High Performance Computing:
Short Introduction

U. of Manitoba and Compute Canada clusters.

Dr. Ali Kerrache

ali.kerrache@umanitoba.ca

University of Manitoba, Sep 20th, 2019.

Outline

- *U. of Manitoba* and *Compute Canada* HPC clusters:

  Grex (UofM), Cedar (SFU), Graham (UW), Beluga (ETS), …
- Connect & transfer files to HPC clusters:

  ***Linux**, **Mac**, Windows, **ssh**, **Putty**, MobaXterm, …*
- Software environment on HPC clusters:

  Compilers, Software Installation, Modules, …
- Running, submitting and monitoring jobs:

  Prepare, compile, submit, run, monitor your Jobs, …
- Conclusions and more readings (links).

WestGrid and Compute Canada

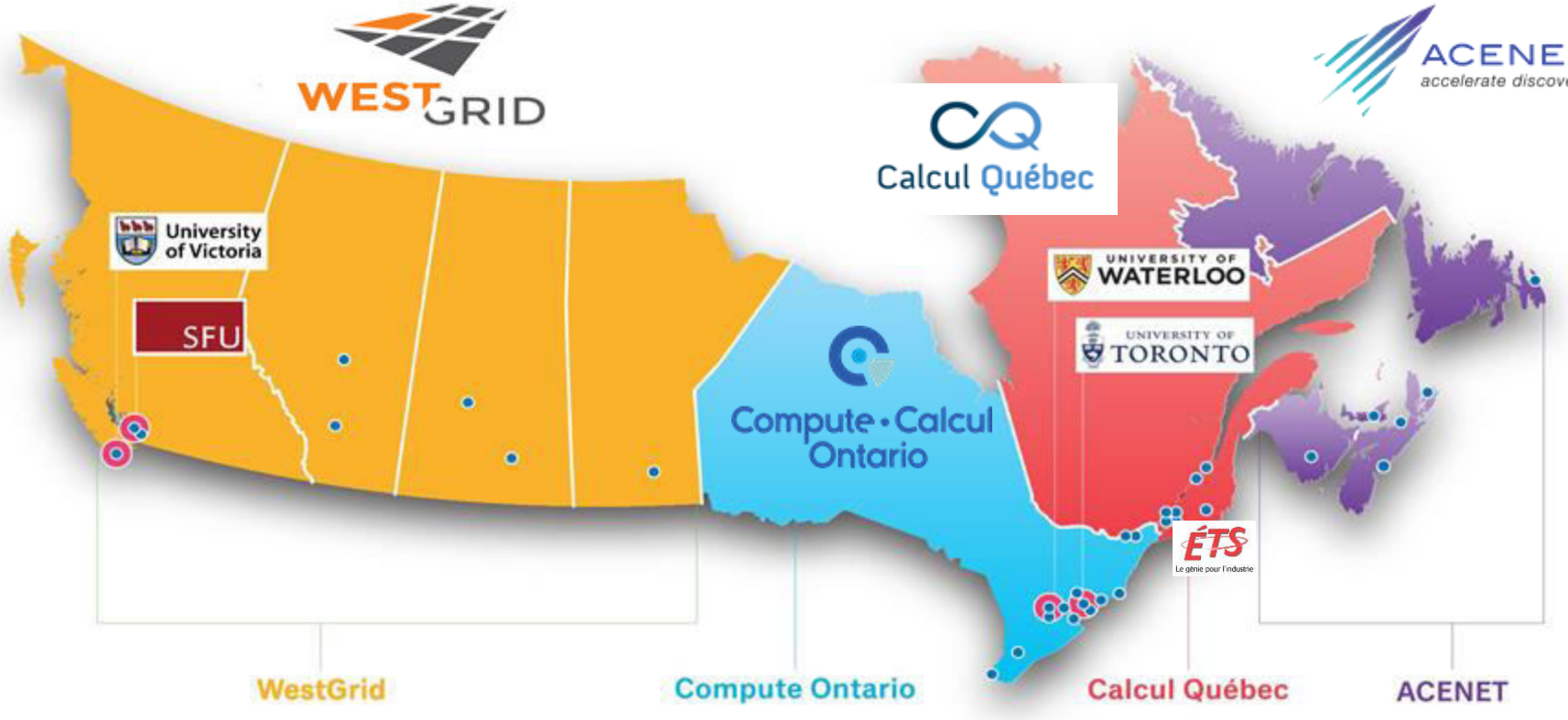
- *WestGrid systems:* https://www.westgrid.ca/
  Grex (UofM): *inherited from WG.*
- *Compute Canada systems:* Cedar, Graham, Beluga, …
  https://docs.computecanada.ca/wiki/Compute_Canada_Documentation

- *Accounts:*
  - Compute Canada Account: https://ccdb.computecanada.ca/security/login
  - WestGrid Account:
    https://www.westgrid.ca/support/accounts/getting_account

- *Support and Help:*
  - support@westgrid.ca
  - support@computecanada.ca

Note: *read and understand the privacy consent.*

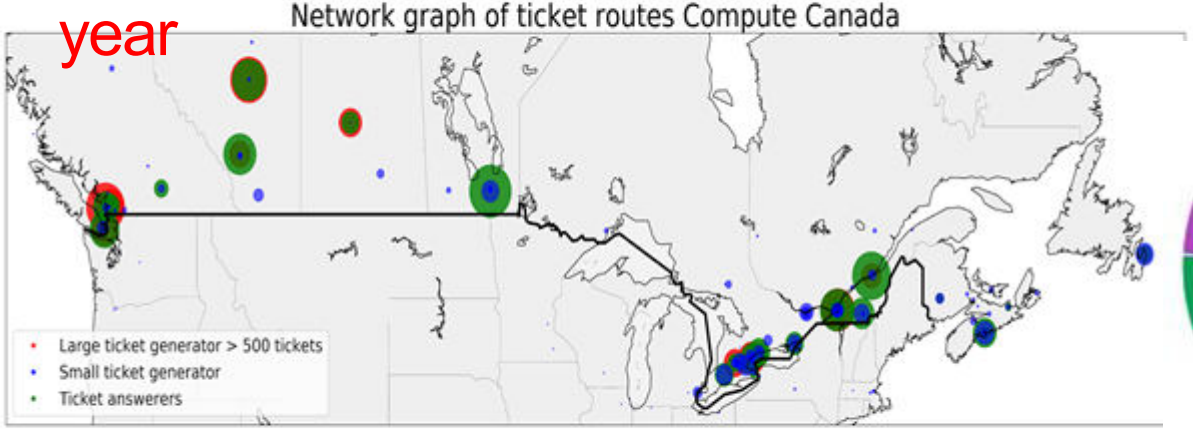
Compute Canada new clusters

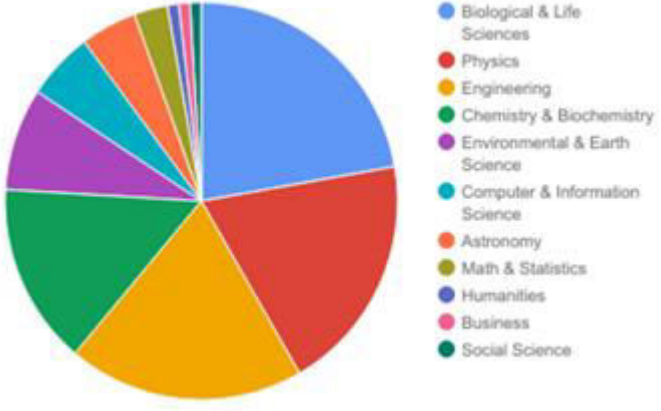


Compute Canada/Grex: people

~200 technical staff

~15,000 user accounts: 20% growth per year





U. of Manitoba:
2 analysts and 2 system admins (IST).
250 roles in CCDB.
Recent usage: 35 active groups (80 users).

- **All research disciplines supported.**
- **Free access for any researcher (and their external collaborators) at a Canadian institution.**

