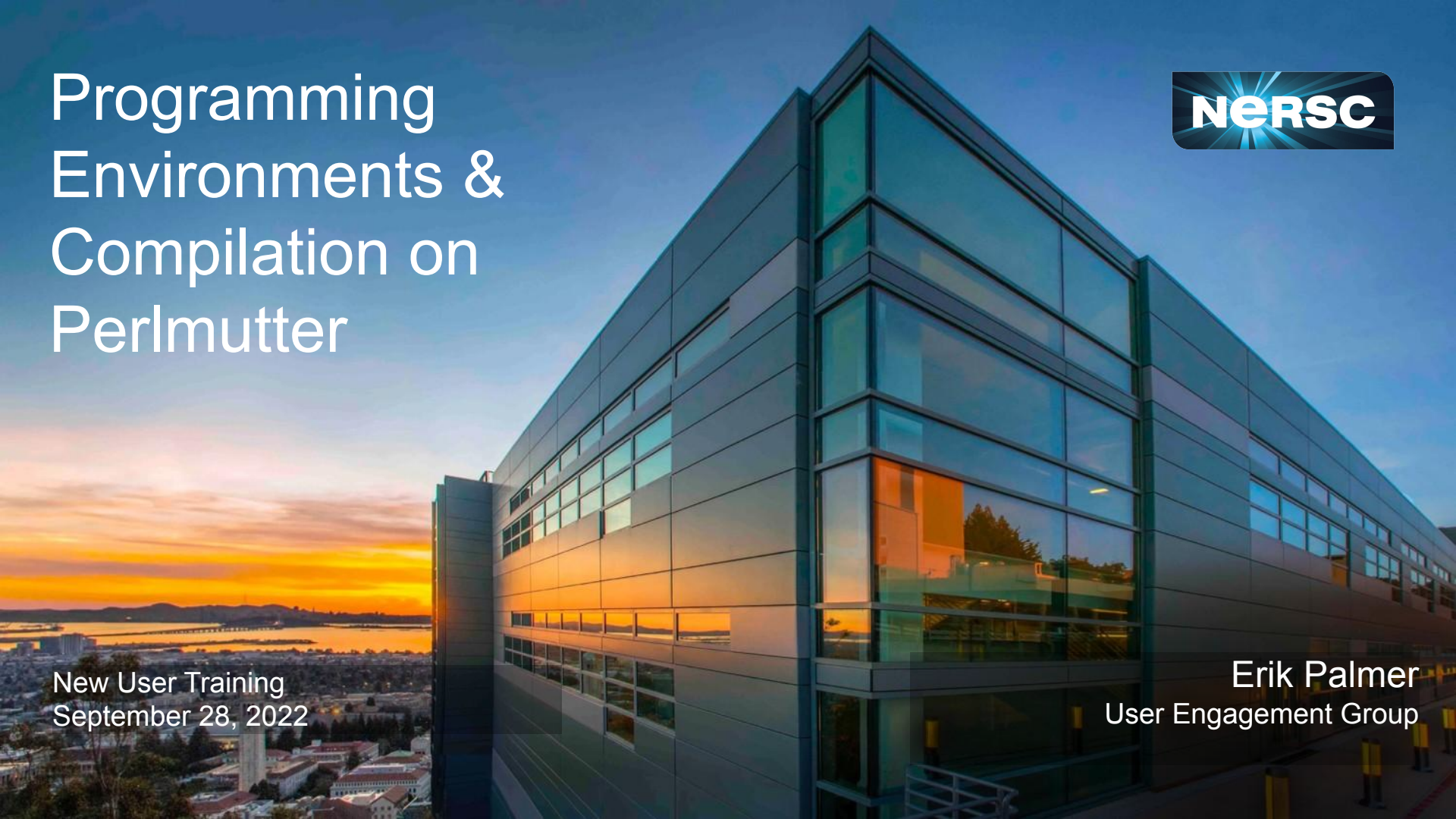
Programming Environments & Compilation on Perlmutter

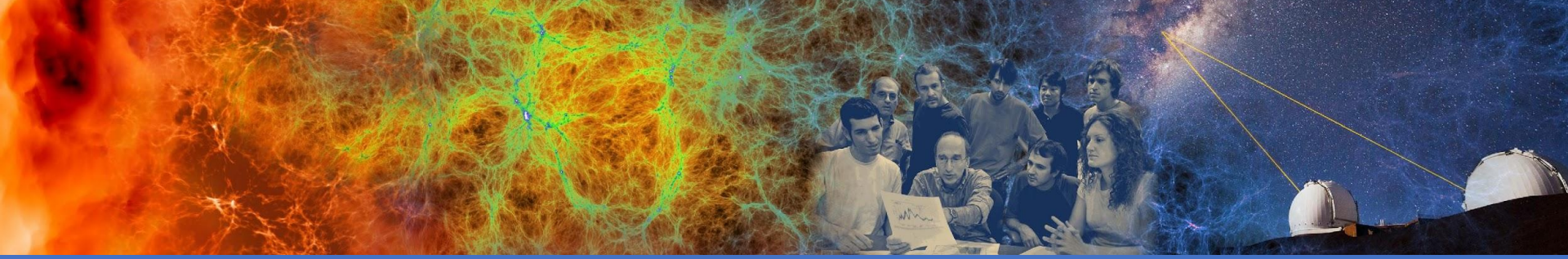


New User Training
September 28, 2022

Erik Palmer
User Engagement Group

Outline

- Modules: Loading Preinstalled Software
- Programing Environments: Configuring Compilers and Libraries
- Shell Configuration Scripts: Automate Repetitive Commands
- Spack: Installing Your Own Dependencies



Modules: Loading Preinstalled Software

Modules For Preinstalled Software: Ex. Python3

```
epalmer@perlmutter:login36:~> $ python --version
Python 2.7.18
epalmer@perlmutter:login36:~> $ module list

Currently Loaded Modules:
  1) gcc/11.2.0         4) libfabric/1.15.0.0   7) cray-libsci/21.08.1.2
  2) craype/2.7.16      5) craype-network-ofi   8) PrgEnv-gnu/8.3.3
  3) cray-dsmml/0.2.2   6) cray-mpich/8.1.17

epalmer@perlmutter:login36:~> $ module load python
epalmer@perlmutter:login36:~> $ module list

Currently Loaded Modules:
  1) gcc/11.2.0         4) libfabric/1.15.0.0   7) cray-libsci/21.08.1.2
  2) craype/2.7.16      5) craype-network-ofi   8) PrgEnv-gnu/8.3.3
  3) cray-dsmml/0.2.2   6) cray-mpich/8.1.17    9) python/3.9-anaconda-2021.11

epalmer@perlmutter:login36:~> $ python --version
Python 3.9.7
```

