SHARCNET™

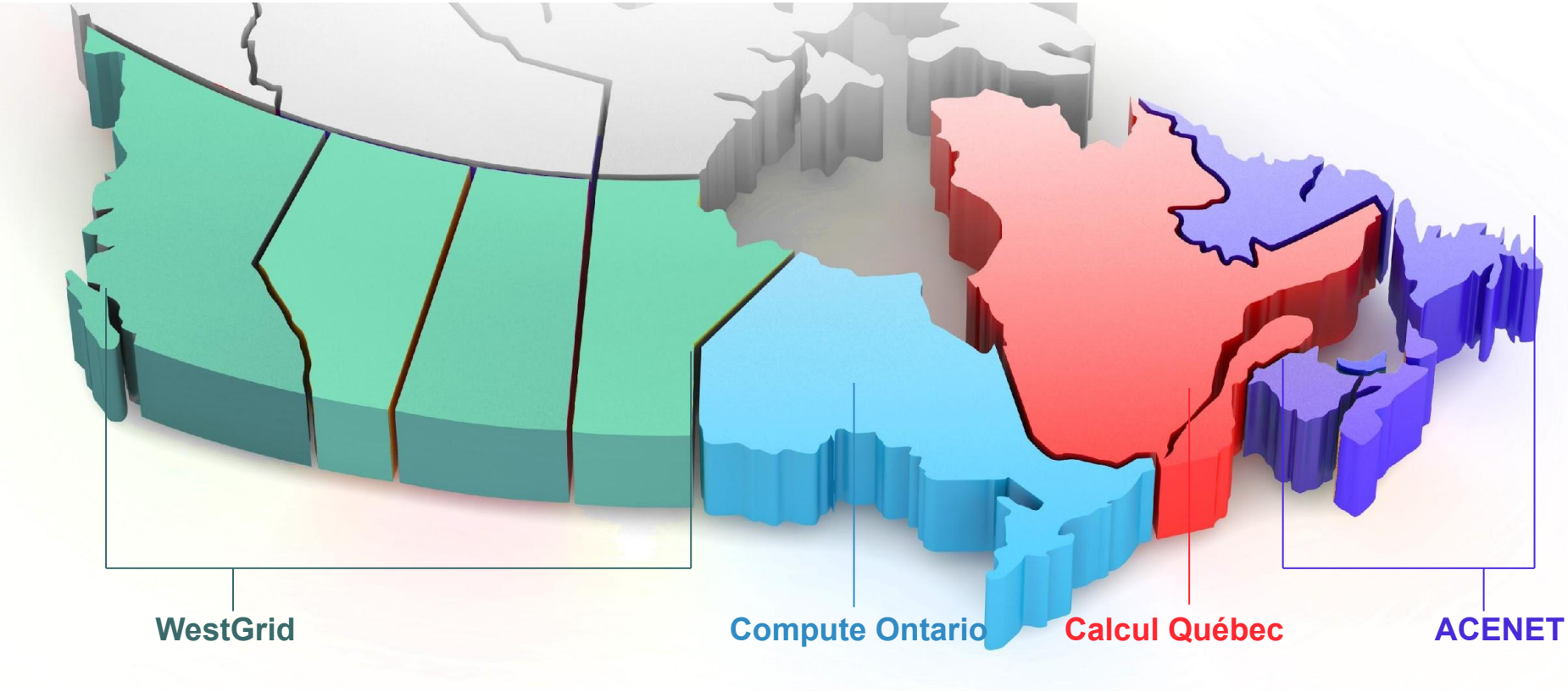
computecanada

New User Seminar

SHARCNET™

Single account...

computecanada



One can access all national supercomputers across the country, for free.

SHARCNET™

A consortium of 19 Ontario institutions providing advanced computing resources and support...

