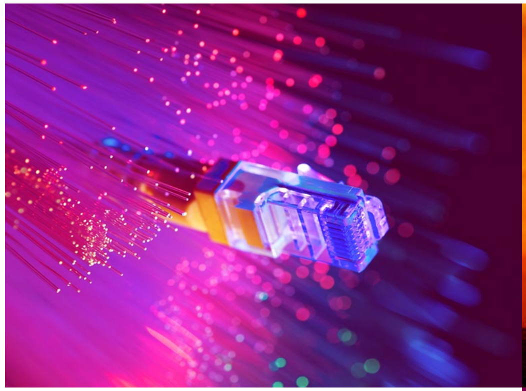


Workflows with Singularity Containers

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

Outline

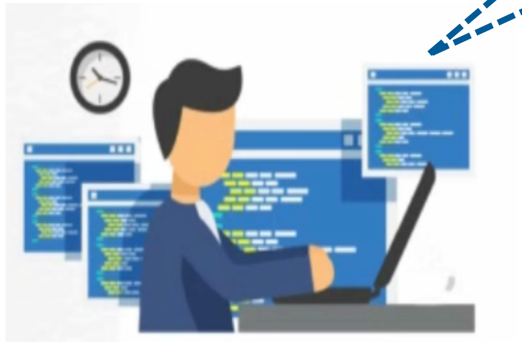
- Why containers in workflows?
- Configuring singularity with Nextflow
- Puhti recipe for running Nextflow pipeline
- Reporting and visualisation
- Running *nf-core* pipelines

Why Containers in Workflows?

Why Containers in Workflows?

Developer

Workflow works fine for me



Tester

No. It does NOT work here

Nextflow Pipeline with Containers

- Built-in integration with containers (and Conda)
- Advantages
  - Maintainability
  - Portability
  - Reproducibility
- Popular containers
  - Docker
  - Singularity

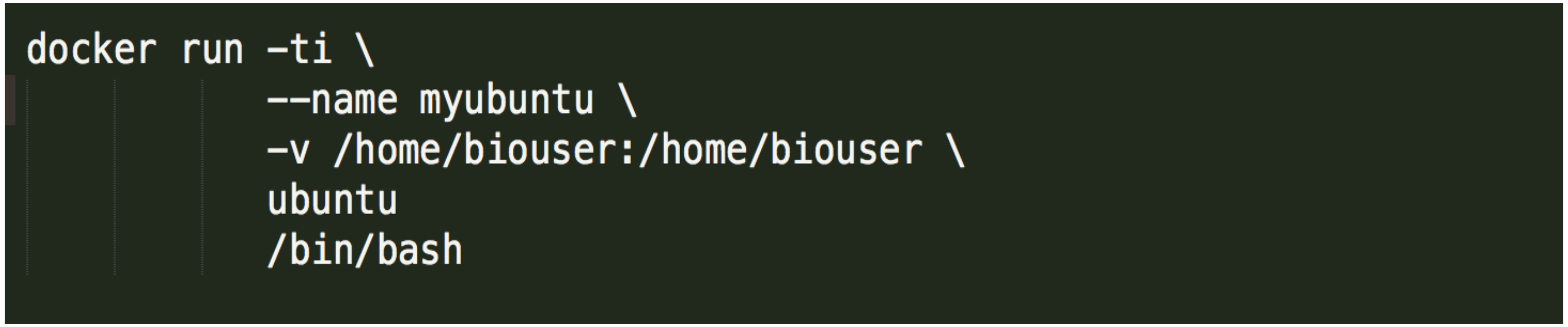
Nextflow Integrates Nicely with Containers

Mounting host's folders, staging inputs and starting containers

A docker example for mounting volumes:

```
docker run [options] -v <HOST_PATH>:<CONTAINER_PATH>     IMAGE  [CMD]
```

