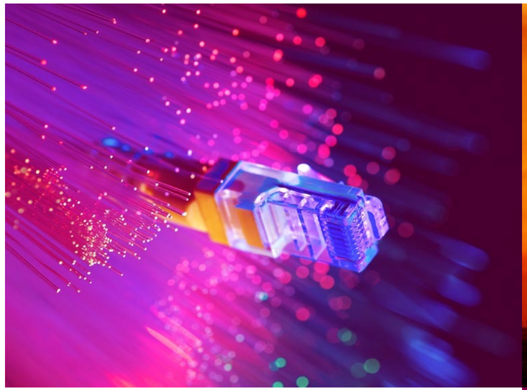


Nextflow with Singularity Containers

CSC – Suomalainen tutkimuksen, koulutuksen, kulttuurin ja julkishallinnon ICT-osaamiskeskus

Outline

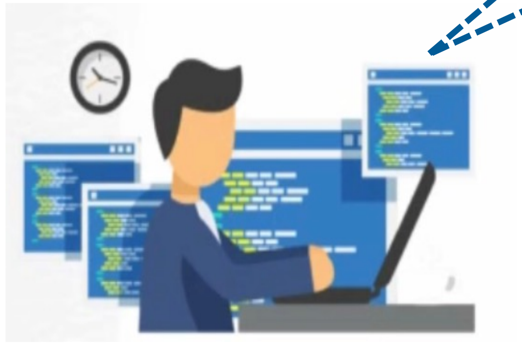
- Why containers in workflows?
- Configuring singularity with Nextflow
- Running Nextflow workflows on Puhti
- Reporting and visualisation of workflows
- Working with *nf-core* workflows

Why Containers in Workflows?

Why Containers in Workflows?

Developer

Workflow works fine for me



Tester

No. It does NOT work here

Nextflow workflows with Containers

- Built-in integration with containers
- Advantages
  - Maintainability
  - Portability
  - Reproducibility
- Popular container choices
  - Docker
  - Singularity
  - (conda)

Which software to choose for Nextflow ?

nextflow +

Docker

OR

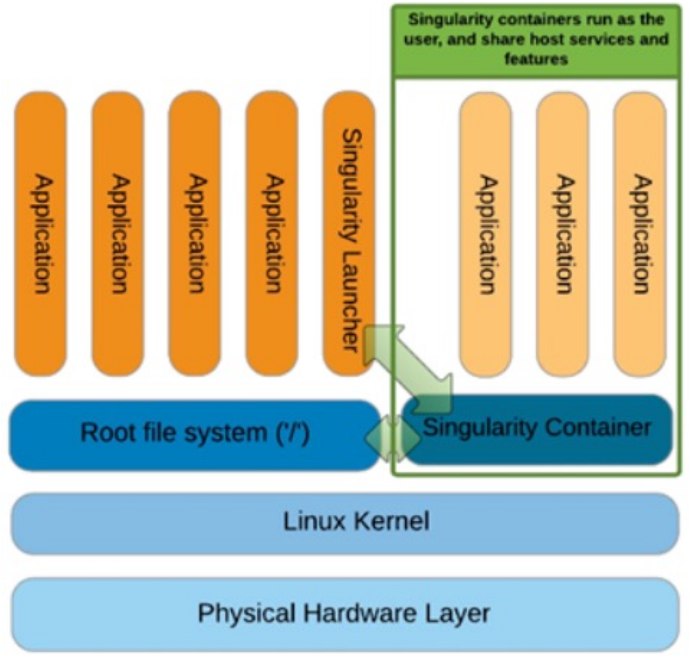
Conda

OR

Singularity/Apptainer

Singularity Containers

- No dependency of a daemon
- Can be run as a simple user
  - Avoid permission headaches and hacks
- More easily portable
- Image/container is a file (or directory)

