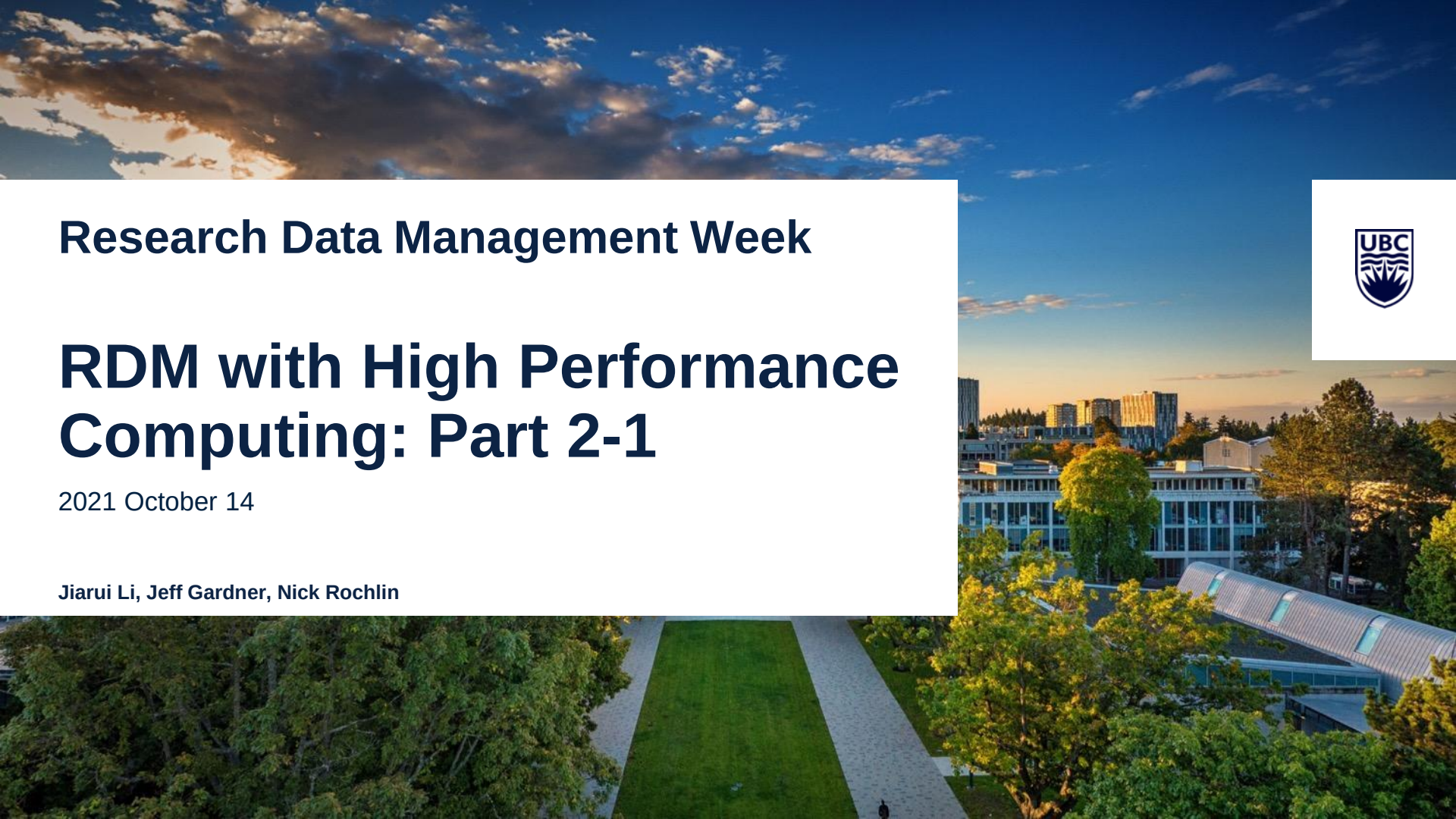


Research Data Management Week

RDM with High Performance Computing: Part 2-1

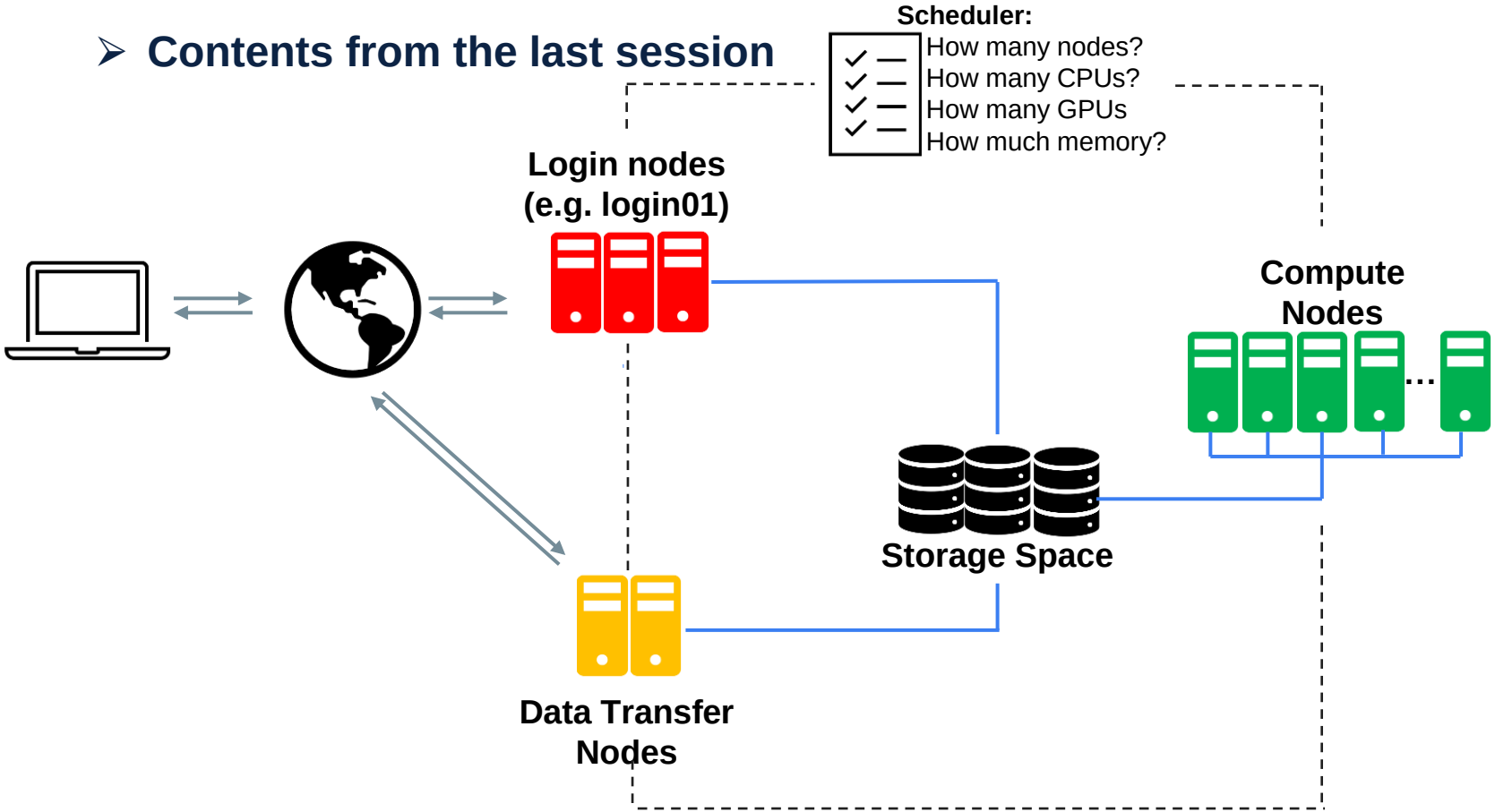
2021 October 14

Jiarui Li, Jeff Gardner, Nick Rochlin

Outline

- Your first task: organize your folder
- Transfer data to Sockeye
- Make readme files
- Get the software ready
- Run analysis
- Manage the outputs after the analysis
- Transfer data out from Sockeye

➢ Contents from the last session



➢ Contents from the last session

- Login
  - VPN
    - Windows
    - Linux
    - Mac
  - ssh <cwl>@sockeye.arc.ubc.ca
    - Linux: Terminal
    - Mac: Terminal
    - Win10: MobaXTerm, Putty or Powershell

Contents from the last session (demo)

Your directories:

```
/home/<CWL>
/arc/project/tr-rdm4hpc-1
/scratch/tr-rdm4hpc-1
```

Check your allocation:

```
print_quota
groups
print_members
ls
ls -l
ls -al
du -h
```

```
[jli106@sbc01 ~]$ ll /
total 96
drwxr-xr-x    6 root root  4096 May 19 12:39 arc
lrwxrwxrwx    1 root root     7 May 13  2018 bin -> usr/bin
dr-xr-xr-x    5 root root  4096 May 19 12:40 boot
drwxr-xr-x    6 root root  4096 Mar  2 00:34 cm
drwxr-xr-x    4 root root  4096 Mar 11 13:28 cvmfs
drwxr-xr-x   21 root root  3660 May 19 12:41 dev
