MINISTRY OF BUSINESS, INNOVATION & EMPLOYMENT
HĪKINA WHAKATUTUKI

THE UNIVERSITY OF AUCKLAND NEW ZEALAND

NIWA Taihoro Nukurangi

UNIVERSITY of OTAGO Te Whare Wānanga o Otāgo NEW ZEALAND

Manaaki Whenua Landcare Research

NeSI New Zealand eScience Infrastructure

From interactive data exploration to model fitting

A Data Science journey on NeSI

10 February 2022 - Maxime Rio, Chris Scott and Alexander Pletzer

Few words about myself

A Data Scientist at NIWA

- help researchers and customers analyze their data
- develop new analysis methods

A Data Science Engineer at NeSI

- optimise code and workflows for users
- train users to use the platform
- test and deploy new tools
- point of contact for data science related offerings

and I do like penguins

Computational Science consultancy

- A service offered to NeSI platform users, generally at no cost to the researcher
- NeSI **Research Software Engineers** and **Data Science Engineer** work directly with research group members to **raise the capability** of the research group

Our team can assist with:

- **Workflow parallelisation** – allowing more inputs to be processed simultaneously
- **Software parallelisation** – use of technologies such as OpenMP or MPI to process one single input more quickly
- **Code optimisation** – redesign of algorithms to improve overall speed or efficiency of resource use
- **Improving I/O performance** – speed up reading from or writing to the disk, or to reduce the amount of data that must be read or written
- **Porting to GPU** – accelerate code by offloading computations to a coprocessor
- **Improving software sustainability** – introducing best practices such as version control and unit testing

Contact support@nesi.org.nz

From early EDA to full automation

collection of scripts
and notebooks

non-interactive jobs
on big machines

fully automated pipeline
(towards MLOps)

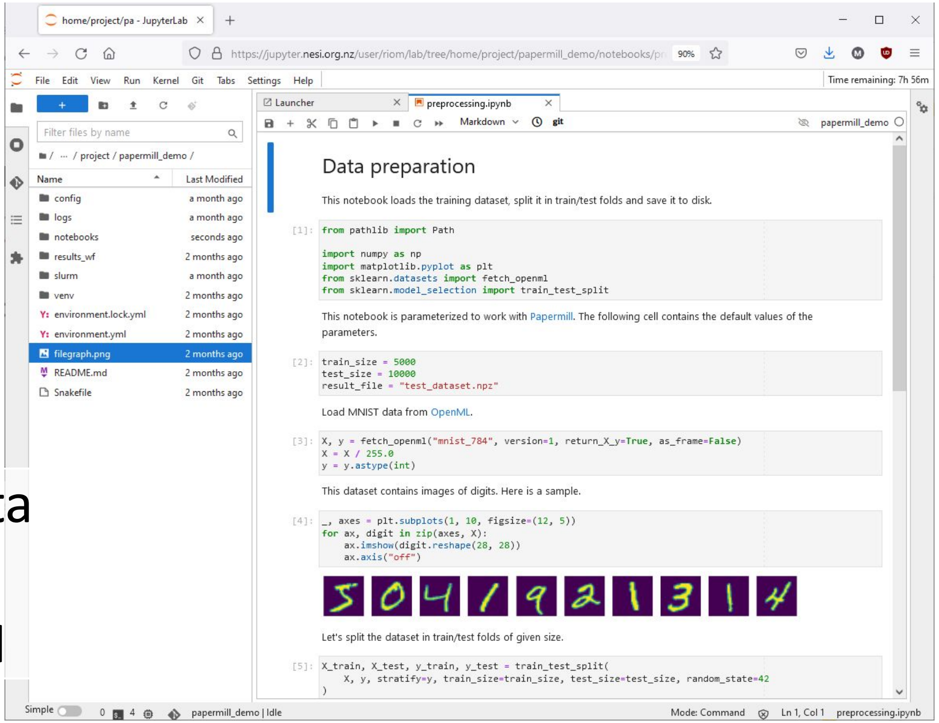
Interactive environments

Jupyter @ https://jupyter.nesi.org.nz

- JupyterLab
- Virtual Desktop
- RStudio (coming)
- Matlab (coming)

