

Community Hydrological Modelling Workflows

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

Kasra Keshavarz
Research Scientist
Schulich School of Engineering
University of Calgary

March 21st, 2024

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



**UNIVERSITY OF
CALGARY**

Acknowledgements



GLOBAL WATER FUTURES



Environment and
Climate Change Canada

Environnement et
Changement climatique Canada



**Digital Research
Alliance** of Canada

**Alliance de recherche
numérique** du Canada

Special thanks to:

- Wouter Knoben (Ucalgary), Shervan Gharari (Usask), Martyn Clark (Ucalgary), Ala Bahrami (Usask), Guoqiang Tang (NCAR), Cooper Albano (Usask), John Pomeroy (Usask), Dan Princz (ECCC), Travis Logan (Ouranos), Pierre Pellerin (Ouranos), Dikraa Khedhaouiria (ECCC) and everyone present today in the session.



Outline

- Community Hydrological Modelling Recipe
- Digital Infrastructure on Digital Research Alliance of Canada's HPCs
- Introducing Community Workflows & Tools
- Practice Example – Bow River At Banff
- Troubleshooting

Outline

- Community Hydrological Modelling Recipe
- Digital Infrastructure on Digital Research Alliance of Canada's HPCs
- Introducing Community Workflows & Tools
- Practice Example – Bow River At Banff
- Troubleshooting

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.

- Developing relevant workflows on High Performance Computing (HPC) facilities
- Keeping workflows open source and reproducible for the scientific community
- Keeping workflows efficient for real-world applications
- Separate “model-agnostic” and “model-specific” configuration parts

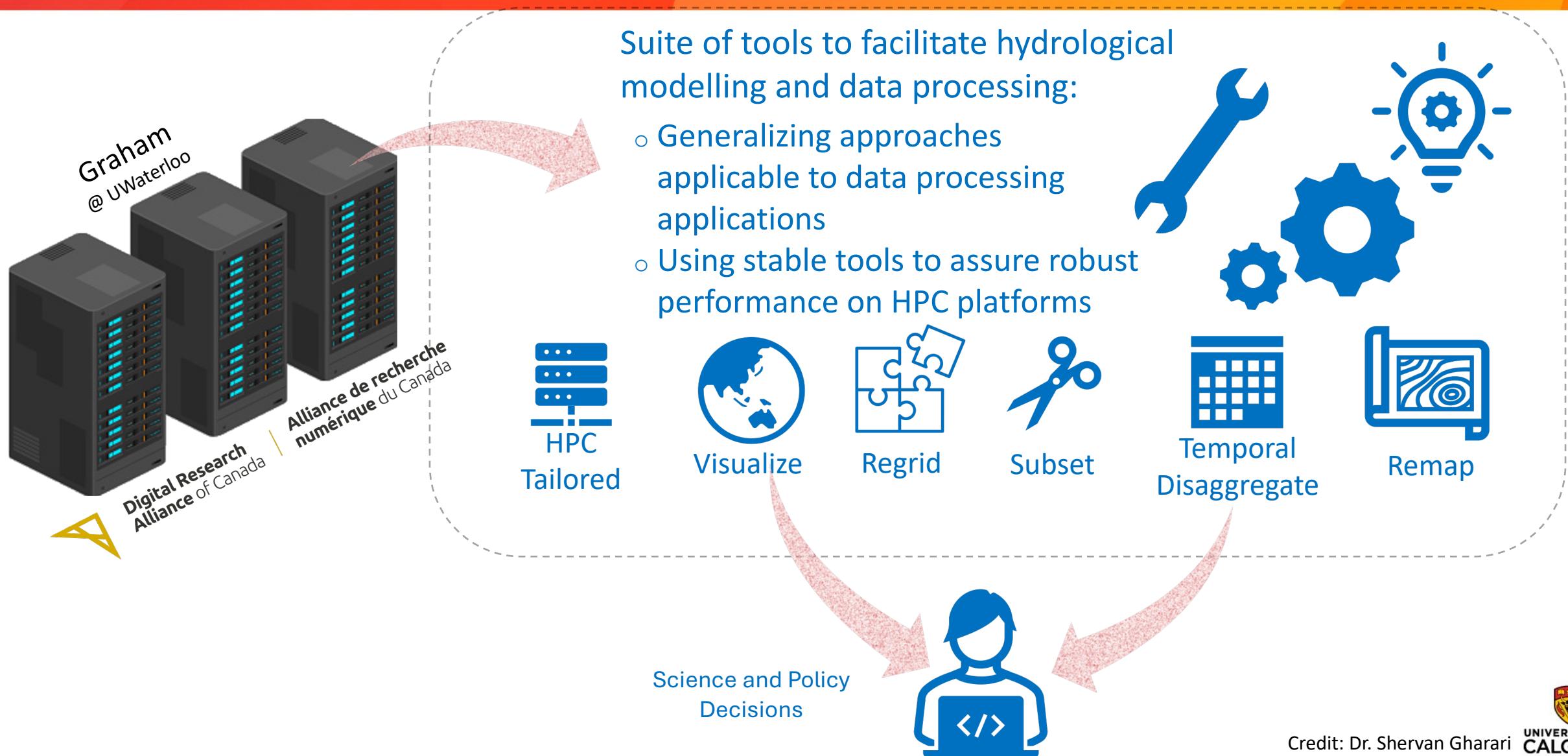


UNIVERSITY OF
CALGARY

Outline

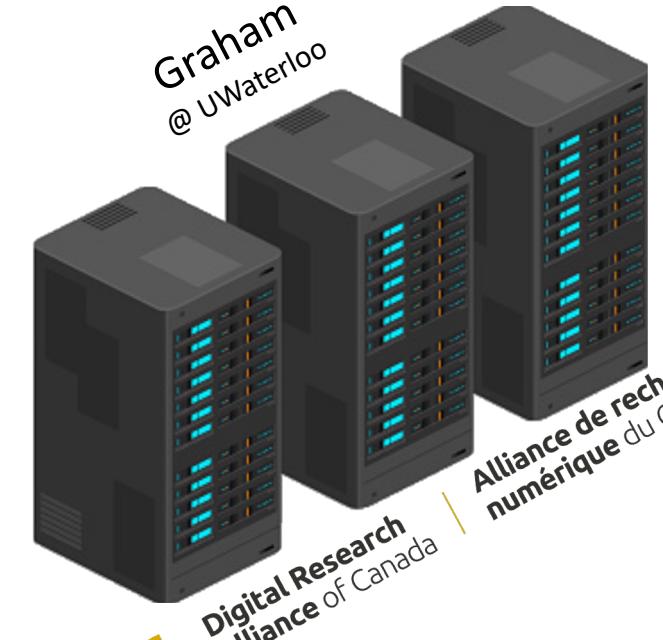
- Community Hydrological Modelling Recipe
- Digital Infrastructure on Digital Research Alliance of Canada's HPCs
- Introducing Community Workflows & Tools
- Practice Example – Bow River At Banff
- Troubleshooting

Digital Infrastructure



Digital Infrastructure

Graham
@ uWaterloo



Remote
access
accounts

rpp-kshook
rrg-mclark



Digital Infrastructure



Our recommendation is to access Graham using either of the following methods:

jupyterlab

<https://jupyterhub.sharcnet.ca/>

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



\$ ssh USERNAME@graham.alliancecan.ca

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



MobaXTerm



PuTTY



MacOS Terminal

Any questions so far?

Digital Infrastructure



<https://jupyterhub.sharcnet.ca/>



- Multi-user Server Service for Jupyter Sessions
- User Authentication Capability
- Resource Customization
- Customizable Work Environment

Sign in

Username:

Password:

Digital Infrastructure

The screenshot shows the JupyterHub Server Options page. At the top, there's a navigation bar with the JupyterHub logo, 'Home', 'Token', a user name 'kasra545', and a 'Logout' button. Below the navigation, the title 'Server Options' is centered. On the left, there are two account icons: 'rpp-kshook' (GWF logo) and 'rrg-mclark' (University of Calgary logo). A red box labeled 'Safe to ignore' has an arrow pointing to the 'rpp-kshook' icon. To the right of the accounts are 'Reservation' (set to 'None'), 'Account' (set to 'rpp-kshook_cpu'), 'Number of cores' (set to 4), and 'Time (hours)' (set to 3). A red box labeled 'Today's session is 3 hours only' has an arrow pointing to the 'Time (hours)' field. Below these are 'Memory (MB)' (set to 16000) and a checkbox for 'Enable core oversubscription?' (unchecked). A red box labeled 'How much memory you need? 16000 MB is enough for today' has an arrow pointing to the 'Memory (MB)' field. Further down are 'GPU configuration' (set to 'None') and 'User interface' (set to 'JupyterLab'). A red box labeled 'Do we do any Deep Learning? No.' has an arrow pointing to the 'User interface' field. At the bottom is a large orange 'Start' button. Several red boxes with text annotations are overlaid on the interface.

Safe to ignore

rpp-kshook

rrg-mclark

Safe to ignore

Reservation

None

Account

rpp-kshook_cpu

Number of cores

4

Time (hours)

3

Memory (MB)

16000

Enable core oversubscription? Recommended for interactive usage

GPU configuration

None

User interface

JupyterLab

Start

Today's session is 3 hours only

How much memory you need? 16000 MB is enough for today

Do we do any Deep Learning? No.

Strong recommendation is to use JupyterLab interface

Digital Infrastructure

<https://jupyterhub.sharcnet.ca/>



General tips:

- Understand your setup: Be resourceful! Asking too much computational resource can keep you in the queue forever; public HPCs are shared resources. Asking for too little resources can make your Jupyter session slow and laggy.



DALLE generated image

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.

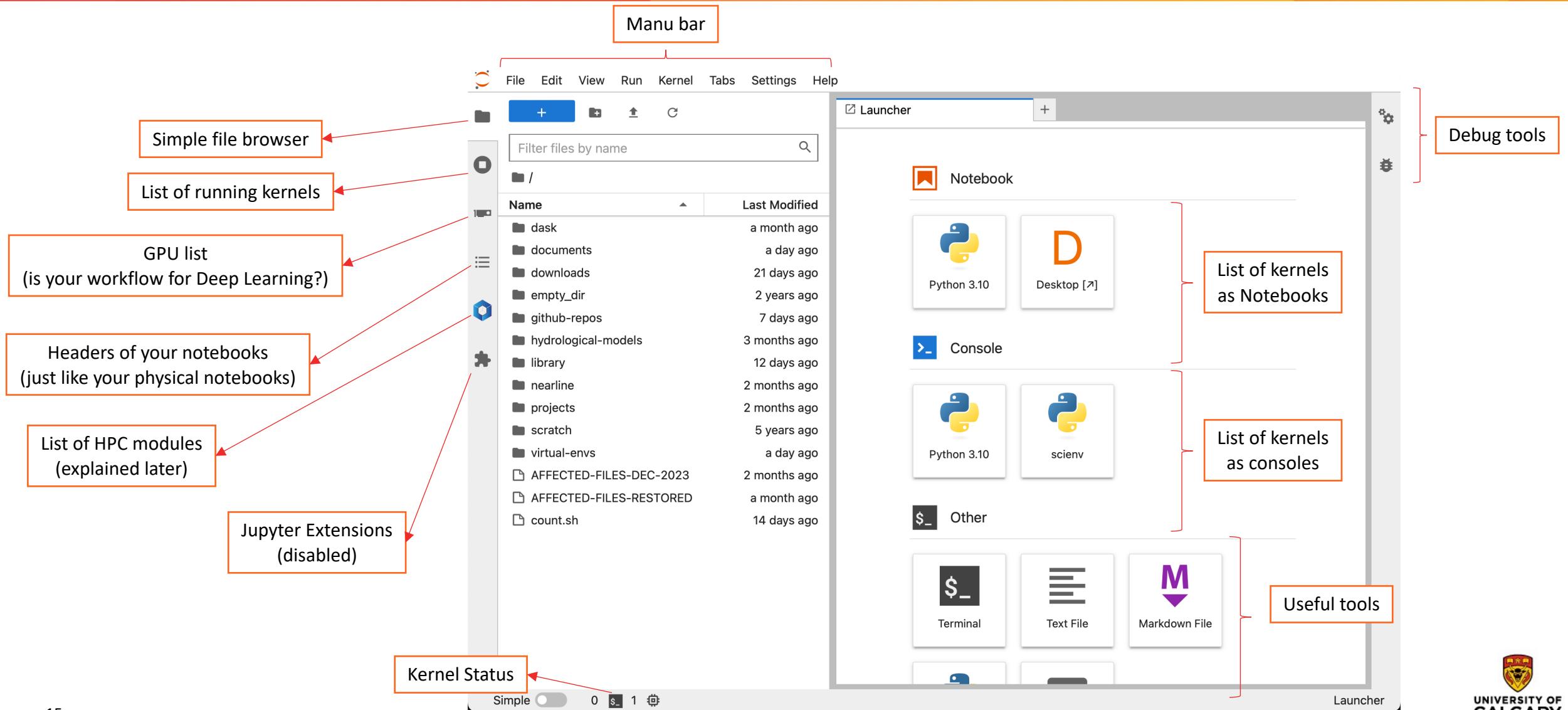


DALLE generated image



UNIVERSITY OF
CALGARY

Digital Infrastructure



Digital Infrastructure



HPCs comes with various “**modules**”

