University of Manitoba

compute canada | calcul canada

Introduction to HPC: Basics

Steps from getting an account to submitting jobs

UofM-Autumn-Workshop 2021

Nov 1st-2nd, 2021

Ali Kerrache

University of Manitoba

University of Manitoba

Outline: HPC step by step

- ★ Apply for an account: Compute Canada / Grex
- ★ Available resources:
  - Grex: U. of Manitoba researchers and their collaborators.
  - Compute Canada: cedar, graham, beluga, niagara, cloud.
- ★ Basic tools for using HPC clusters:
  - Linux shell (Terminal): separate presentation
  - Connect to a cluster: ssh client, PuTTY, MobXterm
  - File system / storgae: home, scratch, project, nearline
  - Transfer files: scp, sftp, WinSCP, GLOBUS ...
  - Submit and monitor jobs: sbatch, salloc, squeue, … etc.

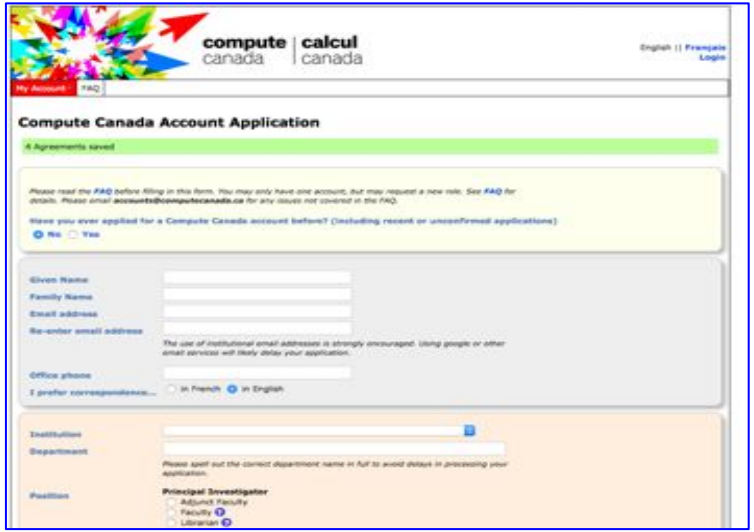
Getting an account

Step 1:

Faculty member registers in the Compute Canada Database (CCDB): http://ccdb.computecanada.ca

Step 2:

Once an account is approved, students / colleagues can register as group members: requires CCRI.



Accounts and roles:

- One account per user
- One or more roles: Grad, PhD, PostDoc, Faculty
- Renewed each year {spring}
- Not shared.

Access to Grex:

- Compute Canada account.
- UofM researchers and their collaborators.

Resources: CPU/GPU, Storage

Access to Compute Canada and Grex resources is **free of charge** for the eligible researchers.

Storage on Grex:

- /home [100 GB per user]
- /global/scratch [2+ TB per user]

Storage on Compute Canada:

- ★ /home [50 GB per user]
- ★ /scratch [20 TB per user]
- ★ /project [1 TB; up to 10 TB (RAS)]
- ★ /nearline [2 TB]

Compute Canada:

- ★ Everyone gets a “Default” share.
- ★ Every PI gets 1 TB of storage by default.
- ★ Default storage up to 10 TB {on request }.
  Send an email to: support@computecanada.ca
- ★ Resource Allocation Competitions (about 80%) are held annually, to distribute resources based on proposal’s merit.
- ★ The remaining 20% are used for default share {active default accounts}.
- ★ More resources: require RAC
  - Storage > 10 TB
  - CPU > 50 Core years

University of Manitoba

Compute Canada resources

University of Manitoba

Compute Canada clusters

System	CPUs	GPUs	Storage	Notes
Cedar	94k	1352	29 PB	NVidia P100; V100 Volta GPUs
Graham	42k	520	19 PB	NVidia P100; V100; T4 GPUs
Beluga	28k	688	27 PB	NVidia V100 GPUs
Narval	80k	632	25 PB	NVidia A100
Niagara; Mist	73k	216	16 PB	Large parallel jobs; [4 NVIDIA V100-32GB]
Arbutus	16k	108	17.3 PB	Physical cores: generally hyper-threaded.

https://docs.computecanada.ca/wiki/Compute_Canada_Documentation

Grex: a cluster for UofM users

- ★ Grex upgrade:
  - **42** compute nodes [**52** cores, **87000M** of memory]
  - **12** nodes [40 cores, *Intel CascadeLake* 6248 2.5GHz]
  - **2 login nodes:** not yet enabled for users.
- ★ GP resources: first arrived, first served
  - **320** nodes of **12** cores [**47000M** per node].
  - **2** GPUs [**32** cores, Tesla V100 **32GB**, 178 GB]
- ★ Contributed hardware [GPUs]:
  - partitions with suffix "-b": stamps-b, livi-b
  - could be used by any other user [up to 7 days].
- ★ Allocated Resources: 2664 CPU cores.
  - **12** nodes [40 cores, *Intel CascadeLake* 6248 2.5GHz, 384 GB RAM]
  - **42** nodes [52 cores, *Intel CascadeLake* 6230R 2.1GHz , 93 GB RAM]

University of Manitoba

Grex resources and partitions

Partition	Nodes [CPUs/GPUs]	Cores/node	Cores	Memory	Max Wall Time
compute*	316	12	3456	46 GB	21 days
largemem	12	40	480	380 GB	14 days
skylake	42	52	2184	87 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	1 [16 V100 - 32 GB]	48	48	1.5 TB	21 days / 7 days
*	370 nodes + 6 GPUs	*	6120	*	*

CPU/GPU resources [nodes, cores, memory, wall time, partitions],

Structure of HPC clusters

University of Manitoba

HPC: workflow and tools

Connect to a cluster

Linux:

ssh; X2Go

Mac:

ssh, X2Go

Windows:

Putty, MobaXterm, ...

