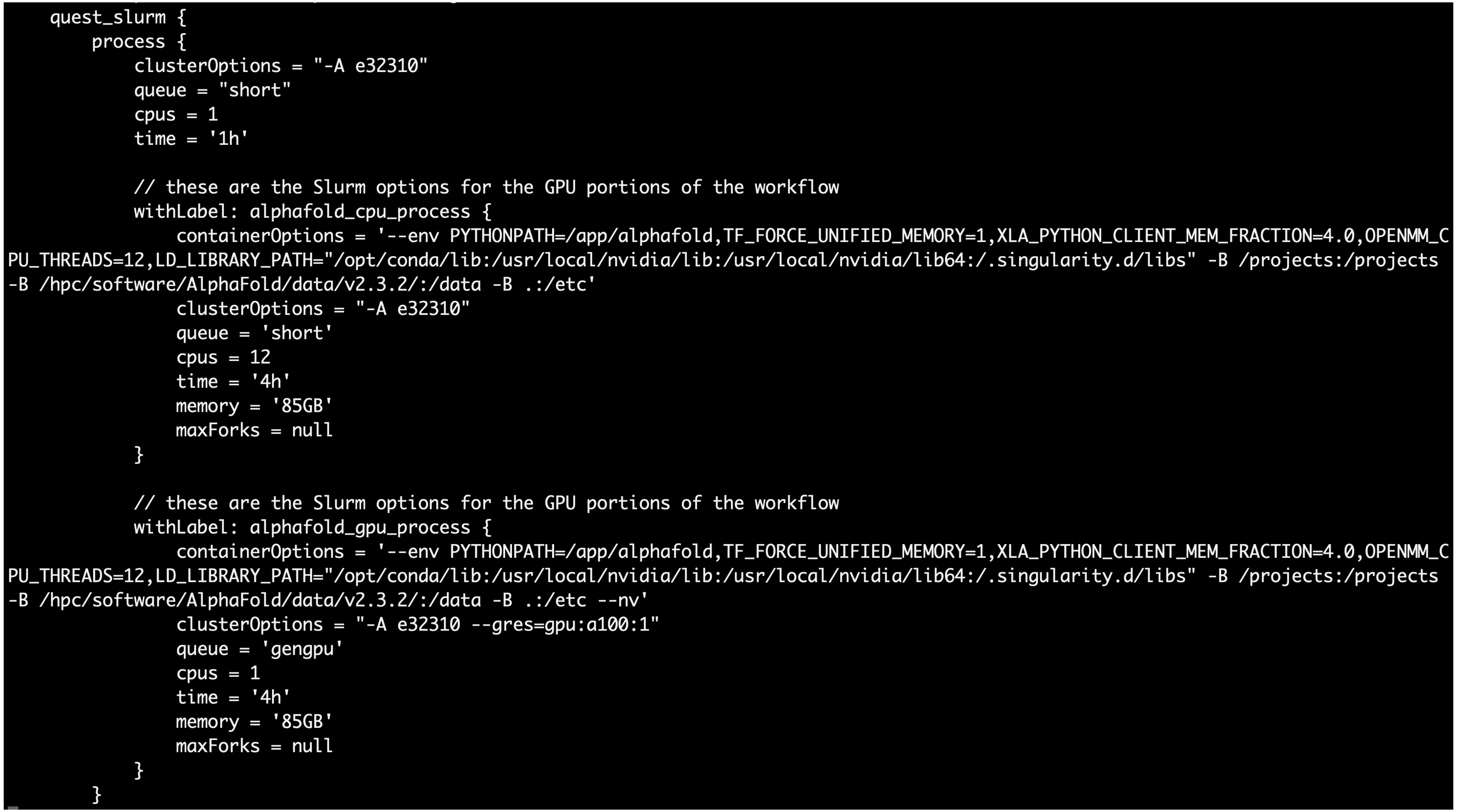
```
process blast {
    input:
    path ch_fasta
    val db_name_in
    path db_dir_in

    output:
    path 'blast_result'

    publishDir params.out, mode:'copy', overwrite: true

    """
    blastp -db $db_dir_in/$db_name_in -query $ch_fasta -outfmt 6 > blast_result
    """
}
```