160 Modules Currently Available on Perlmutter

Nsight-Compute: Nsight-Compute/2022.1.1
Nsight-Systems: Nsight-Systems/2022.2.1
PrgEnv-aocc: PrgEnv-aocc/8.3.3
PrgEnv-cray: PrgEnv-cray/8.3.3
PrgEnv-gnu: PrgEnv-gnu/8.3.3
PrgEnv-nvhpc: PrgEnv-nvhpc/8.3.3
PrgEnv-nvidia: PrgEnv-nvidia/8.3.3
adios2:
allinea-forge:
amber:
amrex:
aocc: aocc/3.2.0
aocc-mixed: aocc-mixed/3.2.0
arm-forge:
arpack-ng: arpack-ng/3.8.0
atp:
berkeleygw:
boost: boost/1.78.0-gnu
cce:
cce-mixed:
chapel:
cmake: cmake/3.22.0
codee:
conduit:
cpe:
cpe-cuda:
cpu: cpu/1.0
cray-R:
cray-ccdb:
cray-cti:
cray-dsmml: cray-dsmml/0.2.2
cray-dyninst:
cray-fftw:
cray-hdf5:
cray-hdf5-parallel:
cray-libpals:
cray-libsci:
cray-lustre-client-ofed:
cray-mpich-abi:
cray-mrnet:
cray-netcdf:
cray-netcdf-hdf5parallel:
cray-openshmemx:
cray-pals:
cray-parallel-netcdf:
cray-pmi:
cray-pmi-lib: cray-pmi-lib/6.0.17
cray-python:
cray-stat:
cray-ucx: cray-ucx/2.7.0-1
craype:
craype-accel-amd-gfx908: craype-accel-amd-gfx908
craype-accel-amd-gfx90a: craype-accel-amd-gfx90a
craype-accel-host: craype-accel-host
craype-accel-nvidia70: craype-accel-nvidia70
craype-accel-nvidia80: craype-accel-nvidia80
craype-dl-plugin-ftr: craype-dl-plugin-ftr/22.06.1.2
craype-dl-plugin-py3:
craype-hugepages128M: craype-hugepages128M
craype-hugepages16M: craype-hugepages16M
craype-hugepages1G: craype-hugepages1G
craype-hugepages256M: craype-hugepages256M
craype-hugepages2G: craype-hugepages2G
craype-hugepages2M: craype-hugepages2M
craype-hugepages32M: craype-hugepages32M
craype-hugepages4M: craype-hugepages4M
craype-hugepages512M: craype-hugepages512M
craype-hugepages64M: craype-hugepages64M
craype-hugepages8M: craype-hugepages8M
craype-network-none: craype-network-none
craype-network-ofi: craype-network-ofi
craype-network-ucx: craype-network-ucx
craype-x86-milan: craype-x86-milan
craype-x86-milan-x: craype-x86-milan-x
craype-x86-rome: craype-x86-rome
craype-x86-spr: craype-x86-spr
craype-x86-trento: craype-x86-trento
craypkg-gen:
cudatoolkit:
cudnn:
darshan:
dvs: dvs/2.15_4.3.62-2.3_25.1__g030d7732
dyninst:
e4s:
espresso:
fast-mkl-amd: fast-mkl-amd/fast-mkl-amd
fftw:
gasnet:
gcc:
gcc-mixed: gcc-mixed/11.2.0
gdb4hpc:
globalarrays:
gpu: gpu/1.0
gromacs:
gsl:
hypre:
idl: idl/8.5
jgi: jgi/python-jamo
kokkos-kernels:
libfabric: libfabric/1.15.0.0
lmod: lmod
mathematica: mathematica/13.0.1
matlab:
mercury:
metis:
mfem:
mongodb:
mpark-variant:
namd:
nccl: nccl/2.11.4
nccmp:
nco:
netcdf-fortran:
nvhpc:
nvhpc-mixed:
nvidia:
nvidia-mixed:
openpmd-api:
papi:
papyrus:
parallel: parallel/20210922
parallel-netcdf:
parsec:
pdt:
perftools: perftools
perftools-base:
perftools-lite: perftools-lite
perftools-lite-events: perftools-lite-events
perftools-lite-gpu: perftools-lite-gpu
perftools-lite-hbm: perftools-lite-hbm
perftools-lite-loops: perftools-lite-loops
perftools-preload: perftools-preload
petsc:
pwanalyzer:
python: python/3.9-anaconda-2021.11
pytorch:
qthreads:
raja:
sanitizers4hpc: sanitizers4hpc/1.0.0
settarg: settarg
slepc:
spack:
sundials:
superlu:
swig:
tau:
tensorflow:
totalview:
training: training/perlmutter-jan2022
trilinos:
umap:
umpire:
upcxx:
valgrind4hpc:
vasp:
vasp-tpc: vasp-tpc/6.2.1-gpu
xpmem: xpmem/2.4.4-2.3_12.2__gff0e1d9.shasta
zfp:
zlib: zlib/1.2.11

Modules Loaded at Login

Modules Loaded by Default:

1) craype-x86-milan
2) libfabric/1.15.0.0
3) craype-network-ofi
4) perftools-base/22.06.0
5) xpmem/2.4.4-2.3_12.2__gff0e1d9.shasta
6) gcc/11.2.0
7) craype/2.7.16
8) cray-dsmml/0.2.2
9) cray-mpich/8.1.17
10) cray-libsci/21.08.1.2
11) PrgEnv-gnu/8.3.3
12) xalt/2.10.2
13) darshan/3.3.1
14) Nsight-Compute/2022.1.1
15) Nsight-Systems/2022.2.1
16) cudatoolkit/11.7
17) craype-accel-nvidia80
18) gpu/1.0

- CPU Architecture
- Default Programming Environment and Compiler
- GPU Architecture and CUDA-Aware MPI

Modules with Lmod

Most Common

- `module list`
- `module load/unload`
- `module swap`
- `module show`
- `module spider`

Cool Tricks

- `module --redirect -r spider . | grep <string>`
- `ml -t`
- `module reset` (expected after the next Perlmutter maintenance.)

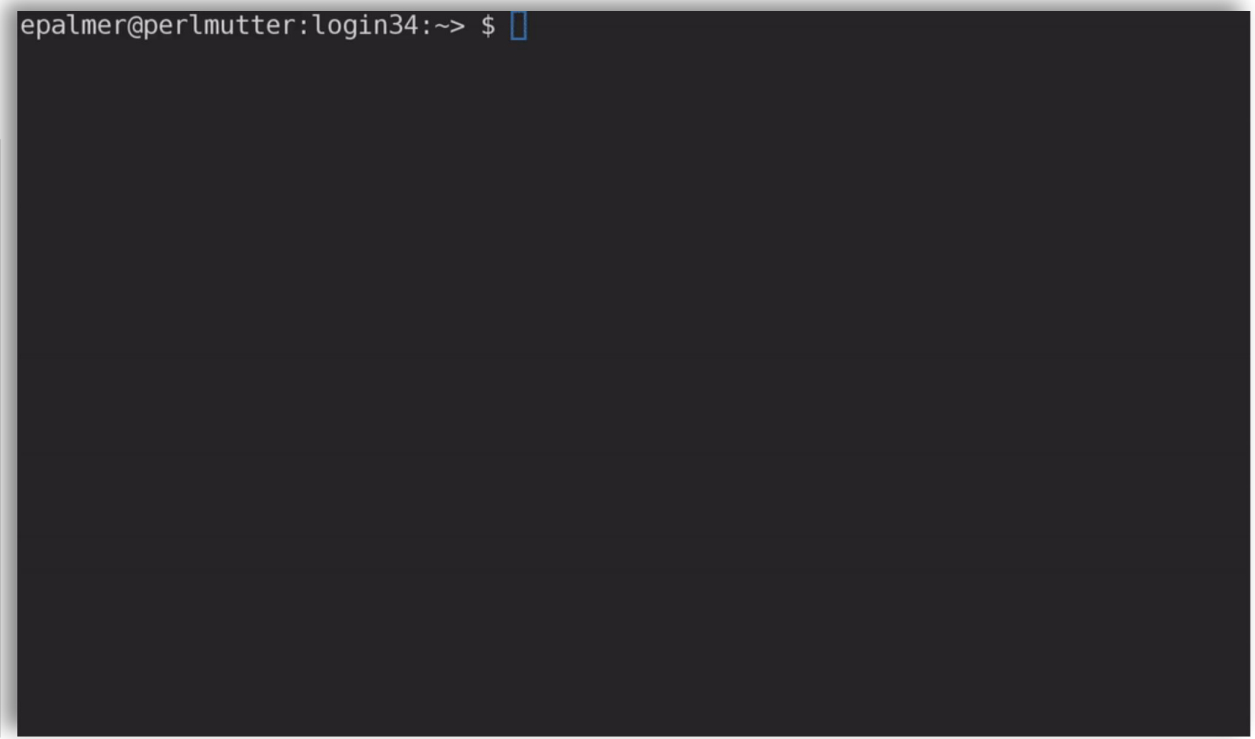
More information: `man module` or https://docs.nersc.gov/environment/lmod/

Modules with Lmod

No Longer Recommended

- `module avail`

  *Only shows packages that can be loaded into the current module environment (due to hierarchy) – use `module spider` instead