Transfer files

Linux:

scp; sftp; rsync

Mac:

ssp, sftp; rsync; …

Windows:

WinScp, FileZilla, MobaXterm, …

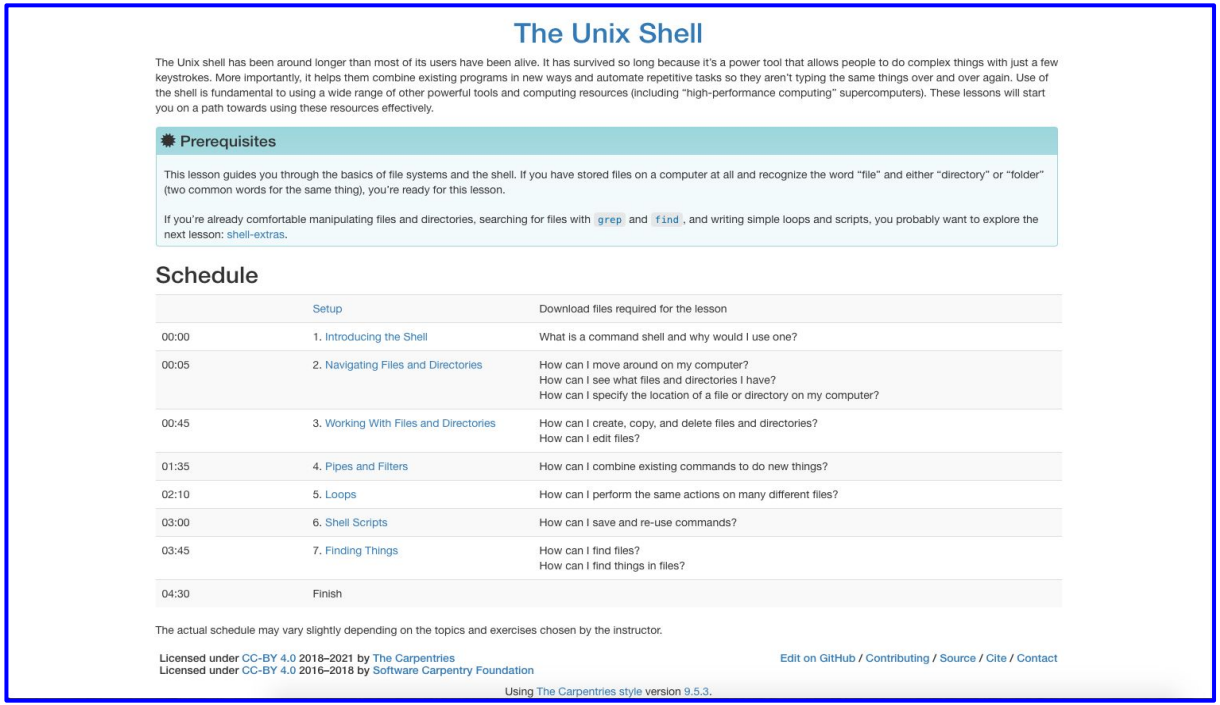
HPC

- Connect
- Transfer files
- Compile codes
- Test jobs
- Run jobs
- Analyze data
- Visualisation

https://monitor.hpc.umanitoba.ca/doc/

University of Manitoba

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 with files and directories
- Working with files and directories
- Pipes and filters
- Loops
- Shell scripts
- Finding files and programs

University of Manitoba

How to connect to a cluster?

Syntaxe:

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

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

Windows: install PuTTy, MobaXterm, …

Mac: install XQuartz for x11 forwarding

Connect from a terminal:

Grex: ~$ ssh -XY username@grex.westgrid.ca

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

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

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

- **password**
- **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**.

Connect from Windows

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

University of Manitoba

X2Go: Linux/Mac/Windows

Why X2Go: access to GUI

How to use X2Go?

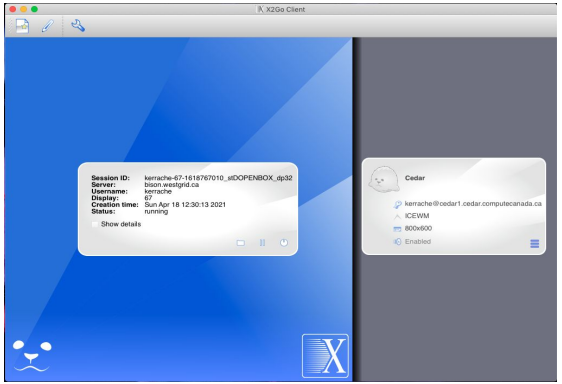
- Ask first if X2Go is installed on the remote cluster.
- If yes, install X2Go client on your laptop or Desktop.
- Linux, Windows, Mac (requires XQuartz)
- Launch X2Go.
- Create a session and connect:

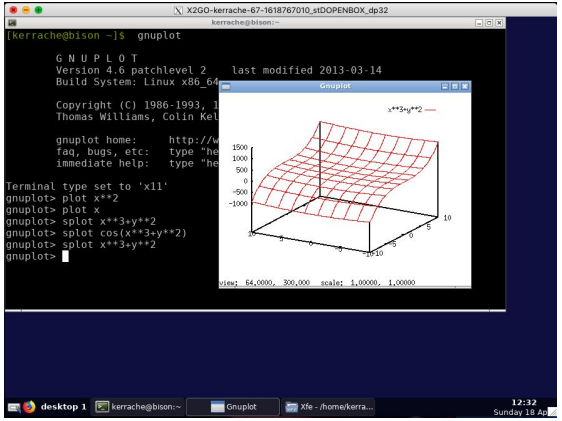
  Login: your user name

  Host: bison.westgrid.ca (or tatanka.westgrid.ca)

