

Community Hydrological Modelling Workflows

**Online Training Session with Federal, Provincial, Territorial, and
International 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

Darri Eythorsson, PhD
Postdoctoral Associate
Schulich School of Engineering
University of Calgary

Martyn Clark, PhD
Professor and Schulich Research Chair
United Nations Hub Co-Director
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



Yukon

ACCESS



UNIVERSITY OF
CALGARY

Outline

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

Outline

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

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. Knoben¹ , M. P. Clark^{1,2} , J. Bales³, A. Bennett⁴ , S. Gharari⁵ , C. B. Marsh⁵ , B. Nijssen⁶ , A. Pietroniro⁷ , R. J. Spiteri⁸ , G. Tang¹ , D. G. Tarboton⁹ , and A. W. Wood¹⁰

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

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

⁴
⁵
**Kasra Keshavarz^{1*}, Alain Pietroniro¹, Darri Eythorsson¹, Mohamed Ismaiel
Ahmed¹, Paul Coderre¹, Wouter Knoben¹, Shervan Gharari¹, Martyn Clark¹**

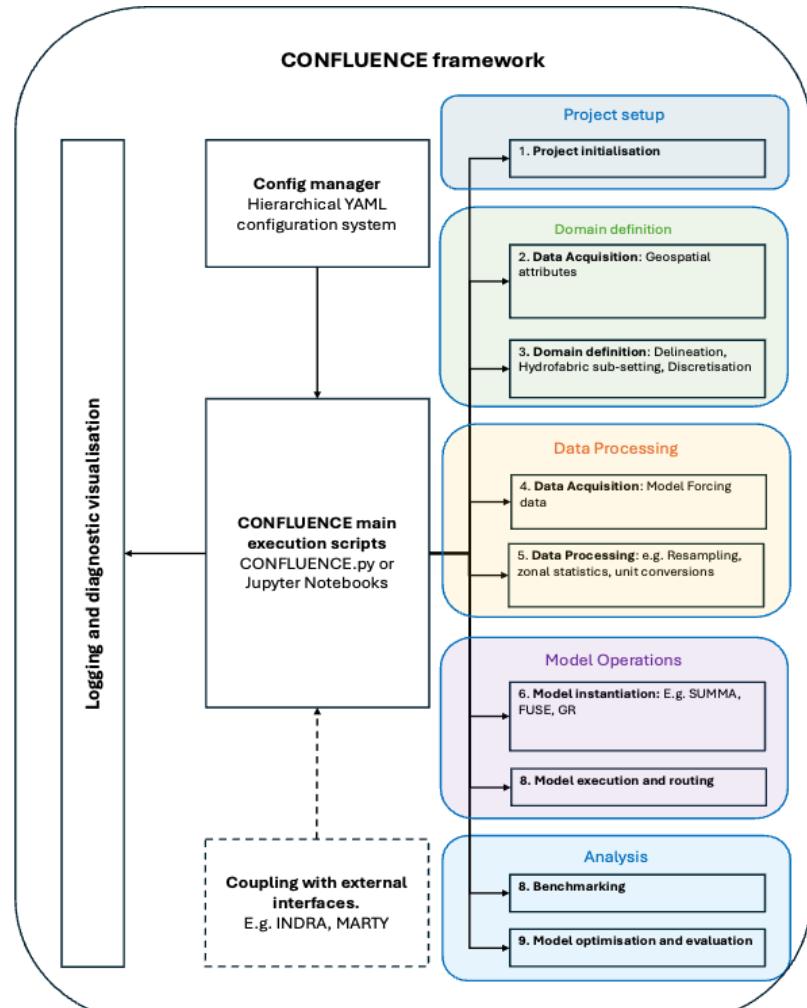
⁶
⁷
¹Department of Civil Engineering, Schulich School of Engineering, University of Calgary, Calgary, AB
T2N 1N4, Canada

⁸
Key Points:

- ⁹
¹⁰
¹¹
¹²
¹³
¹⁴
 - A generalized model-agnostic framework is proposed to streamline configurations of process-based hydrological models
 - The framework advances reproducibility in hydrological modelling by addressing complexities involved in handling dynamic and static datasets
 - Prototype applications in a high-performance computing environment demonstrate 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

Recipe for Community Hydrological Modelling



Extended workflow to seamlessly include:

- Domain definition
- Model execution
- Model evaluation and optimization
- Benchmarking
- Visualization and reporting

Design principles:

- Modularized methodology for extendibility and interoperability
- Isolation of model decisions in a central configuration system
- Facilitating machine operability for computational exploration and optimization

Outline

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

Computational Resources

- Wikipedia (accessed February 13th, 2025)
"High-performance computing (HPC) is the use of supercomputers and computer clusters to solve advanced computation problems."
- The idea is to expand computational performance horizontally (multiple CPUs), instead of vertically (one massive CPU)
- High Performance Computing (HPC) systems are comprised of various “nodes”
- HPC nodes are located inside rack enclosures



9

<https://top500.org/>



UNIVERSITY OF
CALGARY

Computational Resources



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

The JupyterLab logo, featuring the word "jupyter" in a dark grey sans-serif font and "lab" in a larger orange sans-serif font, all contained within a rounded rectangular frame with an orange border.

- 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

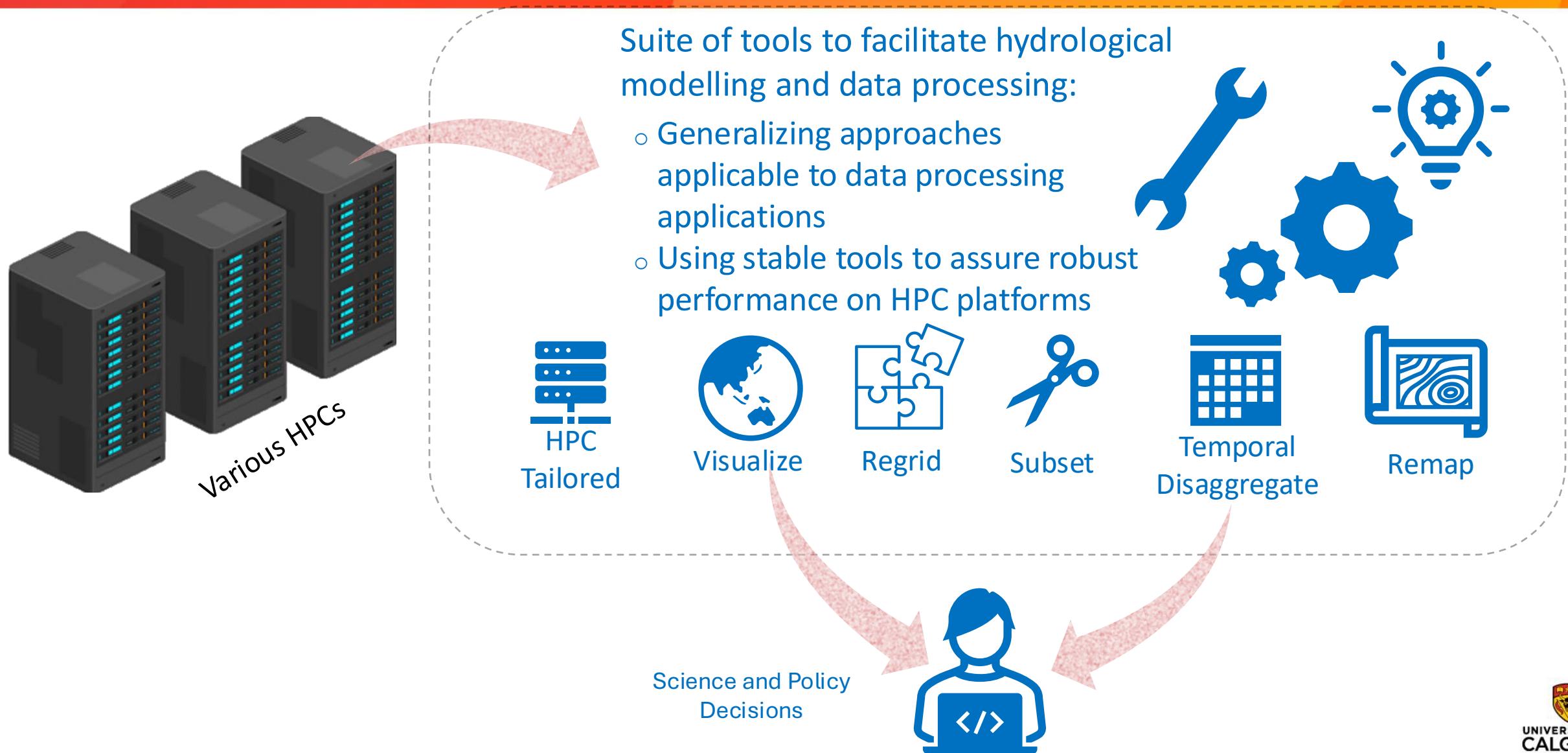


PuTTY



MacOS Terminal

Computational Resources



Computational Resources

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



Anvil at Purdue University



Digital Research
Alliance of Canada |
Alliance de recherche
numérique du Canada

Fir at Simon Fraser University



UNIVERSITY OF
CALGARY

ARC at the University of Calgary



UNIVERSITY OF
CALGARY

Computational Resources

Since not everybody has access to an HPC, we need to separate into two groups:



ARC at the University of Calgary



Computational Resources



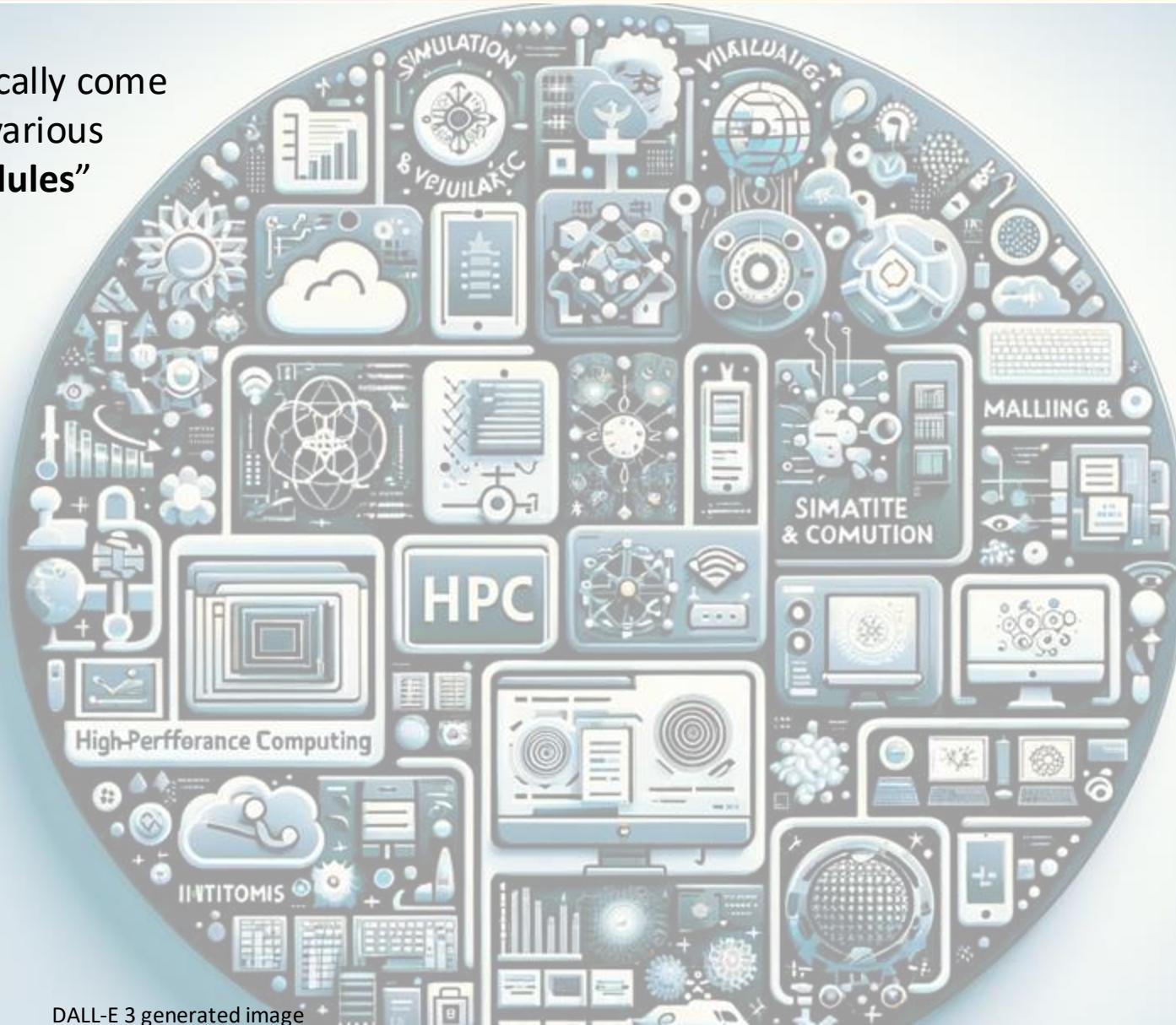
HPCs typically come with various “modules”

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

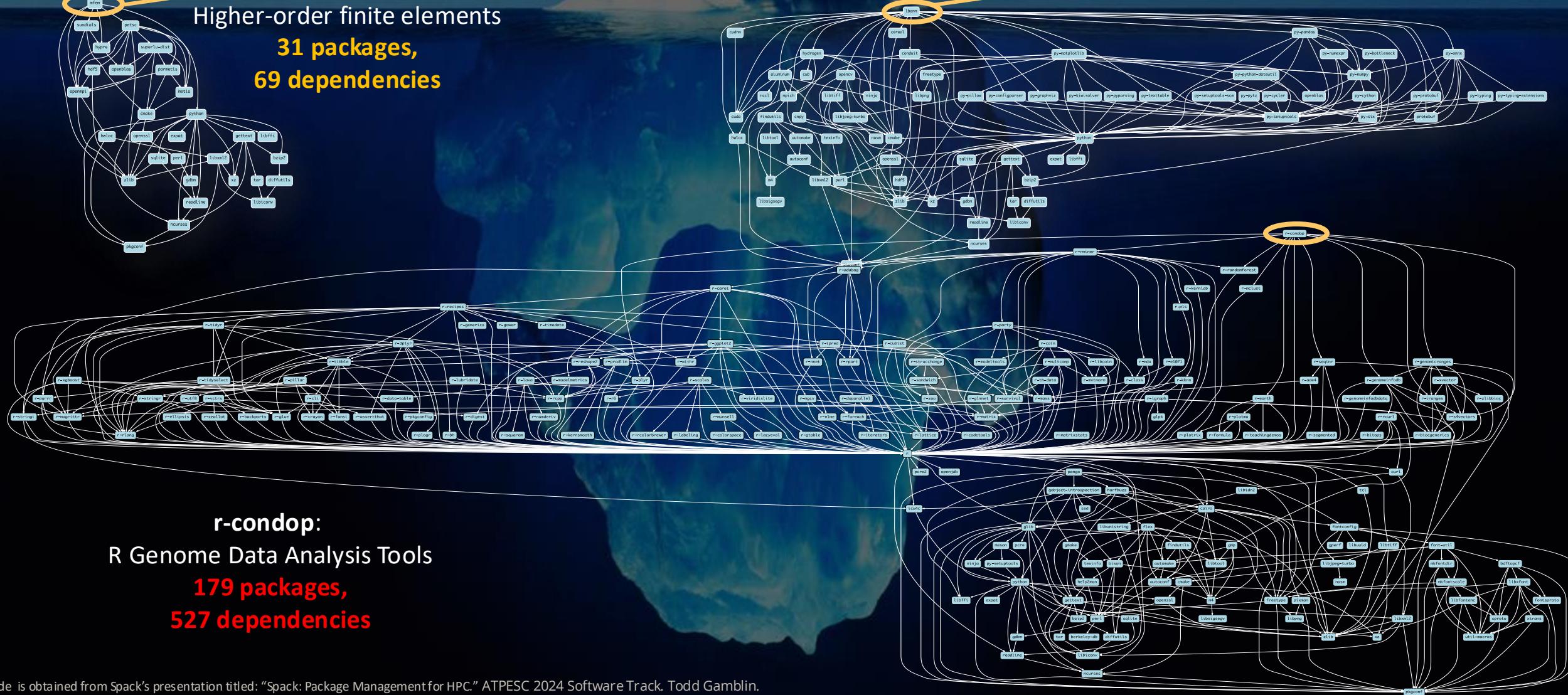
Modern scientific codes rely on icebergs of dependency libraries

MFEM: Higher-order finite elements **31 packages,** **69 dependencies**

LBANN: Neural Nets for HPC

R Genome Data Analysis Tools

179 packages,
527 dependencies



Any questions so far?

Computational Resources



Thanks to [OVH](#), [GESIS Notebooks](#) and [2i2c](#) for supporting us! 🎉
mybinder.org has updated the base image to Ubuntu 22.04! See the [upgrade guide](#) for details.

