University of Manitoba

Digital Research Alliance of Canada

Beginning with High Performance Computing: HPC Quick Start Guide

All you should know to get started and use HPC clusters

UofM-Spring-Workshop 2023

May 17th-19th, 2023

Ali Kerrache

HPC Analyst

University of Manitoba

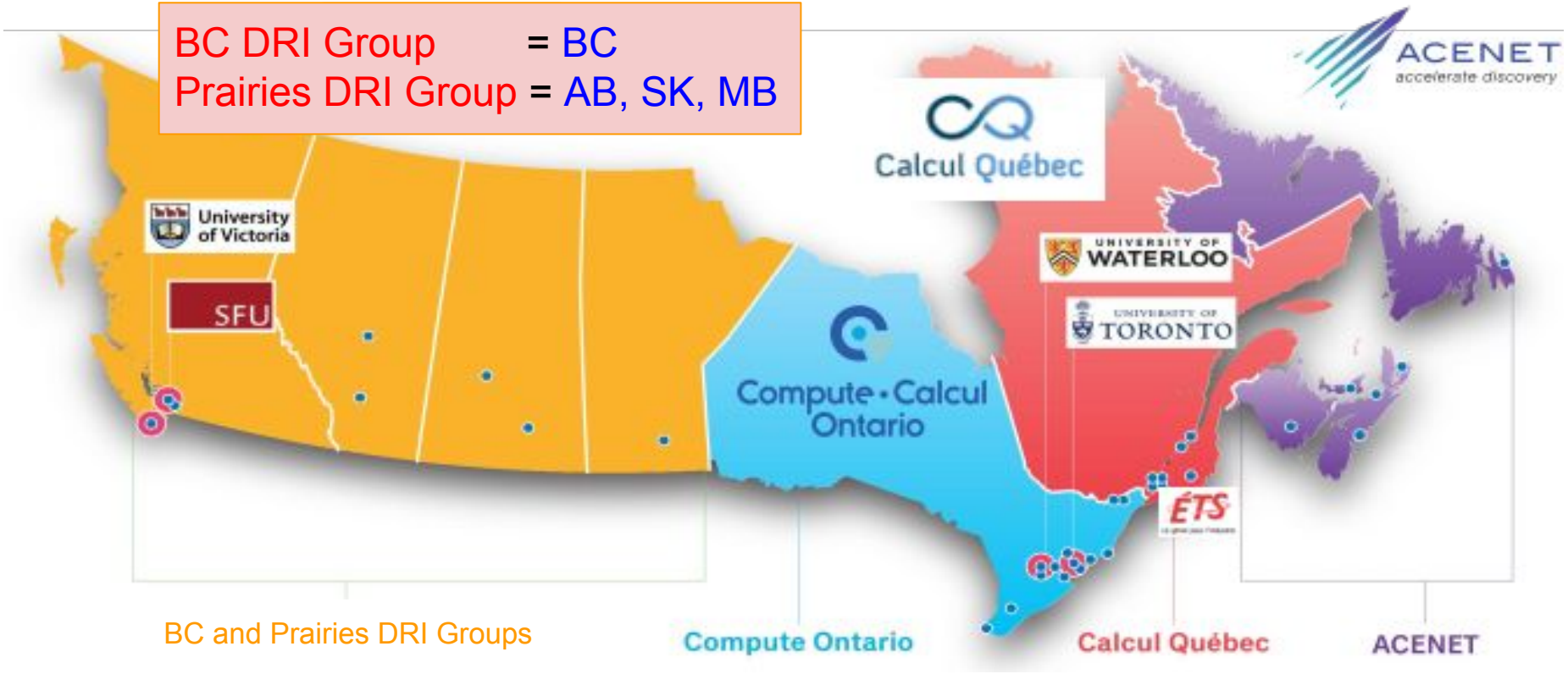
Outline

- Available resources for UofM researchers/collaborators:
  - The Alliance: cedar, graham, beluga, narval, niagara, cloud.
  - Grex: a local HPC clusters at **UofM.**

- Quick Start Guide for using HPC resources:
  - Get an account (+active role): CCDB
  - Linux shell (Terminal, command line, edit files, …)
  - Connect to a cluster: ssh, PuTTY, MobaXterm, X2Go, OOD
  - Transfer files: scp, rsync, sftp, WinSCP, FileZilla, …
  - Install programs and/or use existing modules (Lmod)
  - Submit and monitor jobs: sbatch, salloc, squeue, seff … etc.

The Alliance and its partners

BC DRI Group = BC
Prairies DRI Group = AB, SK, MB



University of Manitoba

The Alliance clusters

Cluster	Cores	GPUs	Storage	Notes
Cedar	94,528	1352	29 PB	NVidia P100; V100 Volta GPUs
Graham	41,548	520	19 PB	NVidia P100; V100; T4 GPUs
Beluga	28,000	688	27 PB	NVidia V100 GPUs
Narval	73,088	636	24.5 PB	NVidia A100 GPUs [40 GB memory]
Niagara; Mist	80,640	216	16 PB	Large parallel jobs; [4 NVIDIA V100-32GB]
Arbutus	16,008	108	17.3 PB	Physical cores: generally hyper-threaded.
GP cloud	*	*	*	Available on all General Purpose clusters.

https://docs.alliancecan.ca/wiki/Technical_documentation

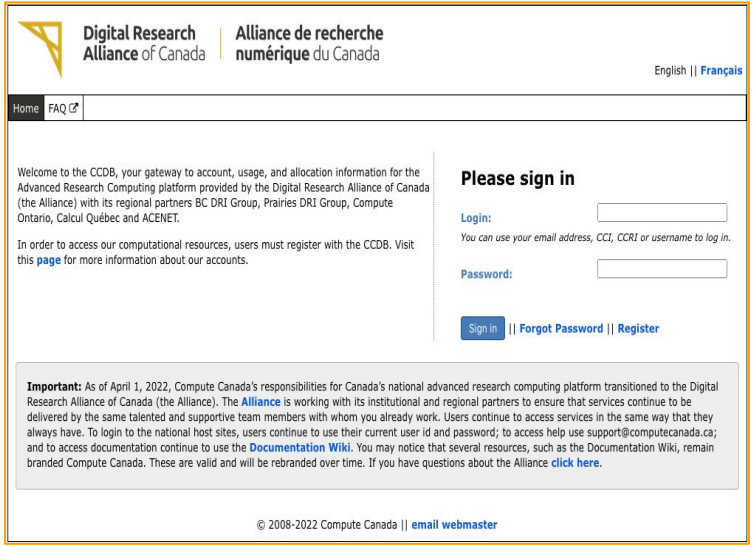
Resources on Grex: partitions

Partition	Nodes [CPUs/GPUs]	Cores	Total	Memory	Wall Time
compute[1]	312	12	3456	46 GB	21 days
largemem	12	40	480	376 GB	14 days
skylake	42	52	2184	96 GB	21 days
gpu	2 [4 V100 - 32 GB]	32	64	187 GB	3 days
stamps; -b	3 [4 V100 - 16 GB]	32	96	187 GB	21 days / 7 days
livi; -b	[16 V100 - 32 GB]	48	48	1.5 TB	21 days / 7 days
agro; -b	2 AMD [A30]	24	48	250 GB	21 days / 7 days
test	-	18	18	500 GB	12 hours

[1] to be decommissioned in the near future.

https://um-grex.github.io/grex-docs/

Access to Alliance clusters / Grex



The Alliance: Rapid Access Service

⇒ 10 TB of storage/cluster.

Send an email to: support@tech.alliancecan.ca

⇒ RAC for storage > 10 TB.