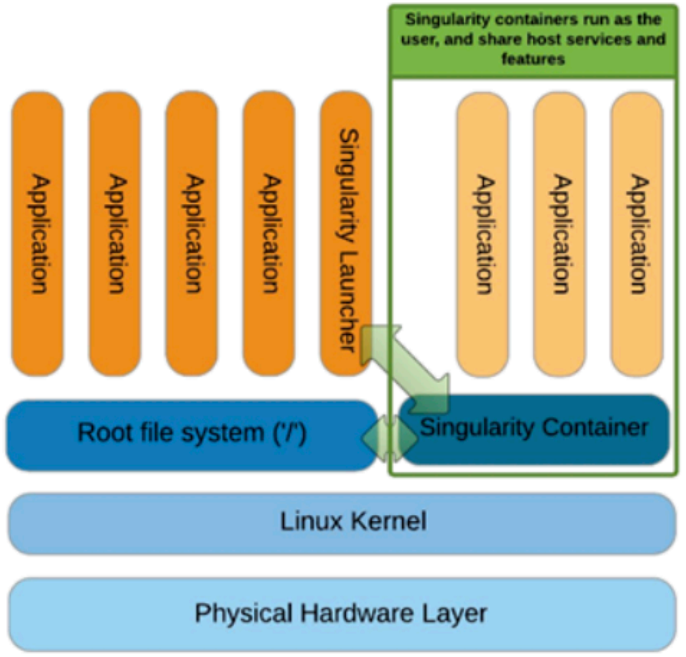
using -v



```
docker run -ti \
           --name myubuntu \
           -v /home/biouser:/home/biouser \
           ubuntu
           /bin/bash
```

Use of Singularity Containers is even better

- 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)

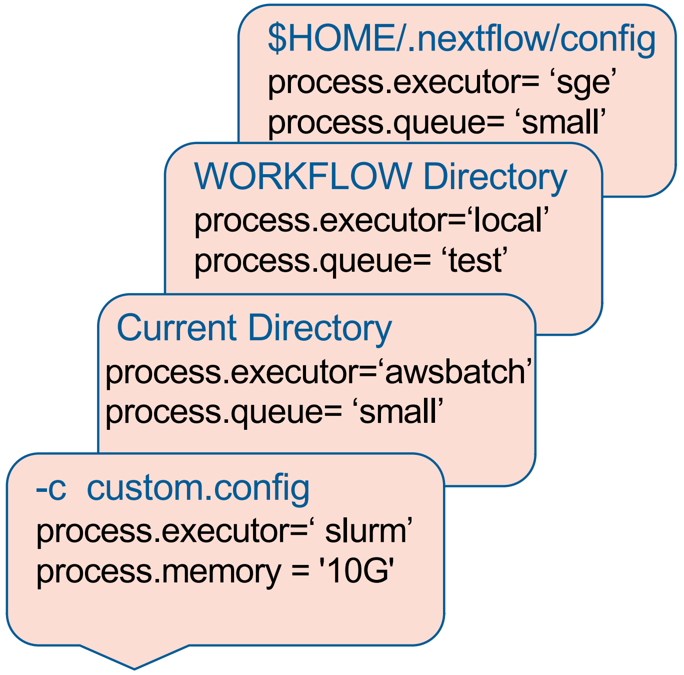


HPC Container
Singularity

Configuring Singularity with Nextflow

Nextflow Configuration File(s)

- Nextflow can load pipeline configurations from multiple locations:
  - home directory
  - workflow directory (if not current dir)
  - current directory
  - config file is given with -c <config file>
- Understand the overriding behaviour