[Do](#)



Launching your Binder...

Read our [advice for speeding up your Binder launch](#).

Waiting Building Pushing Launching

Build Logs

[show](#) [view raw](#)

Here is a non-interactive preview on [nbviewer](#) while we start a server for you.
Your binder will open automatically when it is ready.



UNIVERSITY OF
CALGARY

Computational Resources



ARC at the University of Calgary

OPEN
OnDemand



Welcome to ARC OnDemand

Pinned Apps A featured subset of [all available apps](#)

Interactive Apps

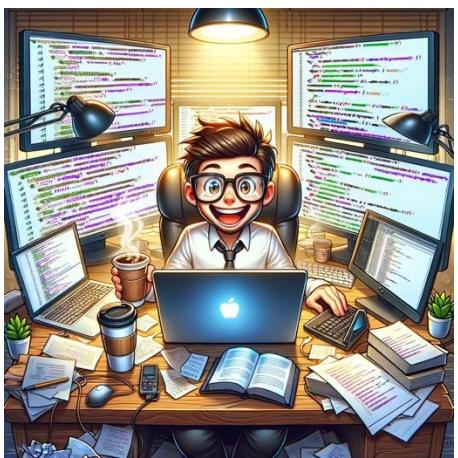


Computational Resources



General tips:

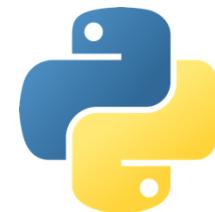
- Understand your setup: Be resourceful—Asking too much computational resource can keep you in the queue forever
- Choose the Right Kernel: Be mindful—Jupyter provides you with all sort of programming environments; make sure to choose the right (customized) programming kernel for your work.
- Monitor Performance and Document: Be diligent—As you work more with the environment, you get to know the requirements of your computations. Keep track of your workflow's performance and document your work.



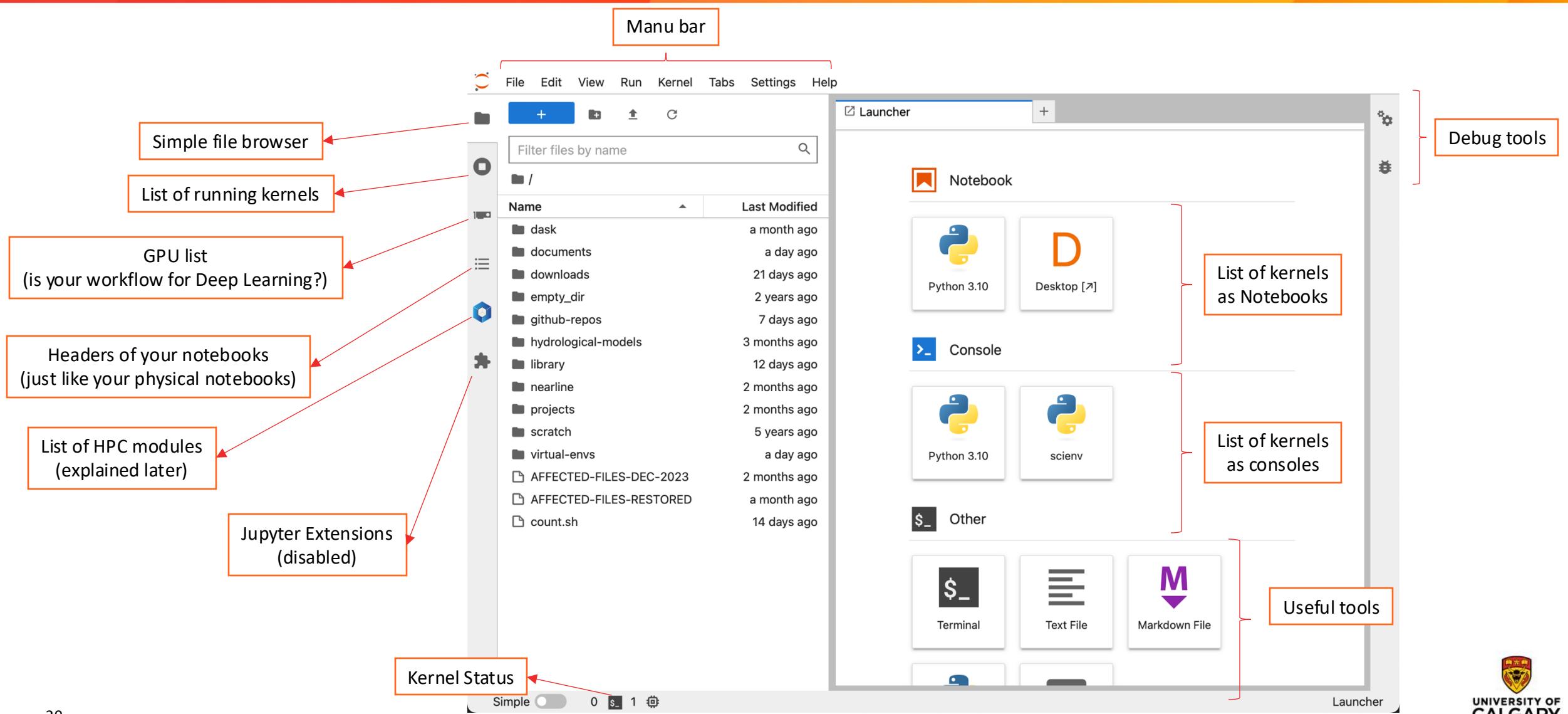
DALL-E 3 generated image



DALL-E 3 generated image



Computational Resources



Computational Resources

The screenshot shows the JupyterLab interface. On the left, a sidebar displays a list of modules:

- Loaded modules:** 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.
- List of HPC modules (LMOD Extension):** StdEnv/2016.4, StdEnv/2018.3, StdEnv/2023, abaqus/2021, abinit/9.2.2, abinit/9.6.2, actc/1.1, adf/2019.305, admixture/1.3.0, adol-c/2.7.2.
- Unavailable modules:** 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 top right, there is a "Launcher" section with icons for Notebook, Console, and Other (Terminal, Text File, Markdown File). A "Restore a previously saved collection" button is also present.

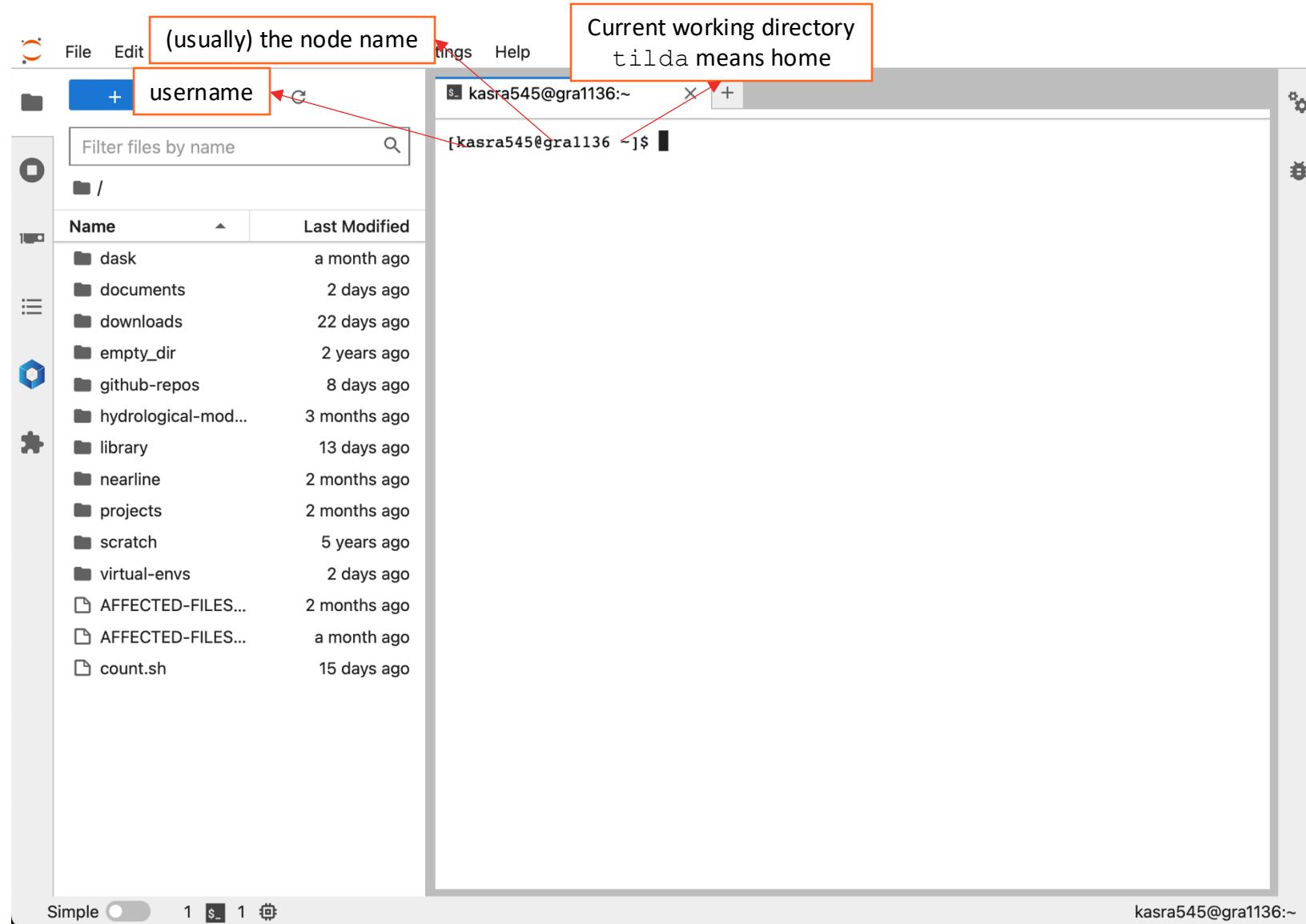
Annotations:

- Loaded modules:** Boxed and connected to the "Loaded modules" list in the sidebar.
- Create a collection out of loaded modules:** Boxed and connected to the "Restore a previously saved collection" button.
- A saved list of modules is called a “collection”**: Boxed annotation pointing to the "Launcher" section.
- Using JupyterLab interface**: Boxed annotation pointing to the "Launcher" section.
- For creating collections could be time consuming.**: Boxed annotation pointing to the "Launcher" section.
- We will use the Terminal to build a collection later.**: Boxed annotation pointing to the "Terminal" icon in the "Other" section of the launcher.

Computational Resources

All instructions are available
on this presentation's
repository:

<https://github.com/CH-Earth/community-modelling-workflow-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/CH-Earth/community-modelling-workflow-training>

The screenshot shows a terminal window with the following interface elements:

- Top Bar:** File, Edit, View, Help, Computing node name (kasra.keshavarz1@fc1), Current working directory (tilda means home), and a user icon.
- Left Sidebar:** A file browser with a search bar containing "username". It lists various directories and files, such as alliancecan-training-data, archive_mesh_baker, archive_mesh_marmot, backups, benchmarking-calibration, benchmarking-frdr, benchmarking-models, cache, datatool, docker-conts, downloads, empty_dir, enci619.05_2024w, github-repos, mesh-calgary, ondemand, outs, privatemodules, projects, scratch, spack, spack-keys, syncrosim, test, and virtual-envs. The "username" folder is highlighted.
- Terminal Area:** The main pane displays command-line output:

```
[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
- Bottom Status Bar:** Simple, 1 \$, 0, and a bell icon.
- Bottom Footer:** kasra.keshavarz1@fc1:~ 0

Getting a list of available modules

Loading every module we need

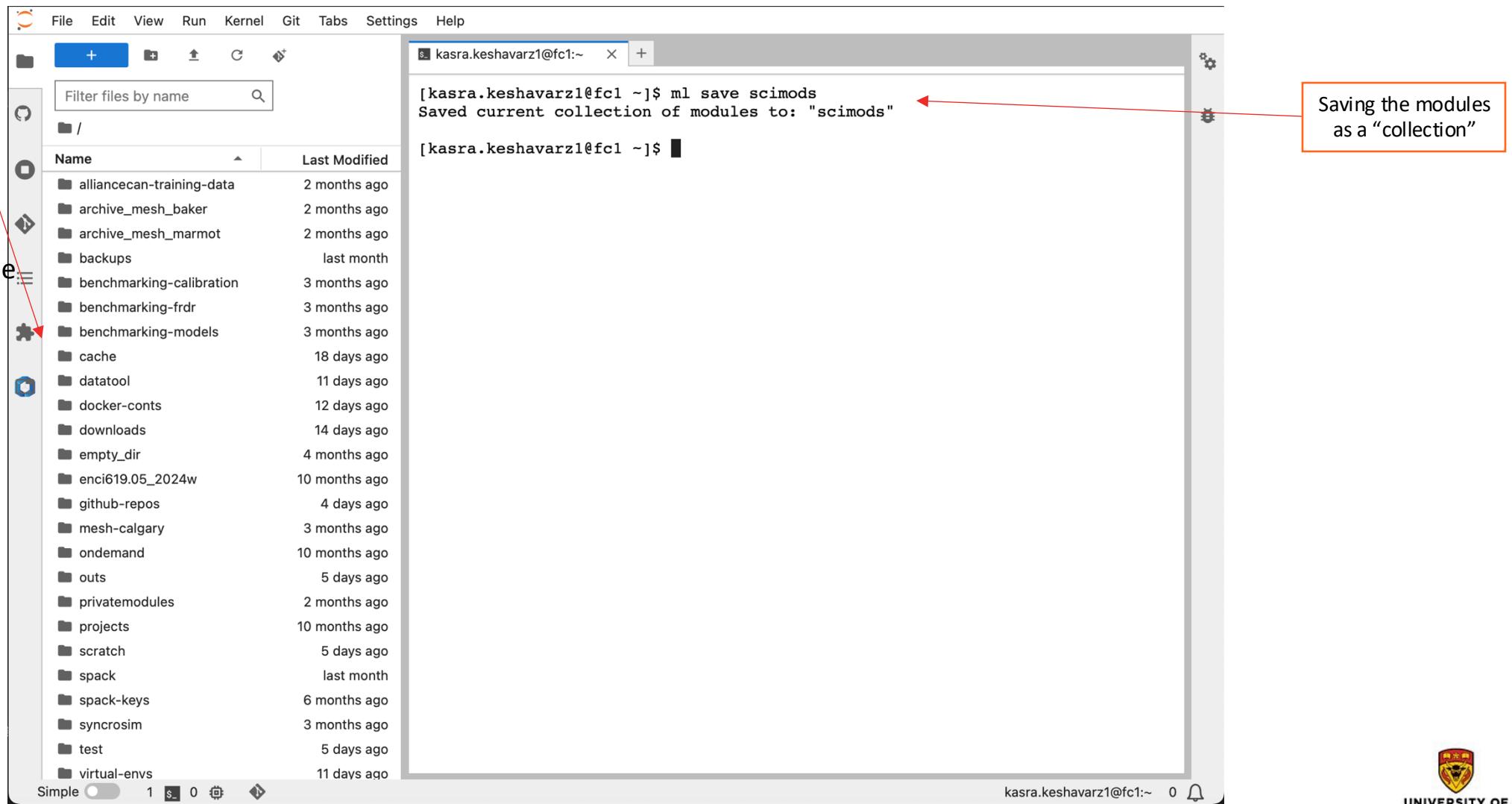
Double checking whatever I have loaded

Computational Resources

Exactly similar to what we can do in this tab

All instructions are available on this presentation's repository:

<https://github.com/CH-Earth/community-modelling-workflow-training>



The screenshot shows a terminal window with a file browser sidebar. The terminal command history includes:

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

A red box highlights the terminal window, and a red arrow points from the text "Exactly similar to what we can do in this tab" to the terminal icon in the file browser sidebar. Another red arrow points from the text "Saving the modules as a ‘collection’" to the word "scimods" in the terminal output.

File browser sidebar content:

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, a sidebar lists 'LOADED MODULES' with items like glibc/2.28, gcc-runtime/14.2.0, libmd/1.0.4, etc. A red box highlights the sidebar with the text 'Check out what module collections you have'. A red arrow points from this box to the sidebar. Another red box highlights the 'scimods' entry in the list with 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 center, a terminal window shows the command 'ml save scimods' and its output. A red box highlights the terminal window with the text 'Restore the "scimods" (science modules)'. A red arrow points from this box to the terminal window. A modal dialog titled 'Restore collection' is open, showing a dropdown menu with 'scimods' selected. A red arrow points from this box to the dropdown menu. At the bottom of the dialog are 'Cancel' and 'Restore' buttons.

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)

Restore collection

Select a collection to restore : scimods

Cancel Restore

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-debugipy/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 ~]\$

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?

Computational Resources



Python Programming Language

- Community Hydrological Modelling Workflows: 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
- Graham is no exception: Python is fully supported on Graham

Computational Resources

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

Please note that Binder users, need to run the following command on MyBinder Session due to a bug reported in JupyterLab:

