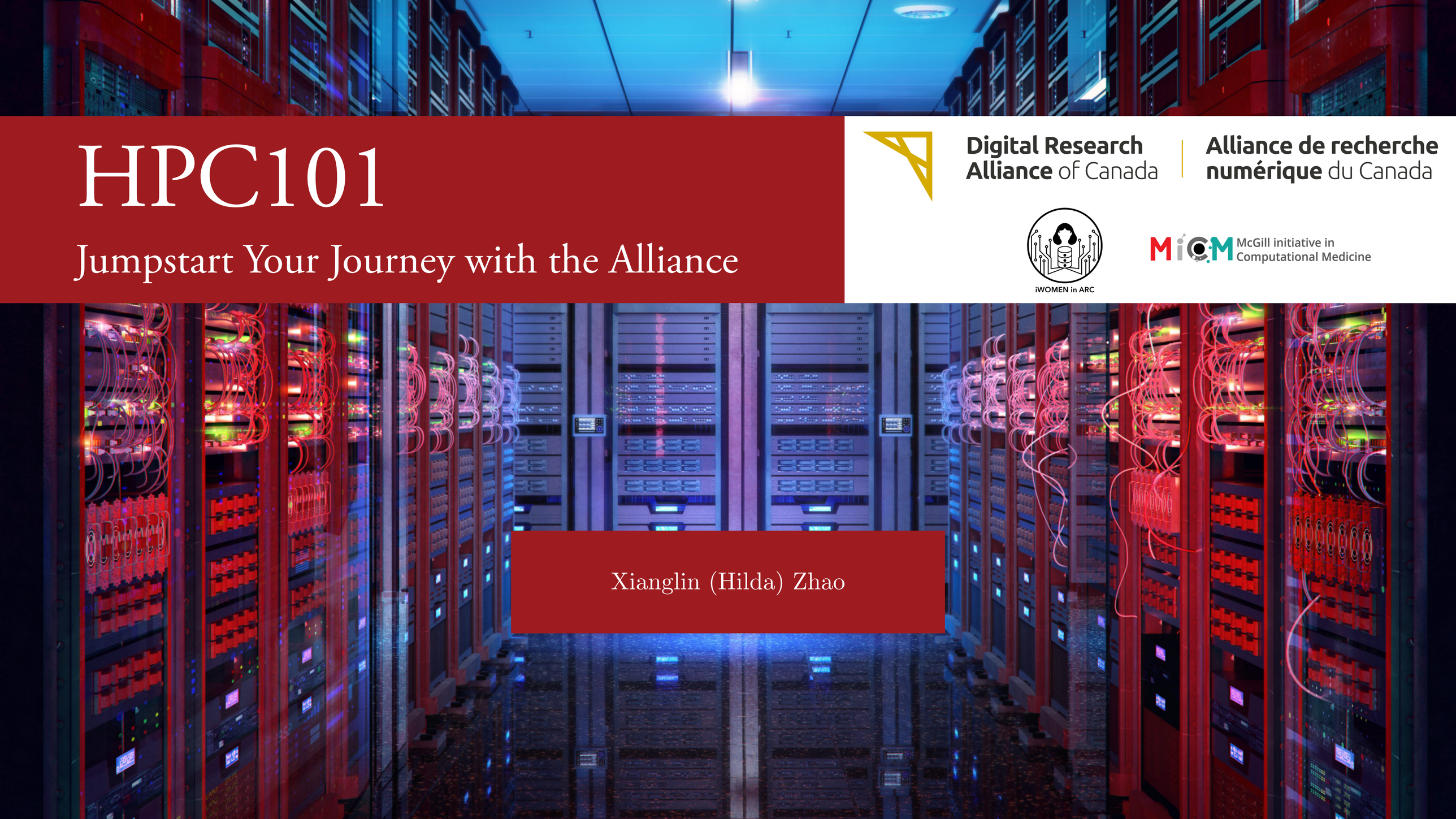


# HPC101

## Jumpstart Your Journey with the Alliance

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

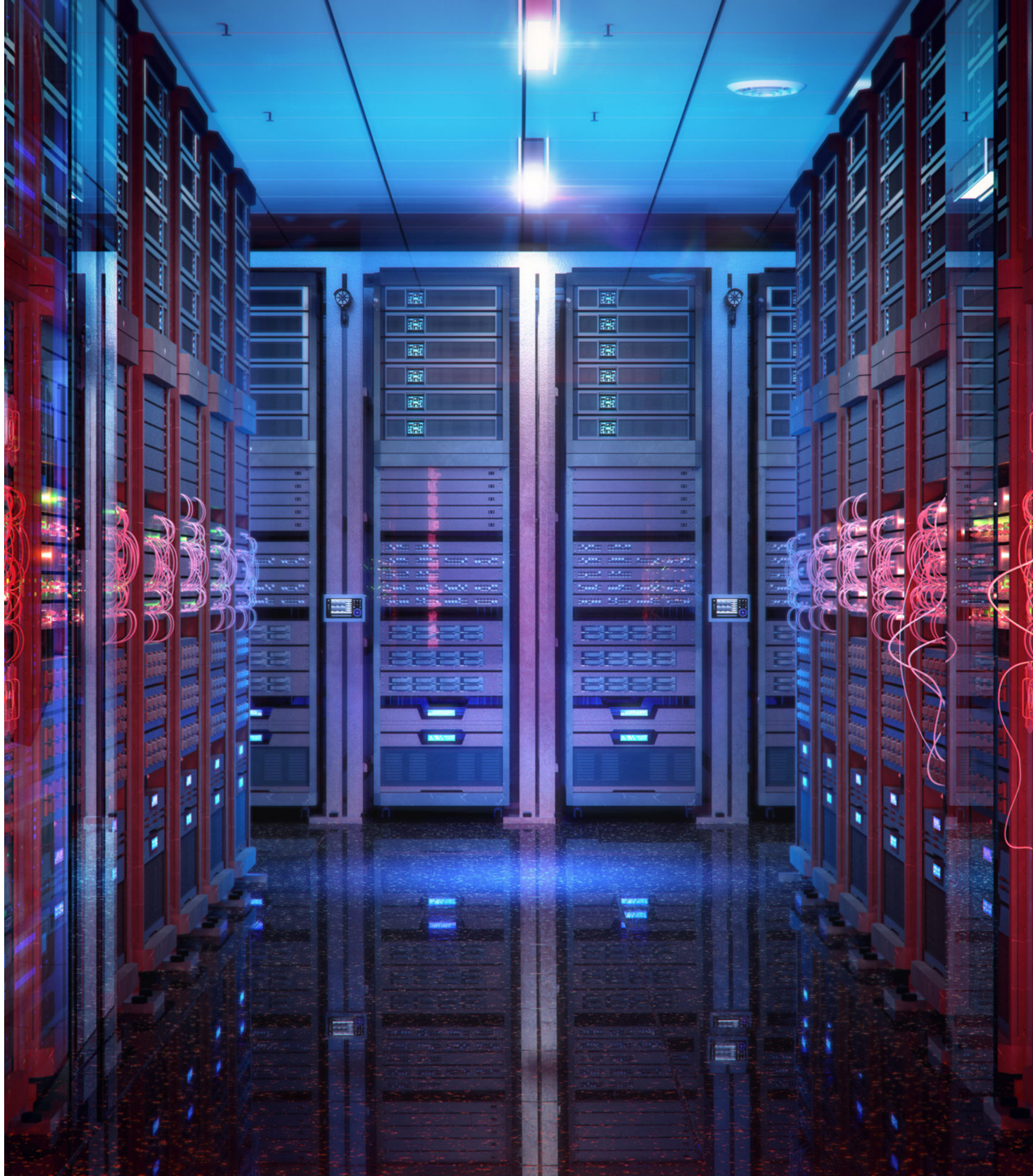
iWOMEN in ARC

MiCM McGill initiative in Computational Medicine

Xianglin (Hilda) Zhao