My main usages

- Load, manipulate and visualize data
- Develop code on small examples
- Ensure my code is running on NeSI


home/project/pa - JupyterLab
https://jupyter.nesi.org.nz/user/riom/lab/tree/home/project/papermill_demo/notebooks/pr
File Edit View Run Kernel Git Tabs Settings Help
Time remaining: 7h 56m
Launcher
preprocessing.ipynb
Markdown
papermill_demo
Filter files by name
/ … / project / papermill_demo /
Simple
papermill_demo | Idle
Mode: Command
Ln 1, Col 1
preprocessing.ipynb

Name	Last Modified
config	a month ago
logs	a month ago
notebooks	seconds ago
results_wf	2 months ago
slurm	a month ago
venv	2 months ago
environment.lock.yml	2 months ago
environment.yml	2 months ago
filegraph.png	2 months ago
README.md	2 months ago
Snakefile	2 months ago

Data preparation

This notebook loads the training dataset, split it in train/test folds and save it to disk.

```
[1]: from pathlib import Path

     import numpy as np
     import matplotlib.pyplot as plt
     from sklearn.datasets import fetch_openml
     from sklearn.model_selection import train_test_split
```

This notebook is parameterized to work with Papermill. The following cell contains the default values of the parameters.

```
[2]: train_size = 5000
     test_size = 10000
     result_file = "test_dataset.npz"
```

Load MNIST data from OpenML.

```
[3]: X, y = fetch_openml("mnist_784", version=1, return_X_y=True, as_frame=False)
     X = X / 255.0
     y = y.astype(int)
```

This dataset contains images of digits. Here is a sample.

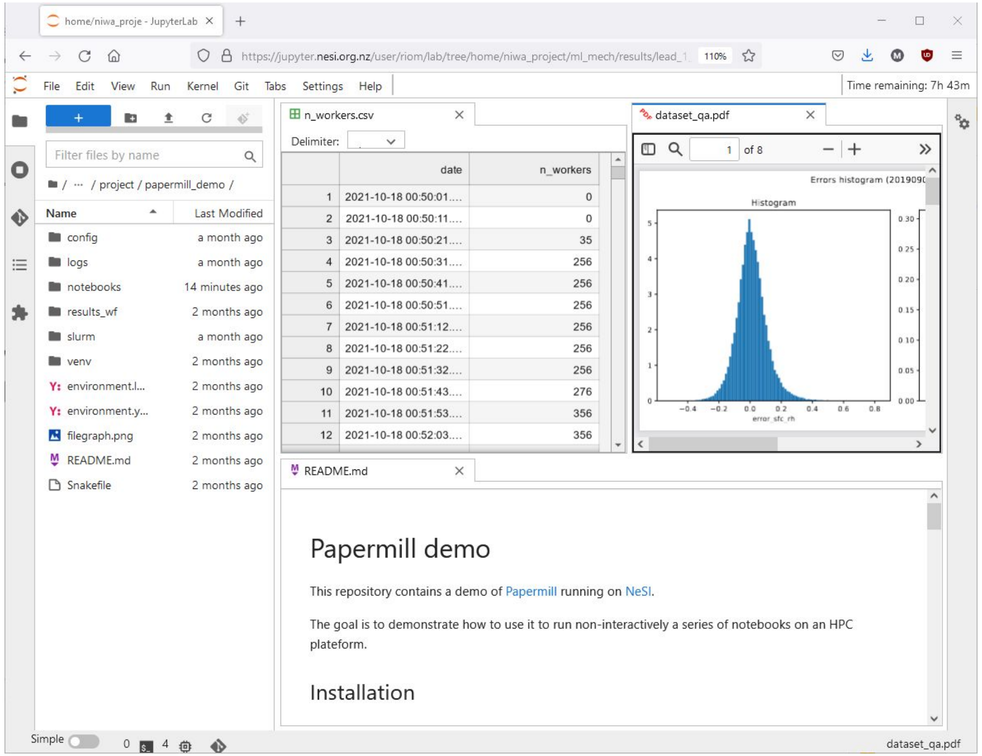
```
[4]: _, axes = plt.subplots(1, 10, figsize=(12, 5))
     for ax, digit in zip(axes, X):
         ax.imshow(digit.reshape(28, 28))
         ax.axis("off")
```

Let's split the dataset in train/test folds of given size.

```
[5]: X_train, X_test, y_train, y_test = train_test_split(
         X, y, stratify=y, train_size=train_size, test_size=test_size, random_state=42
     )
```