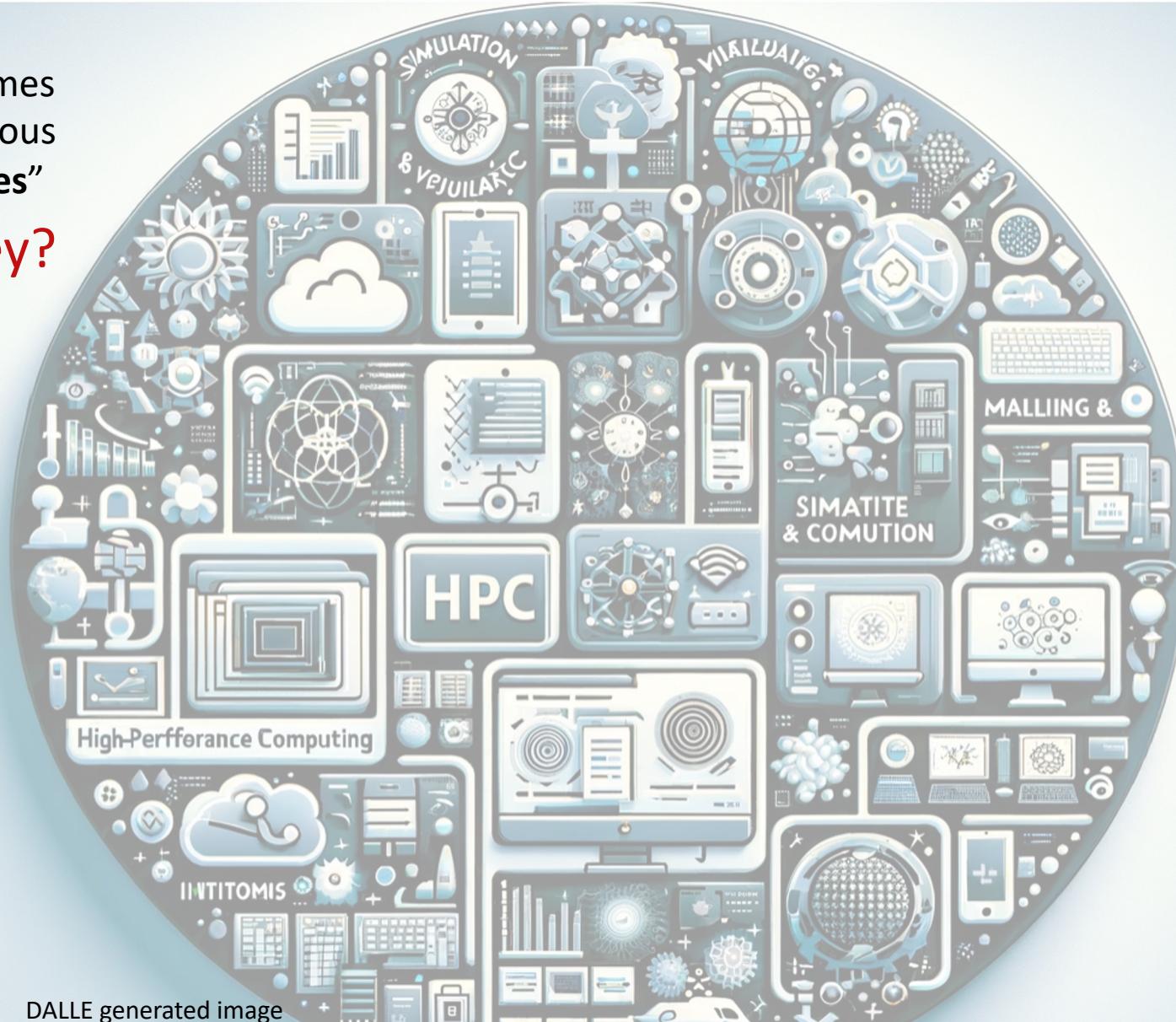
But where are they?

Flexibility

Different users have different requirements

Customization

Users may need specific versions of software or libraries

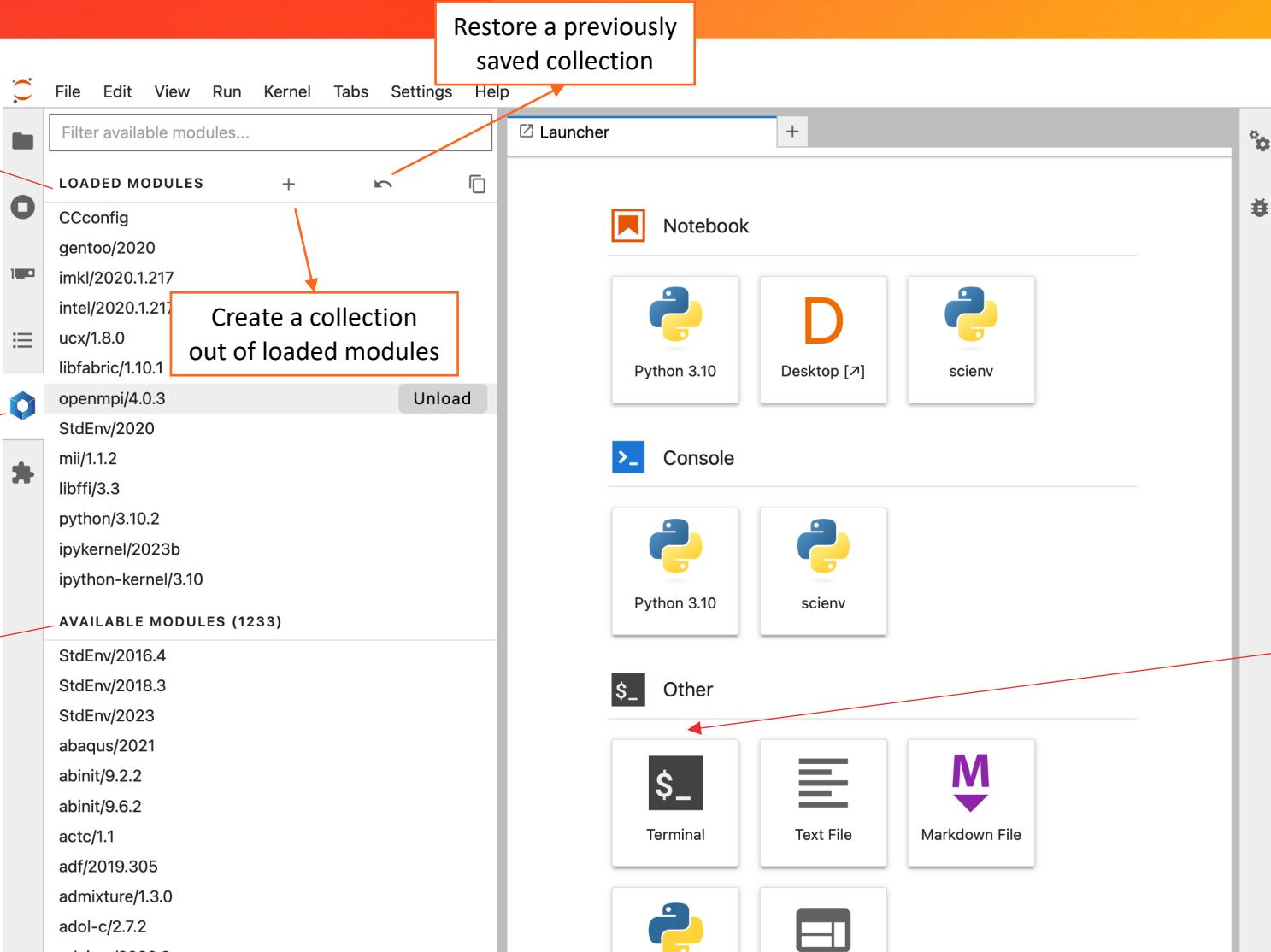


Optimization software packages are often optimized for the underlying hardware architecture

Productivity

Having commonly used software readily available saves users time and effort

Digital Infrastructure



A saved list of modules is called a “collection”

Using JupyterLab interface

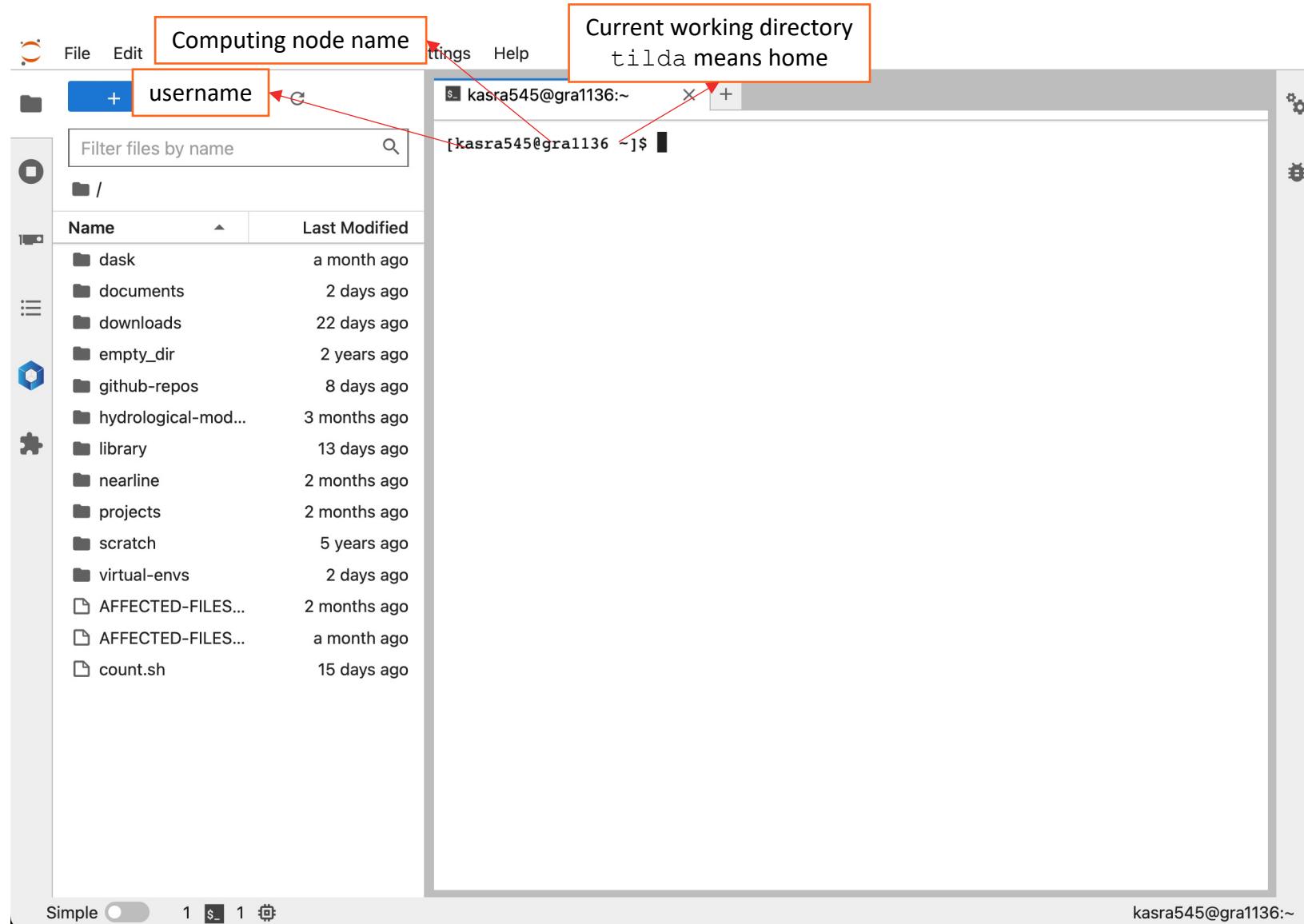
For creating collections could be time consuming.