# Introduction to Shell

## Module 3

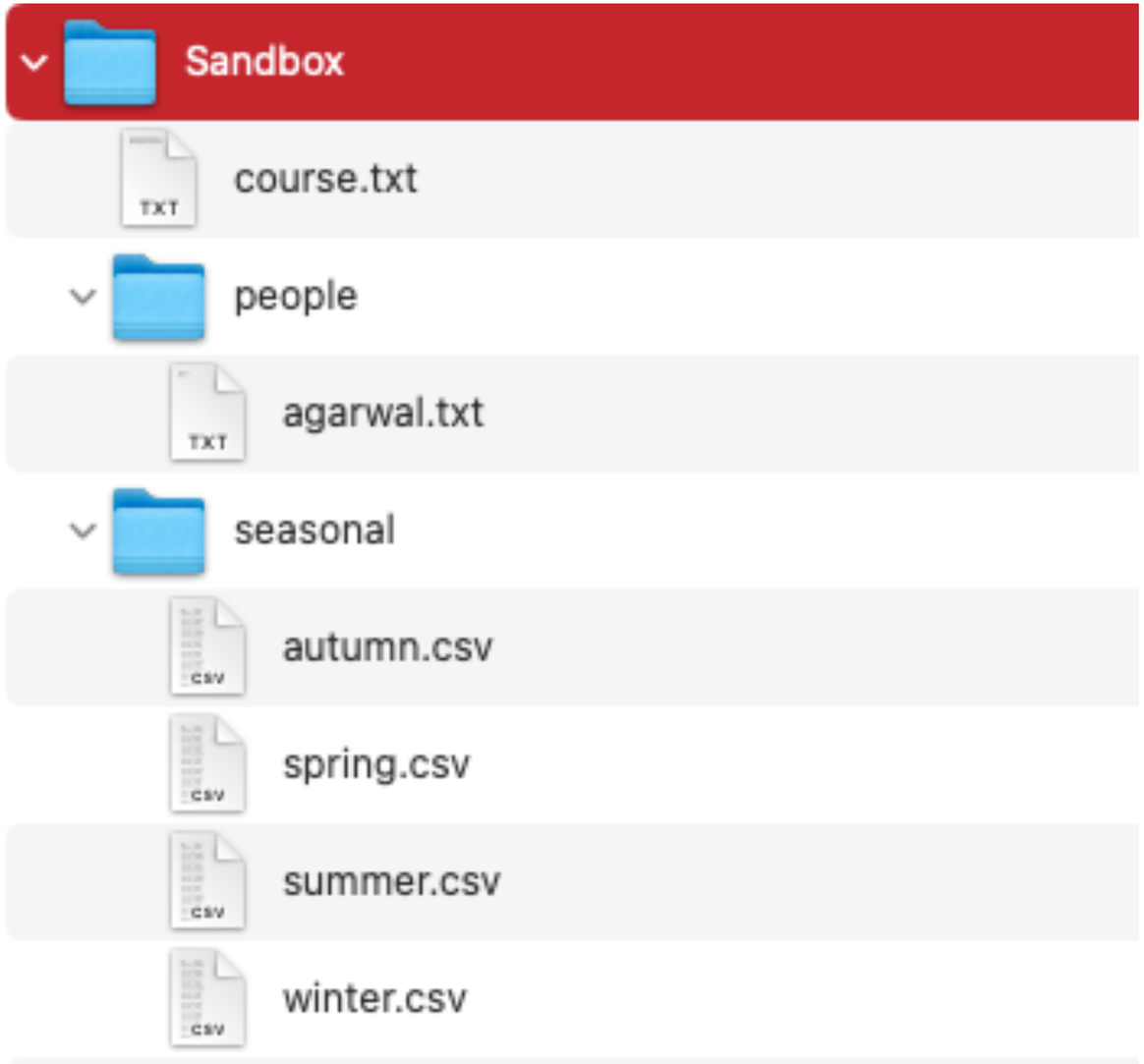


Digital Research Alliance of Canada | Alliance de recherche numérique du Canada

# Introduction to shell

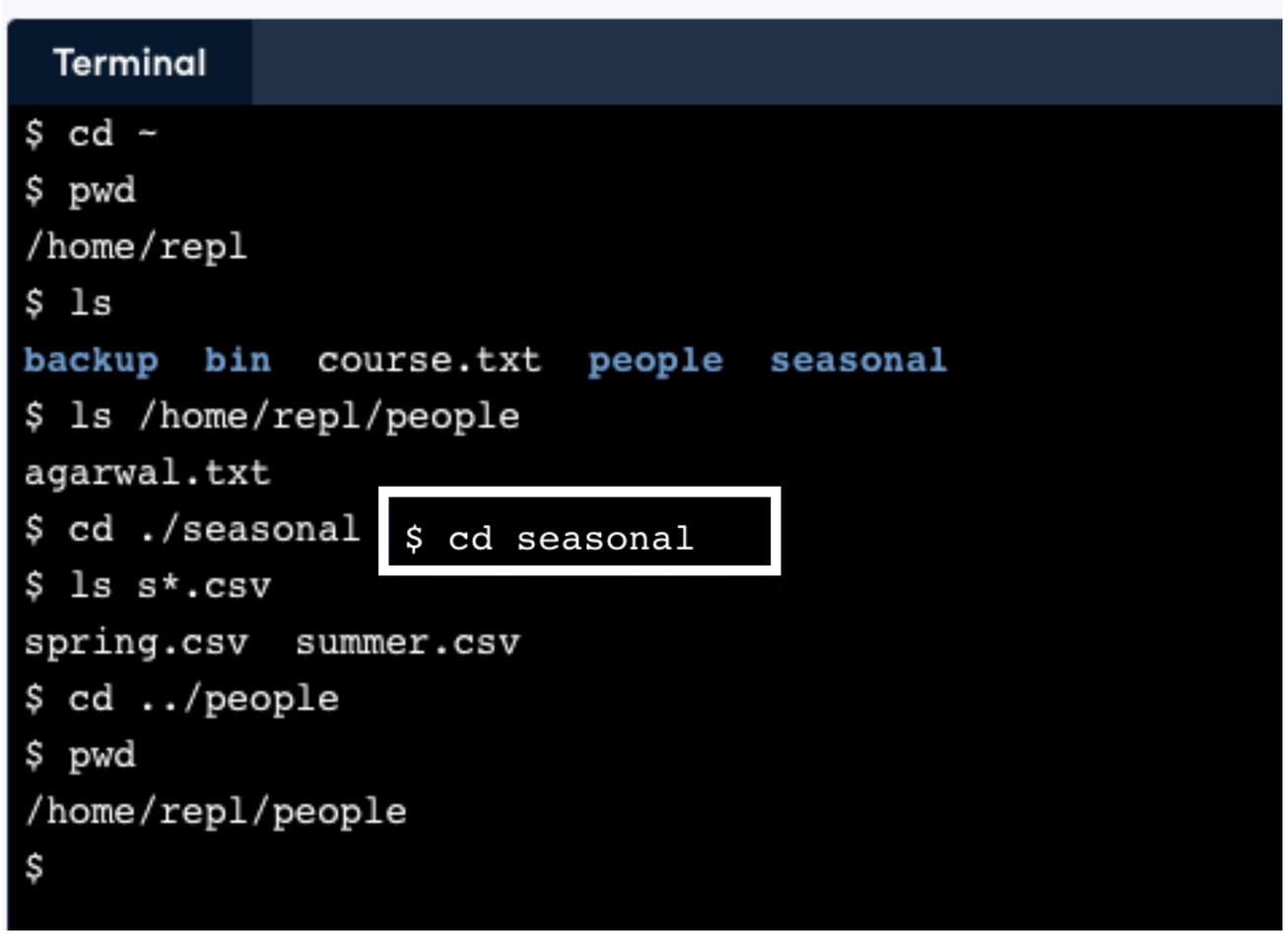
- How do we interact with operating system?
  - Your computer uses Graphical User Interface, translating clicking into commands
  - HPCs are based on Linux: no GUI, we have to use command line shell
  - You enter a command, the shell runs some other programs, prints the output, and lets you know when it is ready to take the next command
- Even if you don't use HPC, it is also a powerful tool on your own machine!
  - Mac OS: "Terminal"  Windows: "CMD" ?
- There will be lots of commands on the following slides – no need to remember all of them
- Note: be cautious to run suspicious commands! In this workshop we work in the "Sandbox"