Compute Canada Resources

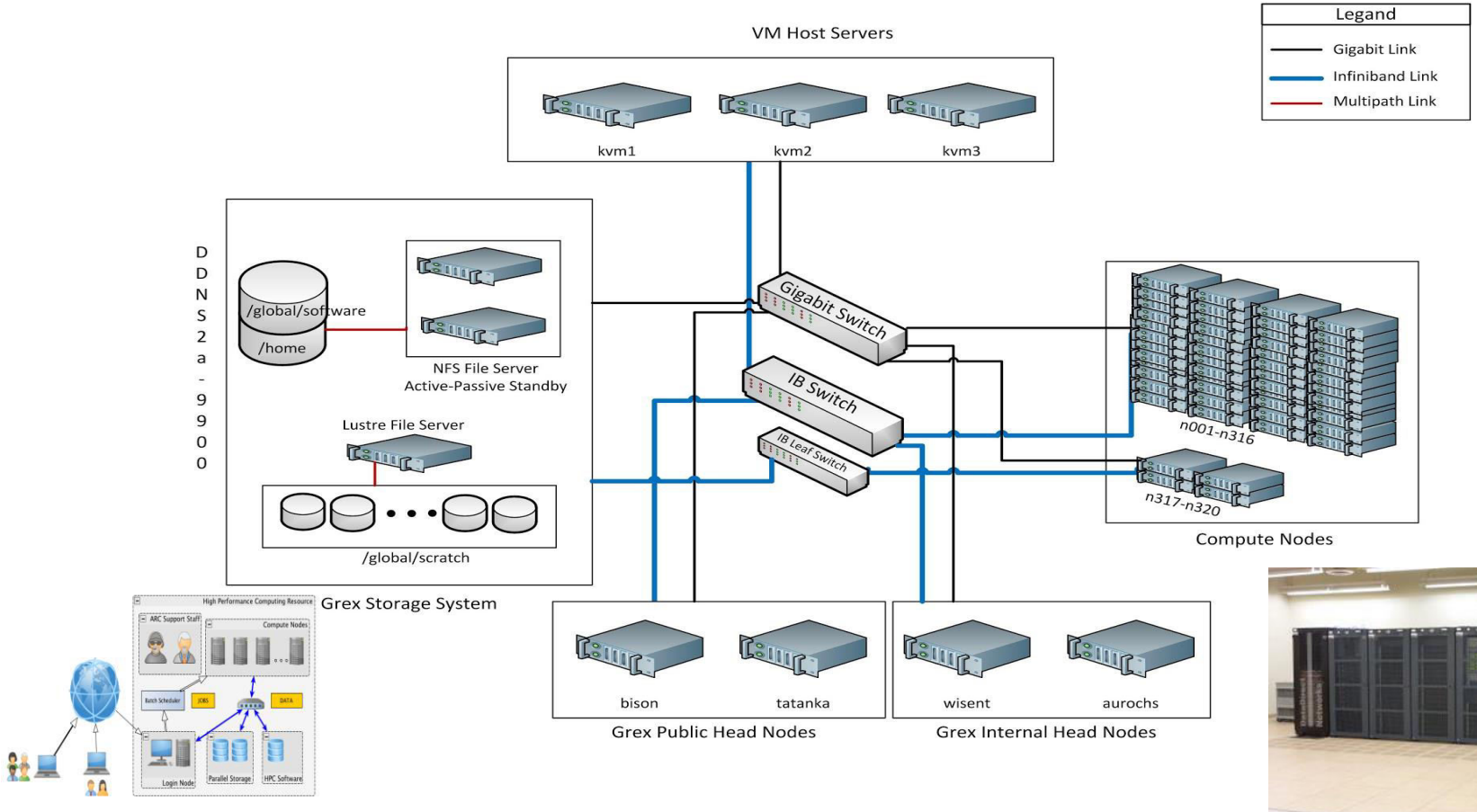
System	Cores	GPUs	Storage	Notes
Cedar	58,416	584	13 PB	NVidia P100 GPUs
Graham	33,472	320	12 PB	NVidia P100 GPUs
Beluga	34,880	688	2.6 PB	NVidia V100 GPUs
Niagara	60,000	0	2 PB	No GPUs; Large parallel jobs.
Arbutus (cloud)	9,048	0	5.7 PB	Physical cores: generally hyper-threaded.
GP Cloud partitions				Cloud partitions are available on GP systems for special purposes.

Resources at U. of Manitoba

	CPUs	GPUs	RAM/node	Storage	Notes
Grex	3780 (315 x 12 CPU) Intel Nehalem 2010	0	48 GB 94 GB	0.4 PB /global/scratch	WG: defunded in 2018 Now: UofM
Free for all CCL	768 (12 * 64 CPU) AMD Opterons 2011	0	256 GB	?	Free for all, use UM NetID
OpenStack	256 cores?	0			By request

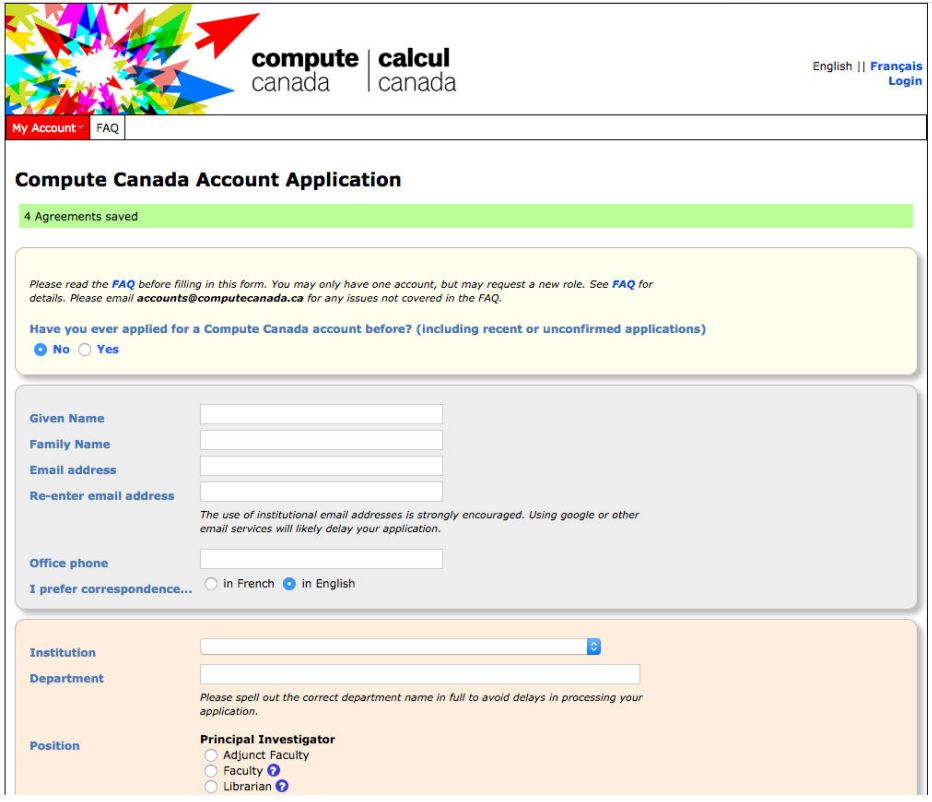
- https://www.westgrid.ca/support/systems/Grex
- http://umanitoba.ca/computing/ist/service_catalogue/index.html

Structure of a HPC clusters: Grex



Accessing Compute Canada

- **Step 1**:
  - Faculty member registers in the Compute Canada Database (CCDB)
    - http://ccdb.computecanada.ca
- **Step 2**:
  - Once account is approved, students / colleagues can register as group members:
    - CCDB account Gives access to the New systems:
  - Access to Compute Canada resources is free for the eligible researchers.
  - Everyone gets a “Default” share.
  - Every PI gets 1 TB of storage by default.
  - Resource Allocation Competitions are held annually, to distribute resources based on proposal’s merit.