We will use the Terminal to build a collection later.

Digital Infrastructure

All instructions are available on this presentation's repository:

<https://github.com/kasra-keshavarz/community-modelling-workflow-training>



Digital Infrastructure

Exactly similar to what we can do in this tab

All instructions are available on this presentation's repository:

<https://github.com/kasra-keshavarz/community-modelling-workflow-training>

The screenshot shows a terminal window with several annotations explaining the commands being run.

- Computing node name:** Kasra545@gra1136
- Current working directory:** tilda means home
- Username:** username
- File Explorer:** Shows a list of files and folders, with a note: "Exactly similar to what we can do in this tab".
- Module Loading:** The user is loading the Standard Environment and the Gnu C Compiler version 9.3.0.
- Module Collection:** A large bracket on the right side groups the following annotations:
 - Loading "Standard Environment"
 - Loading Gnu C Compiler version 9.3.0
 - Loading every module we need
- Module Saving:** The user is saving all modules as a collection called "science modules".

```
[kasra545@gra1136 ~]$ module load StdEnv/2020
Lmod is automatically replacing "intel/2020.1.217" with "gcc/9.3.0".
[kasra545@gra1136 ~]$ module load gcc/9.3.0
Lmod is automatically replacing "intel/2020.1.217" with "gcc/9.3.0".
[kasra545@gra1136 ~]$ module load libfabric/1.10.1 ipykernel/2023b \
>     sqlite/3.38.5 postgresql/12.4 gdal/3.5.1 \
>     udunits/2.2.28 cdo/2.2.1 gentoo/2020 \
>     imkl/2020.1.217 openmpi/4.0.3 scipy-stack/2023b \
>     jasper/2.0.16 freexl/1.0.5 geos/3.10.2 \
>     libaec/1.0.6 mpi4py/3.1.3 StdEnv/2020 \
>     gcc/9.3.0 libffi/3.3 hdf5/1.10.6 \
>     libgeotiff/proj901/1.7.1 librtp topo-proj9/1.1.0 \
>     proj/9.0.1 eccodes/2.25.0 netcdf-fortran/4.5.2 \
>     mii/1.1.2 ucx/1.8.0 python/3.10.2 \
>     netcdf/4.7.4 cfitsio/4.1.0 \
>     libspatialite-proj901/5.0.1 expat/2.4.1 \
>     yaxt/0.9.0 libspatialindex/1.8.5
Lmod is automatically replacing "intel/2020.1.217" with "gcc/9.3.0".
Lmod is automatically replacing "intel/2020.1.217" with "gcc/9.3.0".
The following have been reloaded with a version change:
  1) ipykernel/2023a => ipykernel/2023b
  2) scipy-stack/2023a => scipy-stack/2023b
[kasra545@gra1136 ~]$ module save scimods
Saved current collection of modules to: "scimods"
[kasra545@gra1136 ~]$
```

Simple 1 \$ 1

kasra545@gra1136:~

UNIVERSITY OF CALGARY

Digital Infrastructure

Refresh



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

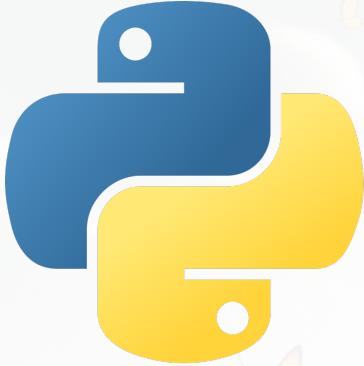
The screenshot shows a JupyterLab interface with the following elements:

- Toolbar:** File, Edit, View, Run, Kernel, Tabs, Settings, Help.
- Search Bar:** Filter available modules...
- Module List:** LOADED MODULES (CCconfig, gentoo/2020, StdEnv/2020, ipykernel/2023a, sqlite/3.38.5, postgresql/12.4, geos/3.10.2, udmunits/2.2.28, scipy-stack/2023a, imkl/2020.1.217, gcc/9.3.0, jasper/2.0.16, gdal/3.5.1, libaec/1.0.6, libfabric/1.10.1, openmpi/4.0.3, mpi4py/3.1.3, cdo/2.2.1, libff/3.3, hdf5/1.10.6, libgeotiff-proj901/1.7.1, librtp topo-proj9/1.1.0, proj/9.0.1, eccodes/2.25.0, netcdf-fortran/4.5.2).
- Modal Dialog:** Restore collection. It says "Select a collection to restore : scimods". It has "Cancel" and "Restore" buttons.
- Bottom Status Bar:** Simple, 1 \$ 1, and a file icon.
- Bottom Footer:** kasra545@gra1136:~

Annotations with red boxes and arrows:

- "Check out what module collections you have" points to the module list.
- "Wait for a few seconds so Jupyter loads all modules saved in "scimods" collection" points to the modal dialog.
- "Restore the "scimods" (science modules)" points to the "scimods" selection in the modal dialog.
- "This is only a one-time process. You can always access all the collections you saved from Jupyter's LMOD Extension tab" is a general note on the right.

Any questions so far?



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

Digital Infrastructure

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

Digital Infrastructure

High Performance Computing (HPC) systems are comprised of various “nodes”

The idea is to expand horizontally, instead of vertically.

HPC “nodes” are located inside a “rack enclosure”



DALLE generated image



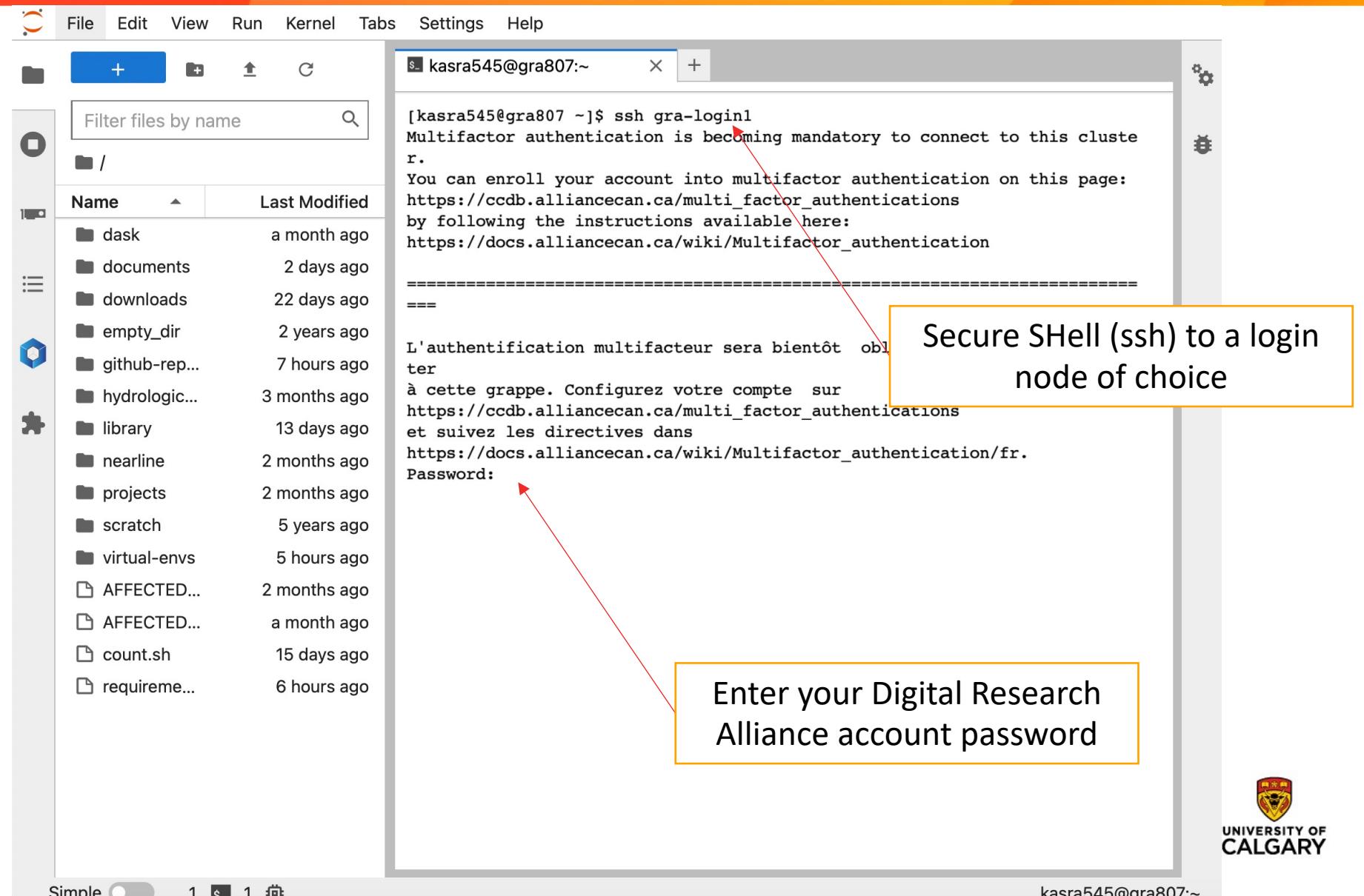
JupyterHub server lands your JupyterLab session on one of these nodes or, more clearly, “computing” nodes

On Digital Research Alliance HPCs, computing nodes do not have access to an active internet connection unfortunately

Digital Infrastructure

First, we need to switch to the “login node”

Login nodes do have access to an active internet connection



The screenshot shows a terminal window titled "kasra545@gra807:~". The window contains the following text:

```
[kasra545@gra807 ~]$ ssh gra-login1
Multifactor authentication is becoming mandatory to connect to this cluster.
You can enroll your account into multifactor authentication on this page:
https://ccdb.alliancecan.ca/multi_factor_authentications
by following the instructions available here:
https://docs.alliancecan.ca/wiki/Multifactor_authentication
=====
===
L'authentification multifacteur sera bientôt obligatoire
à cette grappe. Configurez votre compte sur
https://ccdb.alliancecan.ca/multi_factor_authentications
et suivez les directives dans
https://docs.alliancecan.ca/wiki/Multifactor_authentication/fr.
Password:
```

Two red arrows point from callout boxes to specific parts of the terminal output:

- A red arrow points from the text "Secure SHell (ssh) to a login node of choice" to the line "[kasra545@gra807 ~]\$ ssh gra-login1".
- A red arrow points from the text "Enter your Digital Research Alliance account password" to the "Password:" prompt.

The terminal interface includes a file browser on the left and various system status indicators at the bottom.

Digital Infrastructure

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

You are logged into a “login node” with an active internet connection

The screenshot shows a terminal window titled "kasra545@gra-login1:~". The terminal prompt is "[kasra545@gra-login1 ~]\$". A red arrow points from the text "You are logged into a ‘login node’ with an active internet connection" to the terminal window. To the left of the terminal is a file manager interface showing a list of files in the current directory. The files listed include "dask", "documents", "downloads", "empty_dir", "github-rep...", "hydrologic...", "library", "nearline", "projects", "scratch", "virtual-envs", "AFFECTED...", "count.sh", and "requireme...". The "AFFECTED..." files are shown with a preview icon. The terminal window has tabs at the top, and there are icons for settings and gear in the top right corner.