```
bash --norc --noprofile
```

See the following webpages for ongoing discussions:

<https://github.com/jupyterhub/mybinder.org-deploy/issues/3210#event-16270107011>

&

<https://github.com/jupyterhub/repo2docker/issues/1405>

Computational Resources

The screenshot shows a terminal window titled 'kasra.keshavarz1@fc1:~' with the following command and output:

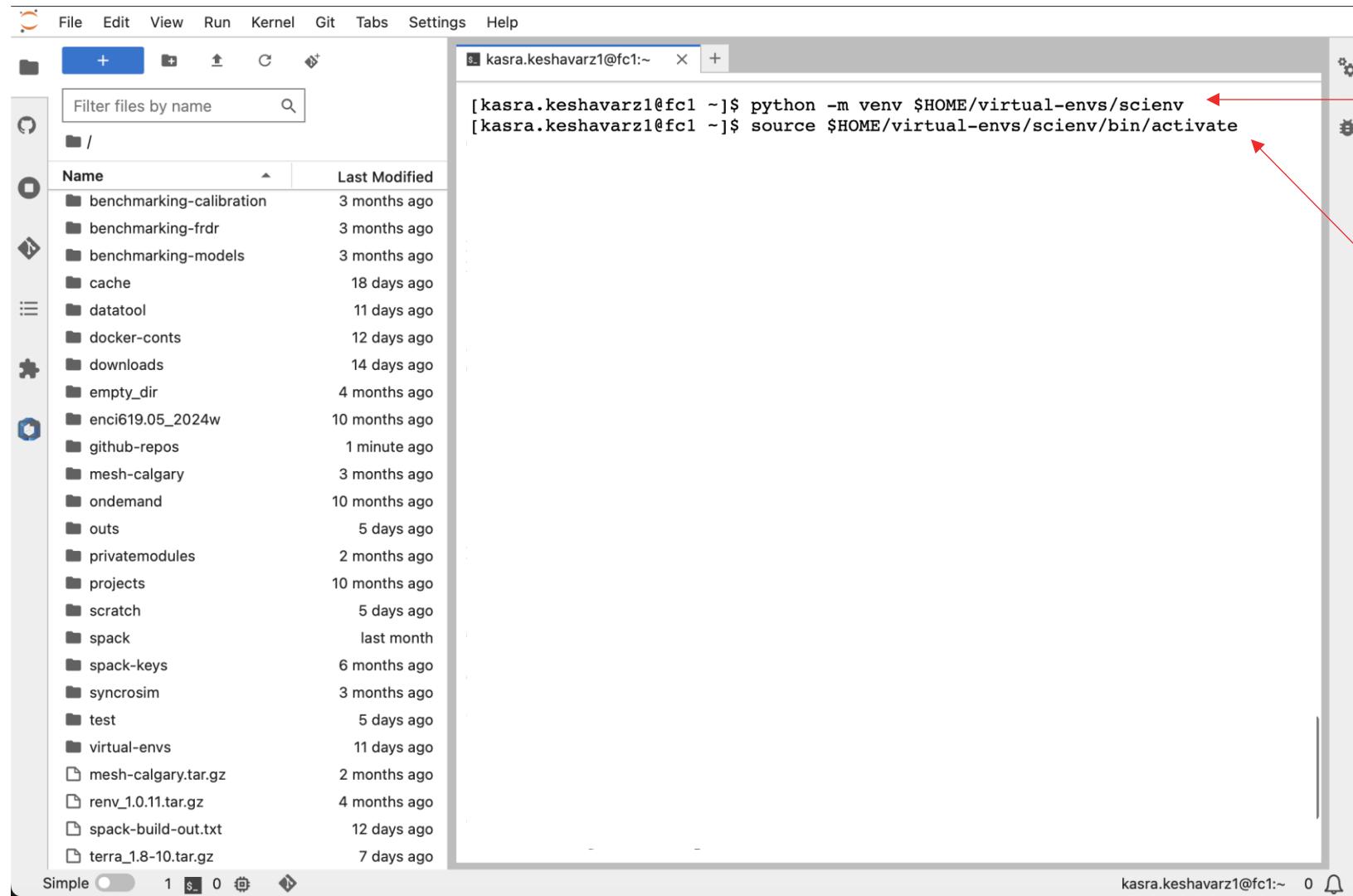
```
[kasra.keshavarz1@fc1:~]$ git clone https://github.com/kasra-keshavarz/community-modelling-workflow-training.git $HOME/github-repos/community-workflows
Cloning into '/home/kasra.keshavarz1/github-repos/community-workflows'...
remote: Enumerating objects: 160, done.
remote: Counting objects: 100% (67/67), done.
remote: Compressing objects: 100% (43/43), done.
remote: Total 160 (delta 31), reused 51 (delta 22), pack-reused 93 (from 1)
Receiving objects: 100% (160/160), 148.55 MiB | 53.06 MiB/s, done.
Resolving deltas: 100% (57/57), done.
```

The terminal is part of a desktop environment with a file manager sidebar on the left. The sidebar lists various directories and files, including 'benchmarking-calibration', 'benchmarking-frdr', 'benchmarking-models', 'cache', 'datatool', 'docker-conts', 'downloads', 'empty_dir', 'enci619.05_2024w', 'github-repos', 'mesh-calgary', 'ondemand', 'outs', 'privatemodules', 'projects', 'scratch', 'spack', 'spack-keys', 'syncrosim', 'test', 'virtual-envs', and several tar.gz files like 'mesh-calgary.tar.gz', 'renv_1.0.11.tar.gz', 'spack-build-out.txt', and 'terra_1.8-10.tar.gz'. The file manager also has a search bar and a filter by name feature.

"clone"-ing the repository

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

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
```

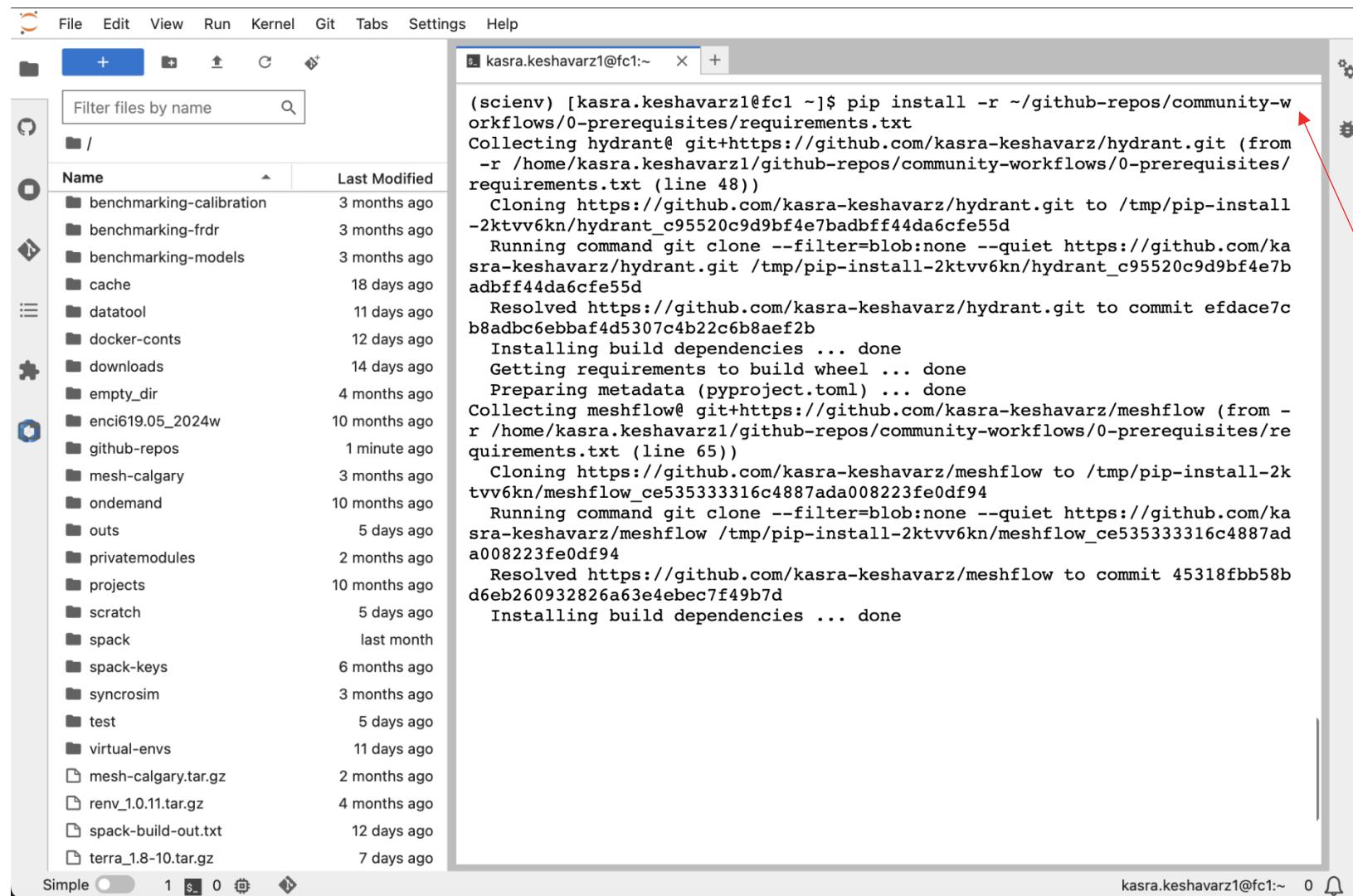
A red arrow points from the text "Creating an empty Virtual Environment" to the first command in the terminal. Another red arrow points from the text "Activating the empty Virtual Environment" to the second command.

Creating an empty Virtual Environment

Activating the empty Virtual Environment

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

Computational Resources



```
(scienv) [kasra.keshavarz1@fc1:~]$ pip install -r ~/github-repos/community-workflows/0-prerequisites/requirements.txt
Collecting hydrant@ git+https://github.com/kasra-keshavarz/hydrant.git (from -r ~/github-repos/community-workflows/0-prerequisites/requirements.txt (line 48))
  Cloning https://github.com/kasra-keshavarz/hydrant.git to /tmp/pip-install-2ktvv6kn/hydrant_c95520c9d9bf4e7badbff44da6cfe55d
    Running command git clone --filter=blob:none --quiet https://github.com/kasra-keshavarz/hydrant.git /tmp/pip-install-2ktvv6kn/hydrant_c95520c9d9bf4e7badbff44da6cfe55d
      Resolved https://github.com/kasra-keshavarz/hydrant.git to commit efdace7cb8adbc6ebbafe4d5307c4b22c6b8aef2b
        Installing build dependencies ... done
        Getting requirements to build wheel ... done
        Preparing metadata (pyproject.toml) ... done
Collecting meshflow@ git+https://github.com/kasra-keshavarz/meshflow (from -r ~/github-repos/community-workflows/0-prerequisites/requirements.txt (line 65))
  Cloning https://github.com/kasra-keshavarz/meshflow to /tmp/pip-install-2ktvv6kn/meshflow_ce53533316c4887ada008223fe0df94
    Running command git clone --filter=blob:none --quiet https://github.com/kasra-keshavarz/meshflow /tmp/pip-install-2ktvv6kn/meshflow_ce53533316c4887ada008223fe0df94
      Resolved https://github.com/kasra-keshavarz/meshflow to commit 45318fbb58bd6eb260932826a63e4ebec7f49b7d
        Installing build dependencies ... done

```

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 this 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

Below this, under "AVAILABLE MODULES (1233)", are:

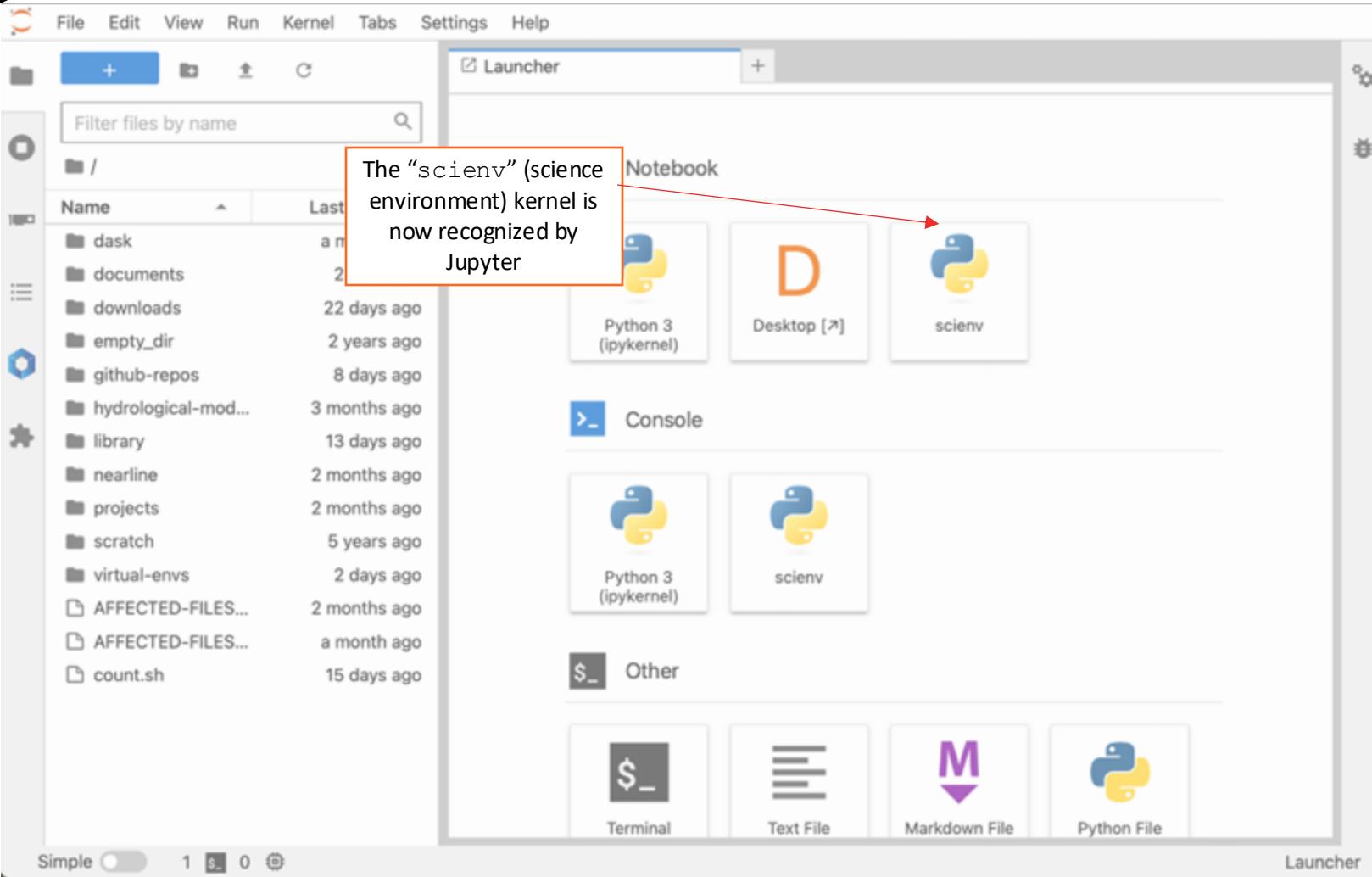
- StdEnv/2016.4
- StdEnv/2018.3
- StdEnv/2023
- slurm/2021

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 highlights the message: "“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:



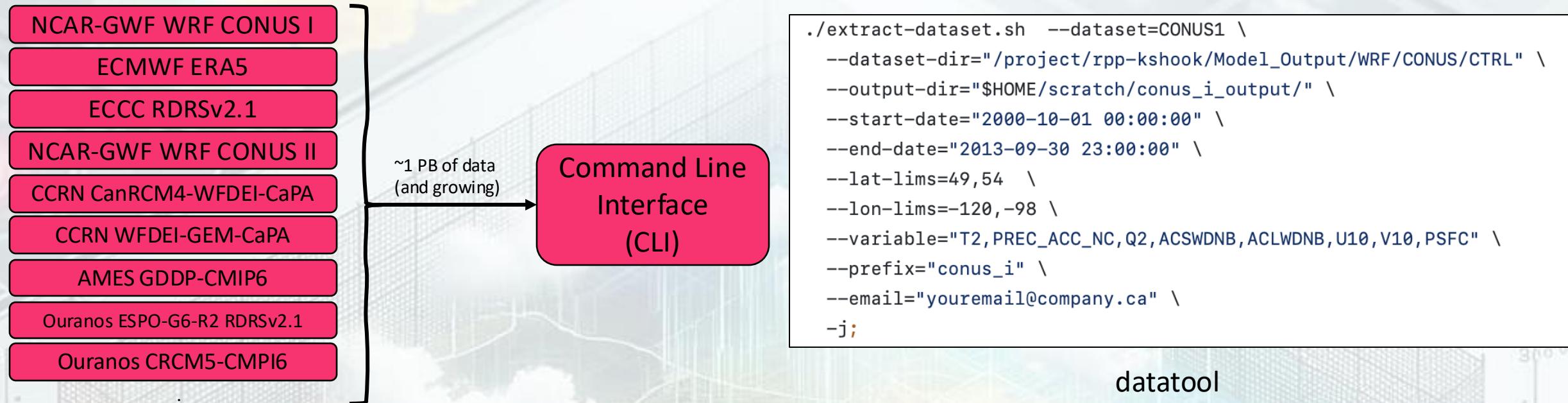
The "scie nv" (science environment) kernel is now recognized by Jupyter

The screenshot shows the JupyterLab interface. On the left is a file browser listing various directories and files. In the center is the Launcher, which contains several icons for different kernels and environments. A callout box highlights the "scie nv" icon in the "Notebook" section, with the text "The 'scie nv' (science environment) kernel is now recognized by Jupyter". Other icons in the Launcher include "Python 3 (ipykernel)", "Desktop [↗]", "Console", and another "scie nv" icon. Below the Launcher are sections for "Terminal", "Text File", "Markdown File", and "Python File". The bottom of the screen shows a toolbar with "Simple" mode, a code cell, and other controls.