JupyterLab - zoom in

- Notebooks
- File Browser
  - preview .html, .csv, .pdf, markdown, images
  - upload / download
- Text editor & Console (autocomplete)
- Terminal
- Slurm Queue Manager
- Extensible (e.g. Dask dashboard)



Name	Last Modified
config	a month ago
logs	a month ago
notebooks	14 minutes ago
results_wf	2 months ago
slurm	a month ago
venv	2 months ago
environment.l...	2 months ago
environment.y...	2 months ago
filegraph.png	2 months ago
README.md	2 months ago
Snakefile	2 months ago

	date	n_workers
1	2021-10-18 00:50:01....	0
2	2021-10-18 00:50:11....	0
3	2021-10-18 00:50:21....	35
4	2021-10-18 00:50:31....	256
5	2021-10-18 00:50:41....	256
6	2021-10-18 00:50:51....	256
7	2021-10-18 00:51:12....	256
8	2021-10-18 00:51:22....	256
9	2021-10-18 00:51:32....	256
10	2021-10-18 00:51:43....	276
11	2021-10-18 00:51:53....	356
12	2021-10-18 00:52:03....	356

Papermill demo

This repository contains a demo of Papermill running on NeSI.

The goal is to demonstrate how to use it to run non-interactively a series of notebooks on an HPC plateform.

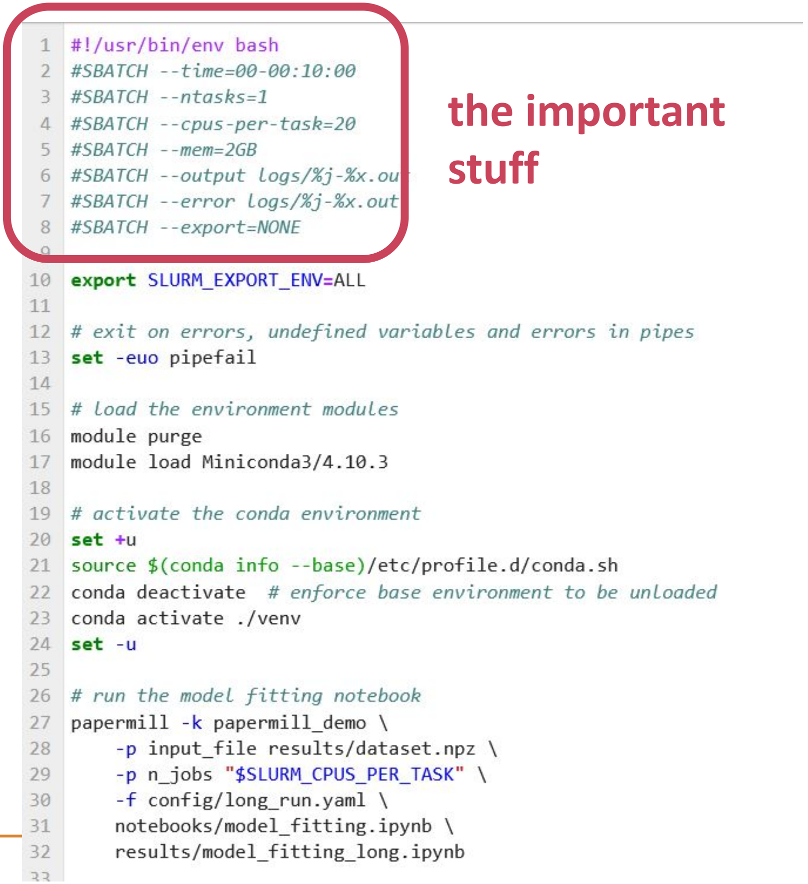
Installation

Slurm to access more resources

- Access the cool hardware
  - more RAM, more cores
  - very large memory nodes
  - A100s GPUs
- Describe job as a Slurm script
  - Bash shell script
  - resource set in a header
  - scheduled in a queue by Slurm
- Use job arrays for repetitive tasks