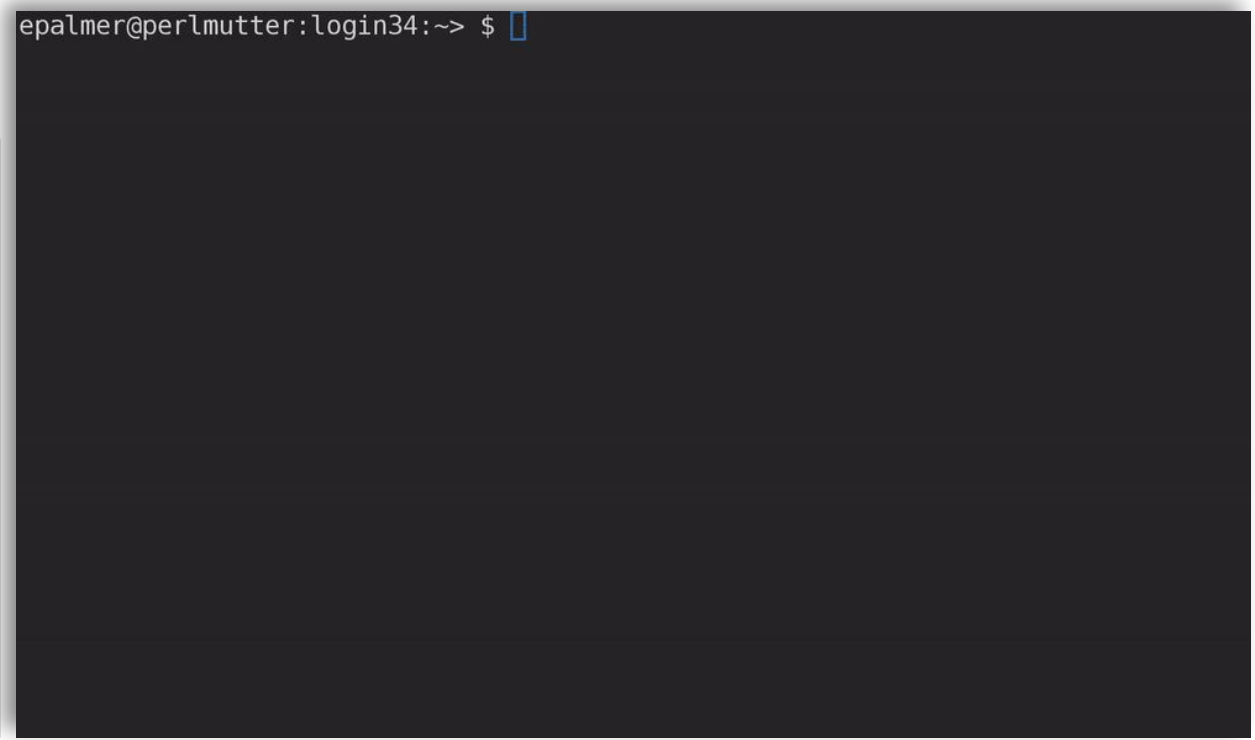
More information: `man module` or https://docs.nersc.gov/environment/lmod/

Modules with Lmod

No Longer Recommended

- `module avail`

  *Only shows packages that can be loaded into the current module environment (due to hierarchy) – use `module spider` instead

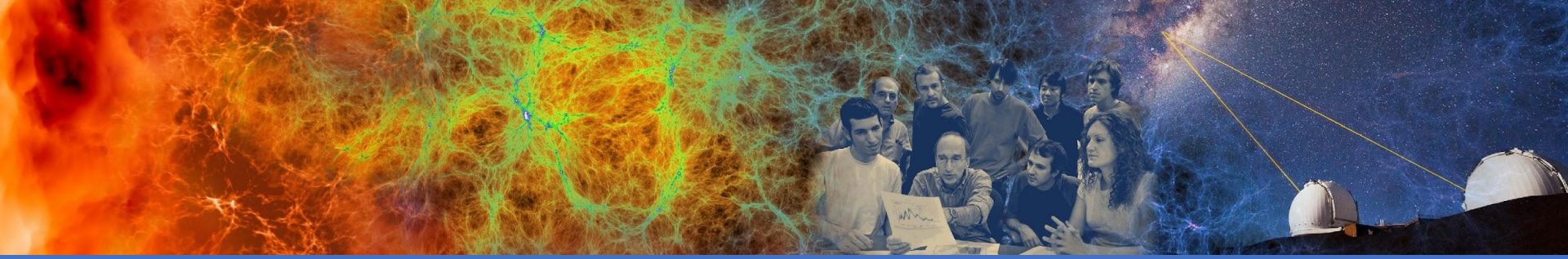


More information: `man module` or https://docs.nersc.gov/environment/lmod/

Loading Modules Modifies Your Environment

```
epalmer@perlmutter:login25:~/Training> $ module show cray-hdf5
-------------------------------------------------------------------------------------------------------------------------
   /opt/cray/pe/lmod/modulefiles/compiler/gnu/8.0/cray-hdf5/1.12.1.5.lua:
-------------------------------------------------------------------------------------------------------------------------
family("hdf5")
conflict("PrgEnv-pathscale")
help([[Release info:  /opt/cray/pe/hdf5/1.12.1.5/release_info]])
whatis("The HDF5 Technology suite includes tools and applications for managing, manipulating, viewing, and analyzing data in the HDF5 format.")
prepend_path("PATH","/opt/cray/pe/hdf5/1.12.1.5/bin")
prepend_path("PKG_CONFIG_PATH","/opt/cray/pe/hdf5/1.12.1.5/gnu/9.1/lib/pkgconfig")
prepend_path("PE_PKGCONFIG_LIBS","hdf5_hl:hdf5")
setenv("PE_HDF5_PKGCONFIG_LIBS","hdf5_hl:hdf5")
prepend_path("PE_FORTRAN_PKGCONFIG_LIBS","hdf5hl_fortran:hdf5_fortran")
setenv("PE_HDF5_FORTRAN_PKGCONFIG_LIBS","hdf5hl_fortran:hdf5_fortran")
prepend_path("PE_CXX_PKGCONFIG_LIBS","hdf5_hl_cpp:hdf5_cpp")
setenv("PE_HDF5_CXX_PKGCONFIG_LIBS","hdf5_hl_cpp:hdf5_cpp")
setenv("CRAY_HDF5_DIR","/opt/cray/pe/hdf5/1.12.1.5")
setenv("PE_HDF5_DIR","/opt/cray/pe/hdf5/1.12.1.5")
setenv("CRAY_HDF5_VERSION","1.12.1.5")
setenv("CRAY_HDF5_PREFIX","/opt/cray/pe/hdf5/1.12.1.5/gnu/9.1")
setenv("HDF5_DIR","/opt/cray/pe/hdf5/1.12.1.5/gnu/9.1")
setenv("HDF5_ROOT","/opt/cray/pe/hdf5/1.12.1.5/gnu/9.1")
prepend_path("CRAY_LD_LIBRARY_PATH","/opt/cray/pe/hdf5/1.12.1.5/gnu/9.1/lib")
prepend_path("MODULEPATH","/opt/cray/pe/lmod/modulefiles/hdf5/gnu/8.0/cray-hdf5/1.12.1")
```

Path Changes
Environment Variables
Other Info



Programming Environments: Configuring Compilers and Libraries

Programming Environments
Control Compilers and Libraries on Perlmutter

Language	Wrapper	PrgEnv-gnu (default)	PrgEnv-nvidia	PrgEnv -cray
C++	CC	g++	nvc++	crayCC (Clang)
C	cc	gcc	nvc	craycc (Clang)
Fortran	ftn	gfortran	nvfortran	crayftn
MPI	-	cray-mpich	cray-mpich	cray-mpich
More info:		module show PrgEnv-gnu	module show nvidia	module show cce

Compiler Wrappers and a Useful Option

`-v` and `-craype-verbose` will show all the inputs added by the craype (Cray Programming Environment) to the compiler when the wrappers (CC, cc, ftn) are used.

```
epalmer@nid005015:~/Training> gcc helloworld_openmp.c -fopenmp -o hello

epalmer@nid005015:~/Training> cc -craype-verbose helloworld_openmp.c -fopenmp -o hello

gcc -march=znver3 -D__CRAY_X86_MILAN -D__CRAY_NVIDIA80
-D__CRAYXT_COMPUTE_LINUX_TARGET -D__TARGET_LINUX__ helloworld_openmp.c
-fopenmp -o hello -Wl,-rpath=/opt/cray/pe/gcc-libs -Wl,-Bdynamic
-I/opt/cray/pe/mpich/8.1.17/ofi/gnu/9.1/include
-I/opt/cray/pe/libsci/21.08.1.2/GNU/9.1/x86_64/include
-I/opt/cray/pe/dsmml/0.2.2/dsmml//include
-I/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/cuda/11.7/nvvm/include
-I/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/cuda/11.7/extras/CUPTI/include …(and more)
```

