JUPYTERHUB, PYTHON, CONTAINERS & MORE:
INTRO TO USING POPULAR OPEN SOURCE TOOLS ON LC

PRESENTED BY
JANE HERRIMAN
LIVERMORE COMPUTING

12/8/21
REASON FOR THIS WORKSHOP

- Working with your favorite software/apps/tools/languages/packages/etc. is different in a cluster/HPC/shared environment than on a desktop!

- We want to give you a few different options for smoothing out your workflow and getting access to the tools you want!
AGENDA

- A few popular open source languages
  - **Python** on LC
    - Python virtual environments
    - How to install python packages: pytorch, tensorflow
  - **Julia** on LC
  - **R** on LC
  - **JupyterHub** (out of the box & with custom kernels)
- How to use **containers**
- How to use **spack**, a package manager
- How to know what you can install
A few popular open source languages

- Python on LC
  - Python virtual environments
  - How to install python packages: pytorch, tensorflow
- Julia on LC
- R on LC
- JupyterHub (out of the box & with custom kernels)
- How to use containers
- How to use spack, a package manager
- How to know what you can install
Use module system to use the python version of your choice!

```
janeh@flash21:~$ which python3
/usr/tce/bin/python3
janeh@flash21:~$ module avail python

----------------------------------------- /usr/tce/modulefiles/Core -----------------------------------------
  python/2.7.11       python/2.7.16 (D)       python/3.6.4
  python/2.7.13       python/3.5.1           python/3.7.2
  python/2.7.14       python/3.6.0           python/3.8.2

Where:
  D: Default Module

Use "module spider" to find all possible modules and extensions. Use "module keyword key1 key2 ..." to search for all possible modules matching any of the "keys".
```

```
janeh@flash21:~$ module load python/3.8.2
janeh@flash21:~$ which python3
/usr/tce/packages/python/python-3.8.2/bin/python3
```
AGENDA

- A few popular open source languages
  - Python on LC
    - **Python virtual environments**
      - How to install python packages: pytorch, tensorflow
  - Julia on LC
  - R on LC
- JupyterHub (out of the box & with custom kernels)
- How to use containers
- How to use spack, a package manager
- How to know what you can install
`virtualenv` binary associated with the python module you’ve loaded is in your path now.

```bash
janeh@flash21:~$ which python3
/usr/tce/packages/python/python-3.8.2/bin/python3
janeh@flash21:~$ which virtualenv
/usr/tce/packages/python/python-3.8.2/bin/virtualenv
```

You can create your own virtual environment via `virtualenv --system-site-package <venv_dir_path>`

```bash
janeh@flash21:~$ virtualenv --system-site-package ~/myvenv
Using base prefix '/collab/usr/gapps/python/build/spack-toss3.4/opt/spack/linux-rhel7-ivybridge/gcc-4.9.3/python-3.8.2-6me27g5yfvrxcmax25kovzjbf22vt'
New python executable in /g/g0/janeh/myvenv/bin/python3.8
Also creating executable in /g/g0/janeh/myvenv/bin/python
Installing setuptools, pip, wheel...
done.
```

https://hpc.llnl.gov/software/development-environment-software/python
Your newly created virtual environment has its own python binaries, pip (package manager), etc.

This is an environment **you** can edit!

To start working with your virtual environment, you need to activate it via `source <venv_dir_path>/bin/activate`
AGENDA

- A few popular open source languages
  - Python on LC
    - Python virtual environments
  - How to install python packages: pytorch, tensorflow
  - Julia on LC
  - R on LC
  - JupyterHub (out of the box & with custom kernels)
- How to use containers
- How to use spack, a package manager
- How to know what you can install
Having created and activated your python virtual environment, you can now use your default `pip` to install tensorflow:

```
(myvenv) janeh@flash21:~/myvenv$ which pip
~/myvenv/bin/pip
(myvenv) janeh@flash21:~/myvenv$ pip install intel-tensorflow
```
Having created and activated your python virtual environment, you can now use your default `pip` to install tensorflow:

```
(myvenv) janeh@flash21:~/myvenv$ which pip
~/myvenv/bin/pip
(myvenv) janeh@flash21:~/myvenv$ pip install intel-tensorflow
```