Step 1: Principal Investigator (PI) or sponsor

Faculty member registers in the Alliance Database (CCDB): https://ccdb.alliancecan.ca/security/login

Step 2: sponsored users

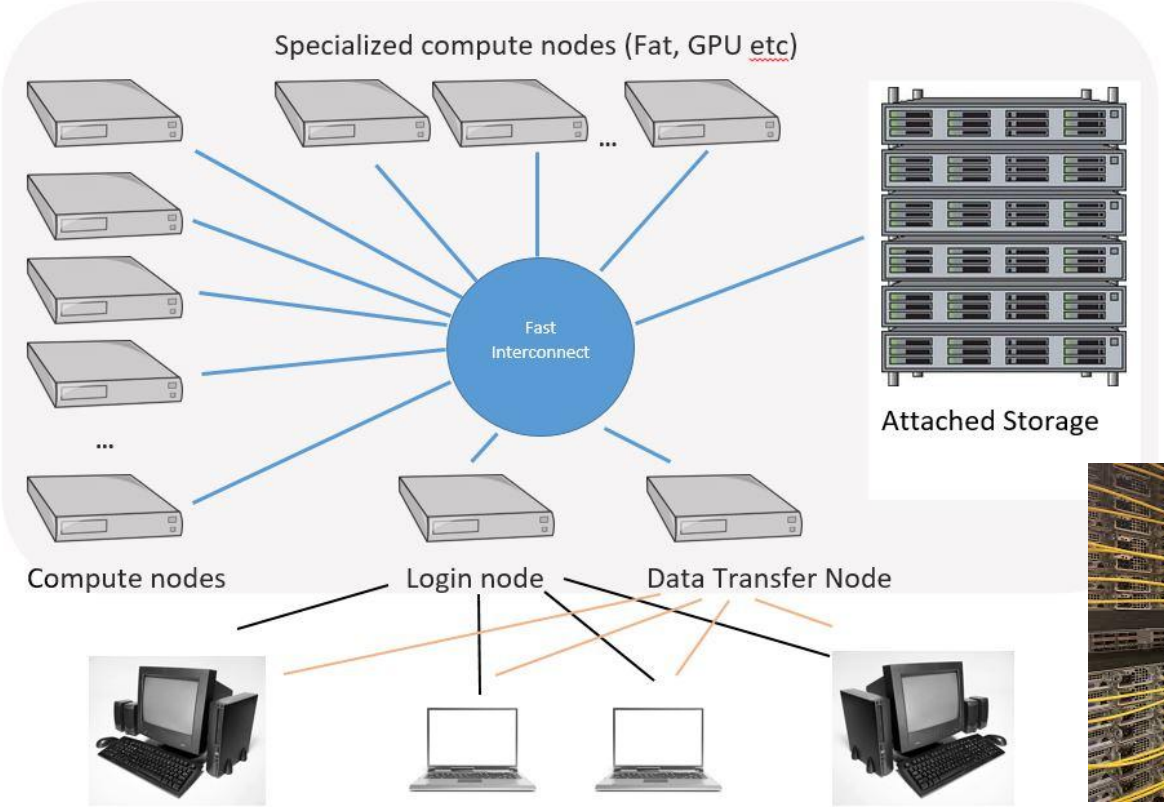
Once PI's account is approved, sponsored users can register as group members (CCRI: abc-123-01).

CCDB account: gives access to new systems / Grex

- Access to resources is free for eligible researchers.
- Every group gets a "default" share; 1 TB of storage.
- Resource Allocation Competitions: about 80 %

  Held each year, valid for 1 year [April till end of March]
- Default Allocations: 20 % are used for default share.

What is an HPC cluster?



Workflow on HPC clusters

Connect to a cluster

Linux/Mac:

⇒ ssh client

⇒ X2Go

Windows:

⇒ Putty

⇒ MobaXterm

Transfer files

Linux, Mac:

⇒ scp, sftp, rsync

Windows:

⇒ WinScp

⇒ MobaXterm

⇒ FileZilla, PuTTy

HPC work

★ Connect

★ Transfer files

★ Compile codes

★ Test jobs

★ Run jobs

★ Analyze data

★ Visualisation

OpenOnDemand: remote web access to supercomputers

https://um-grex.github.io/grex-docs/

Linux: carpentry courses

The Unix Shell

The Unix shell has been around longer than most of its users have been alive. It has survived so long because it's a power tool that allows people to do complex things with just a few keystrokes. More importantly, it helps them combine existing programs in new ways and automate repetitive tasks so they aren't typing the same things over and over again. Use of the shell is fundamental to using a wide range of other powerful tools and computing resources (including "high-performance computing" supercomputers). These lessons will start you on a path towards using these resources effectively.

Prerequisites

This lesson guides you through the basics of file systems and the shell. If you have stored files on a computer at all and recognize the word "file" and either "directory" or "folder" (two common words for the same thing), you're ready for this lesson.

If you're already comfortable manipulating files and directories, searching for files with `grep` and `find`, and writing simple loops and scripts, you probably want to explore the next lesson: shell-extras.

Schedule

	Setup	Download files required for the lesson
00:00	1. Introducing the Shell	What is a command shell and why would I use one?
00:05	2. Navigating Files and Directories	How can I move around on my computer? How can I see what files and directories I have? How can I specify the location of a file or directory on my computer?
00:45	3. Working With Files and Directories	How can I create, copy, and delete files and directories? How can I edit files?
01:35	4. Pipes and Filters	How can I combine existing commands to do new things?
02:10	5. Loops	How can I perform the same actions on many different files?
03:00	6. Shell Scripts	How can I save and re-use commands?
03:45	7. Finding Things	How can I find files? How can I find things in files?
04:30	Finish	

The actual schedule may vary slightly depending on the topics and exercises chosen by the instructor.

Edit on GitHub / Contributing / Source / Cite / Contact

Using The Carpentries style version 9.5.3.

https://swcarpentry.github.io/shell-novice/

Carpentry courses for beginners:

- Introducing the shell
- Navigating/working with files & directories
- Pipes and filters
- Loops
- Shell scripts
- Finding files/programs
- Automate tasks

https://training.westdri.ca/

Linux: basic commands

https://www.digitalocean.com/community/tutorials/linux-commands

Most used commands

- cd; mkdir; mv; rm; ls
- pwd;
- head, tail; less; more
- top; ps; htop
- gzip; tar; bzip2; gunzip
- zip; unzip
- wget; curl
- ssh; scp; sftp
- chmod; chgrp; …

man <command>

<command> - - help; -h

Connect, transfer files, …

- ssh => Secure Shell [connect to a remote machine].
- scp => Secure Copy [copy file to/from a remote host].
- sftp => Secure File Transfer Protocol.
- PuTTY => SSH and Telnet for Windows.
- FileZilla => Utility for transferring files by FTP.
- WinSCP => SFTP/FTP client for Microsoft Windows.
- MobaXterm => Toolbox for remote computing machine.
- X2Go => Remote desktop software for Linux
- OOD => Interface to remote computing resources

How to connect to a cluster?

Syntaxe: ~$ **ssh** [+options] **<username>@<hostname>**

options = -X; -Y {*X11 forwarding*}, …

➔ **Windows:** install PuTTy, MobaXterm, …

➔ **Mac:** install XQuartz {*X11 forwarding*}

Connect from a terminal:

Grex: ~$ ssh -XY <username>@grex.hpc.umanitoba.ca

Grex: ~$ ssh -XY <username>@yak.hpc.umanitoba.ca

Cedar: ~$ ssh -XY <username>@cedar.computecanada.ca

Graham: ~$ ssh -XY <username>@graham.computecanada.ca

Beluga: ~$ ssh -XY <username>@beluga.computecanada.ca

Narval: ~$ ssh -XY <username>@narval.computecanada.ca

https://docs.alliancecan.ca/wiki/SSH_Keys

Very Important

Don't share your password with anyone.

Don't send your password by email.

In case you forgot your password, it is possible to **reset it** from **CCDB**.