  Port: 22

  Session: ICEWM





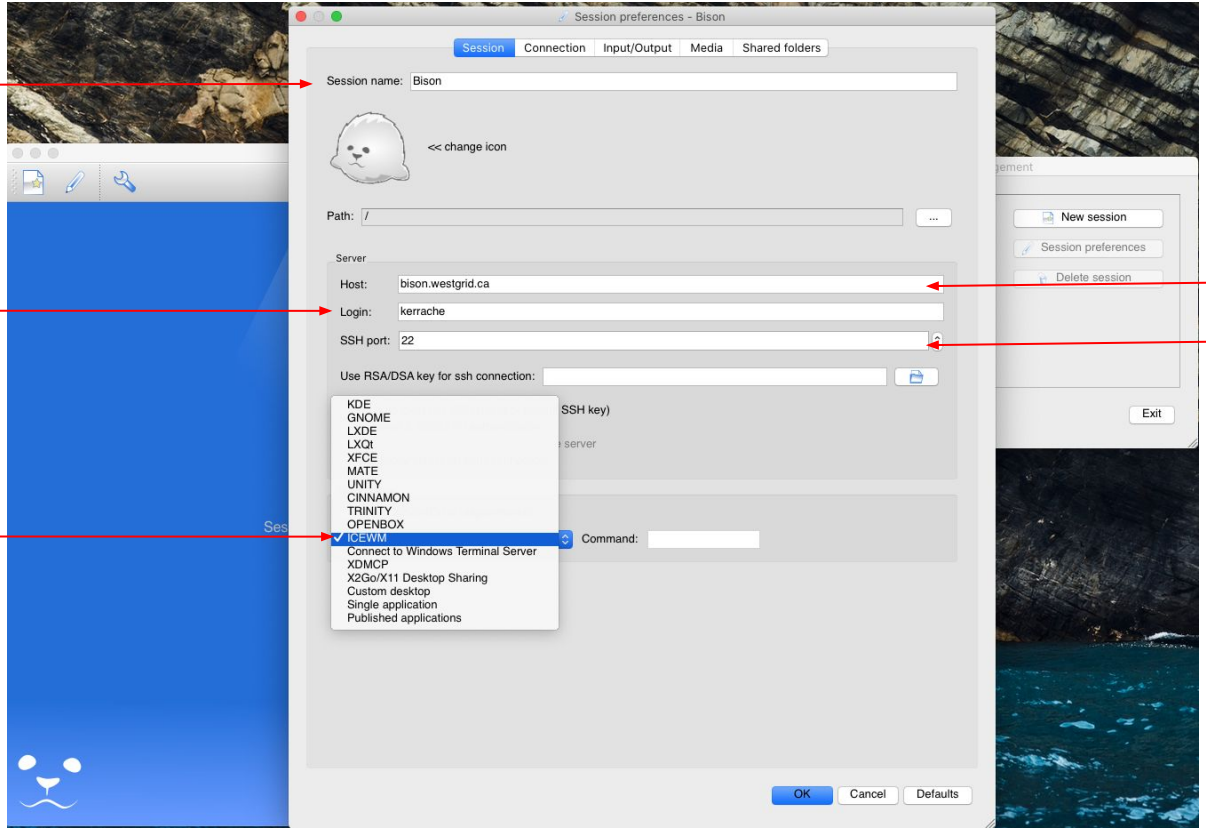
University of Manitoba

X2Go: Linux/Mac/Windows

Session name

User name

Session type



host name

Port

University of Manitoba

File transfer: scp, rsync, sftp

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).

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

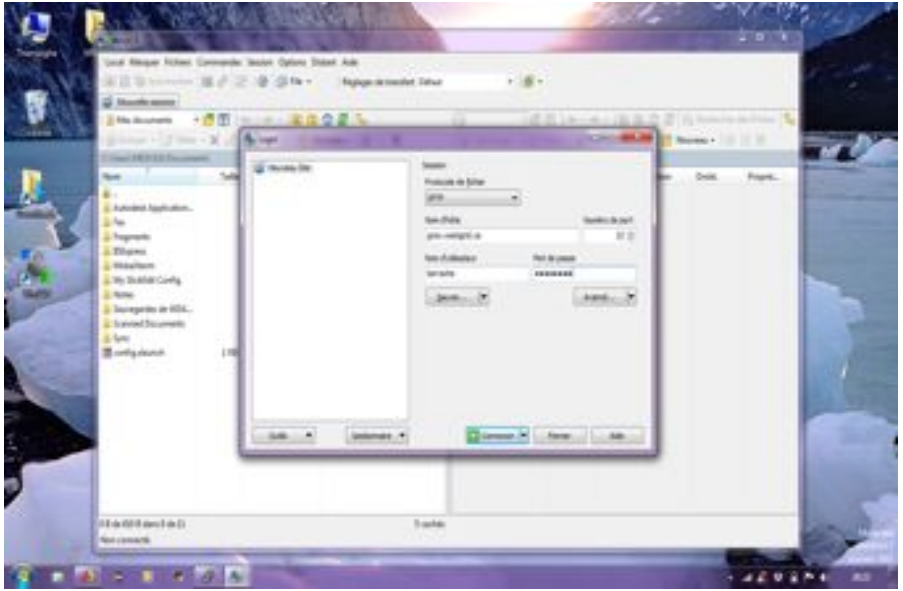
username@grex.westgrid.ca:/home/username/{Your_Dir}; ~/{Your_Dir}