When I did this, I got a complaint about a dependency – nbformat:

```
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts.
plotly 2.2.0 requires nbformat>=4.2, which is not installed.
```

Installing the correct version of nbformat fixed this:

```
(myvenv) janeh@flash21:~/myvenv$ pip install nbformat==4.2
```
After installation, check you can import and use tensorflow:

```bash
(myvenv) janeh@flash21:~/myvenv$ python3
Python 3.8.2 (default, Mar 18 2020, 12:19:58)
[GCC 4.9.3] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow
>>> msg = tensorflow.constant("TensorFlow 2.0 Hello World")
2021-05-26 19:57:17.997095: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
>>> tensorflow.print(msg)
TensorFlow 2.0 Hello World
```

More docs on installation and testing @

https://lc.llnl.gov/confluence/display/LC/TensorFlow+in+LC
Similarly, using your virtual environment’s pip, install pytorch:

```bash
(myvenv) janel@flash21:~/myvenv$ which pip
~/myvenv/bin/pip
(myvenv) janel@flash21:~/myvenv$ pip install torch torchvision
```
INSTALLING PYTHON PACKAGES — PYTORCH (X86)

Similarly, using your virtual environment’s pip, install pytorch:

```bash
(myvenv) janeh@flash21:~/.myvenv$ which pip
~/.myvenv/bin/pip
```

```bash
(myvenv) janeh@flash21:~/.myvenv$ pip install torch torchvision
```

And check that installation is successful:

```python
Successfully installed torch-1.8.1 torchvision-0.9.1
(myvenv) janeh@flash21:~/.myvenv$ python3
Python 3.8.2 (default, Mar 18 2020, 12:19:58)
[GCC 4.9.3] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
>>> x = torch.rand(5, 3)
>>> print(x)
tensor([[0.2729, 0.2918, 0.0528],
        [0.8790, 0.7048, 0.7278],
        [0.2984, 0.0410, 0.3625],
        [0.6315, 0.2937, 0.9328],
        [0.4661, 0.8340, 0.9212]])
```

https://lc.llnl.gov/confluence/display/LC/PyTorch+in+LC
janeh@lassen708:/usr/workspace/janeh$ mkdir -p opence
janeh@lassen708:/usr/workspace/janeh$ tar -xzf /collab/usr/global/tools/opence/${SYS_TYPE}/opence.tar.gz -C opence

janeh@lassen708:/usr/workspace/janeh$ source opence/bin/activate (opence) janeh@lassen708:/usr/workspace/janeh$ conda-unpack

(opence) janeh@lassen708:/usr/workspace/janeh$ python3
Python 3.7.10 (default, Feb 26 2021, 19:30:21)
[ GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
>>> import tensorflow
2021-12-07 14:56:00.454139: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcudart.so.10.2

https://lc.llnl.gov/confluence/display/LC/Deep+Learning+in+LC

https://lc.llnl.gov/confluence/display/LC/2021/03/08/Open-CE+for+Lassen
AGENDA

- A few popular open source languages
  - Python on LC
    - Python virtual environments
    - How to install python packages: pytorch, tensorflow
  - Julia on LC
  - R on LC
    - JupyterHub (out of the box & with custom kernels)
  - How to use containers
  - How to use spack, a package manager
  - How to know what you can install
HAVE YOU USED JULIA BEFORE?