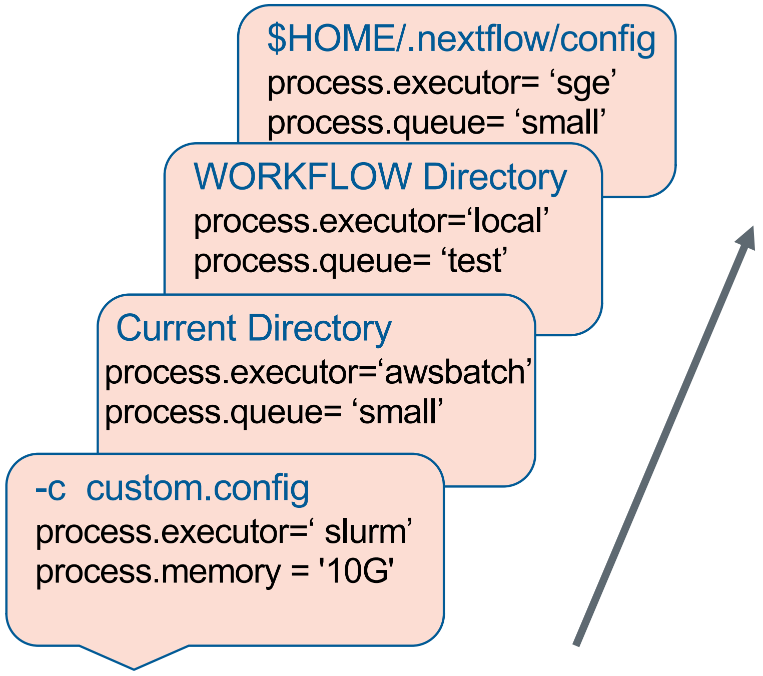


HPC Container
Singularity

Configuring Singularity with Nextflow

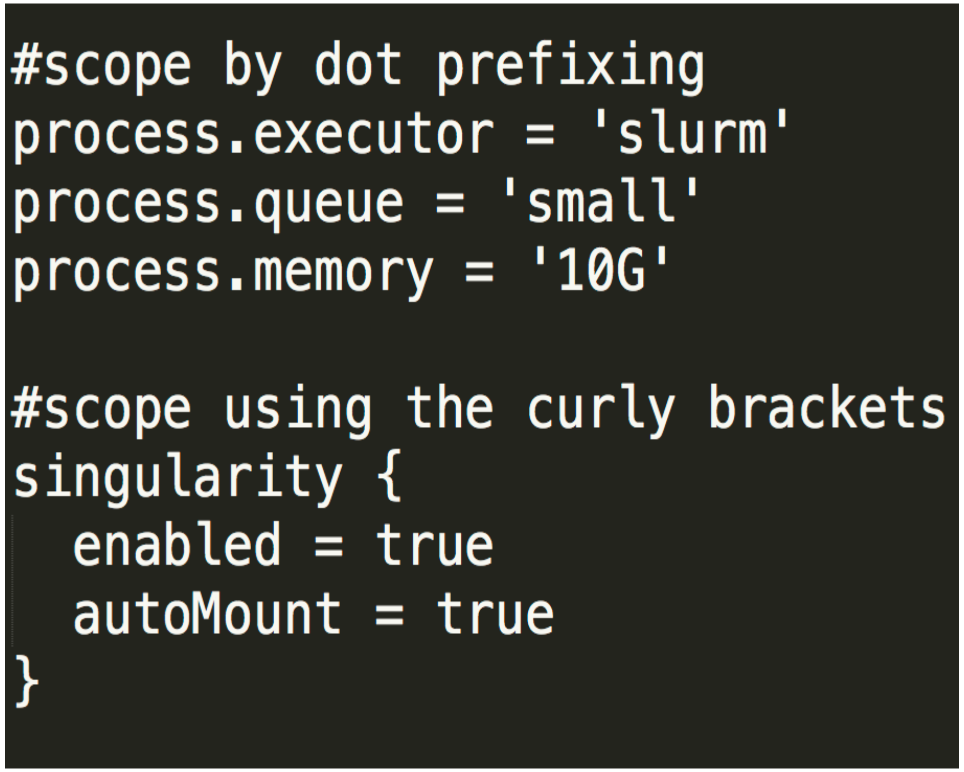
Nextflow Configuration File(s)

