



# High Performance Computing and software environments:

Install and/or use existing software and modules

*UofM-Spring-Workshop 2023*

*May 17<sup>th</sup>-19<sup>th</sup>, 2023*

*Ali Kerrache*  
*HPC Analyst*



- ★ Software distribution on HPC clusters
- ★ Why modules? How to find modules?
- ★ Software stacks on Grex
- ★ How to build software from sources
  - R packages
  - Python packages
  - Perl modules
  - configure/make
  - cmake/make
- ★ Singularity {now: **apptainer**}



## Operating system package managers / repos:

- ★ **Ubuntu:** ~\$ **sudo apt-get install <package>**
- ★ **CentOS:** ~\$ **sudo yum install <package>**
- ★ **On HPC:** users do not have **sudo!** (**NO NEED TO ASK FOR IT**)

## Using a centralized HPC software stack:

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

## Local installation: usually to \$HOME {/home/\$USER} 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



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

User

**Software stacks:** modules for Intel, PGI, OpenMPI, CUDA, MKL, high-level applications. Multiple architectures (sse3, avx, **avx2**, **avx512**)

RSNT

**Nix or gentoo:** 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.

Sys. Admin

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



# Why modules?

## ★ 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

```
[someuser@bison]$ module avail
```

```
CCEnv (S)      GrexEnv (S,L)
```

Where:

**S**: Module is Sticky, requires --force to unload or purge

**L**: Module is loaded



## ★ Grex environment [default]: GrexEnv

- no module loaded by default.
- use `module spider <name of the software>` to search for modules
- **Compilers:** {GCC, Intel}, MKL, PETSc, ... etc.
- Gaussian, ANSYS, MATLAB, ... etc.

## ★ The Alliance (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/2020`

`module load arch/avx512`

`module load StdEnv/2020 gcc/9.3.0 geant4/10.7.3`

```
module load StdEnv/2016.4
```

```
module load arch/sse3
```

```
module load nixpkgs/16.09 gcc/5.4.0 geant4/10.05.p01
```



- ★ **More than 500 modules:**
  - GCC [5,7,9,11]; 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 and update modules on request from users.
  - Search for a program using “**module spider** <name of your program>”
  - If not installed, ask for support “[support@tech.alliancecan.ca](mailto:support@tech.alliancecan.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.



```
----- /opt/lmod/grex/7.6/modulefiles/Core -----
admixture/1.3.0          intel/14.0.2.144      ncl_ncarg/6.4.0      uofm/adf/2019.305-impi
admixture/1.23          (D) intel/15.0.5.223      ncl_ncarg/6.5.0      uofm/adf/2020-impi
ant/1.10.11             intel/2017.8          nodejs/4.4.7         uofm/adf/2020.103-impi
circos/0.69-6          intel/2019.5          (D) nodejs/8.12.0       uofm/adf/2021-impi
cmake/3.12.3           intel/2020.4          nodejs/13.2.0        (D) uofm/adf/2021.102-impi
cmake/3.14.0           j/j903                openbabel/2.3.2      uofm/adf/2021.106-impi (D)
cmake/3.16.9           jags/3.4.0            openbabel/2.4.1      uofm/cfx/15.0
cmake/3.23.2          (D) jags/4.0.0           openbabel/3.0.0      uofm/cfx/16.2
cns/1.3                jags/4.3.0           (D) openbabel/3.1.1    (D) uofm/cfx/18.2
eigen/3.3.7           java/jdk7u25          ovito/2.9.0          uofm/cfx/19.2
fastqc/0.11.9         java/jdk7u45          ovito/3.0.0-dev502   (D) uofm/cfx/20.1
gaussian/g09.b01.unlim java/jdk8u5           pandoc/2.9.2.1       uofm/cfx/20.2
gaussian/g09.b01      java/jdk8u66          perl/5.14.4          (D) uofm/cfx/21.1
gaussian/g09.e01.unlim java/jdk8u92          (D) perl/5.22.1       (D) uofm/feko/2021.2
gaussian/g09.e01     java/jdk13.0.1        perl/5.28.1          uofm/gaussian/g03
gaussian/g16.b01     julia/1.3.0-bin       php/5.6.40           uofm/gaussian/g09.e01
gaussian/g16.c01.avx2.unlim julia/1.5.4-bin      php/7.3.12           (D) uofm/gaussian/g09 (D)
gaussian/g16.c01.avx2 julia/1.6.1-bin       python/2.7.12-miniconda uofm/mathematica/11.0
gaussian/g16.c01     (D) julia/1.7.0-bin    python/3.6-miniconda (D) uofm/matlab/R2014A
gcc/4.8                libccerf/1.4          settarg              uofm/matlab/R2015B
gcc/5.2                lmod                  singularity/3.5.2    uofm/matlab/R2017A
gcc/7.4                (D) ls-prepost/4.7.13  smrtlink/6.0.0.47841 uofm/matlab/R2019B
gcc/9.2                mcr/mcr               stata/14.2-fagfs     uofm/matlab/R2020B2 (D)
gcc/11.2              mkl/10.3.11          (D) stata/15.0-fagfs   (D) uofm/starccm/16.06.010
git-lfs/3.2.0         mkl/11.1.0           tbb/14               uofm/starccm/17.02.008-R8 (D)
git/2.21.0            mkl/2019.5           tbb/2019.5          (D) uofm/umcfid/2.4
gnuplot/5.2.7         molden/5.9            trimmomatic/0.39     vina/1.1.2
go/1.10.4             multiwfn/3.8-gui      uofm/adf/2016-impi-test vmd/1.9.3
go/1.11.5             multiwfn/3.8-nogui   (D) uofm/adf/2016-impi vncworkspace/1.1
go/1.12.12            nbo/6.0              uofm/adf/2017-impi   vtune/2019.4
go/1.13               nbo/7.0              (D) uofm/adf/2017.114-impi vtune/2019.5 (D)
go/1.13.3             (D) ncl_ncarg/6.2.1   (D) uofm/adf/2018dev-impi wine/3.0
intel/12.1.5.339      ncl_ncarg/6.3.0      uofm/adf/2019-impi  xtb/6.5.0-bin
----- /opt/lmod/stacks -----
CCEnv ($) GrexEnv ($,L)
```