A. Yes, a lot

B. Yes, a little

C. Not yet
To add this directory to your PATH:

```bash
export PATH=$PATH:/usr/gapps/julia/bin
```

```csh
setenv PATH $PATH:/usr/gapps/julia/bin
```
janeh@quartz1154:/usr/gapps/julia/bin$ ./.julia

-__-     -(_)-_     | Documentation: https://docs.julialang.org
(_)     |  (_)(_)     | Type "?" for help, "?" for Pkg help.
  _ _ _ _ _| | _ _ _ _ | Version 1.6.3 (2021-09-23)
   |   |   |   |_/ `_` | Official https://julialang.org/ release
  `/ \___'-` _ _ _\___' _
  `/\___/

(@v1.6) pkg> add Example
  Updating registry at `~/.julia/registries/General`
  Updating git-repo `https://github.com/JuliaRegistries/General.git`
  Resolving package versions...
  Updating `~/.julia/environments/v1.6/Project.toml`
[7876af07] + Example v0.5.3
  Updating `~/.julia/environments/v1.6/Manifest.toml`
[7876af07] + Example v0.5.3

Precompiling project...
  1 dependency successfully precompiled in 3 seconds

julia> using Example

julia> hello("LLNL")
"Hello, LLNL"
AGENDA

- A few popular open source languages
  - Python on LC
    - Python virtual environments
    - How to install python packages: pytorch, tensorflow
  - Julia on LC
  - **R on LC**
    - JupyterHub (out of the box & with custom kernels)
  - How to use containers
  - How to use spack, a package manager
  - How to know what you can install
HAVE YOU USED R BEFORE?

A. Yes, a lot
B. Yes, a little
C. Not yet
R v3.6 lives on LC machines at `:/usr/bin/R`:

```
janeh@flash21:~$ which R
/usr/bin/R
janeh@flash21:~$ R

R version 3.6.0 (2019-04-26) -- "Planting of a Tree"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-redhat-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
```
To customize your environment/add packages, `install.packages('<package name>')`.

```r
> install.packages('IRkernel')
Installing package into '/usr/lib64/R/library'
(as 'lib' is unspecified)
Warning in install.packages("IRkernel") :
  'lib = "/usr/lib64/R/library"' is not writable
Would you like to use a personal library instead? (yes/No/cancel) y
Would you like to create a personal library
'~/R/x86_64-redhat-linux-gnu-library/3.6'
to install packages into? (yes/No/cancel) y
```

You’ll be prompted to create a personal library where you can install packages – similar to how we created a virtual environment to install python packages.
AGENDA

- A few popular open source languages
  - Python on LC
    - Python virtual environments
    - How to install python packages: pytorch, tensorflow
  - Julia on LC
  - R on LC
- JupyterHub (out of the box & with custom kernels)
- How to use containers
- How to use spack, a package manager
- How to know what you can install
HAVE YOU USED JUPYTER NOTEBOOKS BEFORE?

A. Yes, a lot
B. Yes, a little
C. Not yet
On the CZ, log in to JupyterHub at https://lc.llnl.gov/jupyter

After authenticating with your pin and RSA token, you’ll be taken to a screen where you can select a server.
Selecting a machine and hitting “Start server” should take you to a view of your home directory with a heading like
Selecting a machine and hitting “Start server” should take you to a view of your home directory.

You can now create a “New” notebook, as below, and select a kernel. “Python 3” should be available by default!
See

https://hpc.llnl.gov/services/jupyter

for more info on JupyterHub!
AGENDA

- A few popular open source languages
  - Python on LC
    - Python virtual environments
    - How to install python packages: pytorch, tensorflow
  - Julia on LC
  - R on LC
- JupyterHub (out of the box & with custom kernels)
- How to use containers
- How to use spack, a package manager
- How to know what you can install
CUSTOM JUPYTERHUB KERNELS

We’ll talk about creating custom kernels for:

- Python
- Julia
- R

For each of these, you’ll need to:

- Install a package (ipykernel, IJulia, or IRkernel)
- Check for & possibly create `~/.local/share/jupyter/kernels/<Directory for custom kernel name>/kernel.json`
We’ll talk about creating custom kernels for:

- **Python**
- **Julia**
- **R**

For each of these, you’ll need to:

- Install a package (ipykernel, IJulia, or IRkernel)
- Check for & possibly create `~/.local/share/jupyter/kernels/<Directory for custom kernel name>/kernel.json`
Install ipykernel into your python virtual environment:

```
janeh@flash21:~/myvenv$ source bin/activate
(myvenv) janeh@flash21:~/myvenv$ pip install ipykernel
```
Install your custom kernel to .local in your home directory

```
(myvenv) janel@flash21:~/myvenv$ python3 -m ipykernel install --prefix=${HOME}/.local/ --name 'myvenv_kernel' --display-name 'myvenv_kernel'
```