- Nextflow can load pipeline configurations from multiple locations:
  - Home directory
    - $HOME/.nextflow/config
  - workflow project directory
    - ~/.nextflow/assets/nextflow-io/rnaseq/nextflow.config
  - current directory
    - $PWD/nextflow.config
  - config file is given with -c <config file>
- Understand the overriding behaviour
  - process.executor='slurm'
  - process.queue= 'small'
  - process.memory='10G'



Configuration Files: scopes

- Configuration settings can be organized in different scopes
- Nextflow scopes
  - *- env*
  - *- params*
  - *- process*
  - *- and many other*

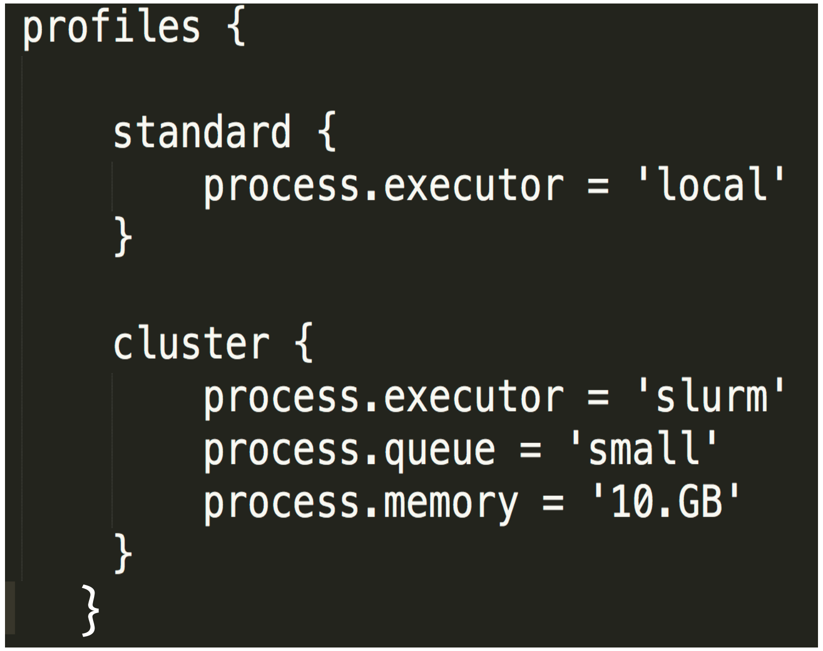


```
#scope by dot prefixing
process.executor = 'slurm'
process.queue = 'small'
process.memory = '10G'

#scope using the curly brackets
singularity {
  enabled = true
  autoMount = true
}
```

Configuration Files: profiles

- A profile is a set of configuration attributes that can be activated when launching a pipeline execution
- Configuration files can contain the definition of one or more profiles.
- Use *-profile* flag to activate attributes *via* command line



```
profiles {

    standard {
        process.executor = 'local'
    }

    cluster {
        process.executor = 'slurm'
        process.queue = 'small'
        process.memory = '10.GB'
    }
   }
```

Configuring Singularity with Nextflow

- Command line interface option :   -with-singularity

- In ‘nextflow.config’ file as a *profile* option:

```
singularity {
process.container = 'quay.io/nextflow/rnaseq-nf:v1.1’
singularity.enabled = true
singularity.autoMounts = true
}
```

Command: nextflow run main.nf -profile singularity

Running Nextflow workflows on Puhti

Puhti Recipe for Running Nextflow Pipeline

- ✓ Prepare your singularity images if needed
- ✓ Load Nextflow environment on Puhti
- ✓ Set-up your Nextflow pipeline dependencies
- ✓ Prepare batch job for Nextflow pipeline

