Installing and Using Software on FASRC Cluster

Plamen Krastev, PhD

Harvard - FAS Research Computing

Objectives

- To advise you on the best practices for using software applications on the FASRC cluster
- To provide the basic knowledge required for installing your software applications on the FASRC cluster

Overview

- Software modules
- Installing `Python`, `R` and `Julia` packages
- Using precompiled (numeric, I/O, etc) software libraries
- `Singularity` basics
- Installing software yourself
  - `Spack`
- RC VDI portal (GUI apps, such as MATLAB, RStudio Server, Jupyter Notebook, Remote Desktop, and more)

LMOD Module System

LMOD: ENVIRONMENTAL MODULES SYSTEM

https://www.tacc.utexas.edu/research-development/tacc-projects/lmod

- Environment Modules provide a convenient way to dynamically modify the user's environment through module files (Lua-based scripting files)
- Software module files define various environment variables, such as `PATH`, `LD_LIBRARY_PATH`, `LIBRARY_PATH`, `CPATH`, `FPATH`, etc., so that executables, header files and required libraries can be found by the specific software application
- Software can be "loaded" and "unloaded" dynamically
- Hierarchies (incremental module loading/unloading) prevent software conflicts
- Various software versions can coexist
- Harvard modified LMOD software modules system: `HeLMOD` (Harvard Extensions for Lmod deployment) - https://github.com/fasrc/helmod

HeLmod: software modules

Compilers: `gcc`, `Intel`

CUDA and cuDNN

MPI Libraries: `OpenMPI`, `Mpich`, `Intel-MPI`

Basic numerical and I/O libraries (e.g., `gsl`, `HDF5`, `NetCDF`)

Common software packages (e.g., `Python`, `R`, `LAMMPS`, `GAMESS`, `GROMACS`, `CP2K`, etc., but customization is left to the user through `spack`)

Commercial software (e.g., `MATLAB`, `Mathematica`, `IDL`, `Stata`, `SAS`)

Software Modules

- Software modules basics

```
module load gcc/14.2.0-fasrc01          # Load Compiler
module load openmpi/5.0.5-fasrc01       # Load MPI library
module load netcdf-c/4.9.2-fasrc06      # Load NetCDF Library

module list                             # List loaded modules
module purge                            # Unload all modules
module display netcdf-c/4.9.2-fasrc06 # Display module environment

module avail                            # Show ALL available modules in the MODULEPATH
module avail netcdf-c                   # Show available netcdf-c modules
```

- Finding modules

```
module spider openmpi                   # Find available OpenMPI modules
module spider openmpi/5.0.5-fasrc01     # List more information including how to load it
```

Modules: How do they work? (1)

```
[pkrastev@holy7c24103 ~]$ module load gcc/14.2.0-fasrc01
[pkrastev@holy7c24103 ~]$ which gcc
/n/sw/helmod-rocky8/apps/Core/gcc/14.2.0-fasrc01/bin/gcc
[pkrastev@holy7c24103 ~]$ gcc --version
gcc (GCC) 14.2.0
[pkrastev@holy7c24103 ~]$ ls -l /n/sw/helmod-rocky8/apps/Core/gcc/14.2.0-fasrc01/
total 2760
drwxr-xr-x. 2 root root     641 Sep  6 12:05 bin
-rw-r--r--. 1 root root  684603 Aug  1 04:18 ChangeLog
-rw-r--r--. 1 root root   18002 Aug  1 04:17 COPYING
drwxr-xr-x. 3 root root      21 Sep  6 12:04 include
drwxr-xr-x. 2 root root     330 Sep  6 12:05 INSTALL
drwxr-xr-x. 2 root root    2053 Sep  6 12:05 lib
drwxr-xr-x. 3 root root    2748 Sep  6 12:05 lib64
drwxr-xr-x. 3 root root      21 Sep  6 12:04 libexec
-rw-r--r--. 1 root root    2858 Sep  6 11:53 modulefile.lua
-rw-r--r--. 1 root root 1186346 Aug  1 04:19 NEWS
-rw-r--r--. 1 root root    1026 Aug  1 04:17 README
drwxr-xr-x. 6 root root      95 Sep  6 12:05 share
```