Installed kerspec myvenv_kernel in /g/g0/janel/.local/share/jupyter/kernels/myvenv_kernel

Check that a JSON file with expected format exists for your new kernel:

```
(myvenv) janel@flash21:~/myvenv$ cd ~/.local/share/jupyter/kernels/
(myvenv) janel@flash21:~/myvenv$ cd ~/.local/share/jupyter/kernels/$ ls
R dssi-demo-kernel julia-1.5 myvenv_kernel testkernel
```

```
(myvenv) janel@flash21:~/myvenv$ cd myvenv_kernel/
(myvenv) janel@flash21:~/myvenv$ cd myvenv_kernel/$ ls
kernel.json logo-32x32.png logo-64x64.png
```

```
(myvenv) janel@flash21:~/myvenv$ cat kernel.json
{
  "argv": [
    "/g/g0/janel/myvenv/bin/python3",
    "-m",
    "ipykernel_launcher",
    "-f",
    "{connection_file}"
  ],
  "display_name": "myvenv_kernel",
  "language": "python"
}
```
CUSTOM JUPYTERHUB KERNELS: PYTHON

Now, go to JupyterHub and check
(1) you can access the new kernel
(2) the kernel has any packages you installed to that virtual environment
Now, go to JupyterHub and check

1. you can access the new kernel

2. the kernel has any packages you installed to that virtual environment

```
In [1]: import torch
import tensorflow

In [2]: msg = tensorflow.constant("Another hello world");
tensorflow.print(msg)

Another hello world

In [3]: torch.rand(3,3)

Out[3]: tensor([[0.8326, 0.7758, 0.8238],
                [0.7299, 0.2683, 0.5755],
                [0.5272, 0.9425, 0.7744]])
```
See the sections on custom kernels and home directory installation at

https://hpc.llnl.gov/services/jupyter

for more info!
CUSTOM JUPYTERHUB KERNELS

We’ll talk about creating custom kernels for:

- Python
- Julia
- R

For each of these, you’ll need to:

- Install a package (ipykernel, IJulia, or IRkernel)
- Check for & possibly create `~/.local/share/jupyter/kernels/<Directory for custom kernel name>/kernel.json`
First, install IJulia:

```bash
janeh@oslic7:~$ julia

_ _ _ _ _(_)_ _ | Documentation: https://docs.julialang.org
(_) _ _ | ( _) ( _) | Type "?" for help, "[]" for Pkg help.
| | | | | | | / _` | |
| | | | | | | | ( _ ) | Version 1.6.3 (2021-09-23)
_/ \____' _ _ _ _ _ \____' _ | Official https://julialang.org/ release
|__/ 

(@v1.6) pkg> add IJulia
  Updating registry at `~/.julia/registries/General`
  Updating git-repo `https://github.com/JuliaRegistries/General.git`
  Resolving package versions...
  Installed VersionParsing — v1.2.1
  Installed ZeroMQ_jll —— v4.3.4+0
  Installed libsodium_jll — v1.0.20+0
  Installed Conda ———— v1.6.0
  Installed Parsers ———— v2.1.2
  Installed JSON ———— v0.21.2
  Downloaded artifact: libsodium
  Downloaded artifact: ZeroMQ
  Updating `~/.julia/environments/v1.6/Project.toml`
  [7073ff75] + IJulia v1.23.2
  Updating `~/.julia/environments/v1.6/Manifest.toml`
```
Go to `~/.local/share/jupyter/kernels`.

```bash
janeh@oslic7:~/.local/share/jupyter/kernels$ ls
R         julia-1.5  myvenv_kernel  spack-R
dssi-demo-kernel julia-1.6.3  opence    testkernel
janeh@oslic7:~/.local/share/jupyter/kernels$ ls julia-1.6.3/
kernel.json
janeh@oslic7:~/.local/share/jupyter/kernels$ cat julia-1.6.3/kernel.json
{
  "display_name": "Julia 1.6.3",
  "argv": [
    "/collab/usr/gapps/julia/julia-1.6.3-x86-64/bin/julia",
    "-i",
    "--color=yes",
    "--project=@.",
    "/g/g0/janeh/.julia/packages/IJulia/e8kqU/src/kernel.jl",
    
  ],
  "language": "julia",
  "env": {},
  "interrupt_mode": "signal"
}
```
CUSTOM JUPYTERHUB KERNELS: JULIA