Name	Last Modified
dask	a month ago
documents	2 days ago
downloads	22 days ago
empty_dir	2 years ago
github-rep...	7 hours ago
hydrologic...	3 months ago
library	13 days ago
nearline	2 months ago
projects	2 months ago
scratch	5 years ago
virtual-envs	5 hours ago
AFFECTED...	2 months ago
AFFECTED...	a month ago
count.sh	15 days ago
requireme...	7 hours ago

Simple 1 \$ 1

kasra545@gra-login1:~

Digital Infrastructure

The screenshot shows a terminal window titled 'kasra545@gra-login1:~'. The user has run the command 'git clone https://github.com/kasra-keshavarz/community-modelling-workflow-training.git ./github-repos/community-workflows'. The terminal output shows the progress of the cloning process, including object enumeration, counting, compressing, and receiving objects from the remote repository. A red arrow points from the text 'Create a "github-repos" directory to organize downloaded repositories' to the line 'mkdir -p ./github-repos'. Another red box highlights the word 'clone' in the command 'git clone'.

```
[kasra545@gra-login1 ~]$ mkdir -p ./github-repos
[kasra545@gra-login1 ~]$ git clone https://github.com/kasra-keshavarz/community-modelling-workflow-training.git ./github-repos/community-workflows
Cloning into './github-repos/community-workflows'...
remote: Enumerating objects: 64, done.
remote: Counting objects: 100% (64/64), done.
remote: Compressing objects: 100% (52/52), done.
remote: Total 64 (delta 11), reused 56 (delta 7), pack-reused 0
Receiving objects: 100% (64/64), 521.57 KiB | 4.01 MiB/s, done.
Resolving deltas: 100% (11/11), done.
[kasra545@gra-login1 ~]$
```

Create a “github-repos” directory to organize downloaded repositories

“clone”-ing the repository

Digital Infrastructure

1	Only for personal use	Home space 50 GB per user
2	Only for data processing	Scratch space 20 TB per user
3	NOT for personal use – only for shared resources	Project space rrp-kshook: 2.5 PB  rrg-mclark: 0.2 PB 



Any questions so far?

Digital Infrastructure

Let's set up our Python virtual environment



Digital Infrastructure

Although the modules are loaded in your JupyterLab session, Terminal is a separate entity, and we must reload the modules there as well:

restoring "scimods"

Checking what has been loaded

Is it equal to the list on left?

Filter available 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

AVAILABLE MODULES (1233)

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

[kasra545@gra-login1:~]\$ module restore scimods
Restoring modules from user's scimods
[kasra545@gra-login1 ~]\$ module list

Currently Loaded Modules:

Module	Type	Description
1) CCConfig	(geo)	20) cdo/2.2.1
2) gentoo/2020	(S)	21) libffi/3.3
3) StdEnv/2020	(S)	22) hdf5/1.10.6
4) cmake/3.27.7	(t)	23) libgeotiff-proj901/1.7.1
5) sqlite/3.38.5	(geo)	24) librtp topo-proj9/1.1.0
6) geos/3.10.2	(geo)	25) proj/9.0.1
7) postgresql/12.4	(t)	26) eccodes/2.25.0
8) udnits/2.2.28	(t)	27) netcdf-fortran/4.5.2
9) imkl/2020.1.217	(math)	28) mii/1.1.2
10) gcccore/.9.3.0	(H)	29) ucx/1.8.0
11) gcc/9.3.0	(t)	30) python/3.10.2
12) jasper/2.0.16	(vis)	31) netcdf/4.7.4
13) libaec/1.0.6	(io)	32) cfitsio/4.1.0
14) ipykernel/2023b	(vis)	33) freexl/1.0.5
15) scipy-stack/2023b	(math)	34) libspatialite-proj901/5.0.1
16) gdal/3.5.1	(geo)	35) expat/2.4.1
17) libfabric/1.10.1	(t)	36) yaxt/0.9.0
18) openmpi/4.0.3	(m)	37) libspatialindex/1.8.5
19) mpi4py/3.1.3	(t)	38) arrow/13.0.0

Where:

- S:** Module is Sticky, requires --force to unload or purge
- m:** MPI implementations / Implémentations MPI
- math:** Mathematical libraries / Bibliothèques mathématiques
- io:** Input/output software / Logiciel d'écriture/lecture
- t:** Tools for development / Outils de développement
- vis:** Visualisation software / Logiciels de visualisation
- geo:** Geography libraries/apps / Logiciels de géographie
- phys:** Physics libraries/apps / Logiciels de physique

H: Hidden Module

Simple 1 \$ 1

kasra545@gra-login1:~

Digital Infrastructure

The screenshot shows a terminal window with the following content:

```
kasra545@gra-login1:~
```

LOADED MODULES

Module	Type
gcccore/.9.3.0	(H)
gcc/9.3.0	(t)
jasper/2.0.16	(vis)
libaec/1.0.6	
ipykernel/2023b	
scipy-stack/2023b	(math)
gdal/3.5.1	(geo)
libfabric/1.10.1	
openmpi/4.0.3	(m)
mpi4py/3.1.3	(t)
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	

AVAILABLE MODULES (1233)

Module
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

Where:

- S:** Module is Sticky, requires --force to unload or purge
- m:** MPI implementations / Implémentations MPI
- math:** Mathematical libraries / Bibliothèques mathématiques
- io:** Input/output software / Logiciel d'écriture/lecture
- t:** Tools for development / Outils de développement
- vis:** Visualisation software / Logiciels de visualisation
- geo:** Geography libraries/apps / Logiciels de géographie
- phys:** Physics libraries/apps / Logiciels de physique
- H:** Hidden Module

```
[kasra545@gra-login1 ~]$ python -m venv --version
usage: venv [-h] [--system-site-packages] [--symlinks | --copies]
             [--clear] [--upgrade] [--without-pip] [--prompt PROMPT]
             [--upgrade-deps]
             ENV_DIR [ENV_DIR ...]

venv: error: the following arguments are required: ENV_DIR
[kasra545@gra-login1 ~]$ python -m virtualenv --version
virtualenv 20.13.0 from /cvmfs/soft.computeCanada.ca/easybuild/software/20
20/avx2/Core/python/3.10.2/lib/python3.10/site-packages/virtualenv/_init_
_.PY
[kasra545@gra-login1 ~]$
```

Are we all using the same version?

Digital Infrastructure

The screenshot shows a terminal window titled "kasra545@gra-login1:~". The left sidebar lists "LOADED MODULES" including CCconfig, gentoo/2020, imkl/2020.1.217, intel/2020.1.217, ucx/1.8.0, libfabric/1.10.1, openmpi/4.0.3, StdEnv/2022, mii/1.1.2, and libffi/3.3. A red box highlights the text "“virtual-envs” directory is in your home directory". Another red box highlights the instruction "Now build a Python Virtual Environment; put it under the “virtual-envs” directory, and call it “scienv” after switching to a “login node”". The right pane displays a list of modules with their types: t (tools), vis (visualization), geo (geography), phys (physics), and io (input/output). Below this is a "Where:" section with module descriptions and category mappings. At the bottom, a command-line session shows the creation of a virtual environment named "scienv" in the user's home directory.

```
Filter available 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/2022  
mii/1.1.2  
libffi/3.3  
“virtual-envs” directory is in your home directory  
Now build a Python Virtual Environment; put it under the “virtual-envs” directory, and call it “scienv” after switching to a “login node”  
abaqus/2021  
abinit/9.2.2  
abinit/9.6.2  
actc/1.1  
adf/2019.305  
admixture/1.3.0  
Filter available modules...  
kasra545@gra-login1:~ +  
11) gcc/9.3.0 (t) 30) python/3.10.2 (t)  
12) jasper/2.0.16 (vis) 31) netcdf/4.7.4 (io)  
13) libaec/1.0.6 32) cfitsio/4.1.0 (vis)  
14) ipykernel/2023b 33) freexl/1.0.5 (t)  
15) scipy-stack/2023b (math) 34) libspatialite-proj901/5.0.1  
16) gdal/3.5.1 (geo) 35) expat/2.4.1 (t)  
17) libfabric/1.10.1 36) yaxt/0.9.0 (t)  
18) openmpi/4.0.3 (m) 37) libspatialindex/1.8.5 (phys)  
19) mpi4py/3.1.3 (t) 38) arrow/13.0.0 (t)  
Where:  
Module is Sticky, requires --force to unload or purge  
MPI implementations / Implémentations MPI  
Mathematical libraries / Bibliothèques mathématiques  
Input/output software / Logiciel d'écriture/lecture  
Tools for development / Outils de développement  
Visualisation software / Logiciels de visualisation  
Geography libraries/apps / Logiciels de géographie  
Physics libraries/apps / Logiciels de physique  
Hidden Module  
t: MPI implementations / Implémentations MPI  
vis: Mathematical libraries / Bibliothèques mathématiques  
geo: Input/output software / Logiciel d'écriture/lecture  
phys: Tools for development / Outils de développement  
H: Hidden Module  
kasra545@gra-login1 ~]$ python -m venv --version  
usage: venv [-h] [--system-site-packages] [--symlinks | --copies]  
           [--clear] [--upgrade] [--without-pip] [--prompt PROMPT]  
           [--upgrade-deps]  
           ENV_DIR [ENV_DIR ...]  
venv: error: the following arguments are required: ENV_DIR  
[kasra545@gra-login1 ~]$ python -m virtualenv --version  
virtualenv 20.13.0 from /cvmfs/soft.computeCanada.ca/easybuild/software/20  
20/avx2/Core/python/3.10.2/lib/python3.10/site-packages/virtualenv/_init_  
.py  
[kasra545@gra-login1 ~]$ python -m virtualenv ./virtual-envs  
/scienv  
Simple 1 $ 1 ⌂  
kasra545@gra-login1:~
```

Digital Infrastructure

The screenshot shows a terminal window titled 'kasra545@login1:~'. The terminal contains the following command sequence:

```
kasra545@login1 ~$ source ~/virtual-envs/scienv/bin/activate  
(scienv) [kasra545@login1 ~]$ pip install -r ~/github-repos/community-workflows/0-prerequisites/requirements.txt
```

A red box highlights the command 'source ~/virtual-envs/scienv/bin/activate' and its resulting prompt '(scienv)'. Another red box highlights the command 'pip install -r ~/github-repos/community-workflows/0-prerequisites/requirements.txt'. Two red arrows point from these highlighted areas to callout boxes on the right.

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

AVAILABLE MODULES (1233)

- 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

Simple 1 \$ 1

After creation,
let's activate the
virtual
environment

Install the
packages included
in the training
session's
repository

Digital Infrastructure

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 is a section for "AVAILABLE MODULES (1233)" which lists:

- 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 this output with the text: "“scienv” virtual environment is now accessible through JupyterLab".

Digital Infrastructure

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 is a file browser with a sidebar containing icons for files, notebooks, and other types of documents. The main area displays a list of files and their last modified times. On the right is a 'Launcher' panel with sections for 'Notebook', 'Console', and 'Other'. The 'Notebook' section contains icons for 'Python 3 (ipykernel)', 'Desktop [↗]', and 'scienv'. A red box highlights the 'scienv' icon, and a red arrow points from the text 'The "scienv" (science environment) kernel is now recognized by Jupyter' to it. The 'Console' section also has 'Python 3 (ipykernel)' and 'scienv' icons. The 'Other' section includes icons for 'Terminal', 'Text File', 'Markdown File', and 'Python File'. At the bottom, there are buttons for 'Simple' mode and a 'Launcher' button.