$HOME/.nextflow/config

```
process.executor= ‘sge’
process.queue= ‘small’
```

WORKFLOW Directory

```
process.executor=‘local’
process.queue= ‘test’
```

Current Directory

```
process.executor=‘awsbatch’
process.queue= ‘small’
```

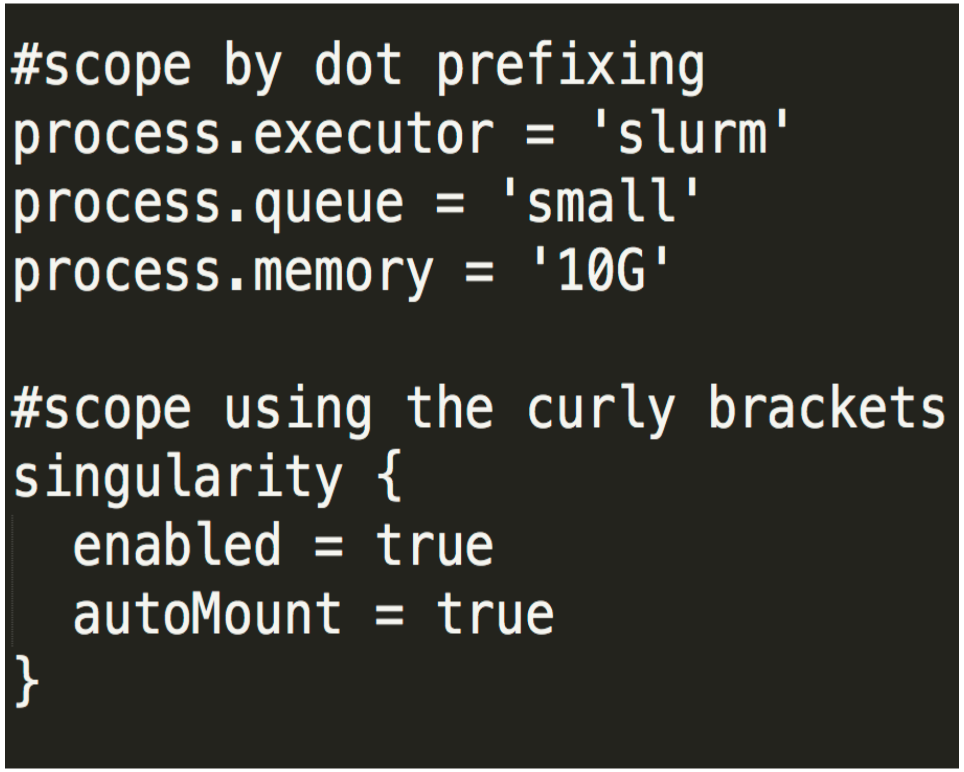
-c custom.config

```
process.executor=‘ slurm’
process.memory = '10G'
```

```
- process.executor=’slurm’
- process.queue= ‘small’
- process.memory=’10G’
```

