

# Model Agnostic Framework

**Online Training Session Provincial, Territorial, and University Research Partners**

Kasra Keshavarz  
Research Scientist  
Schulich School of Engineering  
University of Calgary

Alain Pietroniro, PhD, PEng  
Professor and Canada Research Chair  
Schulich School of Engineering  
University of Calgary



UNIVERSITY OF  
**CALGARY**

# Acknowledgements



**GLOBAL WATER FUTURES**



**Digital Research  
Alliance of Canada**

**Alliance de recherche  
numérique du Canada**



**UNIVERSITY OF  
CALGARY**



**UNU  
INWEH**



Environment and  
Climate Change Canada

Environnement et  
Changement climatique Canada

*Alberta*  
Government



**ACCESS**



UNIVERSITY OF  
CALGARY

# Outline

- Community Hydrological Modelling Recipes
- Computational Resources for the Training Session
- Setting up ECCC's National Water Model—MESH

# Outline

- Community Hydrological Modelling Recipes
- Computational Resources for the Training Session
- Setting up ECCC's National Water Model—MESH

# Recipe for Community Hydrological Modelling



## Water Resources Research®

### RESEARCH ARTICLE

10.1029/2021WR031753

#### Key Points:

- Reproducible, transparent modeling increases confidence in model simulations and requires careful tracking of all model configuration steps
- We show an example of model configuration code applied globally that is traced and shared through a version control system
- Standardizing file formats and sharing of code can increase efficiency and reproducibility of modeling studies

#### Correspondence to:

W. J. M. Knoben,  
wouter.knoben@usask.ca

### Community Workflows to Advance Reproducibility in Hydrologic Modeling: Separating Model-Agnostic and Model-Specific Configuration Steps in Applications of Large-Domain Hydrologic Models

W. J. M. Knoben<sup>1</sup> , M. P. Clark<sup>1,2</sup> , J. Bales<sup>3</sup>, A. Bennett<sup>4</sup> , S. Gharari<sup>5</sup> , C. B. Marsh<sup>5</sup> , B. Nijssen<sup>6</sup> , A. Pietroniro<sup>7</sup> , R. J. Spiteri<sup>8</sup> , G. Tang<sup>1</sup> , D. G. Tarboton<sup>9</sup> , and A. W. Wood<sup>10</sup>

<sup>1</sup>Centre for Hydrology, University of Saskatchewan, Canmore, AB, Canada, <sup>2</sup>Department of Geography and Planning, University of Saskatchewan, Saskatoon, SK, Canada, <sup>3</sup>Consortium of Universities for the Advancement of Hydrologic Science, Inc, Cambridge, MA, USA, <sup>4</sup>Hydrology & Atmospheric Sciences, University of Arizona, Tucson, AZ, USA, <sup>5</sup>Centre for Hydrology, University of Saskatchewan, Saskatoon, SK, Canada, <sup>6</sup>Civil and Environmental Engineering, University of Washington, Seattle, WA, USA, <sup>7</sup>Schulich School of Engineering, Department of Civil Engineering, University of Calgary, Calgary, AB, Canada, <sup>8</sup>Department of Computer Science, University of Saskatchewan, Saskatoon, SK, Canada, <sup>9</sup>Utah Water Research Laboratory, Utah State University, Logan, UT, USA, <sup>10</sup>National Center for Atmospheric Research, Boulder, CO, USA

Knoben, W. J. M., Clark, M. P., Bales, J., Bennett, A., Gharari, S., Marsh, C. B., Nijssen, B., Pietroniro, A., Spiteri, R. J., Tang, G., Tarboton D. G. & Wood, A. W. (2022). Community Workflows to Advance Reproducibility in Hydrologic Modeling: Separating Model-Agnostic and Model-Specific Configuration Steps in Applications of Large-Domain Hydrologic Models. *Water Resources Research*, 58(11), e2021WR031753.

- Community Workflows to Advance Reproducibility in Hydrologic Modeling (CWARHM)
- Developing first generation of community hydrological modelling workflows
- Keeping workflows open source and reproducible
- Keeping workflows efficient for real-world applications
- Separating model-agnostic and model-specific configuration parts



UNIVERSITY OF  
CALGARY

# Recipe for Community Hydrological Modelling

manuscript submitted to *Environmental Modelling and Software*

## 1 Streamlining hydrological model applications by 2 enhancing model-agnostic and model-specific 3 configuration engines

4 Kasra Keshavarz<sup>1\*</sup>, Alain Pietroniro<sup>1</sup>, Mohamed Moghairib<sup>1</sup>, Wouter J. M.  
5 Knoben<sup>1</sup>, Paul Coderre<sup>1</sup>, Darri Eythorsson<sup>1</sup>, Shervan Gharari<sup>1</sup>, Martyn P.  
6 Clark<sup>1</sup>

7 <sup>1</sup>Department of Civil Engineering, Schulich School of Engineering, University of Calgary, Calgary, AB  
8 T2N 1N4, Canada

### 9 Key Points:

- 10 A generalized model-agnostic framework is proposed to streamline configurations  
11 of process-based hydrological models
- 12 The framework advances reproducibility in hydrological modelling by addressing  
13 complexities involved in handling dynamic and static datasets
- 14 Prototype applications in a high-performance computing environment demonstrate  
15 the effectiveness of the framework in reducing setup time

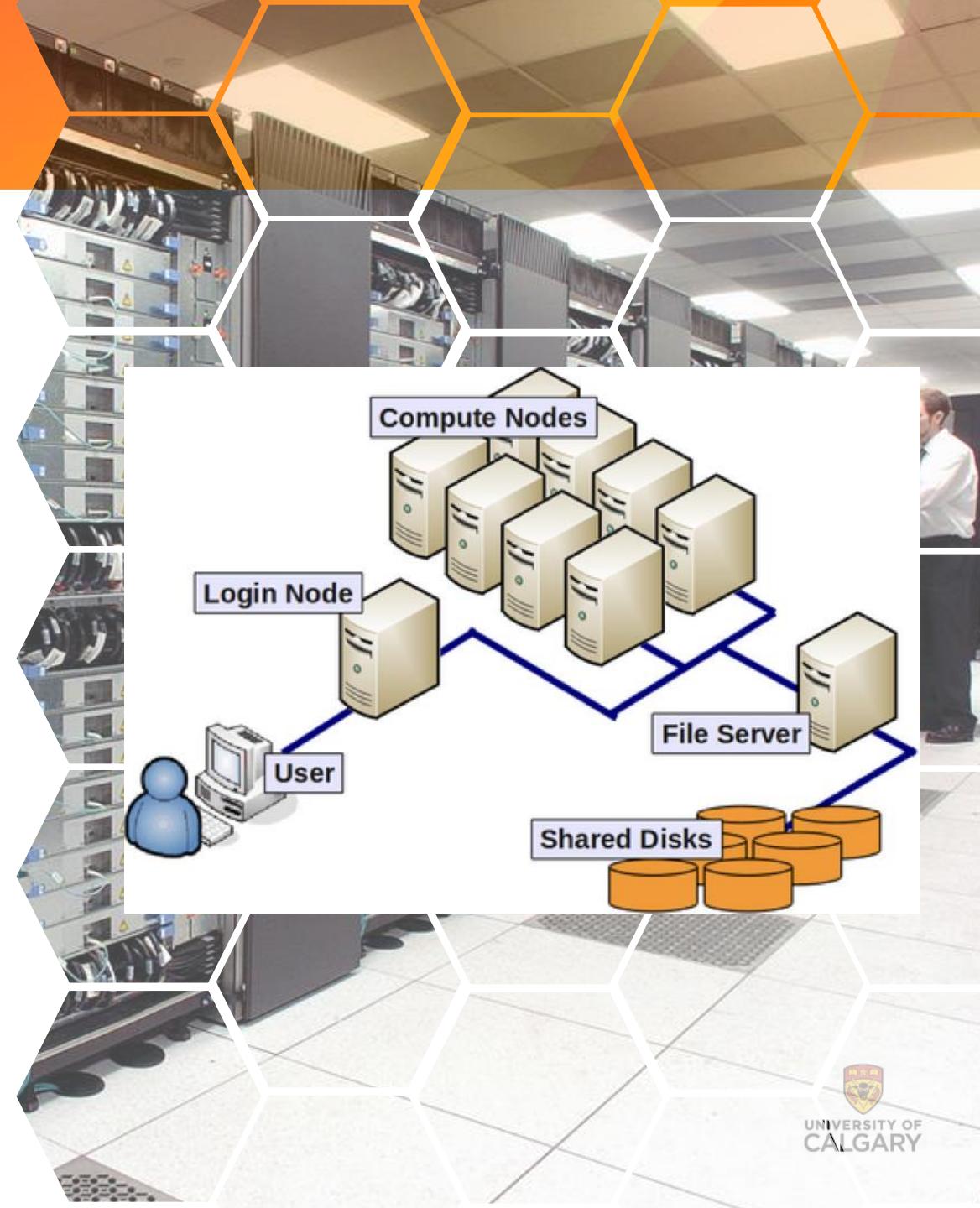
- Enhancing and proposing model-agnostic and model-specific engines for HPC environments
- Modularized methodology for the model-agnostic step for various data types and preprocessing needs
- Expanding model-specific workflows for other community modelling systems including ECCC MESH and SMHI HYPE
- Proposing strategies to objectively orchestrate the configuration stage of hydrological models

# Outline

- Community Hydrological Modelling Recipes
- Computational Resources for the Training Session
  - Launch a Jupyter session on Fir
  - Create a “collection of HPC modules”
  - Create a Python virtual environment
- Setting up ECCC’s National Water Model—MESH

# Computational Resources

- Wikipedia (accessed January 27<sup>th</sup>, 2025)  
*"High-performance computing (HPC) is the use of supercomputers and computer clusters to solve advanced problems."*
- The idea is to expand computational performance horizontally (multiple CPUs), instead of vertically (one massive CPU)
- High Performance Computing (HPC) systems provide multiple login and compute nodes for users



<https://top500.org/>

# Computational Resources



Our recommendation is to connect to HPCs using either of the following methods:

## jupyterlab

- Multi-user environment
- Resource efficient
- Collaborative environment
- Reproducibility mechanisms
- Strong community support
- Ease of use



```
$ ssh USERNAME@HPC_URL
```

- Maximal flexibility
- Remote Access
- Steep learning curve
- Full control of command sessions
- Strong community support



MobaXTerm



PuTTY



MacOS Terminal



UNIVERSITY OF  
CALGARY

# Computational Resources

How can we launch a Jupyter session on HPCs? A few examples...



Anvil at Purdue University



ARC at the University of Calgary



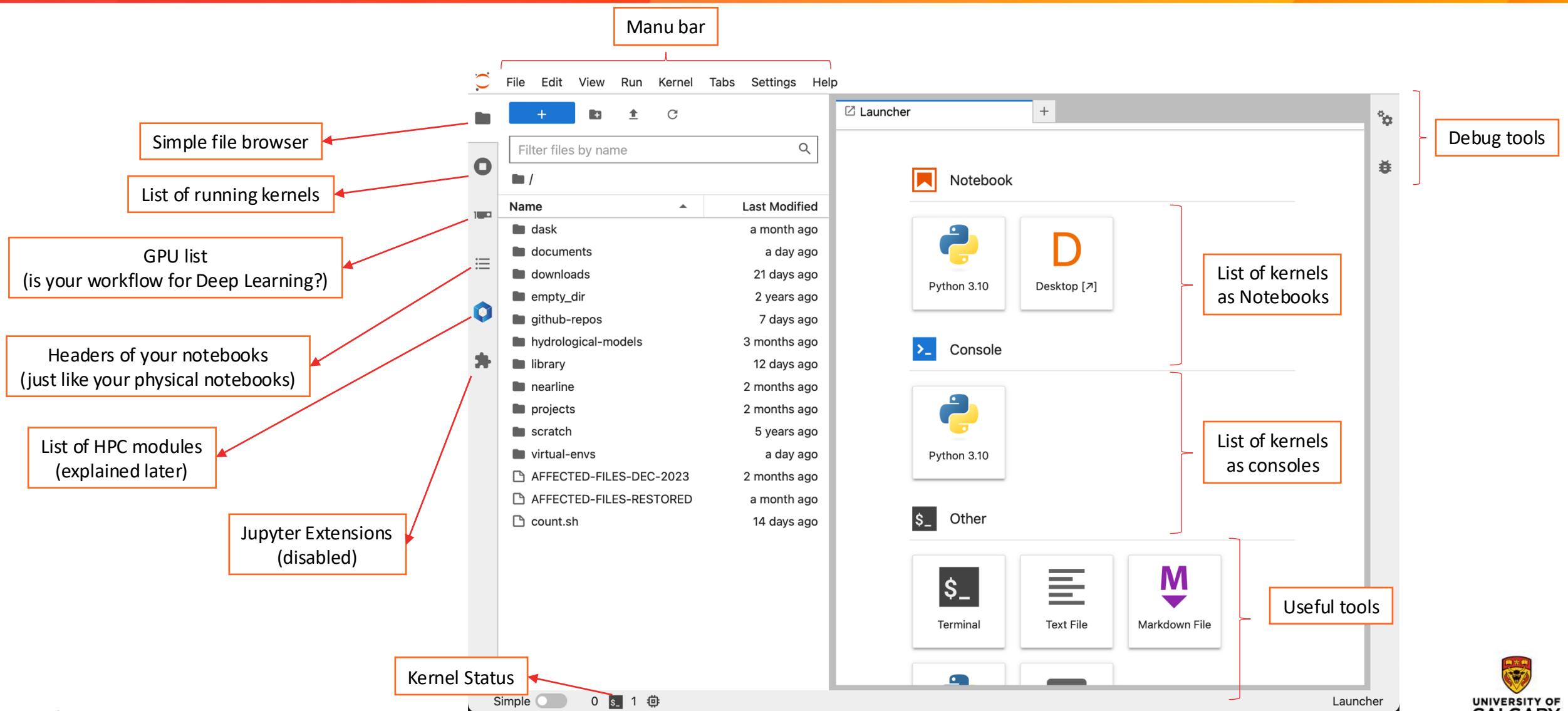
Any questions so far?

Let's try launching a Jupyter session!