Slurm to access more resources

- Access the cool hardware
  - more RAM, more cores
  - very large memory nodes
  - A100s GPUs
- Describe job as a Slurm script
  - Bash shell script
  - resource set in a header
  - scheduled in a queue by Slurm
- Use job arrays for repetitive tasks



```bash
#!/usr/bin/env bash
#SBATCH --time=00-00:10:00
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=20
#SBATCH --mem=2GB
#SBATCH --output logs/%j-%x.ou
#SBATCH --error logs/%j-%x.out
#SBATCH --export=NONE

export SLURM_EXPORT_ENV=ALL

# exit on errors, undefined variables and errors in pipes
set -euo pipefail

# load the environment modules
module purge
module load Miniconda3/4.10.3

# activate the conda environment
set +u
source $(conda info --base)/etc/profile.d/conda.sh
conda deactivate  # enforce base environment to be unloaded
conda activate ./venv
set -u

# run the model fitting notebook
papermill -k papermill_demo \
    -p input_file results/dataset.npz \
    -p n_jobs "$SLURM_CPUS_PER_TASK" \
    -f config/long_run.yaml \
    notebooks/model_fitting.ipynb \
    results/model_fitting_long.ipynb
```

the important stuff

Reproducibility: the software stack

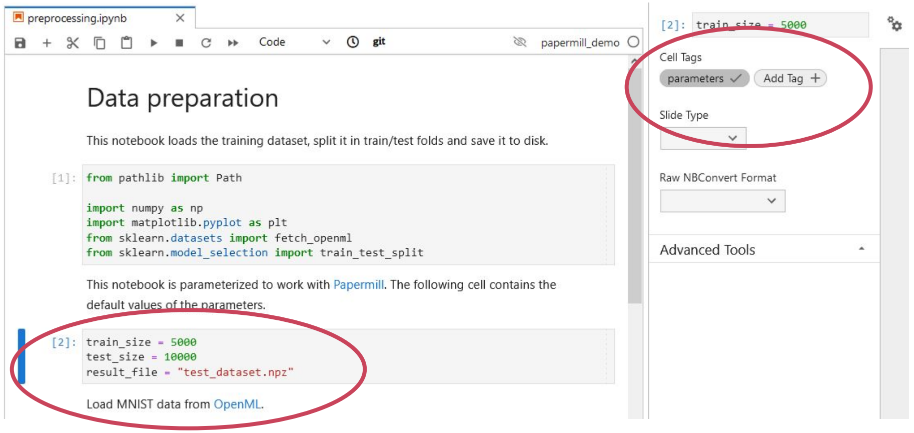
- Environment modules
  - prepared by NeSI team with 💕
  - optimized for our hardware
- Python virtual environments
  - only for Python packages
  - can use Environment modules
- Conda environments
  - Python and other software (e.g. opencv)
  - less nice interactions with Env. modules
- Singularity containers

Workflow management system

- Split large code into smaller pieces
    - self-contained scripts
    - notebooks run with **papermill**
    - use files as inputs and outputs
- **Snakemake** (or other) to schedule tasks
    - parallelise work when possible
    - fine grained control on resources (GPUs, memory, cores)
    - avoid recomputing when possible

Check our training material @ https://github.com/nesi/snakemake_workshop

papermill (digression)



preprocessing.ipynb

Code · git · papermill_demo

Data preparation

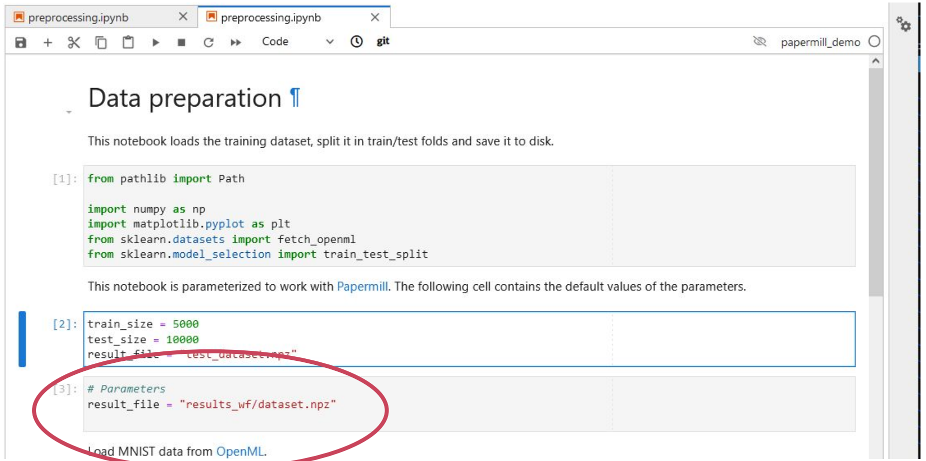
This notebook loads the training dataset, split it in train/test folds and save it to disk.

```
[1]: from pathlib import Path

     import numpy as np
     import matplotlib.pyplot as plt
     from sklearn.datasets import fetch_openml
     from sklearn.model_selection import train_test_split
```