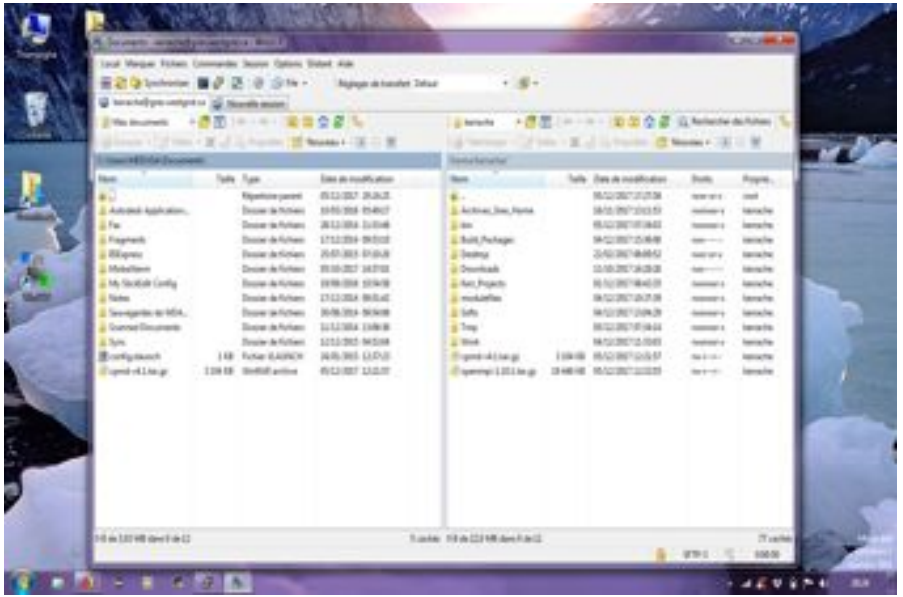
username@grex.westgrid.ca:~/{Your_Dir}

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

File transfer: FileZilla, WinSCP

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



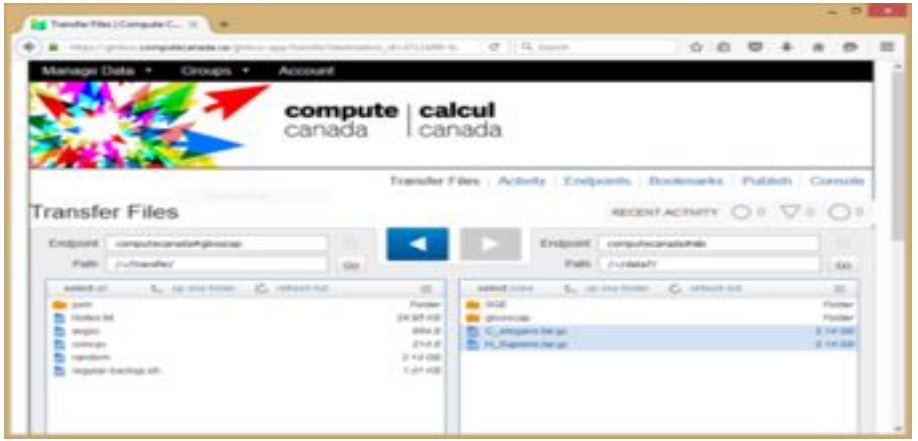


- Navigate on your local machine.
- Navigate on remote machine.
- Copy your files (works on both ways).

File transfer: Globus

For more information:
https://docs.computecanada.ca/wiki/Globus/en

Works between Compute Canada clusters



- Globus is a service for fast, reliable, and secure data transfer.
- Automatically tuning transfer settings, restarting interrupted transfers, and checking file integrity.

File system and quota

Compute Canada:

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

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

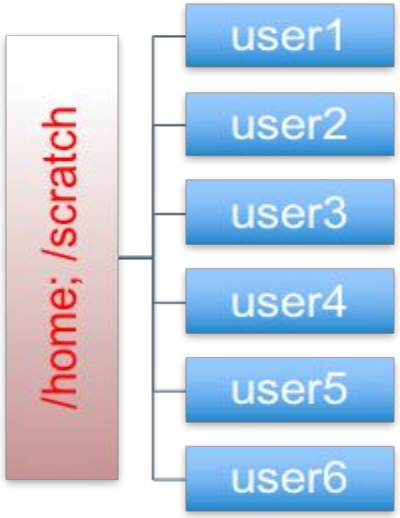
Grex:

/home/$USER:

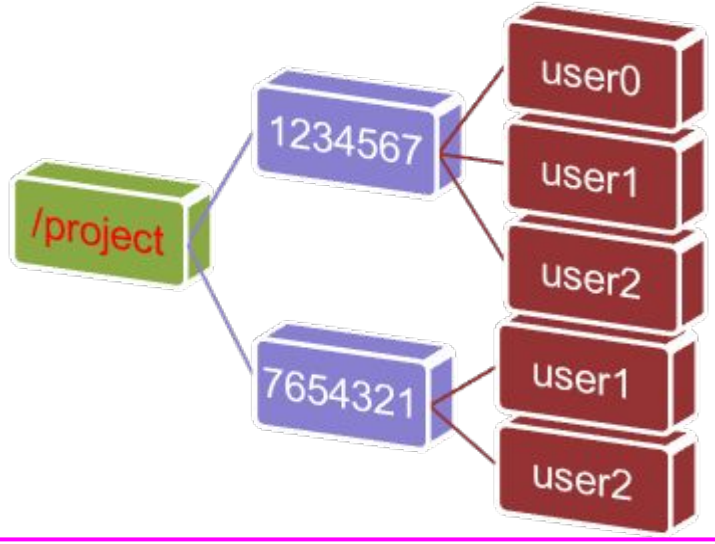
100 GB, backup

/global/scratch/$USER:

2+ TB, no backup, no purge.



Project in Compute Canada clusters



1 TB per group; extension up to 10 TB

Backup; Allocatable via RAC (>10 TB)

University of Manitoba

Quota: diskusage_report

```
[kerrache@cedar1: ~]$ diskusage_report
```

Description	Space	# of files
/home (user kerrache)	72M/50G	652/500k
/scratch (user kerrache)	1978M/20T	8517/1000k
/project (group kerrache)	0/2048k	0/500k
/project (group def-kerrache-ab)	40G/1000G	327/500k
/project (group def-kerrache)	1838G/10T	9623/500k

```
[kerrache@tatanka ~]$  diskusage_report
```

Description (FS)	Space (U/Q)	# of files (U/Q)
/home (kerrache)	226M/31G	2381/500k
/global/scratch (kerrache)	519G/2147G	27k/1000k

University of Manitoba

Software on HPC clusters

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

Up to a user, …

Free Software: GNU Public License.

Open Source, Binaries, Libraries, …

Commercial Software:

Contact us with some details about the license, …

We install the program and protect it with a POSIX group.