Shared
Hierarchical
Academic
Research
Computing
NETwork

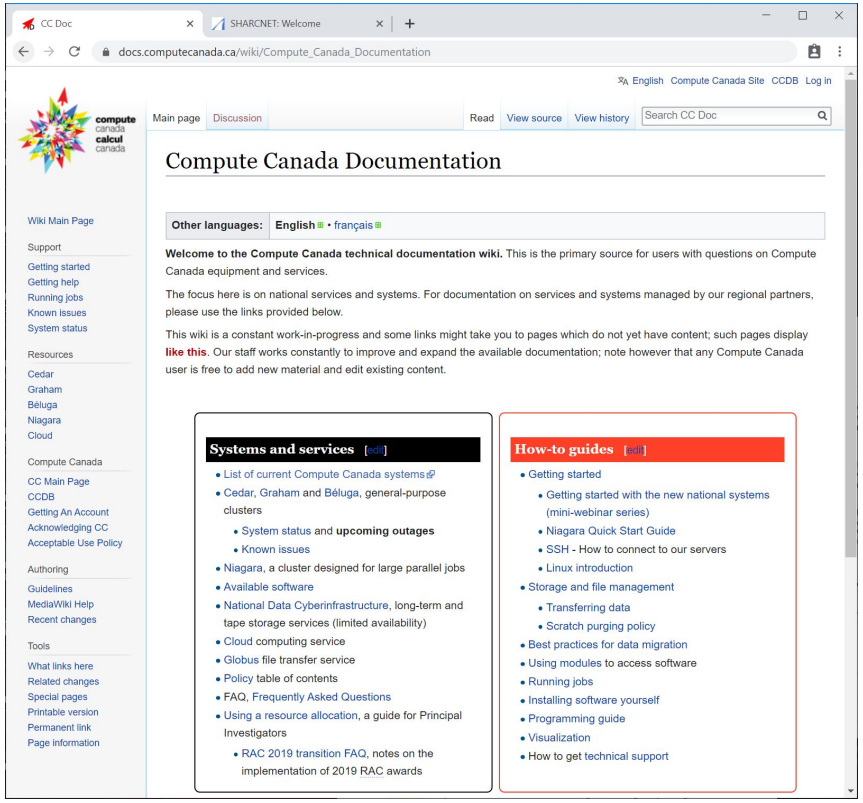
computecanada

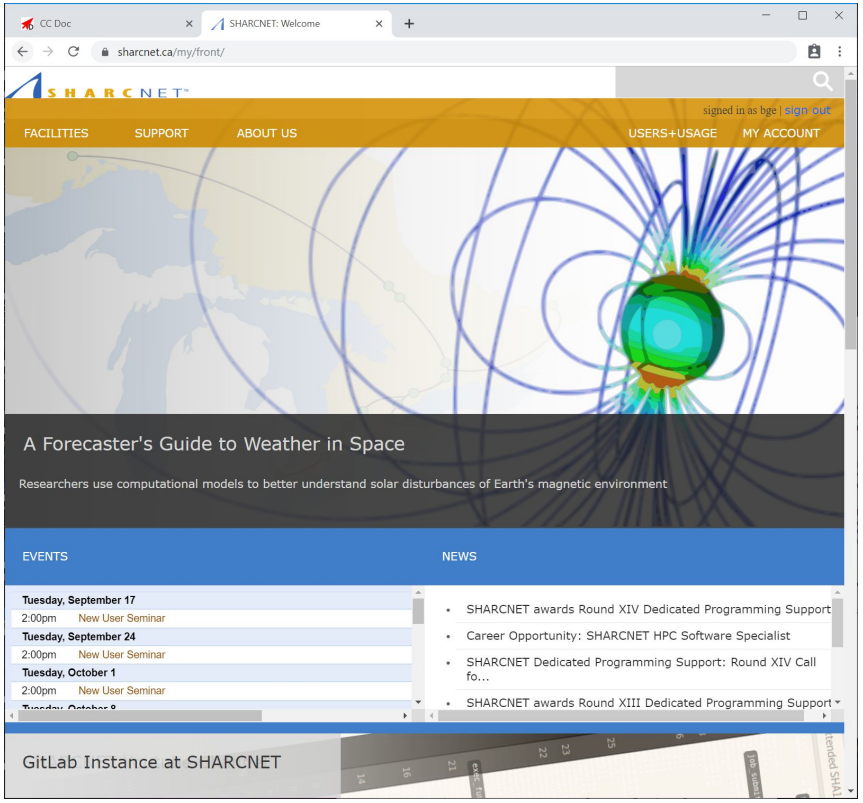
- Member of Compute Canada and Compute Ontario
- 3,000+ Canadian and international users

- ~50,000 CPU cores
- 370+ GPUs
- 10 Gb/s network
- 100 Gb/s between nat'l centres