This notebook is parameterized to work with Papermill. The following cell contains the default values of the parameters.

```
[2]: train_size = 5000
     test_size = 10000
     result_file = "test_dataset.npz"
```

Load MNIST data from OpenML.



preprocessing.ipynb · preprocessing.ipynb

Code · git · papermill_demo

Data preparation ¶

This notebook loads the training dataset, split it in train/test folds and save it to disk.

```
[1]: from pathlib import Path

     import numpy as np
     import matplotlib.pyplot as plt
     from sklearn.datasets import fetch_openml
     from sklearn.model_selection import train_test_split
```

This notebook is parameterized to work with Papermill. The following cell contains the default values of the parameters.

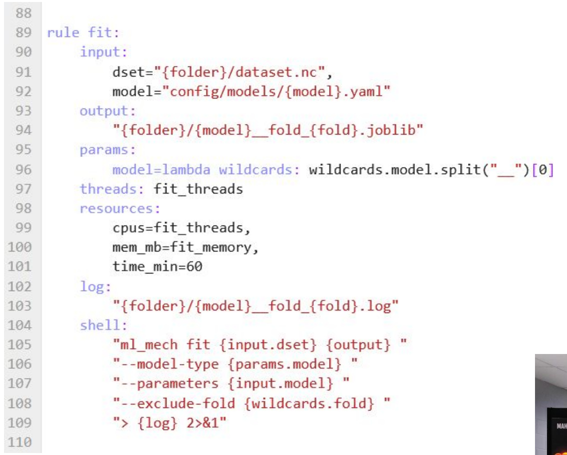
```
[2]: train_size = 5000
     test_size = 10000
     result_file = "test_dataset.npz"

[3]: # Parameters
     result_file = "results_wf/dataset.npz"
```

Load MNIST data from OpenML.

papermill -k **kernel_name** -p parameter value input.ipynb output.ipynb

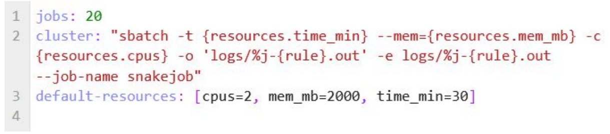
Snakemake workflow example

1. Describe tasks and resources in a **Snakefile**



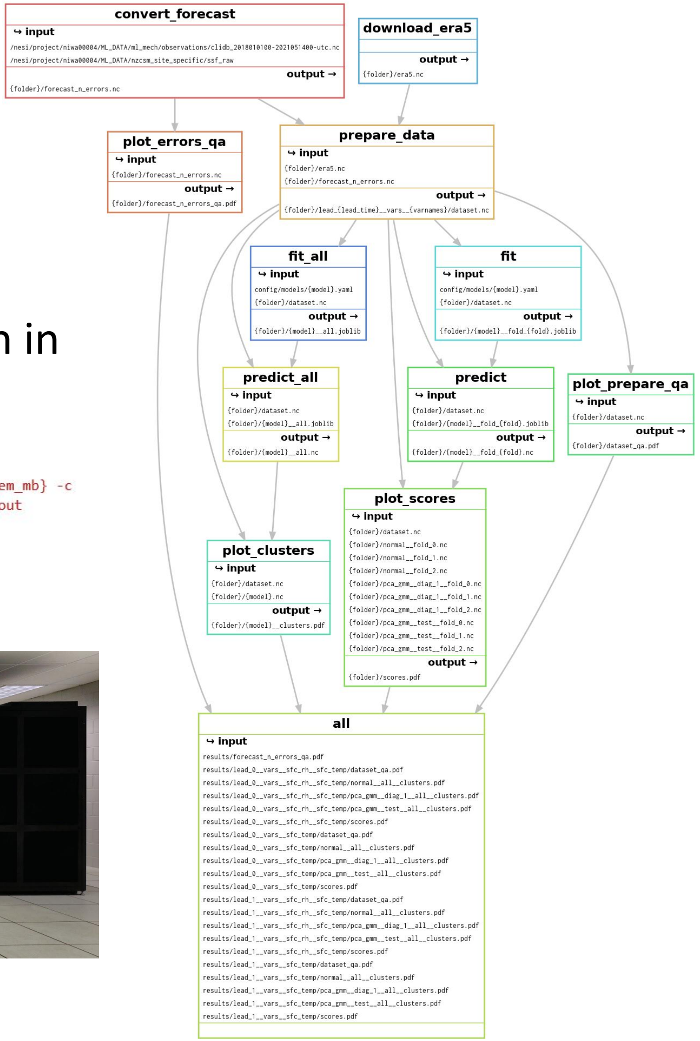
```

rule fit:
    input:
        dset="{folder}/dataset.nc",
        model="config/models/{model}.yaml"
    output:
        "{folder}/{model}__fold_{fold}.joblib"
    params:
        model=lambda wildcards: wildcards.model.split("__")[0]
    threads: fit_threads
    resources:
        cpus=fit_threads,
        mem_mb=fit_memory,
        time_min=60
    log:
        "{folder}/{model}__fold_{fold}.log"
    shell:
        "ml_mech fit {input.dset} {output} "
        "--model-type {params.model} "
        "--parameters {input.model} "
        "--exclude-fold {wildcards.fold} "
        "> {log} 2>&1"