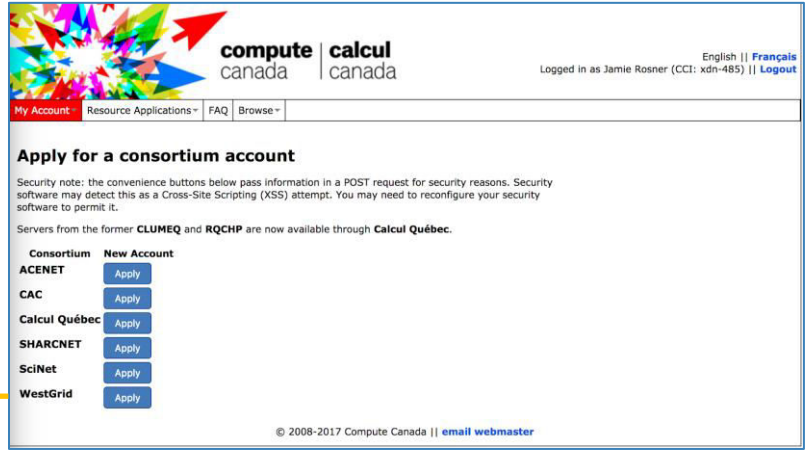
Accessing Grex

Step 3:

- Optional, Regional accounts:

https://ccdb.computecanada.ca/me/facilities

- Needed for Legacy HPC Resources (Grex, MP2, etc.)
- WG: Grex, ownCloud



How to connect to HPC systems?

Connect to a cluster:

- **Linux:**
  ssh; X2Go
- **Mac:**
  ssh, X2Go
- **Windows:**
  Putty, MobaXterm, ...

HPC
- Connect
- Transfer files
- Compile
- Test
- Run
- Analyze Data
- ….

Transfer files:

- **Linux:**
  scp; sftp; rsync; ...
- **Mac:**
  ssh, sftp; rsync; …
- **Windows:**
  WinScp, FileZilla, MobaXterm, …

Connect from Mac/Linux to Grex

Connect to Grex from a terminal:

$ ssh -Y username@grex.westgrid.ca
Password:

Some Information
Massage of The Day

[username@bison ~]$
[username@tatanka ~]$

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**.

File system:

Home directory: /home/username/	30 GB per user; 500 K files; Backup.
Scratch directory: /global/scratch/username/	2 TB per user; 1 M files; No Backup.

Connect to national clusters

Connect from a terminal:

Cedar: $ ssh -XY username@cedar.computecanada.ca
Graham: $ ssh -XY username@graham.computecanada.ca
Beluga: $ ssh -XY username@beluga.computecanada.ca
Niagara: $ ssh -XY username@niagara.computecanada.ca

Compute Canada credentials:

- user name
- password (CCDB)

File system: /home **and** /scratch

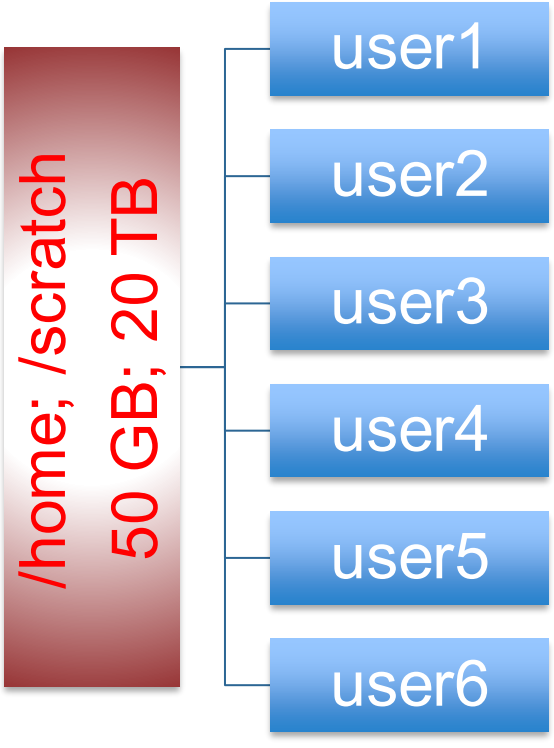
Home directory: /home/username/ 50 GB per user; 500 K files; Backup.
Scratch directory: /scratch/username/ 20 TB per user; 1000 K files; No Backup.

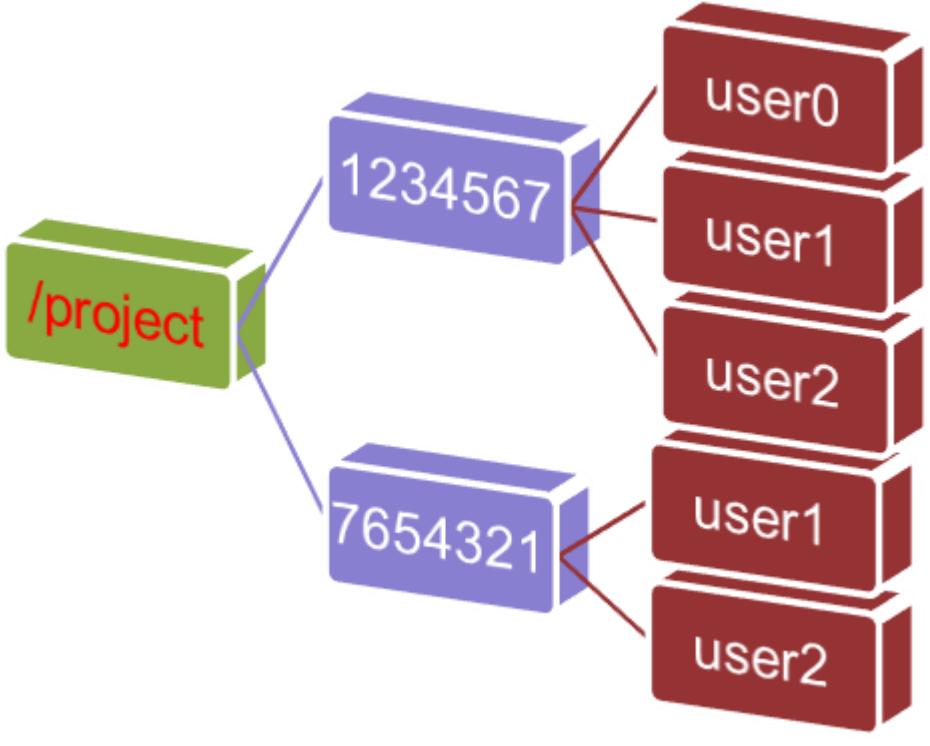
File system: /project 1 TB per group; 500 K files; Backup.

/project/{project-number1}/{user01;user02; …}
/home/{username}/projects/{project-id-1}/{username}
/project/{project-number2}/{user01;user02; …}
/home/{username}/projects/{project-id-2}/{username}

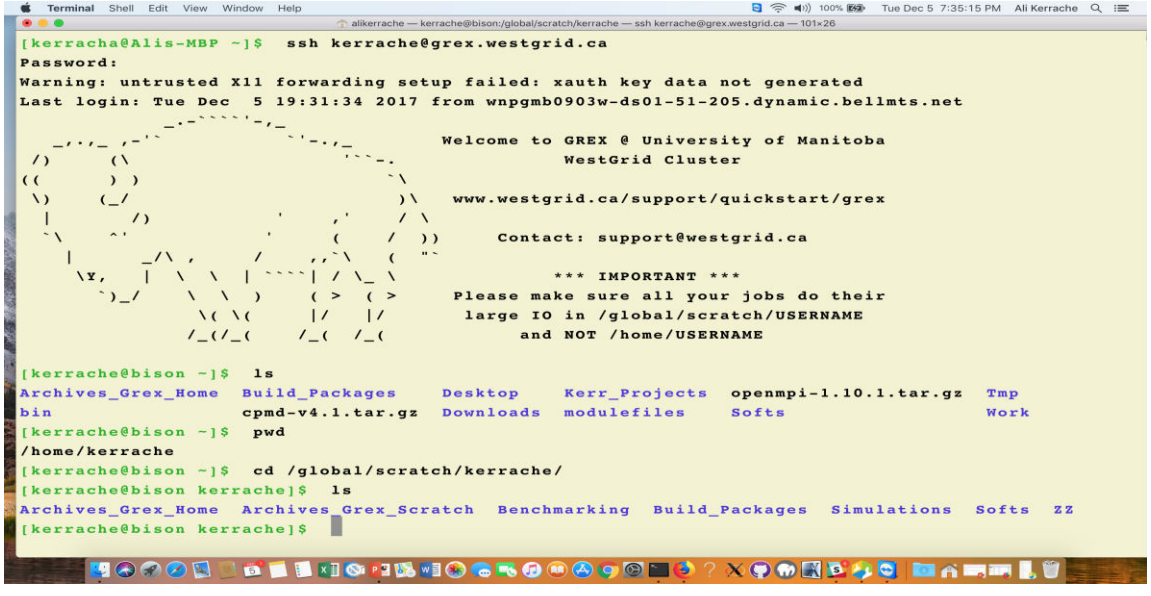
- *Up to 10 TB on request from the PI.*
- More than 10 TB: Requires RAC allocation.

