Formative assessment week 9: Snakemake, Conda, HPC

You have so far:

- Generated the code that performs your analysis
- Developed a reproducible environment for your project
- Integrated the environment and code into a Snakemake pipeline
- Version controlled the code, pipelining and environments using git and GitHub

We will now:

- Migrate the code, data and environments to HPC
- Update the Snakemake configuration to run on HPC
- Document these changes for ease of future reproducibility

1. Setting up on HPC

A. Getting our codebase on HPC

Your project should all be under version control on GitHub. This makes it extremely easy to create a clone of your code on another machine (such as HPC).

1. Login to bluecrystal4
2. Create a directory where you would like to keep your projects. A good option is to create a directory called `$WORK/projects/`
3. Navigate to `$WORK/projects` and clone your GitHub repository there.
   - Add the git module in bluecrystal4
   - Get the git repository's HTTPS URL from GitHub
   - Use `git clone` to clone the repository

If your code is not on github you can use `scp` to copy your code into the relevant directory.

Example steps:

```
mkdir -p $WORK/projects/
cd $WORK/projects
git clone https://github.com/explodecomputer/MSHDS-AHDS-formative.git
```

B. Getting our data on HPC

Use `scp` to copy your original data into `$WORK/projects/<formative>/data/original`. You can see how the data was setup originally on your computer in the `directory-setup-commands.sh` script.

C. Setup your conda environment

You should have conda ready to run on HPC from the practical sessions. Create the environment needed for your formative assessment.

Remember you will need to add the slurm plugin (https://snakemake.github.io/snakemake-plugin-catalog/plugins/executor/slurm.html).

Example:

```
. ~/.initMamba.sh
conda env create -f ahds_formative_environment.yml
pip install snakemake-executor-plugin-slurm
```

D. Check that your scripts work as expected

- Ideally you would only run analysis on HPC by submitting your scripts to the worker nodes. However it can be helpful to check that trivial errors are not present such as incorrect directory names etc before submitting your job. You can check this by **starting** to run your analysis scripts in the login node to see if it works and then killing it as quickly as possible (`ctrl+c`).

2. Create a submission script

Using examples from the week 9 `practical_1` example, create a job submission script that will run your pipeline on HPC by submitting a single job that runs all the steps.

Example file contents for e.g. `run_analysis.sh`:

```bash
#!/bin/bash

set -e

#SBATCH --job-name=test_job
#SBATCH --partition=teach_cpu
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --time=0:10:00
#SBATCH --mem=100M
#SBATCH --account=SSCM033324
#SBATCH --output ./slurm_logs/%j.out


cd "${SLURM_SUBMIT_DIR}"

# Conda environment
source ~/.initMamba.sh
mamba activate ahds_week9

# Setup directories
mkdir -p logs
mkdir -p data/derived
mkdir -p data/original
mkdir -p results

# Run steps
cd code
bash 1-data-check-bm.sh > ../logs/1-data-check-bm.log
bash 2-data-check-accel.sh > ../logs/2-data-check-accel.log
bash 3-data-fix-accel.sh > ../logs/3-data-fix-accel.log
bash 4-list-accel-ids.sh > ../logs/4-list-accel-ids.log
Rscript 5-generate-sample.R > ../logs/5-generate-sample.log
Rscript 6-demo-data-prep.R > ../logs/6-demo-data-prep.log
```

To submit this:

```
sbatch run_analysis.sh
```

3. Use Snakemake to submit the analysis to HPC

A. Snakemake dry-run

Try a dry-run of the whole pipeline *without* submitting it to HPC (e.g. just run it interactively as if you were running it on your local computer). Check for any issues and fix.

Example:

```
snakemake -pn
```

B. Snakemake execution

You have setup a `slurm_profile` that can specify how the Snakemake pipeline can be executed on the HPC worker nodes. Using the examples in the week 9 `practical_2` session, try submitting the whole pipeline to HPC.

Note that it may be prudent to run this within a `tmux` session (why?)

You should have a slurm profile specified in e.g. `~/.config/snakemake/slurm_profile/config.yaml` that looks like:

```
default-resources:
  - slurm_partition="teach_cpu"
  - slurm_account="SSCM033324"
  - mem_mb="100"
  - runtime="10"
  - ntasks="1"
  - nodes="1"
jobs: 10
printshellcmds: True
scheduler: greedy
use-conda: True
```

You can submit the Snakemake pipeline using

```
snakemake --executor slurm --profile slurm_profile
```

Running this inside `tmux` is sensible because it provides a persistent session to allow the Snakemake pipeline to continue watching and submitting jobs to the scheduler. This is helpful for longer running jobs because if you ran it just in your interactive shell session then when you log out (or get disconnected) then the Snakemake process will be killed.

4. Update your project repository

Given the changes that you have made, what should you do to finalise and record these processes?

- Update your `REAMDE.md` to explain how the job should be exectured using a submission script or Snakemake.
- Update your `README.md` to include the config template for Snakemake to run and how to get it installed
- Update your `environment.yml` file to include the slurm plugin e.g. `conda list -e > environment.yml`
- Commit your changes to git and synchronise with GitHub.

5. Advanced: Re-factoring to run in parallel

Currently the whole pipeline is run sequentially. That means that each job waits for the previous job to finish before starting. This does not take advantage of the benefits of HPC which can run multiple jobs in parallel.

Identify areas of your pipeline that could feasibly be split into separate jobs that can run in parallel. Refactor your pipeline to take advantage of this, and implement it into Snakemake.

TBD