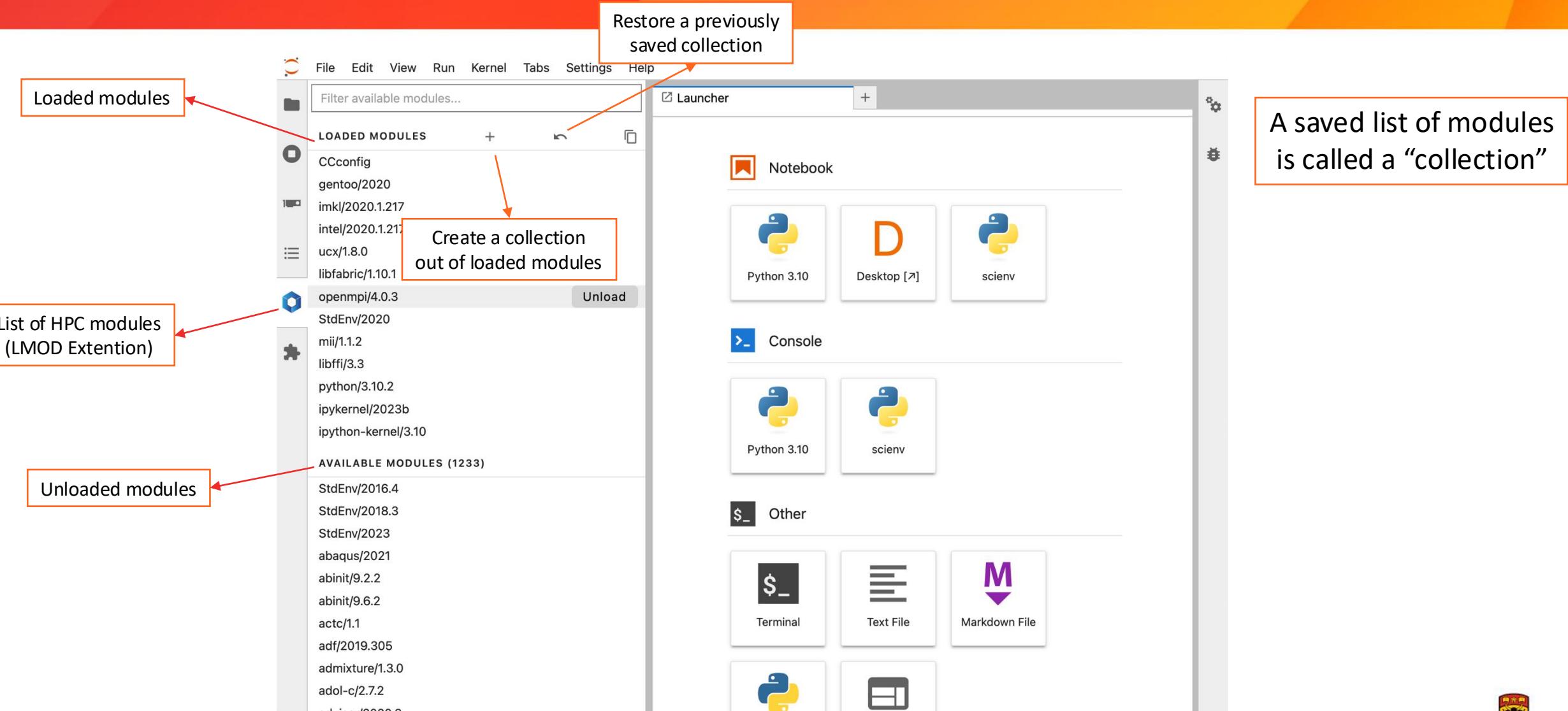
Use the following link in your browser:

<https://jupyterhub.fir.alliancecan.ca>

# Computational Resources



# Computational Resources



Any questions so far?

Let's learn about HPC modules!



# Computational Resources



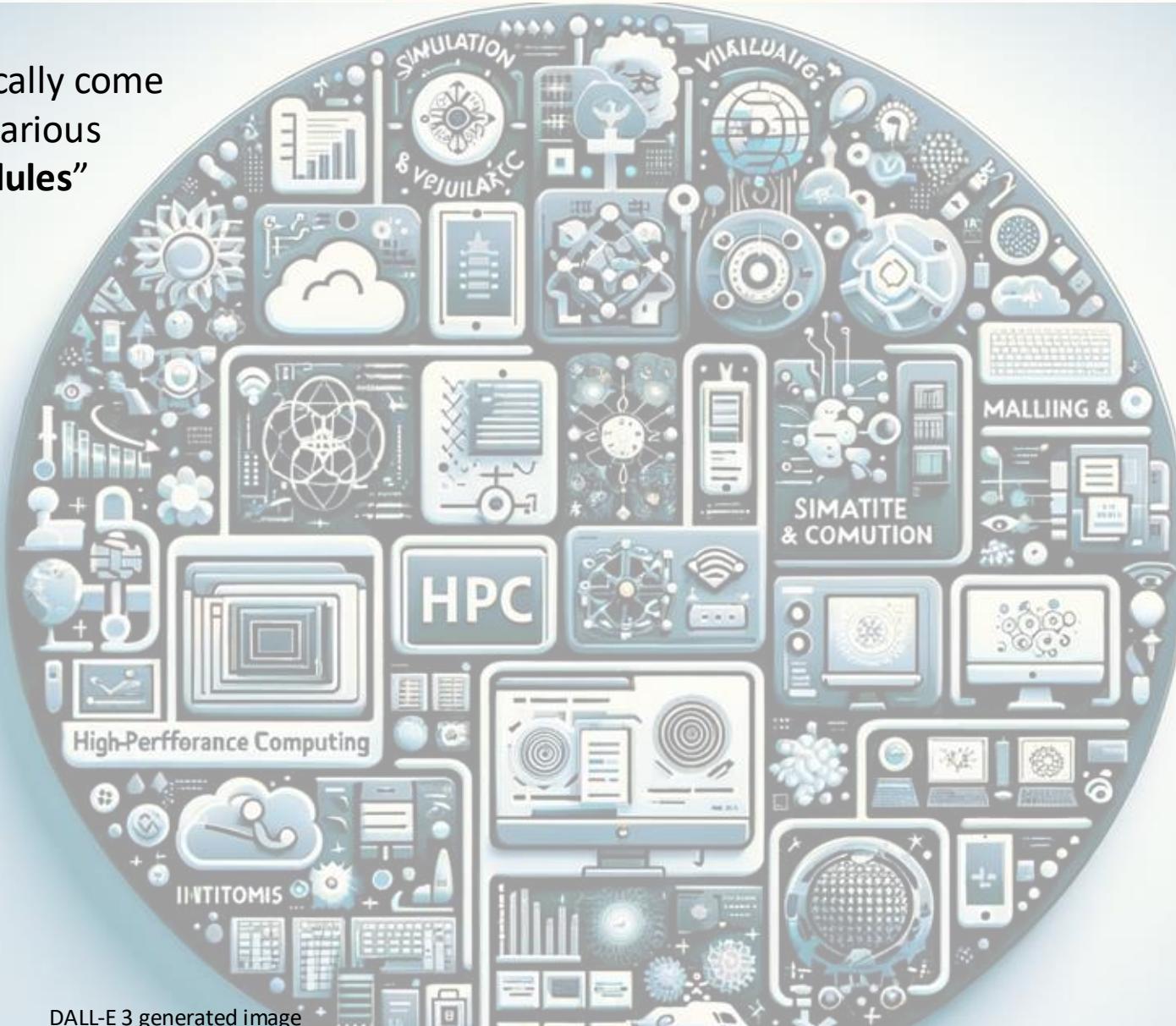
HPCs typically come with various “modules”

## Flexibility

Different users have different requirements

## Customization

Users may need specific versions of software or libraries

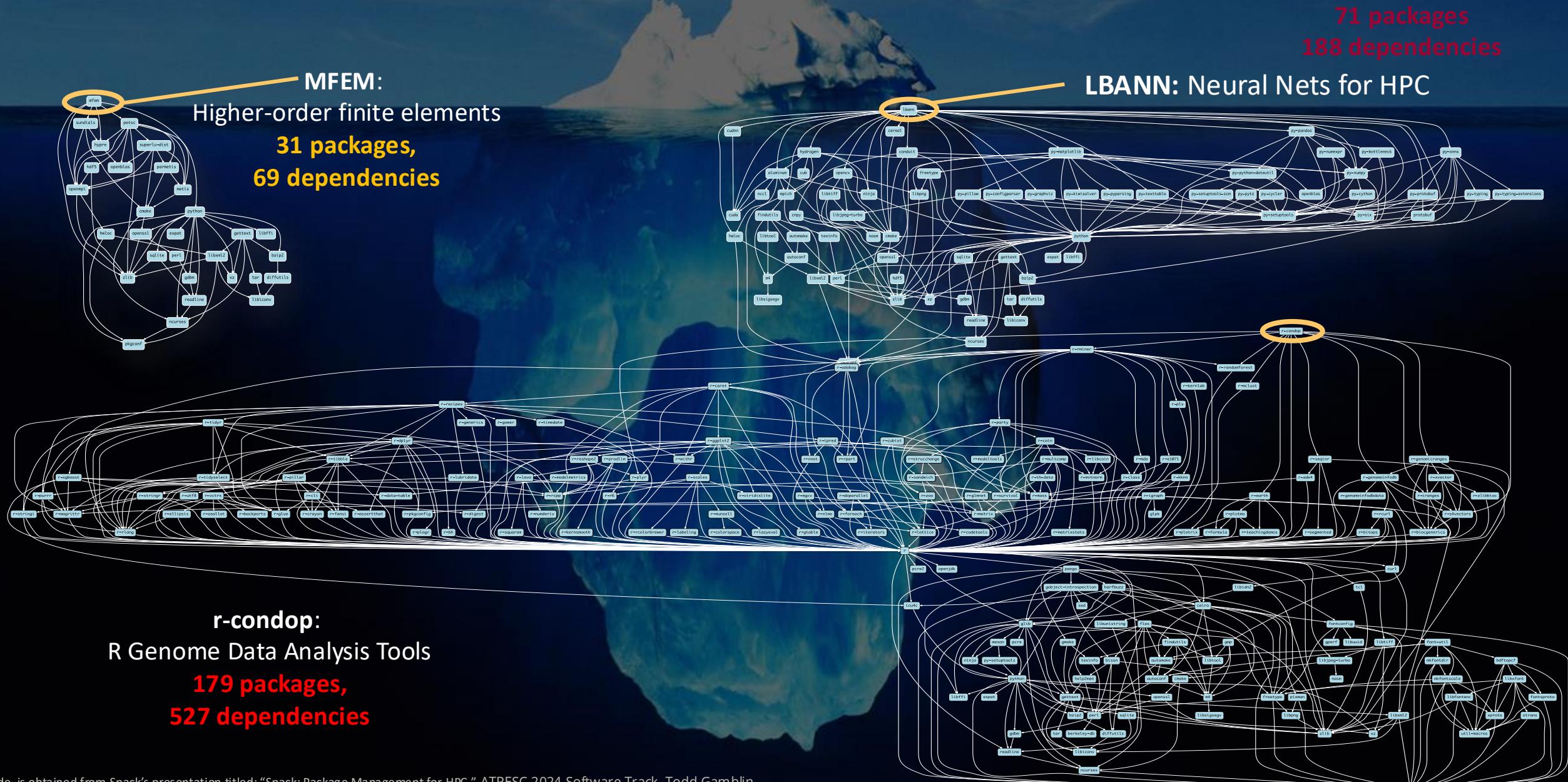


Optimization software packages are often optimized for the underlying hardware architecture

## Productivity

Having commonly used software readily available, saves users time and effort

# Modern scientific codes rely on icebergs of dependency libraries



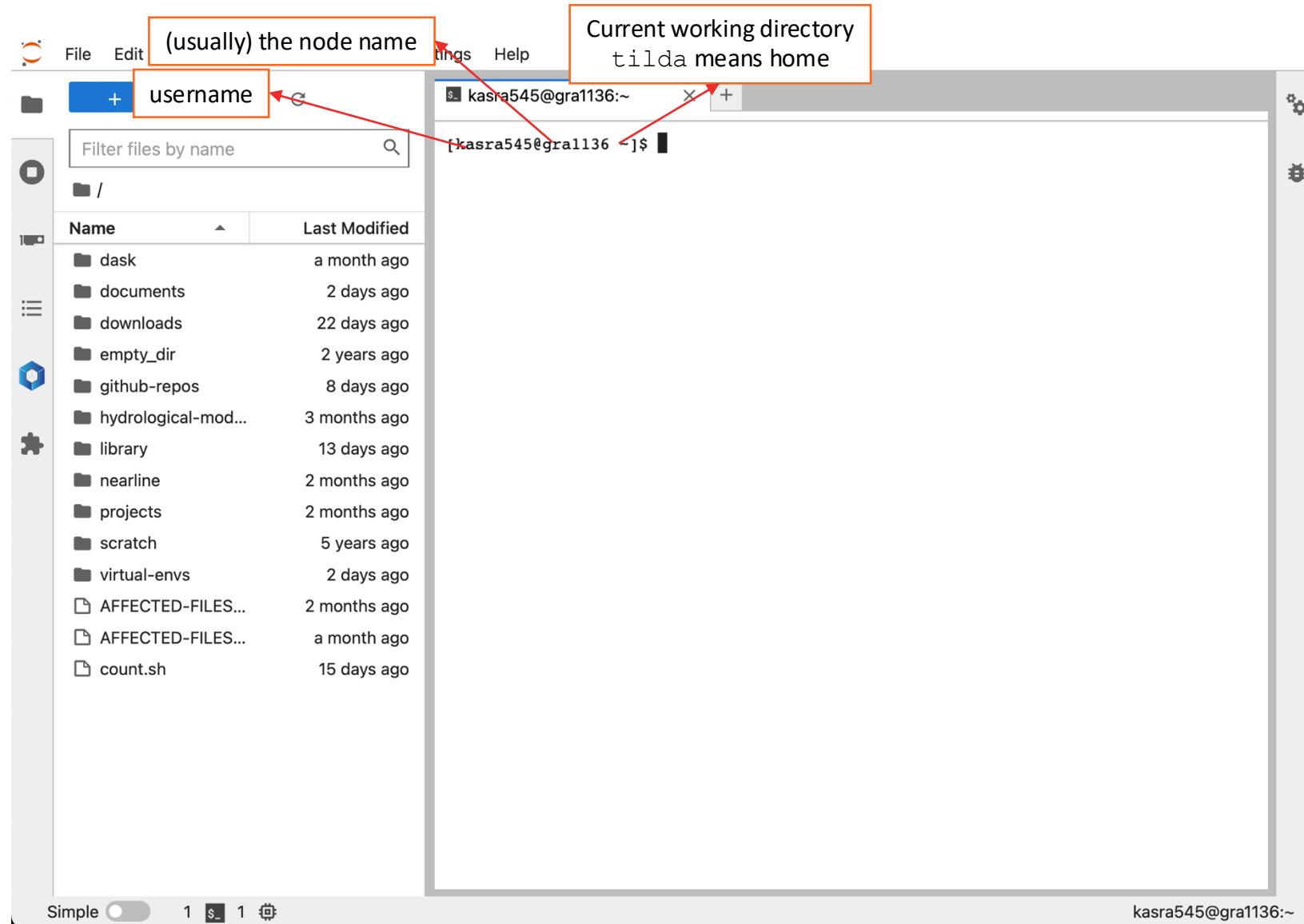
## Any questions so far?

Let's go back to our Jupyter session and get some hands-on experience with HPC modules!

# Computational Resources

All instructions are available  
on this presentation's  
repository:

<https://github.com/kasra-keshavarz/maf-training>



# Computational Resources

Exactly similar to what we can do in this tab

All instructions are available on this presentation's repository:

<https://github.com/kasra-keshavarz/maf-training>

The screenshot shows a terminal window with several annotations:

- Computing node name:** Points to the text "kasra.keshavarz1@fc1" in the top right corner.
- Current working directory:** Points to the text "tilda means home" in the top right corner.
- Getting a list of available modules:** Points to the command "module avail | less".
- Loading every module we need:** Points to the command "module load".
- Double checking whatever I have loaded:** Points to the command "ml list".