File system structure





Connect from a terminal to Grex



Short demonstration:

- Open a terminal
- Connect to a cluster:
  - Grex: ssh username@grex.westgrid.ca
  - cedar: ssh username@cedar.computecanada.ca
  - graham: ssh username@graham.computecanada.ca
  - beluga: ssh username@beluga.computecanada.ca

How to 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 the appropriate address:

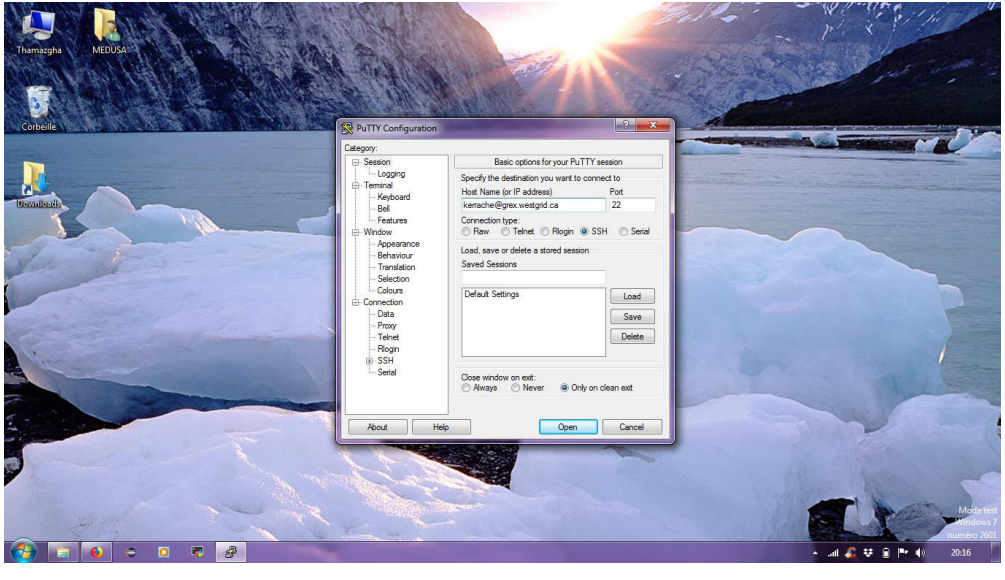
Grex: grex.westgrid.ca
Cedar: cedar.computecanada.ca
Beluga: beluga.computecanada.ca
Graham: graham.computecanada.ca
Niagara: niagara.computecanada.ca

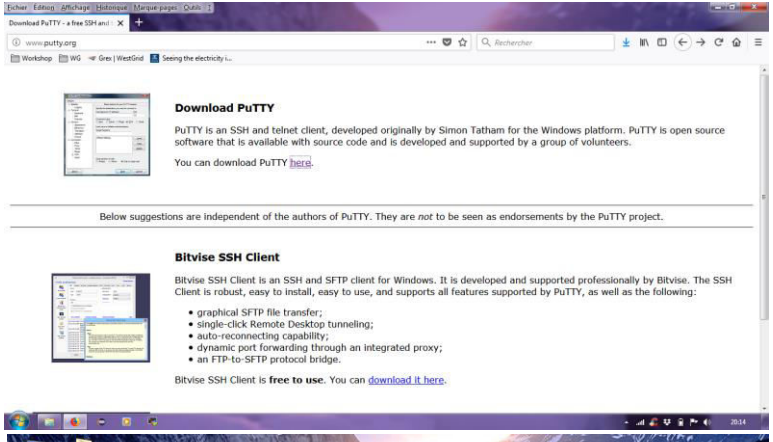
- **Use Cygwin:** same environment as Linux
  - https://www.cygwin.com/

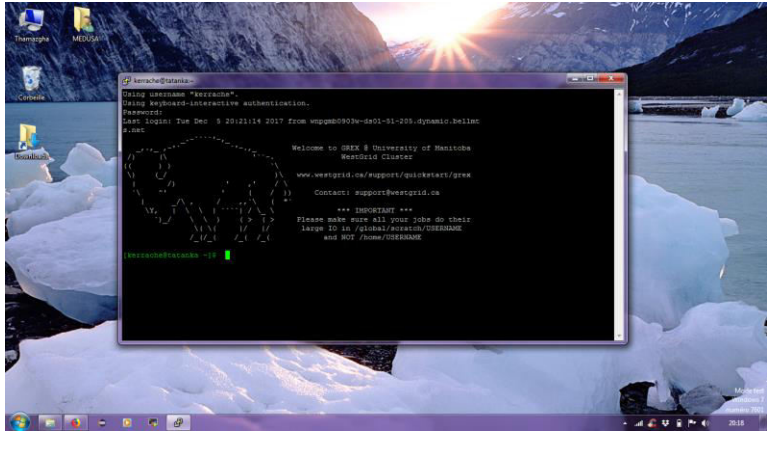
Putty

How to use Putty?

- Download Putty.
- Install it on your laptop or Desktop
- Launch Putty and connect.



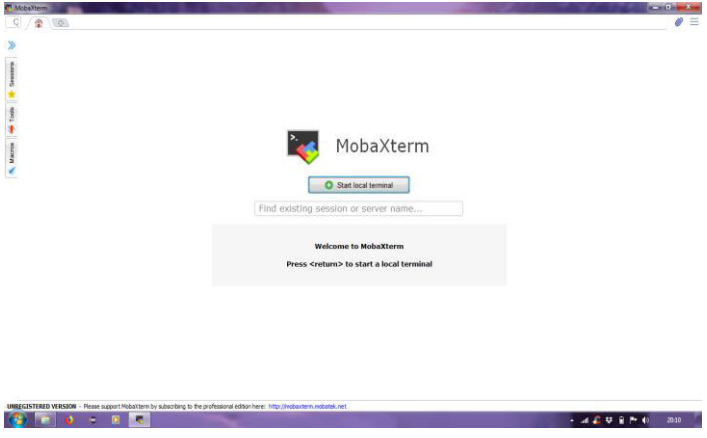


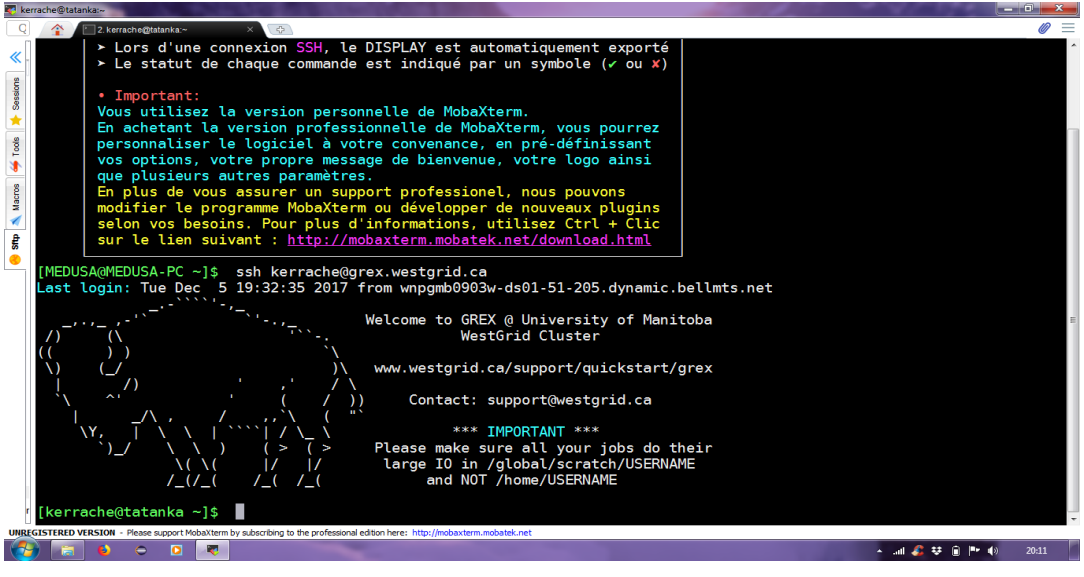


MobaXtem

How to use MobaXterm?

