WESTGRID

compute canada | calcul canada

Software on HPC clusters

UofM-Autumn-Workshop 2021

Novemner 1st-2nd, 2021

Grigory Shamov

(with slides by Dr. Ali Kerrache)

University of Manitoba

University of Manitoba

Outline

- Software distribution on HPC clusters
  - Env. modules, Why modules? How to find software?
  - Local and CC Software stacks on Grex, CVMFS
- How to build software from sources
  - Autotools and CMake
  - Compiling best practices, optimizations for CPUs
- How to deal with interpreted languages/libraries
  - R packages
  - Python modules
- Containers in HPC, : Singularity and others

University of Manitoba

Software distribution

Operating system package managers / repos:

- ★ **Ubuntu:** ~$ ***sudo apt-get install <package>***
- ★ **CentOS:** ~$ ***sudo yum install <package>***
- ★ **On HPC:** users do not have **sudo**!

Using a centralized HPC software stack:

- ★ **Software distributed via CVMFS:** CC software stack (CC clusters), …
- ★ **Local software:** legally restricted software (VASP, Gaussian, …)

Local installation: usually to $HOME or $PROJECT

- ★ **Get the code:** download the sources/binaries: wget, git clone, … etc.
- ★ **Settings:** load dependencies, set environment variables, … etc.
- ★ **Build:** ./configure {cmake ..} +opts; make; make test {check}; make install

University of Manitoba

Why modules?

- **Modules were initially developed at OSC, in TCL , then Lua modules (Lmod) at TACC.**
  - Most HPC centres use either TCL Modules or Lmod. CC & Grex are 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.
- **How it works?**
  - **module** commands dynamically change the Environment
  - Variables like PATH, LD_LIBRARY_PATH are appended or prepended
  - Variables like GAUSS_EXEDIR or ANSYS_HOME are set and unset

University of Manitoba

Why modules?

★ **Most frequently used Lmod module commands :**

- module **list**; module **avail** **# shows loaded and available items**
- module **spider** ; module **spider** <soft>/<version> **# deep search**
- module **show** soft/version; **# shows what module does**
- module **help** <soft>/<version>; module **whatis** <soft>/<version>
- module **load** soft/version; module **unload {rm}** <soft>/<version>
- module **purge**; module --force **purge** **# mass unload**
- module **use** ~/modulefiles; module **unuse** ~/modulefiles

University of Manitoba

Why Lmod modules?

- **Lmod supports a "hierarchical module structure"**
  - As modules grow in number, complexity rises. So one can end up having conflicts between modules and their dependencies..
  - In the Hierarchical structure, module form a dynamic tree structure based on their dependencies (toolchains like Compiler, MPI, CUDA kinds)
- **How it works?**
  - **module** commands dynamically change the MODULEPATH Environment
  - Each toolchain has its own module path , preventing cross-toolchain loads
  - **module avail** only uses the current toolchain's MODULEPATH.
  - A new command **module spider** was added to Lmod to search across all of the hierarchy
  - Switching toolchain components causes automatic module reloads, if works

University of Manitoba

Modules on Grex

- About 450 modules:
  - GCC [5,7,9]; Intel [2014 - 2020].
  - Libraries: HDF5, PETSc, GSL, MKL, Libxc, Boost, ...
  - Gaussian, ANSYS, MATLAB, VASP, MCR, Java, … etc.
  - LAMMPS, GROMACS, ABINIT, QE, VMD, Molden, … etc.
- Software maintenance on Grex:
  - We install programs on request from users.
  - Search for a program using “module spider <name of your program>”
  - If not installed, ask for support “support@computecanada.ca”
  - We will install the module or update the version.
  - For commercial software, contact us before you purchase the code:
    - to check license type.
    - see if it will run under Linux environment, … etc.

Modules and slurm jobs

- **sbatch**, **salloc**, **srun** would propagate the environment from the submitting shell. Including loaded modules.

- But we recommend to add module commands to the job scripts explicitly, for it will be easier to track changes and troubleshoot.