Compiler Wrapper Includes A Lot

epalmer@nid005015:~/Training> **cc -craype-verbose** helloworld_openmp.c -fopenmp -o hello

gcc -march=znver3 -D__CRAY_X86_MILAN -D__CRAY_NVIDIA80 -D__CRAYXT_COMPUTE_LINUX_TARGET
-D__TARGET_LINUX__ helloworld_openmp.c -fopenmp -o hello -Wl,-rpath=/opt/cray/pe/gcc-libs -Wl,-Bdynamic
-I/opt/cray/pe/mpich/8.1.17/ofi/gnu/9.1/include -I/opt/cray/pe/libsci/21.08.1.2/GNU/9.1/x86_64/include
-I/opt/cray/pe/dsmml/0.2.2/dsmml//include -I/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/cuda/11.7/nvvm/include
-I/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/cuda/11.7/extras/CUPTI/include
-I/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/cuda/11.7/extras/Debugger/include
-I/opt/cray/xpmem/2.4.4-2.3_12.2__gff0e1d9.shasta/include -L/opt/cray/pe/mpich/8.1.17/ofi/gnu/9.1/lib
-L/opt/cray/pe/mpich/8.1.17/gtl/lib -L/opt/cray/pe/libsci/21.08.1.2/GNU/9.1/x86_64/lib -L/opt/cray/pe/dsmml/0.2.2/dsmml//lib
-L/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/cuda/11.7/lib64/stubs -L/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/cuda/11.7/lib64
-L/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/cuda/11.7/nvvm/lib64
-L/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/cuda/11.7/extras/CUPTI/lib64
-L/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/cuda/11.7/extras/Debugger/lib64
-L/opt/nvidia/hpc_sdk/Linux_x86_64/22.5/math_libs/11.7/lib64 -L/global/common/software/nersc/pm-2021q4/sw/darshan/3.3.1/lib
-L/opt/cray/xpmem/2.4.4-2.3_12.2__gff0e1d9.shasta/lib64 -Wl,--as-needed,-lcupti,-lcudart,--no-as-needed -lcuda
-Wl,-rpath=/global/common/software/nersc/pm-2021q4/sw/darshan/3.3.1/lib -Wl,-no-as-needed -ldarshan -lz
-Wl,--as-needed,-lsci_gnu_82_mpi_mp,--no-as-needed -Wl,--as-needed,-lsci_gnu_82_mp,--no-as-needed -ldl
-Wl,--as-needed,-lmpi_gnu_91,--no-as-needed -lmpi_gtl_cuda -Wl,--as-needed,-ldsmml,--no-as-needed -lxpmem
-Wl,--as-needed,-lgfortran,-lquadmath,--no-as-needed -Wl,--as-needed,-lmvec,--no-as-needed -Wl,--as-needed,-lm,--no-as-needed
-Wl,--as-needed,-lpthread,--no-as-needed -Wl,--disable-new-dtags

Automatic Links Provided By The Wrappers

- Depending on modules loaded, compiler wrappers link:

  MPI, LAPACK, Blas, ScaLAPACK, and more,

  automatically.

- Cray modules, such as `cray-hdf5`, `cray-fftw`, etc. are also linked automatically by the compiler wrappers when loaded into the user environment.

Note: Several scientific libraries such as, LAPACK, ScaLAPACK, Blas and QDWH, are included in `cray-libsci`. For more information use: `man intro_libsci.`

CUDA-Aware MPI Enabled By Default

Modules Loaded by Default:

1) craype-x86-milan
2) libfabric/1.15.0.0
3) craype-network-ofi
4) perftools-base/22.06.0
5) xpmem/2.4.4-2.3_12.2__gff0e1d9.shasta
6) gcc/11.2.0
7) craype/2.7.16
8) cray-dsmml/0.2.2
9) cray-mpich/8.1.17
10) cray-libsci/21.08.1.2
11) PrgEnv-gnu/8.3.3
12) xalt/2.10.2
13) darshan/3.3.1
14) Nsight-Compute/2022.1.1
15) Nsight-Systems/2022.2.1
16) cudatoolkit/11.7
17) craype-accel-nvidia80
18) gpu/1.0

Output of `module show gpu`:

```
-------------------------------------------------------------------------------------------
   /global/common/software/nersc/pm-2022.08.4/extra_modulefiles/gpu/1.0.lua:
-------------------------------------------------------------------------------------------
family("hardware")
load("cudatoolkit")
load("craype-accel-nvidia80")
setenv("MPICH_GPU_SUPPORT_ENABLED","1")
```

Note: The gpu module enables support for CUDA-Aware MPI – Allowing MPI to copy data to and from GPUs. `module load cpu` will turn it off.

Manually Specify Cray Compile Wrappers

Some build systems such as CMake or Makefiles may be coded to search for CC, CXX and FC environment variables.

In these cases, it is possible to specify the Cray compile wrappers by setting the environment variables in the following way:

```
CC=$(which cc) CXX=$(which CC) FC=$(which ftn)
```

Or at the configure step,

```
./configure CC=cc CXX=CC FC=ftn F77=ftn
```

More info: https://docs.nersc.gov/development/build-tools/autoconf-make/
https://docs.nersc.gov/development/build-tools/cmake/

Modules Link Dynamically by Default

- Many modules prepend the `LD_LIBRARY_PATH` or `CRAY_LD_LIBRARY_PATH`, and have their shared libraries dynamically linked.
  e.g.

```
module load gsl
CC gsl_test.cpp -lgsl -lgslcblas -o gsl_test
```

- If you are compiling your own shared libraries, consider using the option, `-Wl,-rpath=<library path>`. Cray wrappers build dynamically linked executables by default.
- On Perlmutter static compilation with `-static` or `CRAYPE_LINK_TYPE=static` can fail and is not supported.

More info: https://docs.nersc.gov/development/compilers/wrappers/

Other Good-to-Know Compiler Settings

GNU	Cray	Nvidia	Description/ Comment
-O0	-O0	-O1	Default Optimization Level
-Ofast	-Ofast, -flto	-O4, -fast	Aggressive Optimization (some may cause non-bit-identical output)
-fopenmp	-fopenmp	-mp (CPU), -mp=gpu (GPU offload)	Enable OpenMP (not default)
		-acc	Enable OpenACC
-g, -O0	-g, -O0	-g (-O0 by default)	Debug
-v	-v	-v	Verbose

For more information:

`man gcc/gfortran`	`man craycc/crayftn`	`man nvc/nvfortran`

https://docs.nersc.gov/development/compilers/

Example CPU Code Compile with MPI and OpenMP

```
int main(int argc, char *argv[]) {
  int numprocs, rank, namelen;
  char processor_name[MPI_MAX_PROCESSOR_NAME];
  int iam = 0, np = 1;
  MPI_Init(&argc, &argv);
  MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  MPI_Get_processor_name(processor_name, &namelen);

  #pragma omp parallel default(shared) private(iam, np)
  {
    np = omp_get_num_threads();
    iam = omp_get_thread_num();
    printf("Hello from thread %d out of %d from process
           %d out of %d on %s\n",
           iam, np, rank, numprocs, processor_name);
  }
  MPI_Finalize();
 }
```

Hellohybrid.c – contains both MPI and OpenMP commands.