Once you’ve checked `kernel.json` exists for the desired version of Julia, the Julia kernel will be visible on JupyterHub:
Selecting your new custom kernel, you can open a Julia notebook:

```
In [1]: print("Hello world")
Hello world

In [2]: VERSION
Out[2]: v"1.6.3"
```
We’ll talk about creating custom kernels for:

- Python
- Julia
- R

For each of these, you’ll need to:

- Install a package (ipykernel, IJulia, or IRkernel)
- Check for & possibly create `~/.local/share/jupyter/kernels/<Directory for custom kernel name>/kernel.json`
First, fire up the default R on LC:

```
janeh@flash21:~$ which R
/usr/bin/R
janeh@flash21:~$ R

R version 3.6.0 (2019-04-26) -- "Planting of a Tree"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-redhat-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
```

>
CUSTOM JUPYTERHUB KERNELS: R

Install IRKernel via `install.packages('IRkernel')`:

```r
> install.packages('IRkernel')
Installing package into '/usr/lib64/R/library'
(as 'lib' is unspecified)
Warning in install.packages("IRkernel") :
  'lib = "/usr/lib64/R/library"' is not writable
Would you like to use a personal library instead? (yes/No/cancel) y
Would you like to create a personal library
'~/R/x86_64-redhat-linux-gnu-library/3.6'
to install packages into? (yes/No/cancel) y
```

You’ll be prompted to create a personal library where you can install packages – similar to how we created a virtual environment to install python packages.
You’ll next be prompted to select a “CRAN mirror”, a web server from which to download IRkernel. (I chose #75.)

```
--- Please select a CRAN mirror for use in this session ---
Secure CRAN mirrors
1: 0-Cloud [https]
2: Australia (Canberra) [https]
3: Australia (Melbourne 1) [https]
4: Australia (Melbourne 2) [https]
5: Australia (Perth) [https]
6: Austria [https]
7: Brazil (BA) [https]
8: Brazil (BA) [https]
9: Brazil (BA) [https]
75: USA (OR) [https]
76: USA (TN) [https]
77: USA (TX 1) [https]
78: Uruguay [https]
79: (other mirrors)

Selection: 75
trying URL 'https://ftp.osuosl.org/pub/cran/src/contrib/IRkernel_1.2.tar.gz'
Content type 'application/x-gzip' length 62663 bytes (61 KB)
=====================================================================
downloaded 61 KB
```
Create a new directory called `.local/share/jupyter/kernels/R`

```
janeh@flash21:~$ mkdir .local/share/jupyter/kernels/R/
```
Create a new directory called `.local/share/jupyter/kernels/R`

```
janeh@flash21:~$ mkdir .local/share/jupyter/kernels/R/
janeh@flash21:~$ cd .local/share/jupyter/kernels/R/
janeh@flash21:~/.local/share/jupyter/kernels/R$ touch kernel.json
```
CUSTOM JUPYTERHUB KERNELS: R

Create a new directory called `.local/share/jupyter/kernels/R`

```
janeh@flash21:~$ mkdir .local/share/jupyter/kernels/R/
```

where you will create a file, `kernel.json`

```
janeh@flash21:~$ cd .local/share/jupyter/kernels/R/
janeh@flash21:~/.local/share/jupyter/kernels/R$ touch kernel.json
```

and populate it with the following information:

```
janeh@flash21:~/.local/share/jupyter/kernels/R$ cat kernel.json
{"argv": ["/usr/bin/R", "--quiet", "-e", "IRkernel::main()", "--args", "{connection_file}"],
"display_name":"R",
"language":"R"
}
```
Now, go to JupyterHub and check that you can use R!
Now, go to JupyterHub and check that you can use R!