Preparing Singularity Images if Needed

- Pull a Singularity image from a singularity registry
  - Use Puhti (e.g., singularity pull shub://vsoch/hello-world)
- Convert a Docker image to Singularity one
  - Puhti can work (e.g, singularity pull docker://tensorflow/tensorflow:latest)
- Buid a Singularity image from scratch
  - Puhti can’t be used

Load Nextflow Environment on Puhti

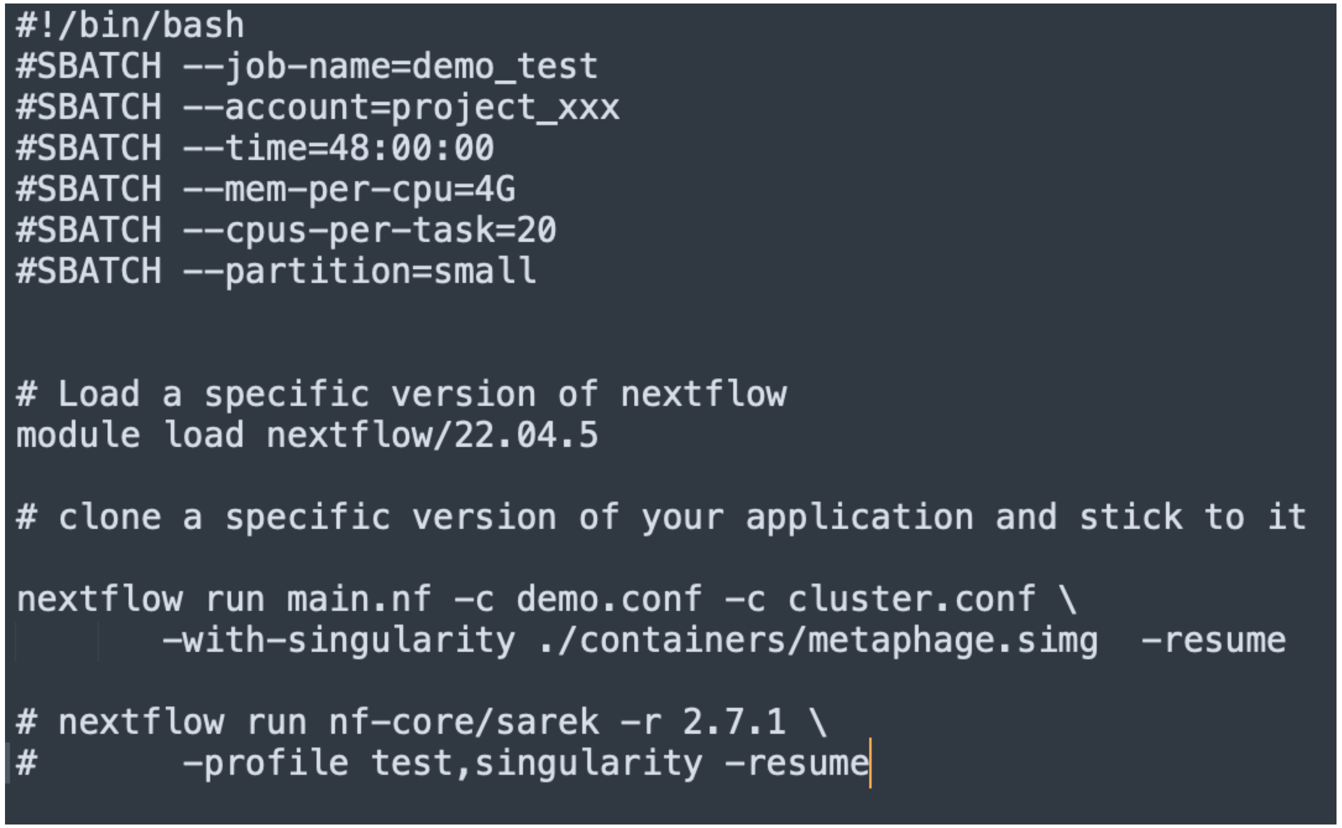
- Puhti uses module system (lmod) to manage software stack
- Nexflow is installed a module
  - module load nextflow
- Own installation :
  - wget -qO- https://get.nextflow.io | bash && mv nextflow ~/bin/
- One temporarily switch to specific version :
  - NXF_VER=20.04.0 nextflow run hello …