**Module Available List:**

```
[kasra.keshavarz1@fc1 ~]$ module avail | less
[kasra.keshavarz1@fc1 ~]$ module load \
> gcc/14.2.0 htop/3.3.0 glibc/2.28 libaec/1.0.6 \
> gcc-runtime/14.2.0 hdf5/1.14.3 openssl/3.3.1 lz4/1.9.4 \
> libevent/2.1.12 snappy/1.1.10 numactl/2.0.14 c-blosc/1.21.5 \
> opa-psm2/11.2.230 netcdf-c/4.9.2 krb5/1.21.2 \
> netcdf-fortran/4.6.1 libedit/3.1-20230828 openjpeg/2.3.1 \
> libxcrypt/4.4.35 eccodes/2.34.0 openssh/9.8p1 \
> fftw/3.3.10 ucx/1.17.0 libunistring/1.2 \
> openmpi/4.1.6 libidn2/2.3.7 sqlite/3.46.0 \
> nghttp2/1.62.0 libmd/1.0.4 curl/8.7.1 \
> libbsd/0.12.2 libjpeg-turbo/3.0.3 expat/2.6.2 \
> libtiff/4.6.0 libffi/3.4.6 proj/9.4.1 \
> util-linux-uuid/2.40.2 cdo/2.4.3 python/3.11.7 \
> antlr/2.7.7 python-venv/1.0 gs1/2.7.1 \
> py-numpy/1.26.4 nco/5.2.4 gdal/3.9.2 \
> tree/2.1.0 geos/3.12.2 which/2.21 \
> udunits/2.2.28 r/4.4.1 slurm/24.11.0-1 \
> py-mpi4py/4.0.0 qt/5.15.14;
[kasra.keshavarz1@fc1 ~]$ ml list
```

**Currently Loaded Modules:**

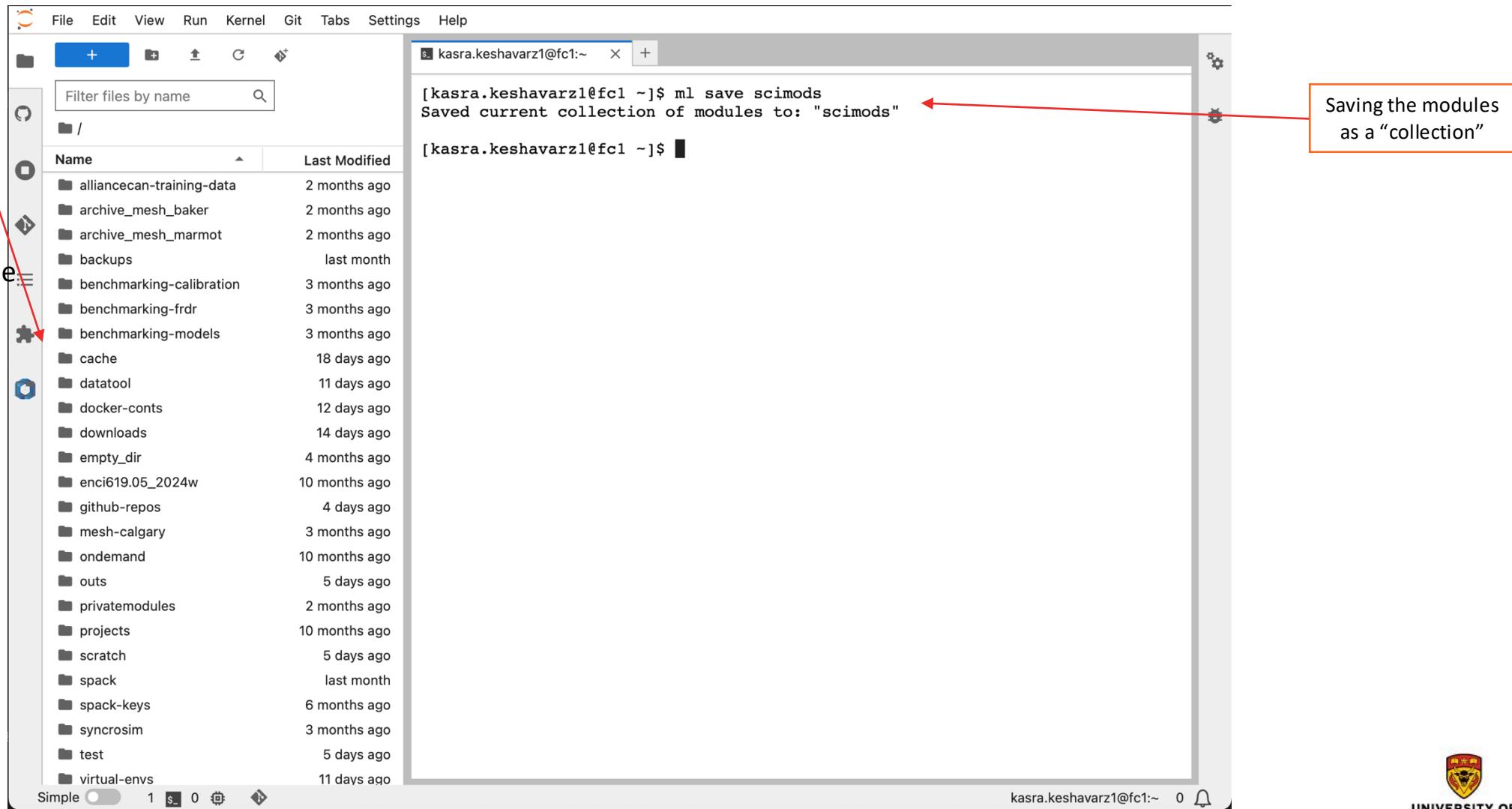
```
1) gcc/14.2.0          28) nghttp2/1.62.0
2) htop/3.3.0           29) libmd/1.0.4
3) glibc/2.28            30) curl/8.7.1
4) libaec/1.0.6          31) libbsd/0.12.2
5) gcc-runtime/14.2.0     32) libjpeg-turbo/3.0.3
6) hdf5/1.14.3           33) expat/2.6.2
7) openssl/3.3.1          34) libtiff/4.6.0
8) lz4/1.9.4              35) libffi/3.4.6
9) libevent/2.1.12         36) proj/9.4.1
10) snappy/1.1.10         37) util-linux-uuid/2.40.2
11) numactl/2.0.14          38) cdo/2.4.3
12) c-blosc/1.21.5        39) python/3.11.7
13) opa-psm2/11.2.230      40) antlr/2.7.7
14) netcdf-c/4.9.2          41) python-venv/1.0
15) krb5/1.21.2             42) gs1/2.7.1
```

# Computational Resources

Exactly similar to what we can do in this tab

All instructions are available on this presentation's repository:

<https://github.com/kasra-keshavarz/maf-training>



The screenshot shows a terminal window with the following content:

```
[kasra.keshavarz1@fc1:~]$ ml save scimods
Saved current collection of modules to: "scimods"
```

A red arrow points from the text "Exactly similar to what we can do in this tab" to the terminal window. Another red arrow points from the text "Saving the modules as a ‘collection’" to the word "scimods".

The terminal window also displays a file list on the left side:

Name	Last Modified
alliancecan-training-data	2 months ago
archive_mesh_baker	2 months ago
archive_mesh_marmot	2 months ago
backups	last month
benchmarking-calibration	3 months ago
benchmarking-frdr	3 months ago
benchmarking-models	3 months ago
cache	18 days ago
datatool	11 days ago
docker-conts	12 days ago
downloads	14 days ago
empty_dir	4 months ago
enci619.05_2024w	10 months ago
github-repos	4 days ago
mesh-calgary	3 months ago
ondemand	10 months ago
outs	5 days ago
privatemodules	2 months ago
projects	10 months ago
scratch	5 days ago
spack	last month
spack-keys	6 months ago
syncrosim	3 months ago
test	5 days ago
virtual-envs	11 days ago

# Computational Resources

Refresh



the JupyterLab session webpage, and see if you can find the collection you just saved:

The screenshot shows the JupyterLab interface. On the left, there's a sidebar titled "LOADED MODULES" with a list of modules. A red box highlights the text "Check out what module collections you have". A red arrow points from this box to the "Filter available modules..." input field at the top of the sidebar. Another red box highlights the text "Wait for a few seconds so Jupyter loads all modules saved in 'scimods' collection". A red arrow points from this box to the "scimods" entry in the loaded modules list. In the main area, a terminal window shows the command "[kasra.keshavarz1@fc1 ~]\$ ml save scimods" and its output "Saved current collection of modules to: 'scimods'". A modal dialog box titled "Restore collection" has a dropdown menu set to "scimods". A red box highlights the text "Restore the 'scimods' (science modules)" and a red arrow points from this box to the "Restore" button in the dialog. The bottom status bar shows the user's name "kasra.keshavarz1@fc1:" and a small bell icon.

Check out what module collections you have

Wait for a few seconds so Jupyter loads all modules saved in "scimods" collection

Restore the "scimods" (science modules)

File Edit View Run Kernel Git Tabs Settings Help

Filter available modules...

LOADED MODULES

glibc/2.28  
gcc-runtime/14.2.0  
libmd/1.0.4  
libbsd/0.12.2  
expat/2.6.2  
libffi/3.4.6  
libcrypt/4.4.35  
openssl/3.3.1  
sqlite/3.46.0  
util-linux-uuid/2.40.2  
python/3.11.7  
python-venv/1.0  
py-async-lru/1.0.3  
py-editables/0.5  
py-packaging/23.1  
py-pathspec/0.11.1  
py-pluggy/1.5.0  
py-trove-classifiers/2023.8.7  
py-hatchling/1.21.0  
py-hatch-jupyter-builder/0.8.3  
py-traitlets/5.14.3  
py-comm/0.1.4  
py-debugpy/1.6.7  
py-decorator/5.1.1  
py-parso/0.8.3  
py-setuptools/69.2.0  
py-jedi/0.18.2

[kasra.keshavarz1@fc1 ~]\$ ml save scimods  
Saved current collection of modules to: "scimods"

[kasra.keshavarz1@fc1 ~]\$

Restore collection

Select a collection to restore : scimods

Cancel Restore

Simple 1 \$ 0 0 0 kasra.keshavarz1@fc1:~ 0

This is only a one-time process.  
You can always access all the collections you saved from Jupyter's module extension tab

Any questions so far?

We can now proceed with creating a Python virtual environment

# Computational Resources



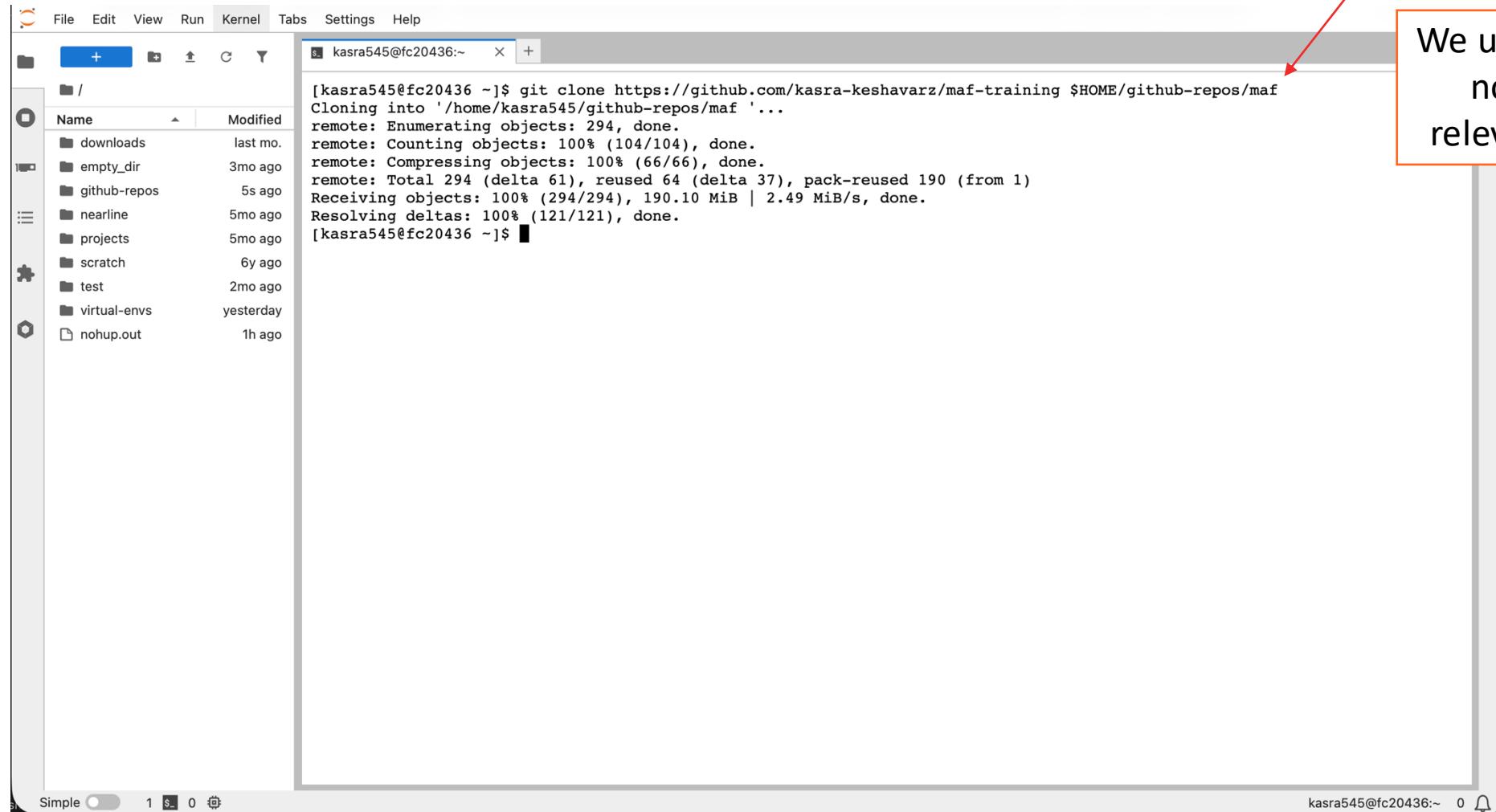
## Python Programming Language

- MAF: much of our workflows are offered in Python package formats
- Python is Powerful: Python always can offer you a useful workflow
- Reproducible Environments: Python's **Virtual Environments** enables this requirement
- Supported on HPCs and Commercial Cloud regardless of your Operating System: it can be used everywhere
- Fir is no exception: Python is fully supported on many HPCs including Fir, by default.

# Computational Resources

Before setting up the Python Environment, we need to clone (download) the training session's GitHub repository

# Computational Resources



The screenshot shows a Jupyter Notebook interface with a terminal tab open. The terminal window displays the command `git clone https://github.com/kasra-keshavarz/maf-training $HOME/github-repos/maf` being run, followed by the output of the cloning process, which includes object enumeration, counting, compressing, receiving objects, and resolving deltas. The terminal prompt ends with `[kasra545@fc20436 ~]$`. To the left of the terminal is a file browser pane showing a directory structure with various folders like `downloads`, `empty_dir`, and `github-repos`.

“clone”-ing the repository

We use this terminal session now to download the relevant GitHub repository

# Computational Resources

Now we can set up a virtual environment using the list of packages described in the training session's repository

Let's go back to our Jupyter session!