- *Download MobaXterm.*
- *Install it on your laptop or Desktop*
- *Launch MobaXterm and connect.*





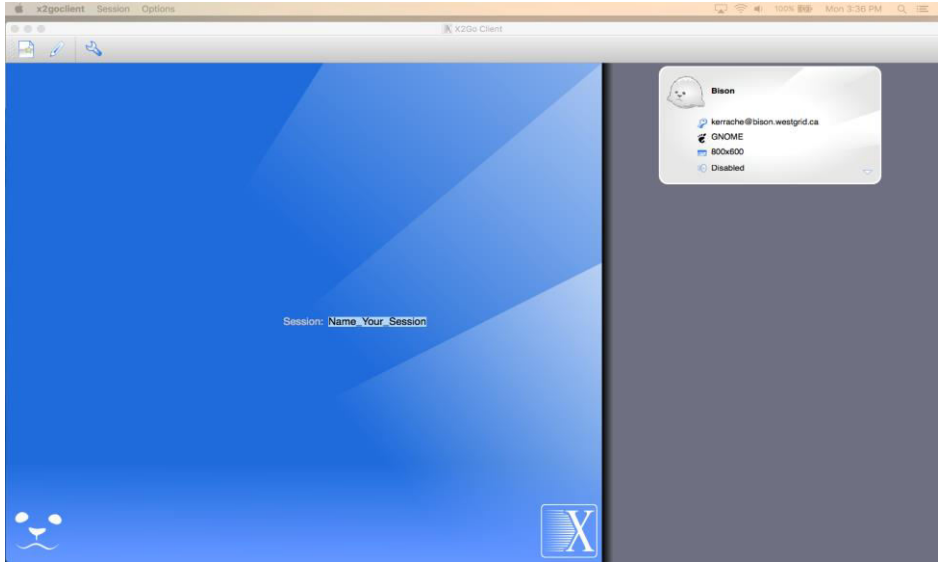
How to connect:

- ✓ *Login: your user name*
- ✓ *Host: grex.westgrid.ca*
- ✓ *Password: your password*
- ✓ *Port: 22*

Use X2Go: Linux/Mac/Windows

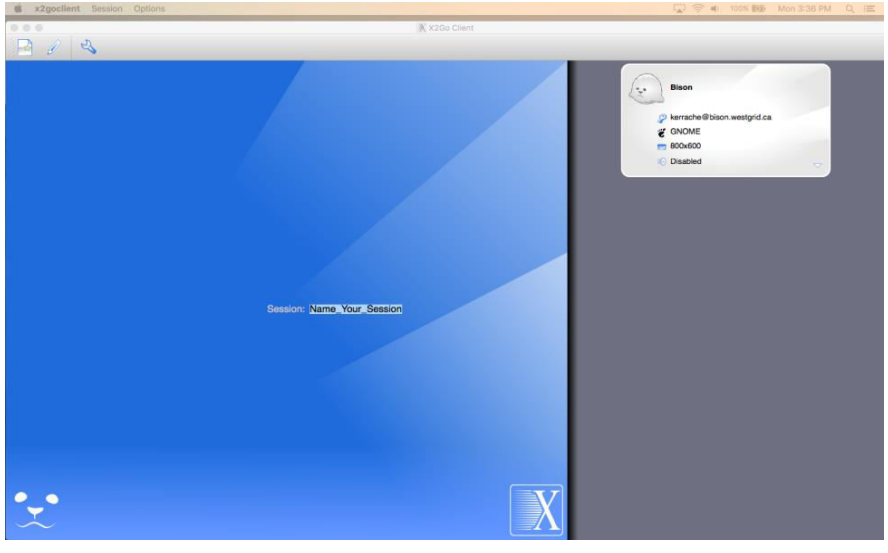
- **Why X2Go:** Access to GUI
- **How to use X2Go?**
  - Ask first of X2Go is installed on the remote machine.
  - If yes, install X2Go client on your laptop or Desktop.
  - Linux, Windows, Mac (XQuartz)

  - Launch X2Go.
  - Create a session and connect.
  - ✓ **Login:** your user name
  - ✓ **Host:** bison.westgrid.ca
  - ✓ **Port:** 22
  - ✓ **Session:** GNOME