University of Manitoba

Software on HPC clusters

- ★ Number-crunching software environment: compilers, libraries, ...
  - Compilers (GCC, Intel), BLAS/LAPACK/PETSc, 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.computecanada.ca/wiki/Available_software
- ★ **Grex:**
  - **CCEnv:** access to public repository of Compute Canada
  - **GrexEnv:** modules installed locally on Grex.

University of Manitoba

Find a software on a cluster

Lmod

- **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.
- **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

University of Manitoba

Running jobs on a cluster

- When you connect you get interactive session on a login node:
  - Resources there are limited: used for basic operations
    - editing files, compiling codes, download or transfer data, submit and monitor jobs, run short tests {no memory intensive test}
  - Performance can suffer greatly from oversubscription
- 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 <some options> myscript.sh
- For interactive work, submit interactive jobs: salloc
  - SLURM uses salloc for interactive jobs
  - The jobs will run on dedicated compute nodes [not on head node]

University of Manitoba

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 using?
    - 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 “my-script.sh” with 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

Example of interactive job: salloc

```
[kerrache@bison ~]$  salloc --ntasks=1 --cpus-per-task=4 --mem-per-cpu=2500M
--time=1:00:00 --account=def-kerrache --x11
salloc: using account: def-kerrache
salloc: using default partition
salloc: Pending job allocation 4782977
salloc: job 4782977 queued and waiting for resources
salloc: job 4782977 has been allocated resources
salloc: Granted job allocation 4782977
salloc: Waiting for resource configuration
salloc: Nodes c318 are ready for job
[kerrache@c318 ~]$  sq
       JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
       4782977   compute     sh kerrache  R       2:09      1 c318
```

University of Manitoba

Minimal job script: sbatch

```
#!/bin/bash
#SBATCH --account=def-somegroup
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --mem=256M
#SBATCH --time=3:00:00   # DD-HH:MN:SS
#SBATCH --partition=compute

echo "Starting run at: `date`"
# Add your commands here:
./your-program {+options}

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 my-script.sh
- squeue -u $USER [or sq]
- seff -d <JOB_ID>

Partition:

- partition-list
- sinfo -p <partition name>

University of Manitoba

Example of script: Serial job

```
#!/bin/bash
#SBATCH --account=def-somegroup
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=2500M
#SBATCH --time=3-00:00:00
#SBATCH --partition=compute

# Load appropriate modules:
module load gcc/9.2 samtools/1.10
echo "Starting run at: `date`"
samtools <command> [options]
echo "Program finished with exit code $? at: `date`"
```

```
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --mem=1500M
#SBATCH --partition=compute
```

```
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --mem=8000M
#SBATCH --partition=largemem
```

Submit the job:

- sbatch myscript.sh
- sbatch --partition=compute myscript.sh
- squeue -j <JOB_ID>

University of Manitoba

Example of script: Gaussian

```
#!/bin/bash
#SBATCH --account=def-somegroup
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=2500M
#SBATCH --time=3-00:00:00
#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`"
```

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

```
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=52
#SBATCH --mem=0
#SBATCH --partition=skylake
```

```
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=8
#SBATCH --mem-per-cpu=1500M
#SBATCH --partition=skylake
```

University of Manitoba

Example of script: MPI

```
#!/bin/bash
#SBATCH --account=def-somegroup
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=8
#SBATCH --mem-per-cpu=1500M
#SBATCH --time=3-00:00:00
#SBATCH --partition=compute

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

```
#SBATCH --ntasks=16
#SBATCH --mem-per-cpu=1500M
#SBATCH --partition=compute
```

```
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=12
#SBATCH --mem=0
#SBATCH --partition=compute
```

```
#SBATCH --ntasks=96
#SBATCH --mem-per-cpu=1500M
#SBATCH --partition=skylake
```

```
#SBATCH --ntasks=144
#SBATCH --mem-per-cpu=1200M
```

University of Manitoba

Monitor your jobs

```
squeue -u $USER; [-t RUNNING]; [-t R]; [-t PENDING]; [-t PD]      # list all current jobs.
squeue -p PartitionName                                           # list all jobs in a partition.
sinfo -p PartitionName; --state=idle,alloc      # view information about partitions/nodes.
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 --states=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.
```

University of Manitoba

Estimating resources: CPUs

- ★ How to estimate the CPU resources?
  - No direct answer: it depends on the code
  - Serial code: 1 core [--ntasks=1 --mem=2500M]
  - Threaded and OpenMP: no more than available cores on a node [--cpus-per-task=12]
  - MPI jobs: can run across the nodes [--nodes=2 --ntasks-per-node=12 --mem=0].
- ★ Are threaded jobs very efficient?
  - Depends on how the code is written
  - Does not scale very well
  - Run a benchmark and compare the performance and efficiency.
- ★ Are MPI jobs very efficient?
  - Scale very well with the problem size
  - Limited number of cores for small size: when using domain decomposition
  - Run a benchmark and compare the efficiency.

University of Manitoba

Estimating resources: memory

- ★ How to estimate the memory for my job?
  - No direct answer: it depends on the code
  - Java applications require more memory in general
  - Hard to estimate the memory when running R, Python, Perl, …
- ★ To estimate the memory, run tests:
  - Interactive job, ssh to the node and run top -u $USER {-H}
  - Start smaller and increase the memory
  - Use whole memory of the node; seff <JOBID>; then adjust for similar jobs
  - MPI jobs can aggregate more memory when increasing the number of cores
- ★ What are the best practices for evaluation the memory:
  - Run tests and see how much memory is used for your jobs {seff; sacct}
  - **Do not oversubscribe the memory** since it will affect the usage and the waiting time: accounting group charged for resources reserved and not used properly.

University of Manitoba

Estimating resources: run time

- ★ How to estimate the run time for my job?
  - No direct answer: it depends on the job and the problem size.
  - See if the code can use checkpoints, then split the simulation into multiple short jobs.
  - For linear problems: use a small set; then estimate the run time accordingly if you use more steps (extrapolate).
- ★ To estimate the time, run tests:
  - Over-estimate the time for the first tests and adjust for similar jobs and problem size.
- ★ What are the best practices for time used to run jobs?
  - Have a good estimation of the run time after multiple tests.
  - Analyse the time used for previous successful jobs.
  - Add a margin of 15 to 20 % of that time to be sure that the jobs will finish.
  - Do not overestimate the wall time since it will affect the start time: longer jobs have access to smaller partition on the cluster.