Outline

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

Setting up ECCC's National Water Model—MESH

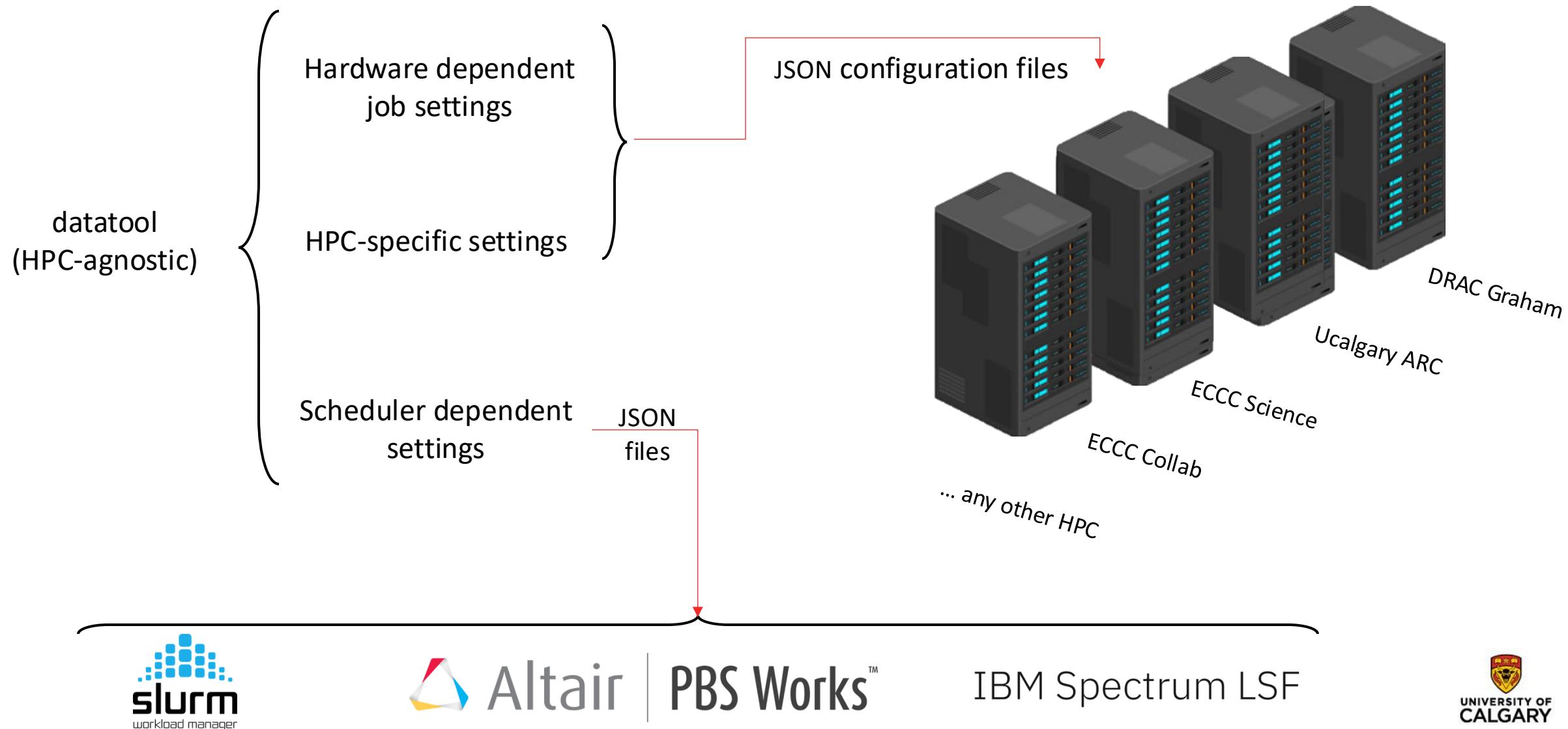


Steps:

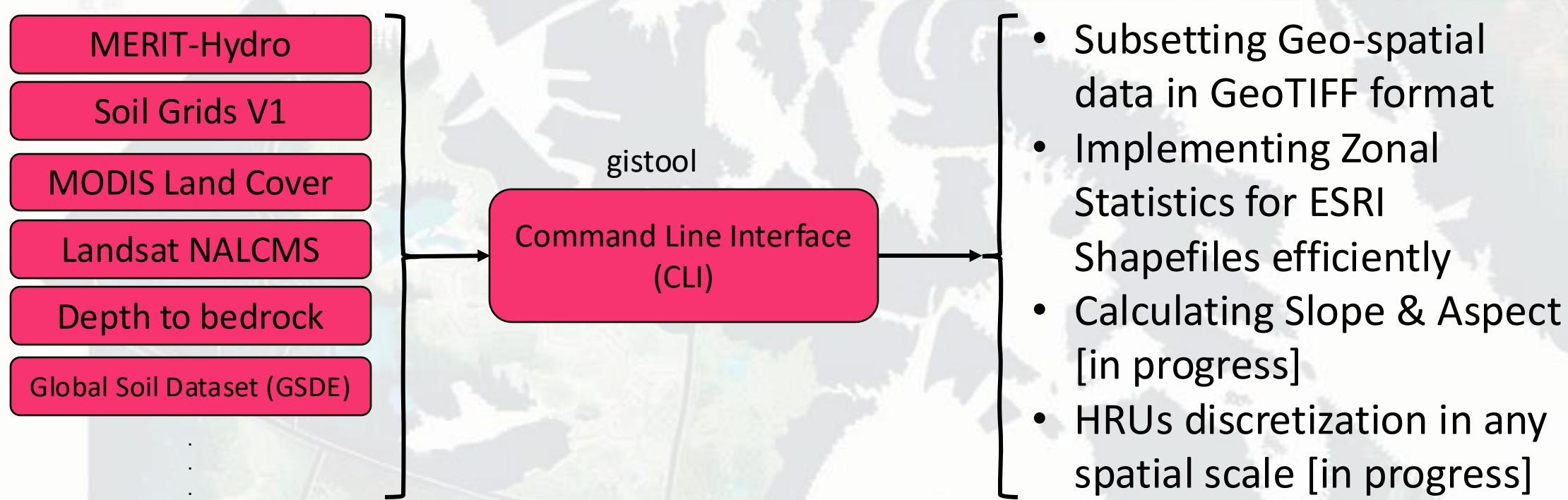
1. Login to cluster of choice,
2. Make sure you have access to the designated project space and allocation,
3. Clone the remote GitHub repository,
4. Subset any spatial and temporal extents of interest for any of the available datasets with only **one line of code**.

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

Setting up ECCC's National Water Model—MESH



Setting up ECCC's National Water Model—MESH

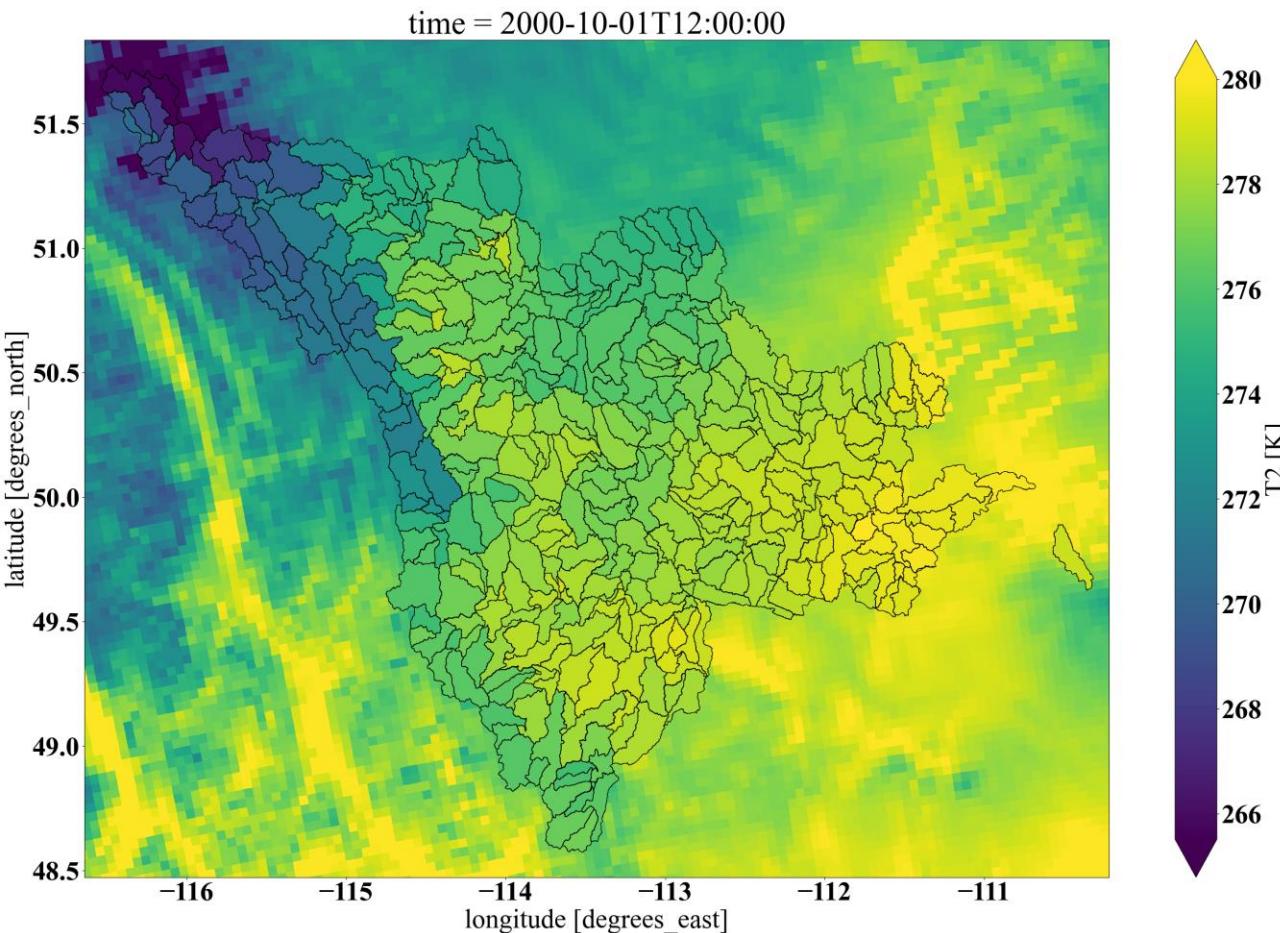


Steps:

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



```
kasras-mbp:easy more ShervanGharari$ easy more
Usage: easy more [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

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

Any questions so far?

Setting up ECCC's National Water Model—MESH

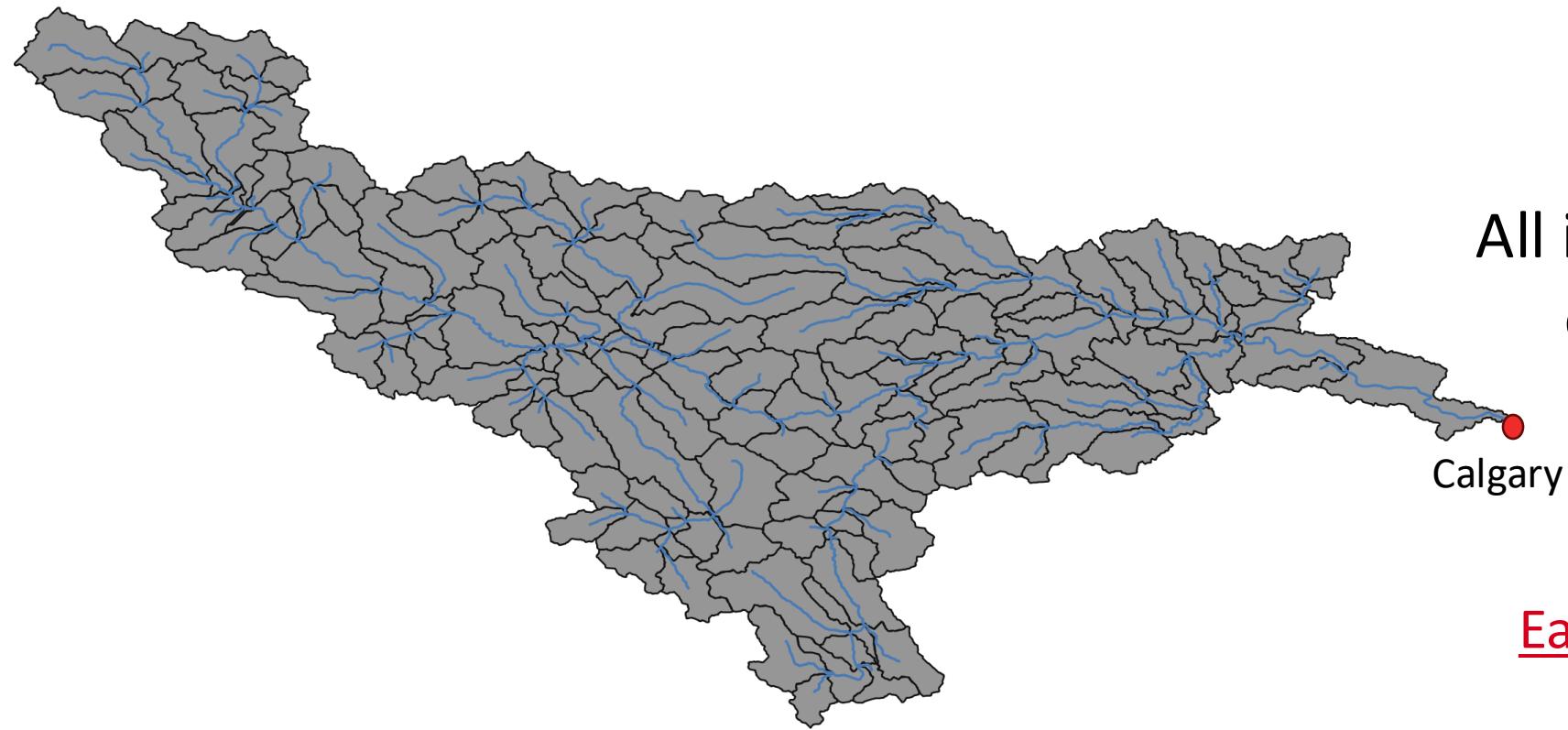
Configuring ECCC's MESH with the following datasets for the Bow River at Calgary:

Meteorological Forcing Dataset	Landcover Dataset	Geospatial Fabric Dataset	Study Area
Regional Deterministic Reforecast System version 2.1 (RDRSv2.1)	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



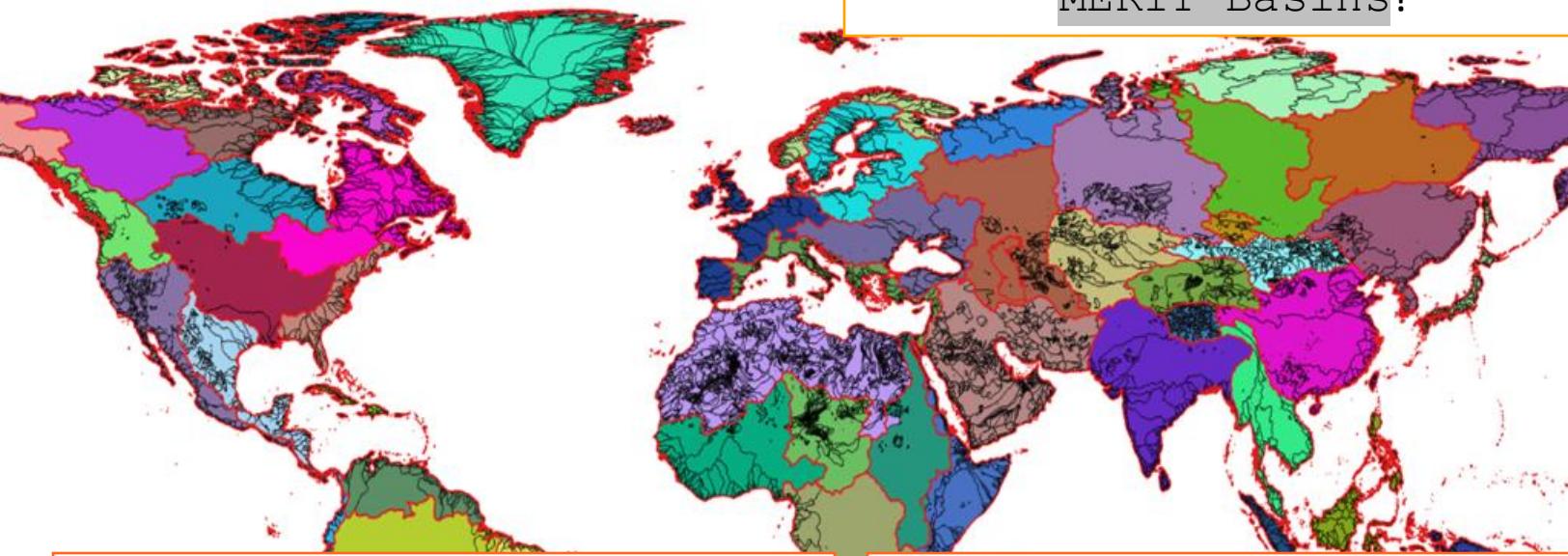
Bow River at Calgary

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/CH-
Earth/community-modelling-
workflow-training](https://github.com/CH-Earth/community-modelling-workflow-training)

Setting up ECCC's National Water Model—MESH



What information is provided through MERIT-Basins?

- 25 km² drainage area threshold for channels
- Corrected/refined basin/region definitions
- Below 25 km² 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

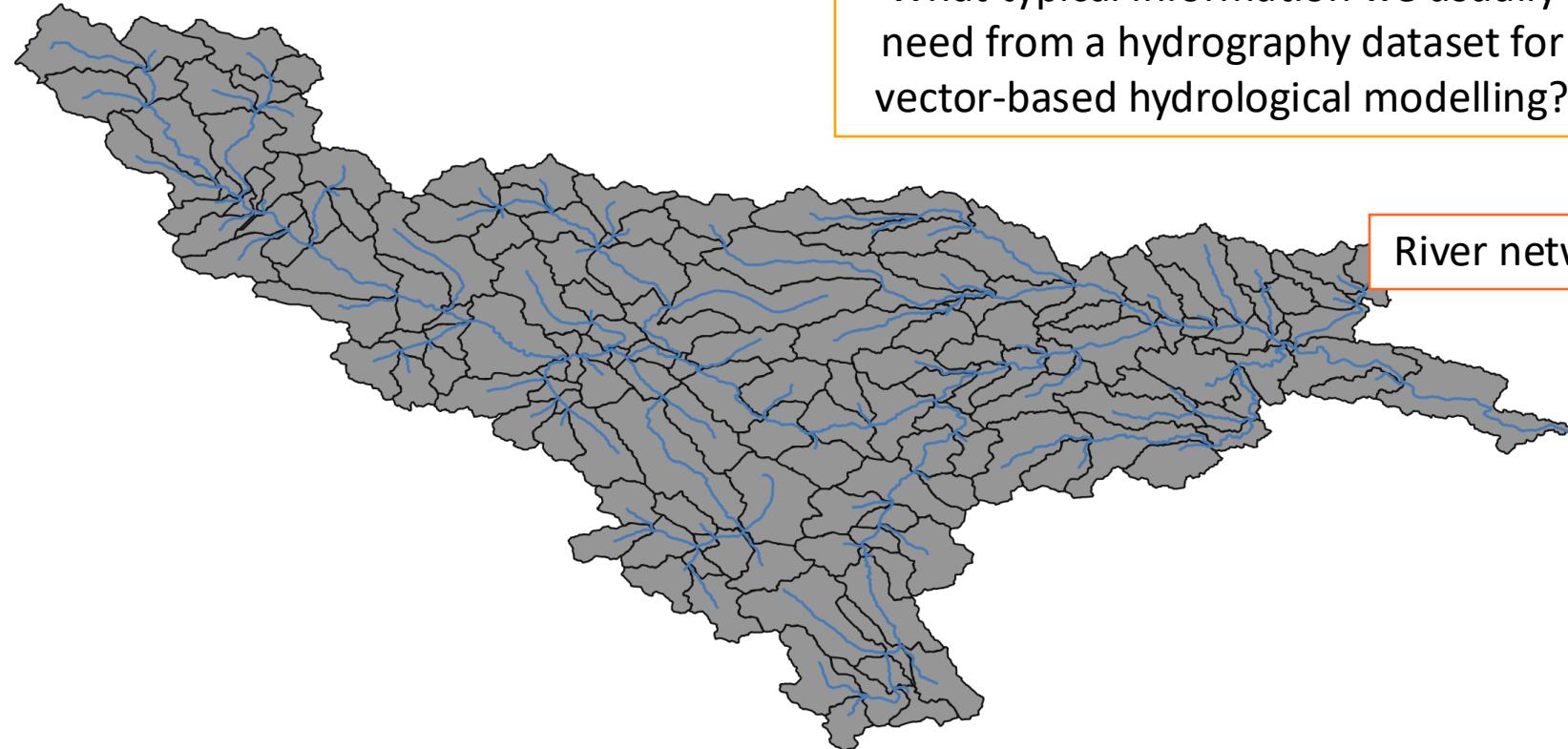
Subbasins:

- 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:

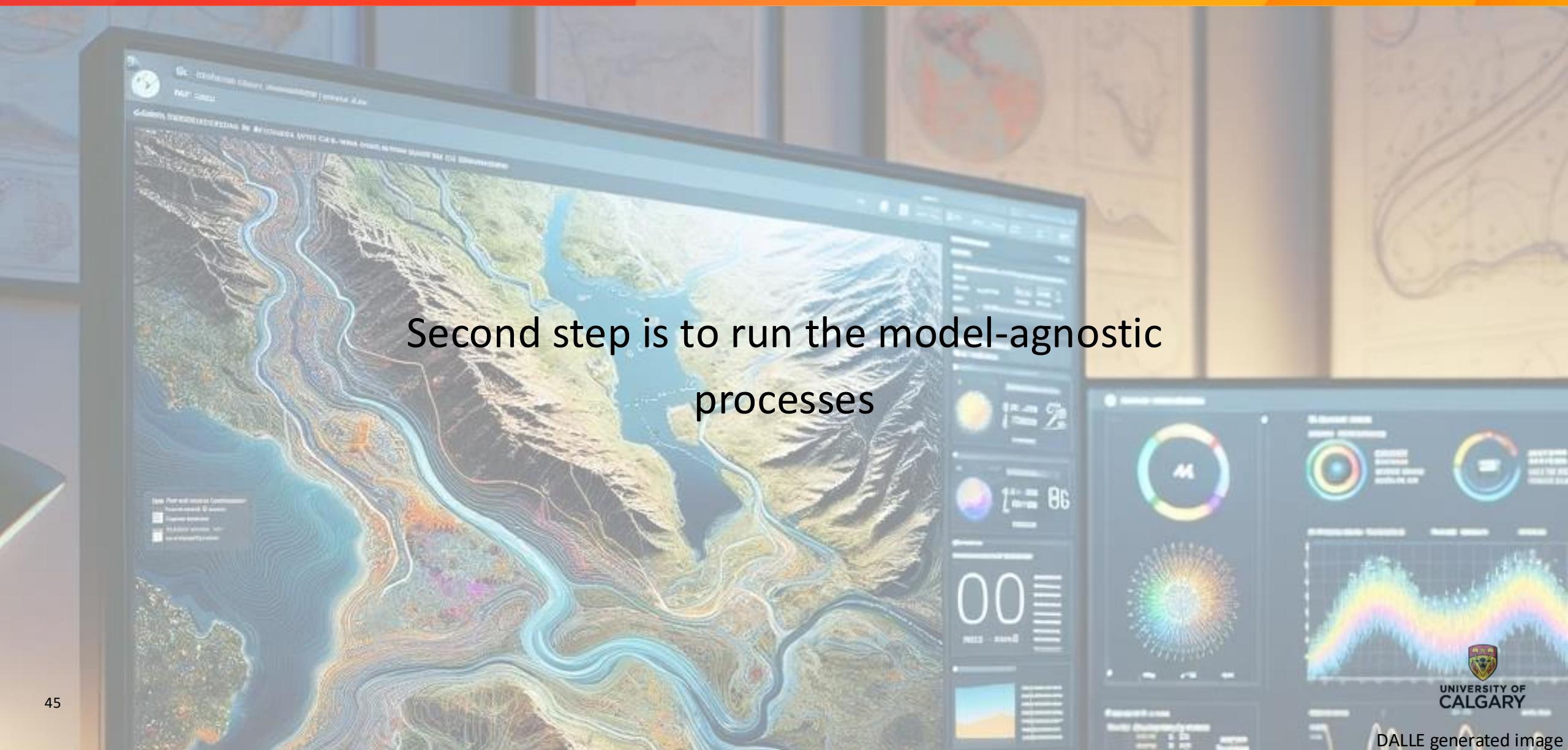
- Slope [optional]
- Width [optional]
- etc.

Subbasins:

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

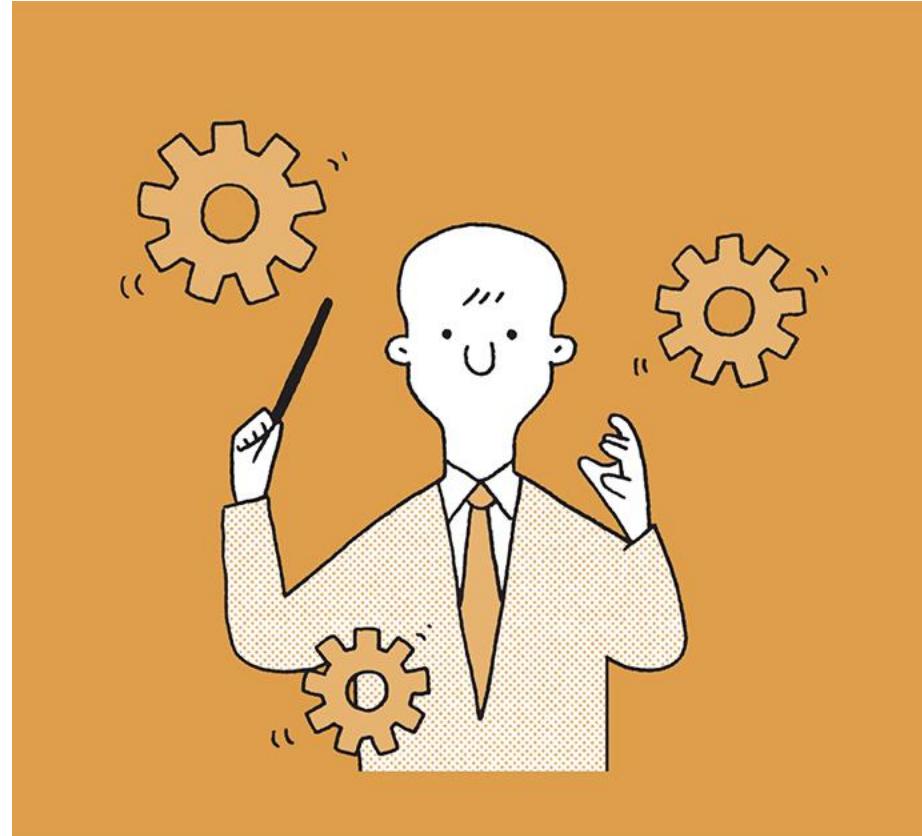


Setting up ECCC's National Water Model—MESH



Second step is to run the model-agnostic processes

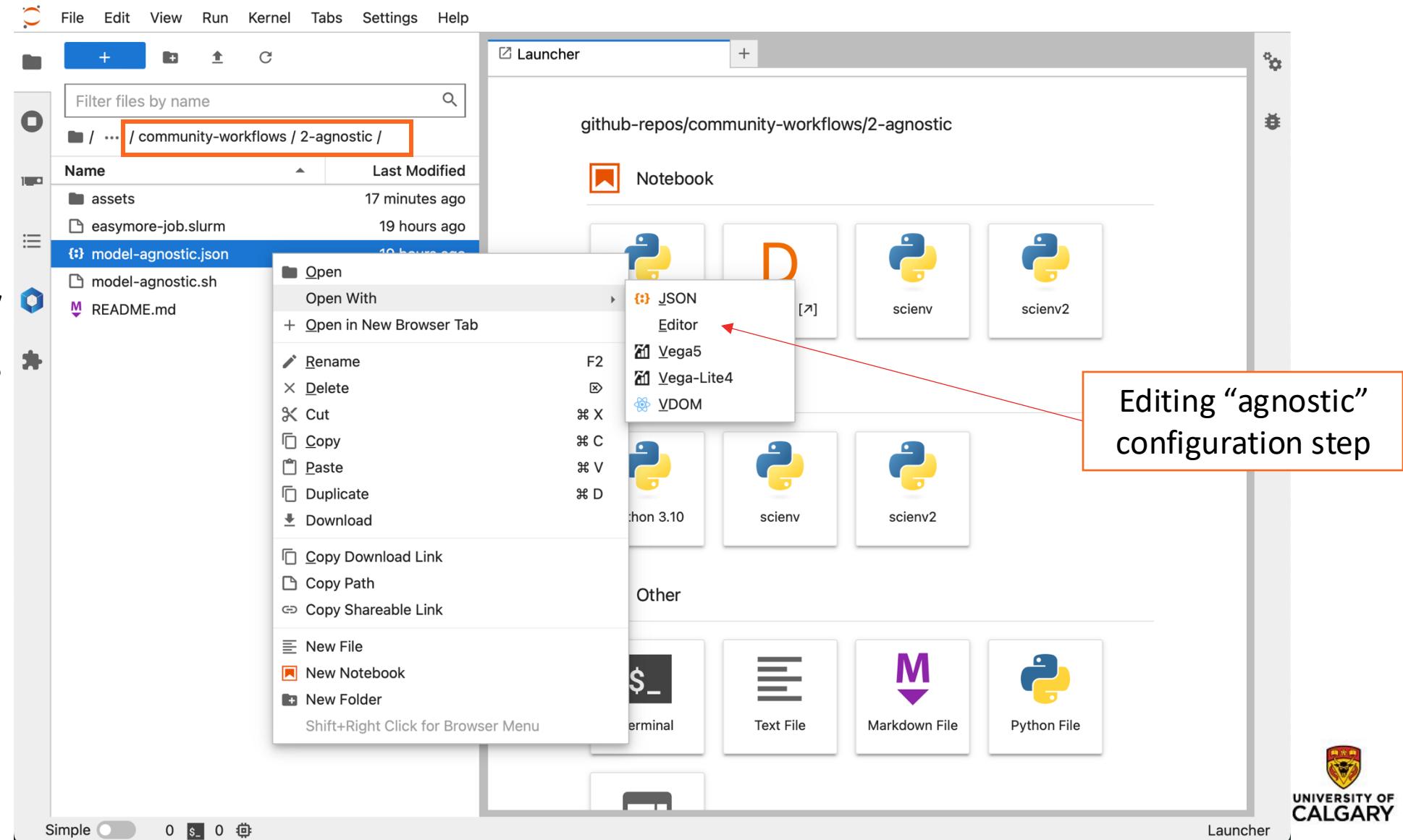
Setting up ECCC's National Water Model—MESH



- 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

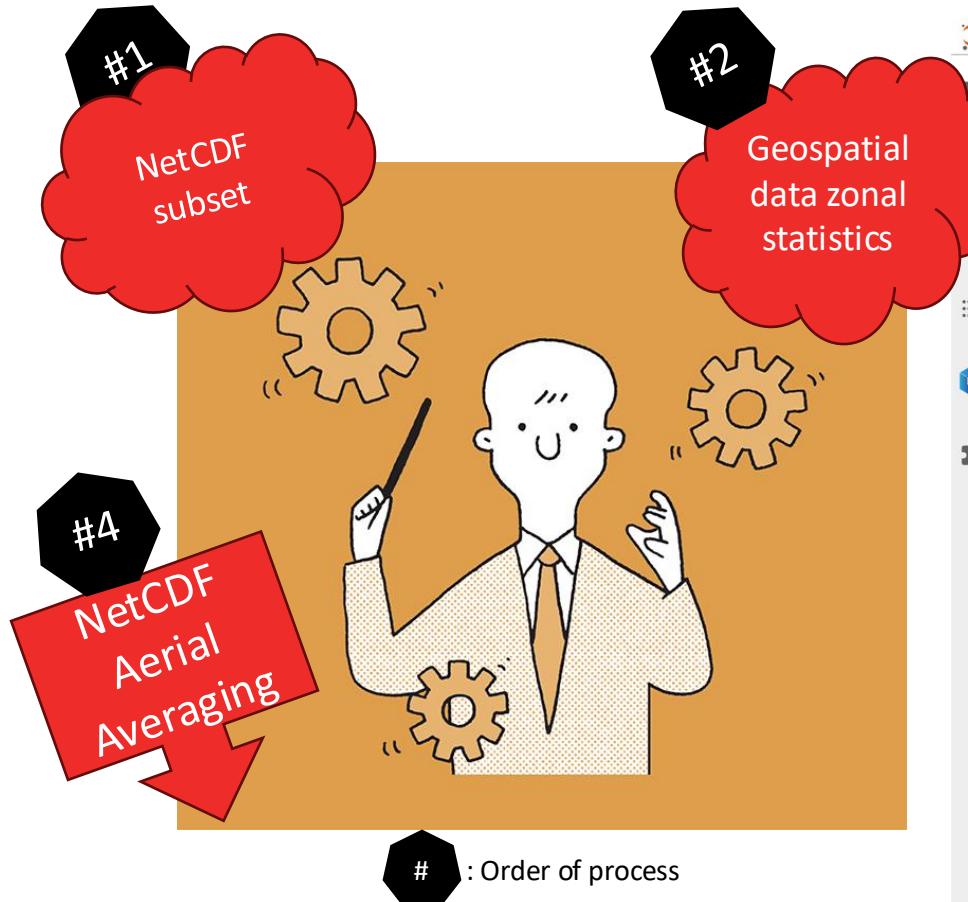
The screenshot shows a Jupyter Notebook interface with a file browser on the left and a code editor on the right. The code editor displays the contents of `model-agnostic.json`. The file defines an orchestrator configuration for an 'agnostic' step, containing executable paths and arguments.