Where to look for information and get help

Online documentations





https://docs.computecanada.ca/ https://www.sharcnet.ca/

Where to look for information and get help

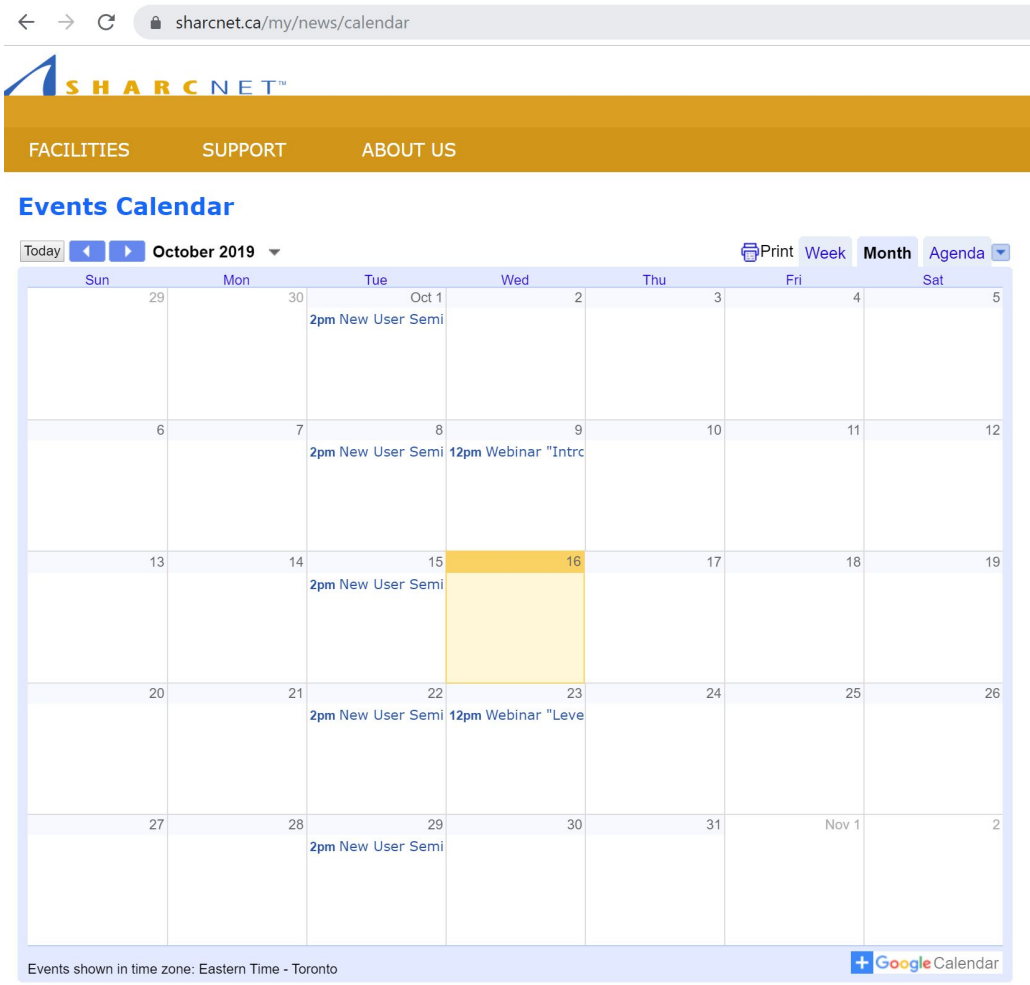
Youtube videos - search for SHARCNET channel

- http://youtube.sharcnet.ca
- Two-minute short videos on a variety of simple topics
- Programming clusters with/without MPI
- Parallel programming using threads
- Programming GPUs
- Understanding how scheduler works
- Visualization of scientific data
- Tools for bioinformatics
- Tools and platforms for data science
- Popular programming languages such as C/C++, Fortran, Matlab/Octave, R, Python, Julia, CUDA, etc.

Where to look for information and get help

Online events

- New user seminar every Tuesday at 2pm.
- Bi-weekly general interest seminars at noon on Wednesday.
- @SHARCNET



Where to look for information and get help

Interactive help

- Ticketing system via support@computecanada.ca or help@sharcnet.ca
- E-mail us - check staff contact info on https://www.sharcnet.ca/
- Phone us
- Office visit

Use of systems
Installation of software
Access to commercial software and site licence
Debugging and optimizing code
Programming
Consultation on various research problems
Grant application for compute hardware
...