Modules: How do they work? (2)

```
[pkrastev@holy7c24103 ~]$ module display gcc/14.2.0-fasrc01
...
   /n/sw/helmod-rocky8/modulefiles/Core/gcc/14.2.0-fasrc01.lua:
...
whatis("Name: gcc")
whatis("Version: 14.2.0-fasrc01")
whatis("Description: the GNU Compiler Collection")
setenv("CC","gcc")
setenv("CXX","g++")
setenv("FC","gfortran")
setenv("F77","gfortran")
setenv("GCC_HOME","/n/sw/helmod-rocky8/apps/Core/gcc/14.2.0-fasrc01")
setenv("GCC_LIB","/n/sw/helmod-rocky8/apps/Core/gcc/14.2.0-fasrc01/lib64")
setenv("GCC_INCLUDE","/n/sw/helmod-rocky8/apps/Core/gcc/14.2.0-fasrc01/include")
prepend_path("PATH","/n/sw/helmod-rocky8/apps/Core/gcc/14.2.0-fasrc01/bin")
prepend_path("CPATH","/n/sw/helmod-rocky8/apps/Core/gcc/14.2.0-fasrc01/lib64/gcc/x86_64-pc-linux-gnu/14.2.0/include")
prepend_path("CPATH","/n/sw/helmod-rocky8/apps/Core/gcc/14.2.0-fasrc01/lib64/gcc/x86_64-pc-linux-gnu/14.2.0/install-tools/include")
prepend_path("CPATH","/n/sw/helmod-rocky8/apps/Core/gcc/14.2.0-fasrc01/lib64/gcc/x86_64-pc-linux-gnu/14.2.0/plugin/include")
prepend_path("CPATH","/n/sw/helmod-rocky8/apps/Core/gcc/14.2.0-fasrc01/include")
prepend_path("FPATH","/n/sw/helmod-rocky8/apps/Core/gcc/14.2.0-fasrc01/lib64/gcc/x86_64-pc-linux-gnu/14.2.0/include")
prepend_path("FPATH","/n/sw/helmod-rocky8/apps/Core/gcc/14.2.0-fasrc01/lib64/gcc/x86_64-pc-linux-gnu/14.2.0/install-tools/include")
prepend_path("FPATH","/n/sw/helmod-rocky8/apps/Core/gcc/14.2.0-fasrc01/lib64/gcc/x86_64-pc-linux-gnu/14.2.0/plugin/include")
prepend_path("FPATH","/n/sw/helmod-rocky8/apps/Core/gcc/14.2.0-fasrc01/include")
prepend_path("INFOPATH","/n/sw/helmod-rocky8/apps/Core/gcc/14.2.0-fasrc01/share/info")
prepend_path("LD_LIBRARY_PATH","/n/sw/helmod-rocky8/apps/Core/gcc/14.2.0-fasrc01/lib")
prepend_path("LIBRARY_PATH","/n/sw/helmod-rocky8/apps/Core/gcc/14.2.0-fasrc01/lib")
prepend_path("LD_LIBRARY_PATH","/n/sw/helmod-rocky8/apps/Core/gcc/14.2.0-fasrc01/lib64")
prepend_path("LIBRARY_PATH","/n/sw/helmod-rocky8/apps/Core/gcc/14.2.0-fasrc01/lib64")
prepend_path("MANPATH","/n/sw/helmod-rocky8/apps/Core/gcc/14.2.0-fasrc01/share/man")
prepend_path("MODULEPATH","/n/sw/helmod-rocky8/modulefiles/Comp/gcc/14.2.0-fasrc01")
setenv("FASRCSW_COMP_NAME","gcc")
setenv("FASRCSW_COMP_VERSION","14.2.0")
setenv("FASRCSW_COMP_RELEASE","fasrc01")
family("Comp")
```

Modules: Hierarchies (1)

Use of groupings is important for proper working of scientific applications.

- Applications built with specific compiler flavor/version need to be linked with libraries compiled with the same compiler flavor and version
- **Message Passing Interface (MPI)** library allows for communication between tasks on a distributed memory computers with many processors
- Parallel applications and libraries must be built with a matching MPI library and compiler

Instead of using a flat namespace, we can use module hierarchies:

- Simple technique because once users chose a compiler and MPI implementation, they can only load modules that match that compiler and MPI implementation
- FASRC follow's TACC's convention:

```
# MODULEPATH_ROOT=/n/sw/helmod-rocky8/modulefiles
${MODULEPATH_ROOT}/{Core, Comp, MPI}
```