# Sandbox in GUI



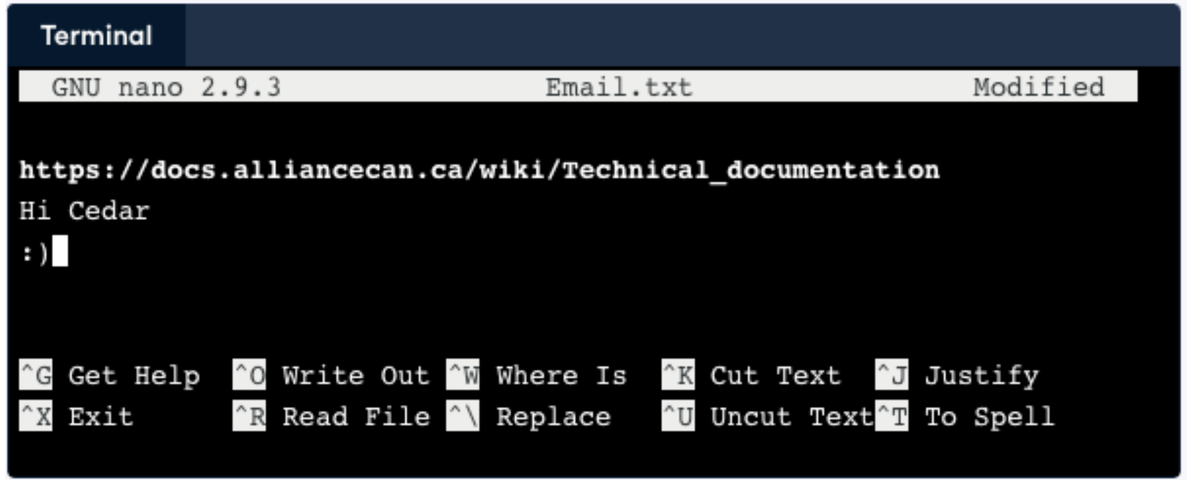
# Use keyboard & Travel along the paths

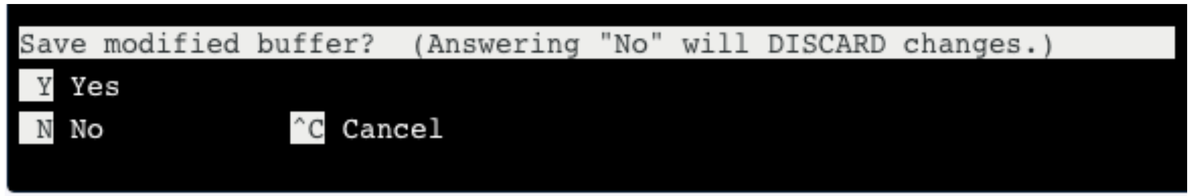
- Some shortcuts:
  - arrow keys [up/down; across sessions]
  - `ctrl+A` [go to the start of line]
  - `ctrl+E` [go to the end of line]
  - `ctrl+C` [stop the current process]
- Useful tools to save some typing
  - wildcard: `*` [Match any character]
  - `tab` [automatically complete path]
  - `tab*2` [show all options for ambiguous path]
- Some commands:
  - `pwd` [print working directory]
  - `ls` [listing items]
  - `cd` [change to directory]
- Paths
  - Absolute paths begin with `/`
  - Relative paths start from current directory
  - `.` [Current directory]
  - `..` [Parent directory]
  - `~` [Home directory]



# Modify the files

- Modify files and directories
  - `cp` [copy]
    - `cp course.txt course_new.txt`
    - `cp spring.csv autumn.csv people`
  - `mv` [move or rename]
    - `mv spring.txt autumn.txt ..`
    - `mv course.txt course-A.txt`
  - `rm` [remove] ⚠️
    - `rm course_new.txt`
  - `rmdir` [remove directory; only delete empty]
  - `mkdir` [?]
    - `mkdir year`
- Create a file
  - `touch Email.txt` [create a file without editing]
  - `nano Email.txt` [text file edit; create if not exist]