Defining a workflow: `main.nf`

- Nextflow refers to each of these steps that can be converted into a command as a “process”
- the **main.nf** file defines all the **processes**, and the **workflow**
- workflow defines the series of processes by using processes names as functions and inputs

```
workflow {

    /*
     * Create a channel emitting the given query fasta file(s).
     * Split the file into chunks containing as many sequences as defined by the parameter 'chunkSize'.
     * Finally, assign the resulting channel to the variable 'ch_fasta'
     */
    Channel
        .fromPath(params.query)
        .splitFasta(by: params.chunkSize, file:true)
        .set { ch_fasta }

    db_name = file(params.db).name
    db_dir = file(params.db).parent

    /*
     * Execute a BLAST job for each chunk emitted by the 'ch_fasta' channel
     * and emit the resulting BLAST matches.
     */
    blast_data = blast(ch_fasta, db_name, db_dir)
    ch_hits = top_hits(blast_data)

    /*
     * Each time a file emitted by the 'blast' process, an extract job is executed,
     * producing a file containing the matching sequences.
     */
    ch_sequences = extract(ch_hits, db_name, db_dir)

    /*
     * Collect all the sequences files into a single file
     * and print the resulting file contents when complete.
     */

    ch_sequences
        .collectFile(name: params.out)
        .view { file -> "matching sequences:\n ${file.text}" }

    /*
    *  ch_sequences contains multiple protein hits. However, we need
    *  feed a single protein per alphafold job.
    */

    ch_sequences
        .splitFasta( by: 1 , file: true)
        .set { ch_alphafold_in }

    /*
    *  Create an output variable which is the output directory of the
    *  alphafold_cpu job so that we can link it to the alphafold_gpu job.
    */

    alphafold_cpu(ch_alphafold_in) | alphafold_gpu

}
```

Assigning Compute Resources: `nextflow.config`



```
    quest_slurm {
        process {
            clusterOptions = "-A e32310"
            queue = "short"
            cpus = 1
            time = '1h'

            // these are the Slurm options for the GPU portions of the workflow
            withLabel: alphafold_cpu_process {
                containerOptions = '--env PYTHONPATH=/app/alphafold,TF_FORCE_UNIFIED_MEMORY=1,XLA_PYTHON_CLIENT_MEM_FRACTION=4.0,OPENMM_C
PU_THREADS=12,LD_LIBRARY_PATH="/opt/conda/lib:/usr/local/nvidia/lib:/usr/local/nvidia/lib64:/.singularity.d/libs" -B /projects:/projects
-B /hpc/software/AlphaFold/data/v2.3.2/:/data -B .:/etc'
                clusterOptions = "-A e32310"
                queue = 'short'
                cpus = 12
                time = '4h'
                memory = '85GB'
                maxForks = null
            }

            // these are the Slurm options for the GPU portions of the workflow
            withLabel: alphafold_gpu_process {
                containerOptions = '--env PYTHONPATH=/app/alphafold,TF_FORCE_UNIFIED_MEMORY=1,XLA_PYTHON_CLIENT_MEM_FRACTION=4.0,OPENMM_C
PU_THREADS=12,LD_LIBRARY_PATH="/opt/conda/lib:/usr/local/nvidia/lib:/usr/local/nvidia/lib64:/.singularity.d/libs" -B /projects:/projects
-B /hpc/software/AlphaFold/data/v2.3.2/:/data -B .:/etc --nv'
                clusterOptions = "-A e32310 --gres=gpu:a100:1"
                queue = 'gengpu'
                cpus = 1
                time = '4h'
                memory = '85GB'
                maxForks = null
            }
        }
```

Exercise: Define Slurm account

- Make a personal copy of a nextflow pipeline

```
    module load git/2.37.2

    git clone git@github.com:nuitrcs/nextflow-workshop.git
```

- Navigate to nextflow.config for the blast pipeline

```
  cd nextflow-workshop/blast-example

  nano nextflow.config
```

- Edit nextflow.config to use this classroom allocation e32680
  - Hint: It’s defined in 3 places.

Exercise: Lauch the pipeline

- Load the needed modules

```
module load blast/2.9.0

module load nextflow/23.04.3

module load singularityce/4.3.1-gcc-8.5.0
```

- Run nextflow

```
nextflow run -profile quest_slurm -resume .
```

What is it doing?

- Running each process defined in our workflow as a job submitted through Slurm
- The nextflow command needs to continue to run for as long as the pipeline needs to submit jobs
- Creating and working with in ./work which will contain all intermediate files
- Writing log output to .nextflow.log
  - each time you run this workflow if will create a new file and append a number to the old one
- Writing process updates to the consult and checking off complete jobs

nf-core: A global community effort to collect a curated set of open-source analysis pipelines built using Nextflow.