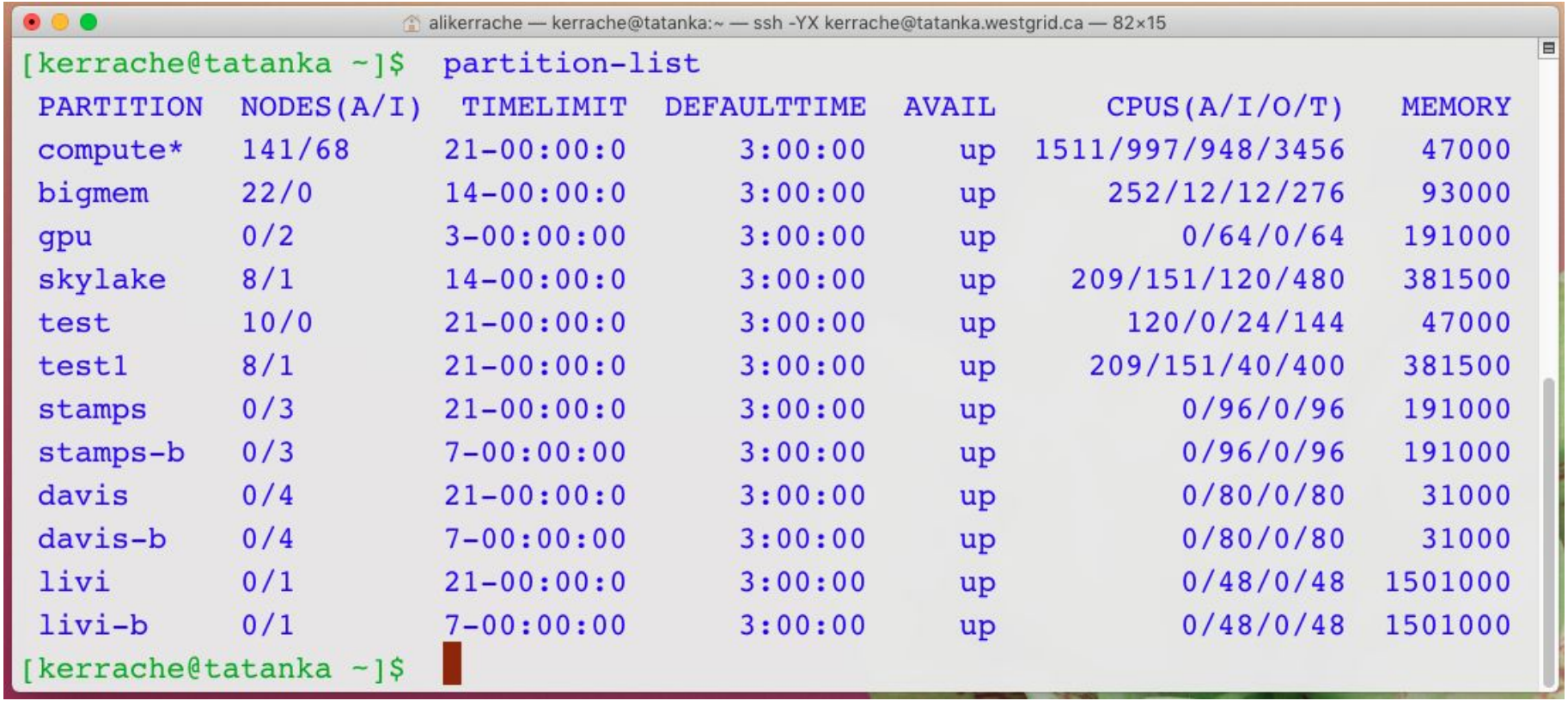
Useful commands/custom scripts

- ★ Custom scripts:
  - partition-list
  - grex-summarize-queue
  - slurm-nodes-state
- ★ Other commands:
  - sinfo -p <partition name>
  - sinfo --state=idle,alloc,mix
  - scontrol show reservation
  - squeue -p <partition name> ; -t R; -t PD
  - share -U --user <your user name>
  - scontrol scontrol show partition <partition name>
  - sshare -U --user <user name>
  - sshare -l -A <account> --format=Account,EffectvUsage,LevelFS

```
[kerrache@bison ~]$  sinfo --state=idle
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
compute*  up 21-00:00:0      3   idle n[135,181,283]
gpu       up 3-00:00:00      2   idle g[324-325]
largemem  up 14-00:00:0      2   idle n[326,335]
skylake   up 21-00:00:0      0      n/a
stamps    up 21-00:00:0      3   idle g[321-323]
stamps-b  up 7-00:00:00      3   idle g[321-323]
livi      up 21-00:00:0      1   idle g338
livi-b    up 7-00:00:00      1   idle g338
```