- module avail
- module spider python
- module spider java
- module load gcc ompi
- module avail
- module spider <soft>
- module spider <soft>/<ver>
- module show <soft>
- module purge

If not available:  
 → contact support  
[support@tech.alliancecan.ca](mailto:support@tech.alliancecan.ca)



# Find and load Gaussian

**Gaussian:** restricted software; requires a registration

<https://docs.alliancecan.ca/wiki/Gaussian>

<https://um-grex.github.io/grex-docs/docs/grex/software/specific/gaussian/>

`[someuser@bison ]$ module spider gaussian`

---

`gaussian:`

---

**Versions:**

`gaussian/g09.e01`

`gaussian/g16.b01`

`gaussian/g16.c01.avx2`

`gaussian/g16.c01`

```
[someuser@tatanka ~]$ module load gaussian/g16.c01
Loading Gaussian version 16.c01
```

**Available:** Grex, Cedar and Graham

---

For detailed information about a specific "gaussian" 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 gaussian/g16.c01`

---

# Find and load ORCA

## ORCA:

- restricted software
- requires a registration

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

<https://um-grex.github.io/grex-docs/docs/grex/software/specific/orca/>

`[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`

```
Available: Grex and all Alliance clusters
```

# Find and load LAMMPS

```
[someuser@bison ]$ module spider lammmps
```

```
lammmps:
```

```
[someuser@bison ]$ module spider lammmps/29Sep21
```

## Versions:

```
lammmps/5Jun19
```

```
lammmps/5Nov16
```

```
lammmps/11Aug17
```

```
lammmps/23Jun22
```

```
lammmps/29Sep21
```

```
lammmps/30jul16
```

```
[~@tatanka ~]$ module load intel/2019.5 ompi/3.1.4 lammmps/29Sep21
```

Lmod is automatically replacing "gcc/7.4" with "intel/2019.5".

```
Unloading module: gcc/7.4.0
```

```
Loading module: LAMMPS/29Sep21
```

For detailed information about a specific "lammmps" 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 lammmps/29Sep21
```

```
[someuser@bison ~]$ module spider matlab
```

---

```
uofm/matlab:
```

---

**Versions:**

```
uofm/matlab/R2014A  
uofm/matlab/R2015B  
uofm/matlab/R2017A  
uofm/matlab/R2019B  
uofm/matlab/R2020B2
```