Configuration Files: scopes

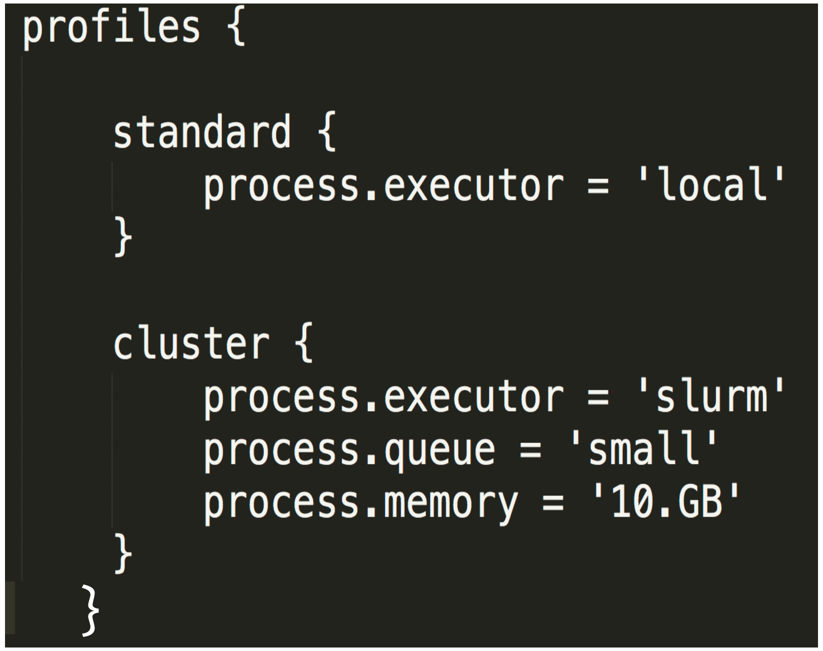
- Configuration settings can be organized in different scopes
- Nextflow scopes
  - *- env*
  - *- params*
  - *- process*
  - *- executor…*



```
#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



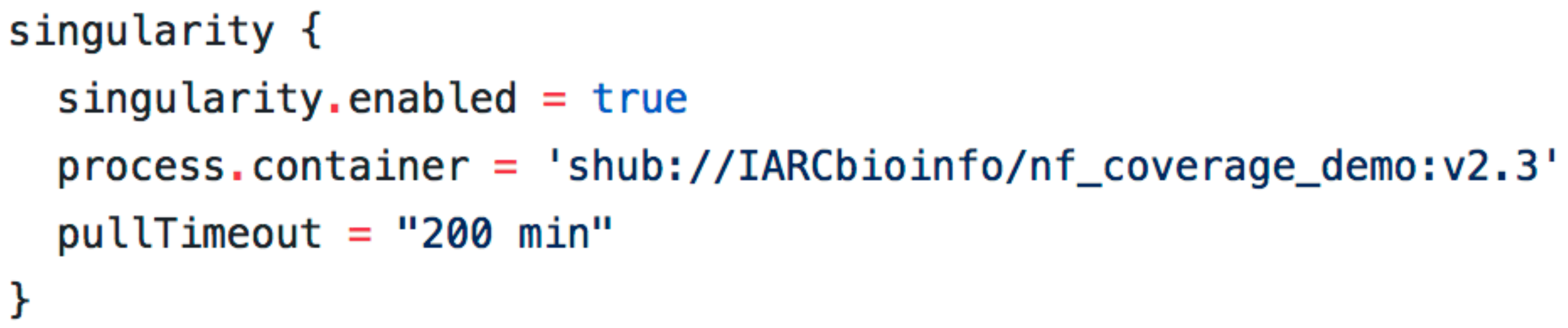
```
profiles {

    standard {
        process.executor = 'local'
    }

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

Configuring Singularity with Nextflow

Options to use singularity with nextflow:

- Commandline option : –with-singularity /path/to/image.img

- In ‘nextflow.config’ file as profile



```
singularity {
  singularity.enabled = true
  process.container = 'shub://IARCbioinfo/nf_coverage_demo:v2.3'
  pullTimeout = "200 min"
}
```