drwxr-xr-x  129 root root 12288 May 19 12:39 etc
lrwxrwxrwx    1 root root     9 Apr 23  2019 home -> /arc/home
lrwxrwxrwx    1 root root     7 May 13  2018 lib -> usr/lib
lrwxrwxrwx    1 root root     9 May 13  2018 lib64 -> usr/lib64
drwxrwxrwt    2 root root  4096 Feb 18  2019 local
drwx------    2 root root 16384 Jun 24  2020 lost+found
drwxr-xr-x    2 root root  4096 Apr 10  2018 media
drwxr-xr-x    2 root root  4096 Apr 10  2018 mnt
drwxr-xr-x    7 root root  4096 Mar 18  2020 opt
dr-xr-xr-x 1000 root root     0 May 19 12:37 proc
lrwxrwxrwx    1 root root    12 Apr 23  2019 project -> /arc/project
dr-xr-x---    6 root root  4096 Dec 16 16:08 root
drwxr-xr-x   36 root root  1260 May 19 12:41 run
lrwxrwxrwx    1 root root     8 May 13  2018 sbin -> usr/sbin
drwxr-xr-x  177 root root 12288 Jun  4 15:11 scratch
lrwxrwxrwx    1 root root    13 Apr 23  2019 software -> /arc/software
drwxr-xr-x    2 root root  4096 Nov 18  2020 srv
dr-xr-xr-x   13 root root     0 May 19 12:37 sys
drwxr-xr-x    2 root root  4096 Nov 15  2019 tftpboot
drwxrwxrwt   15 root root  4096 Jun  7 10:35 tmp
drwxr-xr-x   14 root root  4096 Feb 24  2019 usr
drwxr-xr-x   22 root root  4096 Feb 24  2019 var
[jli106@sbc01 ~]$
```