Modules: Hierarchies (2)

```
[pkrastev@holy7c24103 ~]$ module load gcc/14.2.0-fasrc01 openmpi/5.0.5-fasrc01 hdf5/1.14.4-fasrc01
[pkrastev@holy7c24103 ~]$ module spider hdf5/1.14.4-fasrc01

----------------------------------------------------------------------------------------------------------------
  hdf5: hdf5/1.14.0-fasrc01
----------------------------------------------------------------------------------------------------------------
 Description:
     HDF5 is a data model, library, and file format for storing and managing data.


   You will need to load all module(s) on any one of the lines below before the "hdf5/1.14.4-fasrc01" module is available to load.

     gcc/14.2.0-fasrc01  mpich/4.2.2-fasrc01
     gcc/14.2.0-fasrc01  openmpi/5.0.5-fasrc01
     intel/24.0.1-fasrc01  intelmpi/2021.11-fasrc01
     intel/24.2.1-fasrc01  mpich/4.2.2-fasrc01
     intel/24.2.1-fasrc01  openmpi/5.0.5-fasrc01

   Help:
     hdf5/1.14.4-fasrc01
     HDF5 is a data model, library, and file format for storing and managing data.
```

Python Programs (1)

Mamba replacing Anaconda: fast, robust, and cross-platform package manager

Python 3

```
[pkrastev@holy7c24103 ~]$ module load python/3.12.5-fasrc01
[pkrastev@holy7c24103 ~]$ module list

Currently Loaded Modules:
  1) Miniforge3/24.7.1-fasrc01   2) python/3.12.5-fasrc01
```

Python 2

```
[pkrastev@holy7c24103 ~]$ module load python/2.7.16-fasrc01
[pkrastev@holy7c24103 ~]$ module list

Currently Loaded Modules:
  1) Anaconda2/2019.10-fasrc01   2) python/2.7.16-fasrc01
```

Python Programs (2)

- Mamba is a tool to manage `conda` environments
- Mamba uses the same commands and configuration options as `conda`
- You can swap almost all commands between `conda` & `mamba`
- https://docs.rc.fas.harvard.edu/kb/python-package-installation

```
# Load a Python module
module load python/3.12.5-fasrc01
# Create local conda environment in ~/.conda/envs/ENV_NAME
mamba create -n ENV_NAME PACKAGE_LIST
# Use the new environment
mamba activate ENV_NAME
# Install a new package named MY_PACKAGE
mamba install MY_PACKAGE
# If the package is not available with conda/mamba use pip
pip install MY_PACKAGE
# If you have problems updating a package first remove it
mamba uninstall MY_PACKAGE
# Deactivate the environment
mamba deactivate
```

Python Programs (3)

For optimal performance it is recommended to use the `--prefix` option to create / relocate your conda environments to your LAB space, e.g.,

`/n/holylabs/LABS/<PI_LAB>/Lab`

or

`/n/holylabs/LABS/<PI_LAB>/Users/${USER}`

```
# Load a Python module, e.g.,
module load python/3.12.5-fasrc01

# Create a conda environment in LAB space, e.g.,
mamba install -y --prefix=/n/holylabs/LABS/<PI_LAB>/Lab/conda/<ENV_NAME> PACKAGE_LIST

# Activate the conda environment
mamba activate /n/holylabs/LABS/<PI_LAB>/Lab/conda/<ENV_NAME>
```

R Programs

- When loading R from the LMOD module system, only basic R packages are loaded in your environment
- Use the `R_LIBS_USER` environment variable to specify location of customized R package installations
- https://docs.rc.fas.harvard.edu/kb/r-and-rstudio

```
# Load R module, e.g.,
module load R/4.4.1-fasrc01

# Set R_LIBS_USER to your location for R packages, e.g.,
export R_LIBS_USER=$HOME/apps/R:$R_LIBS_USER

# Start R
R

# Inside R, install the desired package, e.g.,
> install.packages("Rcpp")
```

Julia

```
# Use lab storage, e.g.,
[jharvard@holy7c12104 ~]$ cd /n/holylabs/LABS/jharvard_lab/Users/jharvard/software/

# Download julia and extract
[jharvard@holy7c12104 software]$ wget  \
https://julialang-s3.julialang.org/bin/linux/x64/1.10/julia-1.10.5-linux-x86_64.tar.gz
[jharvard@holy7c12104 software]$ tar xvfz julia-1.10.5-linux-x86_64.tar.gz

# Add julia to path
[jharvard@holy7c12104 julia-1.10.5]$ export PATH=$PATH:/n/holylabs/LABS/jharvard_lab/Users/jharvard/software/julia-1.10.5/bin

[jharvard@holy7c12104 julia-1.10.5]$ julia
               _
   _       _ _(_)_     |  Documentation: https://docs.julialang.org
  (_)     | (_) (_)    |
   _ _   _| |_  __ _   |  Type "?" for help, "]?" for Pkg help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 1.10.5 (2024-08-27)
 _/ |\__'_|_|_|\__'_|  |  Official https://julialang.org/ release
|__/                   |

julia>
```

https://docs.rc.fas.harvard.edu/kb/julia

Using Software Libraries (1)