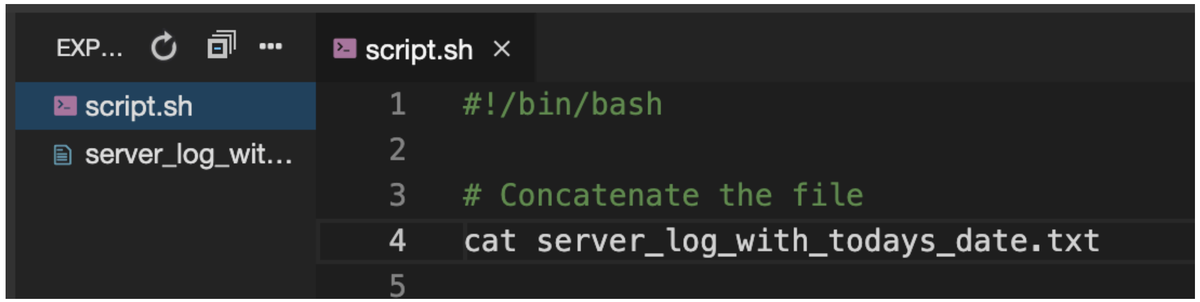



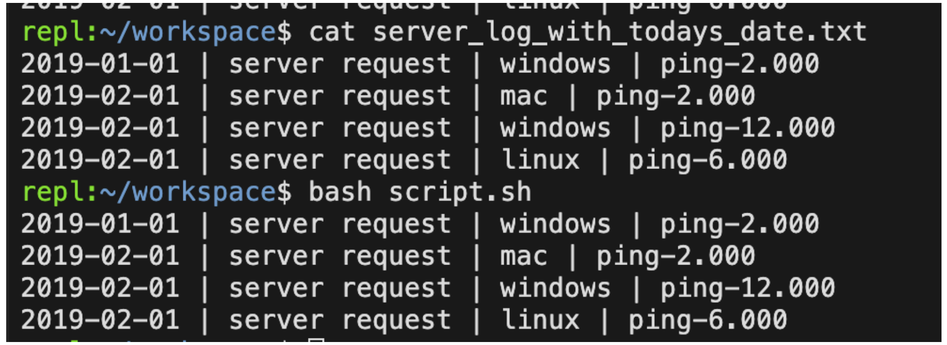

# Print something out & More handy tricks

- Print something out
  - `cat` [look at the content of some file]
  - `echo` [print]
  - `head` [look at the beginning of some file]
  - `tail` [?]
- General structure: <Command> <-options> <arguments>
  - `head -n 3 course.txt`
- `man` [manual of commands] [:q to quit]
  - `man head`
- `>` [store output in a new file]
  - `head -n 3 course.txt > Files.txt`
- `>>` [append output to a file]
  - `ls seasonal >> Files.txt`
- `|` [pipe: pass the output as the input of the next command]

# Introduction to Bash scripting

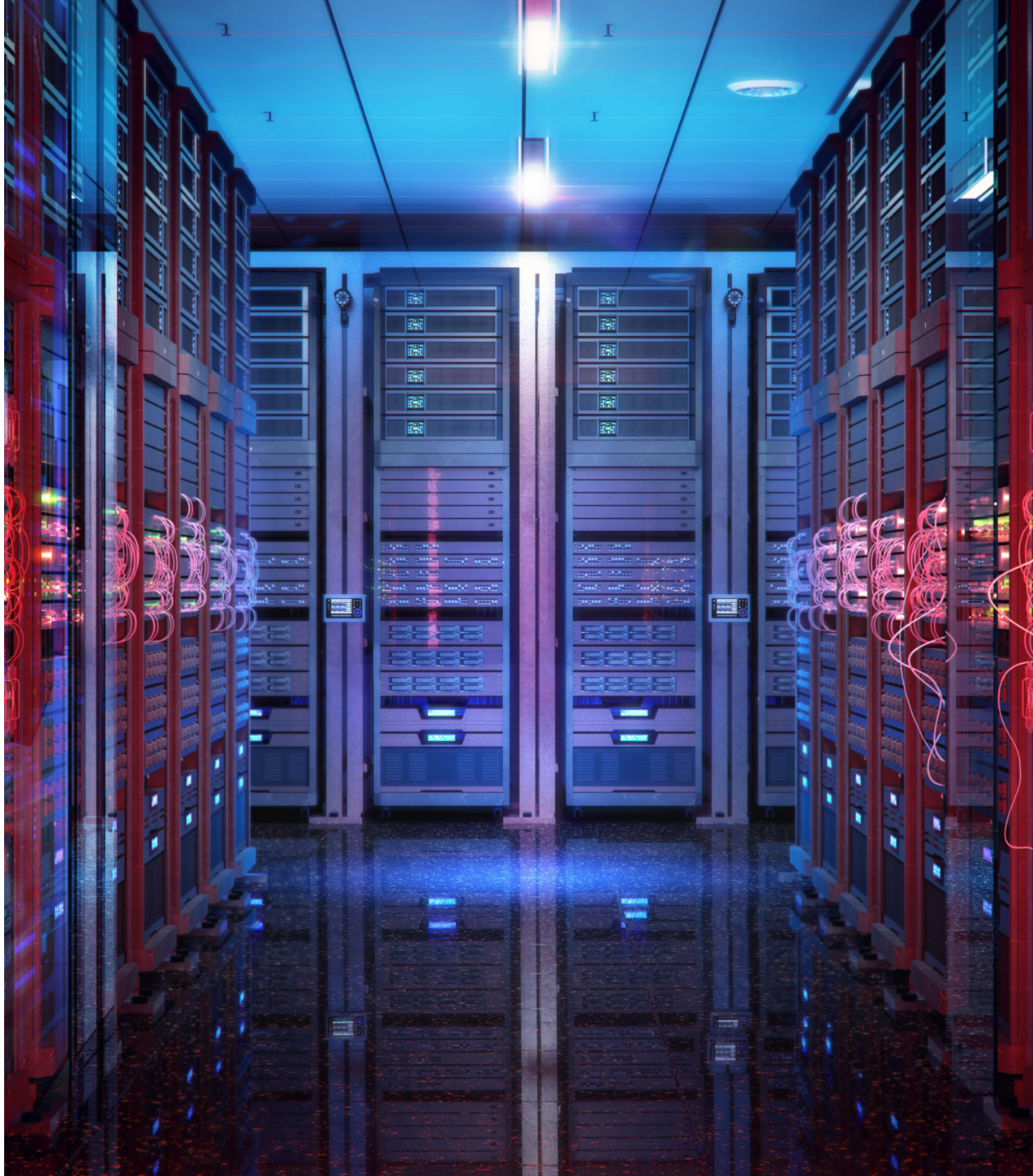
- `nano script.sh` and write some commands in it.
- If you `bash` [run] the `.sh` file, then the shell just run the commands
  - "bin" the location of executable files
  - "bash" Bourne Again Shell, is an app