Questions?

Anyone cannot connect to VPN?

Anyone cannot connect to Sockeye?

The first task: organize your project folders

Before you start build your folder, try to answer:

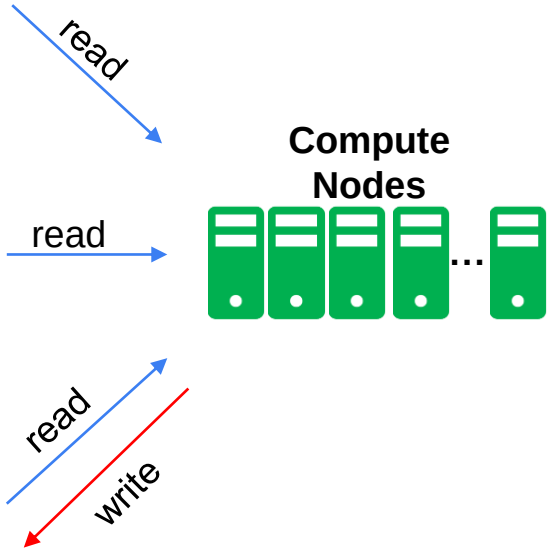
- What type of data/files I have?
  - Raw input data
  - Meta data
  - Script(s)/code
  - Intermediate files
  - Testing data/files
  - Final results
  - Log files

- How big are they and whether you would like others to access to them?

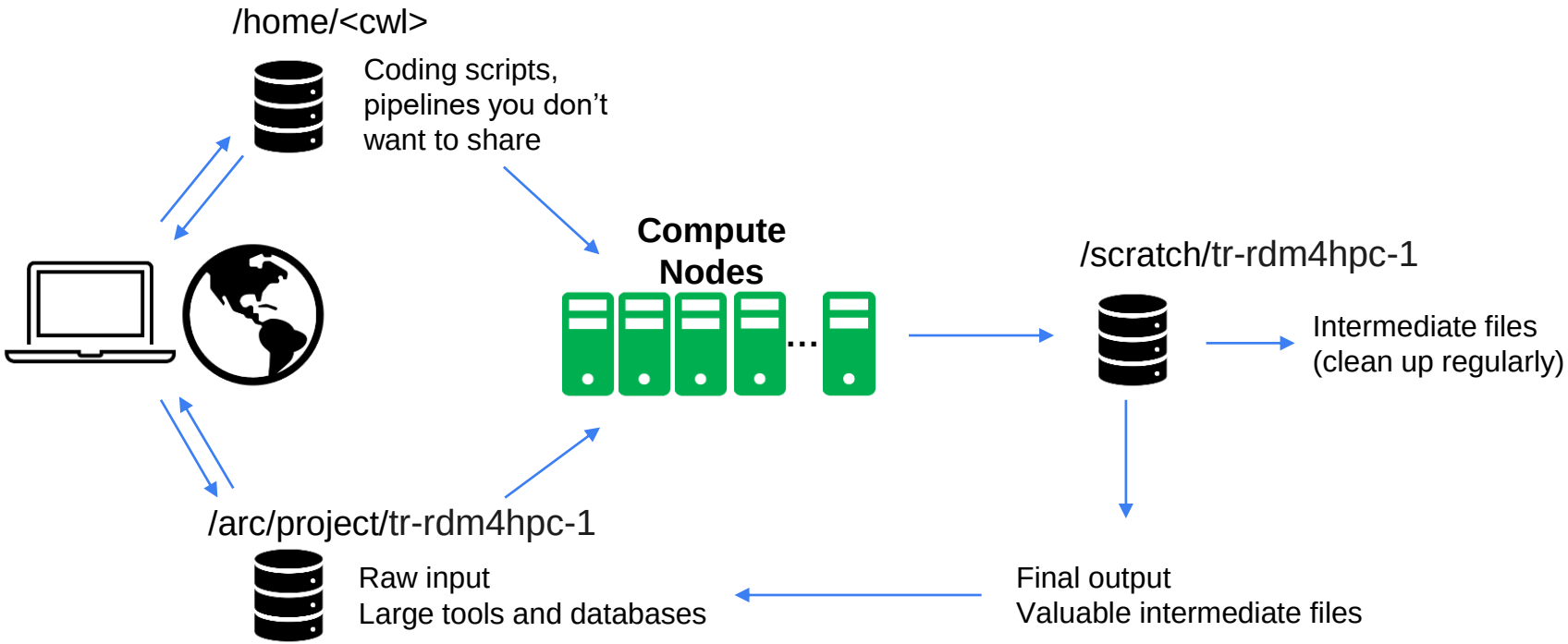
- Where should I put my files and why?