The computing environment

Clusters across the country

- cedar.computecanada.ca
- graham.sharcnet.ca
- niagara.computecanada.ca
- beluga.computecanada.ca

Cloud services

- arbutus.cloud.computecanada.ca
- graham.cloud.comptecanada.ca
- east.cloud.computecanada.ca

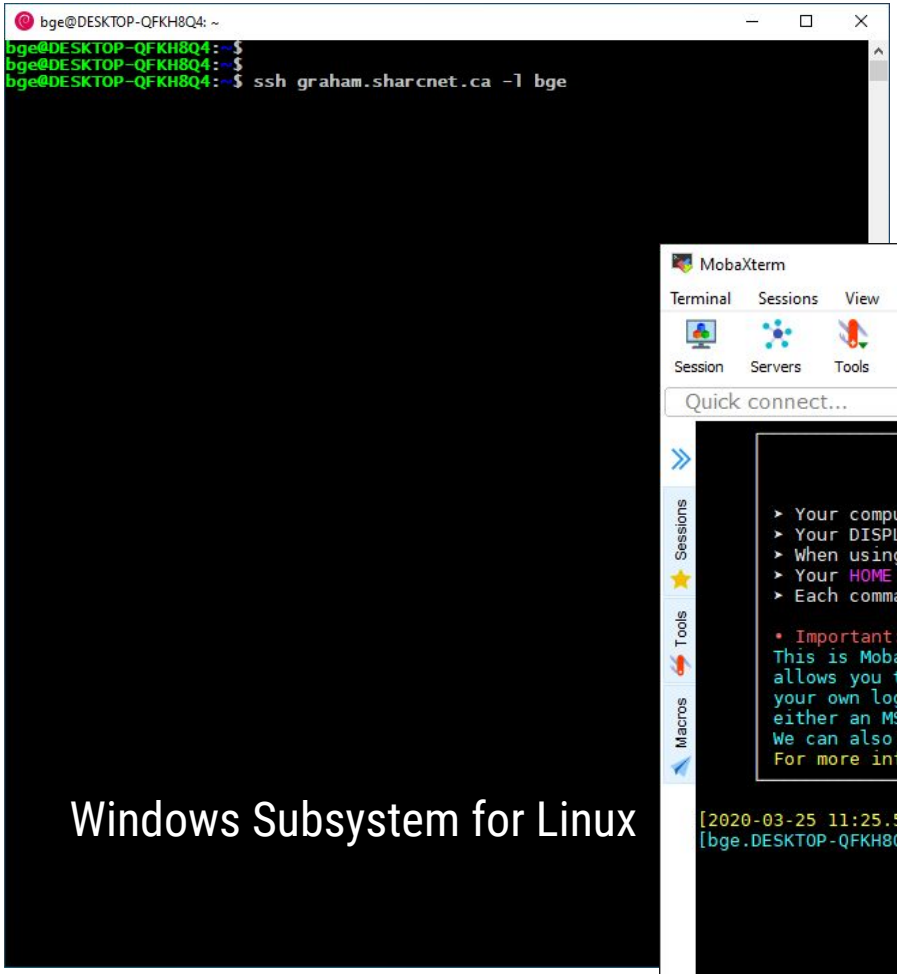
NB. Files are NOT shared across systems. You need move and copy files around as needed.