Prepare Your Application Dependencies

- Databases
- Move Singularity images to correct path
- Actual files/samples

Run Nextflow as a Batch Job

- Submit nextflow pipeline as a batch job
- Run all the computations in allocated job
- Avoid using ‘slurm’ executor from nextflow
- For hight-throughput jobs use ‘Hyperqueue’ executor



```
#!/bin/bash
#SBATCH --job-name=demo_test
#SBATCH --account=project_xxx
#SBATCH --time=48:00:00
#SBATCH --mem-per-cpu=4G
#SBATCH --cpus-per-task=20
#SBATCH --partition=small


# Load a specific version of nextflow
module load nextflow/22.04.5

# clone a specific version of your application and stick to it

nextflow run main.nf -c demo.conf -c cluster.conf \
       -with-singularity ./containers/metaphage.simg  -resume

# nextflow run nf-core/sarek -r 2.7.1 \
#       -profile test,singularity -resume
```

Reporting and Visualisation of workflows

Reporting and Visualisation of pipeline

Useful optional flags for creating reports and visualisation

```
-with-dag
-with-timeline
-with-report
```

- Execution report

```
nextflow run <nextflow_script> -with-report <file-name>.html
```

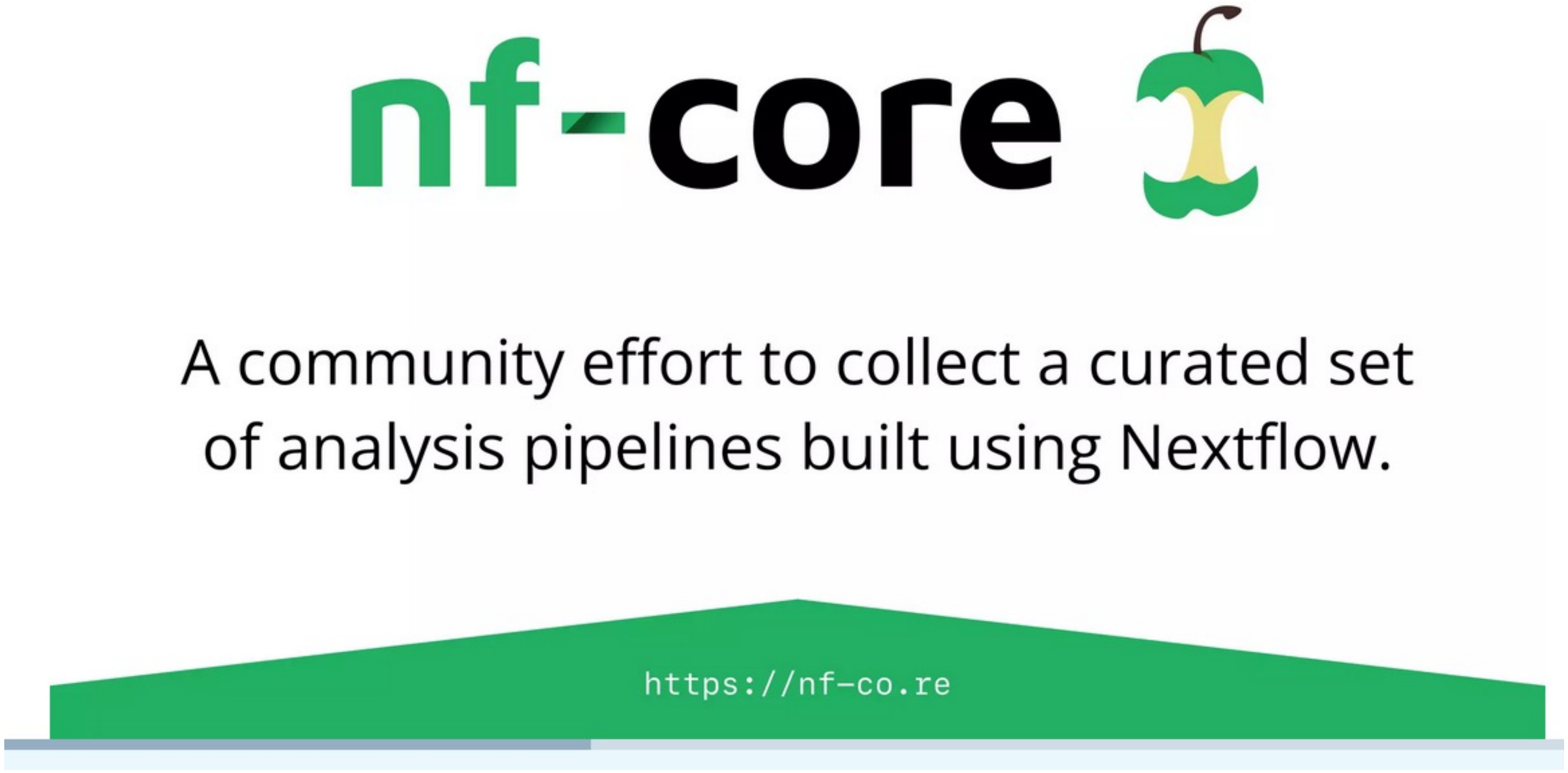
- DAG visualisation

```
nextflow run  <nextflow_script>  -with-dag <file-name>.dot
```