```

2. Describe task submission in a **profile** configuration file



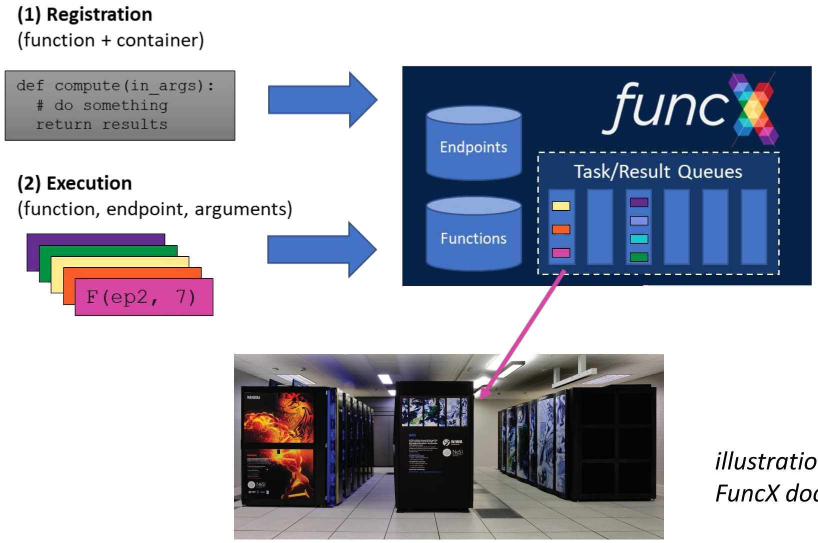
```
jobs: 20
cluster: "sbatch -t {resources.time_min} --mem={resources.mem_mb} -c {resources.cpus} -o 'logs/%j-{rule}.out' -e logs/%j-{rule}.out --job-name snakejob"
default-resources: [cpus=2, mem_mb=2000, time_min=30]