# Submitting and Monitoring Jobs

## Module 4



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

# How to connect to AllianceCan clusters

- `ssh` [secure shell, a standard to connect to remote machines securely]
  - `ssh <user>@<cluster>.alliancecan.ca`
    - `ssh <user>@micm-edia.calculquebec.cloud`
  - you will be asked to input your password and multifactor authentication

# Submit a job

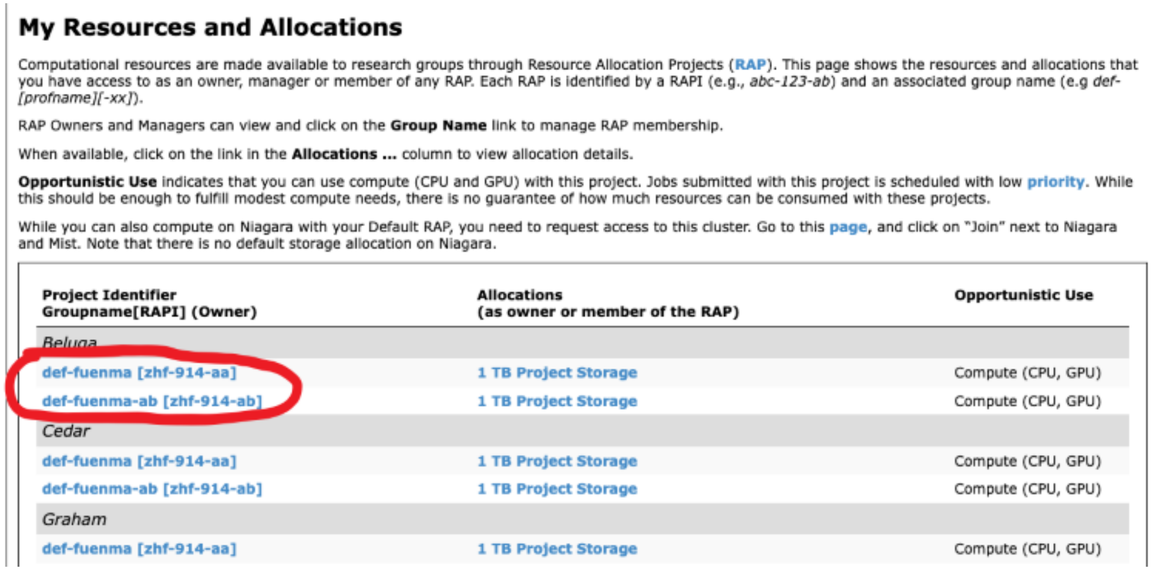
- `sbatch` [submit a job]

```
$ sbatch simple_job.sh
Submitted batch job 123456
```