Your workspace in Sockeye

- Your workspace - Suitable for:
  - /home/<cwl>
    - Software
    - Configuration files
    - Interactive analysis
  - /arc/project/tr-rdm4hpc-1
    - Project data, software
    - Persistent data
    - Interactive analysis
  - /scratch/tr-rdm4hpc-1
    - Batch jobs
    - High IOPS, throughput
    - Interactive analysis



- **The life cycle of your analysis**



- **Organize your folders – example project**

RDM project: how many steps of putting a cow into a fridge?

Step 1: open the fridge

Step 2: put the cow in

Step 3: close the fridge

Before you start doing analysis, make folders first:

- ✔ Raw input data
- ✔ Meta data — project/raw
- ✔ Script/code
- ✔ Intermediate files
- ✔ Testing data/files — scratch with version control
- ✔ Final results — project/results
- ✔ Log files — As needed

- **Organize your folder (hands-on)**

✔ Making a folder with project name and a short description inside

```
mkdir /arc/project/tr-rdm4hpc-1/<cwl>
mkdir /scratch/tr-rdm4hpc-1/<cwl>
ls -l /arc/project/tr-rdm4hpc-1/<cwl>
ls -l /scratch/tr-rdm4hpc-1/<cwl>
cd /arc/project/tr-rdm4hpc-1/<cwl>
pwd
mkdir RDM
cd RDM
vi readme
```

✔ You can put the following information in readme:

- ✔ When did this project start
- ✔ Who participated the project
- ✔ What is the goal of this project
- ✔ Important changes and the path of valuable files

- **Organize your folder in project (hands-on)**

✔ Make a folder for storing initial raw data (including the meta data)

```
# we are here: /arc/project/tr-rdm4hpc-1/<cwl>/RDM
mkdir raw_data
cd raw_data
```

✔ If your data come in batches

✔ Use dates YYYY-MM-DD (why?)

```
~~mkdir Jan,2021 Apr,2021~~
mkdir 2021-01 2021-04
```

✔ Use batch name

```
~~mkdir batch1 batch2 batch10~~
mkdir batch01 batch02 batch10
```

✔ Create readme files for each dataset

- **Organize your folder in project**

✔ Make a folder for reference or public datasets (demo)
  - ✔ One folder/data type
  - ✔ Use the official release version
  - ✔ Create readme files

✔ Making a folder for the analysis results (hands-on)
  - ✔ Make a subfolder for each step

```
mkdir /arc/project/tr-rdm4hpc-1/<cwl>/RDM/results
cd /arc/project/tr-rdm4hpc-1/<cwl>/RDM/results
mkdir open_fridge put_cow_in close_fridge         # looks good?
rm -r -i open_fridge put_cow_in close_fridge
mkdir 01.open_fridge 02.put_cow_in 03.close_fridge
```

  - ✔ Create readme files to describe how you get the final result and why this is “final”

- **Organize your folder in scratch (hands-on)**

✔ Folder structure of analysis (in scratch)

```
cd /scratch/tr-rdm4hpc-1/<cwl>
mkdir RDM    # same project name
cd RDM
mkdir 01.open_fridge 02.put_cow_in 03.close_fridge
cd 01.open_fridge
mkdir v1
```

✔ Sometimes you will find you need to break down one step into multiple. For example, you need to convert the following pdf to text in Step 01.open_fridge: `https://github.com/jerryakii/CowInFridge/raw/main/Fridge.pdf`

```
cd /scratch/tr-rdm4hpc-1/<cwl>/RDM/01.open_fridge/v1
mkdir a.pdf2txt
```

Questions?

➢ Transfer your input data onto Sockeye

- Download
  - Data in public database
  - Data stored in a remote server of commercial companies

- Upload
  - From your laptop
  - From the external driver
  - From your lab computer

- **Download the data to Sockeye (hands-on)**

- Use “wget”

```
cd /arc/project/tr-rdm4hpc-1/<cwl>/RDM/raw_data
wget https://github.com/jerryakii/CowInFridge/raw/main/Fridge.pdf
```

- Use “curl”

```
curl -LJO https://github.com/jerryakii/CowInFridge/raw/main/readme
```

- **Upload the data to Sockeye (demo)**

- For Mac or Linux users, use “scp” or “rsync”

```
scp <from_where> <to_where>

scp file1 <cwl>@sockeye.arc.ubc.ca:/home/<cwl>/rdm4hpc/

rsync –r myfolder <cwl>@sockeye.arc.ubc.ca:/home/<cwl>/rdm4hpc
```

- For Windows users, use WinSCP or MobaXTerm
  https://winscp.net/eng/index.php

For details, please refer to our TUD: https://confluence.it.ubc.ca/display/UARC/Data+Transfer

Questions?