Puhti Recipe for Running Nextflow Pipeline

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

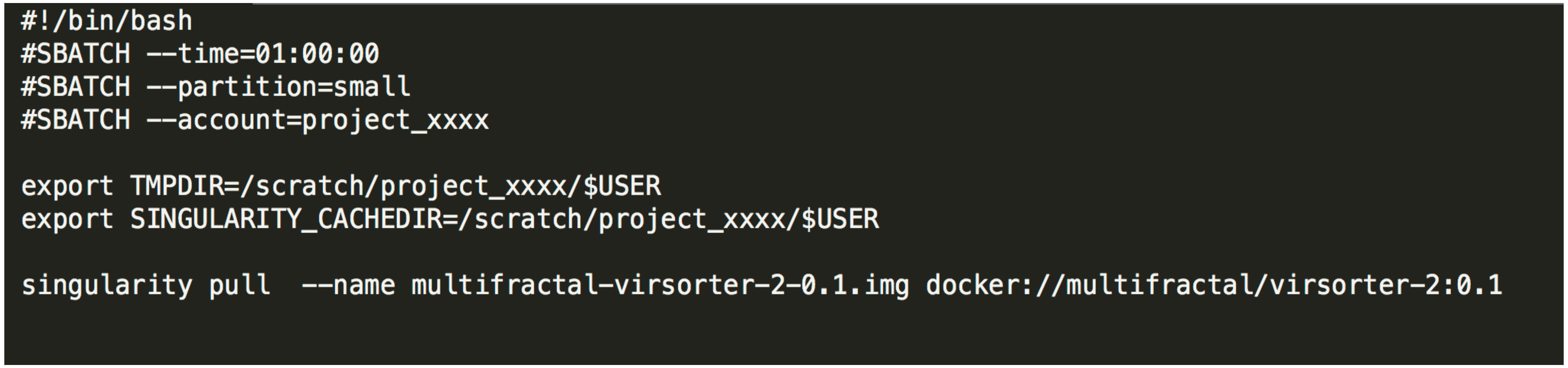
Preparing Singularity Images if Needed

- Pull a Singularity image from a singularity registry
  - Use Puhti
- Convert a Docker image to Singularity one
  - Puhti can work most of the cases
- Buid a Singularity image from scratch
  - Puhti can’t be used

Preparing Singularity Images on Puhti

- Convert docker images to singularity image
    - Interactive node
    - Batch mode

Example batch script:



```
#!/bin/bash
#SBATCH --time=01:00:00
#SBATCH --partition=small
#SBATCH --account=project_xxxx

export TMPDIR=/scratch/project_xxxx/$USER
export SINGULARITY_CACHEDIR=/scratch/project_xxxx/$USER

singularity pull  --name multifractal-virsorter-2-0.1.img docker://multifractal/virsorter-2:0.1
```

Load Nextflow Environment on Puhti

Activate conda environment for nextflow

```
module load bioconda
source activate nextflow
```

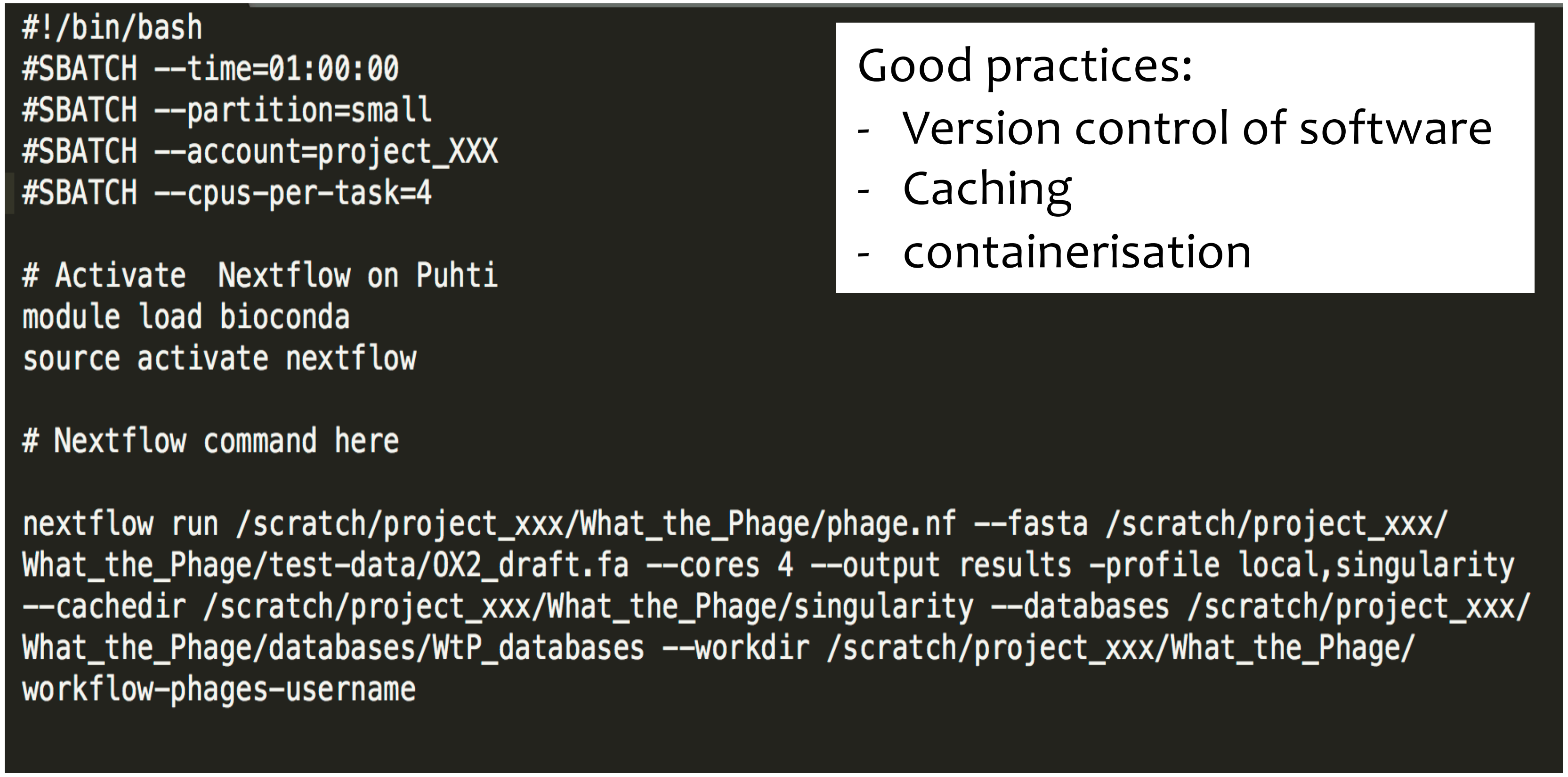
Custom installations

```
export PROJAPPL=/projappl/project_xxx     # Edit the project name
module load bioconda
conda create -n next_flow -c bioconda nextflow=0.30.1  # See note below
source activate next_flow
```