University of Manitoba

Software stacks on Grex

- ★ Grex environment [default]: GrexEnv
  - no module loaded by default, except the GrexEnv
  - use module spider to search for modules
  - Compilers {GCC, Intel}, MKL, PETSc, … etc.
  - Commercial UM-licensed soft: Gaussian, ANSYS, MATLAB, … etc.
- ★ Compute Canada environment [optional]: CCEnv
  - Switch to CCEnv; load a standard environment; choose the architecture[sse3, avx2, avx512], use module spider <soft>

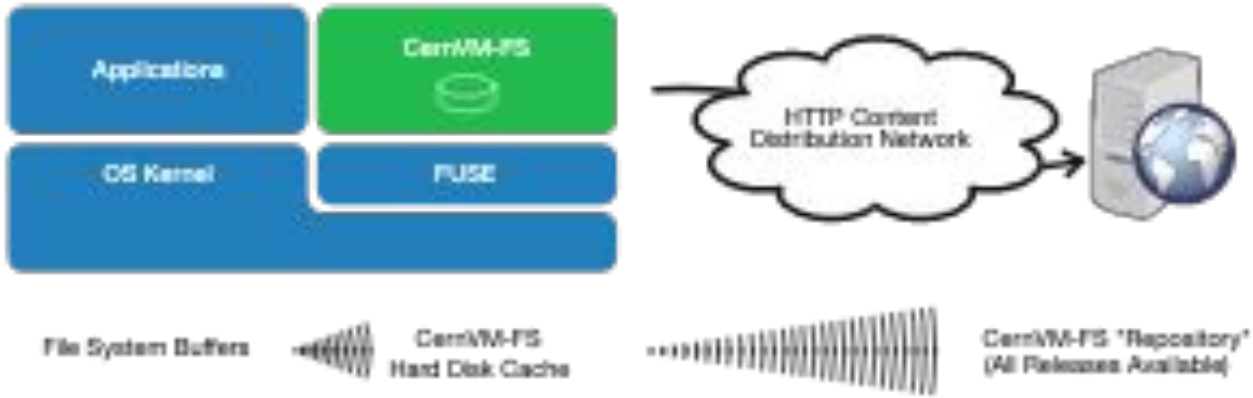
```
module load CCEnv
module load StdEnv/2016
module load arch/sse3
module load nixpkgs/16.09  gcc/5.4.0 geant4/10.05.p01
```

University of Manitoba

CVMFS Software stacks on Grex

- ★ Compute Canada environment [optional]: CCEnv
  - Tries to be independent of base OS, self contained
  - Encapsulates low level OS libs, own LMOD, and has several HPC toolchains. (GCC, Intel)
  - Commercial software on “restricted” repo
  - Innovative, a model for european stack etc.
- ★ OpenScience Grid
  - Singularity containers , or software distros
- ★ MUGIC project for Genomics

CC CVMFS Software Stack



Picture from https://cvmfs.readthedocs.io/en/stable/cpt-overview.html

- CVMFS filesystem for distribution (CC maintains Stratum 0, Stratum 1)
- A base-OS layer included in the distribution and makes the stack almost self-contained software
- HPC software stack is managed by Lmod and Easybuild

University of Manitoba

CC CVMFS Software Stack

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)

Nix or gentoo layers: GNU libc, autotools, make, bash, cat, ls, awk, grep, etc.

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

OS: kernel, daemons, drivers, libcuda, anything privileged (e.g. the sudo command): always local.

University of Manitoba

CC CVMFS Local installations

- ★ R packages: ComputeCanada provide a minimal installation of:
  - R as modules: users can install the packages in their home directory.
- ★ Python as modules: python and scipy-stack
  - users can install the packages needed in their home directory.
  - Wheels for many popular packages are provided on CC CVMFS.
- ★ Perl and bioperl as modules:
  - users can install the packages needed in their home directory.
- ★ Other software installed locally:
  - Home made programs
  - Restricted and licensed software that can not be distributed via CVMFS.
  - Custom software: patch from a user, changing parts of the code, … etc.

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

University of Manitoba

Local software installations: when?