University of Manitoba

Partitions on Grex: partition-list



University of Manitoba

Queue overview

Custom script: grex-summarize-queue

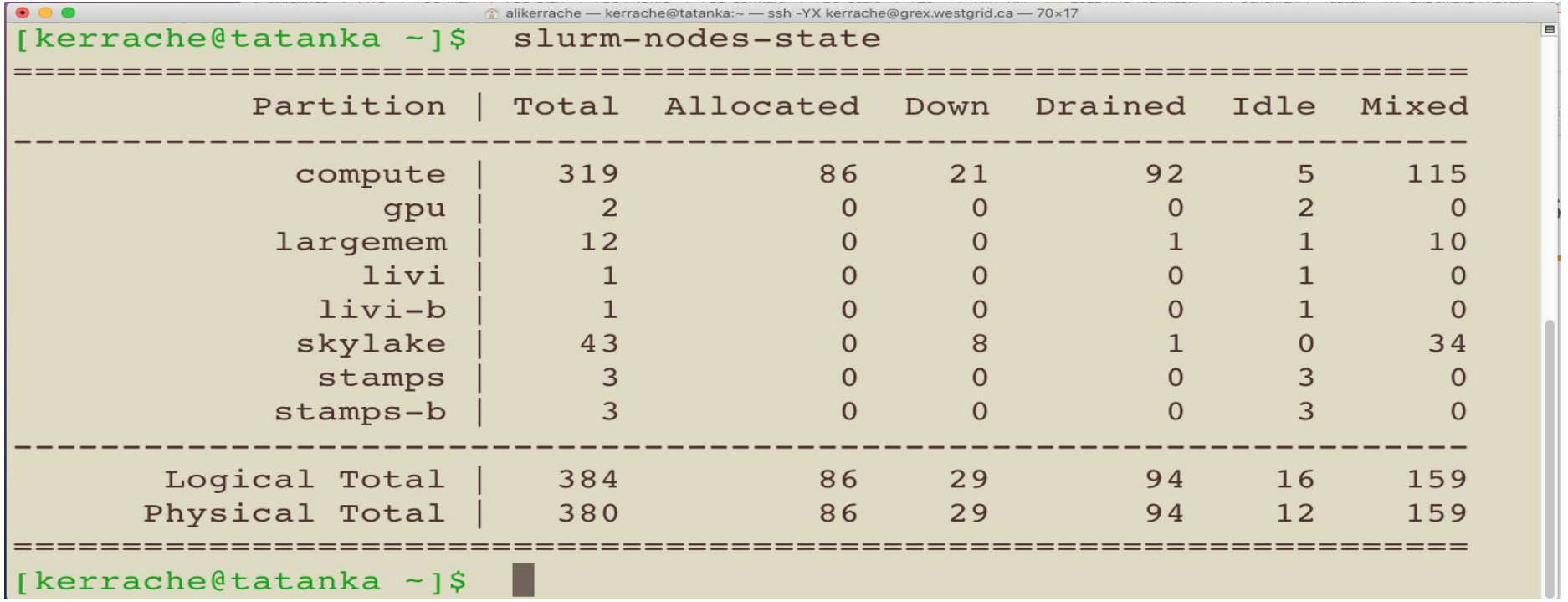
alikerrache — kerrache@bison:~ — ssh -YX kerrache@grex.westgrid.ca — 90×27

```
[kerrache@bison ~]$  grex-summarize-queue
-------------------------------------------------------------------------------
 grex-summarize-queue.py: queue summarized by user.
 ======> Collected at: 2021-04-18 12:11:53.460199
-----------------------------+-----------------------------+--------------------------
  RUNNING-JOBS               |   PENDING-JOBS              |  USER          ACCOUNT
  CORES    GPUS   JOBS HOURS |   CORES    GPUS   JOBS HOURS |  NAME          NAME
-----------------------------+-----------------------------+--------------------------
    864       0      3    71 |       0       0      0     0 |  umbergm5      def-bcwang
    544       0     43   431 |       0       0      0     0 |  zhang86       def-schrecke
    200       0      2     6 |    8600       0     86     9 |  aleila        def-torabi
     96       0      3   197 |      40       0      1   335 |  ismael        def-jhollett
     48       0      2    57 |       0       0      0     0 |  sohail        def-schrecke
     39       0      3   126 |       0       0      0     0 |  babarinv      def-telichev
     36       0      1   111 |       0       0      0     0 |  rico          def-schrecke
     24       0      1   170 |       0       0      0     0 |  xiaojw        def-dengc
     16       0      2   242 |       0       0      0     0 |  singhs34      def-davisrl
     12       0      1   487 |       0       0      0     0 |  umdorge6      def-bibeauel
      4       0      1   196 |       0       0      0     0 |  bergma16      rkm-871-aa
      0       0      0     0 |     120       0      1    95 |  xwj           def-bcwang
-----------------------------+-----------------------------+--------------------------
  TOTAL-RUNNING-JOBS         |   TOTAL-PENDING-JOBS        |  TOTAL         TOTAL
  CORES    GPUS   JOBS HOURS |   CORES    GPUS   JOBS HOURS |  USERS         ACCOUNTS
-----------------------------+-----------------------------+--------------------------
   1883       0     62   182 |    8760       0     88    12 |  12 total users 9 total accounts
-----------------------------+-----------------------------+--------------------------
[kerrache@bison ~]$
```

University of Manitoba

Nodes overview

Custom script: slurm-nodes-state



Documentation for Grex

Unofficial Grex User Guide

Search

Notes of Grex Changes

Accessing Compute Canada resources

Grex HPC Documentation

Access and Usage conditions

Connecting / Transferring data

Storage and Data

Running Jobs

Software

Frequently Asked Questions

Local IT Resources

Support and Training

Disclaimer

User documentation for HPC resources at University of Manitoba

Since you have found this Website, you may be interested in Grex documentation. Grex is the University of Manitoba's High-Performance Computing system.

User documentation for HPC resources at University of Manitoba

- For experienced Grex users
- For new Grex users
  - A Very Quick Start guide
- Useful links

- ★ In the process of migration to GitHub
- ★ The links available from the message of the day
- ★ Pay attention to the message of the day

https://monitor.hpc.umanitoba.ca/doc/
https://um-grex.github.io/grex-docs/

University of Manitoba

Compute Canada documentation

- Systems and services
- Guides

Systems and services [edit]

- List of current Compute Canada systems
- Cedar, Graham and Béluga, general-purpose clusters
  - System status and **upcoming outages**
  - Known issues
- Niagara, a cluster designed for large parallel jobs
- Hélios, a GPU cluster
- Available software
- National Data Cyberinfrastructure, long-term and tape storage services (limited availability)
- Cloud computing service
- Globus file transfer service
- Policy table of contents
- FAQ, Frequently Asked Questions
- Using a resource allocation, a guide for Principal Investigators
  - RAC 2019 transition FAQ, notes on the implementation of 2019 RAC awards

How-to guides [edit]