- Software libraries allow you to use precompiled functions and routines in your applications.
- Many are already installed on the cluster, e.g., `GSL`, `BLAS`, `LAPACK`, `NetCDF`, `HDF5`, `FFTW`, `MKL`, `BOOST`, etc., and are available as software modules.
- Software libraries could also be a part of the OS and are typically located in `/lib` and `/lib64`
- Linking to specific libraries can be done by setting `-l` and `-L` flags, e.g.,

```
# Load required software modules, e.g.,
module load gsl/2.8-fasrc01

# Compile and link the application, e.g.,
gcc -o gsl_int_test.x gsl_int_test.c -O2 -lm -lgsl -lgslcblas
```

https://github.com/fasrc/User_Codes/tree/master/Libraries/GSL

Using Software Libraries (2)

```
#=============================================================
# Make file for gsl_int_test.c
#=============================================================
CFLAGS   = -c -O2
COMPILER = gcc
PRO      = gsl_int_test
OBJECTS  = gsl_int_test.o

LINK_GSL = -lm -lgsl -lgslcblas

${PRO}.x : $(OBJECTS)
	$(COMPILER) -o ${PRO}.x $(OBJECTS) $(LINK_GSL)

%.o : %.c
	$(COMPILER) $(CFLAGS) $(<F)

clean :
	rm -rf *.o *.x *.mod
```

https://github.com/fasrc/User_Codes/tree/master/Libraries/GSL

Singularity (1)

Singularity provides a container runtime and an ecosystem for managing images that is suitable for multi-tenant systems and HPC environments.

Important aspects:

- Each application will have its own container
- Containers are not fully isolated (e.g., host network is available)
- Users have the same *uid* and *gid* when running an application
- You can build containers from:
  - from existing container in *SingularityCE* container library
  - from existing container in Docker Hub
  - from *SingularityCE* definition file on *Sylabs* cloud
  - **NEW:** from *SingularityCE* definition file and `proot` directly on Cannon

https://docs.rc.fas.harvard.edu/kb/singularity-on-the-cluster/

Singularity (2)

Running TensorFlow on a CPU node:

```
# --- Start an interactive session ---
[login-node  ]$ salloc -p test --mem=4G -N 1 -t 60
# --- cd to your SCRATCH folder ---
[compute-node]$ cd $SCRATCH/your_lab/your_user/
# --- Pull the latest TF version from the Docker registry ---
[compute-node]$ singularity pull --name tf2.17_cpu.simg docker://tensorflow/tensorflow:2.17.0
# --- Launch Python and print the TF version ---
[compute-node]$ singularity exec tf2.17_cpu.simg python
… (omitted output)
>>> import tensorflow as tf
>>> print(tf.__version__)
2.17.0
# --- Get examples from keras.io ---
[compute-node]$ git clone https://github.com/keras-team/keras-io.git
# --- Execute the code ---
[compute-node]$ singularity exec tf2.17_cpu.simg python ./keras-io/examples/vision/mnist_convnet.py
… (omitted output)
Test loss: 0.023163778707385063
Test accuracy: 0.9923999905586243
```

Singularity (3)

Running TensorFlow on a GPU node:

```
# --- Start an interactive session on a partition with GPUs, e.g.,
[login-node  ]$ salloc -p gpu_test --gres=gpu:1 --mem=4G -N 1 -t 60
# --- cd to your SCRATCH folder ---
[compute-node]$ cd $SCRATCH/your_lab/your_user/
# --- Pull the latest TF GPU version from the Docker registry ---
[compute-node]$ singularity pull --name tf2.17_gpu.simg docker://tensorflow/tensorflow:2.17.0-gpu
# --- Get examples from keras.io ---
[compute-node]$ git clone https://github.com/keras-team/keras-io.git
# --- Execute the code ---
[compute-node]$ singularity exec --nv tf2.17_gpu.simg python ./keras-io/examples/vision/mnist_convnet.py
… (omitted output)
Test loss: 0.0239315405488014 22
Test accuracy: 0.9915000200271606
```

Singularity (4)

Examples: pulling images from repositories

- **Preparation (start an interactive session and cd to $SCRATCH directory):**

```
[login-node  ]$ salloc -p gpu_test --gres=gpu:1 --mem=4G -N 1 -t 60
[compute-node]$ cd $SCRATCH/your_lab/your_user/
```