```
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     }]
```

Annotations:

- Executable paths**: Points to the `exec` section of the JSON file.
- Orchestrator configuration file**: Points to the main block of the JSON file.
- Executable arguments and options (empty options are ignored)**: Points to the `args` section, specifically the `met` array and its nested objects.
- This is the only file for the “agnostic” step**: Points to the bottom of the JSON file.

Setting up ECCC's National Water Model—MESH



Automated and ordered job submission to HPC

A screenshot of a terminal window showing a file browser and a JSON configuration file. The file browser shows a directory structure with files: assets, easymore-job.slurm, model-agnostic.json, model-agnostic.sh, and README.md. The model-agnostic.json file is selected and shown in the code editor. The code in the model-agnostic.json file is as follows:

```
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

Any questions so far?

Setting up ECCC's National Water Model—MESH

The screenshot shows the JupyterLab interface. On the left, the file browser displays a directory structure under '/github-repos / community-workflows /'. A red arrow points to the folder icon in the sidebar. A callout box on the left says 'Navigate using JupyterLab file browser'. The main area shows the 'Launcher' tab selected, displaying various notebooks and environments. The 'Notebook' section includes entries for '0-prerequisites', '1-geofabric', '2-agnostic', '3-specific', 'LICENSE', and 'README.md'. The 'Console' section shows 'Python 3.10' and 'scienv' environments. The 'Other' section shows '\$_' and 'M' icons.

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

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 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 in the directory "... / community-workflows / 3-specific /". The files listed are "setting_files" (modified 20 hours ago), "meshflow_bb.ipynb" (modified seconds ago, highlighted in blue), and "README.md" (modified 20 hours ago). A red arrow points from the text in the callout box to the "meshflow_bb.ipynb" file in the file browser. The right pane shows the content of the "meshflow_bb.ipynb" notebook. The title of the notebook is "meshflow_bb.ipynb". The main content is titled "Build MESH Model Setup for the Bow River at Banff". It contains text explaining the use of the `MESHFlow` Python package to build a `MESH` model setup for the Bow River at Banff, and a note about keeping geospatial fabric files next to the "agnostic" step's outputs. It includes a command-line code cell for copying files and a code cell for importing necessary libraries. The text "Let's start by importing the necessary libraries:" is followed by a code cell with four lines of Python code. A red arrow points from the text in the callout box to the word "meshflow" in the second line of the code. The code cell continues with more imports and configuration settings. A red arrow points from the text in the callout box to the variable "work_path" in the first line of the configuration section. Callout boxes with arrows point to specific parts of the code and text in the notebook. One callout box highlights the `meshflow` package and its purpose. Another callout box highlights the `work_path` variable and its meaning.

The specific part is written into a Jupyter Notebook
(it can be written as a script or any other form that you prefer)

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

53

Simple 0 \$ 1 scienv | Idle

Mode: Command Mode: Command Ln 1, Col 1 meshflow_bb.ipynb

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

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

```
[3]: # main work path - modify
      work_path = '/home/kasra545/scratch/bb-models/'
```

```
[4]: # using meshflow==v0.1.0-dev1
      # modify each segment to match your settings
      config = {
          'riv': os.path.join(work_path, 'geofabric', 'bb_rivers.shp'),
          'cat': os.path.join(work_path, 'geofabric', 'bb_subbasins.shp'),
          'landcover': os.path.join(work_path, 'gistool-outputs', 'bb_mode'),
          'forcing_files': os.path.join(work_path, 'easymore-outputs'),
          'forcing_vars': '# does the variable list match those of the "'
```

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 under 'community-workflows / 3-specific /'. The right pane is a code editor with a tab titled 'meshflow_bb.ipynb'.

```
[2]: # main work path - modify
work_path = '/home/kasra545/scratch/bb-models/'

# using meshflow==v0.1.0-dev1
# modify each segment to match your settings
config = {
    'riv': os.path.join(work_path, 'geofabric', 'bb_rivers.shp'),
    'cat': os.path.join(work_path, 'geofabric', 'bb_subbasins.shp'),
    'landcover': os.path.join(work_path, 'gistool-outputs', 'bb_mode'),
    'forcing_files': os.path.join(work_path, 'easymore-outputs'),
    'forcing_vars': [ # does the variable list, match those of the forcing files?
        "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",
    ],
    'forcing_units': { # Here, enter RDRS's original variable units
        'RDRS_v2.1_P_P0_SFC': 'millibar',
        'RDRS_v2.1_P_HU_09944': 'kg/kg',
        'RDRS_v2.1_P_TT_09944': 'celsius',
        'RDRS_v2.1_P_UVC_09944': 'knot',
        'RDRS_v2.1_A_PR0_SFC': 'm/hr',
        'RDRS_v2.1_P_FB_SFC': 'W/m^2',
        'RDRS_v2.1_P_FI_SFC': 'W/m^2',
    },
    'forcing_to_units': { # And here, the units that MESH needs to re-project
        'RDRS_v2.1_P_UVC_09944': 'm/s',
        'RDRS_v2.1_P_FI_SFC': 'W/m^2',
        'RDRS_v2.1_P_FB_SFC': 'W/m^2',
        'RDRS_v2.1_A_PR0_SFC': 'mm/s',
        'RDRS_v2.1_P_P0_SFC': 'pascal',
        'RDRS_v2.1_P_TT_09944': 'kelvin',
        'RDRS_v2.1_P_HU_09944': 'kg/kg',
    },
}
```

Annotations pointing to specific parts of the code:

- 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

Page footer: 54

Page footer: Simple 0 \$ 1 scienv | Idle Mode: Command Ln 1, Col 1 meshflow_bb.ipynb

Page footer: UNIVERSITY OF CALGARY

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 in the editor is as follows:

```
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: },
```

Annotations explain the code:

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

Page number 55 is in the bottom left corner, and the University of Maryland logo is in the bottom right corner.

Setting up ECCC's National Water Model—MESH

The screenshot shows a Jupyter Notebook interface with two main 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 a tab titled 'meshflow_bb.ipynb'. The code is a Python dictionary configuration for a 'meshflow' object. Arrows from callout boxes point to specific parts of the code:

- An arrow points to the 'ChnlSlope' key in the 'ddb_to_units' section, which is annotated with 'Units of variables in the “drainage database” file that MESH requires'.
- An arrow points to the 'ChnlLength' key in the same section, which is annotated with 'Minimum values of “drainage database” MESH file'.
- An arrow points to the 'gru_dim' key, which is annotated with 'Renaming GRU dimension name in the MESH “drainage database” file to satisfy diverse versions of MESH'.
- An arrow points to the 'hru_dim' key, which is annotated with 'Renaming sub-basin dimension name in the MESH “drainage database” file if needed'.
- An arrow points to the 'outlet_value' key, which is annotated with 'The value of the outlet river segments created using “Hydrant” (Hydrant default is -9999)'.

Below the code editor, there are several text blocks and arrows:

- A box labeled 'Initiate an instance of the MESHWorkflow object' has an arrow pointing to the line of code: [3]: `exp1 = meshflow.MESHWorkflow(**config)`.
- A box labeled 'And running the “model-specific” step' has an arrow pointing to the line of code: [4]: `exp1.run()`.
- A text block 'We can build an “instance” of the workflow class:' is followed by the code line [3]: `exp1 = meshflow.MESHWorkflow(**config)`.
- A text block 'And, we can run it using:' is followed by the code line [4]: `exp1.run()`.
- A text block 'Once the run is finished, we can checkout the forcing and drainage database file:' is at the bottom of the notebook.

At the bottom of the interface, there are status indicators: 'Simple' (button), '0 \$ 1' (button), 'scienv | Idle' (status), 'Mode: Command' (status), 'Ln 1, Col 1' (status), and 'meshflow_bb.ipynb' (status).

Any questions so far?

Setting up ECCC's National Water Model—MESH

Switching to the relevant Jupyter Notebook
webpage

jupyterlab

Outline

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

Brief background: Yukon River basin study

Research Question:

How can we optimally represent hydrological processes across diverse landscapes in modern process-based hydrological models?

Observations:

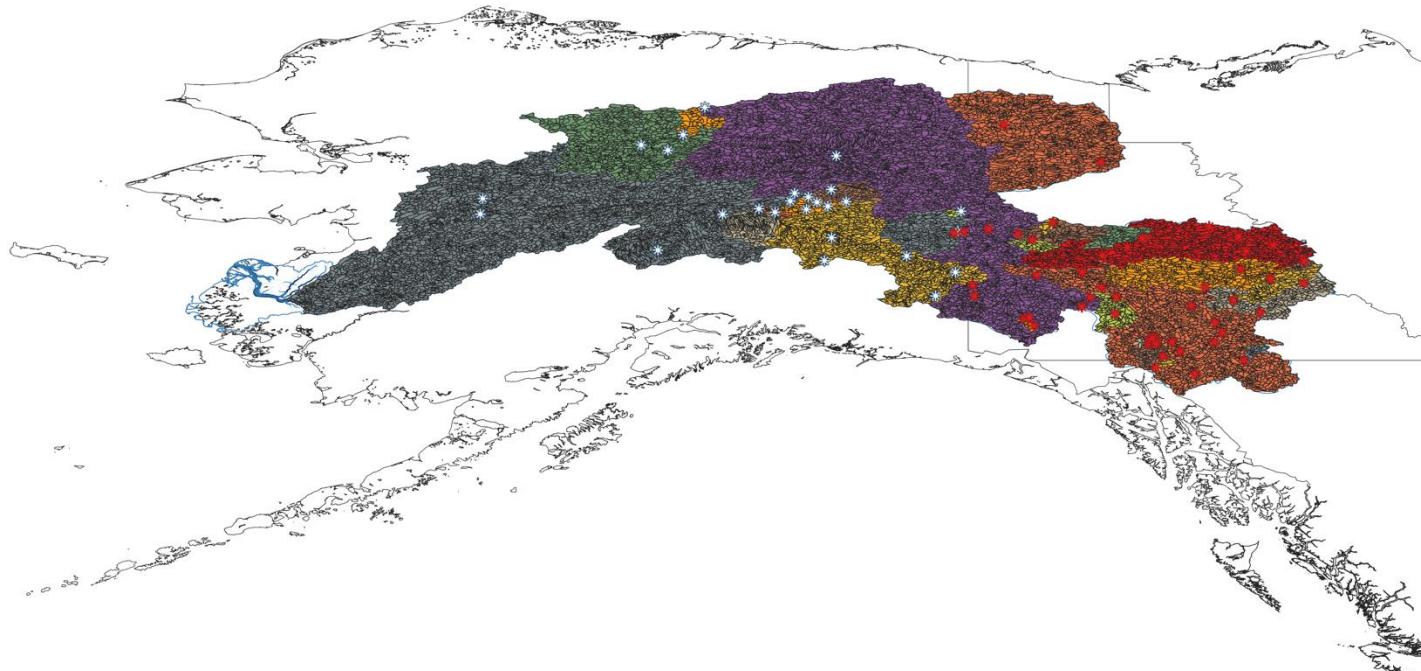
- 53 Stream gauges
- 86 Snow observation stations
- 35 sub-basins with in-situ snow observations

Model: SUMMA

- Flexible framework for model comparison
- 40 processes with multiple representation options
 - Space of possible combinations $> 10^7$
- 126 parameters to optimize for each combination

External decisions:

- Domain representation
- Forcing data
- Attribute data
- Etc.



Brief background:

Research Question

How can we optimally represent processes across diverse landscape hydrological models?

Observations:

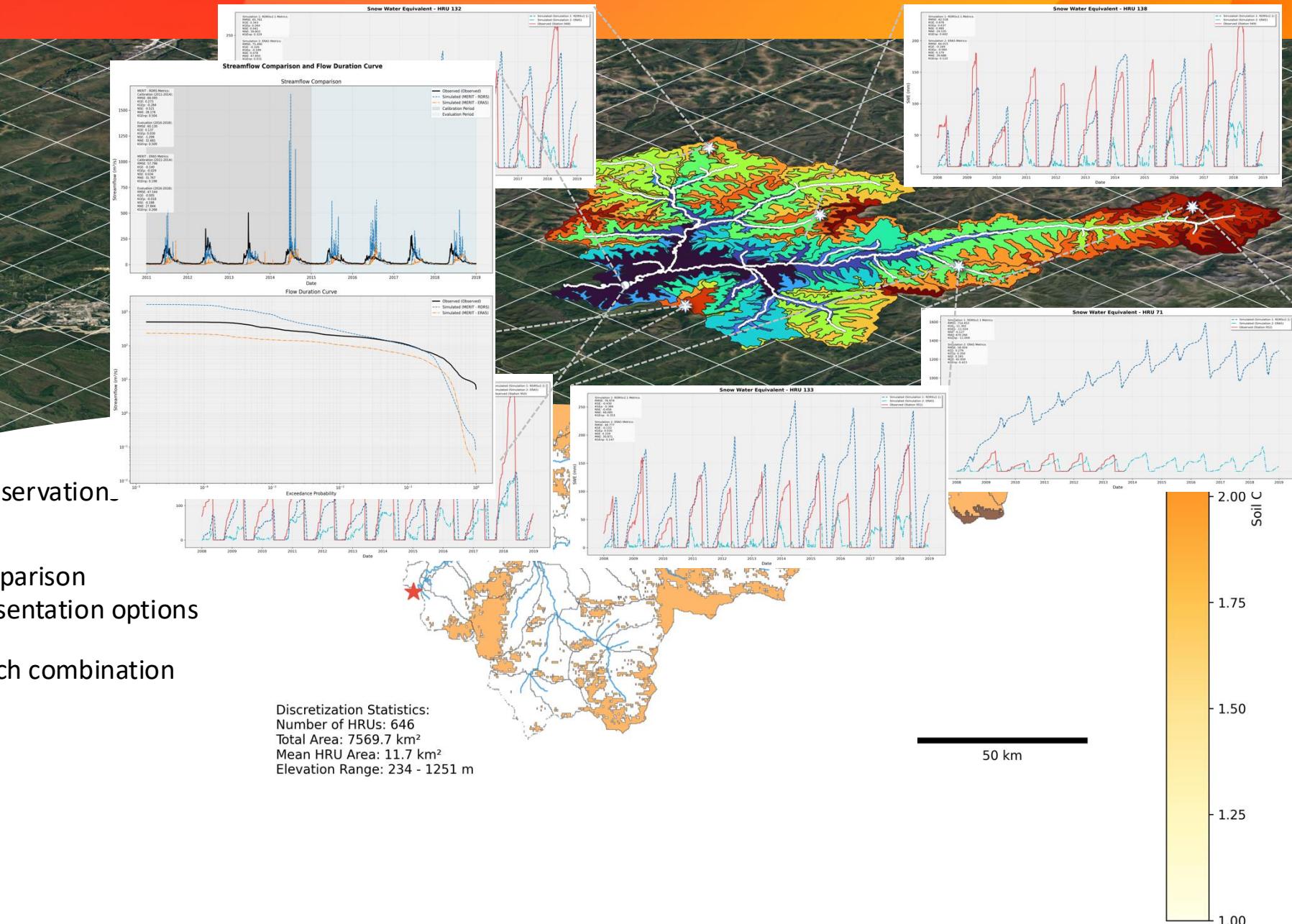
- 53 Stream gauges
- 86 Snow observation stations
- 35 sub-basins with in-situ snow observation

Model: SUMMA

- Flexible framework for model comparison
- ~40 processes with multiple representation options
 - Space of possible combinations $> 10^7$
- 126 parameters to calibrate for each combination

External decisions:

- Domain representation
- Forcing data
- Attribute data
- Etc.