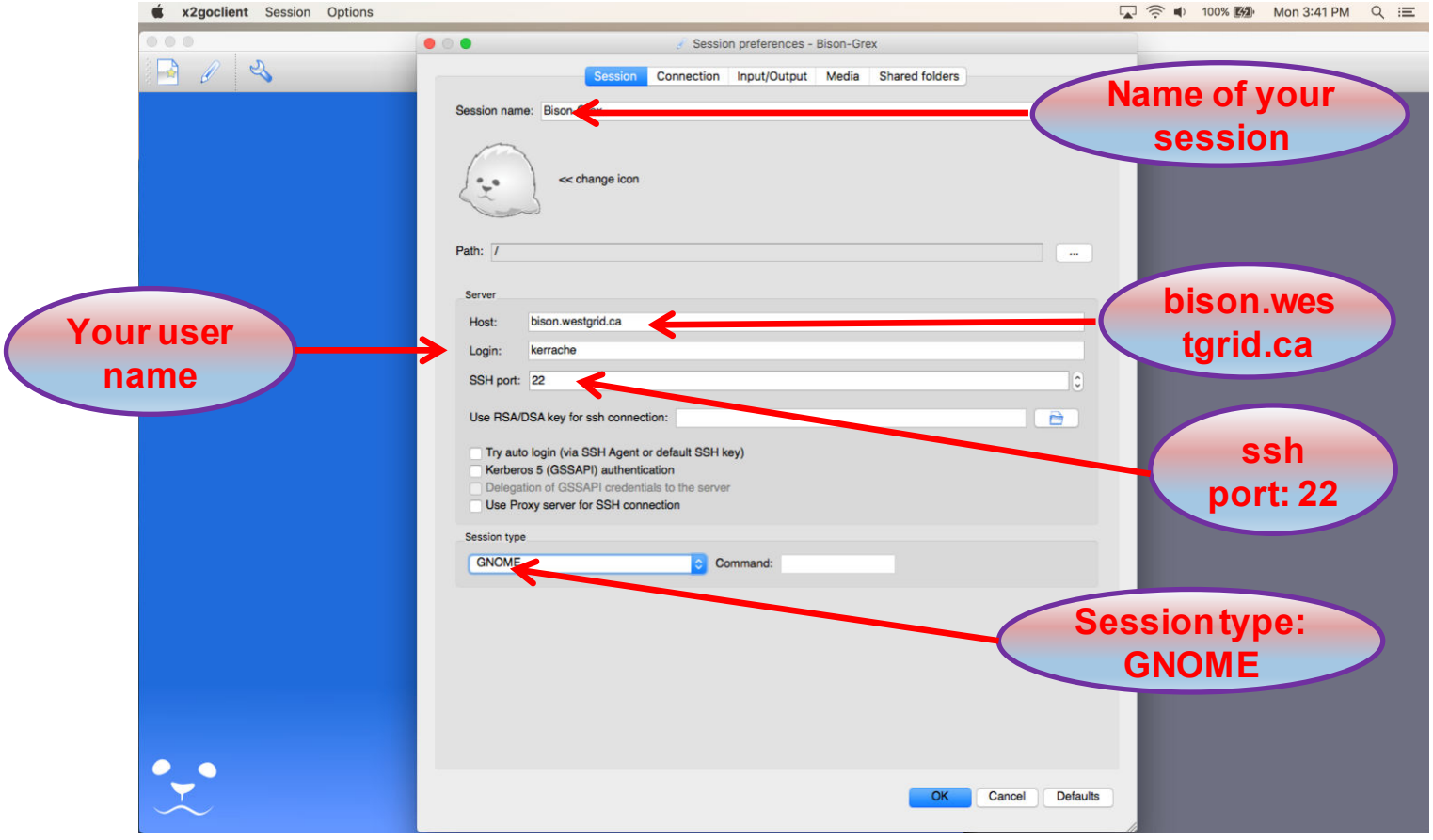


Use X2Go: Linux/Mac/Windows

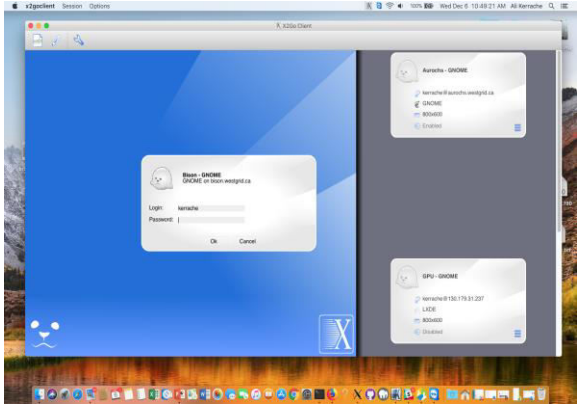
- **Why X2Go:** access to GUI
- **How to use X2Go?**
  - Ask first if X2Go is installed on the remote machine.
  - If yes, install X2Go client on your laptop or Desktop.
  - Linux, Windows, Mac (XQuartz)
  - Launch X2Go.
  - Create a session and connect.
    - ✓ **Login:** your user name
    - ✓ **Host:** bison.westgrid.ca
    - ✓ **Port:** 22
    - ✓ **Session:** GNOME

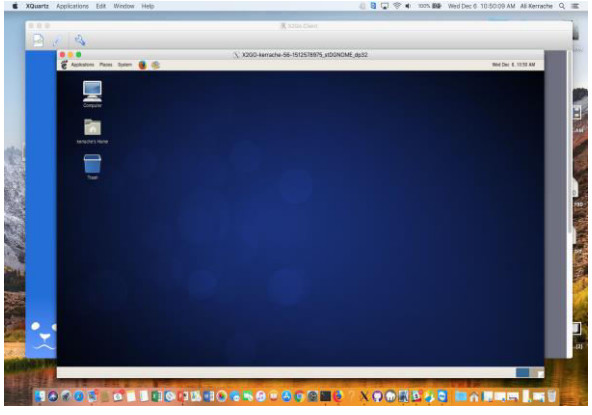


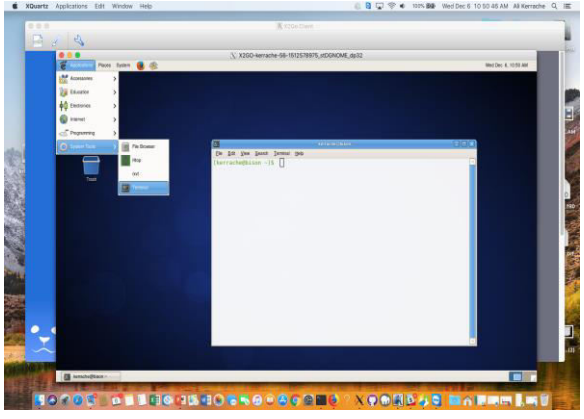
Create a session with X2Go

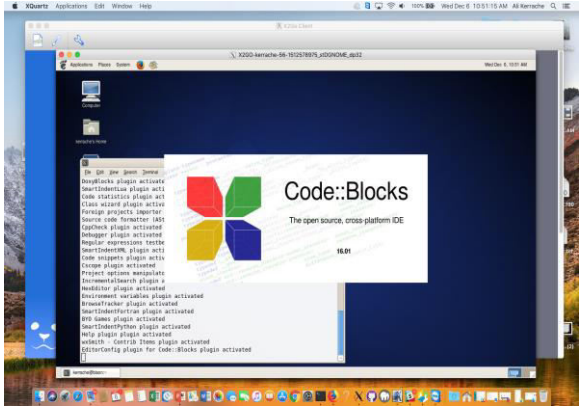


Using a session with X2Go

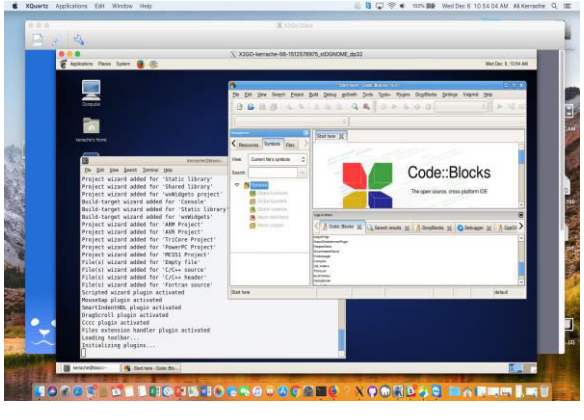








- **Using X2Go:**
  - Launch X2Go.
  - Create a session.
  - Connect.
  - Start a terminal.
  - Load a module.



File Transfer: scp; rsync

- **Terminal:** Linux; Mac; CygWin; MobaXterm.
- **Check if scp; sftp; rsync are supported.**

- **Syntax for scp:** scp [Options] [Target] [Destination]
- **Syntax for rsync:** rsync [Options] [Target] [Destination]

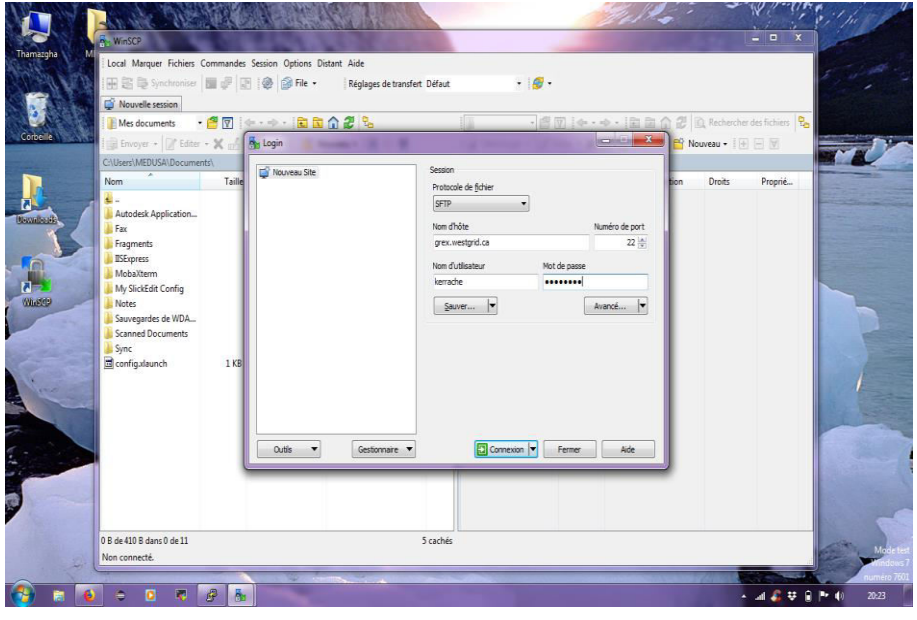
- **Options:** for details use man scp or man rsync from your terminal.
- **Target:** Files or directory to copy (exact path).
- **Destination:** Where to copy the files (exact path).

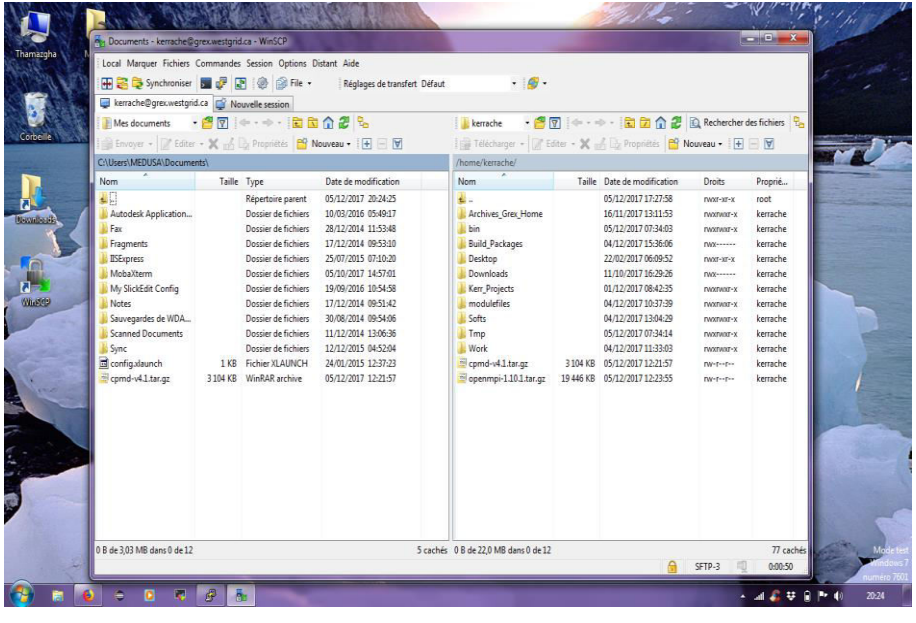
- **Path on remote machine:** examples of a path on Grex.