In [1]: mystring <- "R says hello!"

In [2]: print(mystring)

[1] "R says hello!"

https://hpc.llnl.gov/services/Jupyter/R
A few popular open source languages

- Python on LC
  - Python virtual environments
  - How to install python packages: pytorch, tensorflow
- Julia on LC
- R on LC

- JupyterHub (out of the box & with custom kernels)

How to use containers

- How to use spack, a package manager
- How to know what you can install
HAVE YOU WORKED WITH CONTAINERS (LIKE DOCKER) BEFORE?

A. Yes, a lot
B. Yes, a little
C. Not yet
You can use containers to work with an application that isn’t installed/supported on LC machines, without needing to install the app yourself.

LC uses Singularity rather than Docker to run containers, but your Docker containers will work with Singularity!

For example, say you want the latest python available here:
```
singularity pull <name_for_container_img> docker://<name_from_dockerhub>
```

```
Janeh@flash21:~ /Singularity/DSSI$ singularity pull mypython.img docker://python
INFO: Converting OCI blobs to SIF format
INFO: Starting build...
Getting image source signatures
```

Creates a file, ` `<name_for_container_img>` `
YOU CAN “SHELL INTO” A CONTAINER, ALLOWING YOU TO WORK WITH WHATEVER BINARIES/APPS LIVE INSIDE:

```
janeh@flash21:~/Singularity/DSSI$ which python3
/usr/tce/bin/python3
janeh@flash21:~/Singularity/DSSI$ python3
Python 3.7.2 (default, Feb 26 2019, 08:59:10)
[GCC 4.9.3] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()
janeh@flash21:~/Singularity/DSSI$ singularity shell mypython.img bash
Singularity> which python3
/usr/local/bin/python3
Singularity> python3
Python 3.9.5 (default, May 12 2021, 15:26:36)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```
When you’re “in” the container, you can use the container apps but access your local filesystem*

```
janeh@flash21:~/Singularity/DSSI$ singularity shell mypython.img bash
Singularity> ls
mypython.img
Singularity> pwd
/g/g0/janeh/Singularity/DSSI
Singularity> echo $HOME
/g/g0/janeh
```

*Your home directory is viewable by default, but you have to explicitly “mount” other LC file systems. See the docs!
Without “entering” the container, you can execute commands as if you were inside the container with

```bash
`singularity exec <container> <command to run inside container>`
```

```bash
ejaneh@flash21:~/Singularity/DSSI$ singularity exec mypython.img python3 -c 'print("hello!")'
hello!
```
Similarly, you could work with R v4.1.0 using a container:

```
janeh@flash21:~/Singularity/DSSI$ singularity pull myR.img docker://r-base
```
Similarly, you could work with R v4.1.0 using a container:

```bash
janeh@flash21:~/Singularity/DSSI$ singularity pull myR.img docker:///r-base

janeh@flash21:~/Singularity/DSSI$ singularity shell myR.img bash
Singularity> R
```

```
R version 4.1.0 (2021-05-18) -- "Camp Pontanezen"
Copyright (C) 2021 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

   Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> msg <- "Hello!"
> print(msg)
```
See

https://lc.llnl.gov/cloud/services/Singularity/

for our docs with more examples.
AGENDA

- A few popular open source languages
  - Python on LC
    - Python virtual environments
    - How to install python packages: pytorch, tensorflow
  - Julia on LC
  - R on LC
  - JupyterHub (out of the box & with custom kernels)
- How to use containers
- **How to use spack, a package manager**
- How to know what you can install
Can find docs and tutorials at


As well as a list of all available packages at

https://spack.readthedocs.io/en/latest/package_list.html#r-irkernel
First, install Spack with `git clone...`