# Computational Resources

The screenshot shows a terminal window titled "kasra.keshavarz1@fc1:~" with the following command history:

```
[kasra.keshavarz1@fc1:~]$ python -m venv $HOME/virtual-envs/scienv
[kasra.keshavarz1@fc1 ~]$ source $HOME/virtual-envs/scienv/bin/activate
```

A red arrow points from the text "Creating an empty Virtual Environment" to the first command in the terminal. Another red arrow points from the text "Activating the empty Virtual Environment" to the second command.

On the left, there is a file browser interface showing a list of files and directories in the current directory (~). The list includes:

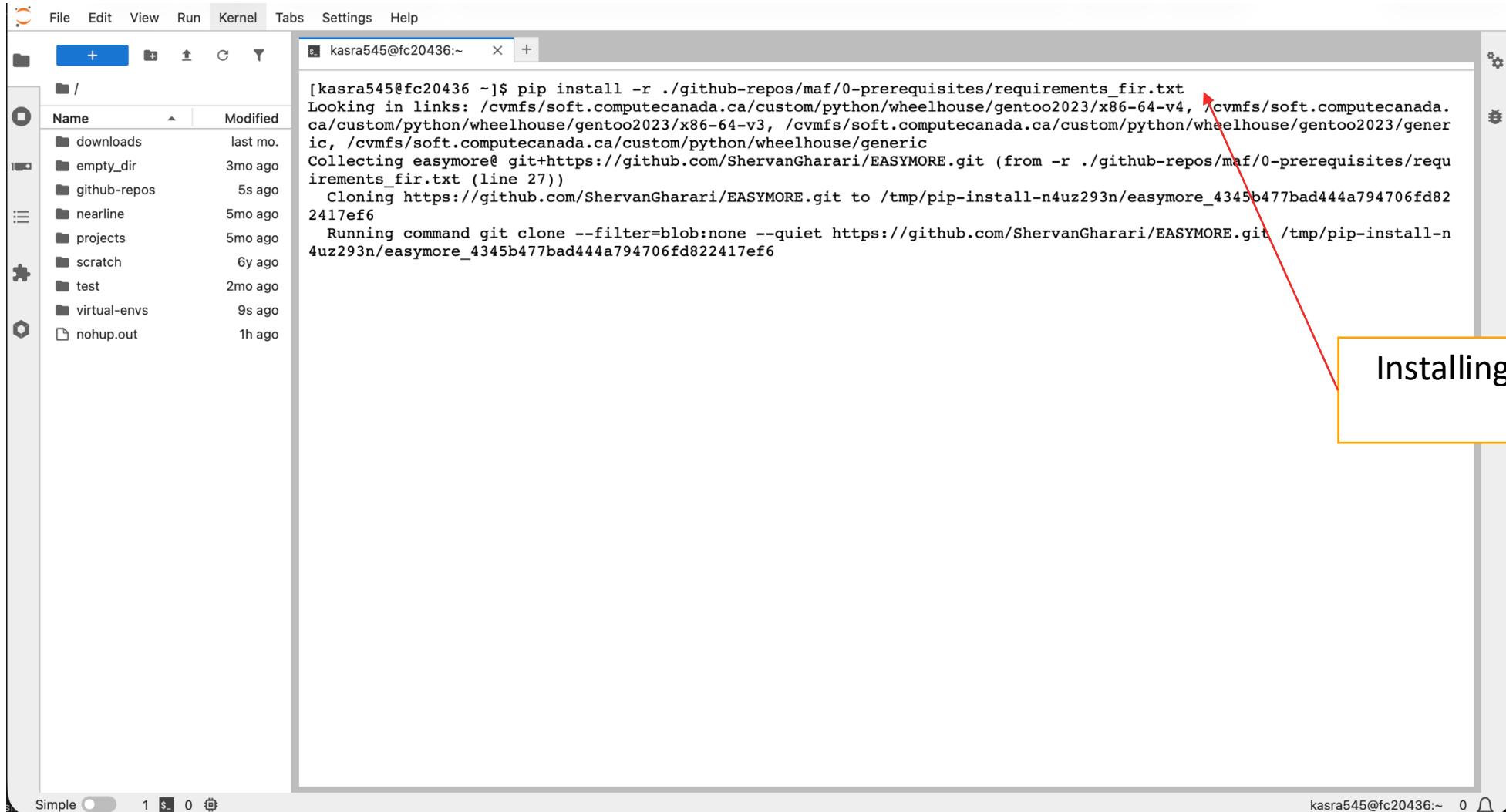
Name	Last Modified
benchmarking-calibration	3 months ago
benchmarking-frdr	3 months ago
benchmarking-models	3 months ago
cache	18 days ago
datatool	11 days ago
docker-conts	12 days ago
downloads	14 days ago
empty_dir	4 months ago
enci619.05_2024w	10 months ago
github-repos	1 minute ago
mesh-calgary	3 months ago
ondemand	10 months ago
outs	5 days ago
privatemodules	2 months ago
projects	10 months ago
scratch	5 days ago
spack	last month
spack-keys	6 months ago
syncrosim	3 months ago
test	5 days ago
virtual-envs	11 days ago
mesh-calgary.tar.gz	2 months ago
renv_1.0.11.tar.gz	4 months ago
spack-build-out.txt	12 days ago
terra_1.8-10.tar.gz	7 days ago

At the bottom of the terminal window, the prompt "kasra.keshavarz1@fc1:~" is visible along with a bell icon and a status bar showing "Simple 1 \$ 0".

Creating an empty  
Virtual Environment

Activating the empty  
Virtual Environment

# Computational Resources



```
[kasra545@fc20436 ~]$ pip install -r ./github-repos/maf/0-prerequisites/requirements_fir.txt
Looking in links: /cvmfs/soft.computeCanada.ca/custom/python/wheelhouse/gentoo2023/x86-64-v4, /cvmfs/soft.computeCanada.ca/custom/python/wheelhouse/gentoo2023/x86-64-v3, /cvmfs/soft.computeCanada.ca/custom/python/wheelhouse/gentoo2023/generic, /cvmfs/soft.computeCanada.ca/custom/python/wheelhouse/generic
Collecting easymore@ git+https://github.com/ShervanGharari/EASYMORE.git (from -r ./github-repos/maf/0-prerequisites/requirements_fir.txt (line 27))
  Cloning https://github.com/ShervanGharari/EASYMORE.git to /tmp/pip-install-n4uz293n/easymore_4345b477bad444a794706fd82417ef6
    Running command git clone --filter=blob:none --quiet https://github.com/ShervanGharari/EASYMORE.git /tmp/pip-install-n4uz293n/easymore_4345b477bad444a794706fd822417ef6
```

Installing necessary Python  
Packages

# Computational Resources

The screenshot shows the JupyterLab interface. On the left, there's a sidebar with icons for file operations (New, Open, Save, etc.) and a search bar labeled "Filter available modules...". Below that is a list of "LOADED MODULES" which includes:

- CCconfig
- gentoo/2020
- imkl/2020.1.217
- intel/2020.1.217
- ucx/1.8.0
- libfabric/1.10.1
- openmpi/4.0.3
- StdEnv/2020
- mii/1.1.2
- libffi/3.3
- python/3.10.2
- ipykernel/2023b
- ipython-kernel/3.10

At the bottom of the sidebar, it says "AVAILABLE MODULES (1233)" followed by a list of three more items:

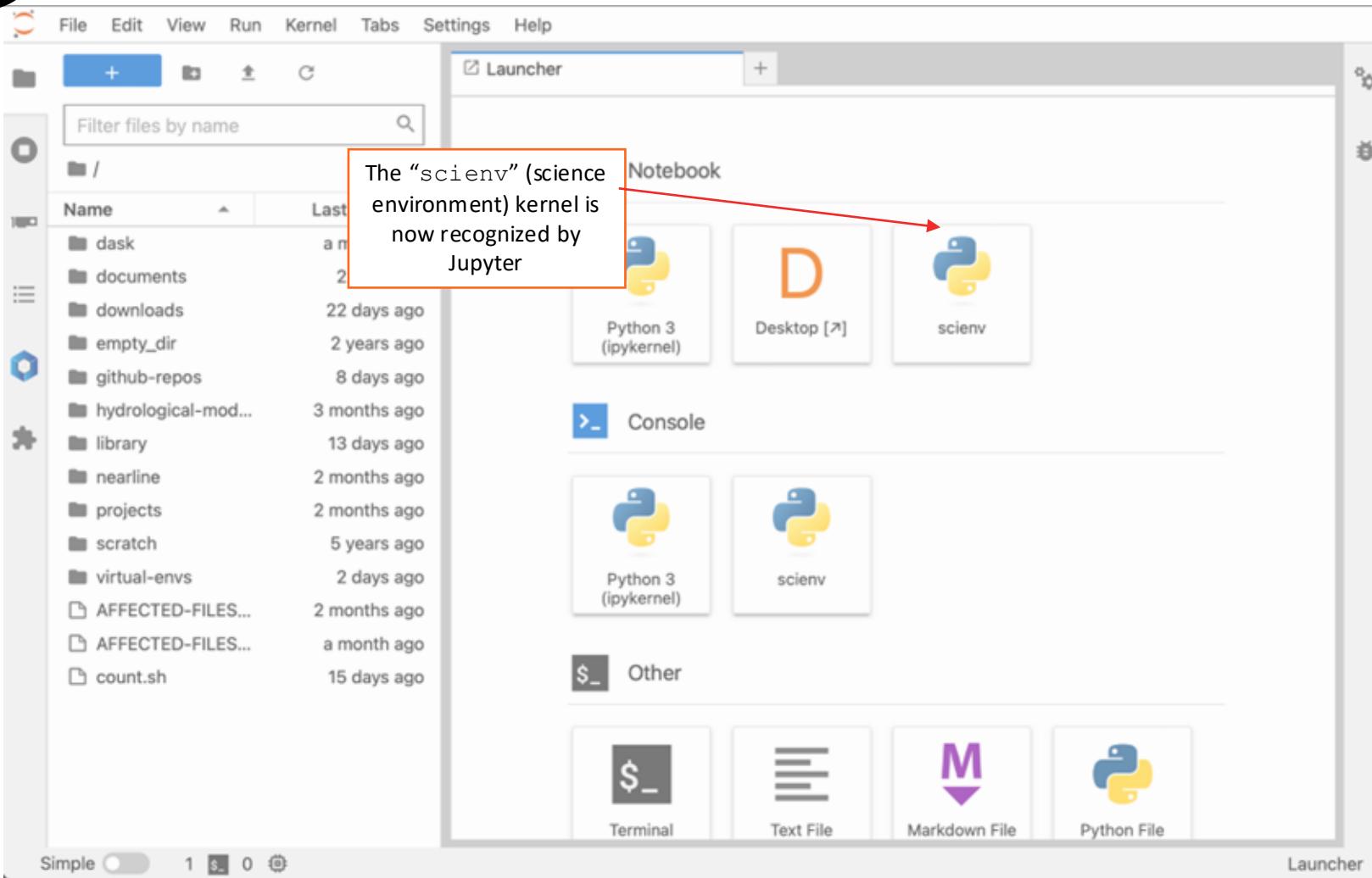
- StdEnv/2016.4
- StdEnv/2018.3
- StdEnv/2023

On the right, there's a terminal window titled "kasra545@gra-login1:~". It shows the command "python -m ipykernel install --user --name "scienv"" being run, followed by the output "Installed kernelspec scienv in /home/kasra545/.local/share/jupyter/kernels/scienv".

A callout box with a yellow border contains the text: "“scienv” virtual environment is now accessible through JupyterLab".

# Computational Resources

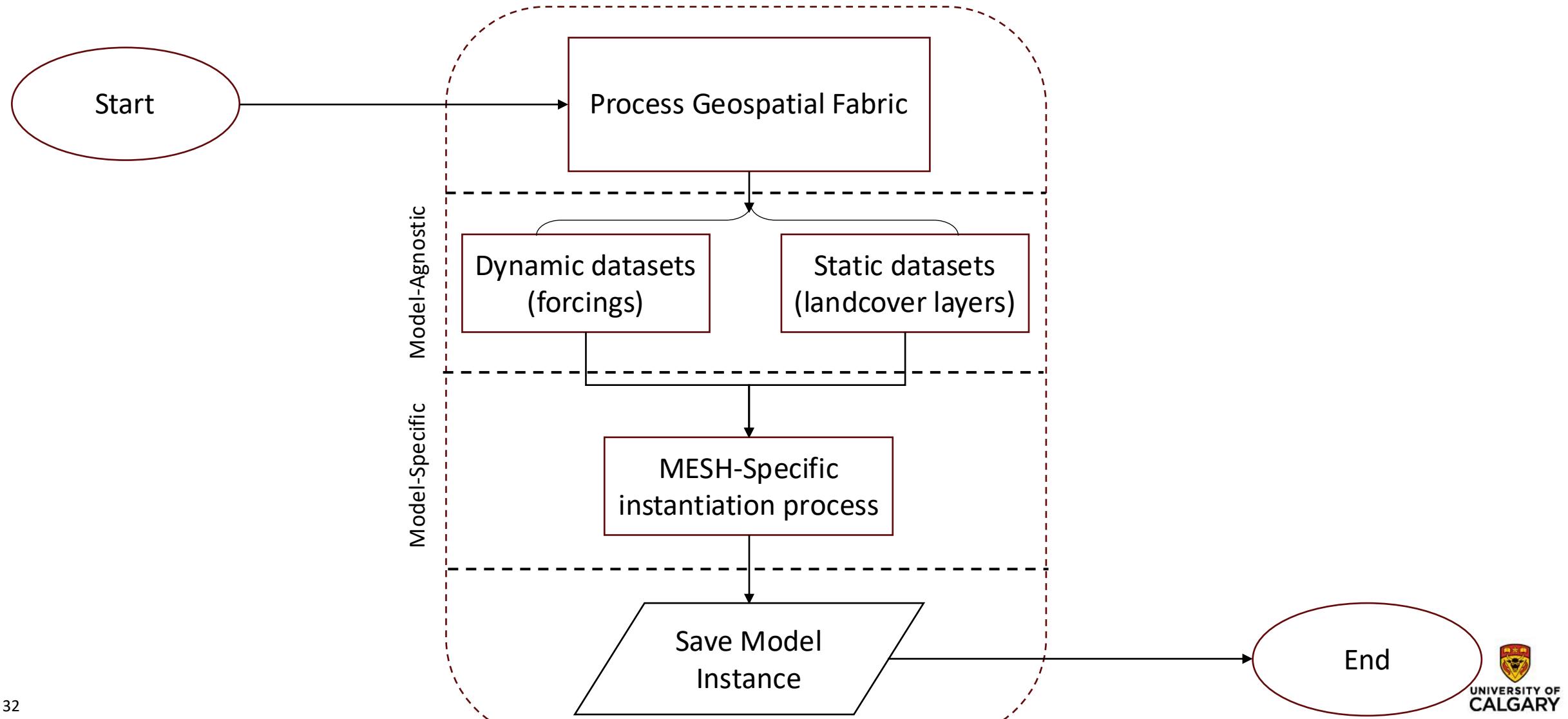
Refresh  the JupyterLab session webpage, and see if you can find the collection you just saved:



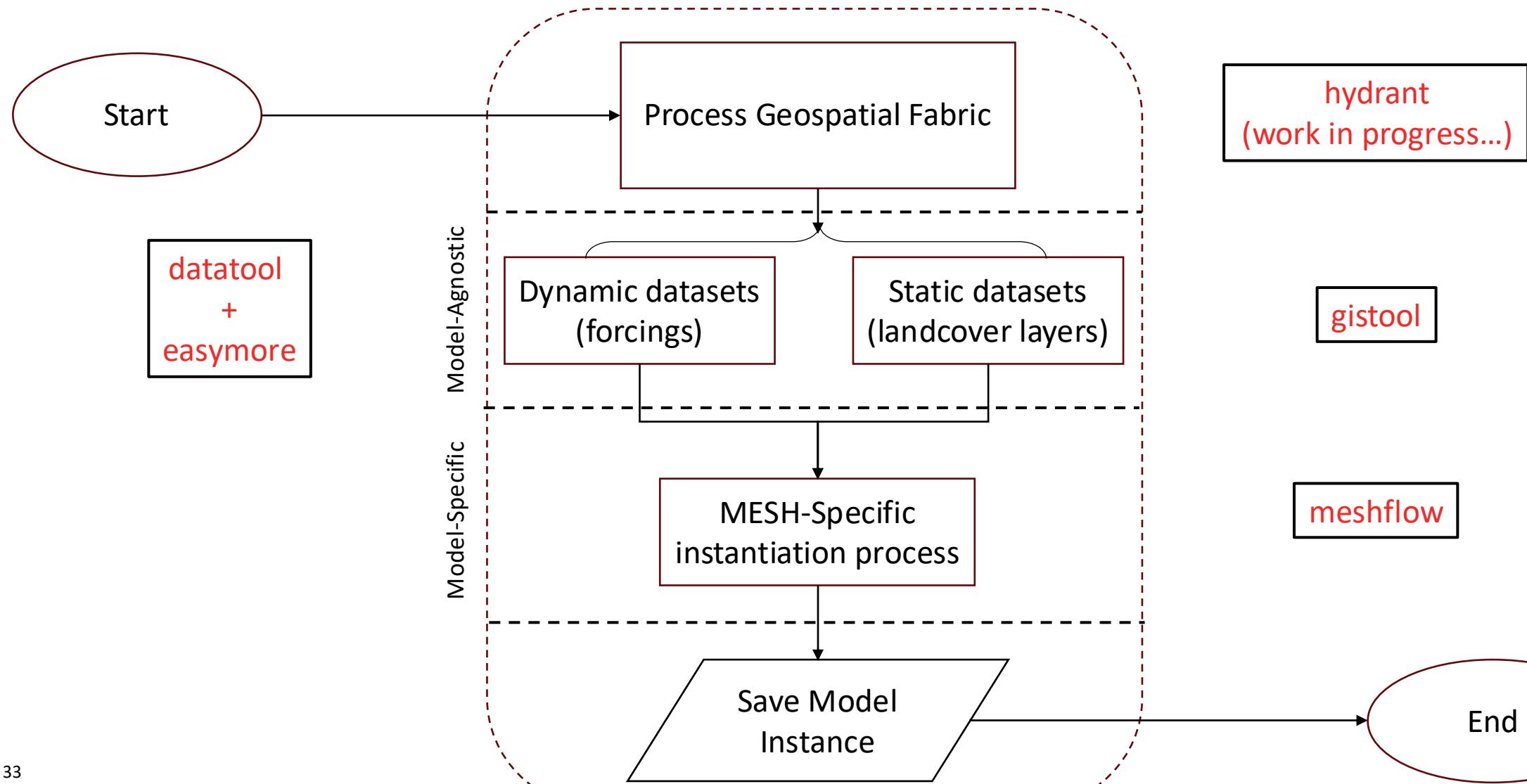
# Outline

- Community Hydrological Modelling Recipes
- Computational Resources for the Training Session
- Setting up ECCC's National Water Model—MESH
  - General overview of steps
  - Introducing tools for each step
  - Running the steps together for the MESH instance

# Setting up ECCC's National Water Model—MESH



# Setting up ECCC's National Water Model—MESH



# Setting up ECCC's National Water Model—MESH

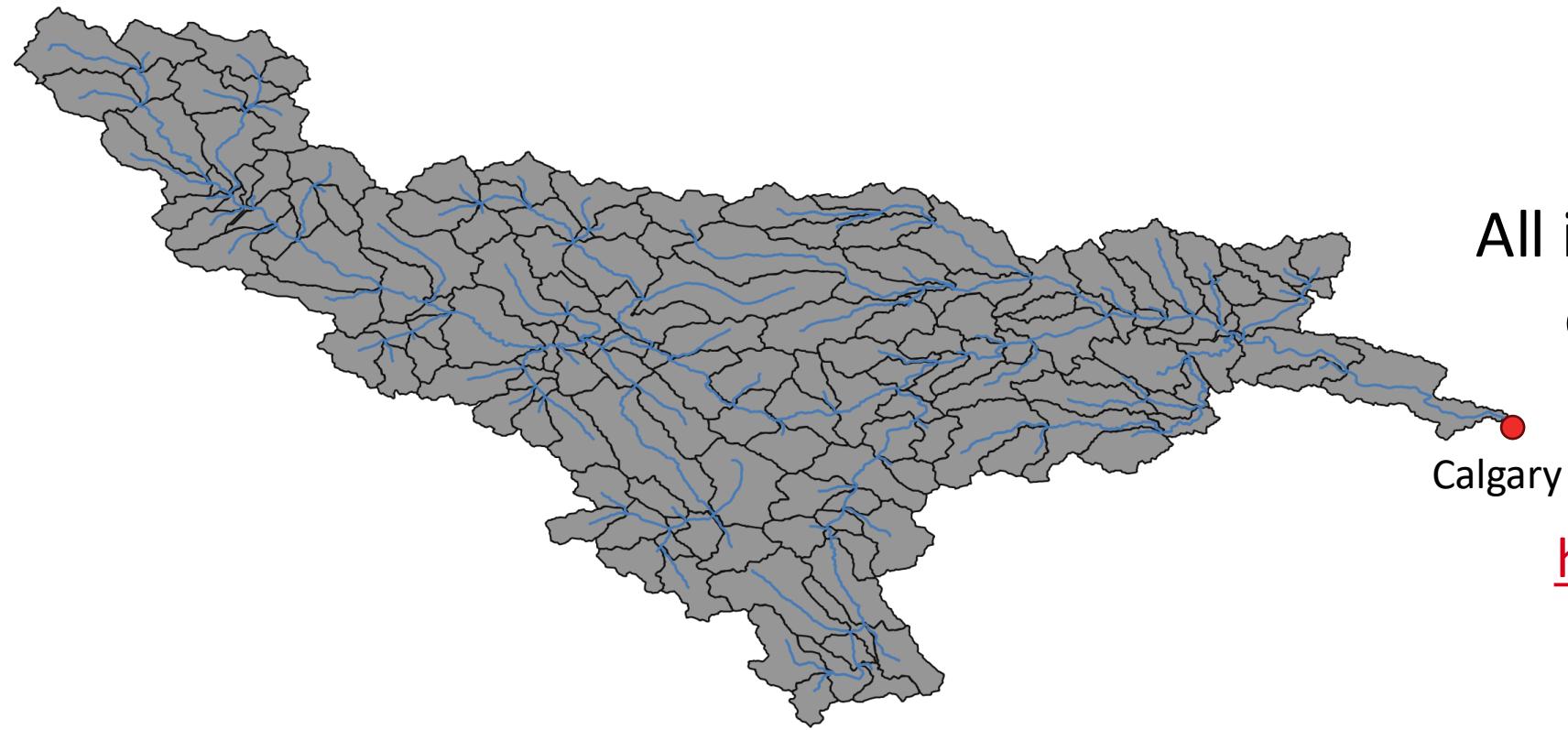
Configuring ECCC's MESH with the following datasets for the Bow River at Calgary:

Meteorological Forcing Dataset	Landcover Dataset	Geospatial Fabric Dataset	Study Area
Canadian Surface Reanalysis version 3.2 (CaSRv3.2)	North American Land Change Monitoring System 2020 (NALCMS 2020)	MERIT-Basins	Bow River at Calgary

# Setting up ECCC's National Water Model—MESH



# Setting up ECCC's National Water Model—MESH



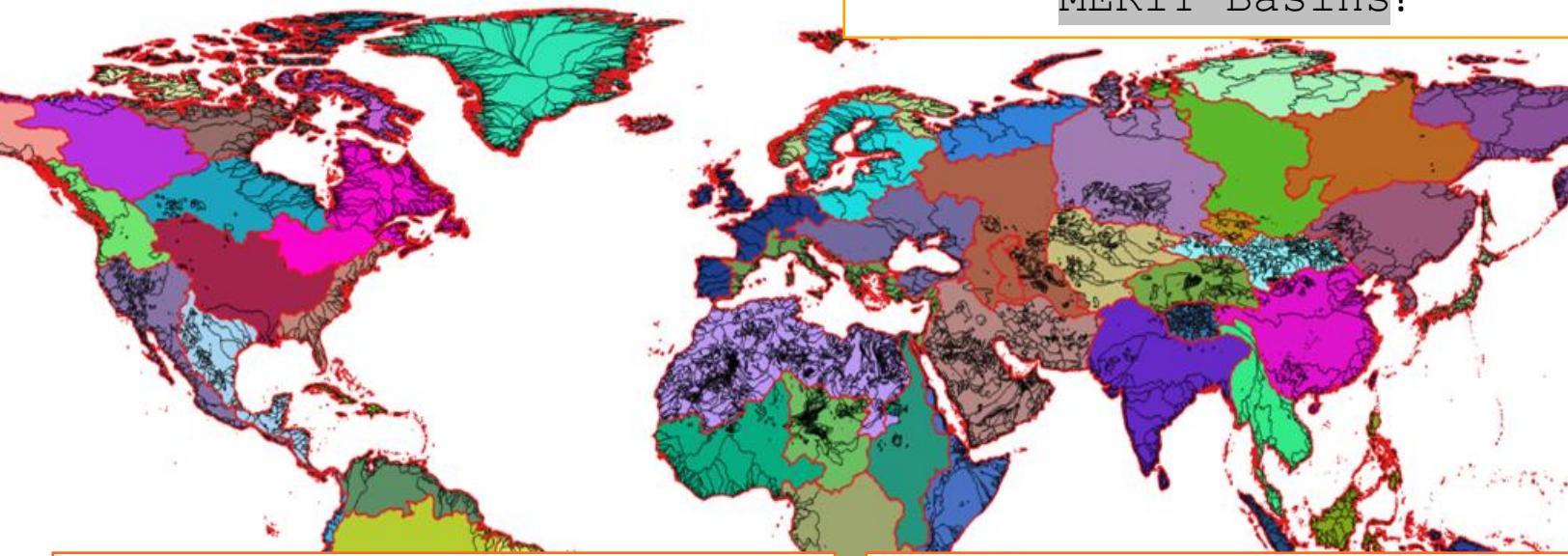
Bow River at Calgary

Sub-basins and river network extracted from MERIT-Basins  
using Hydrant v0.1.0-dev0

All instructions are available  
on this presentation's  
repository:

<https://github.com/kasra-keshavarz/maf-training>

# Setting up ECCC's National Water Model—MESH



What information is provided through  
MERIT-Basins?

- 25 km<sup>2</sup> drainage area threshold for channels
- Corrected/refined basin/region definitions
- Below 25 km<sup>2</sup> non-channelized areas along the coast or some endorheic areas (incomplete catchments or hillslopes) excluded
- ~2.94 million river reaches (unit catchments)

## River network:

- River segment IDs (`COMID`)
- Downstream segment IDs (`NextDownID`)
- Slope (`slope`)
- Length (`lengthkm`)
- Upstream segment IDs (`up1-4`)
- Upstream drainage area (`uparea`)
- River geometries

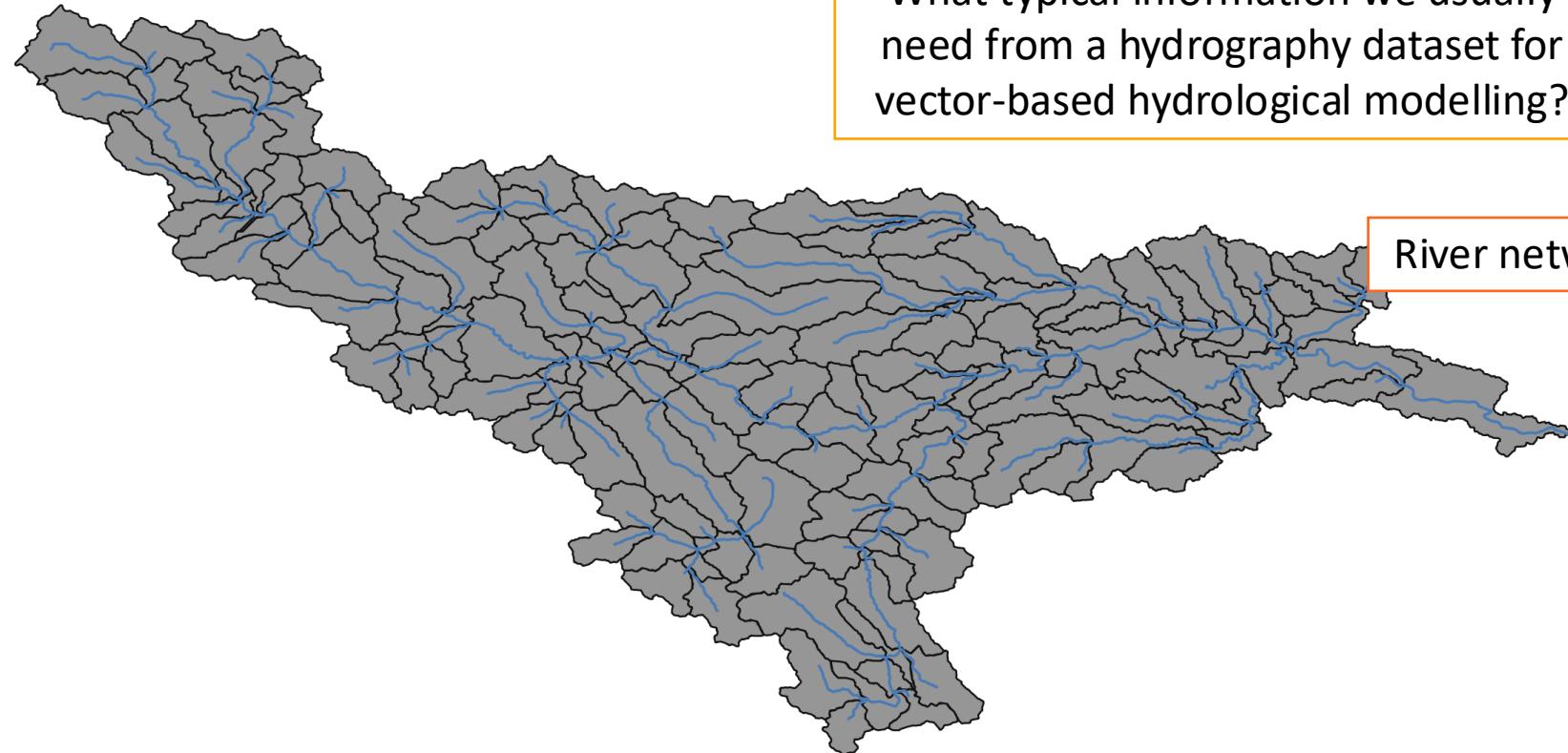
## Subbasins:

- Identical subbasin IDs (`COMID`)
- Subbasin area (`unitarea`)
- Subbasin geometries

## Non-contributing areas:

- Non-contributing area IDs (`FID`)
- Non-contributing area geometries

# Setting up ECCC's National Water Model—MESH



Bow River at Calgary

Sub-basins and river network extracted from MERIT-Basins  
using Hydrant v0.1.0-dev0

What typical information we usually need from a hydrography dataset for vector-based hydrological modelling?