★ **password**

★ **ssh keys**

Connect from Windows machine

- **Install ssh client:**
  - Putty: http://www.putty.org/
  - MobaXterm: https://mobaxterm.mobatek.net/
- **How to connect?**
  - ✓ **Login:** your user name
  - ✓ **Host:** grex.hpc.umanitoba.ca
  - ✓ **Password:** your password
  - ✓ **Port:** 22
- **Use CygWin:** same environment as Linux

X2Go: Linux/Mac/Windows

Why X2Go: Access to GUI

How to use X2Go?

- Ask first if X2Go is installed on the remote machine.
- If yes, install X2Go client on your laptop or Desktop.
- Linux, Windows, Mac (XQuartz)
- Launch X2Go; Create a session and connect.

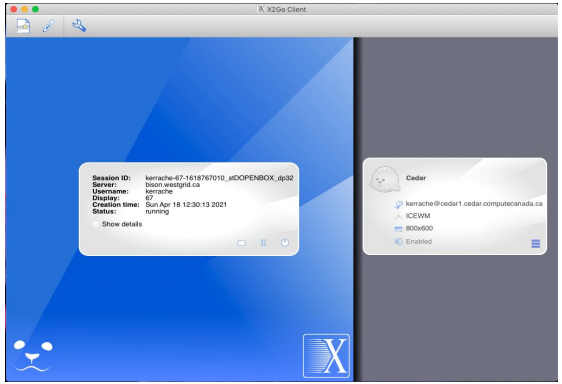
Login: your user name

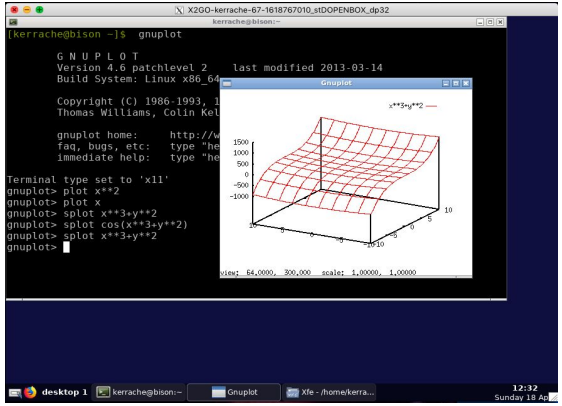
Host: bison.hpc.umanitoba.ca

{or tatanka.hpc.umanitoba.ca}

Port: 22

Session: ICEWM

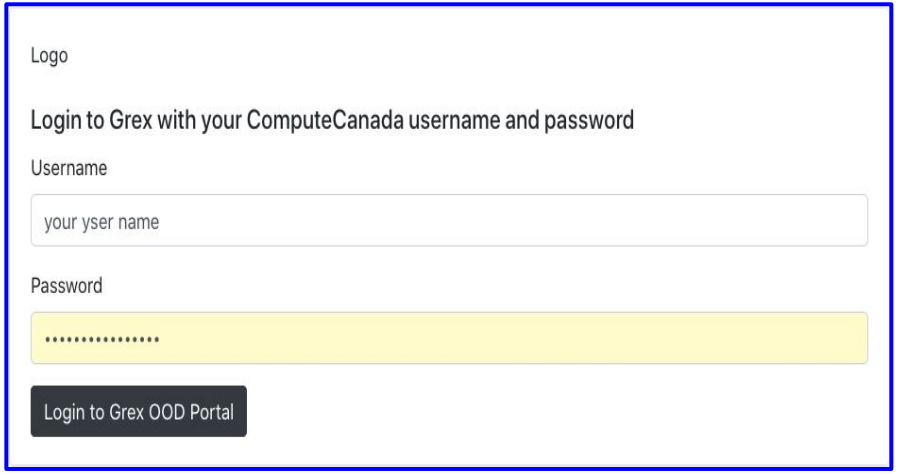


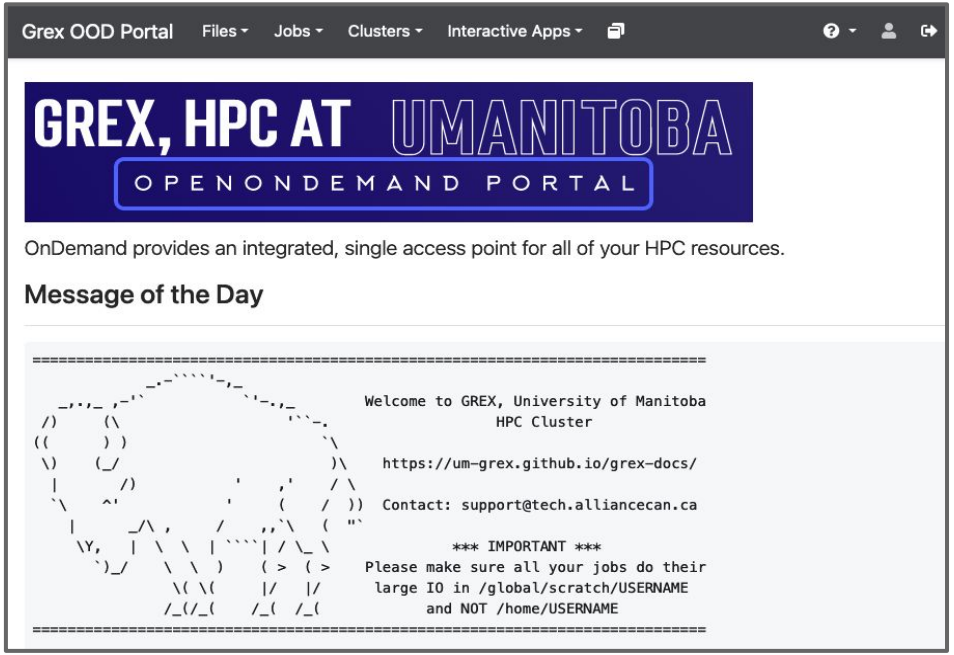


OOD: OpenOnDemand Web Portal

Connect to OOD using: UManitoba VPN:

- ★ Make sure Pulse Secure VPN is connected
- ★ Point your Web browser to https://aurochs.hpc.umanitoba.ca
- ★ Use your Alliance (Compute Canada) username/password to log in to Grex OOD.

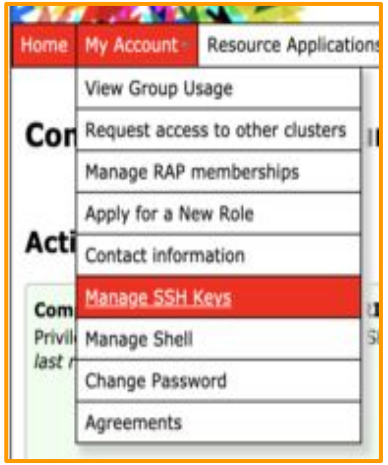




- ★ Run jobs, View jobs, files, … etc.
- ★ Run MATLAB, Gaussview, Desktop, Jupyter, …

Improve security: SSH keys

- Generate ssh keys: https://docs.alliancecan.ca/wiki/SSH_Keys#Generating_an_SSH_Key
  - Private key:
    - keep it in your computer: ~/.ssh/
    - do not share it or copy it to any cluster.
  - Public key:
    - Copy the key to remote machine
    - ssh-copy-id -i mykey someuser@niagara.computecanada.ca



- Copy the public key to:
  - Remote machine [cluster]
  - CCDB
- Mandatory to connect to niagara

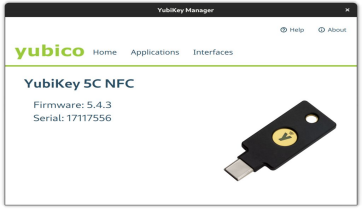
  ssh -i <path to your key> someuser@niagara.computecanada.ca
- Enabled on Grex