- Local installation (user's directory):
  - R packages
  - Python packages: virtual environment, conda
  - Perl modules
- Installation from sources with:
  - make; make test {check}; make install
  - Autotools: configure; make test {check}; make install
  - CMake: ccmake; make test {check}; make install
- Java applications: jar files
- Singularity:
  - build the image and run the program from the container

University of Manitoba

The question for local installs:

- ★ Are parts/dependences already present on the systems?
- ★ Before building every dependency, check if they already are present
  - Especially low-level libraries like MPI, BLAS/LAPACK/FFT
  - MPI might have non-obvious choices for particular hardware
  - Follow the stack: HPC machine provides base toolchain (Compilers, MPI, CUDA, HDF5/NetCDF, etc.);
  - Dont try re-doing everything; instead build just the top level (application, specific dependencies that are missing).
- ★ Decisions might require knowledge of the particular HPC system
  - If unsure, don’t hesitate to contact an HPC analyst

University of Manitoba

Notes on GrexEnv Toolchains

- ★ Systems compilers and interpreters like CC, CXX, F90, PYTHON etc. not necessarily are the compilers to be used.
- ★ Toolchains: we support mainly Intel (for speed) and GCC (for stable, standard compliant for C++). **Intel 2017-2020; GCC 4.8, 7.4, 9.2**
- ★ CPU architecture: -msse4.2 (old nodes), -mavx2 (new nodes)
  - For Intel, can do dispatch: **-axCORE-AVX512,CORE-AVX2,SSE4.2**
  - **-xHost** easy but dangerous (depends on which node you build!)
- ★ MessagePassingInterface (MPI) libraries:
  - OpenMPI 3.1, 4.0 are recommended (there are older vers. Like 1.6.5, we keep them for compatibilit).
  - Intel MPI 2017-2019 are also provided.

University of Manitoba

Java applications

- ★ Download and unpack the code
- ★ Load java module: module load java
- ★ Run the code

- ★ Example: Trimmomatic
  - wget http://www.usadellab.org/cms/uploads/supplementary/Trimmomatic/Trimmomatic-0.39.zip
  - unzip Trimmomatic-0.39.zip

- ★ Run the code

```
module load java
java -jar <path to>/trimmomatic-0.39.jar {+options if any}
```

University of Manitoba

Local installation: Python/Virtualenv

Example: install Numba package that requires LLVM which is not on GrexEnv

Switch to CCEnv and load Pyton module:

```
$> module load CCEnv arch/avx2 # note that this needs skylake!
$> module load StdEnv/2016.4
$> module load python
```

Create a virtual environment “numbaenv” and install Numba:

```
$>  virtualenv numbaenv
$>  source numbaenv/bin/activate
(numbaenv) $>  pip install numba
(numbaenv) $>  python -c "import numba"
```

Needs the same virtual environment activated in job scripts!

University of Manitoba

Local installation: R packages

R packages: rgdal, adegenet, stats, rjags, dplyr, … etc.

Choose your module: module spider r

Load R and dependencies (mkl, gdal, jags, gsl, udunits… etc):

```
module load gcc/7.3.0 r/3.6.0 gdal udunits…
```

Launch R and install the packages:

```
~$ R
> install.packages(”sp")
'lib =/cvmfs/soft.computecanada.ca/easybuild/{..}/R/library"' is not writable
Would you like to use a personal library instead? (yes/No/cancel) yes
Would you like to create a personal library ‘~/R/{…}’ to install packages into? (yes/No/cancel) yes
--- Please select a CRAN mirror for use in this session ---
> install.packages(”dplyr")
```

University of Manitoba

Local installation: perl

Example: Hash::Merge; Logger::Simple; MCE::Mutex; threads …

Load Perl module: module load perl

Install the the first package using cpan:

```
~$ cpan install YAML
Would you like to configure as much as possible automatically? [yes] yes
What approach do you want?  (Choose 'local::lib', 'sudo' or 'manual')
[local::lib] local::lib
Would you like me to append that to /home/$USER/.bashrc now? [yes] yes
```

Install the rest of the packages:

```
~$ cpan install Hash::Merge
~$ cpan install Logger::Simple
~$ cpan install MCE::Mutex
```

University of Manitoba

Installation with make: STAR

★ Download the code:

wget https://github.com/alexdobin/STAR/archive/refs/tags/2.7.8a.tar.gz

★ Unpack the code: tar -xvf 2.7.8a.tar.gz

★ Load GCC compiler: module load gcc

★ Compile the code:

```
cd  STAR-2.7.8a/source
make
```

★ Copy the binaries and set the path:

```
mkdir -p ~/software/star/2.7.8a/bin
cp STAR  ~/software/star/2.7.8a/bin
export PATH=$PATH:~/software/star/2.7.8a/bin
```