- Getting started
  - Getting started with the new national systems (mini-webinar series)
  - Niagara Quick Start Guide
  - SSH - How to connect to our servers
  - Linux introduction
- Storage and file management
  - Transferring data
  - Scratch purging policy
- Best practices for data migration
- Using modules to access software
- Running jobs
- Installing software yourself
- Programming guide
- Visualization
- How to get technical support

- Links to specific documentation by disciplines
- Links to the documentation from regional partners

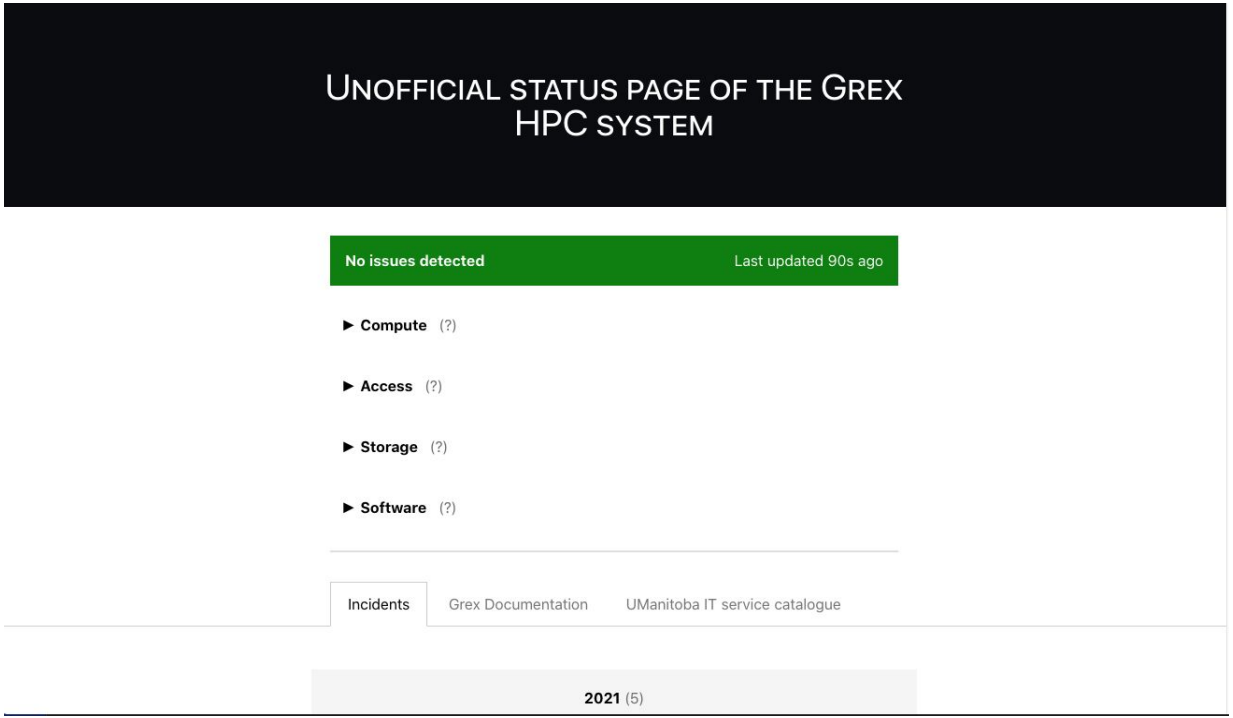
Discipline guides [edit]

- AI and Machine Learning
- Bioinformatics
- Biomolecular simulation
- Computational chemistry
- Computational fluid dynamics (CFD)
- Geographic information systems (GIS)
- Humanities
- Subatomic physics

Regional partners and services [edit]

- WestGrid
- SHARCNET
- SciNet
- Centre for Advanced Computing
- Calcul Québec
- ACENET
- ownCloud storage service

Status page



Past events:

2021 (5)

Brief power outages on October 23, 2021 — 4D AGO
Resolved after 9h 30m of downtime

Planned power outage Aug 31 to Sept 1 — 58D AGO
Resolved after 28h 0m of downtime

Grex update outage, New Status webpage — 3 MONTHS AGO
NOTICE: Grex is online for production. After the last outage, Grex was in production in a test mode for a week since June 9, 2021. We did not get any reports about the new hardware and software. ...

Rebooting the nodes — 3 MONTHS AGO
Resolved after 243h 30m of downtime

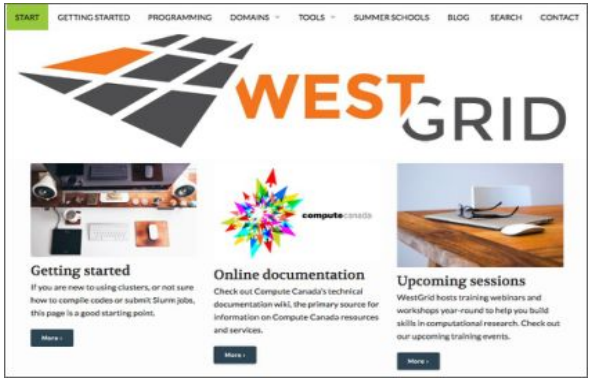
Westgrid network failure — 4 MONTHS AGO
Resolved after 1176h 35m of downtime

https://grex-status.netlify.app/

University of Manitoba

Useful links

Compute Canada: https://docs.computecanada.ca/wiki/Compute_Canada_Documentat
CCDB: https://ccdb.computecanada.ca/security/login
CC Software: https://docs.computecanada.ca/wiki/Available_softwar
Running Jobs: https://docs.computecanada.ca/wiki/Running_jobs
PuTTy: http://www.putty.org/
MobaXterm: https://mobaxterm.mobatek.net/
X2Go: https://wiki.x2go.org/doku.php
Grex: https://monitor.hpc.umanitoba.ca/doc/
https://um-grex.github.io/grex-docs/
Help and support on CC: support@computecanada.ca
WG training material: https://westgrid.github.io/trainingMaterials/



Thank you for your attention

Any question?