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

File Transfer: WinScp, FileZilla

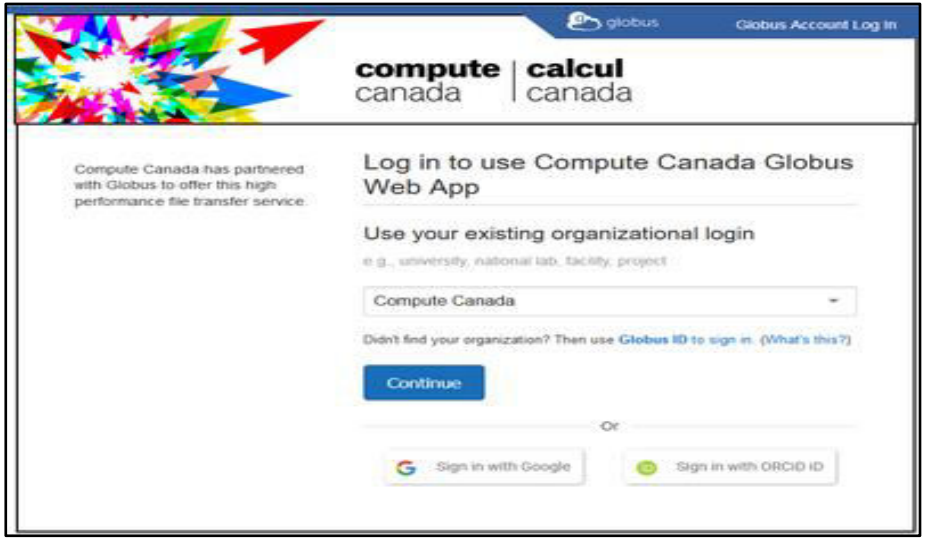
- Install WinScp or FileZilla.
- Launch the program.
- Connect with your credentials.



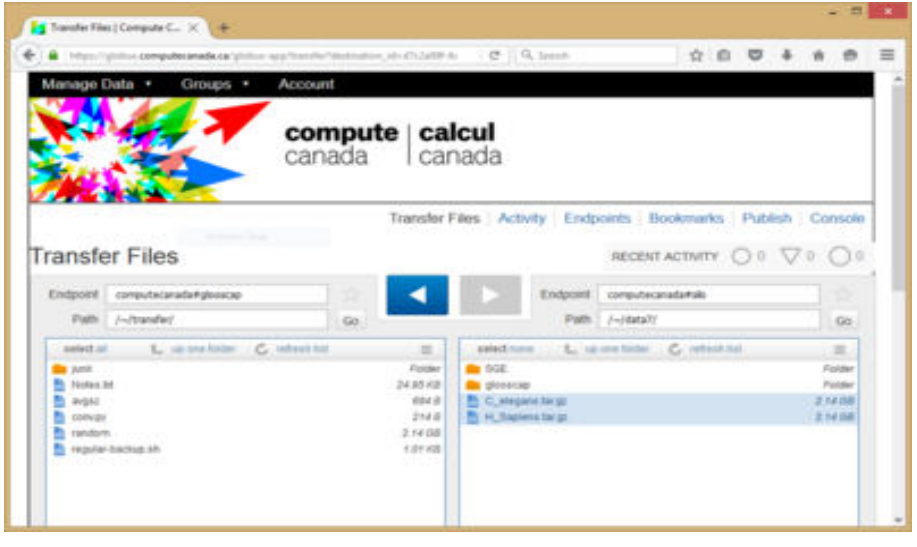


- Navigate on your local machine.
- Navigate on remote machine.
- Copy your files (both ways).

File Transfer: Globus



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



For more information:

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

Software Environment in HPC

- Local installation: /home/$USER; /global/scratch/$USER

  Up to a user (experienced user); Analysts (User's consent).

- Central Installation: /global/software/; /cvfms; …

  Maintained by Analysts, Versions controlled via modules.

- Compilers and other applications:

  Intel and GNU compilers; Tools; Libraries; Visualization; …

Software Environment in HPC

User layer: Python packages, Perl and R modules, home made codes, …

Easybuild layer: modules for Intel, PGI, OpenMPI, CUDA, MKL, high-level applications. Multiple architectures (sse3, avx, avx2, avx512)
`/cvmfs/soft.computecanada.ca/easybuild/{modules,software}/2017`

Nix layer: GNU libc, autotools, make, bash, cat, ls, awk, grep, etc.
`module nixpkgs/16.09 => $EBROOTNIXPKGS=`
`/cvmfs/soft.computecanada.ca/nix/var/nix/profiles/16.09`

Gray area: Slurm, Lustre client libraries, IB/OmniPath/InfiniPath client libraries (all dependencies of OpenMPI).
In Nix layer, but can be overridden using PATH & LD_LIBRARY_PATH.

OS: kernel, daemons, drivers, libcuda, anything privileged (e.g. the sudo command): always local. Some legally restricted software too (VASP).

Which Software to use?

- Home made programs, scripts and tools.

  Up to a user, …

- Free Software: GNU Public License.

  Open Source, Binaries, …

- Commercial Software:

  Contact us with some details about the license, …

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

Software stack on new systems

- CVMFS: CERN Virtual Machine File System.

  All software installed on cvmfs is mounted on new systems.

- RSNT: Research Support National Team.
  - Install and Maintain the software stack on new systems.
  - Documentation.
- Free software accessible via modules.
- Commercial software protected by POSIX groups and available only for the license holders.

Using Modules in HPC

❑ Why use modules?

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

- ❖ Set some default modules:
  - ➢ Intel Compilers, GNU compilers, Other applications.
  - ➢ Open MPI, …

Modules: some commands

- module list
- module avail
- module load soft/version; module unload soft/version
- module show soft/version; module help soft/version
- module purge
- module use ~/modulefiles; module unuse ~/modulefiles

Short demonstration on Grex and Cedar or Graham

Running and submitting jobs

- Connect to a cluster: ssh, Putty, MobaXterm, X2Go.
- Prepare your input: use an editor to change your files.
- Upload or Transfer your files (if prepared elsewhere):
  scp; stfp; WinScp; FileZilla; Globus; …
- Compile your own code (ask for support if needed):
  Intel or GNU compilers; Libraries; Tools; …
- Use existing modules and programs:
- Or ask support team to install new ones if necessary
- Prepare and Test your scripts and programs.
- Read about the scheduler: what is a scheduler?

Why do we need a scheduler?

- High Performance Computing:
  - ✓ Multiple users and different Allocations: RAC; Default, …
  - ✓ Multiple applications.
  - ✓ Multiple jobs: each job has a particular set of resources.

- What happened if there is no scheduler?
  - ➢ Over usage for some resources.
  - ➢ Abuse from users: others will never get a chance to run a job.
  - ➢ Over-subscribing the resources.
  - ➢ Failure of compute nodes.
  - ➢ Efficiency of the cluster: some nodes can be over-loaded while others stay idle.

What a scheduler does?

- Put the jobs on the queue: one or more queues.
- Assign a priority to each job:
  - ✓ Resources asked for: mem, procs, nodes, wall time, …
  - ✓ Recent usage: on Grex, we consider the last three weeks.
  - ✓ RAC or Default allocation.
- Assign a status to a job: Q (queue), R (running), H (hold)
- Run the job when the resources are available.
- Report some stats about the jobs: mem, ...
- Remove the job from the queue when it is done or exceeded the wall time.

Which scheduler to use?