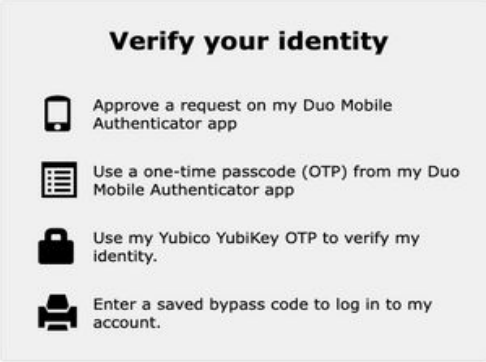
https://docs.alliancecan.ca/wiki/SSH_Keys

Improve security: MFA

- ★ **Multifactor authentication:**
  - Mandatory for all our staff
  - Coming soon for users
- ★ **Grex**
  - ssh keys in CCDB
  - VPN for OpenOnDemand
  - MFA for Grex

```
[name@server ~]$ ssh cluster.computecanada.ca
Duo two-factor login for name
Enter a passcode or select one of the following options:
1. Duo Push to My phone (iOS)
Passcode or option (1-1):abcdefghijklmnopqrstuvwxyz
Success. Logging you in...
```





https://docs.alliancecan.ca/wiki/Multifactor_authentication

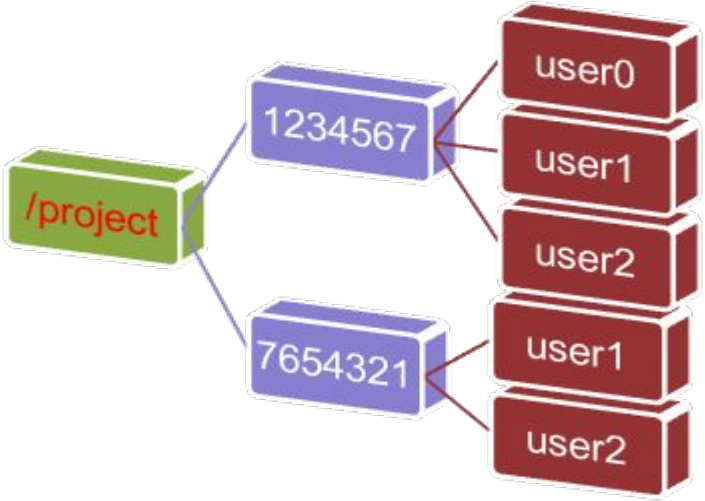
Storage: file systems and quota

the Alliance [Compute Canada]:

/home/$USER: **50** GB, daily backup

/scratch/$USER: **20** TB, no backup, purged

Project: projects/def-professor/$USER



Grex:

/home/$USER:

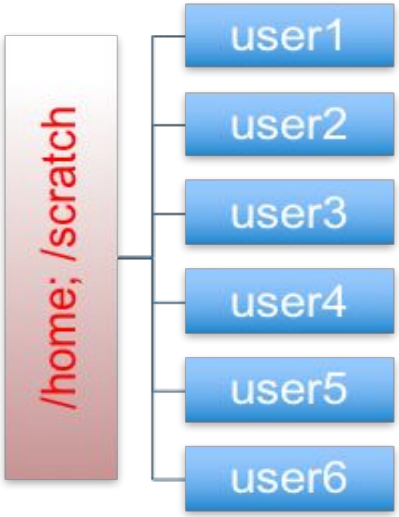
100 GB per user

/global/scratch/$USER:

4 TB, no backup, no purge.

/project

no backup, no purge.



1 TB per group; extension up to 10 TB

Backup; Allocatable via RAC (>10 TB)

Quota: diskusage_report

```
[someuser@cedar1: ~]$ diskusage_report
                                Description          Space          # of files
                  /home (user someuser)          50G/50G          6520/500k
                /scratch (user someuser)         12T/20T          8517/1000k
              /project (group someuser)          0/2048k          0/1025
           /project (group def-someprof)         1200G/10T        500k/500k
           /project (group rrg-someprof)         5838G/40T        250k/2M
```

Over quota

Space under home directory

Inode under project def-somep

```
[someuser@tatanka ~]$  diskusage_report
                          Description (FS)      Space (U/Q)      # of files (U/Q)
                      /home (someuser)          226M/104G        2381/500k
                /global/scratch (someuser)      519G/4294G       27k/1000k
                  /project (def-someprof)       3201G/5242G      17k/2000k
```

- - home
- - scratch
- - project

File transfer: scp, sftp, rsync, …

Terminal: Linux; Mac; CygWin; MobaXterm, PuTTy.

Check if scp; sftp; rsync are supported.

Syntax for scp: scp [+options] [Target] [Destination]

Syntax for rsync: rsync [+options] [Target] [Destination]

Options: for details use man scp or man rsync from your terminal.

Target: file(s) or directory(ies) to copy (exact path).

Destination: where to copy the files (exact path) [hostname:<full path>]

Path on remote machine: examples of a path on Grex.

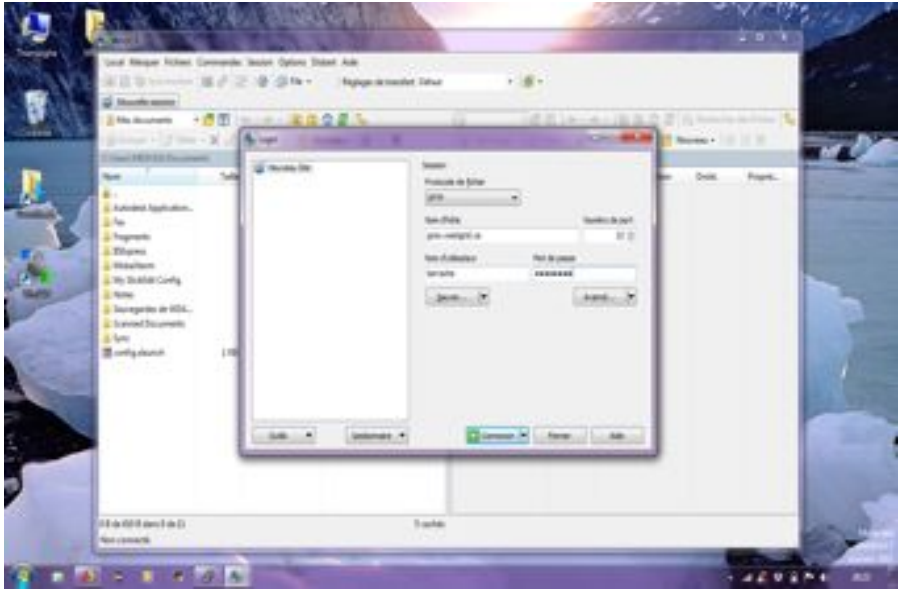
username@grex.hpc.umanitoba.ca:/home/username/{Your_Dir}; ~/{Your_Dir}

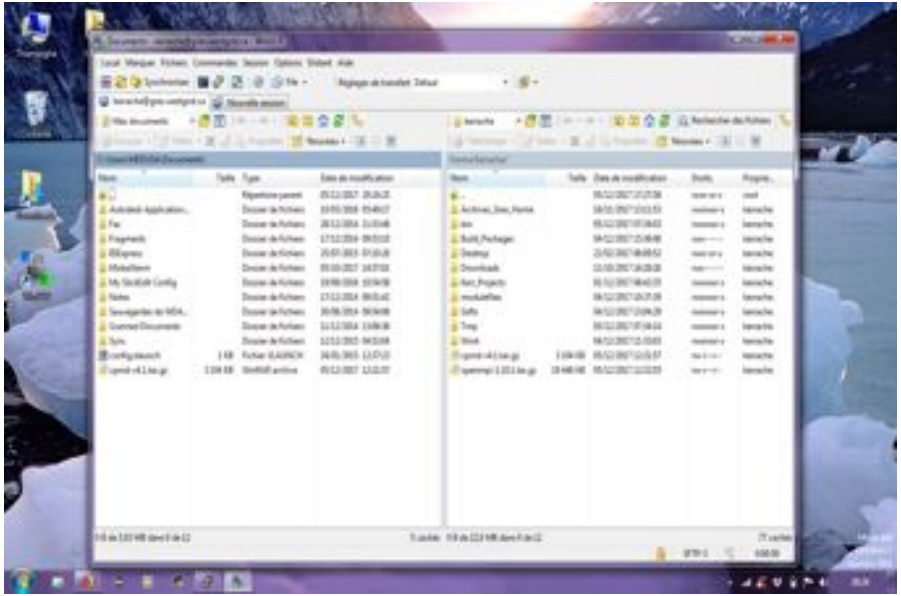
username@grex.hpc.umanitoba.ca:/global/scratch/username/{Your_Dir}

[~@Mac]: scp -r TEST username@grex.hpc.umanitoba.ca:/global/scratch/username/Work

File transfer: FileZilla, WinSCP

- Install WinScp or FileZilla.
- Launch the program.
- Connect with your credentials.