- Timeline report:

```
nextflow run <nextflow_script> -with-timeline <file-name>.html
```

Working with *nf-core* workflows



nf-core

A community effort to collect a curated set of analysis pipelines built using Nextflow.

https://nf-co.re

nf-core Workflows

- A nice resource for reproducible bioinformatics pipelines
- Provides common pipeline structure and usage
  - Current status: released (49); development (19)
- Each pipeline has its own documentation
  - e.g., `nextflow run nf-core/rnaseq -r 3.0 --help`
- Join on slack/twitter for help

How to Use *nf-core* Workflows ?

- One can use *nextflow* command to fetch a nf-core repository during runtime
  - No need for explicit cloning a repository in advance
  - Nextflow clones nf-core repository to $HOME directory (~/.nextflow/cache/assets/nf-core…)

- Syntax:

```
nextflow pull nf-core/<pipeline>  –r <revision>
nextflow run nf-core/<pipeline>  –r <revision>
```

Running nf-core Workflows at CSC (1/2)

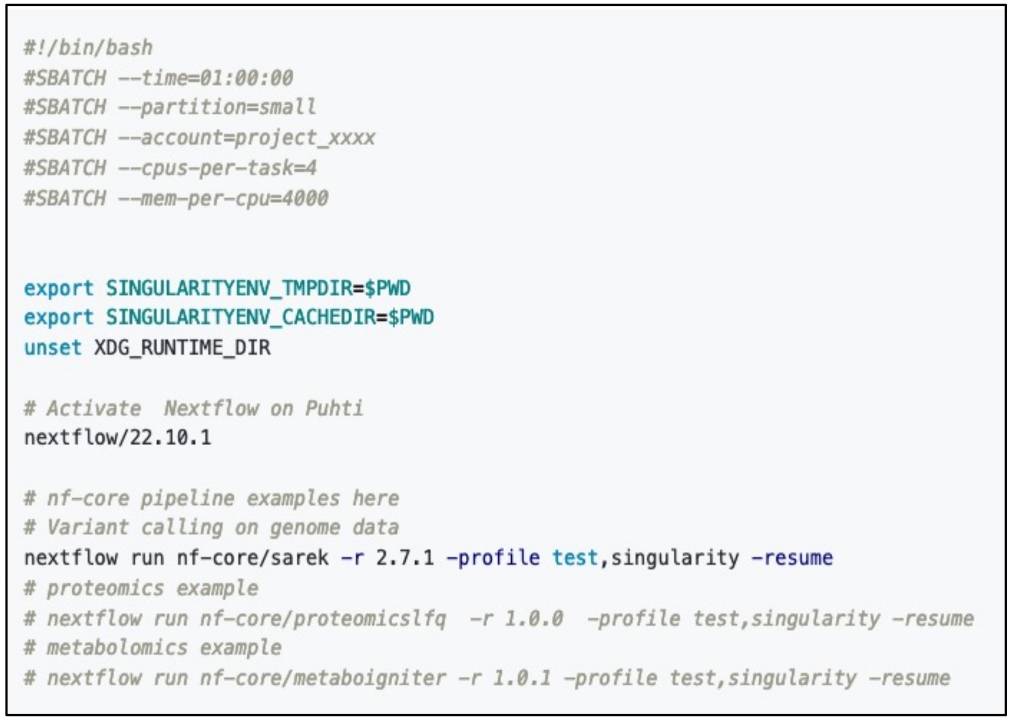
- You don’t need to install any nf-core tools to run nf-core pipelines at CSC
    - Just loading nextflow module is enough
- Use singularity as container engines
    - Conda : csc discourages this approach
    - Docker: no root access for users
- Use test profile to see if pipeline works

Running nf-core Workflows at CSC (2/2)

- Change resources (e.g., CPUs, Memory) as needed in production runs

- Containers building can fail in initial attempts

- Explore more by cloning pipeline locally



```
#!/bin/bash
#SBATCH --time=01:00:00
#SBATCH --partition=small
#SBATCH --account=project_xxxx
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=4000


export SINGULARITYENV_TMPDIR=$PWD
export SINGULARITYENV_CACHEDIR=$PWD
unset XDG_RUNTIME_DIR

# Activate  Nextflow on Puhti
nextflow/22.10.1

# nf-core pipeline examples here
# Variant calling on genome data
nextflow run nf-core/sarek -r 2.7.1 -profile test,singularity -resume