Example from: https://rcc.uchicago.edu/docs/running-jobs/hybrid/index.html

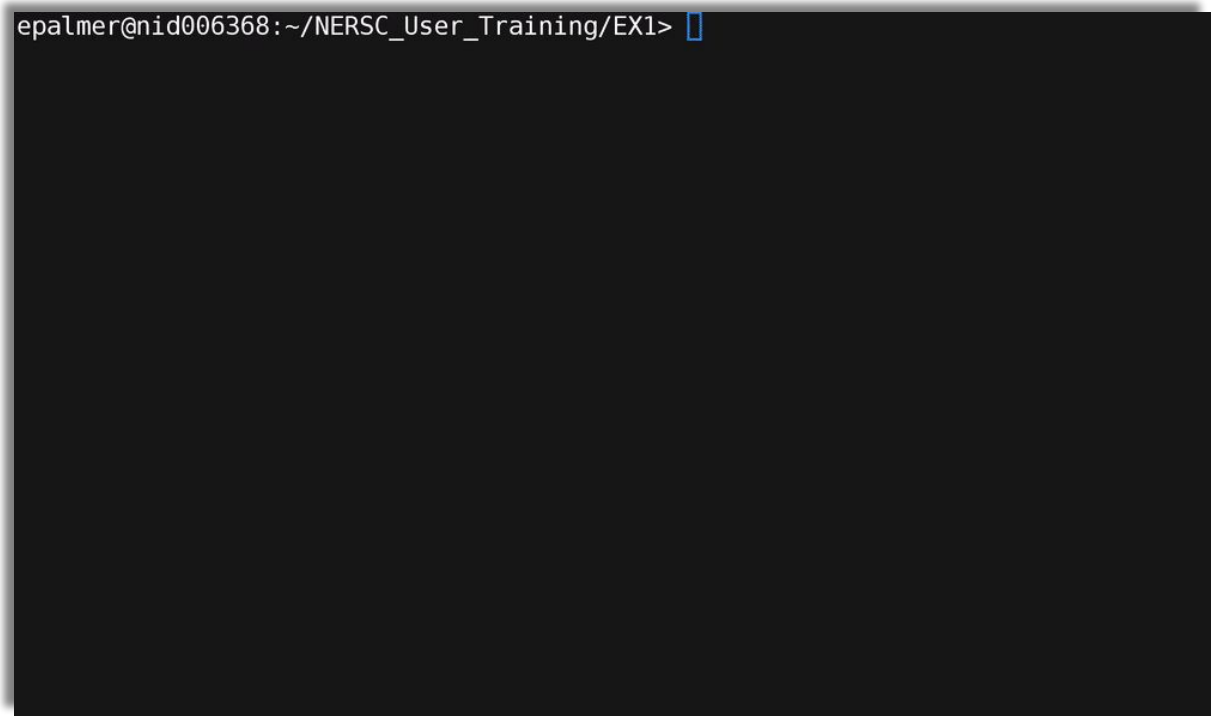
Compile with MPI and OpenMP (CPU only)

```
epalmer@nid006368:~/NERSC_User_Training/EX1> #In this example, we will compile a
 code with MPI and OpenMP
```

Modules Loaded:
PrgEnv-gnu

Compile line: `cc hellohybrid.c -fopenmp -o hellohybrid`

Compile with MPI and OpenMP (CPU only)



Modules Loaded:
PrgEnv-gnu

Compile line:

```
cc hellohybrid.c -fopenmp -o hellohybrid
```

Compile with MPI and OpenMP on GPU

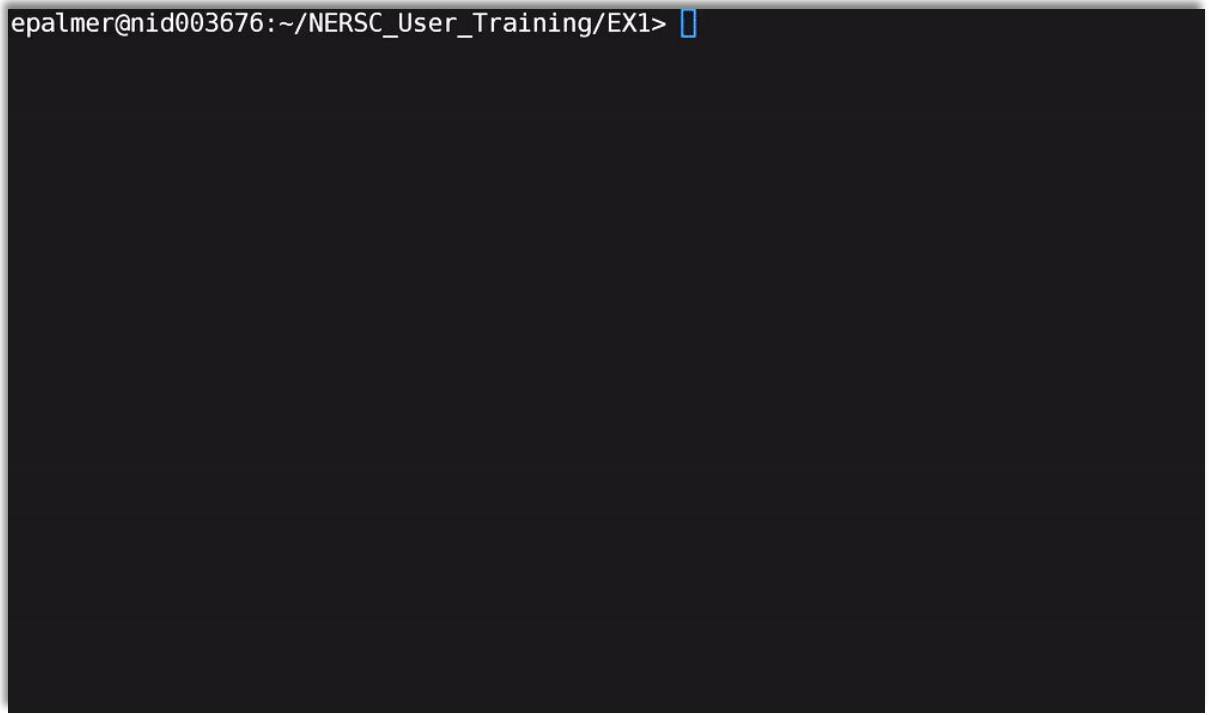
```
epalmer@nid003676:~/NERSC_User_Training/EX1> ls
hellohybrid.c
epalmer@nid003676:~/NERSC_User_Training/EX1> module list

Currently Loaded Modules:
  1) craype-x86-milan                      10) cudatoolkit/11.7
  2) libfabric/1.15.0.0                    11) craype-accel-nvidia80
  3) craype-network-ofi                    12) gpu/1.0
  4) perftools-base/22.06.0                13) nvidia/22.5
  5) xpmem/2.4.4-2.3_12.2__gff0e1d9.shasta 14) craype/2.7.16
  6) xalt/2.10.2                           15) cray-dsmml/0.2.2
  7) darshan/3.3.1                         16) cray-mpich/8.1.17
  8) Nsight-Compute/2022.1.1               17) cray-libsci/21.08.1.2
  9) Nsight-Systems/2022.2.1               18) PrgEnv-nvidia/8.3.3


epalmer@nid003676:~/NERSC_User_Training/EX1> cc
```

Modules Loaded:
PrgEnv-nvidia,
gpu

"-Minfo" is an optional command shows which parts were converted to NVIDIA GPU code.

Compile line: `cc hellohybrid.c -mp=gpu -Minfo -o hellohybrid`

Compile with MPI and OpenMP on GPU



Modules Loaded:
PrgEnv-nvidia,
gpu

"-Minfo" is an optional command shows which parts were converted to NVIDIA GPU code.

Compile line: `cc hellohybrid.c -mp=gpu -Minfo -o hellohybrid`

Compile Commands for Code with OpenACC

Modules Loaded:
PrgEnv-nvidia,
gpu

Compile with OpenACC enabled:

```
cc helloacc.c -acc -Minfo=acc -o helloacc
```

*Optional command shows which parts were converted to NVIDIA GPU code.

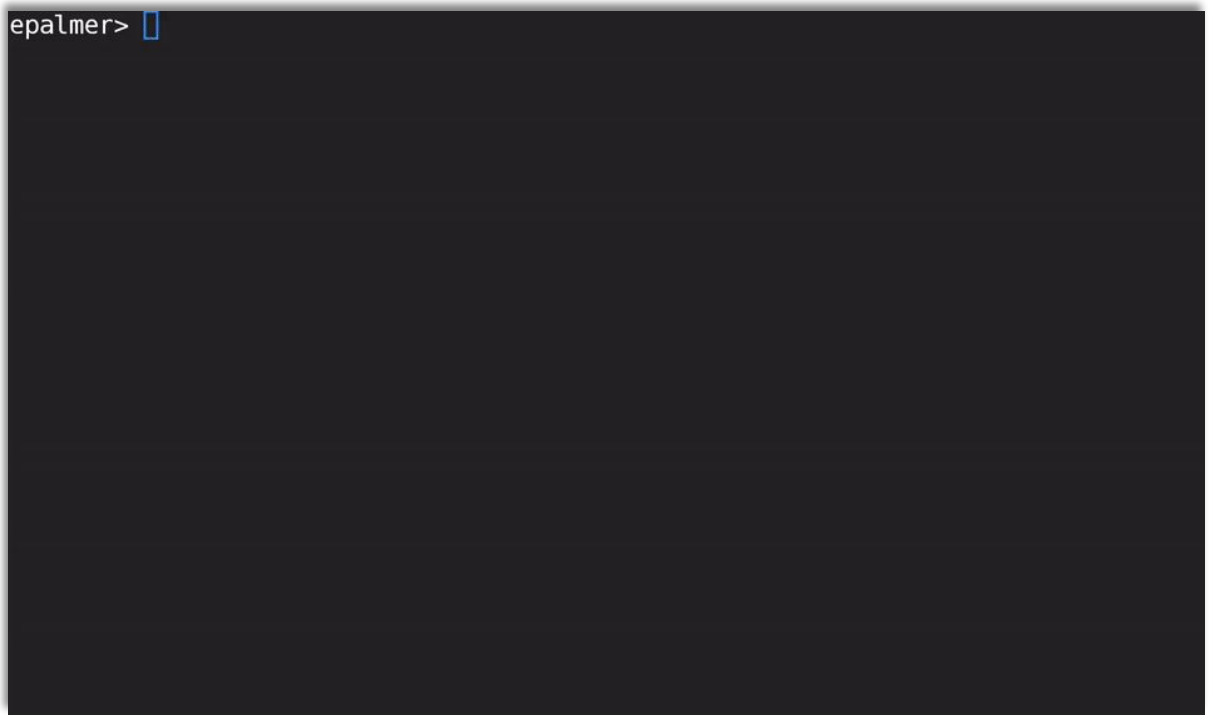
Compile a CUDA Code with CUDA-Aware MPI

```
epalmer> # In this example, we will compile a CUDA code with CUDA-aware MPI
epalmer> ls
kernels.cu  kernels.h  vecAdd.cpp
epalmer> module list
```