  - simple_job.sh [a minimal example]

```
#!/bin/bash
#SBATCH --time=00:15:00
#SBATCH --account=def-someuser
echo 'Hello, world!'
sleep 30
```

  - On general-purpose clusters, this job reserves 1 core and 256MB of memory for 15 minutes.
  - account: log in to CCDB; My Account -> My Resources and Allocations.
  - you will receive a jobID such as 123456



**My Resources and Allocations**

Computational resources are made available to research groups through Resource Allocation Projects (RAP). This page shows the resources and allocations that you have access to as an owner, manager or member of any RAP. Each RAP is identified by a RAPI (e.g., *abc-123-ab*) and an associated group name (e.g *def-[profname][-xx]*).

RAP Owners and Managers can view and click on the **Group Name** link to manage RAP membership.

When available, click on the link in the **Allocations ...** column to view allocation details.

**Opportunistic Use** indicates that you can use compute (CPU and GPU) with this project. Jobs submitted with this project is scheduled with low priority. While this should be enough to fulfill modest compute needs, there is no guarantee of how much resources can be consumed with these projects.

While you can also compute on Niagara with your Default RAP, you need to request access to this cluster. Go to this page, and click on "Join" next to Niagara and Mist. Note that there is no default storage allocation on Niagara.

| Project Identifier Groupname[RAPI] (Owner) | Allocations (as owner or member of the RAP) | Opportunistic Use |
|---|---|---|
| *Beluga* | | |
| def-fuenma [zhf-914-aa] | 1 TB Project Storage | Compute (CPU, GPU) |
| def-fuenma-ab [zhf-914-ab] | 1 TB Project Storage | Compute (CPU, GPU) |
| *Cedar* | | |
| def-fuenma [zhf-914-aa] | 1 TB Project Storage | Compute (CPU, GPU) |
| def-fuenma-ab [zhf-914-ab] | 1 TB Project Storage | Compute (CPU, GPU) |
| *Graham* | | |
| def-fuenma [zhf-914-aa] | 1 TB Project Storage | Compute (CPU, GPU) |

```
#!/bin/bash
#SBATCH --account=def-someuser   # replace this with your own account
#SBATCH -mem-per-cpu=4G         # memory per cpu
#SBATCH -time=0-01:00            # time (DD-HH:MM)
#SBATCH -cpus-per-task=4
module load gcc/9.3.0 StdEnv/2023 r/4.3.1   # modules



Rscript Slow.R > stdout.txt 2>stderr.txt
```

# Monitor current jobs

- `squeue` [list jobs pending or running]
  - `squeue -u $USER` [all my jobs the scheduler is managing, including running and pending]
  - `squeue -u $USER -t RUNNING` [?]

```
$ sq
   JOBID     USER      ACCOUNT       NAME  ST   TIME_LEFT NODES CPUS    GRES MIN_MEM NODELIST (REASON)
  123456   smithj   def-smithj   simple_j   R        0:03     1    1  (null)      4G cdr234  (None)
  123457   smithj   def-smithj   bigger_j  PD  2-00:00:00     1   16  (null)     16G (Priority)
```

# Cancel a job

- `scancel 1234567` [cancel jobID 1234567]
- `scancel -u $USER` [cancel all my jobs running or pending]
- `scancel -t PENDING -u $USER` [cancel my pending jobs]

# Summarize completed jobs

```
$ seff 12345678
Job ID: 12345678
Cluster: cedar
User/Group: jsmith/jsmith
State: COMPLETED (exit code 0)
Cores: 1
CPU Utilized: 02:48:58
CPU Efficiency: 99.72% of 02:49:26 core-walltime
Job Wall-clock time: 02:49:26
Memory Utilized: 213.85 MB
Memory Efficiency: 0.17% of 125.00 GB
```