The computing environment

Cluster computing environment

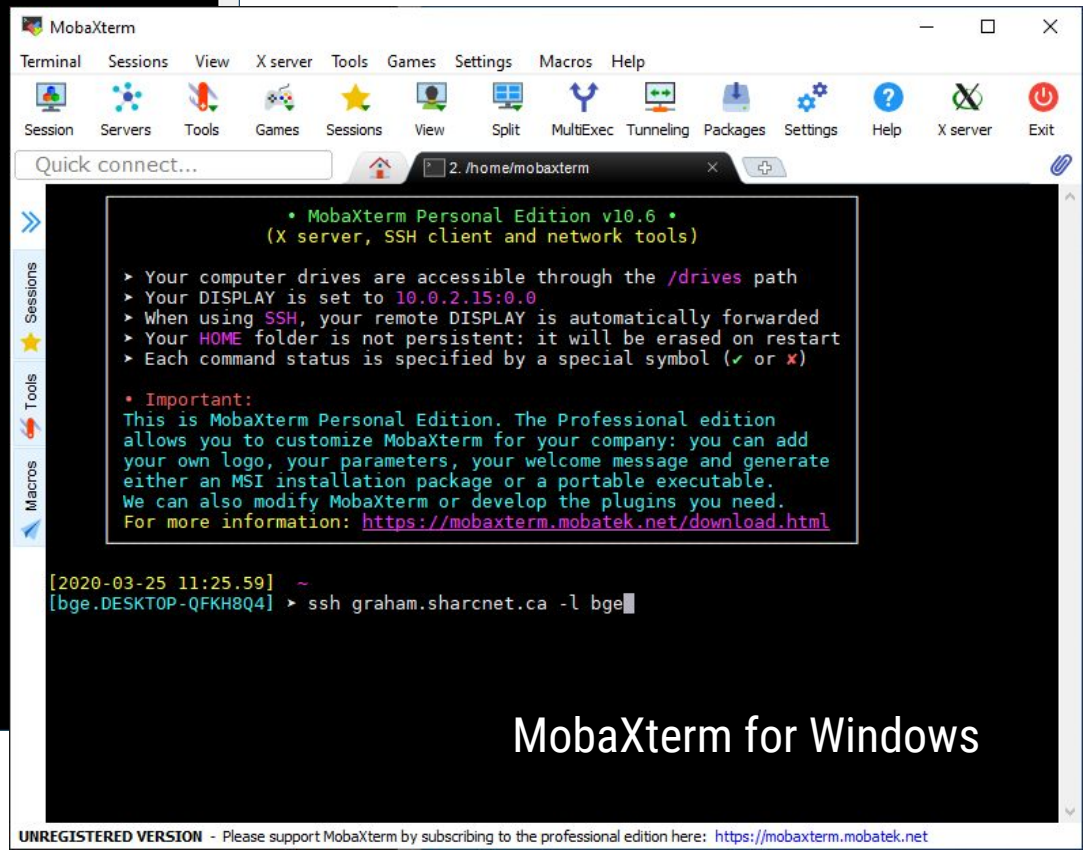
- OS: 64-bit Linux - CentOS
- Languages: C/C++, Fortran, R, Python, Matlab/Octave, Java, Julia, CUDA, etc.
- Access to a variety of software packages
- Parallel development support:
  - MPI, OpenMP, Pthreads, CUDA, OpenACC, OpenCL, DDT
  - C++: Language support for multithreading (since C++-11 standard)
  - Fortran: Language support for parallel programming (since 2003 standard)
  - Julia: Parallel processing constructs, shared and distributed objects
- Data science support:
  - R, Python, Julia, Spark, DASK, etc.
- Batch computing via slurm