Prepare Your Application Dependencies

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

Run Nextflow as a Batch Job



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

# Activate  Nextflow on Puhti
module load bioconda
source activate nextflow

# Nextflow command here

nextflow run /scratch/project_xxx/What_the_Phage/phage.nf --fasta /scratch/project_xxx/
What_the_Phage/test-data/OX2_draft.fa --cores 4 --output results -profile local,singularity
--cachedir /scratch/project_xxx/What_the_Phage/singularity --databases /scratch/project_xxx/
What_the_Phage/databases/WtP_databases --workdir /scratch/project_xxx/What_the_Phage/
workflow-phages-username
```

Good practices:
- Version control of software
- Caching
- containerisation

Reporting and Visualisation

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
```

Running Nextflow Pipeline from GitHubs

- Share pipelines with Github

- If a Nextflow project is hosted in a GitHub repository at the address http://github.com/user/test, one can execute pipeline as below:

```
Nextflow run user/test
```

Pay attention to version control (use e.g., –r v1.1)

- Git clone and launch (offline)

```
Nextflow run main.nf
```

main.nf is nextflow script name

Deploying *nf-core* Pipelines

- A community effort to collect a curated set of analysis pipelines built using Nextflow
- Provides nice guidelines and pipelines
  - Explore more on pipelines
  - pipelines: released (33); development (15)
- Each pipeline has its own documentation
  - e.g., nextflow run nf-core/rnaseq -r 3.0 --help
- Join on slack/twitter