```

3. Run on the big machines 🎉


convert_forecast
↪ input
/nesi/project/niwa00004/ML_DATA/ml_mech/observations/clidb_2018010100-2021051400-utc.nc
/nesi/project/niwa00004/ML_DATA/nzcsm_site_specific/ssf_raw
output →
{folder}/forecast_n_errors.nc
download_era5
output →
{folder}/era5.nc
plot_errors_qa
↪ input
{folder}/forecast_n_errors.nc
output →
{folder}/forecast_n_errors_qa.pdf
prepare_data
↪ input
{folder}/era5.nc
{folder}/forecast_n_errors.nc
output →
{folder}/lead_{lead_time}__vars__{varnames}/dataset.nc
fit_all
↪ input
config/models/{model}.yaml
{folder}/dataset.nc
output →
{folder}/{model}__all.joblib
fit
↪ input
config/models/{model}.yaml
{folder}/dataset.nc
output →
{folder}/{model}__fold_{fold}.joblib
predict_all
↪ input
{folder}/dataset.nc
{folder}/{model}__all.joblib
output →
{folder}/{model}__all.nc
predict
↪ input
{folder}/dataset.nc
{folder}/{model}__fold_{fold}.joblib
output →
{folder}/{model}__fold_{fold}.nc
plot_prepare_qa
↪ input
{folder}/dataset.nc
output →
{folder}/dataset_qa.pdf
plot_clusters
↪ input
{folder}/dataset.nc
{folder}/{model}.nc
output →
{folder}/{model}__clusters.pdf
plot_scores
↪ input
{folder}/dataset.nc
{folder}/normal__fold_0.nc
{folder}/normal__fold_1.nc
{folder}/normal__fold_2.nc
{folder}/pca_gmm__diag_1__fold_0.nc
{folder}/pca_gmm__diag_1__fold_1.nc
{folder}/pca_gmm__diag_1__fold_2.nc
{folder}/pca_gmm__test__fold_0.nc
{folder}/pca_gmm__test__fold_1.nc
{folder}/pca_gmm__test__fold_2.nc
output →
{folder}/scores.pdf
all
↪ input
results/forecast_n_errors_qa.pdf
results/lead_0__vars__sfc_rh__sfc_temp/dataset_qa.pdf
results/lead_0__vars__sfc_rh__sfc_temp/normal__all__clusters.pdf
results/lead_0__vars__sfc_rh__sfc_temp/pca_gmm__diag_1__all__clusters.pdf
results/lead_0__vars__sfc_rh__sfc_temp/pca_gmm__test__all__clusters.pdf
results/lead_0__vars__sfc_rh__sfc_temp/scores.pdf
results/lead_0__vars__sfc_temp/dataset_qa.pdf
results/lead_0__vars__sfc_temp/normal__all__clusters.pdf
results/lead_0__vars__sfc_temp/pca_gmm__diag_1__all__clusters.pdf
results/lead_0__vars__sfc_temp/pca_gmm__test__all__clusters.pdf
results/lead_0__vars__sfc_temp/scores.pdf
results/lead_1__vars__sfc_rh__sfc_temp/dataset_qa.pdf
results/lead_1__vars__sfc_rh__sfc_temp/normal__all__clusters.pdf
results/lead_1__vars__sfc_rh__sfc_temp/pca_gmm__diag_1__all__clusters.pdf
results/lead_1__vars__sfc_rh__sfc_temp/pca_gmm__test__all__clusters.pdf
results/lead_1__vars__sfc_rh__sfc_temp/scores.pdf
results/lead_1__vars__sfc_temp/dataset_qa.pdf
results/lead_1__vars__sfc_temp/normal__all__clusters.pdf
results/lead_1__vars__sfc_temp/pca_gmm__diag_1__all__clusters.pdf
results/lead_1__vars__sfc_temp/pca_gmm__test__all__clusters.pdf
results/lead_1__vars__sfc_temp/scores.pdf

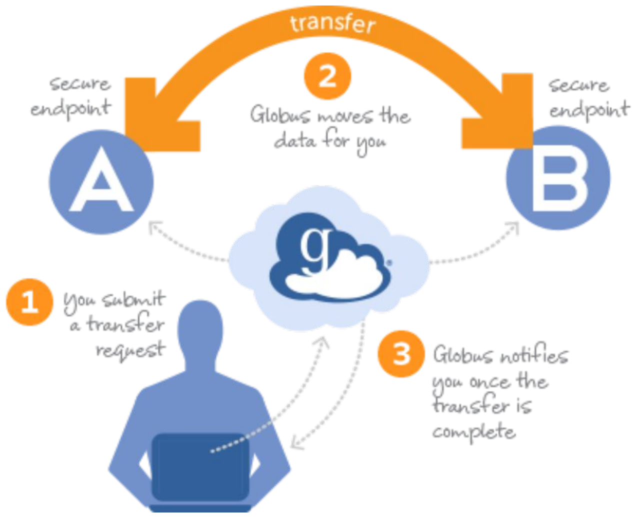
More automation

- **scron** for repetitive task execution
- **Globus** for large data transfer
- **FuncX** for remote task execution



```
def compute(in_args):
  # do something
  return results
```

illustration adapted from FuncX documentation



transfer
secure endpoint
secure endpoint
2
Globus moves the data for you
A
B
1
You submit a transfer request
3
Globus notifies you once the transfer is complete

image from Globus documentation

Final words

- Expression of interest for testing our HGX A100 kits https://tinyurl.com/nesihgx
- Join our mailing lists at http://eepurl.com/grV9if (training alerts, newsletters, event announcements…)
- More webinar recordings on our Youtube channel
- Any question? Email our team support@nesi.org.nz

How would you like to use our platform for Data Science? Please let us know! maxime.rio@nesi.org.nz

Thank you for your attention :-)