- **Pulling from Docker:**

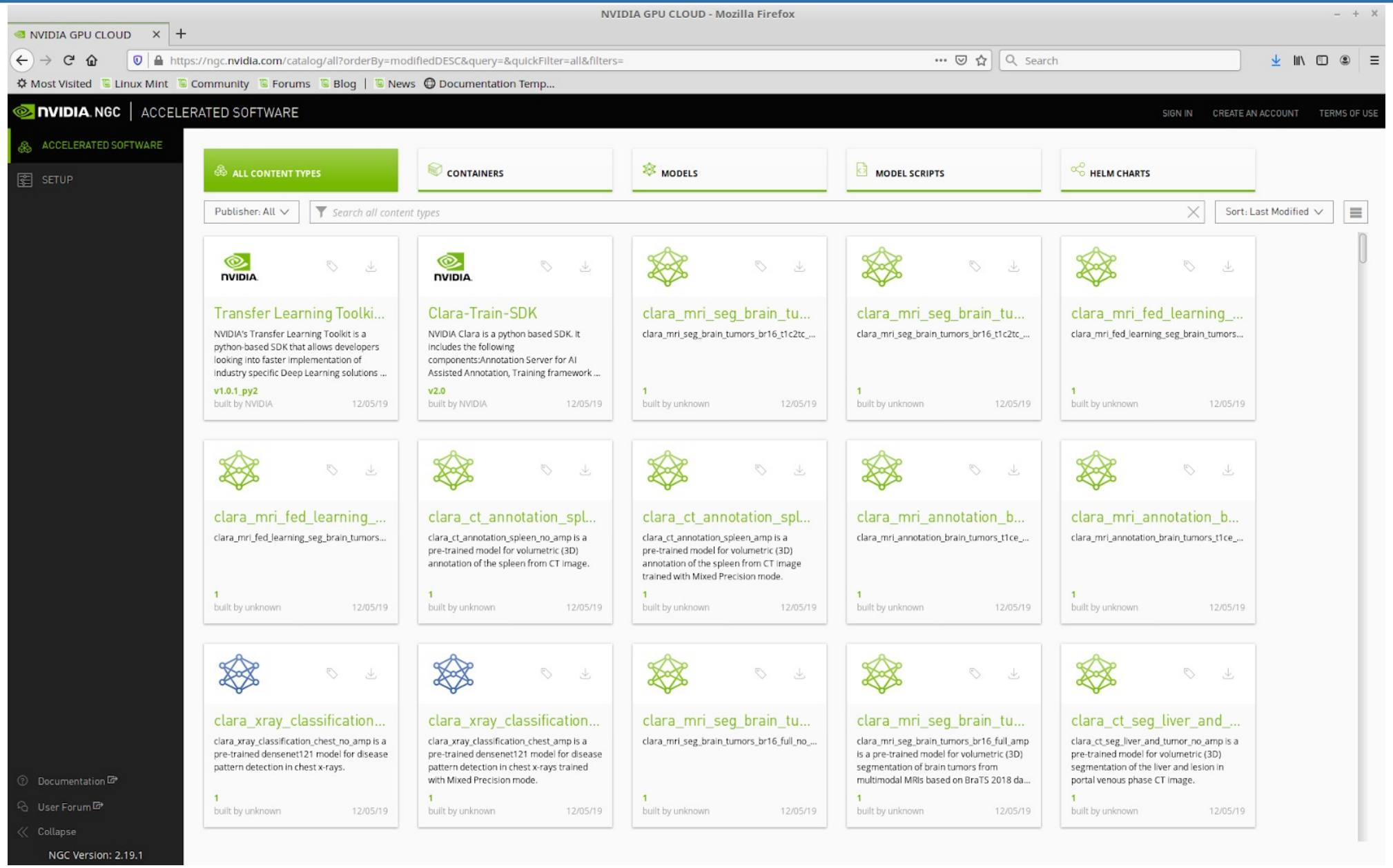
```
[compute-node]$ singularity pull docker://tensorflow/tensorflow:latest
```

- **Pulling from sylab / library -- https://cloud.sylabs.io/library**

```
[compute-node]$ singularity pull library://library/default/ubuntu:21.04
```