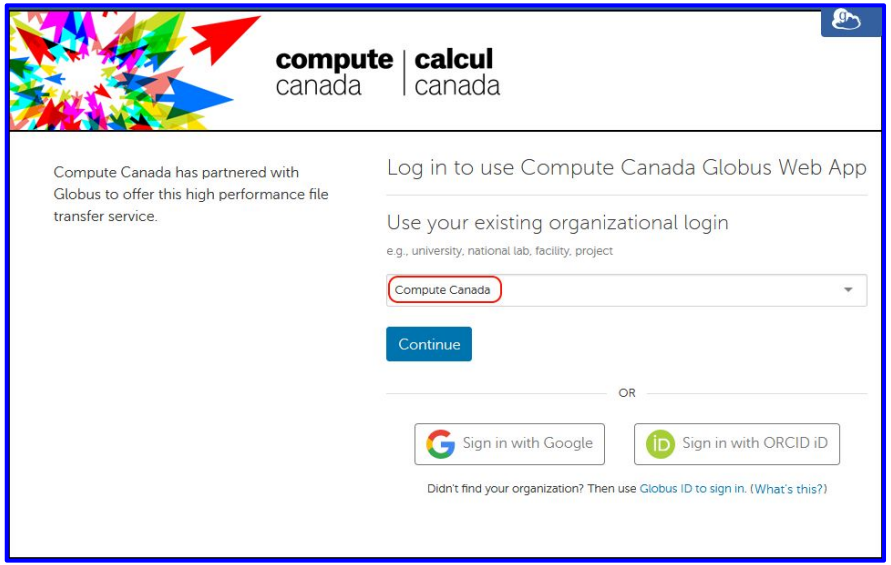




- Navigate on your local machine.
- Navigate on remote machine.
- Copy your files (works on both ways).

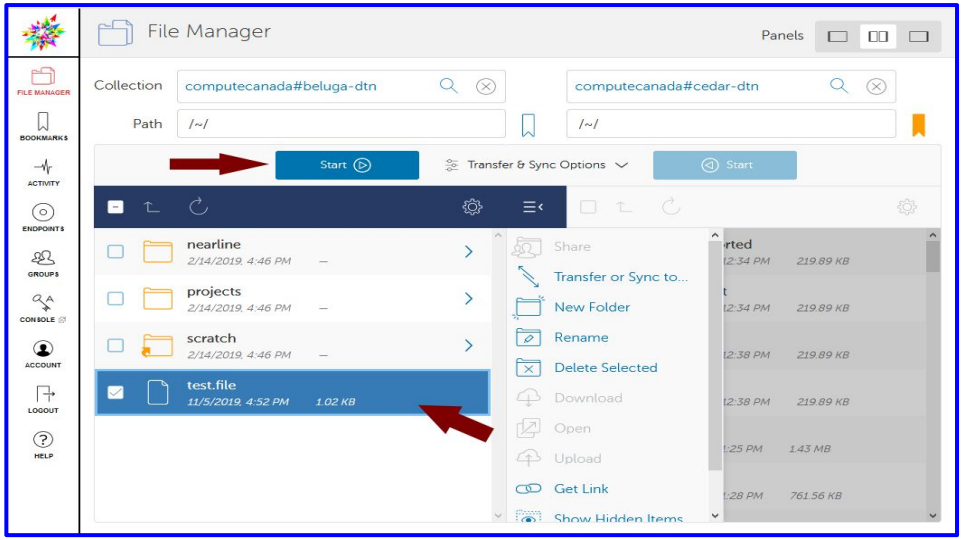
File transfer: Globus

- Launch Globus web interface.
- Connect with your credentials.



https://docs.alliancecan.ca/wiki/Globus/en

- Search for the globus endpoints
- Navigate to your directories
- Initiate the transfer / Log out.



Software and modules

- ★ Software on HPC clusters
- ★ Software distribution
- ★ Available software on HPC clusters
- ★ Find a software: modules

Software distribution

- ★ Operating system package managers / repos
  - **Ubuntu:** $ ***sudo apt-get install bowtie2***
  - **CentOS:** $ ***sudo yum install bowtie2*** # might need EPEL repo
  - **On HPC**, users do not have **sudo**! {**It is not required; no need to ask for it**}
- ★ Local install from sources or binaries, usually to $HOME
  - wget https://github.com/BenLangmead/bowtie2/releases/download/v2.3.4.3/bowtie2-2.3.4.3-linux-x86_64.zip
  - unzip bowtie2-2.3.4.3-linux-x86_64.zip
  - bowtie2-2.3.4.3-linux-x86_64/bowtie2 -?
  - Or build from sources, specifying the PREFIX, **CMAKE_INSTALL_PREFIX** or **--prefix** to $HOME/bowtie2/
  - and add the locations to PATH, LD_LIBRARY_PATH etc.
- ★ Using a centralized HPC stack [modules]
  - installed and maintained by analysts: compilers, libraries, domain specific software, … etc.
  - ask for installing a given program or updating modules if needed.

Software on HPC clusters

★ Home made: programs, scripts and tools, … etc.

Up to a user, … ?

★ Free Software: GNU Public License.

Open Source, Binaries, Libraries, Compilers, Tools, …

★ Commercial Software: restricted [VASP, STATA, …]

➔ Contact support with some details about the license, …

➔ We install the program & protect it with a POSIX group.

Analysts / Sys admins

Available software on HPC clusters

- ★ Number-crunching software environments:
  - Compilers (GCC, Intel), BLAS/LAPACK/PETSc, BLIS, MPI, OpenMP, … etc.
- ★ Dynamic languages and libraries: R, Python, Perl, Julia, ...
- ★ Domain-specific applications and packages:
  - Engineering, Chemistry, Physics, Machine-Learning, ...
  - Biomolecular, Genomics etc.
- ★ CC Centralized software stack, distributed via CVMFS:

  https://docs.alliancecan.ca/wiki/Available_software
- ★ **Grex:**
  - **GrexEnv:** modules installed locally on Grex [more than 500 modules].
  - **CCEnv:** access to public repository of the Alliance.

How to find a software?

Why modules?