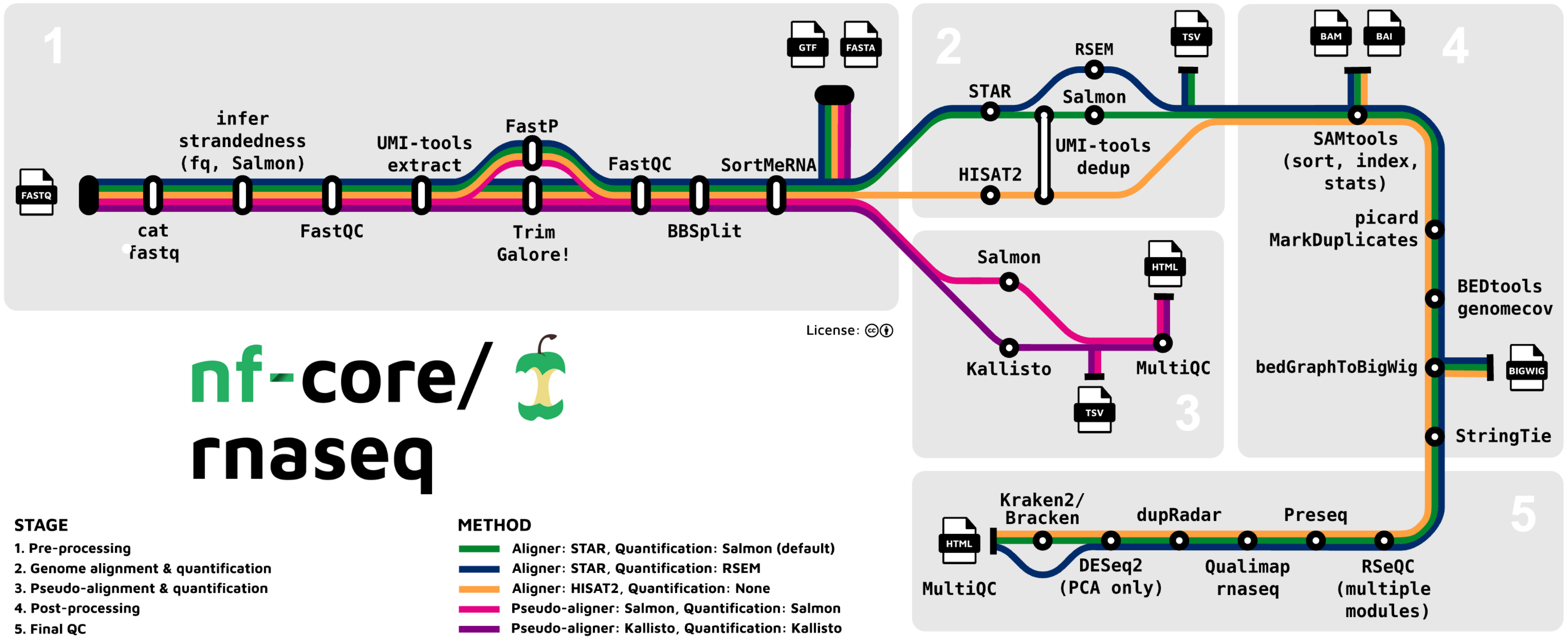
- https://nf-co.re/ - Website with documentation, pipelines, other information.
- https://nf-co.re/docs/ - Direct link to documentation pages
- https://www.youtube.com/@nf-core - Youtube videos provided by nf-core

Currently 84 released pipelines, 45 under development, and 12 archived.

ampliseq, detaxizer, crisprseq, demultiplex, mag, methylseq, rnaseq, demo, smrnaseq, pairgenomealign, chipseq, scnanoseq, multiplesequencealign, taxprofiler, raredisease, fastquorum, isoseq, funcscan, sarek, nanostring, oncanalyser, scrnaseq, denovotranscript, pixelator, proteinfold, reportho, eager, bacass, mhcquant, airrflow, callingcards, epitopeprediction, rnasplice, differentialabundance, bamtofastq

https://nf-co.re/pipelines/

https://nf-co.re/rnaseq/3.17.0/



STAGE
1. Pre-processing
2. Genome alignment & quantification
3. Pseudo-alignment & quantification
4. Post-processing
5. Final QC