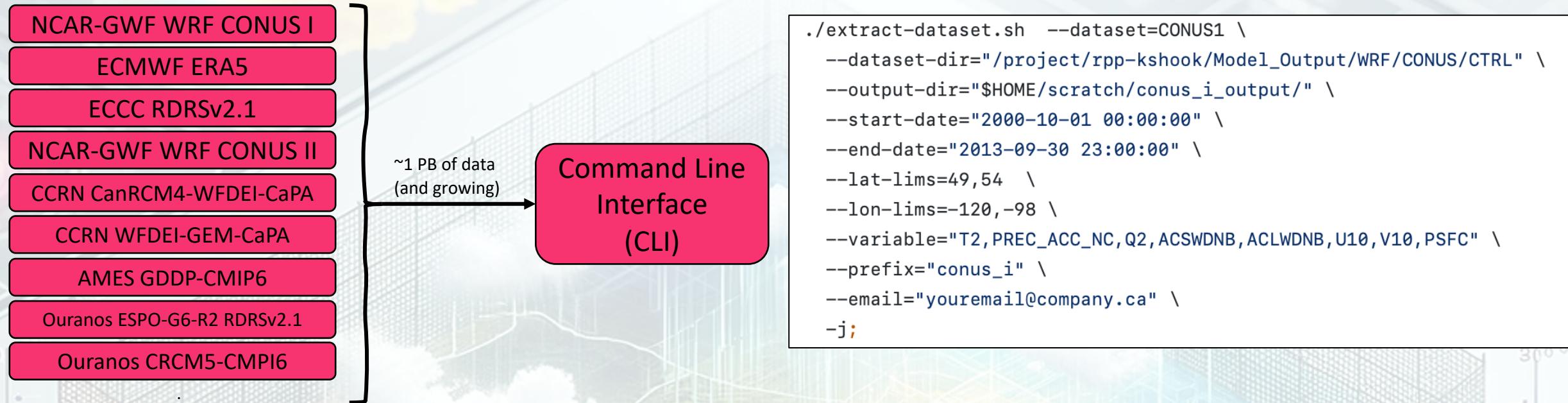
The "scienv" (science environment) kernel is now recognized by Jupyter

Name	Last
dask	a m
documents	2
downloads	22 days ago
empty_dir	2 years ago
github-repos	8 days ago
hydrological-mod...	3 months ago
library	13 days ago
nearline	2 months ago
projects	2 months ago
scratch	5 years ago
virtual-envs	2 days ago
FFECTED-FILES...	2 months ago
FFECTED-FILES...	a month ago
count.sh	15 days ago

Outline

- Community Hydrological Modelling Recipe
- Digital Infrastructure on Digital Research Alliance of Canada's HPCs
- Introducing Community Workflows & Tools
- Practice Example – Bow River At Banff
- Troubleshooting

Introducing Community Workflows & Tools

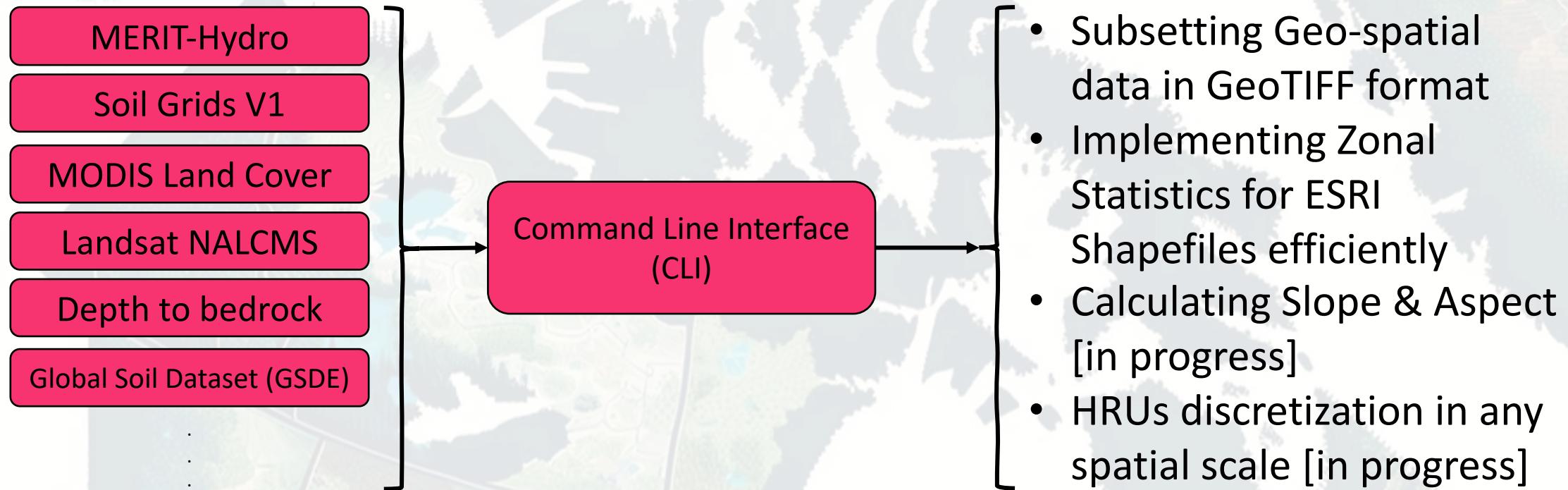


Steps:

1. Login to Digital Alliance of Canada (formerly Compute Canada) Graham cluster,
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/kasra-keshavarz/datatool>

Introducing Community Workflows & Tools

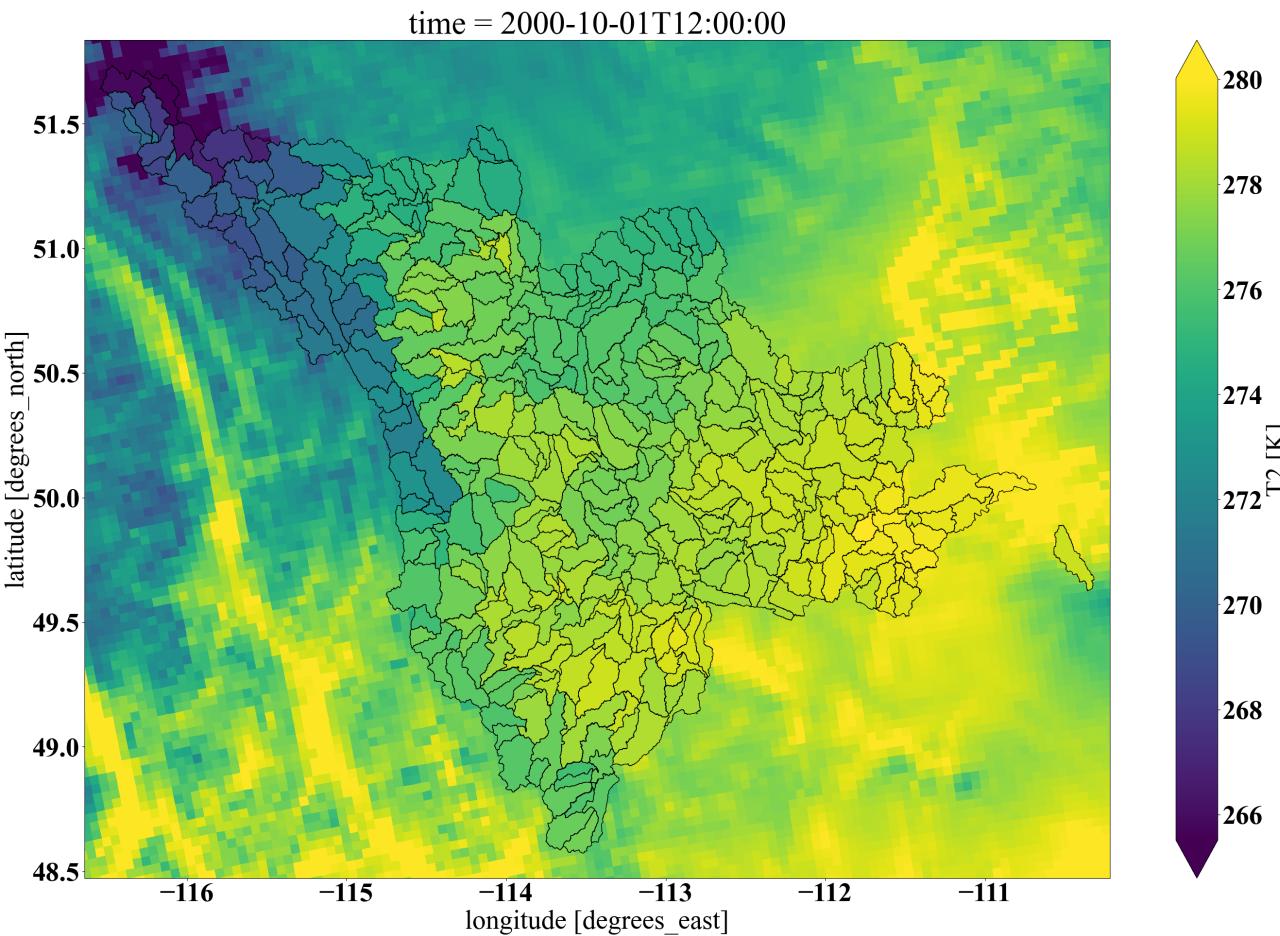


Steps:

1. Login to Digital Alliance of Canada (formerly Compute Canada) Graham cluster,
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/kasra-keshavarz/gistool>