River network:

- River segment ID values
- Immediate downstream segment ID values
- Length

Depending on your routing scheme you may need:

- Slope [optional]
- Width [optional]
- etc.

Subbasins:

- Subbasin ID values
- Correspondence Mapping Table between Subbasin and River Segment IDs (could be identical value)
- Subbasin Area [optional]
- etc.



UNIVERSITY OF CALGARY

Let's prepare our geospatial fabric for the study area

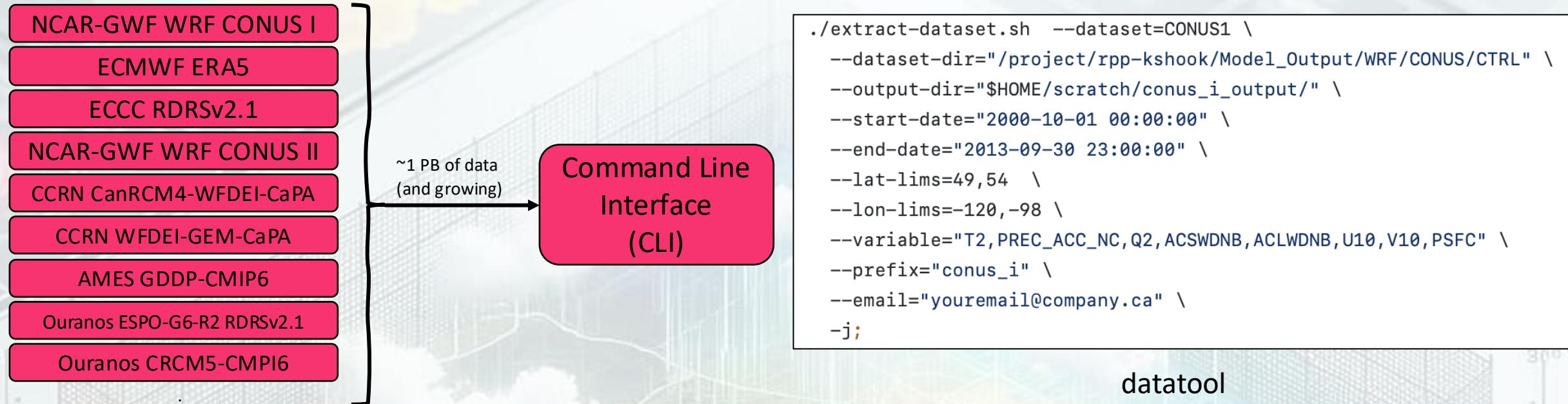
We use MERIT-Basins as the source dataset and extract  
sub-basins and river segments for the area

# Setting up ECCC's National Water Model—MESH



Second step is to run the model-agnostic processes

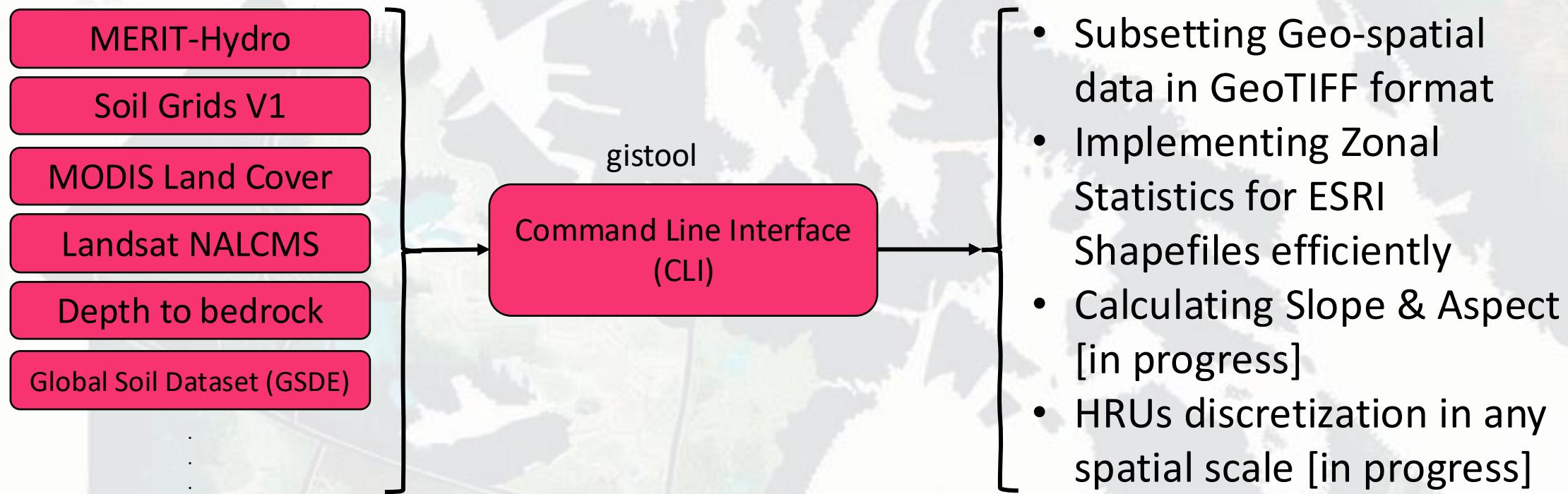
# datatool



## Steps:

1. Login to cluster of choice,
2. Make sure you have access to the designated project space and allocation,
3. Clone the remote GitHub repository,
4. Subset any spatial and temporal extents of interest for any of the available datasets with only **one line of code**.

# gistool

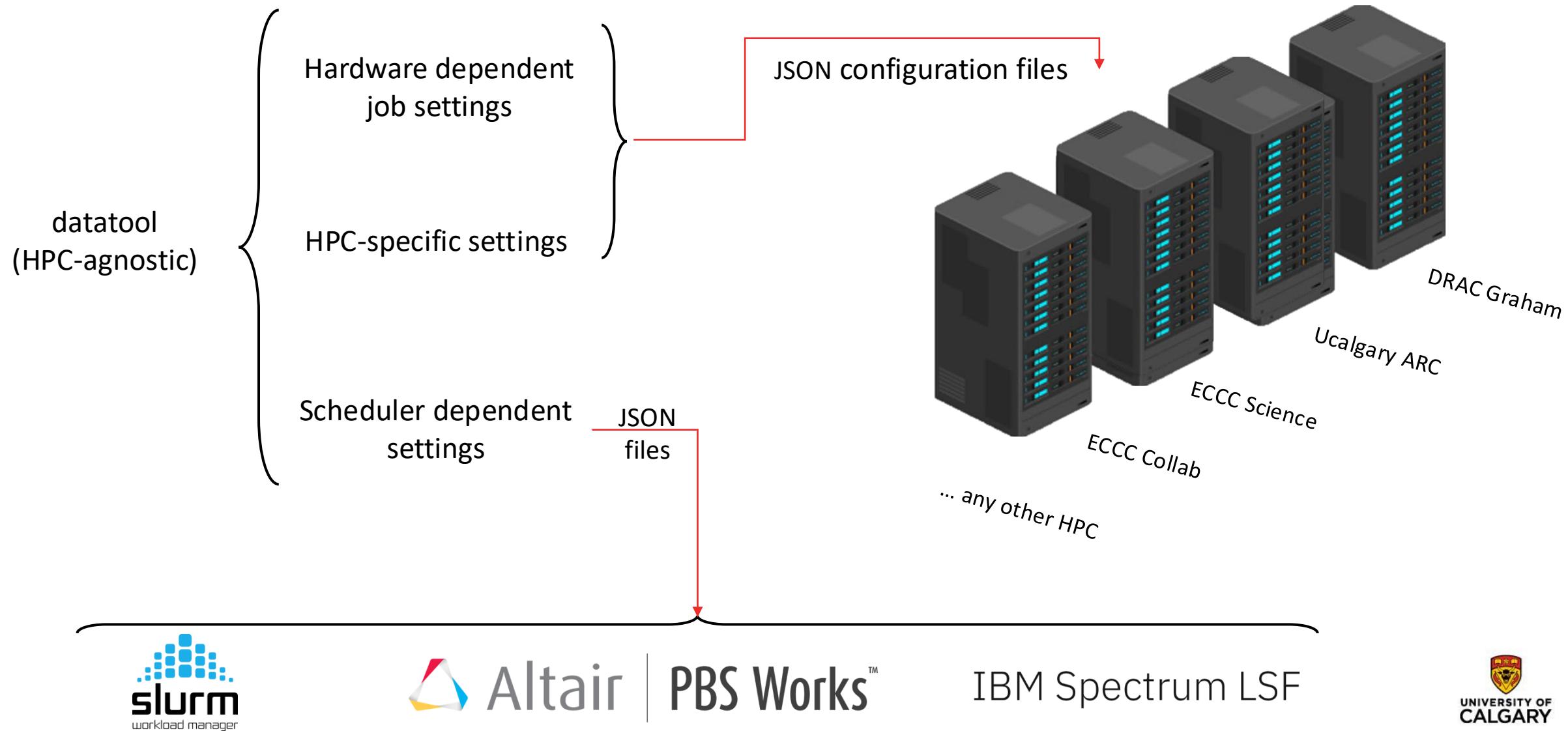


Steps:

1. Login to your cluster of choice,
2. Make sure you have access to the designated project space and allocation,
3. Clone the repository,
4. Subset and/or implement zonal statistics for any spatial and temporal extents of interest for any of the included datasets with only **one line of code**.

<https://github.com/CH-Earth/gistool.git>

# Setting up ECCC's National Water Model—MESH

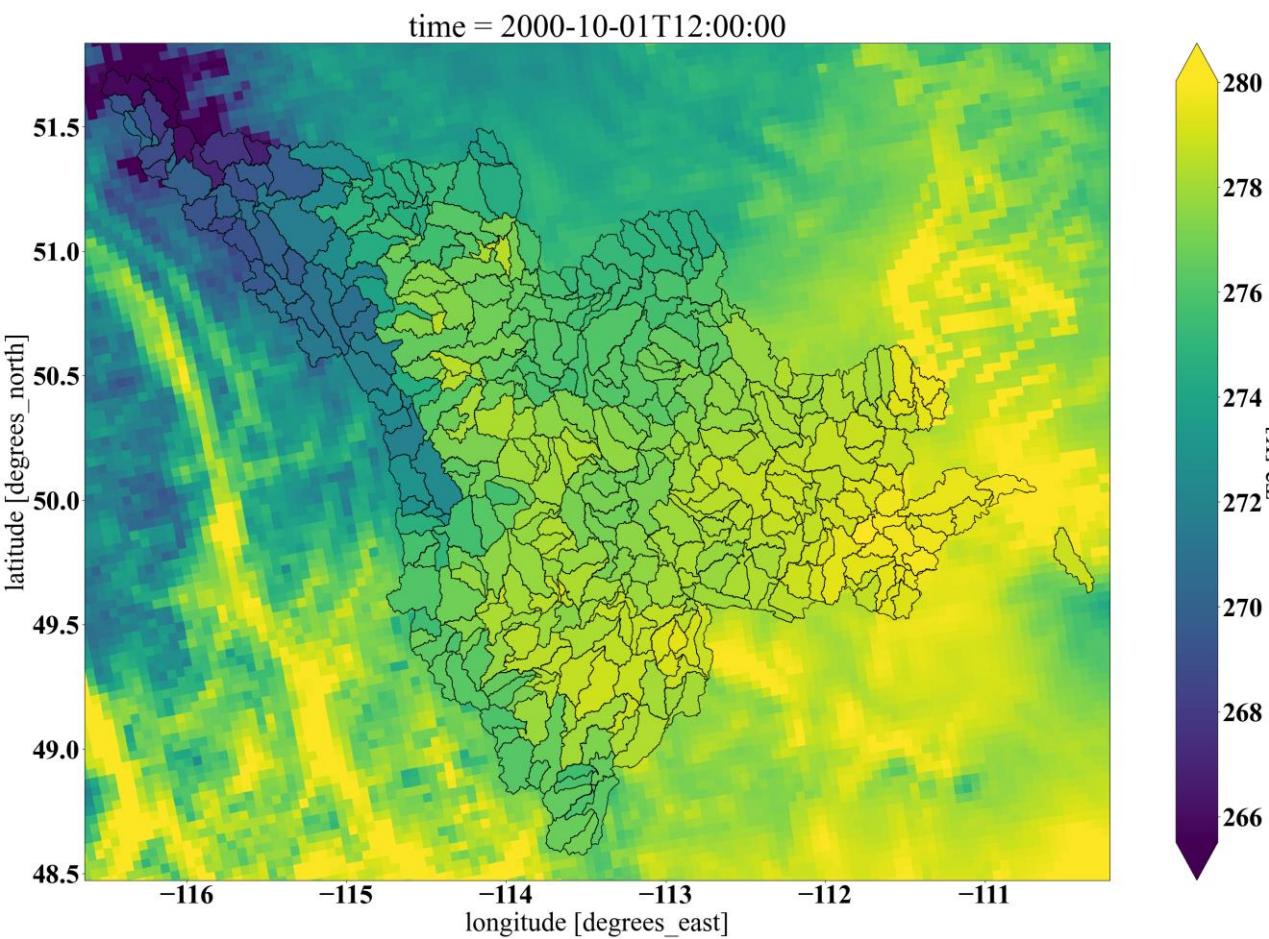


Let's clone *datatool* and *gistool* repositories and see if everything is alright with the HPC-specific JSON file.

Check the accounts you have access to and adjust the JSON file:

```
$ sacctmgr show associations user=$USER
```

# Easymore



<https://github.com/ShervanGharari/EASYMORE.git>

45

```
kasras-mbp:easy more ShervanGharari$ easymore
Usage: easymore [OPTIONS] COMMAND [ARGS]...
```

EASYMORE is a collection of functions that allows extraction of the data from a NetCDF file for a given shapefile such as a basin, catchment, points or lines. It can map gridded data or model output to any given shapefile and provide area average for a target variable.

Options:

- version Show the version and exit.
- help Show this message and exit.

Commands:

- cli Run Easymore using CLI
- conf Run Easymore using a JSON configuration file

For bug reports, questions, and discussions open an issue at  
<https://github.com/ShervanGharari/EASYMORE.git>

Command Line Interface (CLI)  
and  
Python Library

# Any questions so far?



# Setting up ECCC's National Water Model—MESH

Third and final step is to run the model-specific processes

# Setting up ECCC's National Water Model—MESH

The screenshot shows the JupyterLab interface. On the left, a file browser sidebar displays a directory structure under '/github-repos / community-workflows /'. A red arrow points from a callout box to the folder icon in the sidebar header. The callout box contains the text: "Navigate using JupyterLab file browser". The main workspace shows a 'Launcher' panel with sections for 'Notebook', 'Console', and 'Other'. Each section contains icons for different kernels or environments: Python 3.10, Desktop, scienv, and others.