Deploying *nf-core* Pipelines at CSC

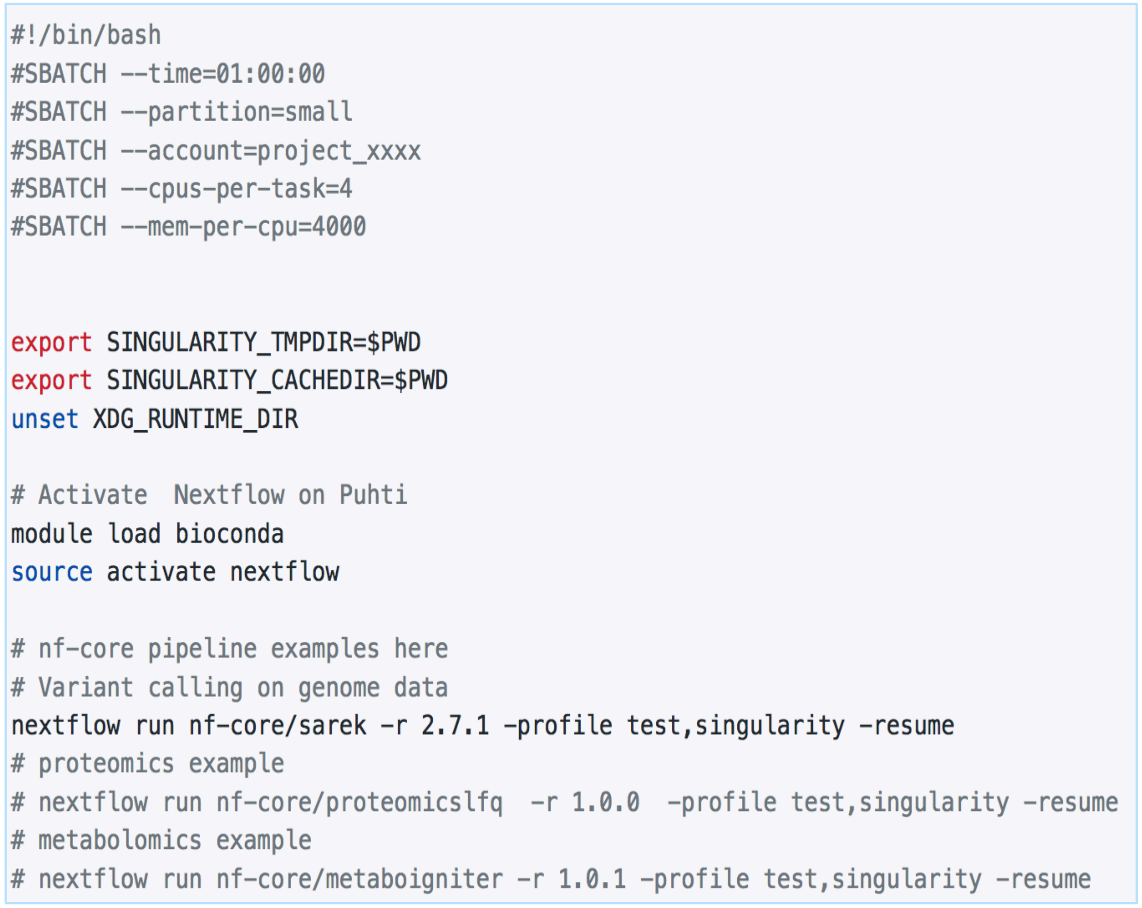
- A basic batch job script to
  test if pipeline works

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

- Containers building can fail in initial attempts

- Explore more by cloning pipeline repository

- Use singularity as profile



```
#!/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 SINGULARITY_TMPDIR=$PWD
export SINGULARITY_CACHEDIR=$PWD
unset XDG_RUNTIME_DIR

# Activate  Nextflow on Puhti
module load bioconda
source activate nextflow

# 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
```

Time for Practicals !!!

- Where to run practicals: Interactive nodes on Puhti
- Tutorials: Nextflow pipeline with containers and other useful features (Tutorial 3)
- Expected outcome from tutorials:
  - Able to use singularity containers in your workflow
  - Understand user-defined profiles
  - Configure reporting and visualisation capabilities
  - Able to deploy nexflow pipelines as batch jobs on Puhti