➢ IMPORTANT: make a readme file!

A readme file is for:

1. describing the structure of the folder
2. explaining the purpose or motivation of making this folder
3. describing the contents of each file
4. describing how these files are generated
5. directing people quickly where the files are
6. recording the changes of the files

A readme file should:

1. be generated one per big folder
2. have dates
3. be short, brief, but clear
4. have records of tracking who created or edited it
5. be up-to-date

- **To generate a readme file**

A readme file must be generated:

1. For raw input data
2. For intermediate results
3. For version control
4. For final results

How to generate a readme file

1. vi/vim
2. nano
3. Other text editor in your PC followed by uploading

- **Generate a readme file for your raw input data**

- ❑ vi/vim
  - ❑ How to open
  - ❑ How to save
  - ❑ How to search
  - ❑ How to quit
  - ❑ How to replace
  - ❑ How to move the cursor
  - ❑ The hidden files if not closing properly

- ❑ You should include:
  - ❑ When this file was created
  - ❑ Who created this file
  - ❑ Where and how did you get this file
  - ❑ What is the version of this file (if applicable)
  - ❑ What is this file used for

- **Generate a readme file for other folders (hands-on)**

1. Version control
2. Final results

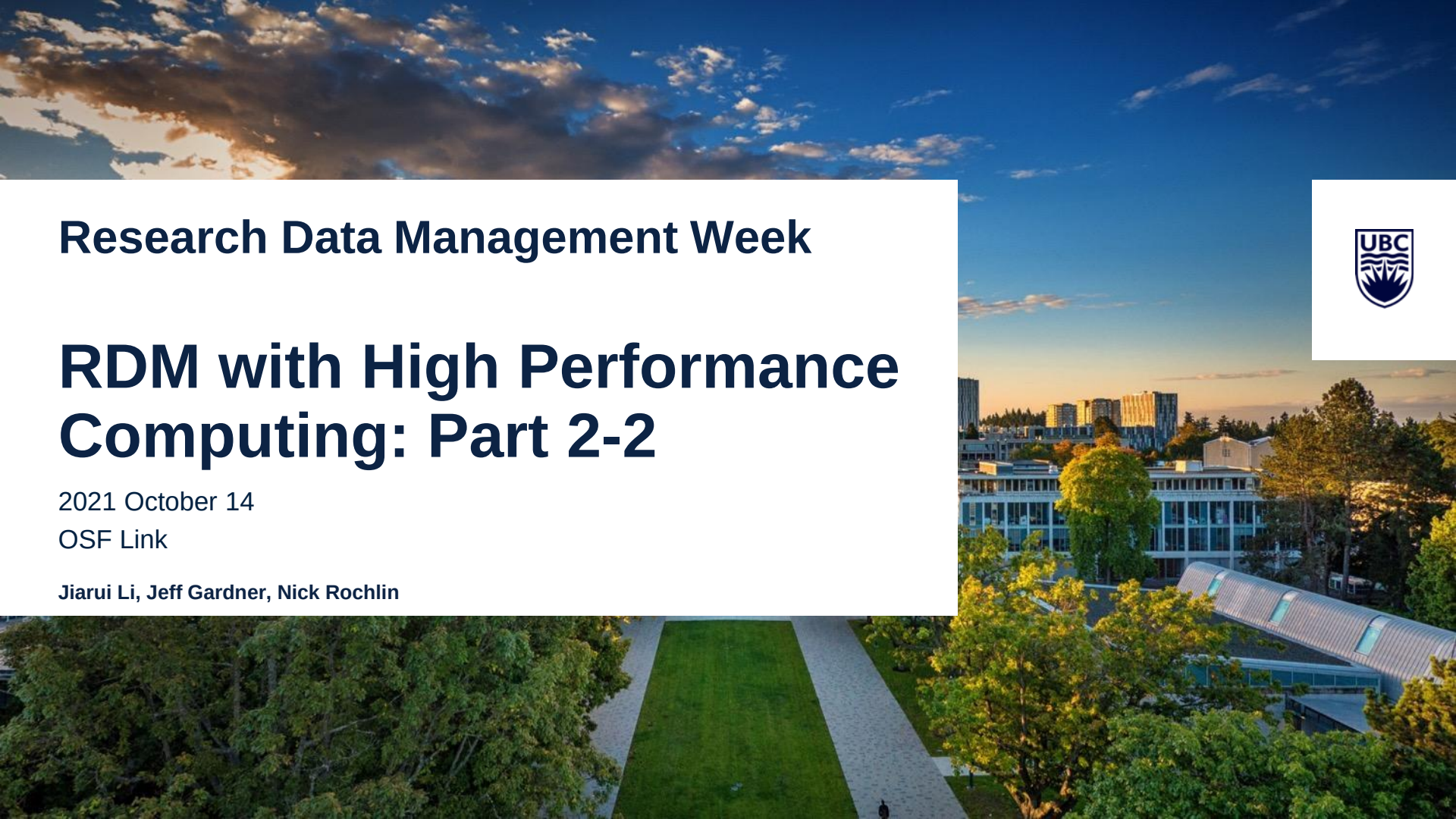
```
cd /arc/project/tr-rdm4hpc-1/<cwl>/RDM/
mkdir final_results
cd final_results
mkdir v1
vi readme
```