- **Pulling from NVIDIA's NGC registry - https://catalog.ngc.nvidia.com**

```
[compute-node]$ singularity pull docker://nvcr.io/nvidia/tensorflow:24.09-tf2-py3
[compute-node]$ singularity exec --nv tensorflow_24.09-tf2-py3.sif python
Python 3.10.12 (main, Jul 29 2024, 16:56:48) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
>>> print(tf.__version__)
2.16.1
```


NVIDIA GPU CLOUD - Mozilla Firefox
NVIDIA GPU CLOUD
https://ngc.nvidia.com/catalog/all?orderBy=modifiedDESC&query=&quickFilter=all&filters=
Search
Most Visited
Linux Mint
Community
Forums
Blog
News
Documentation Temp...
NVIDIA NGC | ACCELERATED SOFTWARE
SIGN IN
CREATE AN ACCOUNT
TERMS OF USE
ACCELERATED SOFTWARE
SETUP
ALL CONTENT TYPES
CONTAINERS
MODELS
MODEL SCRIPTS
HELM CHARTS
Publisher: All
Search all content types
Sort: Last Modified
Transfer Learning Toolki...
NVIDIA's Transfer Learning Toolkit is a python-based SDK that allows developers looking into faster implementation of industry specific Deep Learning solutions ...
v1.0.1_py2
built by NVIDIA 12/05/19
Clara-Train-SDK
NVIDIA Clara is a python based SDK. It includes the following components:Annotation Server for AI Assisted Annotation, Training framework ...
v2.0
built by NVIDIA 12/05/19
clara_mri_seg_brain_tu...
clara_mri_seg_brain_tumors_br16_t1c2tc_...
1
built by unknown 12/05/19
clara_mri_seg_brain_tu...
clara_mri_seg_brain_tumors_br16_t1c2tc_...
1
built by unknown 12/05/19
clara_mri_fed_learning_...
clara_mri_fed_learning_seg_brain_tumors...
1
built by unknown 12/05/19
clara_mri_fed_learning_...
clara_mri_fed_learning_seg_brain_tumors...
1
built by unknown 12/05/19
clara_ct_annotation_spl...
clara_ct_annotation_spleen_no_amp is a pre-trained model for volumetric (3D) annotation of the spleen from CT image.
1
built by unknown 12/05/19
clara_ct_annotation_spl...
clara_ct_annotation_spleen_amp is a pre-trained model for volumetric (3D) annotation of the spleen from CT image trained with Mixed Precision mode.
1
built by unknown 12/05/19
clara_mri_annotation_b...
clara_mri_annotation_brain_tumors_t1ce_...
1
built by unknown 12/05/19
clara_mri_annotation_b...
clara_mri_annotation_brain_tumors_t1ce_...
1
built by unknown 12/05/19
clara_xray_classification...
clara_xray_classification_chest_no_amp is a pre-trained densenet121 model for disease pattern detection in chest x-rays.
1
built by unknown 12/05/19
clara_xray_classification...
clara_xray_classification_chest_amp is a pre-trained densenet121 model for disease pattern detection in chest x-rays trained with Mixed Precision mode.
1
built by unknown 12/05/19
clara_mri_seg_brain_tu...
clara_mri_seg_brain_tumors_br16_full_no_...
1
built by unknown 12/05/19
clara_mri_seg_brain_tu...
clara_mri_seg_brain_tumors_br16_full_amp is a pre-trained model for volumetric (3D) segmentation of brain tumors from multimodal MRIs based on BraTS 2018 da...
1
built by unknown 12/05/19
clara_ct_seg_liver_and_...
clara_ct_seg_liver_and_tumor_no_amp is a pre-trained model for volumetric (3D) segmentation of the liver and lesion in portal venous phase CT image.
1
built by unknown 12/05/19
Documentation
User Forum
Collapse
NGC Version: 2.19.1

Singularity (5)

Building Singularity images from *SingularityCE* definition file and `proot` directly on Cannon

```
# Make ~/bin directory
[jharvard@holy2c02302 ~]$ mkdir -p ~/bin

# Change to ~/bin directory, download proot, and change permissions to make it executable
[jharvard@holy2c02302 ~]$ cd ~/bin
[jharvard@holy2c02302 bin]$ curl -LO https://proot.gitlab.io/proot/bin/proot
[jharvard@holy2c02302 bin]$ chmod +x ./proot

# Obtain, or create a singularity definition (def) file and build the image, e.g.,
[jharvard@holy2c02302 ~]$ singularity build tf-2.12.simg tf-2.12.def
INFO:    Using proot to build unprivileged. Not all builds are supported. If build fails, use --remote or --fakeroot.
INFO:    Starting build...
(omitted output)
INFO:    Creating SIF file...
INFO:    Build complete: tf-2.12.simg
```

https://github.com/fasrc/User_Codes/tree/master/Singularity_Containers#build-a-singularityce-container-from-a-singularity-definition-file