```bash
janeh@flash21:~$ git clone https://github.com/spack/spack.git
Cloning into 'spack'...
remote: Enumerating objects: 283370, done.
remote: Counting objects: 100% (419/419), done.
remote: Compressing objects: 100% (249/249), done.
remote: Total 283370 (delta 134), reused 324 (delta 73), pack-reused 282951
Receiving objects: 100% (283370/283370), 114.64 MiB | 27.24 MiB/s, done.
Resolving deltas: 100% (120374/120374), done.
Updating files: 100% (8162/8162), done.
```
First, install Spack with `git clone...` 

Source your Spack environment to configure paths.

janeh@flash21:~$ git clone https://github.com/spack/spack.git
Cloning into 'spack'...
remote: Enumerating objects: 283370, done.
remote: Counting objects: 100% (419/419), done.
remote: Compressing objects: 100% (249/249), done.
remote: Total 283370 (delta 134), reused 324 (delta 73), pack-reused 282951
Receiving objects: 100% (283370/283370), 114.64 MiB | 27.24 MiB/s, done.
Resolving deltas: 100% (120374/120374), done.
Updating files: 100% (8162/8162), done.
janeh@flash21:~$ cd spack/share/spack/
janeh@flash21:~/spack/share/spack$ . setup-env.sh
And then you’re ready to install packages!

```
spack install <packagename>
```

janeh@flash21:~$ git clone https://github.com/spack/spack.git
Cloning into 'spack'...
remote: Enumerating objects: 283370, done.
remote: Counting objects: 100% (419/419), done.
remote: Compressing objects: 100% (249/249), done.
remote: Total 283370 (delta 134), reused 324 (delta 73), pack-reused 282951
Receiving objects: 100% (283370/283370), 114.64 MiB | 27.24 MiB/s, done.
Resolving deltas: 100% (120374/120374), done.
Updating files: 100% (8162/8162), done.

janeh@flash21:~$ cd spack/share/spack/
janeh@flash21:~/spack/share/spack$ . setup-env.sh
janeh@flash21:~/spack/share/spack$ spack install r

Often this will be enough to get a clean installation.
Here, Spack complained about my compiler (gcc) version:

janeh@flash21:~$ git clone https://github.com/spack/spack.git
Cloning into 'spack'...
remote: Enumerating objects: 283370, done.
remote: Counting objects: 100% (419/419), done.
remote: Compressing objects: 100% (249/249), done.
remote: Total 283370 (delta 134), reused 324 (delta 73), pack-reused 282951
Receiving objects: 100% (283370/283370), 114.64 MiB | 27.24 MiB/s, done.
Resolving deltas: 100% (120374/120374), done.
Updating files: 100% (8162/8162), done.

janeh@flash21:~$ cd spack/share/spack/

janeh@flash21:~/spack/share/spack$ . setup-env.sh

janeh@flash21:~/spack/share/spack$ spack install r

==> Error: Conflicts in concretized spec "r@4.1.0%gcc@4.9.3~X~external-lapack~memory_profiling~rmath arch=linux-rhel7-haswell/zufij6s"
List of matching conflicts for spec:

  icu4c@67.1%gcc@4.9.3 cxxstd=11 arch=linux-rhel7-haswell
  ^python@3.8.10%gcc@4.9.3+bz2+ctypes+dbm~debug+libxml2+lzma~nis~optimizations+pic+pyexpat+pythoncmd+readline+shared+sqlite3+ssl~tix~tkinter~ucs4+uuid+zlib patches=0d98e93189bc278fbc37a50ed7f183bd8aaf249a8e1670a465f0db6bb4f8cf87 arch=linux-rhel7-haswell
USING SPACK

To fix this, I used the module system to find and load a newer version of gcc:

```
janeh@flash21:~/spack/share/spack$ module avail gcc

/usr/tce/modulefiles/Core

  gcc/4.8-redhat  gcc/6.1.0  gcc/7.3.0  gcc/8.3.1  gcc/10.2.1
  gcc/4.9.3 (D)  gcc/7.1.0  gcc/8.1.0  gcc/9.3.1

Where:
  D: Default Module
```

Use "module spider" to find all possible modules and extensions. Use "module keyword key1 key2 ..." to search for all possible modules matching any of the "keys".