Adding a local module

Create a file **star-2.7.8a** under **$HOME/modulefiles** with the following:

```
#%Module1.0

module-whatis "STAR version 2.7.8a"

prepend-path PATH /home/username/software/star/2.7.8a/bin
```

$> **module use $HOME/modulefiles**
$> **module avail** # should list /home/username/modulefiles and star-2.7.8a
$> **module load star-2.7.8a** # loads the local module
$> **which STAR**

University of Manitoba

Installation with GNU Autotools

- ★ Download and unpack the code: wget, … gunzip, … etc.
- ★ Load the modules and dependencies: module load gcc ompi fftw
- ★ Configure the program
  - If configure not included, run: autoreconf -fvi [to generate it].
  - ./configure --help [to see the different options].
  - ./configure --prefix=installdir {+other options}
- ★ Compile and test:
  - make
  - make check
- ★ Install the program:
  - make install

University of Manitoba

Notes on build dependencies

- ★ Most of the time they are modules already present on the systems
- ★ Can be often autodetected in CPATH, FPATH, INCLUDE, LIBRARY_PATH. HDF5_HOME; If not , manually add the options.
- ★ Names: BLAS/LAPACK/FFTW are part of Intel MKL libraries (now OneAPI). Just load the mkl module! Intel has calculator for Include options if not autodetected.
- ★ CPU architecture: -msse4.2 (old nodes), -mavx2 (new nodes)
  - For Intel, can do dispatch: **-axCORE-AVX512,CORE-AVX2,SSE4.2**
  - **-xHost** easy but dangerous (depends on which node you build!)
  - Compatibility vs. speed. CCEnv has “modules” for **arch**/ , GrexEnv doesn’t.

University of Manitoba

Example: options for PETSc

```
./configure --with-blas-lapack-dir=$MKLROOT/lib/intel64  --prefix=${instdir}  --with-cxx-dialect=C++11
--download-scalapack=yes --download-blacs=yes  --download-superlu_dist=yes
--download-mumps=yes --download-parmetis=yes --download-metis=yes --download-spooles=yes
--download-cproto=yes --download-prometheus=yes --with-mkl_pardiso=1
--with-mkl_pardiso-dir=$MKLROOT --with-mkl-sparse-optimize=1  --with-scalar-type=complex
--with-debugging=0  --with-hdf5=yes --with-hdf5-dir=$HDF5HOME  --download-suitesparse=yes
--download-fftw=${fftsrc} --download-amd=yes --download-adifor=yes  --download-superlu=yes
--download-triangle=yes --download-generator=yes  --with-64-bit-pointers=no  --with-cc=mpicc
--CFLAGS='-O2  -msse4.2 -xSSE4.2 -mp1 -I$MKLROOT/include -mkl -fPIC '  --with-cxx='mpicxx'
--CXXFLAGS='-O2 -msse4.2 -xSSE4.2 -mp1 -I$MKLROOT/include -mkl -std=c++11 -fPIC '
--with-fc='mpif90'  --FFLAGS='-O2 -msse4.2 -xSSE4.2 -mp1 -I$MKLROOT/include -mkl -fPIC '
--with-single-library=yes --with-shared-libraries=yes  --with-shared-ld=mpicc
--sharedLibraryFlags="-fpic -mkl -fPIC"  --with-mpi=yes  --with-mpi-shared=yes --with-mpirun=mpiexec
--with-mpi-compilers=yes  --with-x=yes
```

University of Manitoba

Example with Cmake

- ★ Download and unpack the code: wget, … gunzip, … etc.
- ★ Load the modules and dependencies: module load gcc ompi fftw
- ★ Configure the program: you may need to load cmake module
  - ○ mkdir build && cd build
  - ○ cmake .. --help [to see the different options]. Or ccmake ..
  - ○ cmake .. -DCMAKE_INSTALL_PREFIX=installdir {+other options}
- ★ Compile and test:
  - ○ make
  - ○ make check; make test
- ★ Install the program:
  - ○ make install