- Control different versions of the same program.
- Avoid conflicts between different versions and libraries.
- Set the right path to each program or library.

```
[someuser@bison ]$  module avail
–
CCEnv (S)        GrexEnv (S,L)

Where:
   S:  Module is Sticky, requires --force to
unload or purge
   L:  Module is loaded
```

Useful commands for working with modules:

- module **list**; module **avail**
- module **spider** <soft>/<version>
- module **load** soft/version; module **unload {rm}** <soft>/<version>
- module **show** soft/version; module **help** <soft>/<version>
- module **purge**; module --force **purge**
- module **use** ~/modulefiles; module **unuse** ~/modulefiles

Find and load QE

```
[someuser@bison ]$  module spider espresso
----------------------------------------------------------------------------
  espresso:
----------------------------------------------------------------------------
     Versions:
        espresso/5.4.0
        espresso/6.3
        espresso/6.4.1
        espresso/6.5
        espresso/7.0
----------------------------------------------------------------------------
  For detailed information about a specific "espresso" package (including how to load the modules) use the module's full name. Note that names that have a trailing (E) are extensions provided by other modules.
  For example:
     $ module spider espresso/7.0
----------------------------------------------------------------------------
```

```
[someuser@bison ]$  module spider espresso/7.0
```

```
[someuser@bison ]$  module load intel/2020.4  ompi/4.1.0 espresso/7.0
    Loading module: hdf5-1.12.1
    Loading module: libxc/5.2.2
    Loading module: ESPRESSO/7.0
```

Find and load ORCA

ORCA:

https://docs.alliancecan.ca/wiki/ORCA

- **restricted software [requires a registration]**

[someuser@bison]$ module spider orca

```
[someuser@bison ]$  module spider orca/5.0.4
```

---

orca:

Versions:
orca/4.2.1
orca/5.0.1
orca/5.0.2
orca/5.0.4

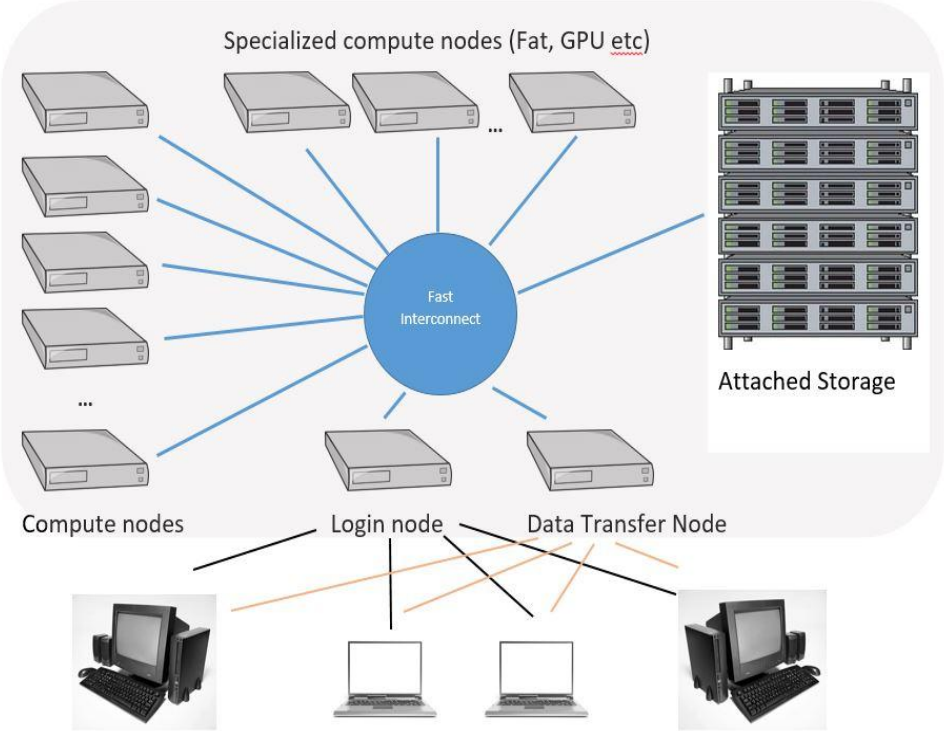
```
[someuser@bison ]$  module load gcc/4.8  ompi/4.1.1 orca/5.0.4
    Loading module: gcc/4.8
    Loading module: ORCA/5.0.4
```

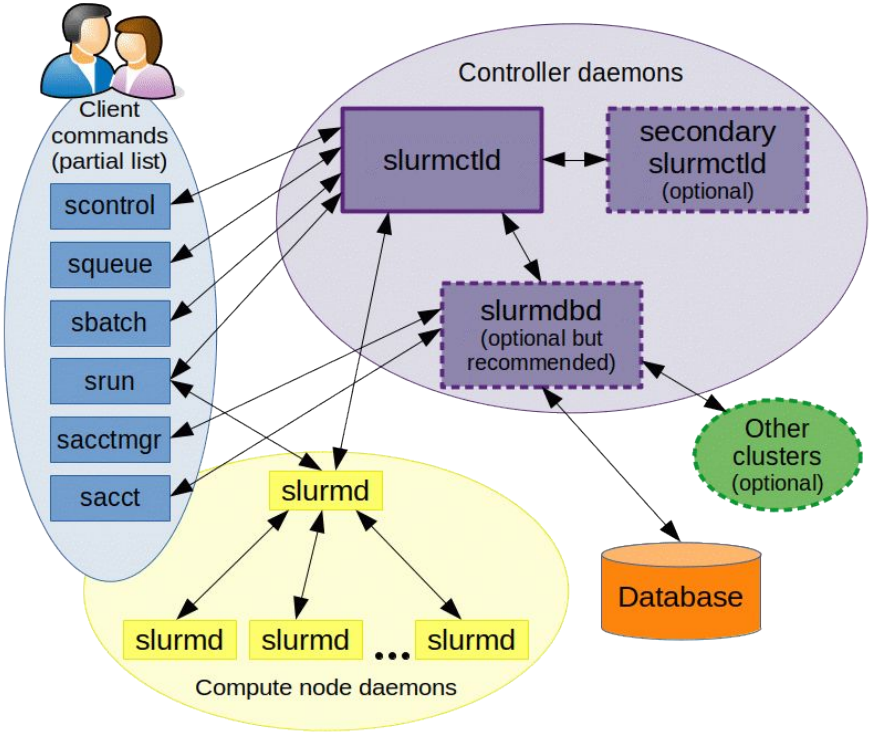
For detailed information about a specific "orca" package (including how to load the modules) use the module's full name. Note that names that have a trailing (E) are extensions provided by other modules.

For example:

$ module spider orca/5.0.4

Running jobs on HPC clusters





University of Manitoba

Running jobs on a cluster

- ★ Job requirements: CPUs, Memory, Time, … etc.
- ★ SLURM template: structure of a job script
- ★ Interactive jobs via salloc
- ★ Example of SLURM script: Gaussian
- ★ SLURM directives
- ★ SLURM environment variables
- ★ Examples: Serial, OpenMP, MPI, GPU
- ★ Monitor and control your jobs: seff, scancel, sacct, …
- ★ *Bundle multiple jobs: job arrays and GLOST*
- ★ *Estimating resources: CPUs, MEM, TIME*
- ★ *How to pick a partition on Grex?*

slurm
workload manager

Scheduler: SLURM

SLURM: Simple Linux Utility for Resource Management

- free and open-source job scheduler for Linux and Unix-like kernels
- used by many of the world's supercomputers and computer clusters.

https://slurm.schedmd.com/overview.html

sacct - sacctmgr - salloc - sattach - sbatch - sbcast - scancel - scontrol - sdiag - seff - sh5util - sinfo - smail - smap - sprio - squeue - sreport - srun - sshare - sstat - strigger - sview

Running jobs on a cluster

- When you connect you get interactive session on a login node:
  - Limited resources: to be used with care for basic operations
    - editing files, compiling codes, download or transfer data, submit and monitor jobs, run short tests {no memory intensive tests}
  - Performance can suffer greatly from over-subscription
- For interactive work, submit interactive jobs: salloc [+options]
  - SLURM uses salloc for interactive jobs [compute nodes]
  - The jobs will run on dedicated compute nodes [CPUs, GPUs]
- Submitting batch jobs for production work is mandatory: sbatch
  - Wrap commands and resource requests in a “job script”: myscript.sh
  - SLURM uses sbatch; submit a job using: sbatch myscript.sh
    sbatch [+options] myscript.sh

Accounting groups

- Submit Interactive job:

```
[~cedar5 scratch]$  salloc --ntasks=1 --mem=4000M
salloc: error: ----------------------------------------
salloc: error: You are associated with multiple _cpu allocations...
salloc: error: Please specify one of the following accounts to submit this job:
salloc: error:   RAS default accounts: def-prof1, def-prof2
salloc: error:          RAC accounts:
salloc: error: Compute-Burst accounts:
salloc: error:        Other accounts: cc-debug,
salloc: error: Use the parameter --account=desired_account when submitting your job
salloc: error: ----------------------------------------
salloc: error: Job submit/allocate failed: Unspecified error
```

- Accounting groups: sshare -U --user <username>
  - if one accounting group, SLURM will take it by default.
  - If more than one, it should be specified via: --account={your accounting group}

Job requirements

What do you need to know before submitting a job?