- **Grex:** Torque and Moab (PBS scripts)
- **Cedar, Graham and Beluga:** SLURM

- There are other schedulers.
- Each system has one scheduler: consult the Guide Start
- Read the documentation about the scheduler before starting any job.

- Even if the schedulers are different, the idea is the same:
  - ✓ Submit and run jobs.
  - ✓ Each scheduler has its own syntax.

Structure of a PBS script

```
#!/bin/bash

#PBS -S /bin/bash
Add the resources and some options

cd ${PBS_O_WORKDIR}
echo "Current working directory is `pwd`"
echo "Starting run at: `date`"

Load appropriate modules.
command line to run your program.

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

File:
test_job.pbs

To adjust for each type of job to submit

Submit the job using the command: qsub test_job.pbs

Most used PBS commands

PBS command	Description
#PBS -S /bin/bash	Set the shell that the job will be executed on compute node.
#PBS -l walltime=12:00:00	Wall time of 12 hours (max of 240 hours).
#PBS -l procs=1 #PBS -l nodes=1:ppn=1	Requests one core (serial job).
#PBS -l procs=48	Request 48 cores: MPI job (OpenMPI).
#PBS -l mem=4000mb	Total memory asked for.
#PBS -l pmem=2500mb	Memory per core (Total mem = pmem * Nc).
#PBS -l nodes=1:ppn=12	Multi-threading (OpenMP) programs.
#PBS -l nodes=4:ppn=12	4 nodes and 12 cores per node = 48 cores.

Most used PBS commands

PBS command	Description
#PBS -M <valid email>	Sets the email address for sending notifications about your job state.
#PBS -m abe	Sets the scheduling system to send you email: a: when the job is aborted by the batch system. B: when the job begins execution. e: when the job terminates.
#PBS -A <Acc. group>	Requests to use a specific accounting group
#PBS -o diffuse_job.out #PBS -e diffuse_job.err	standard error and output streams are directed to two separate files
#PBS -N JOB_NAME	Job name.

https://www.westgrid.ca/support/running_jobs

PBS environment variables

PBS command	Description
PBS_O_WORKDIR	Job's submission directory
PBS_O_HOME	Home directory of submitting user
PBS_JOBID	Unique PBS job id
PBS_NUM_NODES	Number of nodes allocated to the job
PBS_NUM_PPN	Number of procs per node allocated to the job
PBS_NODEFILE	File containing line delimited list on nodes allocated to the job
PBS_TASKNUM	Number of tasks requested
PBS_JOBNAME	User specified job name

https://www.westgrid.ca/support/running_jobs

How to monitor your jobs?

PBS command	Description
qstat -a	List all jobs on the system and their state.
qstat -r	List all running jobs on the system.
qstat –u $USER	List all the jobs under user's account.
showq -r	Show all running jobs.
showq –r –u $USER	Show all running jobs under user's account.
qstat –f <job_id>	List detailed information on Job <job_id>
checkjob <job_id>	List detailed information on Job <job_id>
jobinfo –u $USER	Stats about CPU usage of the user's group.

Short demonstration on Grex

SLURM on new systems

```
#!/bin/bash

#SBATCH --account=def-some_user

Add the resources and some options

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

Load appropriate modules.
command line to run your program.

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

File:
test_job.slurm

To adjust for each type of job to submit

Submit the job using the command: sbatch test_job.slurm

SLURM: some commands

SLURM command	Description
#SBATCH --account=def-some_user	Use the accounting group for jobs.
#SBATCH --time=0-00:05	Wall time in the format: DD-HH:MM
#SBATCH --ntasks=4	Requests 4 cores for MPI job
#SBATCH --nodes=2 #SBATCH –ntasks-per-node=32	Requests 2 whole nodes 32 (graham); 40 (beluga), 48 (cedar)
#SBATCH --mem=4000	Memory of 4GB for the job
#SBATCH --mem-per-cpu=2000M	Memory of 2GB per CPU
#SBATCH --cpus-per-task=4	Number of threads (OpenMP)
#SBATCH --job-name="JOB_NAME"	Job name.
#SBATCH --output=job_output.txt	Standard output.

Monitor your jobs with SLURM

```
squeue -u $USER [-t RUNNING] [-t PENDING]          # list all current jobs.
squeue -p partitionName                            # 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
scontrol show job jobID                            # produce a very detailed 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                                # show idle node(s) on cluster.
scancel [-t PENDING] [-u $USER] [jobID]            # kill/cancel jobs.
```

Torque vs SLURM: serial job

File: serial_job.pbs

```
#!/bin/bash
#PBS -S /bin/bash
#PBS -l procs=1
#PBS -l mem=2500mb
#PBS -l walltime=24:30:00

cd $PBS_O_WORKDIR
echo "Starting run at: `date`"
module load lammps/11aug17
lmp_grex < in.lammps
echo “Program finished at: `date`"
```

qsub serial_job.pbs

File: serial_job.slurm

```
#!/bin/bash

#SBATCH --ntasks=1
#SBATCH --mem=2500M
#SBATCH --time=1-00:30

echo "Starting run at: `date`"
module load lammps/20170331
lmp < in.lammps
echo “Program finished at: `date`"
```

sbatch serial_job.slurm

Torque vs SLURM: OpenMP job

File: omp_job.pbs

```
#!/bin/bash
#PBS -S /bin/bash
#PBS -l nodes=1:ppn=8
#PBS -l pmem=2500mb
#PBS -l walltime=24:30:00
cd $PBS_O_WORKDIR
echo "Starting run at: `date`"
export OMP_NUM_THREADS=8
module load lammps/11aug17
lmp_grex < in.lammps
echo “Program finished at: `date`"
```

qsub omp_job.pbs

File: omp_job.slurm

```
#!/bin/bash

#SBATCH --cpus-per-tasks=8
#SBATCH --mem-per-cpu=2500M
#SBATCH --time=1-00:30

echo "Starting run at: `date`"
export OMP_NUM_THREADS=8
module load lammps/20170331
lmp < in.lammps
echo “Program finished at: `date`"
```

sbatch omp_job.slurm

Torque vs SLURM: MPI job

File: mpi_job.pbs

```
#!/bin/bash
#PBS -S /bin/bash
#PBS -l procs=16
#PBS -l pmem=2500mb
#PBS -l walltime=12:30:00

cd $PBS_O_WORKDIR
echo "Starting run at: `date`"
module load lammps/11aug17
mpiexec lmp_grex < in.lammps
echo “Program finished at: `date`"
```

qsub mpi_job.pbs

File: mpi_job.slurm

```
#!/bin/bash

#SBATCH -ntasks=16
#SBATCH --mem-per-cpu=2500M
#SBATCH --time=0-12:30

echo "Starting run at: `date`"
module load lammps/20170331
srun lmp < in.lammps
echo “Program finished at: `date`"
```

sbatch mpi_job.slurm

Torque vs SLURM: GPU job

File: gpu_job.pbs

```
#!/bin/bash
#PBS -S /bin/bash
#PBS -l node=1:ppn=12:gpus=3
#PBS -l mem=2500mb
#PBS -l walltime=24:30:00
cd $PBS_O_WORKDIR
echo "Starting run at: `date`"
module load cuda
module load [some module]
gpu_program
echo “Program finished at: `date`"
```

qsub gpu_job.pbs

File: gpu_job.slurm

```
#!/bin/bash
#SBATCH --nodes=3
#SBATCH --gres=gpu:1
#SBATCH --mem=4000M
#SBATCH --time=1-00:30

echo "Starting run at: `date`"
module load cuda
module load [some module]
gpu_program
echo “Program finished at: `date`"
```

sbatch gpu_job.slurm

Useful Links

- WestGrid: https://www.westgrid.ca/
- Grex: https://www.westgrid.ca/support/quickstart/Grex
- WG Start Guide: https://www.westgrid.ca/support/quickstart/new_users
- Software on WG: https://www.westgrid.ca/support/software
- Compute Canada: https://docs.computecanada.ca/wiki/Compute_Canada_Documentation
- CCDB: https://ccdb.computecanada.ca/security/login
- CC Software: https://docs.computecanada.ca/wiki/Available_software
- 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
- Help and support on CC: support@computecanada.ca
- WG training material: https://westgrid.github.io/trainingMaterials/