File Edit View Run Kernel Tabs Settings Help

+

Filter files by name

/ github-repos / community-workflows /

Name Last Modified

- 0-prerequisites 14 hours ago
- 1-geofabric 14 hours ago
- 2-agnostic 14 hours ago
- 3-specific 14 hours ago
- LICENSE 14 hours ago
- README.md 14 hours ago

+

Launcher

github-repos/community-workflows

Notebook

Python 3.10 Desktop [↗] scienv

Console

Python 3.10 scienv

Other

\$\_ \$\_ M

Simple 0 \$ 0

Launcher

Let's switch back to the Jupyter environment for the final step.

# Setting up ECCC's National Water Model—MESH

The screenshot shows a Jupyter Notebook interface with two panes. The left pane is a file browser with a sidebar containing icons for file operations like new, upload, and refresh. A search bar at the top says "Filter files by name". Below it is a list of files under the path "... / community-workflows / 3-specific /". The list includes "setting\_files" (modified 20 hours ago), "meshflow\_bb.ipynb" (selected, modified seconds ago), and "README.md" (modified 20 hours ago). A red arrow points from a callout box to "meshflow\_bb.ipynb". The right pane shows the notebook content:

```
meshflow_bb.ipynb
```

We use the `MESHFlow` Python package to build a `MESH` model setup for the Bow River at Banff.

It's also better to keep the geospatial fabric files next to the "agnostic" step's outputs:

```
[ ]: 1 !cp -r ../1-geofabric/bow-at-banff-geofabric/ /home/kasra545/scratch/
```

Let's start by importing the necessary libraries:

```
[ ]: 1 # import necessary libraries
2 import meshflow # version v0.1.0-dev1
3
4 import os # python 3.10.2
```

A red arrow points from a callout box to the line "import meshflow # version v0.1.0-dev1".

Now, let's provide necessary information to set up the `MESH` model:

```
[ ]: 1 # main work path - modify
2 work_path = '/home/kasra545/scratch/bb-models/'
3
4 # using meshflow==v0.1.0-dev1
5 # modify each segment to match your settings
6 config = {
7     'riv': os.path.join(work_path, 'geofabric', 'bb_rivers.shp'),
8     'cat': os.path.join(work_path, 'geofabric', 'bb_subbasins.shp'),
9     'landcover': os.path.join(work_path, 'gistool-outputs', 'bb_mode'),
10    'forcing_files': os.path.join(work_path, 'easymore-outputs'),
11    'forcing_vars': '# does the variable list match those of the "'
```

A red arrow points from a callout box to the line "work\_path = '/home/kasra545/scratch/bb-models/'".

**Meshflow** is the package to build a MESH model out of all the data we processed so far

The root address of where the data is stored

50

Simple 0 \$ 1 scienv | Idle

Mode: Command Ln 1, Col 1 meshflow\_bb.ipynb

UNIVERSITY OF CALGARY

# Setting up ECCC's National Water Model—MESH

The screenshot shows a Jupyter Notebook interface with two panes. The left pane is a file browser showing a directory structure with files like 'setting\_files', 'meshflow\_bb.ipynb', and 'README.md'. The right pane is a code editor with the notebook file open. The code defines a configuration object with various keys mapping to file paths or unit dictionaries. Arrows from the code point to callout boxes on the right, each identifying a specific type of input file or unit requirement.

```
[2]: 1 # main work path - modify
2 work_path = '/home/kasra545/scratch/bb-models/'
3
4 # using meshflow==v0.1.0-dev1
5 # modify each segment to match your settings
6 config = {
7     'riv': os.path.join(work_path, 'geofabric', 'bb_rivers.shp'),
8     'cat': os.path.join(work_path, 'geofabric', 'bb_subbasins.shp'),
9     'landcover': os.path.join(work_path, 'gistool-outputs', 'bb_mode',
10                               'bb_subbasin'),
11     'forcing_files': os.path.join(work_path, 'easymore-outputs'),
12     'forcing_vars': [ # does the variable list, match those of the forcing files?
13         "RDRS_v2.1_P_P0_SFC",
14         "RDRS_v2.1_P_HU_09944",
15         "RDRS_v2.1_P_TT_09944",
16         "RDRS_v2.1_P_UVC_09944",
17         "RDRS_v2.1_A_PR0_SFC",
18         "RDRS_v2.1_P_FB_SFC",
19         "RDRS_v2.1_P_FI_SFC",
20     ],
21     'forcing_units': { # Here, enter RDRS's original variable units
22         'RDRS_v2.1_P_P0_SFC': 'millibar',
23         'RDRS_v2.1_P_HU_09944': 'kg/kg',
24         'RDRS_v2.1_P_TT_09944': 'celsius',
25         'RDRS_v2.1_P_UVC_09944': 'knot',
26         'RDRS_v2.1_A_PR0_SFC': 'm/hr',
27         'RDRS_v2.1_P_FB_SFC': 'W/m^2',
28         'RDRS_v2.1_P_FI_SFC': 'W/m^2',
29     },
30     'forcing_to_units': { # And here, the units that MESH needs to re-project
31         'RDRS_v2.1_P_UVC_09944': 'm/s',
32         'RDRS_v2.1_P_FI_SFC': 'W/m^2',
33         'RDRS_v2.1_P_FB_SFC': 'W/m^2',
34         'RDRS_v2.1_A_PR0_SFC': 'mm/s',
35         'RDRS_v2.1_P_P0_SFC': 'pascal',
36         'RDRS_v2.1_P_TT_09944': 'kelvin',
37         'RDRS_v2.1_P_HU_09944': 'kg/kg',
38     },
39 }
```

River network geometry file

Subbasin geometry file

Landcover fractions file(s)

Prepared forcing file(s)

List of variables to be used

Variables' default units in the NetCDF files

Units of variables that MESH requires

# Setting up ECCC's National Water Model—MESH

The screenshot shows a Jupyter Notebook interface with two panes. The left pane is a file browser showing a directory structure with files: setting\_files, meshflow\_bb.ipynb (selected), and README.md. The right pane is a code editor with the file `meshflow_bb.ipynb`. The code defines several variables:

```
38: 'main_id': 'COMID', # what is the main ID of each river segment?
39: 'ds_main_id': 'NextDownID', # what is the downstream segment ID?
40: 'landcover_classes': { # these are the classes defined for NALCMS
41:     0: 'Unknown',
42:     1: 'Temperate or sub-polar needleleaf forest',
43:     2: 'Sub-polar taiga needleleaf forest',
44:     3: 'Tropical or sub-tropical broadleaf evergreen forest',
45:     4: 'Tropical or sub-tropical broadleaf deciduous forest',
46:     5: 'Temperate or sub-polar broadleaf deciduous forest',
47:     6: 'Mixed forest',
48:     7: 'Tropical or sub-tropical shrubland',
49:     8: 'Temperate or sub-polar shrubland',
50:     9: 'Tropical or sub-tropical grassland',
51:    10: 'Temperate or sub-polar grassland',
52:    11: 'Sub-polar or polar shrubland-lichen-moss',
53:    12: 'Sub-polar or polar grassland-lichen-moss',
54:    13: 'Sub-polar or polar barren-lichen-moss',
55:    14: 'Wetland',
56:    15: 'Cropland',
57:    16: 'Barren lands',
58:    17: 'Urban',
59:    18: 'Water',
60:    19: 'Snow and Ice',
61: },
62: 'ddb_vars': { # the stuff that MESH needs: slope, river length, etc
63:     'slope': 'ChnlSlope',
64:     'lengthkm': 'ChnlLength',
65:     'Rank': 'Rank',
66:     'Next': 'Next',
67:     'landcover': 'GRU',
68:     'unitarea': 'GridArea',
69:     'landcover_names': 'LandUse',
70: },
71: 'ddb_units': {
72:     'ChnlSlope': 'm/m',
73:     'ChnlLength': 'km', # is it in km or m? Please check the units
74:     'Rank': 'dimensionless',
75:     'Next': 'dimensionless',
76:     'GRU': 'dimensionless',
77:     'GridArea': 'km^2', # what was the unit of the GridArea, or square km
78:     'LandUse': 'dimensionless',
79: },
```

Callouts point to specific sections of the code:

- River Segment/Subbasin ID: Points to the `main_id` and `ds_main_id` variables.
- Downstream river segment/subbasin ID: Points to the `ds_main_id` variable.
- Landcover classes of the landcover dataset used: Points to the `landcover_classes` dictionary.
- All the variables to be used in the "drainage database" MESH file: Points to the `ddb_vars` and `ddb_units` sections.
- Units of variables to be included in the "drainage database" MESH file: Points to the `ddb_units` section.

Page footer: 52

Page footer: UNIVERSITY OF ALBERTA

# Setting up ECCC's National Water Model—MESH

The screenshot shows a Jupyter Notebook interface with two panes. The left pane is a file browser showing a directory structure with files like 'setting\_files', 'meshflow\_bb.ipynb', and 'README.md'. The right pane is a code editor for 'meshflow\_bb.ipynb'.

**Code Editor Content:**

```
80     'ddb_to_units': {
81         'ChnlSlope': 'm/m',
82         'ChnlLength': 'm', # This is what MESH needs, no need to change
83         'Rank': 'dimensionless',
84         'Next': 'dimensionless',
85         'GRU': 'dimensionless',
86         'GridArea': 'm^2', # This is what MESH needs, no need to change
87         'LandUse': 'dimensionless',
88     },
89     'ddb_min_values': {
90         'ChnlSlope': 1e-10, # in case there are 0s in the `rivers` SIDs
91         'ChnlLength': 1e-3,
92         'GridArea': 1e-3,
93     },
94     'gru_dim': 'NGRU', # change to `NGRU` for `MESH>=r1860`, keep for now
95     'hru_dim': 'subbasin',
96     'outlet_value': -9999,
97 }
```

**Annotations:**

- Initiate an instance of the `MESHWorkflow` object**: Points to the line `[3]: exp1 = meshflow.MESHWorkflow(**config)`.
- And running the “model-specific” step**: Points to the line `[4]: exp1.run()`.
- Units of variables in the “drainage database” file that MESH requires**: Points to the 'GridArea' entry in the JSON code.
- Minimum values of “drainage database” MESH file**: Points to the 'GridArea' entry in the 'ddb\_min\_values' block.
- Renaming GRU dimension name in the MESH “drainage database” file to satisfy diverse versions of MESH**: Points to the 'gru\_dim' entry.
- Renaming sub-basin dimension name in the MESH “drainage database” file if needed**: Points to the 'hru\_dim' entry.
- The value of the outlet river segments created using “Hydrant” (Hydrant default is -9999)**: Points to the 'outlet\_value' entry.

**Code Cells:**

- [3]: `exp1 = meshflow.MESHWorkflow(**config)`
- [4]: `exp1.run()`

**Output and Notes:**

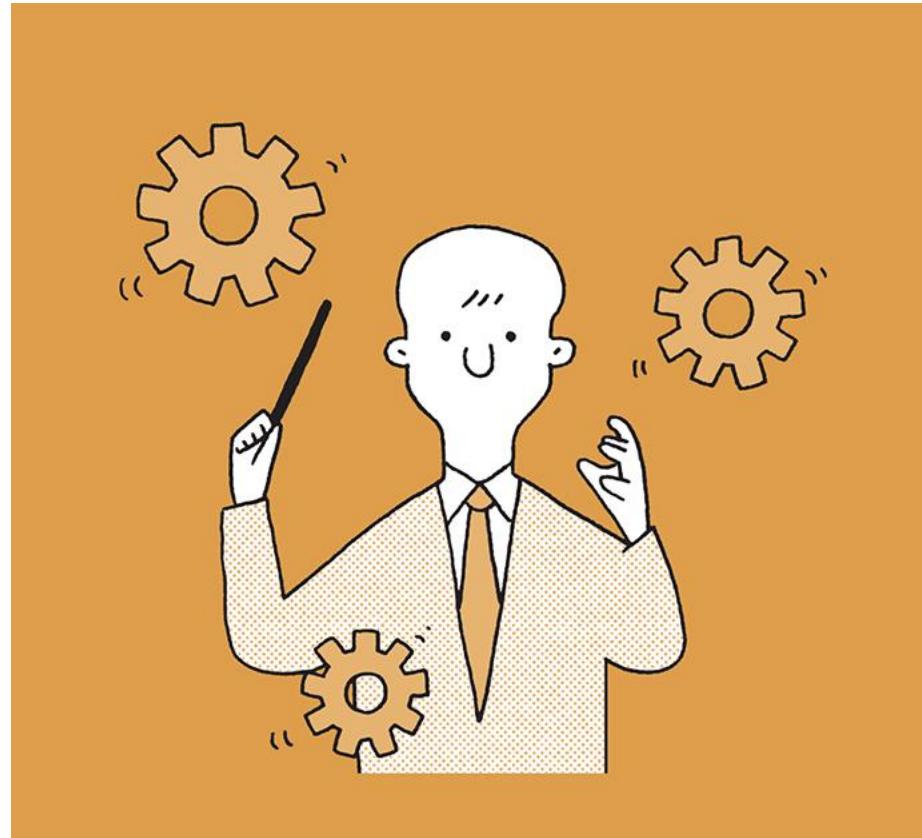
```
/home/kasra545/github-repos/meshflow/src/meshflow/utility/forcing_prep.py:116: FutureWarning: The return type of `Dataset.dims` will be changed to return a set of dimension names in future, in order to be more consistent with `dataArray.dims`. To access a mapping from dimension names to lengths, please use `Dataset.sizes`.  
var = [i for i in ds.dims.keys() if i != 'time']
```

Once the run is finished, we can checkout the forcing and drainage database file:

# Any questions so far?

Before finishing, let's look at one last tool.