Introducing Community Workflows & Tools



```
kasras-mbp:easymore 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

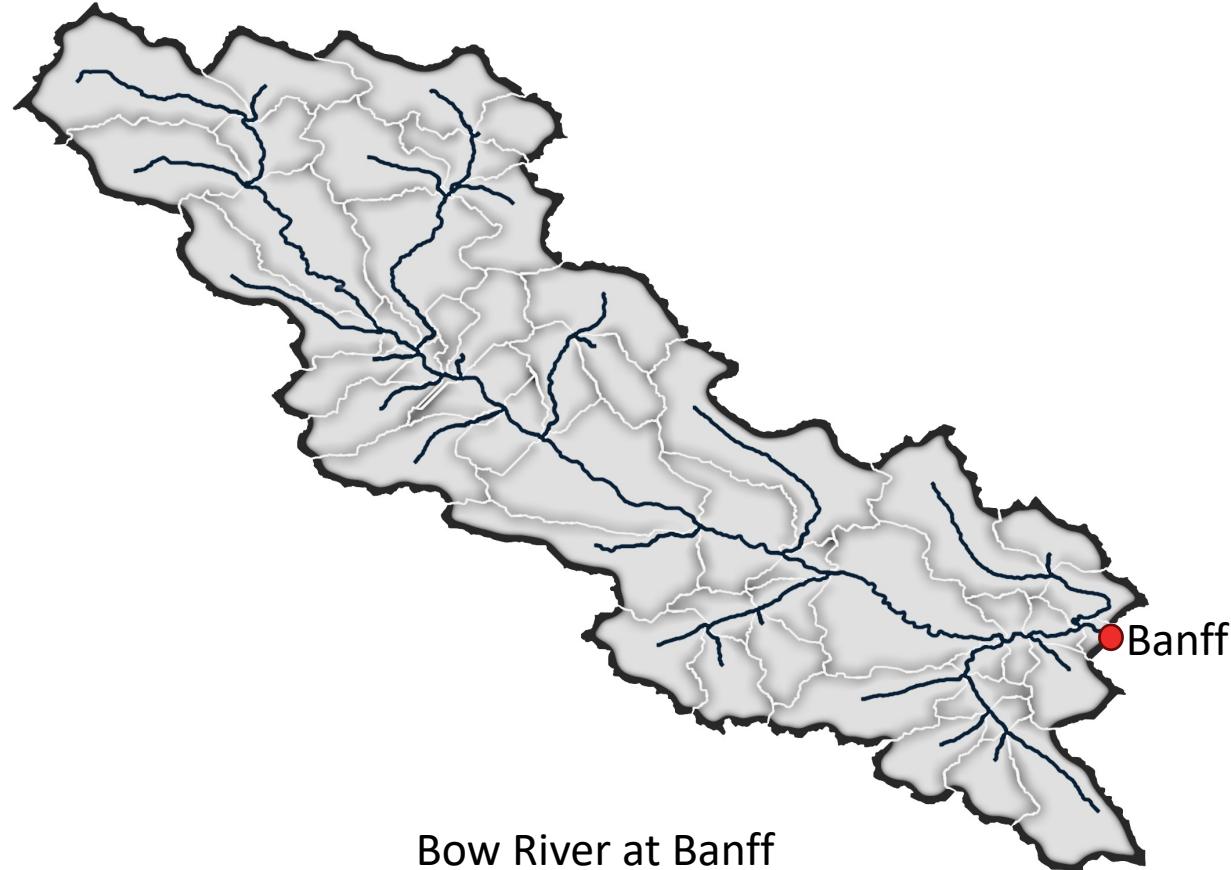
<https://github.com/ShervanGharari/EASYMORE.git>

Any questions so far?

Outline

- Community Hydrological Modelling Recipe
- Digital Infrastructure on Digital Research Alliance of Canada's HPCs
- Introducing Community Workflows & Tools
- Practice Example – Bow River At Banff
- Troubleshooting

Practice Example – Bow River at Banff



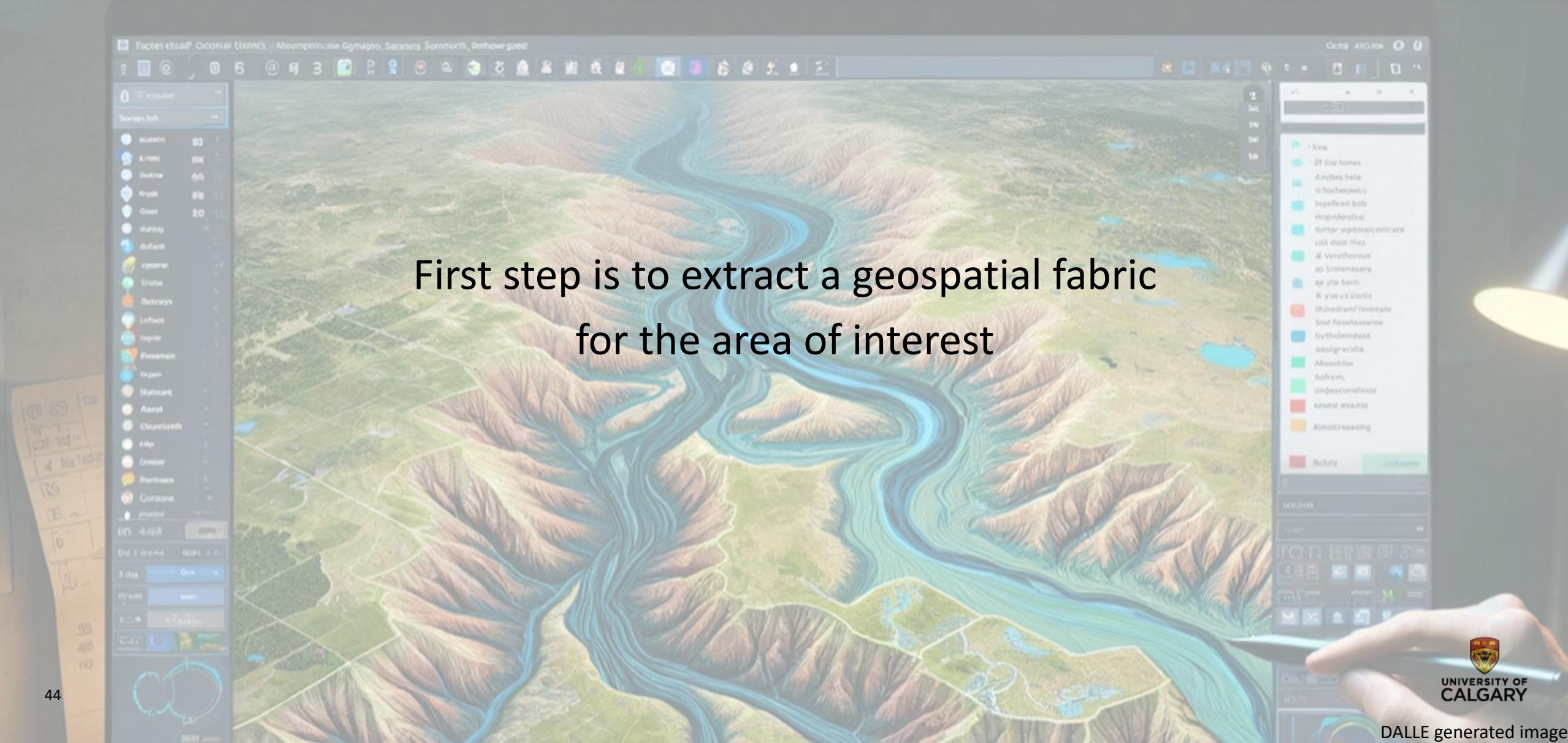
Sub-basins and river network extracted from MERIT-Basins
using Hydrant v0.1.0-dev0

All instructions are available
on this presentation's
repository:

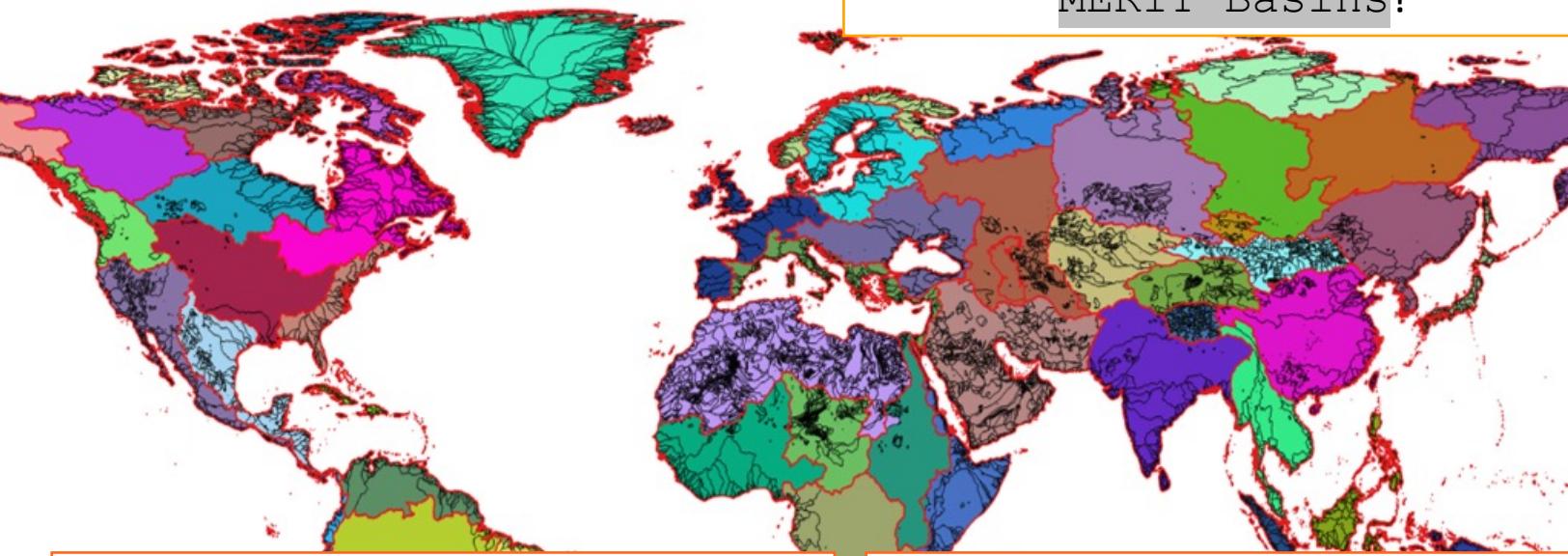
<https://github.com/kasra-keshavarz/community-modelling-workflow-training>

Practice Example – Bow River at Banff

First step is to extract a geospatial fabric
for the area of interest



Practice Example – Bow River at Banff



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

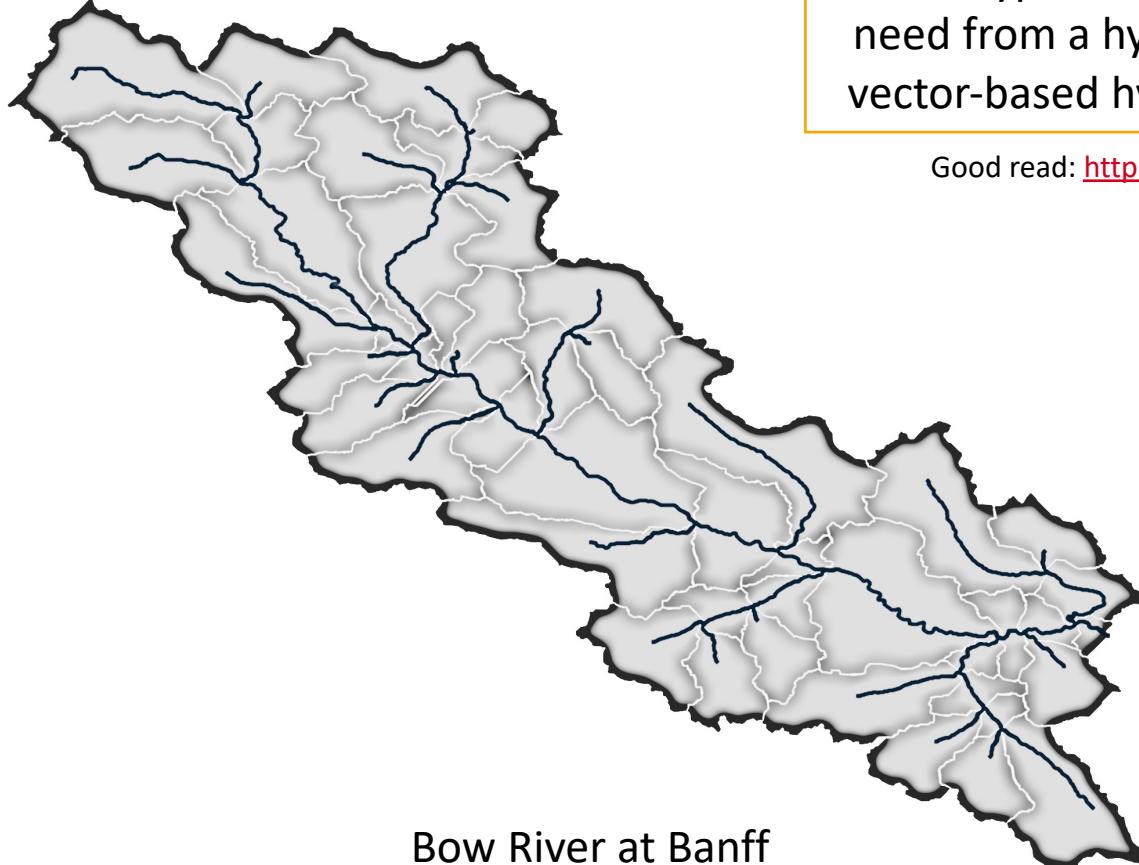
Subbasins:

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

Non-contributing areas:

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

Practice Example – Bow River at Banff



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?

Good read: <http://switchfromshapefile.org>

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

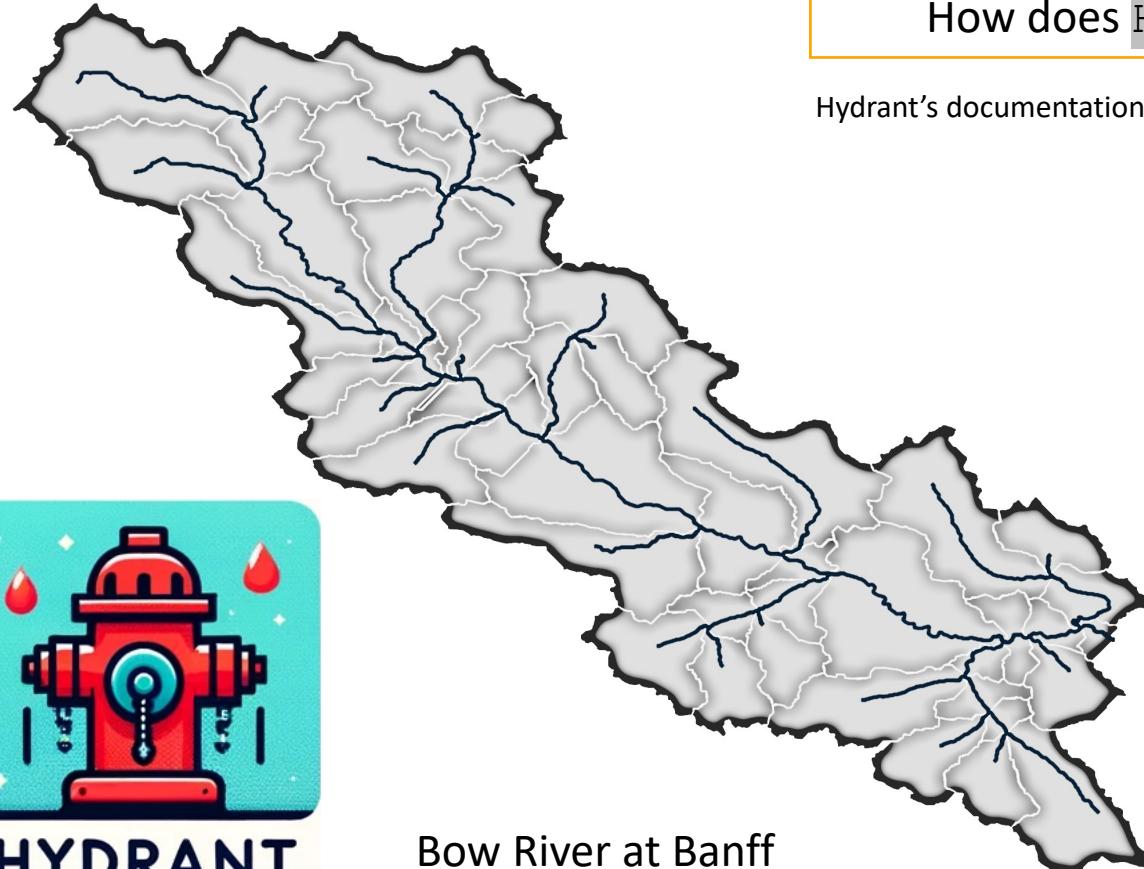
Practice Example – Bow River at Banff



Bow River at Banff

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

47



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?



UNIVERSITY OF
CALGARY



Any questions so far?

Practice Example – Bow River at Banff

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.

Notebook

- Python 3.10
- D
- scienv

Console

- Python 3.10
- scienv

Other

- \$
-
- M

File Browser Content

- 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

Practice Example – Bow River at Banff

The screenshot shows a Jupyter Notebook interface with the following elements:

- File Bar:** File, Edit, View, Run, Kernel, Tabs, Settings, Help.