CONFLUENCE: Community Optimization Nexus For Large-domain Understanding of Environmental Networks and Computational Exploration

Problem:

The inherent complexity of modern process based hydrological models poses substantial challenges to researchers and operators, particularly due to the high dimensionality of the modelling decision space

Solution:

Automation of routine task through seamless integration of existing workflows with clear separation of technical implementation and scientific decision making.

Implementation:

- Single configuration file (YAML)
- Central workflow orchestration script
- Central workflow logging
- Modular design
- Extendable code base



CONFLUENCE



UNIVERSITY OF
CALGARY

CONFLUENCE: Community Optimization Nexus For Large-domain Understanding of Environmental Networks and Computational Exploration

Problem:

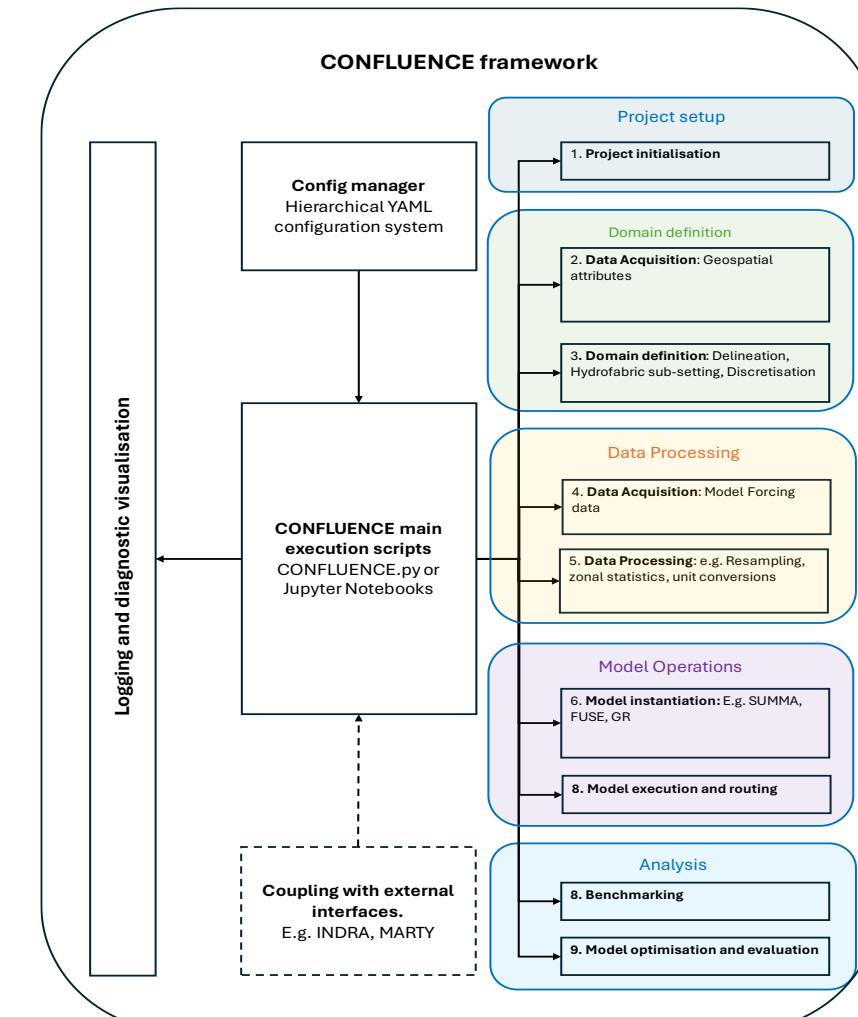
The inherent complexity of modern process based hydrological models poses substantial challenges to researchers and operators, particularly due to the high dimensionality of the modelling decision space

Solution:

Automation of routine task through seamless integration of existing workflows with clear separation of technical implementation and scientific decision making.

Implementation:

- Single configuration file (YAML)
- Central workflow orchestration script
- Central workflow logging
- Modular design
- Extendable code base



CONFLUENCE Workflow (Eythorasson et al., In prep.)

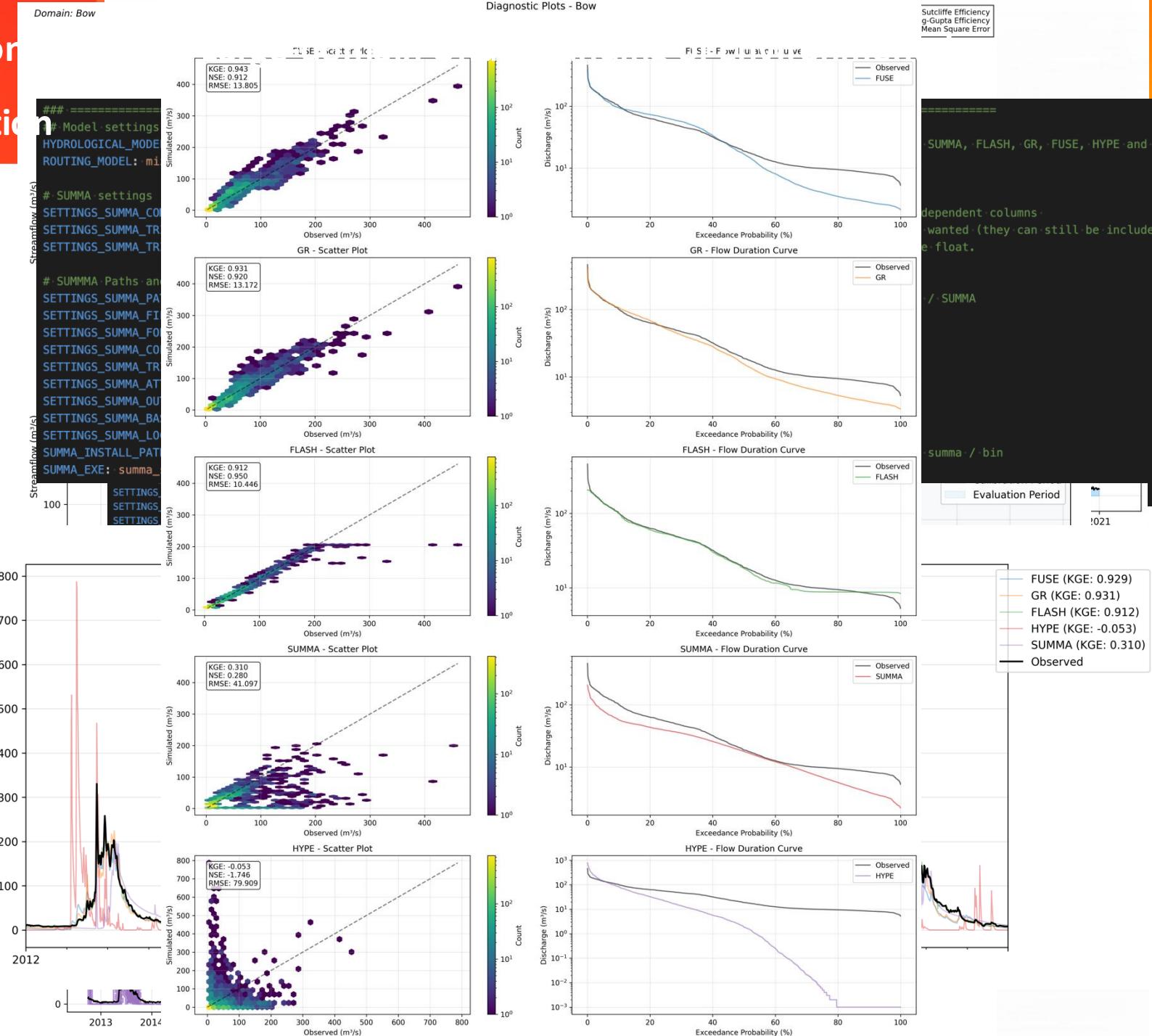


UNIVERSITY OF
CALGARY

CONFLUENCE: Community Optimization Networks and Computational Exploration

Networks and Computational Exploration

- Configuration examples
 - Delineation
 - Discretization
 - Benchmarking
 - Model initiation
 - Model comparison



INDRA: Intelligent Network for Dynamic River Analysis

New problem:

Computational resource restrictions prohibit exhaustive search of the modelling decision space using current optimization algorithms

Solution:

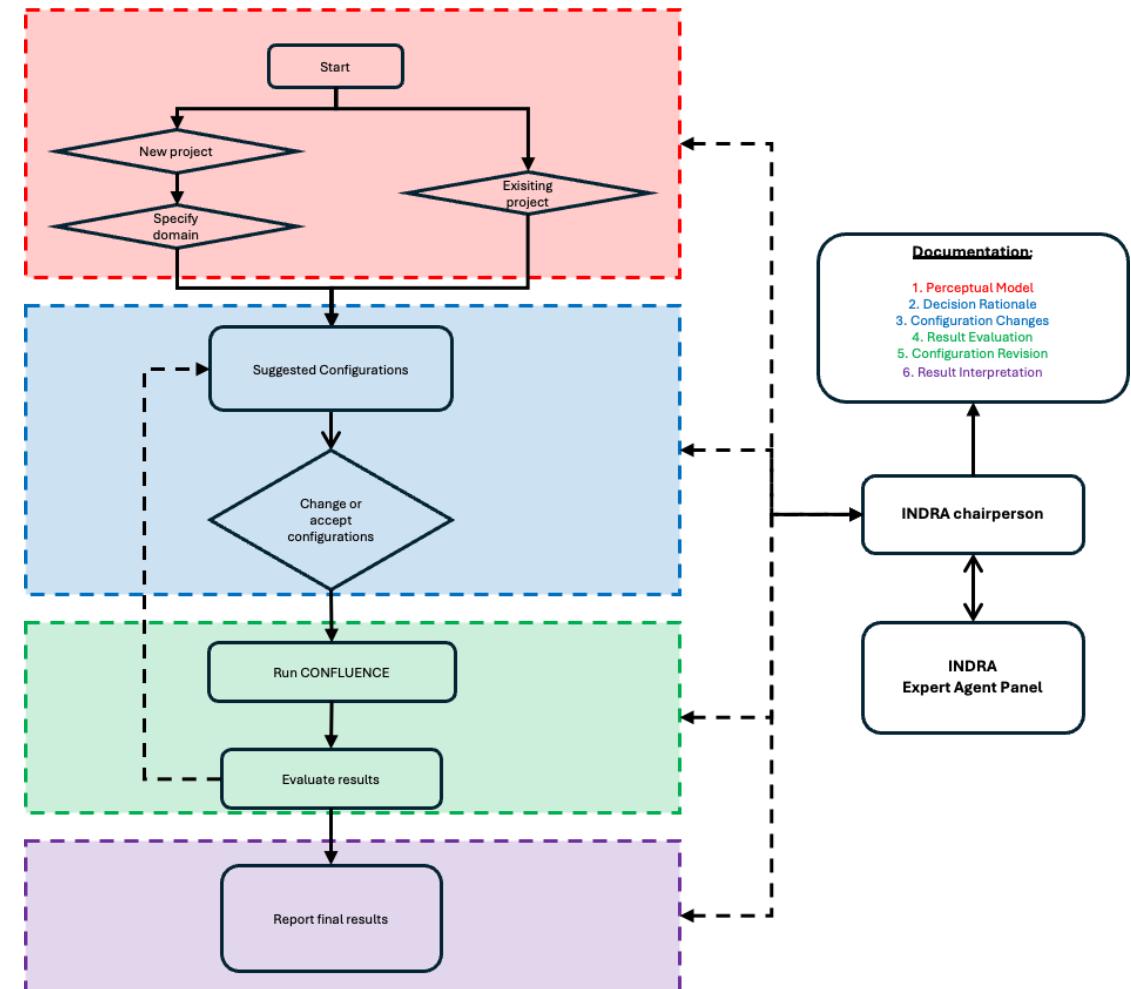
Reduce dimensionality of decision space by selection of a domain expertise informed subset. Leverage novel developments in LLM contextual understanding and multi-agentic AI for decision support and automation of non-essential technical implementation through configuration generation.



INDRA: Intelligent Network for Dynamic River Analysis

Implementation:

- CONFLUENCE coupling with a multi agent framework orchestrated by central LLM Moderator
- Moderator interacts with human operator and analyses natural language input to provide actionable decision support
- Moderator can create a panel of project appropriate AI subject matter experts to deliberate on each decision
- Moderator summarizes panel discussions and generates a CONFLUENCE config file with the suggested settings for the operator to consider.
- If approved moderator executes CONFLUENCE.
- After completed simulation Moderator generates a panel to analyze the results and suggest refinements
- Repeat until results are acceptable



INDRA (Intelligent Network for Dynamic River Analysis, (Eythorsson & Clark, 2025))



UNIVERSITY OF
CALGARY

INDRA: Intelligent Network for Dynamic River Analysis

- Perceptual model generation
 - Not all LLMs created equal
 - Fast moving technology
- Configuration rationale
 - Collaborative conversation summary
- Prompt parsing, e.g.:
 - *python INDRA.py --purpose "Please create a SUMMA model for the Congo River"*

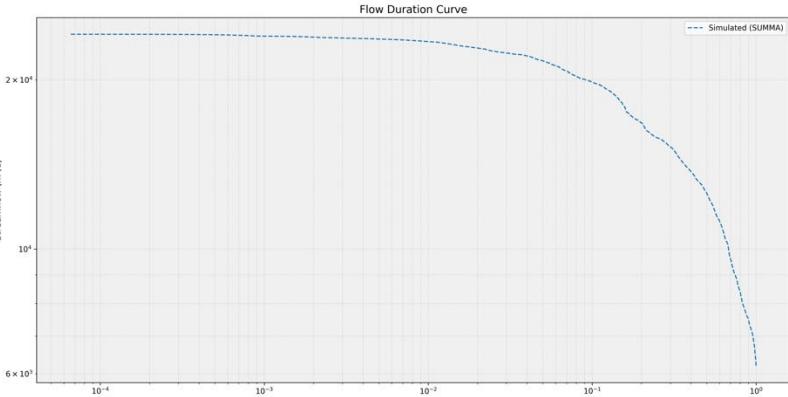
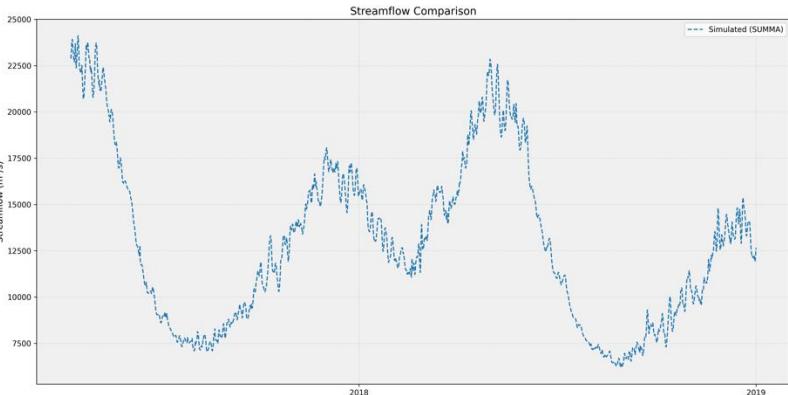
INDRA Perceptual Models Report
=====
Hydrologist Expert Perceptual Model
Perceptual Model ... ar, for Congo River Ba
The Congo River Basin is one of the largest in the world and presents unique challenges and opportunities.
Key Hydrological Processes and Interactions
1. Precipitation: The basin experiences high rainfall (Larague et al., 2001).
2. Evapotranspiration: Dense tropical rainforest covers much of the basin.