Modules Loaded:
PrgEnv-nvidia,
gpu

Note: The flag, `MPICH_GPU_SUPPORT_ENABLED` turns CUDA-Aware MPI on or off. Loading the gpu module, sets this flag to 1, thereby enabling the feature.

Compile a CUDA Code with CUDA-Aware MPI



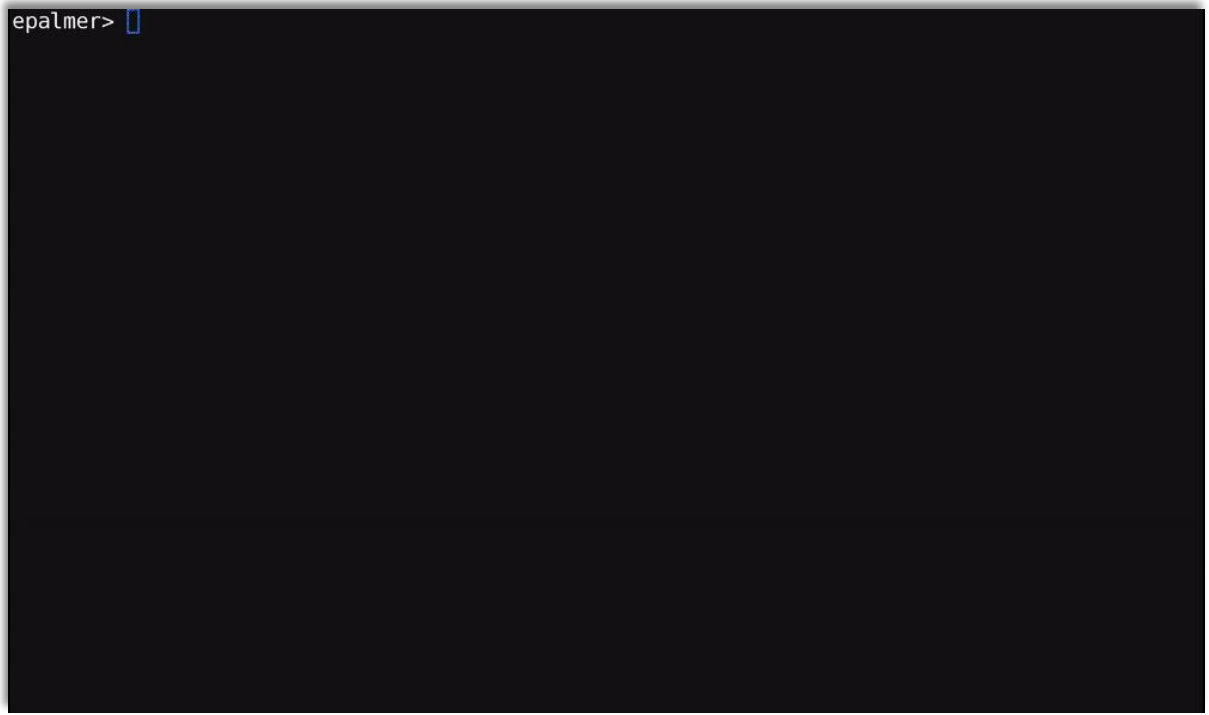
Modules Loaded:
PrgEnv-nvidia,
gpu

Note: The flag, `MPICH_GPU_SUPPORT_ENABLED` turns CUDA-Aware MPI on or off. Loading the gpu module, sets this flag to 1, thereby enabling the feature.

Manually Specify Include, Library Location and Links

```
epalmer> # In this example, we will show how to manually include and link libraries during
 the compile step. The example code I will use, requires
```

Manually Specify Include, Library Location and Links



More Resources on Compiling Code

- **NERSC Docs**:
  Compiling and Building Software (Perlmutter Specific):
  https://docs.nersc.gov/systems/perlmutter/#compilingbuilding-software
  Base Compilers:
  https://docs.nersc.gov/development/compilers/base/#base-compilers-on-nersc-systems
  Compiler Wrappers:
  https://docs.nersc.gov/development/compilers/wrappers/#compiler-wrappers
- **Using Perlmutter Training**:
  https://www.nersc.gov/users/training/events/using-perlmutter-training-jan2022/
  https://github.com/NERSC/Perlmutter_Training_Jan2022
- **Previous New User Training** (Cori Focused):
  https://www.nersc.gov/users/training/events/new-user-training-june-16-2020/

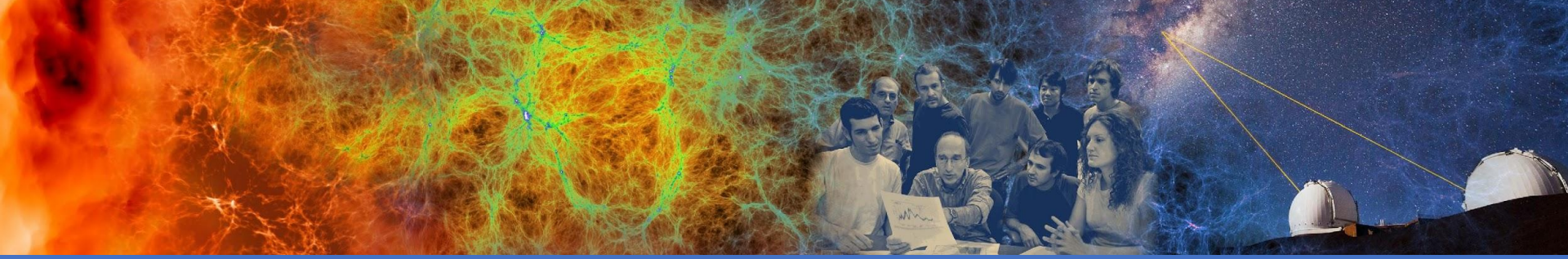
NERSC Programming Environment (Experimental)

Language	Suggested Compile Commands	PrgEnv-llvm
C++	mpic++	clang (Intel)
C	mpicc	clang (Intel)
Fortran	N/A	N/A
MPI	-	cray-mpich

Usage:

```
module use /global/cfs/cdirs/nstaff/cookbg/pe/modulefiles
module load npe
module load PrgEnv-llvm
```

More Info: https://docs.nersc.gov/development/compilers/npe/



Shell Configuration Scripts: Automate Repetitive Commands

Shell Configuration Scripts

Suppose each time you log on you begin by typing the following:

```
module load python
module load PrgEnv-cray
export HYPRE=/global/homes/e/ep/hypre_install_dir
export PATH=”/global/homes/e/ep/bin/:$PATH”
```

And you would like to type less…

Solution:

Add these lines to a shell script!