- File Explorer:** Shows a directory structure under "/.../community-workflows / 1-geofabric". A red box highlights the "pre-process-geospa..." file, which is selected. A callout box says "Double click to launch the Notebook".
- Notebook Tab:** pre-process-geospa... (selected), README.md, Launcher, Markdown.
- Content Area:**
 - Section:** 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:
- Code Cell:** [1]:

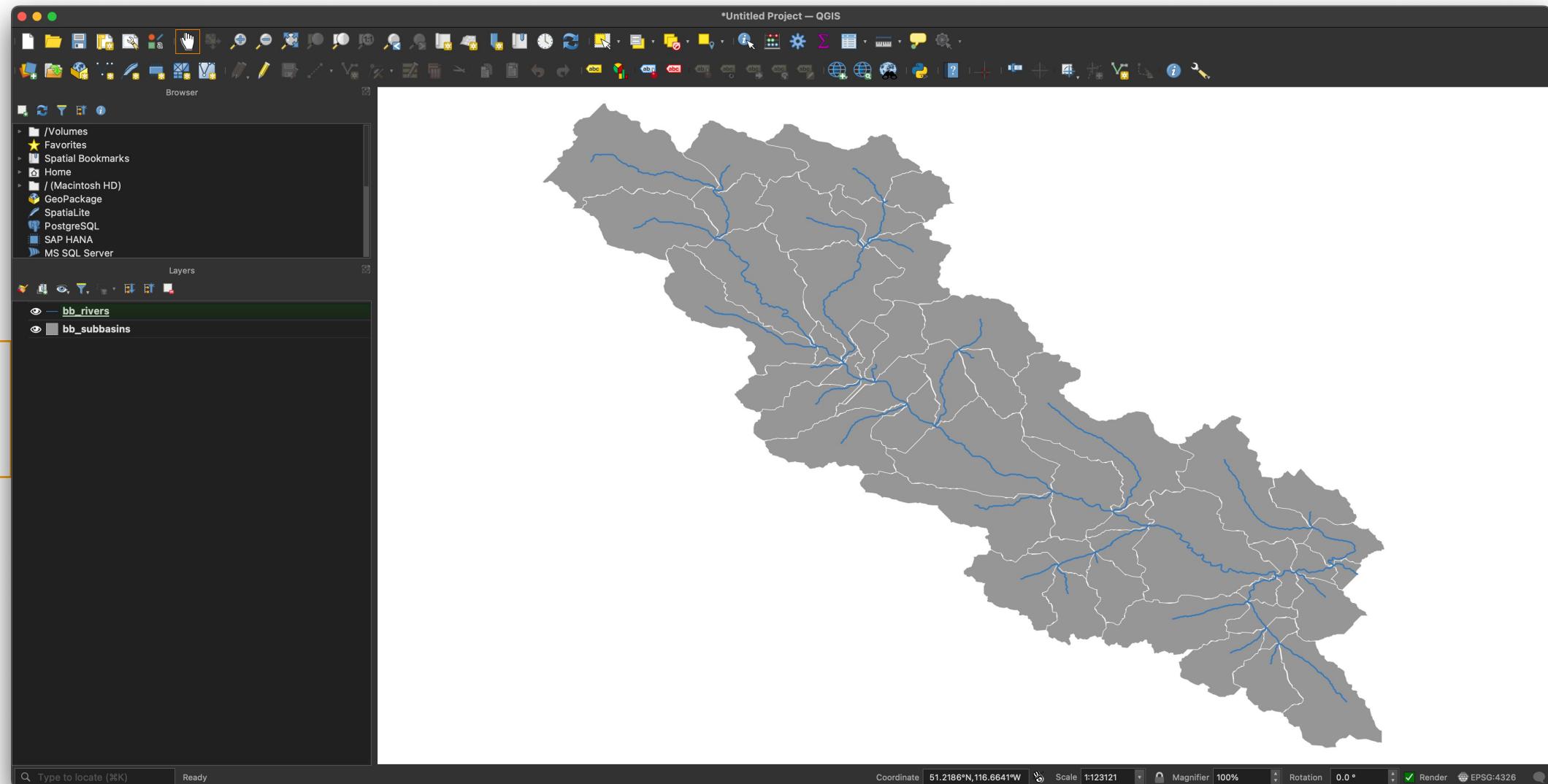
```
1 import geopandas as gpd # version 0.14.3
2 import pandas as pd # version 2.1.1
3 import numpy as np # version 1.24.4
4 import matplotlib.pyplot as plt # version 3.5.1
5
6 from shapely.geometry import Point # version 2.0.0
7
8 import hydrant.topology.geom as gm # version 0.1.0
9
10 import subprocess # built-in Python 3.10.2
11 import os # built-in Python 3.10.2
12 import glob # built-in Python 3.10.2
```
- Callout Boxes:**
 - A red box with an arrow points to the "Basic preparations" section, labeled "Notebook cells".
 - A red box with an arrow points to the code cell, labeled "Tip: review keyboard shortcuts!".
 - A red box with an arrow points to the "Shift + Enter" key, labeled "Use 'Shift + Enter' to execute Notebook cells".

Practice Example – Bow River at Banff

Switching to the relevant Jupyter Notebook
webpage

jupyterlab

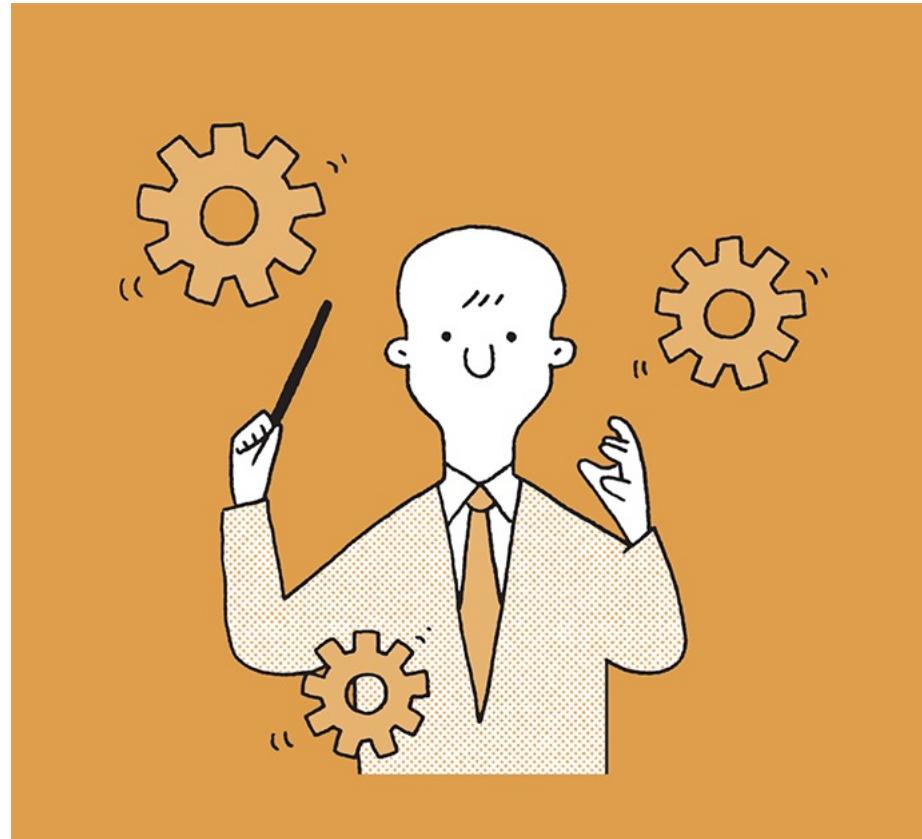
Practice Example – Bow River at Banff



Practice Example – Bow River at Banff

Second step is to run the model-agnostic processes

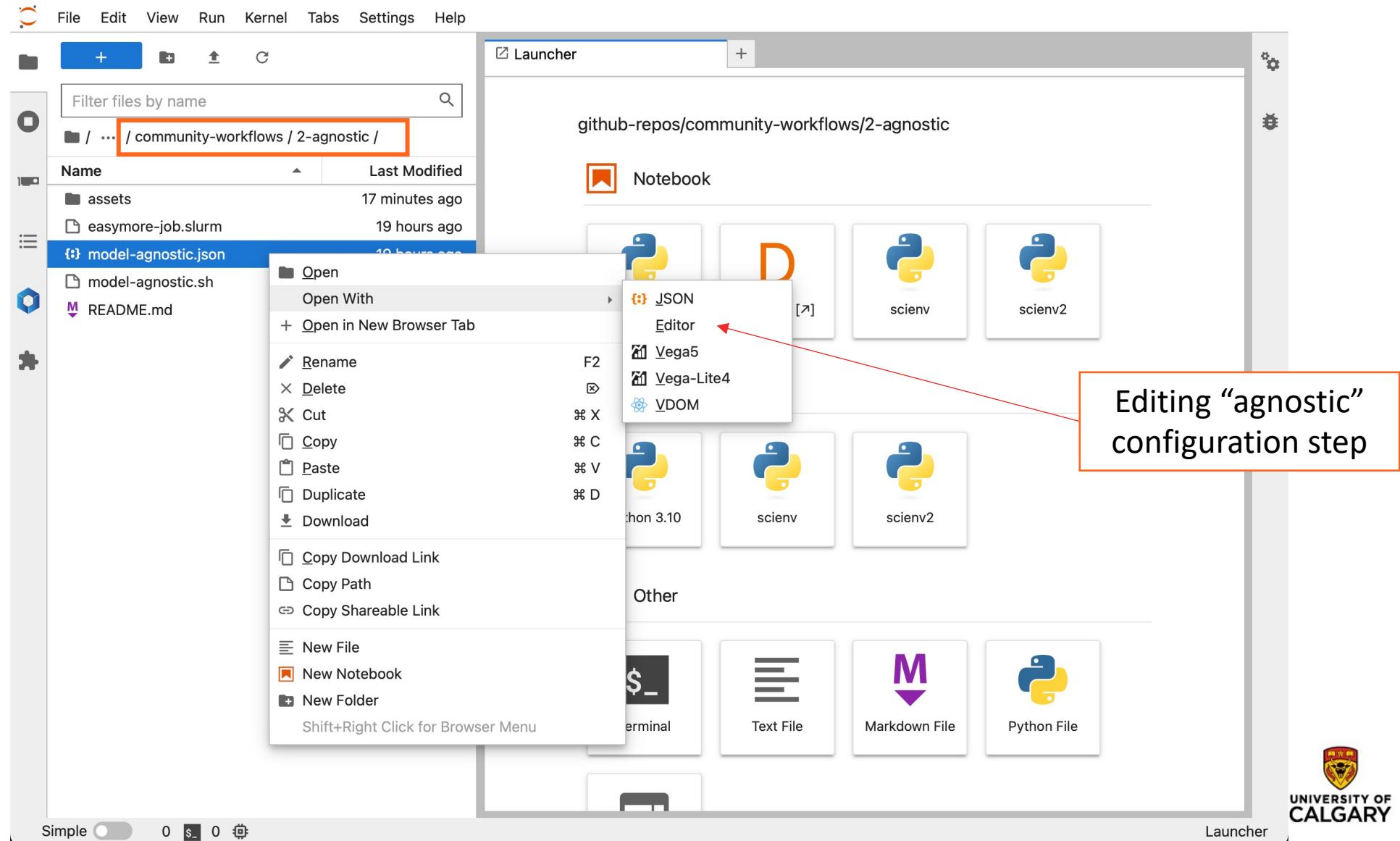
Practice Example – Bow River at Banff



- 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

Practice Example – Bow River at Banff

Agnostic workflow
execution using its
“Orchestrator”



Practice Example – Bow River at Banff

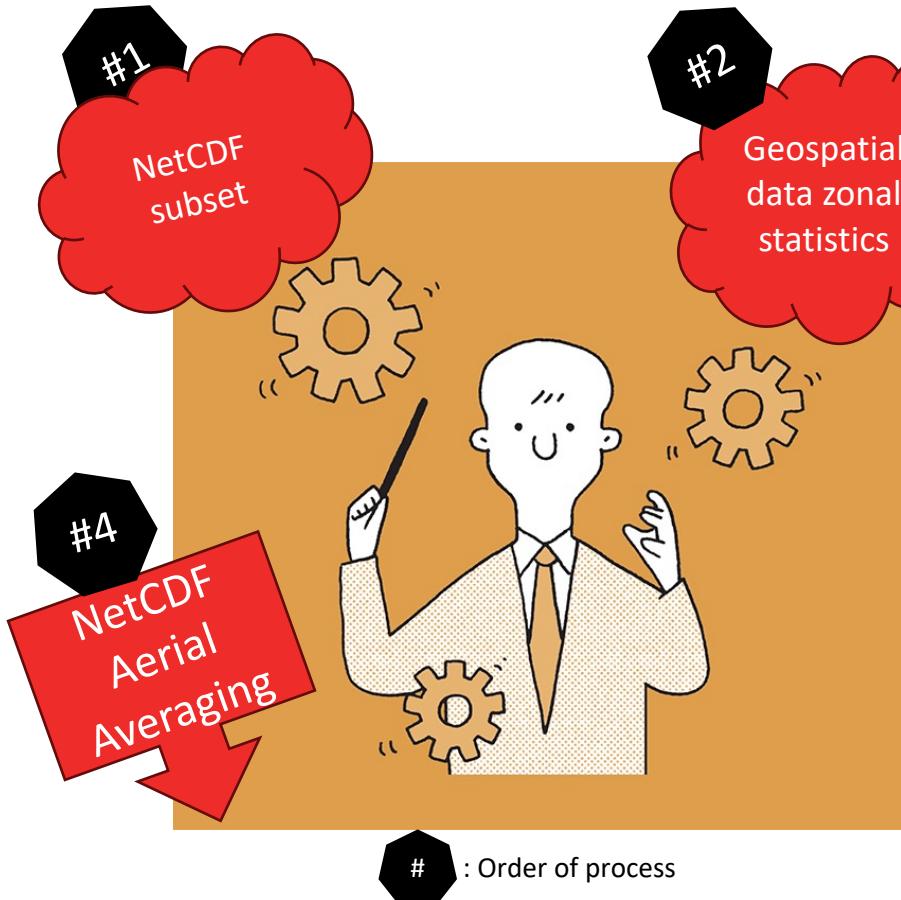
The screenshot shows a Jupyter Notebook interface with a sidebar and a main code editor area. The sidebar includes a file browser, a search bar, and various notebook-related icons. The main area has a tab titled "model-agnostic.json" which contains the following JSON code:

```
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 on the right side of the code editor highlight specific parts of the JSON structure:

- A red bracket on the right side of the code editor covers the entire "args" section, labeled "Executable paths".
- A yellow box surrounds the "args" section, labeled "Orchestrator configuration file".
- A red bracket on the right side of the "args" section covers the "dataset" key under the first "met" entry, labeled "Executable arguments and options (empty options are ignored)".
- A red bracket on the right side of the code editor covers the entire "args" section, labeled "This is the only file for the “agnostic” step".

Practice Example – Bow River at Banff



Automated and ordered job submission to HPC

The screenshot shows a Jupyter Notebook interface. On the left, there's a file browser window titled "Community-workflows / 2-agnostic /". It lists several files: "assets" (modified "an hour ago"), "easymore-job.slurm" (modified "19 hours ago"), "model-agnostic.json" (modified "19 hours ago" and currently selected), "model-agnostic.sh" (modified "19 hours ago"), and "README.md" (modified "19 hours ago"). The "model-agnostic.json" file is open in the main code editor window, which has tabs for "File", "Edit", "View", "Run", "Kernel", "Tabs", "Settings", and "Help". The code editor shows JSON content with line numbers from 92 to 125. A red arrow points to line 121, where a green brace highlights the "order" field. The JSON content includes paths to shapefiles and remapping configurations.

```
92
93     "remap": [
94         {
95             "case-name": "remapped",
96             "cache": "/home/kasra545/scratch/bb-models/easymore-outputs/cache/",
97             "shapefile": "/home/kasra545/github-repos/MESH-Bow-at-Banff/1-
98               geofabric/bow-at-banff-geofabric/bb_subbasins.shp",
99             "shapefile-id": "COMID",
100            "source-nc": "/home/kasra545/scratch/bb-models/datatool-outputs/**/*.nc",
101            "variable-lon": "lon",
102            "variable-lat": "lat",
103            "variable": [
104                "RDRS_v2.1_P_P0_SFC",
105                "RDRS_v2.1_P_HU_09944",
106                "RDRS_v2.1_P_TT_09944",
107                "RDRS_v2.1_P_UVC_09944",
108                "RDRS_v2.1_A_PR0_SFC",
109                "RDRS_v2.1_P_FB_SFC",
110                "RDRS_v2.1_P_FI_SFC"
111            ],
112            "remapped-var-id": "id",
113            "remapped-dim-id": "id",
114            "output-dir": "/home/kasra545/scratch/bb-models/easymore-outputs/",
115            "job-conf": "/home/kasra545/github-repos/MESH-Bow-at-Banff/2-
116              agnostic/easymore-job.slurm",
117            "_flags": [
118                "submit-job"
119            ]
120        },
121        {
122            "order": {
123                "met": 1,
124                "gis": -1,
125                "remap": 2
126            }
127        }
128    ]
129}
```

-1: no order
ordered integers: order of processes

Practice Example – Bow River at Banff

Switching to the JupyterLab webpage
for a quick overview

jupyterlab

Execute the “agnostic orchestrator” in the Terminal using:

```
$ ./model-agnostic.sh model-agnostic.json
```



Any questions so far?

Practice Example – Bow River at Banff

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



Practice Example – Bow River at Banff

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

meshflow_bb.ipynb

Build MESH Model Setup for the Bow River at Banff

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
```

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 "'
```

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

Practice Example – Bow River at Banff

The screenshot shows a Jupyter Notebook interface with two panes. The left pane is a file browser with a search bar and a list of files in the directory `/ ... / community-workflows / 3-specific /`. The right pane is a code editor titled `meshflow_bb.ipynb` containing Python code for setting up a meshflow workflow.

```
[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 with arrows pointing to specific code snippets:

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

UNIVERSITY OF CALGARY

Practice Example – Bow River at Banff

The screenshot shows a Jupyter Notebook interface. On the left is a file browser with a search bar and a list of files. The 'meshflow_bb.ipynb' file is selected. The main area is a code editor with the following Python code:

```
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 on the right side of the code editor highlight specific parts of the configuration:

- 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' section.
- All the variables to be used in the "drainage database" MESH file: Points to the 'ddb_vars' section.
- Units of variables to be included in the "drainage database" MESH file: Points to the 'ddb_units' section.

Page number 64 is in the bottom left corner. The University of Calgary logo is in the bottom right corner.

Practice Example – Bow River at Banff

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 Snippets:

```
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 and Callouts:

- 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 unit definitions in the first code snippet.
- Minimum values of “drainage database” MESH file**: Points to the minimum value definitions in the second code snippet.
- Renaming GRU dimension name in the MESH “drainage database” file to satisfy diverse versions of MESH**: Points to the `'gru_dim': 'NGRU'` line.
- Renaming sub-basin dimension name in the MESH “drainage database” file if needed**: Points to the `'hru_dim': 'subbasin'` line.
- The value of the outlet river segments created using “Hydrant” (Hydrant default is `-9999`)**: Points to the `'outlet_value': -9999` line.

Text Labels:

We can build an “instance” of the workflow class:

[3]: `1 exp1 = meshflow.MESHWorkflow(**config)`

And, we can run it using:

[4]: `1 exp1.run()`

/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:

Practice Example – Bow River at Banff

Switching to the JupyterLab webpage
for a quick overview

jupyterlab

Any questions so far?

Outline

- Community Hydrological Modelling Recipe
- Digital Infrastructure on Digital Research Alliance of Canada's HPCs
- Setting Up Your Account on HPCs
- Introducing Community Workflows & Tools
- Practice Example – Bow River At Banff
- Troubleshooting

Troubleshooting

Whenever you face an issue, follow the steps below:

- “Agnostic” log files are located in the following directories:

\$HOME/.datatool

\$HOME/.gistool

/path/to/EASYMORE/s/temp/directory

/path/to/agnostic/orchestrator/s/directory

Look for files with the following extensions:

.out, .err, .log

Try to understand what has gone wrong. In 80% of the time, there is a minor human error involved.

Troubleshooting

Continuing...

- Make sure Graham storages are responsive, very often than not, the home, scratch, or project spaces experience technical difficulties
- Make sure you have enough space on any storage you are working; the quota command can show you how much storage you are using
- If the project space gets full, communicate this with your PI; users do mistakenly fill up the shared storage without much justification,
- Try troubleshooting technical issues using ChatGPT or any online ChatBot you have access to

Troubleshooting

- Try troubleshooting by Google-ing and searching Stackoverflow
- Avoid using chat mechanisms to exchange critical information; everything needs to be documented properly; undocumented work equals to null
- If nothing worked out, open an “Issue” on the relevant GitHub repository(-ies) providing an MCVE
- Feel free to contribute to any repository you are interested in. Whenever you commit, follow the commit guidelines recommended by the creator of Git
- Fix MESH bugs

Troubleshooting



Linus Torvalds – creator of Linux and Git

Header line: explain the commit in one line (use the imperative)

Body of commit message is a few lines of text, explaining things in more detail, possibly giving some background about the issue being fixed, etc.

The body of the commit message can be several paragraphs, and please do proper word-wrap and keep columns shorter than about 74 characters or so. That way "git log" will show things nicely even when it's indented.

Make sure you explain your solution and why you're doing what you're doing, as opposed to describing what you're doing. Reviewers and your future self can read the patch, but might not understand why a particular solution was implemented.

Reported-by: whoever-reported-it

Signed-off-by: Your Name <youremail@yourhost.com>

Same philosophy applies to GitHub Issues

Thank you for attending

<https://ucalgary.ca/civil>



UNIVERSITY OF
CALGARY