University of Manitoba

Cmake options for GROMACS

```
module load intel/2019.5
module load ompi/3.1.4 fftw
module load cmake

cd gromacs-5.1.4; mkdir build; cd build
cmake -DCMAKE_INSTALL_PREFIX=<path to install dir>  -DBUILD_SHARED_LIBS=off
-DBUILD_TESTING=off  -DREGRESSIONTEST_DOWNLOAD=off
-DCMAKE_C_COMPILER=`which mpicc` -DCMAKE_CXX_COMPILER=`which mpicxx`
-DGMX_BUILD_OWN_FFTW=on -DGMX_SIMD=SSE4.1 -DGMX_DOUBLE=off
-DGMX_EXTERNAL_BLAS=on -DGMX_EXTERNAL_LAPACK=on
-DGMX_FFT_LIBRARY=fftw3 -DGMX_GPU=off -DGMX_MPI=on -DGMX_OPENMP=off
-DGMX_X11=on ../gromacs-5.1.4
make -j4
```

University of Manitoba

Notes on installing binaries

- ★ Usually, use tarballs rather that packages (yum, deb, etc.) as there is no sudo access.
- ★ Dependencies might be there as modules, or parts of modules (libstdc++ comes from newer GCC, etc.).
- ★ Install under $HOME, modify path
  - ○ One standard way is to create a local module
  - ○ Module use ~/modulefiles
  - ○ Module avail; module load ..
- ★ GLIBC dependency missing: no chance, recompile the code from sources or use containerized deployment.
- ★ CPU architecture is different: no chance at all. Esp. for old compute

University of Manitoba

Singularity

Resources: Github, DockerHub, SingularityHub.

Singularity examples: https://github.com/singularityware/singularity/tree/master/examples

- **Documentation:** http://singularity.lbl.gov/user-guide
- **DockerHub:** https://hub.docker.com/explore/
- **SingularityHub:** https://www.singularity-hub.org/

Access to Singularity:

★ **Connect to cluster:** Grex, cedar, graham or beluga:

★ **Load a module:** module load singularity

★ **Build the image:** convert the image from Docker to Singularity

★ **Note:** You may need to use your own Linux machine or VM to build the image

https://docs.computecanada.ca/wiki/Singularity
https://monitor.hpc.umanitoba.ca/doc/docs/grex/software/containers/

University of Manitoba

Singularity

- ★ Alternative for running software: difficult to build from source
- ★ Possibilite to convert Docker images to singularity.
- ★ Singularity installed on all clusters {no Docker for security reasons}
- ★ Build the image:

```
module load singularity
singularity build qiime2-2019.10.sif docker://qiime2/core:2019.10
```

- ★ Run the code via singularity:

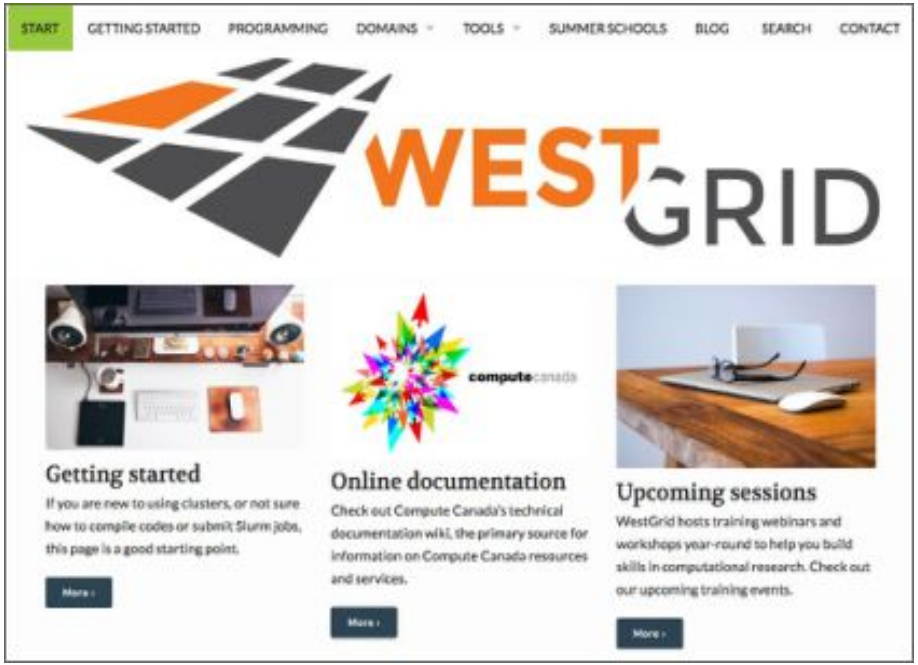
```
singularity exec -B $PWD:/home -B /global/scratch/someuser:/outputs  \
-B /global/scratch/someuser/path/to/inputs:/inputs qiime2-2019.10.sif   \
qiime feature-classifier fit-classifier-naive-bayes \
--i-reference-reads  /outputs/some_output_feature.qza \
--i-reference-taxonomy /outputs/some_output_ref-taxonomy.qza \
--o-classifier /outputs/some_output_classifier.qza
```