Source Your Scripts

Rather than modify your dot files (.bashrc, .bash_profile, etc.), write the same lines in a separate shell script called `config_myenv.sh`:

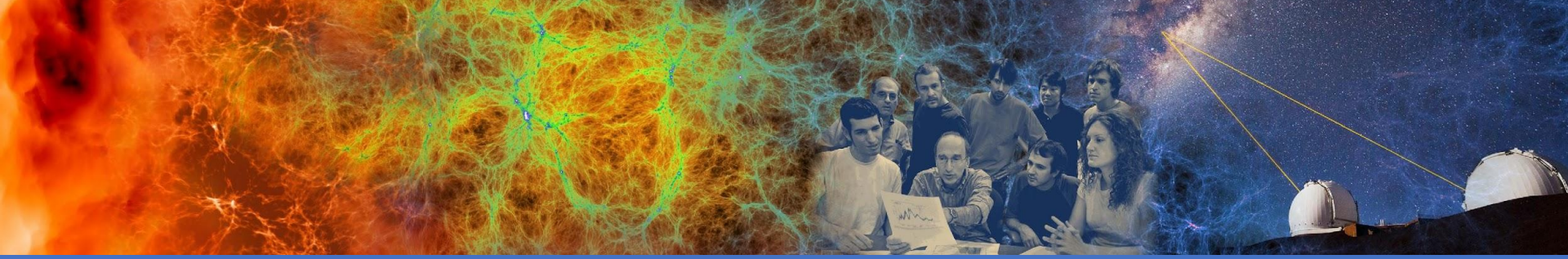
```
#!/bin/bash

module load python
module load PrgEnv-cray
export HYPRE=/global/homes/e/ep/hypre_install_dir
export PATH="/global/homes/e/ep/bin/:$PATH"
```

Then you can make the changes to your environment by typing:

```
epalmer@perlmutter:login35:~> $ source config_myenv.sh
```

Even less typing!

```
epalmer@perlmutter:login35:~> $ . config_myenv.sh
```



Spack: Even More Software!

Spack: A Package Manager for Supercomputers

Spack currently contains 6617 mainline packages

NERSC has two instances of Spack available on Perlmuter:

Instance	Version	Command	# of Packages	Can I Install More?
Spack	0.17.1	module load spack	**130**	Yes
Extreme-scale Scientific Software Stack (E4S)	0.18.0.dev0	module load e4s/22.05	**319**	Yes

More info: Spack – https://docs.nersc.gov/development/build-tools/spack/
E4S – https://docs.nersc.gov/applications/e4s/

Installed Spack Packages

```
epalmer@perlmutter:login36:~> $ spack find
==> 130 installed packages
-- cray-cnl7-haswell / intel@19.1.2.254 --
cfitsio@4.0.0

-- cray-sles15-zen3 / gcc@11.2.0 ---------
ant@1.10.7                glproto@1.4.17
arpack-ng@3.8.0           gperf@3.1
bdftopcf@1.0.5            hypre@2.24.0
bison@3.0.4               icu4c@67.1
bzip2@1.0.6               inputproto@2.3.2
cfitsio@4.0.0             intltool@0.51.0
claw@2.0.3                jdk@11.0.1_13
cmake@3.17.0              kbproto@1.0.7
cmake@3.22.0              krb5@1.19.2
cmake@3.22.1              lcms@2.9
cmake@3.22.2              libbsd@0.11.3
cray-libsci@21.08.1.2     libffi@3.3
cuda@11.7.64              libfontenc@1.1.3
curl@7.66.0               libgit2@1.1.1
docbook-xml@4.4           libice@1.0.9
docbook-xsl@1.79.2        libiconv@1.16
expat@2.4.4               libjpeg@9d
fixesproto@5.0            libmd@1.0.3
flex@2.6.4                libpng@1.6.37
font-util@1.3.2           libpthread-stubs@0
fontconfig@2.13.94        libsm@1.2.3
fontsproto@2.1.3          libssh2@1.10.0
freetype@2.11.1           libtiff@4.3.0
fribidi@1.0.5             libx11@1.7.0
ghostscript-fonts@8.11    libxau@1.0.8

-- cray-sles15-zen3 / nvhpc@21.11 --------
cairo@1.16.0  gperf@3.1  intel-mkl@2020.4
```

Spack

E4S

```
==> 319 installed packages
-- cray-sles15-zen3 / cce@13.0.2 ----------------------------
adios2@2.8.0     cray-libsci@21.08.1.2  gettext@0.21          libarchive@3.5.2  libmd@1.0.4     mbedtls@2.28.0  python@3.8.13  sundials@6.2.0          xz@5.2.5
arpack-ng@3.8.0  cray-mpich@8.1.15       hdf5@1.10.7           libarchive@3.5.2  libpng@1.6.37   metis@5.1.0     readline@7.0   superlu@5.3.0           zfp@0.5.5
bzip2@1.0.6      curl@7.66.0            hdf5@1.12.1           libbsd@0.11.5     libuv@1.44.1    ncurses@6.1     rhash@1.4.2    superlu-dist@7.2.0      zlib@1.2.12
c-blosc@1.21.1   diffutils@3.6          hypre@2.24.0          libfabric@1.15.0.0  libxml2@2.9.13  parmetis@4.0.3  slepc@3.17.1   sz@2.1.12               zstd@1.5.2
cmake@3.23.1     expat@2.4.8            kokkos@3.6.00         libffi@3.4.2      lz4@1.9.3       petsc@3.17.1    snappy@1.1.8   tar@1.30
cmake@3.23.1     gdbm@1.23              kokkos-kernels@3.6.00 libiconv@1.16     lzo@2.10        pkgconf@1.8.0   sqlite@3.38.5  util-linux-uuid@2.33.1

-- cray-sles15-zen3 / gcc@11.2.0 ----------------------------
adios2@2.8.0          dyninst@12.1.0        hpctoolkit@2022.04.15  libiberty@2.37        metis@4.0.3          pkgconf@1.8.0          snappy@1.1.8
adios2@2.8.0          eccodes@2.25.0        hpcviewer@2022.03      libice@1.0.9          metis@5.1.0          plasma@21.8.29         sqlite@3.38.5
amrex@22.05           elfutils@0.186        hpx@1.7.1              libiconv@1.16         mfem@4.4.0           proj@8.2.1             strumpack@6.3.1
antlr@2.7.7           elfutils@0.186        hwloc@2.7.1            libjpeg@9e            mkfontdir@1.0.7      py-beniget@0.4.1       strumpack@6.3.1
arpack-ng@3.8.0       esmf@8.2.0            hypre@2.24.0           libjpeg-turbo@2.1.3   mkfontscale@1.1.2    py-cython@0.29.24      sundials@6.2.0
asio@1.21.0           expat@2.4.8           hypre@2.24.0           libjpeg-turbo@2.1.3   mpark-variant@1.4.0  py-gast@0.5.3          superlu@5.3.0
autoconf@2.69         ffmpeg@4.4.1          hypre@2.24.0           libmd@1.0.4           nasm@2.15.05         py-mpi4py@3.1.2        superlu-dist@7.2.0
automake@1.16.5       fftw@3.3.10           icu4c@67.1             libmonitor@2021.11.08 nccmp@1.9.0.1        py-numpy@1.22.3        sz@2.1.12
bdftopcf@1.0.5        fftw@3.3.10           inputproto@2.3.2       libpciaccess@0.16     ncl@6.6.2            py-periodictable@1.5.0 tar@1.30
berkeley-db@18.1.40   findutils@4.6.0       intel-tbb@2020.3       libpng@1.6.37         nco@5.0.1            py-picmistandard@0.0.19 tasmanian@7.7
binutils@2.38         flex@2.6.3            intel-tbb@2020.3       libpthread-stubs@0.4  ncurses@6.1          py-pip@21.3.1          tau@2.31.1
bison@3.8.2           font-util@1.3.2       intel-xed@2022.04.17   libquo@1.3.1          ncview@2.1.8         py-pip@21.3.1          tcsh@6.24.00
blaspp@2021.04.01     fontconfig@2.13.94    kbproto@1.0.7          libsm@1.2.3           netcdf-c@4.8.1       py-ply@3.11            trilinos@13.0.1
blaspp@2021.04.01     fontconfig@2.13.94    kim-api@2.2.1          libtiff@4.3.0         netcdf-c@4.8.1       py-pybind11@2.8.1      udunits@2.2.28
blaspp@2021.04.01     fontsproto@2.1.3      kokkos@3.6.00          libtool@2.4.7         netcdf-c@4.8.1       py-pyparsing@3.0.6     util-linux-uuid@2.33.1
blt@0.4.1             fortrilinos@2.0.0     kokkos@3.6.00          libunwind@1.6.2       netcdf-fortran@4.5.4 py-python@0.10.0       util-macros@1.19.3
boost@1.79.0          freetype@2.11.1       kokkos-kernels@3.6.00  libunwind@1.6.2       netlib-scalapack@2.2.0 py-scipy@1.8.0       vtk-m@1.7.1
boost@1.79.0          fribidi@1.0.12        kokkos-kernels@3.6.00  libuv@1.44.1          ninja@1.10.2         py-setuptools@59.4.0   warpx@22.05
boost@1.79.0          gasnet@2022.3.0       kokkos-nvcc-wrapper@3.2.00 libx11@1.7.0      ninja@1.10.2         py-setuptools@59.4.0   warpx@22.05
boost@1.79.0          gdbm@1.19             lammps@20220107        libxau@1.0.8          nlohmann-json@3.10.5 py-warpx@22.05         warpx@22.05
butterflypack@2.1.1   gdbm@1.23             lapackpp@2021.04.00    libxaw@1.0.13         numactl@2.0.14       py-warpx@22.05         xcb-proto@1.14.1
bzip2@1.0.6           gettext@0.21          lapackpp@2021.04.00    libxcb@1.14           openblas@0.3.20      py-warpx@22.05         xerces-c@3.2.3
c-blosc@1.21.1        gettext@0.21          lapackpp@2021.04.00    libxdmcp@1.1.2        openblas@0.3.20      py-wheel@0.37.0        xerces-c@3.2.3
cairo@1.16.0          git@2.26.2            libaec@1.0.5           libxext@1.3.3         openjdk@11.0.14.1_1  py-wheel@0.37.0        xextproto@7.3.0
cairo@1.16.0          glib@2.72.1           libaec@1.0.5           libxfont@1.5.2        openjpeg@2.3.1       python@3.8.13          xproto@7.0.31
camp@0.2.2            googletest@1.10.0     libarchive@3.5.2       libxml2@2.9.13        openpmd-api@0.14.4   python@3.8.13          xtrans@1.3.5
camp@0.2.2            gperf@3.1             libarchive@3.5.2       libxml2@2.9.13        otf2@2.3             python@3.8.13          xz@5.2.5
cdo@2.0.5             gperftools@2.9.1      libarchive@3.5.2       libxmu@1.1.2          papi@6.0.0.1         python@3.8.13          xz@5.2.5
cmake@3.23.1          grads@2.2.1           libbsd@0.11.5          libxpm@3.5.12         papi@6.0.0.1         qthreads@1.16          yasm@1.3.0
cmake@3.23.1          gromacs@2021.5        libcerf@1.3            libxrender@0.9.10     parallel@20210922    raja@0.14.0            zfp@0.5.5
cmake@3.23.1          gsl@2.7               libdwarf@20180129      libxt@1.1.5           parallel-netcdf@1.12.2 raja@0.14.0          zfp@0.5.5
conduit@0.8.3         gsl@2.7.1             libdwarf@20180129      lz4@1.9.3             parmetis@4.0.3       readline@7.0           zlib@1.2.12
cray-libsci@21.08.1.2 hdf5@1.8.22           libfabric@1.15.0.0     lzo@2.10              pcre@8.45            renderproto@0.11.1     zstd@1.5.2
cray-mpich@8.1.15     hdf5@1.10.7           libffi@3.3             m4@1.4.18             pdt@3.25.1           rhash@1.4.2
cub@1.13.1            hdf5@1.10.8           libffi@3.4.2           makedepend@1.0.5      perl@5.34.1          shapelib@1.5.0
cuda@11.5.0           hdf5@1.12.1           libfontenc@1.1.3       mbedtls@2.28.0        petsc@3.17.1         slate@2021.05.02
curl@7.66.0           hdf5@1.12.1           libgd@2.2.4            memkind@1.13.0        petsc@3.17.1         slate@2021.05.02
diffutils@3.6         hdf5@1.12.1           libgd@2.2.4            meson@0.62.1          petsc@3.17.1         slepc@3.17.1
dyninst@12.1.0        heffte@2.2.0          libgeotiff@1.6.0       meson@0.62.1          pixman@0.40.0        slepc@3.17.1
```