```
janeh@flash21:~/spack/share/spack$ module load gcc/10.2.1

Lmod is automatically replacing "intel/19.0.4" with "gcc/10.2.1".
```

Due to MODULEPATH changes, the following have been reloaded:

1) mvapich2/2.3
I then registered the new compiler (version 10.2.1 of gcc) and used the syntax

`spack install <package> %<compiler_name>@<compiler_version>`

```
spack install r %gcc@10.2.1
```

After this, R version 4.1.0 installed successfully.

With `spack find` I can see both what’s installed and where.

```
janeh@flash21:~$ spack find
===> 29 installed packages
-- linux-rhel7-haswell / gcc@10.2.1 -----------------------------
berkeley-db@18.1.40  gettext@0.21  libmd@1.0.3  pcre2@10.36  sqlite@3.35.5
bzip2@1.0.8  icu4c@67.1  libunistring@0.9.10  perl@5.32.1  tar@1.34
curl@7.76.1  libbsd@0.11.3  libxml2@2.9.10  pkgconf@1.7.4  util-linux-uuid@2.36.2
diffutils@3.7  libffi@3.3  ncurses@6.2  python@3.8.10  xz@5.2.5
depat@2.3.0  libiconv@1.16  openjdk@11.0.8_10  r@4.1.0  zlib@1.2.11
gdbm@1.19  libidn2@2.3.0  openssl@1.1.1k  readline@8.1

janeh@flash21:~$ spack find r
===> 1 installed package
-- linux-rhel7-haswell / gcc@10.2.1 -----------------------------
r@4.1.0

janeh@flash21:~$ spack find --paths r
===> 1 installed package
-- linux-rhel7-haswell / gcc@10.2.1 -----------------------------
r@4.1.0 /g/g0/janeh/spack/opt/spack/linux-rhel7-haswell/gcc-10.2.1/r-4.1.0-kr3ugnxz2qmoggyeti4ce
pjzx7n3i4yp
`spack load <package>` allows me to find my new package:

```bash
janeh@flash21:$ which R
/usr/bin/R
janeh@flash21:$ spack load r
janeh@flash21:$ which R
~/spack/opt/spack/linux-rhel7-haswell/gcc-10.2.1/r-4.1.0-kr3ugnzx2qmoggyeti4cepjzx7n3i4yp/bin/R
janeh@flash21:$ R
```

R version 4.1.0 (2021-05-18) -- "Camp Pontanezen"
Copyright (C) 2021 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.  
You are welcome to redistribute it under certain conditions.     
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.    
Type 'contributors()' for more information and     
'citation()' on how to cite R or R packages in publications.     

Type 'demo()' for some demos, 'help()' for on-line help, or     
'help.start()' for an HTML browser interface to help.        
Type 'q()' to quit R.

>
AGENDA

- A few popular open source languages
  - Python on LC
    - Python virtual environments
    - How to install python packages: pytorch, tensorflow
  - Julia on LC
  - R on LC
  - JupyterHub (out of the box & with custom kernels)
- How to use containers
- How to use spack, a package manager
- How to know what you can install
WHAT YOU CAN INSTALL

See SMDB for prior approvals (https://smdb.llnl.gov/) and reach out to the hotline if you’re unsure (lc-hotline@llnl.gov)

Reminder! Users / system administrators are responsible for keeping non-LLNL managed software patched and up to date, per CSP IM-2015.

<table>
<thead>
<tr>
<th>Package Name</th>
<th>Version</th>
<th>Unclassified</th>
<th>Classified (iSRD &amp; iSNSI)</th>
<th>LC OCF/SCF</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRKernel</td>
<td>&gt;= 1.1.1</td>
<td>+</td>
<td>+</td>
<td>Approved</td>
</tr>
</tbody>
</table>
Additionally, any python packages in wheelhouse are approved:

https://www-lc.llnl.gov/python/wheelhouse/

Over 1100 packages, including multiple versions.
DISCLAIMER

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.