❑ You should include:
- ❑ When this file was created
- ❑ Who created this file
- ❑ Where and how did you get this file
- ❑ What is the version of this file (if applicable)
- ❑ What is this file used for

Question?

Summary

- Refresh the contents of the last session
- Your first task: organize your folder
- Transfer data to Sockeye
- Make readme files



Research Data Management Week

RDM with High Performance Computing: Part 2-2

2021 October 14

OSF Link

Jiarui Li, Jeff Gardner, Nick Rochlin

Recall from the last hour

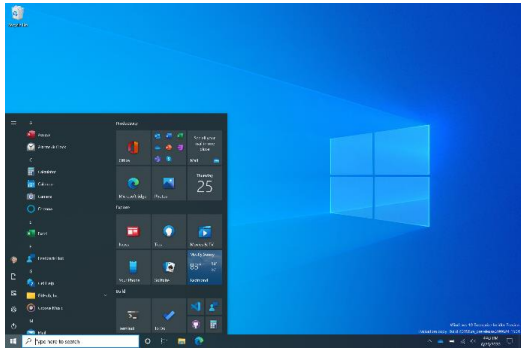
- We organized our folders
- We transferred data to Sockeye
- We made readme files

What's the next step?

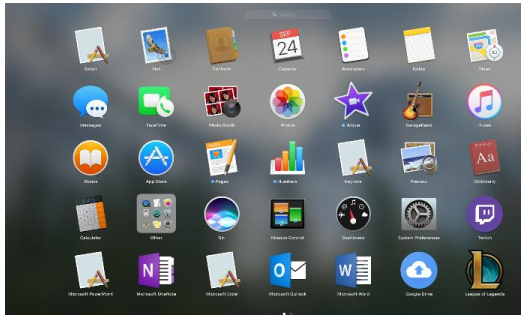
- Get the software ready
- Run analysis
- Manage the outputs after the analysis
- Transfer data out from Sockeye

➢ Get the software ready – find your software

Windows



MacOS



HPC?

```
#########################################################################
#                  The University of British Columbia
#                     Advanced Research Computing
#
#                   Access for authorized users only
#
#                Documentation: https://arc.ubc.ca/docs
#              Terms of Service: https://arc.ubc.ca/sockeye-tos
#                 Support Email: arc.support@ubc.ca
#########################################################################

    EXTENDED SYSTEM MAINTENANCE 2021-06-28 - 2021-06-30
  Please be advised that UBC ARC Sockeye will be unavailable on Monday,
  June 28th from 9:00 AM PT until Wednesday June 30th 6:00 PM PT for
  an extended scheduled maintenance. We will be performing Sockeye
  allocations rollover related tasks, routine maintenance as well as
  vendor-recommended security update patches.

  During the  mintenance, you will be unable to log in or transfer
  data to/from Sockeye. Please visit our web page for more details
  https://confluence.it.ubc.ca/display/UARC

hello profile
(base) [jli106@login01 ~]$ python --version
Python 3.8.5
(base) [jli106@login01 ~]$
```

- **Find the software – if it is installed centrally (hands-on)**

  - To find whether the software is installed centrally:

```
module spider <software name>
module spider python
```

  - To use software installed centrally:

```
python --version
which python

module load python/3.7.3

python --version
which python
```

  - To list all installed software:

```
module avail
```

- **Make sure you use the right version (hands-on)**

The following code only works in python 2 but not in python 3.

```
print "Hello, Python!"
```

Save this code into a test.py file, and run both versions of python:

```
cd /home/<cwl>
echo 'print "Hello, Python!"' > test.py

/usr/bin/python --version
/usr/bin/python test.py

python --version
which python
python test.py
```

- **Make sure you use the right version (hands-on)**

You may see this python shebang in many scripts:

```
#!/usr/bin/python
print "Hello, Python!"
```

Add this shebang to test.py and try:

```
/usr/bin/python --version
/usr/bin/python test.py

python --version
python test.py

chmod u+x test.py
./test.py
```

- **If the software is NOT installed centrally**

You can install it:

1. Locally (to your directories: /home or /project):
    a. Download only, or
    b. Set-up the installation path to your local directories

2. Or, in a virtual environment
    a. Conda
    b. Virtualenv

3. Or, in a container (more advanced)

Environment Variables - Brief overview

- A name and associated value
  - KEY=value
  - KEY=”some other value”
  - KEY=value1:value2