METHOD
- 🟩 Aligner: STAR, Quantification: Salmon (default)
- 🟦 Aligner: STAR, Quantification: RSEM
- 🟧 Aligner: HISAT2, Quantification: None
- 🟥 Pseudo-aligner: Salmon, Quantification: Salmon
- 🟪 Pseudo-aligner: Kallisto, Quantification: Kallisto

Nextflow command

```
cat /projects/e32559/nextflow_example/script.sh


nextflow run nf-core/rnaseq/3.17.0 \
  --input /projects/e32680/07_nextflow_intro/samples.csv \
  --outdir /projects/e32680/<folder>/nextflow_output \
  --gtf /projects/e32680/05_rnaseq_alignment/braker.gtf \
  --fasta /projects/e32680/05_rnaseq_alignment/oh.polished.fasta \
  -profile nu_genomics \
  -with-report \
  -with-trace
```

Nextflow command

```
cat /projects/e32559/nextflow_example/script.sh
```

```
nextflow run nf-core/rnaseq/3.17.0 \
  --input /projects/e32680/07_nextflow_intro/samples.csv \
  --outdir /projects/e32680/<folder>/nextflow_output \
  --gtf /projects/e32680/05_rnaseq_alignment/braker.gtf \
  --fasta /projects/e32680/05_rnaseq_alignment/oh.polished.fasta \
  -profile nu_genomics \
  -with-report \
  -with-trace
```

Let’s run it!

Copy this script to your folder and open it for editing:

```
cp /projects/e32559/nextflow_example/script.sh .
```

```
nano script.sh
```

Change the --outdir to your folder, and replace the -profile with the line below if you are not a member of b1042:

```
-c /projects/e32680/07_nextflow_intro/nf-core.config
```

Save and close nano:

Ctrl+O, enter, Ctrl+X

Lauch the script!

```
sbatch script.sh
```

Look at a completed run

sacct -X -S 111924 # this will show all jobs from November 19th, 2024

```
7133232        script.sh     normal     e32559          1  COMPLETED      0:0
7133237       nf-NFCORE+     normal     e32559          1  COMPLETED      0:0
7133238       nf-NFCORE+     normal     e32559          1  COMPLETED      0:0
7133266       nf-NFCORE+     normal     e32559          2  COMPLETED      0:0
7133267       nf-NFCORE+     normal     e32559         12  COMPLETED      0:0
7133274       nf-NFCORE+     normal     e32559          6  COMPLETED      0:0
7133275       nf-NFCORE+     normal     e32559         12  COMPLETED      0:0
7133283       nf-NFCORE+     normal     e32559         12  COMPLETED      0:0
7133716       nf-NFCORE+     normal     e32559          1  COMPLETED      0:0
7133721       nf-NFCORE+     normal     e32559          6  COMPLETED      0:0
7133875       nf-NFCORE+     normal     e32559          6  COMPLETED      0:0
7133880       nf-NFCORE+     normal     e32559         12  COMPLETED      0:0
7134482       nf-NFCORE+     normal     e32559          6  COMPLETED      0:0
7134483       nf-NFCORE+     normal     e32559          6  COMPLETED      0:0
7134490       nf-NFCORE+     normal     e32559          2  COMPLETED      0:0
7134491       nf-NFCORE+     normal     e32559          2  COMPLETED      0:0
7134509       nf-NFCORE+     normal     e32559          1  COMPLETED      0:0
7134510       nf-NFCORE+     normal     e32559          1  COMPLETED      0:0
7134511       nf-NFCORE+     normal     e32559          1  COMPLETED      0:0
7134553       nf-NFCORE+     normal     e32559          6  COMPLETED      0:0
7134554       nf-NFCORE+     normal     e32559          6  COMPLETED      0:0
7134577       nf-NFCORE+     normal     e32559          6  COMPLETED      0:0
7134636       nf-NFCORE+     normal     e32559          6  COMPLETED      0:0
7134679       nf-NFCORE+     normal     e32559          6  COMPLETED      0:0
7134680       nf-NFCORE+     normal     e32559          6  COMPLETED      0:0
```