- Is the program available? If not, install it or ask support for help.
- What type of program are you going to run?
  - Serial, Threaded [OpenMP], MPI based, GPU, …
- Prepare your input files: locally or transfer from your computer.
- Test your program:
  - Interactive job via salloc: access to a compute node
  - On login node if the test is not memory nor CPU intensive.
- Prepare a script "myscript.sh" with the all requirements:
  - **Memory**, Number of cores, Nodes, Wall time, modules, **partition**, **accounting group**, command line to run the code.
- Submit the job and monitor it: sbatch, squeue, sacct, seff … etc

Interactive jobs via salloc

```
[someuser@bison ]$  salloc --cpus-per-task=4 --mem-per-cpu=1000M --time=1:00:00
salloc: using account: def-someprof
salloc: No partition specified? It is recommended to set one! Will guess
salloc: Pending job allocation 5081294
salloc: job 5081294 queued and waiting for resources
salloc: job 5081294 has been allocated resources
salloc: Granted job allocation 5081294
salloc: Waiting for resource configuration
salloc: Nodes n063 are ready for job
        Load modules + run tests
[someuser@n063 ]$  exit
exit
salloc: Relinquishing job allocation 5081294
```

Equivalent SLURM script:

```
#!/bin/bash
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=1000M
#SBATCH --time=1:00:00
#SBATCH --account=def-someprof
```

Interactive jobs via salloc

```
[someuser@bison ]$  salloc --ntasks=1 --cpus-per-task=4 --mem-per-cpu=1000M
--account=def-someprof --partition=skylake --x11

salloc: using account: def-someprof
salloc: partition selected:skylake
salloc: Granted job allocation 5081297
salloc: Waiting for resource configuration
salloc: Nodes n376 are ready for job
        Load modules + run tests
[someuser@n376 ]$  exit
exit
salloc: Relinquishing job allocation 5081297
```

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=1000M
#SBATCH --mem=4000M
#SBATCH --time=3:00:00
#SBATCH --account=def-someprof
#SBATCH --partition=skylake
```

SLURM: simple template

```
#!/bin/bash

#SBATCH --account=def-somegroup

{Add the resources and some options}

echo "Current working directory is `pwd`"
echo "Starting run at: `date`"

{Load appropriate modules if needed.}
{Command line to run your program.}

echo "Program finished with exit code $? at: `date`"
```

Script: test-job.sh

Parameters to adjust for each type of job to submit: serial, MPI, GPU

Default parameters:

- CPUs: 1
- Time: 0-3:00
- Memory: 256mb

SLURM script: Gaussian

```
#!/bin/bash
#SBATCH --account=def-someprof
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=2500M
#SBATCH --time=7-00:00:00
#SBATCH --partition=compute

# Load appropriate modules:
module load gaussian
echo "Starting run at: `date`"
g16 < my-input.com > my-output.out
echo "Program finished with exit code $? at: `date`"
```

SLURM directives:

- Default: 1 core, 256mb, 3 hours
- **account**, number of tasks, memory per core, wall time, **partition**, …
- Other: Email notification, … etc.

Submit and monitor the job:

- sbatch [some options] myscript.sh
- squeue -u $USER; sq

Partition:

- partition-list; sinfo --format="%20P"
- sinfo -s; sinfo -p <partition name>

SLURM: most used directives

#SBATCH --account=def-someprof	Use the accounting group def-someprof for jobs.
#SBATCH --ntasks=8	Request 8 tasks for MPI job; 1 for serial or OpenMP
#SBATCH --cpus-per-task=4	Number of threads (OpenMP); Threaded application
#SBATCH --ntasks-per-node=4	Request 4 tasks per-node for MPI job
#SBATCH --nodes=2	Request 2 nodes
#SBATCH --mem=1500M	Memory of 1500M for the job
#SBATCH --mem-per-cpu=2000M	Memory of 2000M per CPU
#SBATCH --partition=compute	Partition name: compute, skylake, largemem, gpu, test
#SBATCH --time=3-00:00:00	Wall time in the format: DD-HH:MM:SS

SLURM: environment variables

SLURM_JOB_NAME	User specified job name
SLURM_JOB_ID	Unique slurm job id
SLURM_NNODES	Number of nodes allocated to the job
SLURM_NTASKS	Number of tasks allocated to the job
SLURM_ARRAY_TASK_ID	Array index for this job
SLURM_ARRAY_TASK_MAX	Total number of array indexes for this job: --array=0-999%10
SLURM_CPUS_PER_TASK	Number of threads {OpenMP: OMP_NUM_THREADS}
SLURM_JOB_NODELIST	List of nodes on which resources are allocated to a Job
SLURM_JOB_ACCOUNT	Account under which this job is running.
SLURM_JOB_PARTITION	List of Partition(s) that the job is in.

SLURM script: serial jobs

```
#!/bin/bash
#SBATCH --account=def-someprof
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=2500M
#SBATCH --time=1-00:00:00
#SBATCH --partition=compute

# Load appropriate modules:
module load <software>/<version>
echo "Starting run at: `date`"
program.x [+options and arguments if any]
echo "Program finished with exit code $? at: `date`"
```

SLURM directives:

- Default: 1 core, 256mb, 3 hours
- **account**, tasks = 1, memory per core, wall time, **partition**, …
- Other: E-mail-notification, … etc.

Submit and monitor the job:

- sbatch myscript.sh
- squeue -u $USER; sq; sacct -j JOB_ID

More information:

- partition-list; sinfo --format="%20P"
- Sinfo -s; sinfo -p compute,skylake
- squeue -p compute,skylake -t R {PD}

SLURM script: OpenMP jobs

```
#!/bin/bash
#SBATCH --account=def-someprof
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=2000M
#SBATCH --time=1-00:00:00
#SBATCH --partition=skylake
# Load appropriate modules:
module load <software>/<version>
export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}
echo "Starting run at: `date`"
program.x [+options and arguments if any]
echo "Program finished with exit code $? at: `date`"
```

```
#!/bin/bash
#SBATCH --account=def-someprof
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=2000M
#SBATCH --time=1-00:00:00
#SBATCH --partition=skylake
```

```
#SBATCH --cpus-per-task=N
#SBATCH --mem=<MEM>
```

Partitions:

- compute: N up to 12
- skylake: N up to 52
- largemem: N up to 40

SLURM script: MPI jobs

```
#!/bin/bash
#SBATCH --account=def-someprof
#SBATCH --ntasks=96
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=1200M
#SBATCH --time=2-00:00:00
#SBATCH --partition=skylake

# Load appropriate modules:
module load intel/2019.5  ompi/3.1.4 lammps/29Sep21
echo "Starting run at: `date`"
srun lmp_grex < in.lammps
echo "Program finished with exit code $? at: `date`"
```

```
#SBATCH --nodes=8
#SBATCH --ntasks-per-node=12
#SBATCH --mem=0
#SBATCH --partition=compute
```

```
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=52
#SBATCH --mem=0
#SBATCH --partition=skylake
```

```
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=40
#SBATCH --mem=0
#SBATCH --partition=largemem
```

SLURM script: OpenMP+MPI jobs

```
#!/bin/bash
#SBATCH --account=def-someprof
#SBATCH --nodes=4
#SBATCH --ntasks-per-node=6
#SBATCH --cpus-per-task=2
#SBATCH --mem-per-cpu=1200M
#SBATCH --time=3-00:00:00
#SBATCH --partition=compute

# Load appropriate modules:
module load <software>/<version>
export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}
echo "Starting run at: `date`"
srun program.x [+options and arguments if any]
echo "Program finished with exit code $? at: `date`"
```

```
#SBATCH --nodes=6
#SBATCH --ntasks-per-node=4
#SBATCH --cpus-per-task=2
#SBATCH --mem-per-cpu=1200M
#SBATCH --partition=compute
```

The total memory and CPUs per node should not exceed the available resources on the nodes.