Singularity (6)

Example *SingularityCE* def file: **tf-2.12.def**

```
Bootstrap: docker
From: tensorflow/tensorflow:2.12.0-gpu

%post
    pip install --upgrade pip
    pip install matplotlib
    pip install seaborn
    pip install scipy
    pip install scikit-learn
    pip install jupyterlab
    pip install notebook
```

https://github.com/fasrc/User_Codes/blob/master/AI/TensorFlow/tf-2.12.def

Installing Software: Spack

One-time setup

Clone Spack repo in your lab storage (better performance than `$HOME` directory)

Source spack

Install packages with Spack - some software can take a few hours to build

Job submission

Source spack

Load packages/software with Spack

Run code

https://github.com/fasrc/User_Codes/blob/master/Documents/Software/Spack.md

Spack (1)

One-time setup

```
# Request interactive job
[jharvard@rockylogin ~]$ salloc -p test --mem 12g -t 0-04:00 -c 8

# Use lab storage
[jharvard@holy7c12104 ~]$ cd /n/holylabs/LABS/jharvard_lab/Lab/software/

# Clone spack
[jharvard@holy7c12104 software]$ git clone -c feature.manyFiles=true https://github.com/spack/spack.git

# Source spack
[jharvard@holy7c12104 software]$ cd spack/
[jharvard@holy7c12104 spack]$ . share/spack/setup-env.sh

# Install packages
[jharvard@holy7c12104 spack]$ spack install bzip2                  # Install default/latest version
[jharvard@holy7c12104 spack]$ spack install bzip2@1.0.8            # Specify version
[jharvard@holy7c12104 spack]$ spack install zlib@1.2.13%gcc@8.5.0  # Specify version and compiler
```

Spack (2)

One-time setup

```
# List installed packages
$ spack find

# Uninstall packages, e.g,
$ spack uninstall zlib@1.2.13%gcc@8.5.0

# Load spack packages
$ spack load bzip2
$ which bzip2
/home/spack/opt/spack/linux-rocky8-x86_64/gcc-8.5.0/bzip2-1.0.8-qimzsx6zy45aww52i3uowomnsho5muep/bin/bzip2

# List the loaded packages
$ spack find --loaded
-- linux-rocky8-x86_64 / gcc@8.5.0 -----------------------------
bzip2@1.0.8
==> 1 loaded package

# Unload spack packages
$ spack unload
$ spack find --loaded
==> 0 loaded packages
```

Spack (3)

Group Permissions

By default Spack will match your usual file permissions which typically are set up without group write permission. For lab wide installs of Spack though you will want to ensure that it has group write enforced. You can set this by going to the `etc/spack` directory in your Spack installation and adding a file called `packages.yaml` (or editing the exiting one) with the following contents:

```
packages:
  all:
    permissions:
      write: group
      group: jharvard_lab
```

https://spack.readthedocs.io/en/latest/build_settings.html#package-permissions

Spack (4)

Default Architecture

By default Spack will autodetect which architecture your underlying hardware is and build software to match that. However in cases where you are running on heterogeneous hardware it is best to use a more generic flag. You can set this by going to the `etc/spack` directory in your Spack installation and adding a file called `packages.yaml` (or editing the exiting one) with the following contents:

```
packages:
  all:
    target: [x86_64]
```

https://spack.readthedocs.io/en/latest/build_settings.html#package-preferences

Spack (5)

Compiler Configuration

```
# List available compilers
$ spack compilers
==> Available compilers
-- gcc rocky8-x86_64 ---------------------------------------------
gcc@8.5.0

# Load the required compiler software module, e.g.,
$ module load gcc/14.2.0-fasrc01

# Add this GCC compiler version to the spack compilers
$ spack compiler find
==> Added 1 new compiler to ~/.spack/linux/compilers.yaml
    gcc@14.2.0
==> Compilers are defined in the following files:
    ~/.spack/linux/compilers.yaml

$ spack compilers
==> Available compilers
-- gcc rocky8-x86_64 ---------------------------------------------
gcc@8.5.0  gcc@14.2.0
```

https://github.com/fasrc/User_Codes/blob/master/Documents/Software/Spack.md#compiler-configuration

Spack (6)