```
[someuser@bison ~]$ module spider mcr
```

---

```
mcr: mcr/mcr
```

---

This module can be loaded directly: `module load mcr/mcr`

---

For detailed information about a specific "uofm/matlab" 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 uofm/matlab/R2020B2
```

---



```

----- MPI-dependent avx512 modules -----
abinit/9.2.2      (chem)  lammps-omp/20220623      (chem, D)  plumed/2.6.2      (chem)
abinit/9.6.2      (chem, D)  latte/1.2.1             (chem)     plumed/2.7.0      (chem)
adol-c/2.7.2      (chem)    libcf/1.0.3             (chem)     plumed/2.7.1      (chem)
apbs/1.3          (chem)    libgridxc-mpi/0.8.0     (chem)     plumed/2.7.3      (chem)
berkeleygw/2.1.0 (phys)    met/9.1.1               (phys)     plumed/2.7.4      (chem)
berkeleygw/3.0.1 (phys, D)  mpas/7.0                (chem, D)  plumed/2.8.1      (chem, D)
bigdft/1.8.3     (chem)    mpi4py/3.0.3            (t)        pnetcdf/1.9.0     (io)
boost-mpi/1.72.0 (t)       mpi4py/3.1.2            (t)        pnetcdf/1.10.0    (io)
boost-mpi/1.80.0 (t, D)    mpi4py/3.1.3           (t, D)     pnetcdf/1.12.2    (io, D)
cdo/1.9.8        (geo)     mrbayes/3.2.7           (bio)      psi4/1.3.2         (chem)
cdo/2.0.4        (geo)     mscg/1.7.3.1           (chem)     psi4/1.4           (chem)
cdo/2.0.5        (geo, D)  mumps-metis/5.2.1      (t)        psi4/1.5           (chem, D)
cfour-mpi/2.1    (chem)    mumps-parmetis/5.3.5   (t)        quantumpresso/6.5 (chem)
cgns/3.4.1       (phys)    ncl/6.6.2              (vis)      quantumpresso/6.6 (chem)
cgns/4.1.0       (phys)    ncview/2.1.8           (vis)      quantumpresso/6.7 (chem)
cgns/4.1.2       (phys, D)  nektar++/5.0.1         (math)     quantumpresso/6.8 (chem)
comblblas/1.6.2 (chem)    netcdf-c++-mpi/4.2     (io)       quantumpresso/7.0 (chem, D)
cp2k/7.1         (chem)    netcdf-c++4-mpi/4.3.1 (io)       raxml/8.2.12      (bio)
cp2k/8.2         (chem)    netcdf-fortran-mpi/4.5.2 (io)       ray/3.0.1         (bio)
cp2k/9.1         (chem, D)  netcdf-fortran-mpi/4.6.0 (io, D)    repasthpc/2.2.0   (bio)
cpmd/4.3         (chem)    netcdf-mpi/4.7.4       (io)       repasthpc/2.3.0   (bio, D)
cslib/20180813  (chem)    netcdf-mpi/4.9.0       (io, D)    rosetta/3.10      (chem)
dakota/6.13      (t)       neuron/7.8.2           (bio)      rosetta/3.12      (chem)
delft3d/62441   (chem)    neuron/8.0.0           (bio, D)   rosetta/3.13      (chem)
dl_poly4/4.10.0 (chem)    nwchem/6.8.1           (chem)     rosetta/2019.21.60746 (chem, D)
elpa/2020.05.001 (math)    nwchem/7.0.2-p1       (chem, D)  scalapack/2.1.0   (math)
esmf/8.0.1       (geo)     octopus/10.1           (chem)     scotch/6.0.9      (math)
esmf/8.2.0       (geo, D)  openfoam-extend/4.1    (phys)     shengbte/1.1.1    (phys)
etsf_io-mpi/1.0.4 (chem)    openfoam/6             (phys)     shengbte/1.5.0    (phys, D)
fds/6.7.5        (chem)    openmolcas/20.10       (chem)     siesta/4.0.1      (chem)
fds/6.7.6        (chem)    openmx/3.9             (chem)     siesta/4.1-b4     (chem)
fds/6.7.7        (chem)    openmx/3.9.9           (chem, D)  siesta/4.1-MaX-3.0 (chem)
fds/6.7.8        (chem)    osu-micro-benchmarks/5.6.2 (t)       siesta/4.1.5      (chem, D)
fds/6.7.9        (D)      p4est/2.2              (math)     slepc/3.14.2      (chem)
fftw-mpi/3.3.8   (math)    parallelio/2.5.4       (chem)     sundials/2.7.0    (chem)
ga/5.7.2         (t)      paraview-offscreen/5.8.0 (vis)     sundials/5.3.0    (D)
lines 1-38