Install or Load a Spack Package

Steps	Already Installed in Spack/E4S	Available via Spack/E4S
1	`spack find -v`	`spack list`
	Match package config. and compiler	Search list of ~6200 avail. packages
2	`spack load <package>`	`spack info <package>`
	Load desired package/package variant into environment	Get information about package options
3		`spack install <package>`
		Installs the package into Spack
4		`spack load <package>`

Extra Step: `spack load --sh <package>`
Shows the modifications made to your environment when a package is loaded.

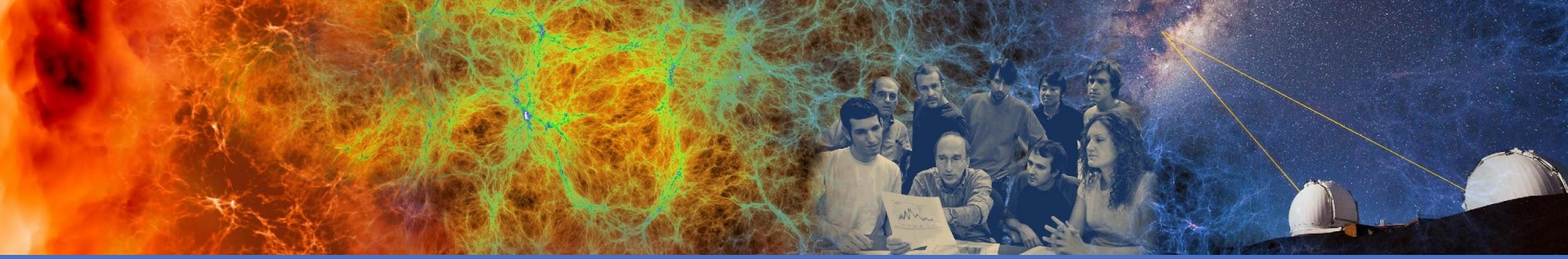
Example: Install a Dependency with Spack

```
epalmer> # In this example, suppose I am compiling a code that depends on a
 package or version of a package that is not currently installed as a module
 on Perlmutter. To demonstrate, I will compile a code that requires
```

Suppose my example code, `needs_hypre.cpp`, requires Hypre.

Example: Install a Dependency with Spack

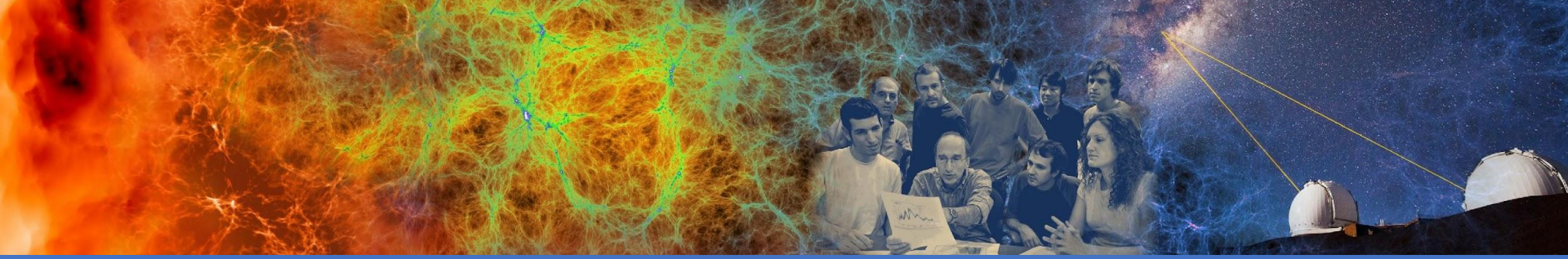
Suppose my example code, `needs_hypre.cpp`, requires Hypre.



Key Suggestions:

- Use `module spider`
- The compiler wrappers: `CC`, `cc`, and `ftn`, add convenience
- Additional software is available via Spack/E4S
- Modify your environment with:

```
source config_myenv.sh
```



Thanks for your Attention!

Documentation for Programming Environments and Compilation:

- Modules – https://docs.nersc.gov/environment/lmod/
- Programming Environment & Compilers - https://docs.nersc.gov/systems/perlmutter/#compilingbuilding-software
- Shell Scripts – https://docs.nersc.gov/environment/shell_startup/
- Spack – https://docs.nersc.gov/development/build-tools/spack/
- E4S – https://docs.nersc.gov/applications/e4s/perlmutter/22.05/

More questions? Need help? ... **http://help.nersc.gov/**