Connecting to clusters via SSH



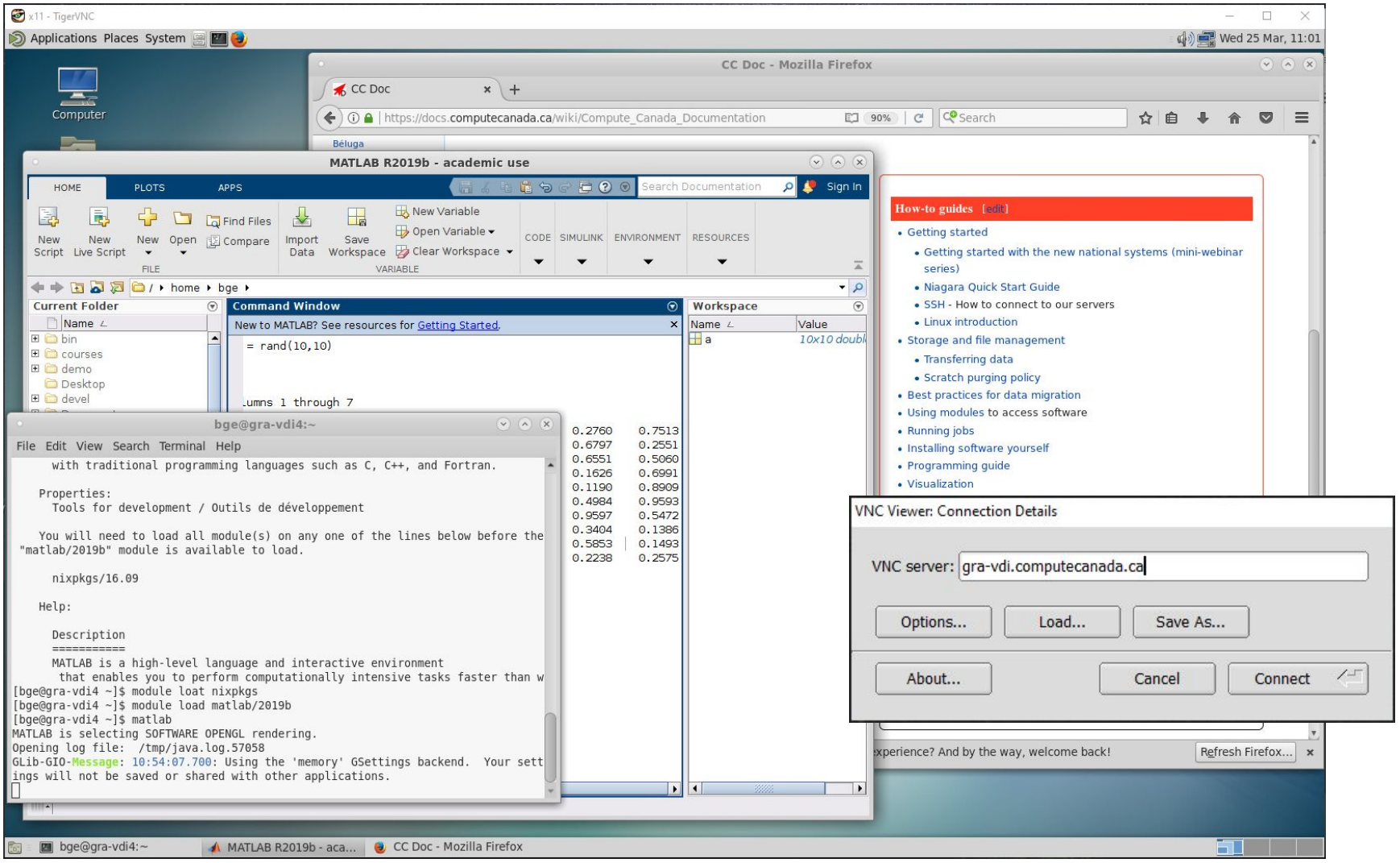
Windows Subsystem for Linux

For Windows users we recommend



MobaXterm for Windows

Connecting to graham virtual desktop via VNC



Accessing and managing files

File systems

- /**home**: 50G, 0.5m files; backed up regularly
- /**project**: 1T per group, 0.5m files, upto 10T per group; backed up.
- /**scratch**: 20T per user, 1m files, upto 100T; 2 months of life.
- **nearline**: to store files not currently in use, but may be needed later. 5T per group. NOT available on compute nodes.

*NB: Please **DO NOT** store everything, remove the files no longer in use to save space.*