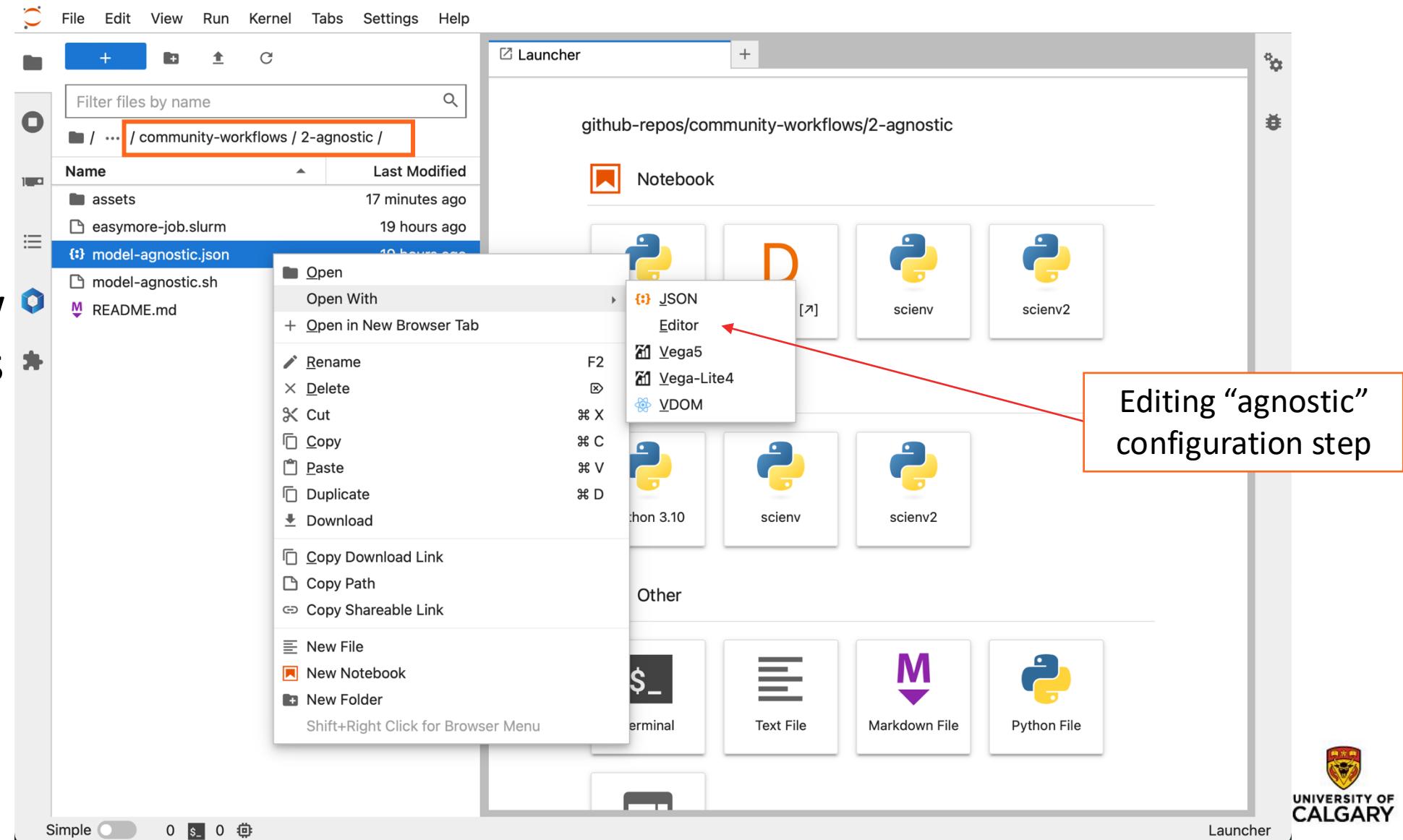
# Setting up ECCC's National Water Model—MESH



- Through a tool called “agnostic orchestrator” all the relevant data needed for this setup is processed by the Graham HPC
- We have previously downloaded all the relevant tools for the “agnostic” step
- We use the “agnostic orchestrator” configuration file instruct the file processing hierarchy

# Setting up ECCC's National Water Model—MESH

Agnostic workflow  
execution using its  
“Orchestrator”



# Setting up ECCC's National Water Model—MESH

The screenshot shows a Jupyter Notebook interface with a file browser on the left and a code editor on the right. The code editor displays a JSON file named `model-agnostic.json`. The file defines an orchestrator configuration with sections for executable paths and arguments.

```
1 {  
2     "exec": {  
3         "met": "/home/kasra545/github-repos/datatool/extract-dataset.sh",  
4         "gis": "/home/kasra545/github-repos/gistool/extract-gis.sh",  
5         "remap": "easymore cli"  
6     },  
7  
8     "args": {  
9         "met": [{  
10             "dataset": "RDRS",  
11             "dataset-dir": "/project/rrg-mclark/data/meteorological-data/rdrsv2.1/",  
12             "variable": [  
13                 "RDRS_v2.1_P_P0_SFC",  
14                 "RDRS_v2.1_P_HU_09944",  
15                 "RDRS_v2.1_P_TT_09944",  
16                 "RDRS_v2.1_P_UVC_09944",  
17                 "RDRS_v2.1_A_PR0_SFC",  
18                 "RDRS_v2.1_P_FB_SFC",  
19                 "RDRS_v2.1_P_FI_SFC"  
20             ],  
21             "output-dir": "/home/kasra545/scratch/bb-models/datatool-outputs",  
22             "start-date": "1980-01-01T13:00:00",  
23             "end-date": "1980-01-5T12:00:00",  
24             "lat-lims": "",  
25             "lon-lims": "",  
26             "shape-file": "/home/kasra545/github-repos/MESH-Bow-at-Banff/1-geofabric/bow-at-banff-geofabric/bb_subbasins.shp",  
27             "model": "",  
28             "ensemble": "",  
29             "prefix": "bb_model_",  
30             "email": "kasra.keshavarz1@ucalgary.ca",  
31             "account": "rrg-mclark",  
32             "_flags": [  
33                 "submit-job",  
34                 "parsable"  
35             ]  
36         ]  
37     }  
}
```

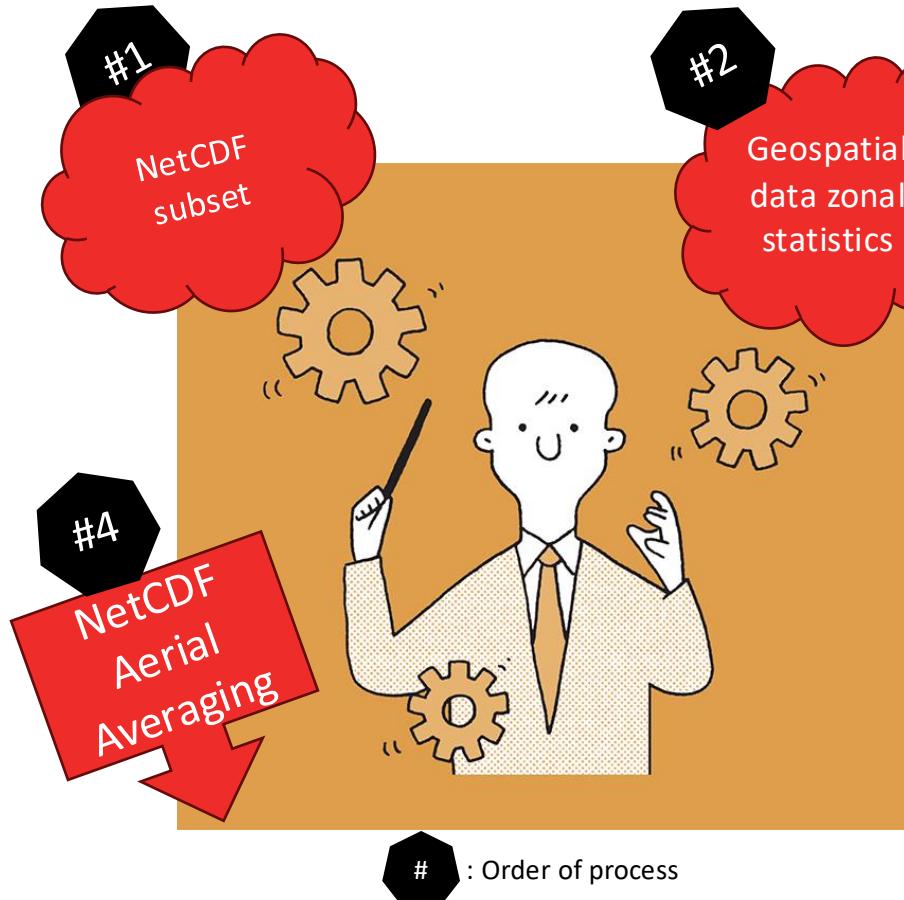
**Executable paths**

**Orchestrator configuration file**

**Executable arguments and options  
(empty options are ignored)**

**This is the only file for the “agnostic” step**

# Setting up ECCC's National Water Model—MESH



Automated and ordered job submission to HPC

The terminal window displays a file browser interface and a code editor for a JSON file named `model-agnostic.json`.

**File Browser:**

- File
- Edit
- View
- Run
- Kernel
- Tabs
- Settings
- Help

Search bar: Name by name

Path: community-workflows / 2-agnostic /

Table of contents:

- assets
- easymore-job.slurm
- model-agnostic.json
- model-agnostic.sh
- README.md

Last Modified column shows times: an hour ago, 19 hours ago, 19 hours ago, 19 hours ago, 19 hours ago.

**Code Editor (model-agnostic.json):**

```
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
"remap": [
  "case-name": "remapped",
  "cache": "/home/kasra545/scratch/bb-models/easymore-outputs/cache/",
  "shapefile": "/home/kasra545/github-repos/MESH-Bow-at-Banff/1-geofabric/bow-at-banff-geofabric/bb_subbasins.shp",
  "shapefile-id": "COMID",
  "source-nc": "/home/kasra545/scratch/bb-models/datatool-outputs/**/*.nc",
  "variable-lon": "lon",
  "variable-lat": "lat",
  "variable": [
    "RDRS_v2.1_P_P0_SFC",
    "RDRS_v2.1_P_HU_09944",
    "RDRS_v2.1_P_TT_09944",
    "RDRS_v2.1_P_UVC_09944",
    "RDRS_v2.1_A_PR0_SFC",
    "RDRS_v2.1_P_FB_SFC",
    "RDRS_v2.1_P_FI_SFC"
  ],
  "remapped-var-id": "id",
  "remapped-dim-id": "id",
  "output-dir": "/home/kasra545/scratch/bb-models/easymore-outputs/",
  "job-conf": "/home/kasra545/github-repos/MESH-Bow-at-Banff/2-agnostic/easymore-job.slurm",
  "_flags": [
    "submit-job"
  ]
},
"order": {
  "met": 1,
  "gis": -1,
  "remap": 2
}
```

-1: no order

ordered integers: order of processes

Ln 1, Col 1 Spaces: 4 model-agnostic



UNIVERSITY OF  
CALGARY

# Any questions so far?



Thank you for attending

<https://ucalgary.ca/civil>



UNIVERSITY OF  
**CALGARY**

# Backup slides



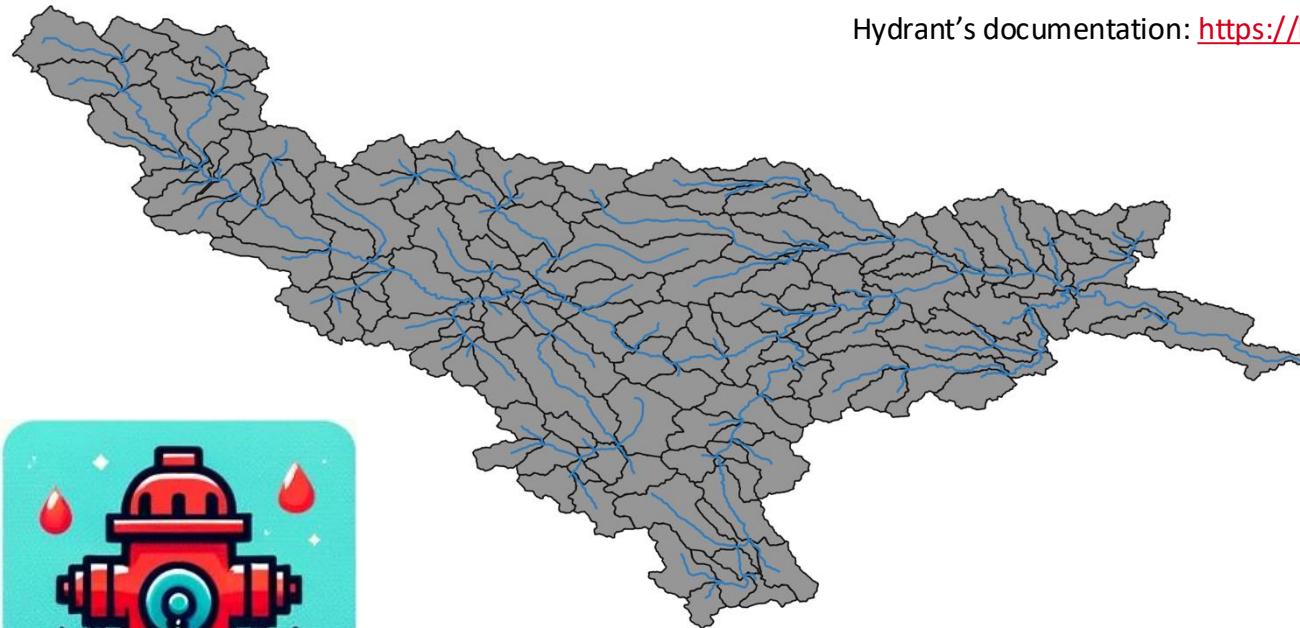
UNIVERSITY OF  
**CALGARY**

# Computational Resources

1	Only for personal use	Home space 500 GB per user on ARC
2	Only for data processing	Scratch space 15 TB per experiment
3	NOT for personal use – only for shared resources	Project space comphyd_lab: 0.2 PB 



# Setting up ECCC's National Water Model—MESH



**HYDRANT**

Bow River at Banff

Sub-basins and river network extracted from MERIT-Basins  
using Hydrant v0.1.0-dev0

How does HydrAnt help us?

Hydrant's documentation: <https://hydrantpy.readthedocs.org>



- Extracting subsets of hydrography datasets given a point or a boundary area,
- Hydrant is data-agnostic; if the input hydrography is valid and standard, you can use Hydrant,
- Common sanity checks for datasets:
  - Is water going upstream?
  - Is there a cycle of river segments?
  - etc.
- Aggregation methods to dissolve elements of hydrography datasets
- Knowledge extraction:
  - What is the longest branch of river segments?
  - What are the upstream segment of a certain point in the river network?
  - What are the downstream segments of a river segment?
  - What are the Strahler order of river segment?

# Setting up ECCC's National Water Model—MESH

The screenshot shows a Jupyter Notebook interface. On the left, a file browser displays a folder structure under 'community-workflows / 1-geofabric'. A red box highlights the 'pre-process-geospa...' file, with a callout 'Double click to launch the Notebook' pointing to it. The main area shows a notebook tab titled 'pre-process-geospa...'. The first section, 'Basic preparations', contains text about extracting a geospatial fabric from the MERIT-Basins dataset. The second section contains code to import Python libraries. A red box labeled 'Notebook cells' points to the code cell. A callout 'Tip: review keyboard shortcuts!' points to the red box. Another callout 'Use "Shift + Enter" to execute Notebook cells' points to the right edge of the code cell.

File Edit View Run Kernel Tabs Settings Help

Filter files by name

/ ... /community-workflows / 1-geofabric

Name Last Modified

bow-at-banff-geofab... 14 hours ago

pre-process-geospa... 14 hours ago

README.md 14 hours ago

Double click to launch the Notebook

Basic preparations

In this Notebook, the geospatial fabric for the "Bow River at Banff" gauge is extracted from the MERIT-Basins dataset.

If you are using Graham HPC, and have access to Clark's Research Group allocation ( rrg-mclark ), you may find MERIT-Basins layers under the following path:  
/project/rrg-mclark/data/geospatial-data/MERIT-Basins/

Let's get started with our workflow and import necessary Python libraries:

```
[1]:  
1 import geopandas as gpd # version 0.14.3  
2 import pandas as pd # version 2.1.1  
3 import numpy as np # version 1.24.4  
4 import matplotlib.pyplot as plt # version 3.5.1  
5  
6 from shapely.geometry import Point # version 2.0.0  
7  
8 import hydrant.topology.geom as gm # version 0.1.0  
9  
10 import subprocess # built-in Python 3.10.2  
11 import os # built-in Python 3.10.2  
12 import glob # built-in Python 3.10.2
```

Mode: Command Ln 1, Col 1 pre-process-geospatial-fabric.ipynb

Use "Shift + Enter" to execute Notebook cells

Notebook cells

Tip: review keyboard shortcuts!