- Case sensitive

  Upper case by convention (e.g. HOME)

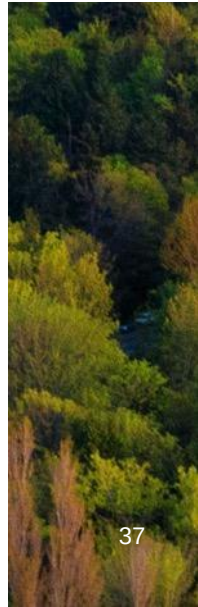
- Retrieve variable value by using $ before variable name (e.g. $HOME)

Environment Variables - Brief overview

Variable	Description
HOME	The pathname of your home directory (e.g. /home/<cwl>)
SHELL	The name of the shell you're using (e.g. /bin/bash)
PATH	The list of directories searched when you enter the name of an executable file or program
USER	Your username (e.g. CWL)

Environment Variables - Brief overview (hands-on)

```
printenv
printenv <VARIABLE NAME>
printenv HOME

echo $<VARIABLE NAME>

echo HOME
echo $HOME
cd /arc/project/tr-rdm4hpc-1
pwd
cd $HOME
pwd

echo $PATH
```



Install the tool locally (hands-on)

```
cd $HOME
mkdir cowsay
cd cowsay
wget https://github.com/Code-Hex/Neo-cowsay/releases/download/v1.0.3/cowsay_1.0.3_Linux_x86_64.tar.gz
tar -xf cowsay_1.0.3_Linux_x86_64.tar.gz
$HOME/cowsay/cowsay Hello
```

```
cowsay Hello
```

Install the tool locally (hands-on)

Why is “cowsay” not working?

- because you need to adjust your software environment

This is a simple example of modifying your environment

```
echo $PATH
export PATH=$HOME/cowsay:$PATH
cowsay HELP!
```

Install the tool locally (hands-on)

You can add this new $PATH to $HOME/.bashrc file

```
cd $HOME
echo "export PATH=$HOME/cowsay:$PATH" >> .bashrc
source .bashrc
```

.bashrc stores commands you would like to run everytime when you start a new Sockeye session

- **Install by conda (hands-on)**

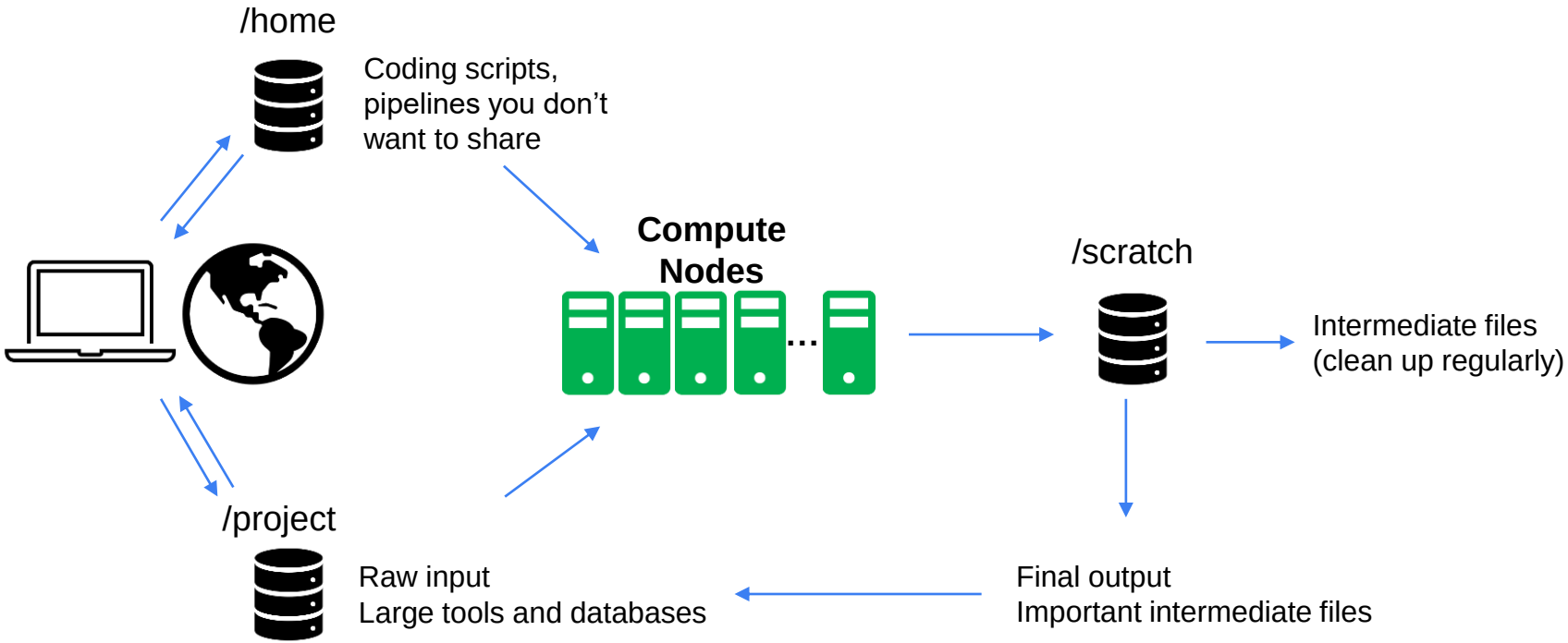
How can I install the latest version of python?