```

## Software stacks:

- CCEnv (S)
- GrexEnv (S,L)

## On Grex:

module load CCEnv  
 module load arch/avx512  
 module load StdEnv/2020  
 module avail

## Jobs:

skylake partition  
 largemem partition



```
[~@yak ~]$ module spider geant4
```

Versions:

```
geant4/9.6.p04
```

```
geant4/10.02.p03
```

```
geant4/10.04.p02
```

```
geant4/10.06
```

```
geant4/10.7.3
```

```
geant4/11.1.0
```

Other possible modules matches:

```
geant4-data
```

To find other possible module matches execute:

```
$ module -r spider '.*geant4.*'
```

For detailed information about a specific "geant4" 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 geant4/11.1.0
```

```
[~@yak ~]$ module load CCEnv
```

```
[~@yak ~]$ module load arch/avx512
```

```
[~@yak ~]$ module load StdEnv/2020
```

```
[~@yak ~]$ module spider geant4
```

```
[~@yak ~]$ module spider geant4/11.1.0
```

```
[~@yak ~]$ module load StdEnv/2020 gcc/9.3.0 geant4/11.1.0
```



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



- ★ **R** packages: minimal installation
  - **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.
- ★ **Perl** and **bioperl** as modules:
  - users can install the packages needed in their home directory.
- ★ Other software installed locally:
  - **Home made programs** {up to a user or a group}
  - **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.alliancecan.ca/wiki/Installing\\_software\\_in\\_your\\_home\\_directory](https://docs.alliancecan.ca/wiki/Installing_software_in_your_home_directory)





# Local installation: R packages

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

Choose a module version: module spider r

Load R and dependencies (gdal, geos, 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")
```

```
[~]$ module spider r
Versions:
r/4.0.0
r/4.0.2
r/4.0.5
r/4.1.0
r/4.1.2
r/4.2.1
r/4.2.2
```



# Local installation: Python

- ★ Load the modules:
  - `module load python`
- ★ Create a virtual environment
  - `virtualenv ~/my_venv`
- ★ Activate the virtual environment
  - `source ~/my_venv/bin/activate`
- ★ Update pip
  - `pip install --no-index --upgrade pip`
- ★ Install the packages
  - `pip install pandas`
  - `pip install -r requirements.txt`
  - `python setup.py install`

```
module load gcc gcc/11.2 python/3.11.2
virtualenv ~/my_venv
source ~/my_venv/bin/activate
pip install cutadapt
deactivate
```

```
module load gcc gcc/11.2 python/3.11.2
source ~/my_venv/bin/activate
cutadapt [+options]
deactivate
```

<https://docs.alliancecan.ca/wiki/Python>



**Example:** Hash::

**Load Perl module:** module load perl

**Install the the first package using cpan or cpanm:**

```
~$ 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 using cpan or cpanm:**

```
~$ cpan install Hash::
```

```
~$ cpan install Logger::
```

```
~$ cpan install MCE::
```



# Installation with make: **STAR**

- ★ Download the code {wget; curl; git clone; ...}:

```
wget https://github.com/alexdobin/STAR/archive/refs/tags/2.7.10b.tar.gz
```

- ★ Unpack the code: `tar -xvf 2.7.10b.tar.gz`

- ★ Load GCC compiler: `module load gcc`

**STAR**: Spliced Transcripts  
Alignment to a Reference

- ★ Compile the code:

```
cd STAR-2.7.10b/source  
make
```

```
~/.bash_profile
```

```
PATH=$PATH:$HOME/bin:$HOME/software/star/2.7.10b/bin  
export PATH
```

- ★ Copy the binaries and set the path:

```
mkdir -p ~/software/star/2.7.10b/bin
```

```
cp STAR ~/software/star/2.7.10b/bin
```

```
export PATH=$PATH:${HOME}/software/star/2.7.10b/bin
```



# Installation with configure/make

- ★ Download and unpack the code: `wget, ... gunzip, ... etc.`
- ★ Load the modules and dependencies: `module load gcc omp fftw`
- ★ Configure the program
  - If configure not included, run: `autoreconf -fvi` [to generate it].
  - `./configure --help` [to see the different options].
  - `./configure --prefix=<path to install dir> {+other options}`
- ★ Compile and test:
  - `make; make -j4`
  - `make check; make test`
- ★ Install the program:
  - `make install`



- ★ Download the source files:  
`wget https://bitbucket.org/nygcresearch/treemix/downloads/treemix-1.13.tar.gz`
- ★ Unpack the source files: `tar -xvf treemix-1.13.tar.gz`
- ★ Change the directory: `cd treemix-1.13/`
- ★ Load the modules: `module load gcc boost`
- ★ Configure: `./configure --prefix=/home/$USER/software/treemix/1.13`
- ★ Compile and install: `make && make install`
- ★ Set a path: `export PATH=$PATH:$HOME/software/treemix/1.13/bin`
- ★ Usage in a job script:  
`module load gcc boost`  
`export PATH=$PATH:$HOME/software/treemix/1.13/bin`  
`treemix {+options if any}`

# Example: 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
```

# Example: **ABINIT**

module load intel ompi gsl netcdf

instdir=<path to the installation directory>

```
./configure --prefix=${instdir} --enable-mpi --enable-mpi-io --with-fft-flavor=fftw3-mkl  
--with-linalg-flavor=mkl --with-math-flavor=gsl --enable-debug="no"  
--enable-optim="standard" --enable-64bit-flags --with-linalg-libs="-L$MKLROOT/lib/intel64  
-lmkl_scalapack_lp64 -lmkl_blacs_openmpi_lp64 -lmkl_intel_lp64 -lmkl_sequential  
-lmkl_core -lm" --with-fft-incs="-I$MKLROOT/include/fftw -I$MKLROOT/interfaces/fftw3xf"  
--with-fft-libs="-L$MKLROOT/interfaces/fftw3xf -lfftw3xf_intel_lp64"  
--with-dft-flavor="atompaw+libxc+wannier90" --with-trio-flavor="netcdf" --enable-lotf  
--enable-macroave --enable-gw-dpc CC=mpicc CXX=mpic++ FC=mpif90 F77=mpif77  
F90=mpif90
```





# Example with cmake/make

- ★ Download and unpack the code: `wget, ... gunzip, ... etc.`
- ★ Load the modules and dependencies: `module load gcc omp fftw`
- ★ Configure the program: `you may need to load cmake module`
  - `mkdir build && cd build`
  - `cmake .. --help` [to see the different options].
  - `cmake .. -DCMAKE_INSTALL_PREFIX=installdir {+other options}`
- ★ Compile and test:
  - `make; make -j8`
  - `make check; make test`
- ★ Install the program:
  - `make install`

Using the GUI:

```
ccmake .. {+ options}
```



# Cmake options for GROMACS

- ★ Download and unpack the source files
- ★ Load modules:  
module load intel/15.0 ompi/3.1.4 fftw cmake
- ★ configure; compile; install  
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; make install



# Cmake options for CERN-ROOT

★ Download and unpack the source files

★ Load modules:

```
module load gcc/9.2 tbb/2019.5 gsl/2.7.1 python/3.7.5 fftw/3.3.8
```

```
module load cfitsio/4.0.0 mkl/2019.5 cmake/3.16.9 git
```

```
export FFTW_DIR=/global/software/cent7/fftw/3.3.8-gcc92-ompi314
```

★ configure; compile; install

```
mkdir build && cd build
```

```
cmake -DCMAKE_INSTALL_PREFIX=<path to install dir> -DGCC_INSTALL_PREFIX=/global/software/cent7/gcc-9.2.0-rpath-76/ \  
-DPYTHON_EXECUTABLE=`which python` -DGSL_CONFIG_EXECUTABLE=/global/software/cent7/gsl/2.7.1-gcc92/bin/gsl-config \  
-DGSL_INCLUDE_DIR=/global/software/cent7/gsl/2.7.1-gcc92/include \  
-DGSL_LIBRARY=/global/software/cent7/gsl/2.7.1-gcc92/lib/libgsl.so \  
-DGSL_ROOT_DIR=/global/software/cent7/gsl/2.7.1-gcc92/ -DCMAKE_C_COMPILER=`which gcc` \  
-DCMAKE_CXX_COMPILER=`which g++` -DCMAKE_FORTRAN_COMPIER=`which gfortran` \  
-DCFITSIO_INCLUDE_DIR=/global/software/cent7/cfitsio/4.0.0-gcc92/include \  
-DCFITSIO_LIBRARY=/global/software/cent7/cfitsio/4.0.0-gcc92/lib/libcfitsio.so \  
-DJEMALLOC_ROOT_DIR=/usr -Dpython3=ON -Dpcre=ON -Dzlib=ON -Dunuran=ON -Dexplicitlink=ON -Dminuit2=ON -Droofit=ON \  
-DCMAKE_SKIP_RPATH=ON -Dxrootd=OFF -Dmysql=OFF -Dkrb5=OFF -Dodbc=OFF -Doracle=OFF \  
-DQT_QMAKE_EXECUTABLE=/usr/bin/qmake-qt4 <path to source directory>
```