INDRA Initial Configuration Decisions for Seine

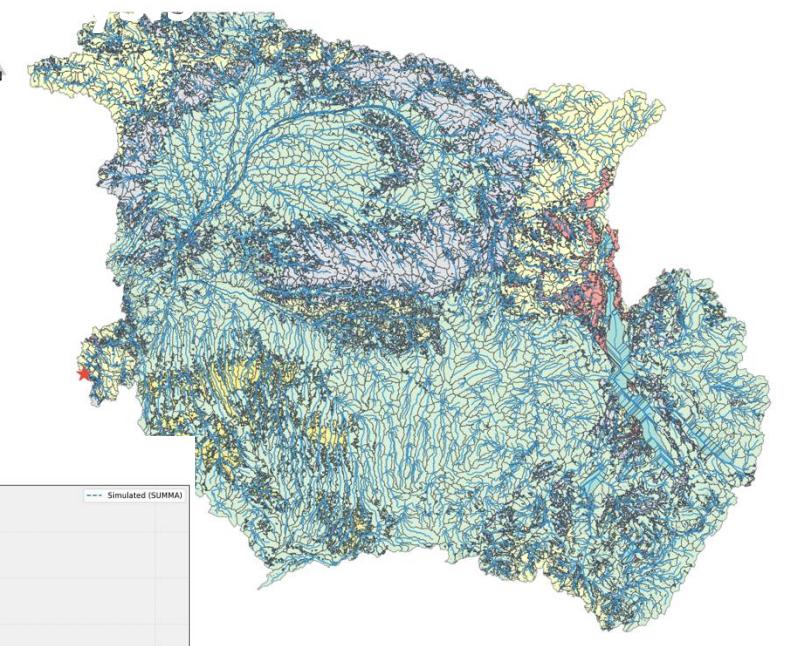
Expert-Suggested Configuration Parameters:

```
HYDROLOGICAL_MODEL: SUMMA
ROUTING_MODEL: mizuroute
FORCING_DATASET: ERA5
DOMAIN_DISCRETIZATION: elevation
ELEVATION_BAND_SIZE: 100
MIN_HRU_SIZE: 10
POUR_POINT_COORDS: 48.796667/1.687778
BOUNDING_BOX_COORDS: 49.75/-0.25/47.75/4.25
PARAMS_TO_CALIBRATE:
upperBoundHead, lowerBoundHead, upperBoundTheta, lowerBoundTheta,
z0Snow, z0Soil, Kc25, Ko25, vcmax25, canopyTop, summerAI, rootDepth,
DECISION_OPTIONS: {'soilCatTbl': ['STAS'], 'vegeParTbl': ['intertive'], 'fderivMeth': ['analytic'], 'LAT_method': ['analytic']}
```

Streamflow Comparison and Flow Duration Curve



Domain Discretization: Soilclass



onal efficiency for the Seine's topography. (1-4), resolution and computational demands. Int on the main stem of the Seine, avoiding confluences and the estuary. The entire Seine watershed with a buffer for safety.levant to the Seine basin, including soil moisture, snow processes, vegetation and rivers:

watershed, considering its climate, vegetation, and soil characteristics. For operate climates, while the chosen groundwater and routing options (*qTopmod1*) process representation, computational efficiency, and data availability. observational data.

arid basin using SWAT. Journal of Hydrology, 388(3-4), 336-349. variable Infiltration Capacity model. Water Resources Research, 31(6), 1471-1482. atmosphere and the land surface. Journal of Applied Meteorology, 33(3), 521-542. election of input parameters. Texas A&M University, College Station, TX. watershed hydrology using a quasi-physical modeling approach. Water Resources Research, 35(11), ed runoff variability in semi-arid watersheds. Journal of Hydrology, 333(2-4), 556-568.

DELTA: Deliberately Expert Liquid Topography Assistant

Next problem:

Human and economic resource restrictions bottleneck scientific exploration and documentation.

Solution:

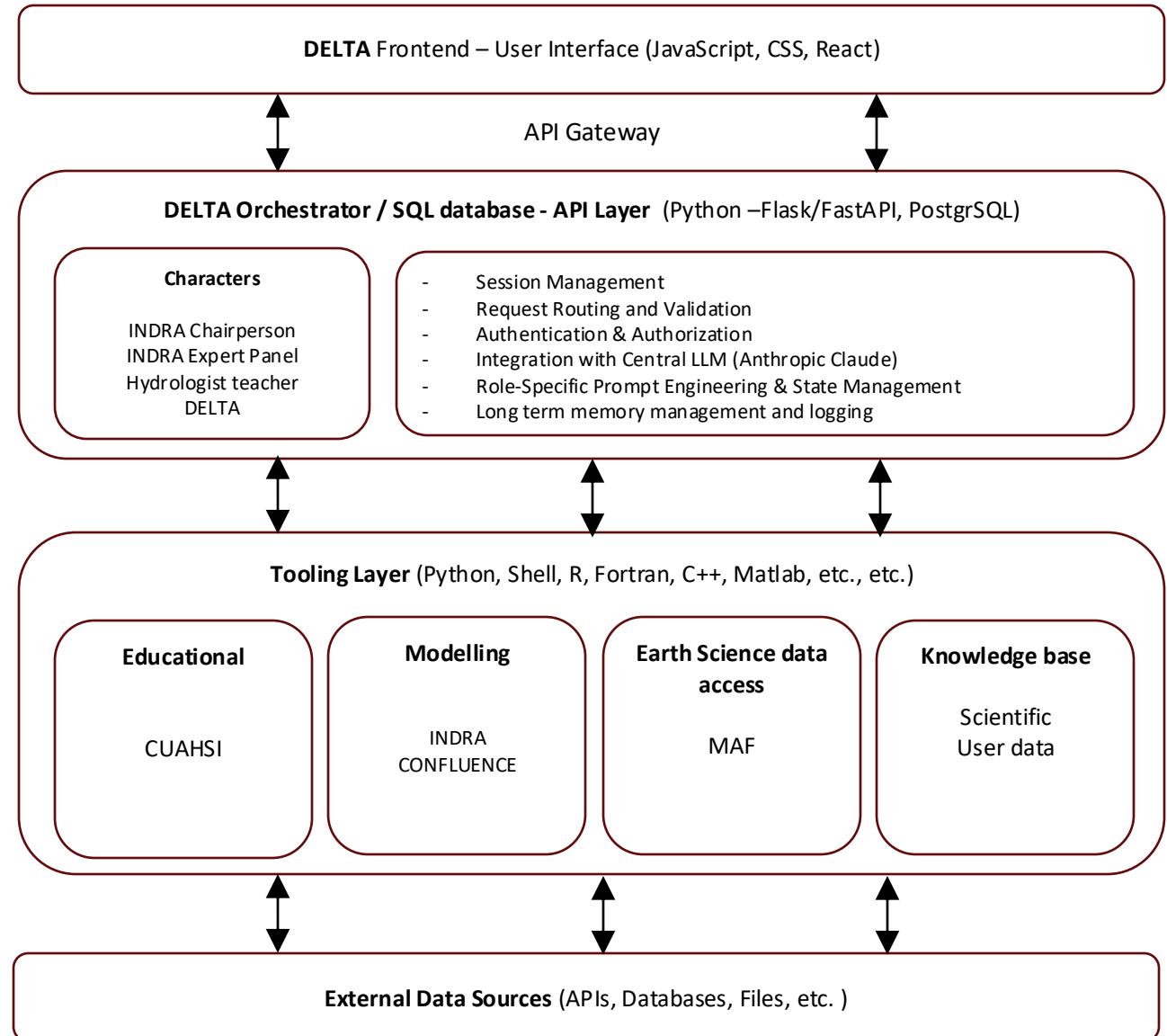
Individual and domain tailored AI research assistance. With appropriate access to research tools such as MAF/INDRA/CONFLUENCE along with data and computational resources to use them. Freely available and easily accessible online in all languages.



DELTA: Deliberately Expert Liquid Topography Assistant

Implementation:

- Online frontend
 - User Interface
 - Collaborative exploration
 - Result exports
- Backend orchestration
 - Frontend API
 - LLM / AI / TTS service APIs
 - Misc. communication with tools
- Toolbox (extendable)
 - Modeling workflows (CONFLUENCE / INDRA)
 - Data analysis (MAF)
 - Knowledge database (SQL)
 - Educational content
- External data sources
- Prototype framework [online](#)



Future Directions

- Continue fusion of community assets
 - Design wrappers for individual assets
 - Optimize central architecture
- Develop AI enhancement, e.g.:
 - Validation protocols
 - Evaluation philosophy
 - Retrieval Augmented Generation (RAG)
 - Prompt engineering
- Develop interfaces
 - Develop integration philosophy
 - Estimate resource requirements



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 



Any questions so far?

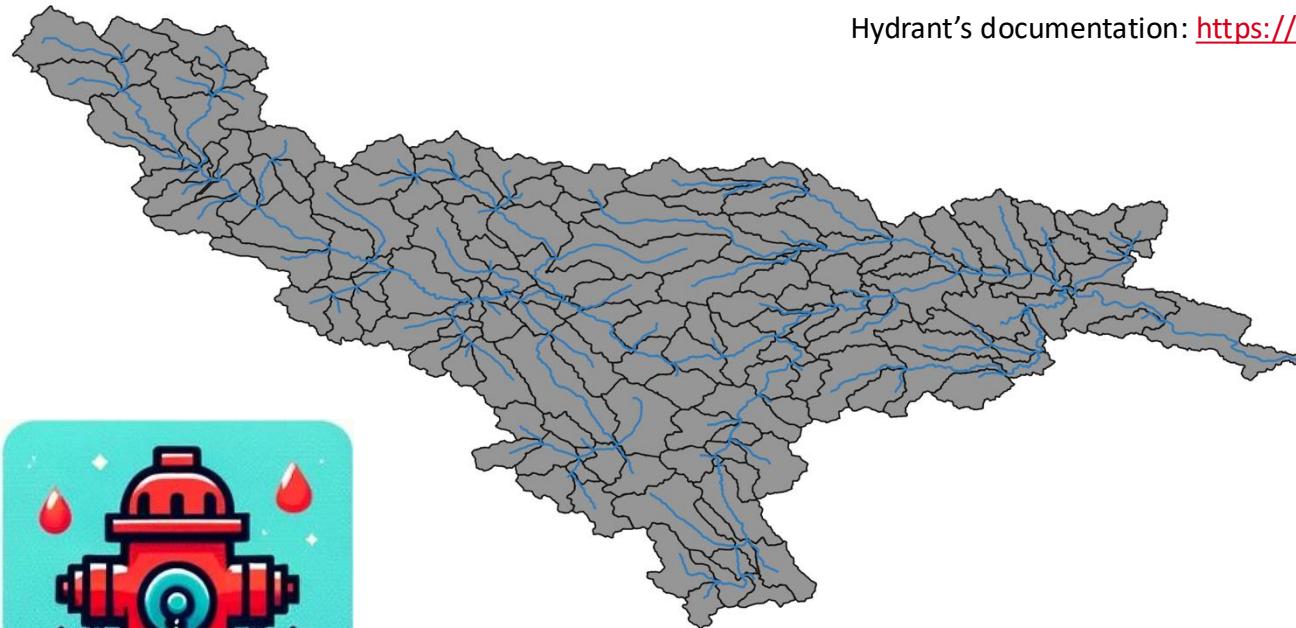
Computational Resources

Let's set up our Python virtual environment



DALLE generated image

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?

Practice Example – Bow River at Banff

Navigate using JupyterLab file browser

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 the text 'Navigate using JupyterLab file browser' to the sidebar. The sidebar includes a search bar, a '+' button, and a list of files: '0-prerequisites', '1-geofabric', '2-agnostic', '3-specific', 'LICENSE', and 'README.md'. The 'README.md' file is currently selected, indicated by a blue highlight. The main content area shows the 'README.md' file's content:

Library requirements

General

Certain libraries and binary executables are necessary to run the workflows in this repository. Below necessary libraries for general usage are mentioned:

1. CDO (Climate Data Operators >=v2.2.1),
2. ecCodes (>=v2.25.0),
3. Expat XML parser (>=v2.4.1),
4. GDAL (>=3.5.1),
5. GEOS (>=3.10.2),
6. HDF5 (>=1.10.6),
7. JasPer (>=2.0.16),
8. libaec (>=1.0.6),
9. libfabric (>=1.10.1),
10. libffi (>=3.3),
11. libgeotiff (>=1.7.1),
12. librtp topo (>=1.1.0),
13. libspatialindex (>=1.8.5),
14. libspatialite (>=5.0.1),
15. netcdf-fortran (>=4.5.2),
16. netcdf (>=4.7.4),
17. postgresql (>=12.4),
18. proj (>=9.0.1),
19. python (>=3.10.2),

Simple 0 \$ 0 README.md

Setting up ECCC's National Water Model—MESH

The screenshot shows a Jupyter Notebook interface. On the left is a file browser with a list of files in the directory `/.../community-workflows/1-geofabric`. The file `pre-process-geospa...` is selected and highlighted with a blue background. A red arrow points from a callout box labeled "Double click to launch the Notebook" to this selected file. The main area contains a Markdown cell titled "Basic preparations". A red arrow points from a callout box labeled "Tip: review keyboard shortcuts!" to the keyboard shortcut "Shift + Enter" mentioned in the text. Another red arrow points from a callout box labeled "Notebook cells" to the code cell below. The code cell contains the following Python code:

```
[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
```

In the bottom right corner, there is a logo for the University of Calgary.

Use “Shift + Enter” to execute Notebook cells

Double click to launch the Notebook

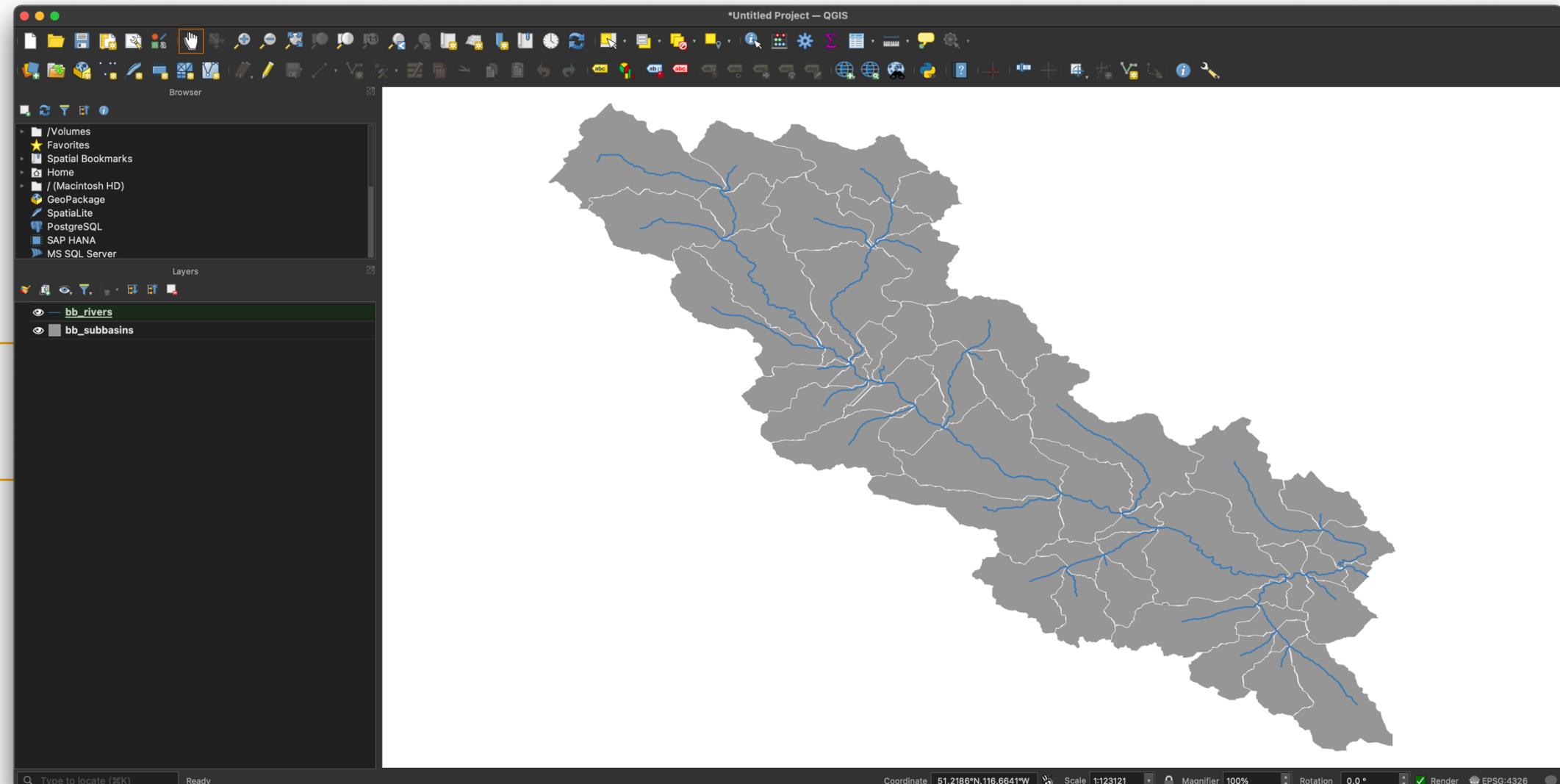
Notebook cells

Tip: review keyboard shortcuts!

78

Simple 0 \$ 2 scienv | Idle Mode: Command Ln 1, Col 1 pre-process-geospatial-fabric.ipynb

Setting up ECCC's National Water Model—MESH



Future Directions