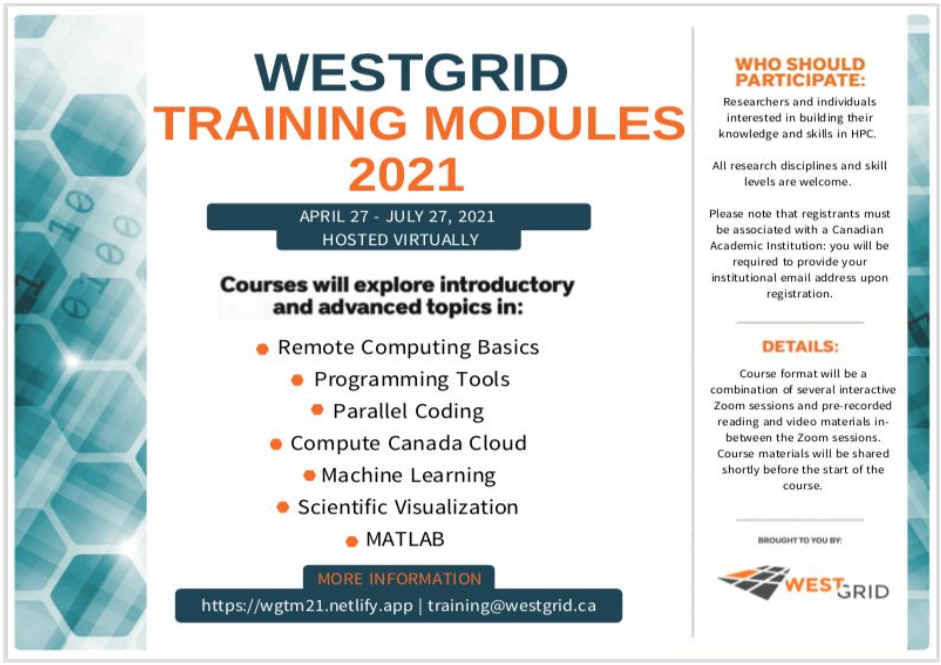
University of Manitoba

Links

- ★ FAQ: https://docs.computecanada.ca/wiki/Frequently_Asked_Questions
- ★ Jobs:
  - https://docs.computecanada.ca/wiki/Running_jobs
  - https://docs.computecanada.ca/wiki/Job_scheduling_policies#Percentage_of_the_nodes_you_have_access_to
  - https://docs.computecanada.ca/wiki/Advanced_MPI_scheduling#Whole_nodes
  - https://docs.computecanada.ca/wiki/Using_GPUs_with_Slurm
- ★ Storage:
  - https://docs.computecanada.ca/wiki/Storage_and_file_management#Filesystem_Quotas_and_Policies
  - https://docs.computecanada.ca/wiki/Project_layout?
  - https://docs.computecanada.ca/wiki/Transferring_data
- ★ Software:
  - https://docs.computecanada.ca/wiki/Available_software
  - https://docs.computecanada.ca/wiki/Utiliser_des_modules/en
  - https://docs.computecanada.ca/wiki/Installing_software_in_your_home_directory
  - https://docs.computecanada.ca/wiki/Python
  - https://docs.computecanada.ca/wiki/R

Upcoming WestGrid trainings





https://westgrid.github.io/trainingMaterials

https://www.westgrid.ca/events