Answer: “conda”

```
mkdir $HOME/myconda

module load miniconda3/4.6.14
conda config --add channels conda-forge

conda create -p $HOME/myconda/myenv
source activate $HOME/myconda/myenv

conda install -c anaconda python

which python
python --version

# let’s try a python package that converts pdf to text
conda install -c conda-forge pdftotext
```

Questions?

➢ Run your analysis by submitting a job (hands-on)



➢ Run your analysis by submitting a job (hands-on)

Prepare the job script, input file, and your python code.

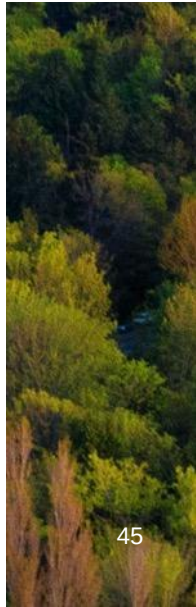
```
cd /scratch/tr-rdm4hpc-1/<cwl>/RDM/01.open_fridge/v1/a.pdf2txt
mkdir $HOME/scripts
cp /arc/project/tr-rdm4hpc-1/pdf2txt.py $HOME/scripts          # why not in RDM/01.open_fridge?
cp /arc/project/tr-rdm4hpc-1/pdf2txt.pbs pdf2txt_<cwl>.pbs
vi pdf2txt_<cwl>.pbs
```

```
#!/bin/bash

#PBS -l walltime=00:10:00,select=1:ncpus=1:mem=2gb
#PBS -N pdf2txt_<cwl>
#PBS -A tr-rdm4hpc-1

#########################################################
module load miniconda3/4.6.14
cd $PBS_O_WORKDIR
source activate $HOME/myconda/myenv

python $HOME/scripts/pdf2txt.py /arc/project/tr-rdm4hpc-1/<cwl>/RDM/raw_data/Fridge.pdf /scratch/tr-
rdm4hpc-1/<cwl>/RDM/01.open_fridge/v1/a.pdf2txt/fridge.txt
```

➢ Run your analysis by submitting a job (hands-on)

Submit and monitor your job:

```
# we are here: /scratch/tr-rdm4hpc-1/<cwl>/RDM/01.open_fridge/v1/a.pdf2txt
qsub pdf2txt_<cwl>.pbs
qstat -u <cwl>
```



Questions?

➢ Post-job data management (hands-on)

Files you want to keep:

- job script
- final output

Files you don't want to keep

- intermediate files
- log files if you are sure there is no value

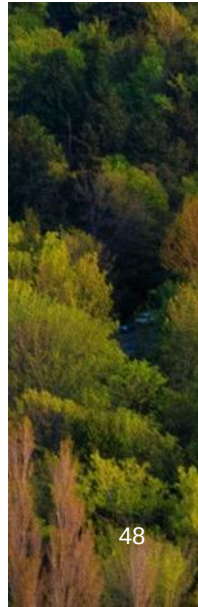
```
mkdir -p /arc/project/tr-rdm4hpc-1/<cwl>/RDM/01.open_fridge/v1/a.pdf2txt

# we are here: /scratch/tr-rdm4hpc-1/<cwl>/RDM/01.open_fridge/v1/a.pdf2txt
cp fridge.txt /arc/project/tr-rdm4hpc-1/<cwl>/RDM/01.open_fridge/v1/a.pdf2txt
cp pdf2txt_<cwl>.pbs /arc/project/tr-rdm4hpc-1/<cwl>/RDM/01.open_fridge/v1/a.pdf2txt

# the following is optional depending on what you get from "print_quota"
rm –i pdf2txt.tmp pdf2txt_<cwl>.pbs.o<123456> pdf2txt_<cwl>.pbs.e<1234567>
```

➢ Post-job data management (hands-on)

You can archive and zip files to save the space:

```
cd /arc/project/tr-rdm4hpc-1/<cwl>/RDM/01.open_fridge/v1/

tar -cvzf a.pdf2txt
```

Then update your readme file



➢ Transfer the results out from Sockeye (hands-on)

Use scp, WinSCP or MobaXTerm.

Questions?

➢ Summary

- In HPC, different directories serve different purposes.
- Usually you are dealing with large datasets, so a readme file is valuable to help build a reproducible workflow and for others to understand your data.
- A good organization of your folder can make your life easy where there are lots of files and file types.
- In HPC, you don’t have root permission, which means you often need to install programs locally and manage your software environment wisely, for which “conda” is a good option in most cases.