How to access

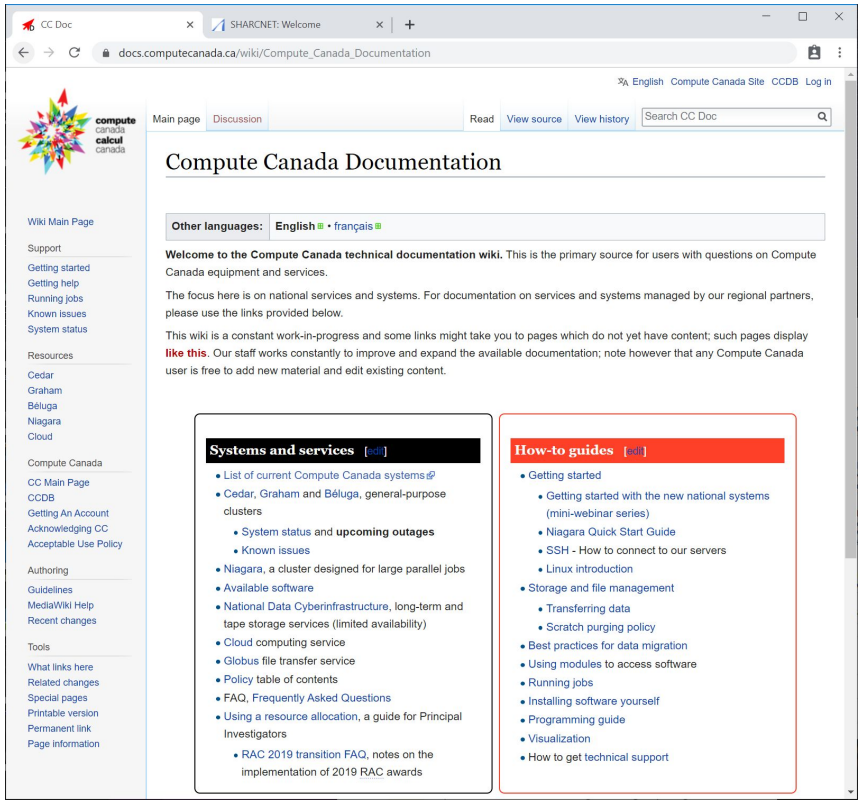
- SSH connect to a cluster and access files there.
- Mount remote cluster file systems on your laptop, desktop and access files as if they were local. See Youtube videos on how to mount cluster file system with SSHFS. YouTube CA

Moving files to and from

- Via SSHFS mounted drives (as if local) YouTube CA
- scp, sftp.
- globus via web browser.

What software packages are available?

Check for software on the web site



Check for software while on a cluster

$ **module avail**

$ **module spider** *keyword*

Running jobs using a slurm script - *myjob*.sh

Submitting a serial job

```
#!/bin/bash
#SBATCH --time=01:00 # DD-HH:MM
#SBATCH --account=def-user

R --no-save --args input 23 output < sample.R
```

To see what account groups you have access to, use command **sshare -U** or **salloc** by itself

sbatch *myjob*.sh

Submitting a series of jobs

```
#!/bin/bash
#SBATCH --time=01:00
#SBATCH --account=def-user
#SBATCH --array=1-200

R --no-save --args input $SLURM_ARRAY_TASK_ID \
output < sample.R
```

In this example, slurm generate 200 integers from 1 to 200 and stores in the environment variable $SLURM_ARRAY_TASK_ID one at a time.

The R script sample.R reads 3 args: “input”, an integer ***N*** ranging from 1 to 200 and “output”. It reads input from **input*N*.csv** and writes results to output **output*N*.csv** for each of the 200 cases.

Running jobs using a slurm script - *myjob*.sh

Submitting a threaded job

```
#!/bin/bash
#SBATCH --account=def-user
#SBATCH --time=0-03:00
#SBATCH --cpus-per-task=32
#SBATCH --ntasks=1
#SBATCH --mem=20G

export \
  OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

./myprog.exe
```

sbatch *myjob*.sh

Submitting a parallel job

```
#!/bin/bash
#SBATCH --account=def-user
#SBATCH --time=5-00:00
#SBATCH --ntasks=32
#SBATCH --mem-per-cpu=4G

srun ./mympiprog.exe
```

Why my jobs don't start

- Your jobs are waiting because of past usage. Everyone has a share. The share usage is completely "reset" in 2 weeks but after 1 week the usage is halved so some of your jobs may start then.
- Requesting much more resources (runtime, CPU cores, memory) than what is actually needed will result in a longer queue wait time, for no good reason. ***Tip: request only what the job needs, with a bit of leeway for time and memory***.
- You may try other clusters such as beluga, cedar and niagara.
- If your job can efficiently use multiples of 32 cpu cores, it gains access to a larger set of nodes, so may wait less than a by-core job. ***Tip: use --nodes=N and --ntasks-per-node=32 sbatch arguments to request by-node allocation***.

Common mistakes to avoid

- Do not run significant programs on login nodes, nor run programs directly on compute nodes.
- Do not specify a maximum job run time blindly (say, 7 days), or more memory than required for your program
  - pick an appropriate value, eg. 130% of the measured/expected run time or memory per processor
- Do not create millions of tiny files, or large amounts (> GB) of uncompressed (eg. ASCII) output
  - aggregate files with tar, use binary or compressed file formats

Q&A