Compiler Configuration

Modify `~/.spack/linux/compilers.yaml` to read:

```
- compiler:
    spec: gcc@=14.2.0
    paths:
      cc: /n/sw/helmod-rocky8/apps/Core/gcc/14.2.0-fasrc01/bin/gcc
      cxx: /n/sw/helmod-rocky8/apps/Core/gcc/14.2.0-fasrc01/bin/g++
      f77: /n/sw/helmod-rocky8/apps/Core/gcc/14.2.0-fasrc01/bin/gfortran
      fc: /n/sw/helmod-rocky8/apps/Core/gcc/14.2.0-fasrc01/bin/gfortran
    flags: {}
    operating_system: rocky8
    target: x86_64
    modules: [gcc/14.2.0-fasrc01]
    environment: {}
    extra_rpaths: []
```

https://github.com/fasrc/User_Codes/blob/master/Documents/Software/Spack.md#compiler-configuration

Spack (7)

MPI Configuration

```
# Determine the MPI location / prefix
$ module load gcc/14.2.0-fasrc01 openmpi/5.0.5-fasrc01
$ echo $MPI_HOME
/n/sw/helmod-rocky8/apps/Comp/gcc/14.2.0-fasrc01/openmpi/5.0.5-fasrc01

# Edit manually the packages configuration file ~/.spack/packages.yaml
# Include the following content:
packages:
  openmpi:
    externals:
    - spec: openmpi@5.0.5%gcc@14.2.0
      prefix: /n/sw/helmod-rocky8/apps/Comp/gcc/14.2.0-fasrc01/openmpi/5.0.5-fasrc01
    buildable: False

# Example: Build HDF5 version 1.14.0 with gcc@14.2.0 and openmpi@5.0.5
$ module purge
$ spack install hdf5@1.14.0 % gcc@14.2.0 ^ openmpi@5.0.5
```

https://github.com/fasrc/User_Codes/blob/master/Documents/Software/Spack.md#mpi-configuration

Installing R packages with Spack

```
# request interactive job
[jharvard@rockylogin ~]$ salloc -p rocky --mem 12g -t 0-04:00 -c 8

# use lab storage
[jharvard@holy7c12104 ~]$ cd /n/holylabs/LABS/jharvard_lab/Lab/software/spack

# source spack
[jharvard@holy7c12104 spack]$ . share/spack/setup-env.sh

# install R packages with spack
[jharvard@holy2c02302 spack]$ spack install r-rgdal

# load spack packages
[jharvard@holy2c02302 spack]$ spack load r-rgdal

# launch R and load libraries
[jharvard@holy2c02302 spack]$ R
> library(rgdal)
```

https://github.com/fasrc/User_Codes/blob/master/Languages/R/R_packages_with_spack.md

SLURM jobs with Spack

```
#!/bin/bash
#SBATCH -J r_spack              # Job name
#SBATCH -c 1                    # Number of cores (--cpus-per-task)
#SBATCH -t 0-00:10              # Runtime in D-HH:MM, minimum of 10 minutes
#SBATCH -p test                 # Partition to submit to
#SBATCH --mem=4g                # Memory for all cores in GB (see also --mem-per-cpu)
#SBATCH -o myoutput_%j.out      # File to which STDOUT will be written, %j inserts jobid
#SBATCH -e myerrors_%j.err      # File to which STDERR will be written, %j inserts jobid

# source spack
. /n/holylabs/LABS/jharvard_lab/Users/jharvard/spack/share/spack/setup-env.sh

# load spack packages
spack load r-codetools
spack load r-rgdal
spack load r-raster
spack load r-terra

# run R code
Rscript --vanilla r_spack_load_libs.R > r_spack_load_libs.Rout
```

RC VDI Portal

- Interactive computing portal
  https://vdi.rc.fas.harvard.edu/pun/sys/dashboard
- Need to be on RC VPN
- Provides GUI apps, such as
  - Remote Desktop
  - Rstudio Server
  - Jupyter Notebook
  - MATLAB
  - COMSOL
  - and more

Request Help - Resources

- https://docs.rc.fas.harvard.edu/kb/support/
  - Portal
    - http://portal.rc.fas.harvard.edu/rcrt/submit_ticket
  - Email
    - rchelp@rc.fas.harvard.edu
  - Office Hours
    - Wednesday noon-3pm https://harvard.zoom.us/j/97676134704
  - Training Materials
    - https://docs.rc.fas.harvard.edu/kb/training-materials/
  - Training Calendar
    - https://www.rc.fas.harvard.edu/upcoming-training/

Thank you! Questions? Comments?

Plamen Krastev, PhD

Harvard - FAS Research Computing