- ★ 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}`



# Easybuild as package manager

---

All CC software stack is installed using easybuild

- ★ Software process:
  - A user request to install software: **Ticket**
  - Evaluation of the software: **HPC analyst**
  - Opening an internal ticket for software installation
    - An analyst will install the software
    - Local installation
    - Update the recipe on GitHub
    - Compile the program, push to dev and prod
  - An analyst will follow up with the user: original ticket



# Using easybuild as a user

<https://github.com/ComputeCanada/easybuild-easyconfigs/tree/computecanada-main/easybuild/easyconfigs>

- Search for the recipe: `eb -S ont-guppy`
- Copy the recipes:
  - ◆ `eb ont-guppy-6.2.1.eb --try-software-version=6.4.8 --copy-ec`
  - ◆ `eb ont-guppy-6.2.1-gcccuda-2020a.eb --try-software-version=6.4.8 --copy-ec`
- Edit the recipes and change the checksums:
  - ◆ `eb ont-guppy-6.4.8.eb --force --inject-checksums`
  - ◆ `eb ont-guppy-6.4.8-gcccuda-2020a.eb --force --inject-checksums`
- Install the modules:
  - ◆ `eb ont-guppy-6.4.8.eb --force --sourcepath=.`
  - ◆ `eb ont-guppy-6.4.8-gcccuda-2020a.eb --force --sourcepath=.`

<https://docs.easybuild.io/>

`eb <some recipe> [+options]`

**Resources:** [Github](#), [DockerHub](#), SingularityHub, Aptainer.

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

- ★ **Documentation:** <https://singularityware.github.io/user-guide.html>
- ★ **DockerHub:** <https://hub.docker.com/explore/>
- ★ **SingularityHub:** <https://www.singularity-hub.org/>
- ★ **Aptainer:** <https://apptainer.org/docs/>

<https://docs.alliancecan.ca/wiki/Singularity/en>  
<https://um-grex.github.io/grex-docs/>

**Access to Singularity:**

- ★ **Connect to cluster:** Grex, cedar, graham, beluga or narval:
- ★ **Load a module:** module load singularity [or aptainer]
- ★ **Build the image:** convert the image from Docker to Singularity [Aptainer]
- ★ **Note:** You may need to use your own Linux machine or VM to build the image.

- ★ **Alternative for running software:** difficult to build from source
- ★ Possibility to **convert Docker** images to **singularity**.
- ★ **Singularity installed on all clusters** {no Docker for security reasons}
- ★ **Build the image:**
  - module load singularity
  - singularity build qiime2-2021.11.sif docker://quay.io/qiime2/core:2021.11
- ★ **Run the code via singularity:**

```
singularity exec -B $PWD:/home -B /global/scratch/someuser:/outputs \  
-B /global/scratch/someuser/path/to/inputs:/inputs <path to qiime2-2021.11.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
```



★ FAQ:

- [https://docs.alliancecan.ca/wiki/Frequently\\_Asked\\_Questions](https://docs.alliancecan.ca/wiki/Frequently_Asked_Questions)

★ Software:

- [https://docs.alliancecan.ca/wiki/Available\\_software](https://docs.alliancecan.ca/wiki/Available_software)
- [https://docs.alliancecan.ca/wiki/Utiliser\\_des\\_modules/en](https://docs.alliancecan.ca/wiki/Utiliser_des_modules/en)
- [https://docs.alliancecan.ca/wiki/Installing\\_software\\_in\\_your\\_home\\_directory](https://docs.alliancecan.ca/wiki/Installing_software_in_your_home_directory)

★ Grex: <https://um-grex.github.io/grex-docs/>

*Thank you for your attention*

*Any question?*