```
#SBATCH --nodes=5
#SBATCH --ntasks-per-node=8
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=1000M
#SBATCH --partition=skylake
```

SLURM script: OpenMP+MPI jobs

```
#SBATCH --nodes=8
#SBATCH --ntasks-per-node=12
#SBATCH --cpus-per-task=1
#SBATCH --mem=0
#SBATCH --partition=compute
```

Job ID: 1234567
Cluster: grex
User/Group: someuser/someuser
State: COMPLETED (exit code 0)
Nodes: 8
Cores per node: 12
CPU Utilized: 156-11:07:22
CPU Efficiency: 99.22% of 157-16:44:48 core-walltime
Job Wall-clock time: 1-15:25:28
Memory Utilized: 218.00 GB (estimated maximum)
Memory Efficiency: 59.37% of 367.19 GB (45.90 GB/node)

The job used:

- **96 CPUs**
- **about 2400 M per core**

The job may wait longer on the queue to start:
it requires 8 nodes to be available
=> Optimize the resources

```
#SBATCH --ntasks=96
#SBATCH --mem-per-cpu=2400M
#SBATCH --partition=compute
```

```
#SBATCH --ntasks=162
#SBATCH --mem-per-cpu=1200M
#SBATCH --partition=skylake
```

SLURM script: GPU jobs

```
#!/bin/bash
#SBATCH --account=def-someprof
#SBATCH --gpu=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=6
#SBATCH --mem-per-cpu=4000M
#SBATCH --time=0-3:00:00
#SBATCH --partition=gpu
# Load appropriate modules:
module load <software>/<version>
echo "Starting run at: `date`"
program.x [+options and arguments if any]
echo "Program finished with exit code $? at: `date`"
```

SLURM directives:

- Default: 1 core, 256mb, 3 hours
- **account**, number of tasks, memory per core, wall time, **partition**, …
- Other: E-mail-notification, … etc.

Submit and monitor the job:

- sbatch [some options] myscript.sh
- squeue -u $USER

Partition:

- partition-list; sinfo --format="%20P"
- sinfo -p <partition name>

Monitor and control your jobs

```
squeue -u $USER [-t RUNNING] [-t PENDING]                    # list all current jobs.
squeue -p PartitionName [compute, skylake, largemem]   # list all jobs in a partition.
sinfo                                                # view information about Slurm partitions.
sacct -j jobID --format=JobID,MaxRSS,Elapsed    # resources used by completed job.
sacct -u $USER --format=JobID,JobName,AveCPU,MaxRSS,MaxVMSize,Elapsed
seff  -d jobID                          # produce a detailed usage/efficiency report for the job.
sprio [-j jobID1,jobID2] [-u $USER]                          # list job priority information.
sshare -U --user $USER                                       # show usage info for user.
sinfo --state=idle; -s; -p <partition>              # show idle nodes; more about partitions.
scancel [-t PENDING] [-u $USER] [jobID]                      # kill/cancel jobs.
scontrol show job -dd jobID                         #show more information about the job.
```

Information about the cluster

- **sinfo:** check the nodes (idle, drain, down), …

  sinfo --state=idle {shows idle nodes on the cluster}
  sinfo --R {shows down, drained and draining nodes and their reason}
  sinfo --Node --long {shows more detailed information}
  sinfo --p largemem {shows more detailed information}

- **scontrol:** to see reservations and more

[~@gra-login1: ~]$ scontrol show res <Outage> --oneliner
ReservationName=Outage StartTime=2022-10-25T08:50:00 EndTime=2022-10-26T10:00:00 Duration=1-01:10:00 Nodes=gra[1-1257,1262-1325,1337-1338,1342] NodeCnt=1324 CoreCnt=44396 Features=(null) PartitionName=(null) Flags=MAINT,IGNORE_JOBS,SPEC_NODES,ALL_NODES TRES=cpu=44396 Users=root Groups=(null) Accounts=(null) Licenses=(null) State=INACTIVE BurstBuffer=(null) Watts=n/a MaxStartDelay=(null)

SLURM: messages

- None: the job is running (ST=R)
- PartitionDown: one or more partitions are down (the scheduler is paused)
- Resources: the resources are not available for this job at this time
- Nodes required for job are DOWN, DRAINED or RESERVED for jobs in higher priority partitions: similar to Resources.
- Priority: the job did not start because of its low priority
- Dependency: the job did not start because it depends on another job that is not done yet.
- JobArrayTaskLimit: the user exceeded the maximum size of array jobs
  - [~@tatanka ~]$ scontrol show config | grep MaxArraySize
    MaxArraySize = 2000
- ReqNodeNotAvail, UnavailableNodes: n314: node not available

University of Manitoba

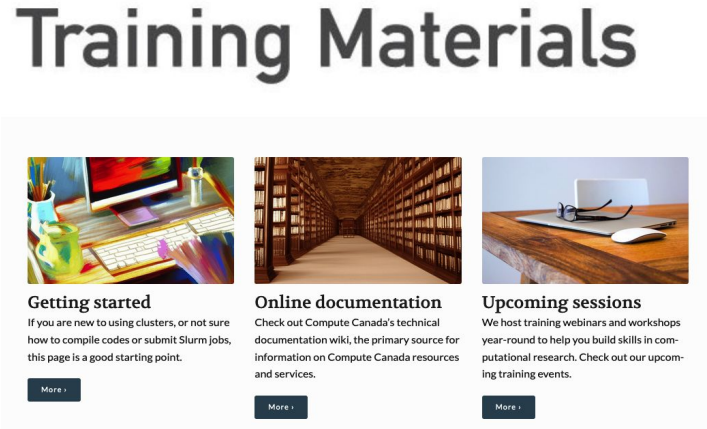
Summary

- Account and active role:
  - CCDB
- Have a look to the documentation:
  - Hardware, available tools, …
  - policies?
  - login nodes
  - storage, …
- Tools to connect and transfer files
- Access to storage: home, scratch, project
- Access to a program to use:
  - Install the program or ask for it.
  - Use the existing modules

- Test jobs:
  - Login node
  - Interactive job via salloc
- Write a job script:
  - Slurm directives
  - Modules
  - Command line to run the code
- Monitor jobs:
  - Sacct; seff, optimize jobs
- Analyze data:
  - Post processing
  - Visualization

More readings

- The Alliance [Compute Canada]: https://docs.alliancecan.ca/wiki/Main_Page
- CCDB: https://ccdb.computecanada.ca/security/login
- CC Software: https://docs.alliancecan.ca/wiki/Available_software
- Running Jobs: https://docs.alliancecan.ca/wiki/Running_jobs
- SLURM: https://slurm.schedmd.com/
- PuTTy: http://www.putty.org/
- MobaXterm: https://mobaxterm.mobatek.net/
- X2Go: https://wiki.x2go.org/doku.php
- Grex: https://um-grex.github.io/grex-docs/



→ WG training material: https://training.westdri.ca/

→ Help and support {Grex+Alliance}: support@tech.alliancecan.ca

University of Manitoba

Thank you for your attention

Any question?