Parent job

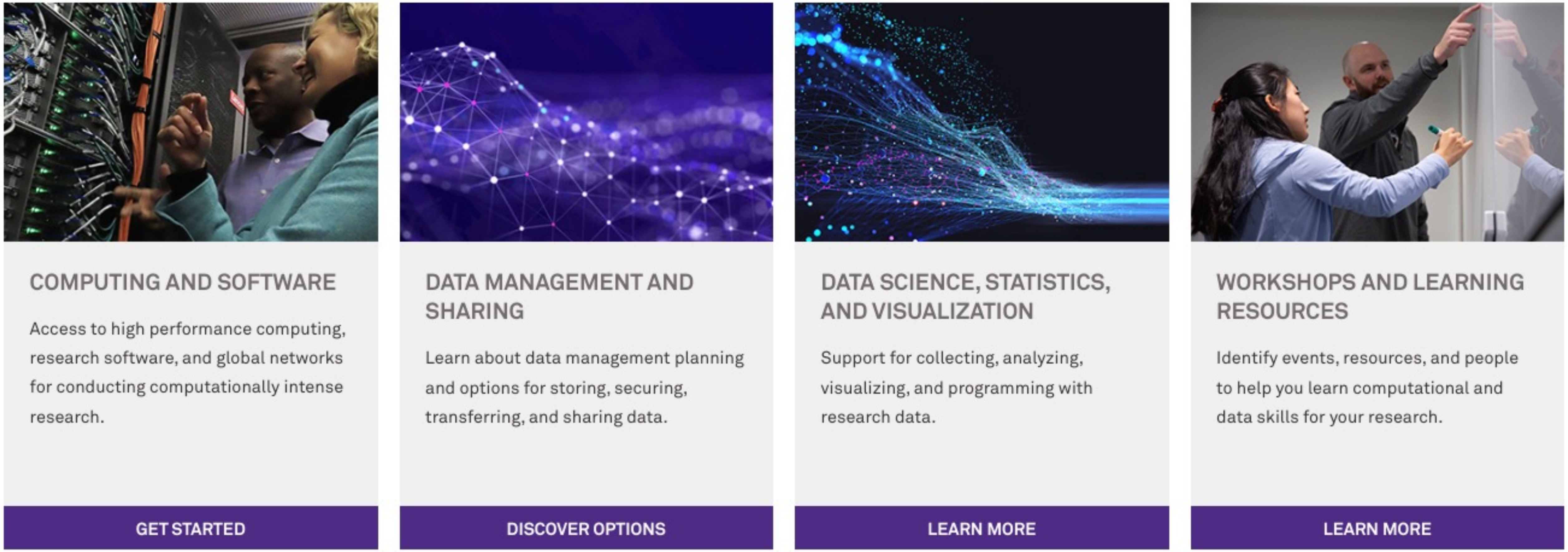
Subsequent jobs launched by nextflow pipeline

Pipeline output - what to expect when it works

- **nextflow_output**
  - **fastqc** - QC reports for samples
  - **multiqc** - QC reports for processed samples
  - **star_salmon** - bulk of output from alignment and annotation steps
  - **trimgalore** - output of sample processing with Trimgalore, and QC after trimming

Output and log files - to help troubleshoot

- nextflow will create the following in your working directory
  - **.nextflow** hidden folder (ls -a needed to see this)
  - **.nextflow.log** hidden log file (ls -a needed to see this)
  - **work** folder
  - **nextflow_output** folder (which we named in the submission script)
    - because we added --with-report and --with-trace this will include a **pipeline** folder with
      - execution_report_2024-11-19_09-53-12.html
      - execution_timeline_2024-11-19_09-53-12.html
      - execution_trace_2024-11-19_09-53-12.txt
- slurm will create the following in your working directory
  - **slurm-<jobID>.out** log file from slurm



COMPUTING AND SOFTWARE

Access to high performance computing, research software, and global networks for conducting computationally intense research.

GET STARTED

DATA MANAGEMENT AND SHARING

Learn about data management planning and options for storing, securing, transferring, and sharing data.

DISCOVER OPTIONS

DATA SCIENCE, STATISTICS, AND VISUALIZATION

Support for collecting, analyzing, visualizing, and programming with research data.

LEARN MORE

WORKSHOPS AND LEARNING RESOURCES

Identify events, resources, and people to help you learn computational and data skills for your research.

LEARN MORE

Questions?

Open a ticket with us - quest-help@northwestern.edu

Reach me in particular - haley.carter@northwestern.edu

Visit our website - https://www.it.northwestern.edu/departments/it-services-support/research/

Our team does consultations as well as workshops, let us know if you'd like to chat about your research, troubleshoot something together, or even have us come to a lab meeting!