# proteomics example
# nextflow run nf-core/proteomicslfq  -r 1.0.0  -profile test,singularity -resume
# metabolomics example
# nextflow run nf-core/metaboigniter -r 1.0.1 -profile test,singularity -resume
```

An example batch script for running nf-core pipeline

Debugging nf-core Workflows

- Set Singularity/Apptainer cache/tmp directory explicitly to avoid disk space problems
  - By default, singularity/apptainer cache is in $HOME directory
- Copying sinularity images from galaxyproject can sometimes cause issues
  - Try pulling image from the same host system where you are deploying nf-core pipelines

Debugging nf-core Workflows

- Set $TEMPDIR explicity to a folder that is writable as sometimes tempdir
  - Nextflow may not mount it; you need to mount
- Some software have hardcoded /tmp paths; these can’t be reset by env variable. Use the following trick
  - --bind writable dir :/tmp

Running Workflows at Scale

- Avoid unnecessary reads and writes of data on Lustre file system to improve I/O performance
  - If unavoidable, use fast local NVMe disk, not Lustre (*i.e.* /scratch)
- Don’t run too many/short *job steps* – they will bloat Slurm accounting DB
  - Avoid *slurm* scheduler
- Don’t run too long jobs without a restarting option. Increased risk of something going wrong, resulting in lost time/results
  - Use -resume flag

.

Running Workflows at Scale

- Don’t use Conda installations on Lustre (/projappl, /scratch, $HOME)
  - Containerize Conda environments instead to improve performance
- Don’t create a lot of files, especially within a single folder
  - If you’re creating 10 000+ files, you should probably rethink your workflow
  - Consider removing temporary files after job is finished
- Whenever possible, separate serial jobs from parallel ones for efficient usage of resources.