



Spack

User Guide

Ver. 1.0

Last updated: Dec 08, 2021

www.cdac.in

Copyright Notice

Copyright © 2021 Centre for Development of Advanced Computing
All Rights Reserved.

Any technical documentation that is made available by C-DAC (Centre for Development of Advanced Computing) is the copyrighted work of C-DAC and is owned by C-DAC. This technical documentation is being delivered to you as is, and C-DAC makes no warranty as to its accuracy or use. Any use of the technical documentation or the information contained therein is at the risk of the user. C-DAC reserves the right to make changes without prior notice.

No part of this publication may be copied without the express written permission of C-DAC.

Trademarks

CDAC, CDAC logo, NSM logo are trademarks or registered trademarks.

Other brands and product names mentioned in this manual may be trademarks or registered trademarks of their respective companies and are hereby acknowledged.

Intended Audience

This document is meant for PARAM Pravega users.

Typographic Conventions

Symbol	Meaning
Blue underlined text	A hyperlink or link you can click to go to a related section in this document or to a URL in your web browser.
Bold	The names of menus, menu items, headings, and buttons.
<i>Italics</i>	Variables or placeholders or special terms in the document.
<code>Console text</code>	Console commands

Getting help

For technical assistance, use ticketing tool.

Give us your feedback

We value your feedback. Kindly send your comments on content of this document to samirs@cdac.in .
Please include the page number of the document along with your feedback.

**DISCLAIMER**

The information contained in this document is subject to change without notice. C-DAC shall not be liable for errors contained herein or for incidental or consequential damages in connection with the performance or use of this manual.

Contents

Spack	4
Introduction.....	4
To Use Pre-installed Applications from Spack.....	5
To install new application.....	7
Uninstalling Packages	10
Using Environments	11
Packaging (For Application developers).....	13
Sample SLURM script for OpenMP applications/programs. to use spack	15
Sample SLURM script for MPI applications/programs. to use spack.....	15
References.....	16

Spack

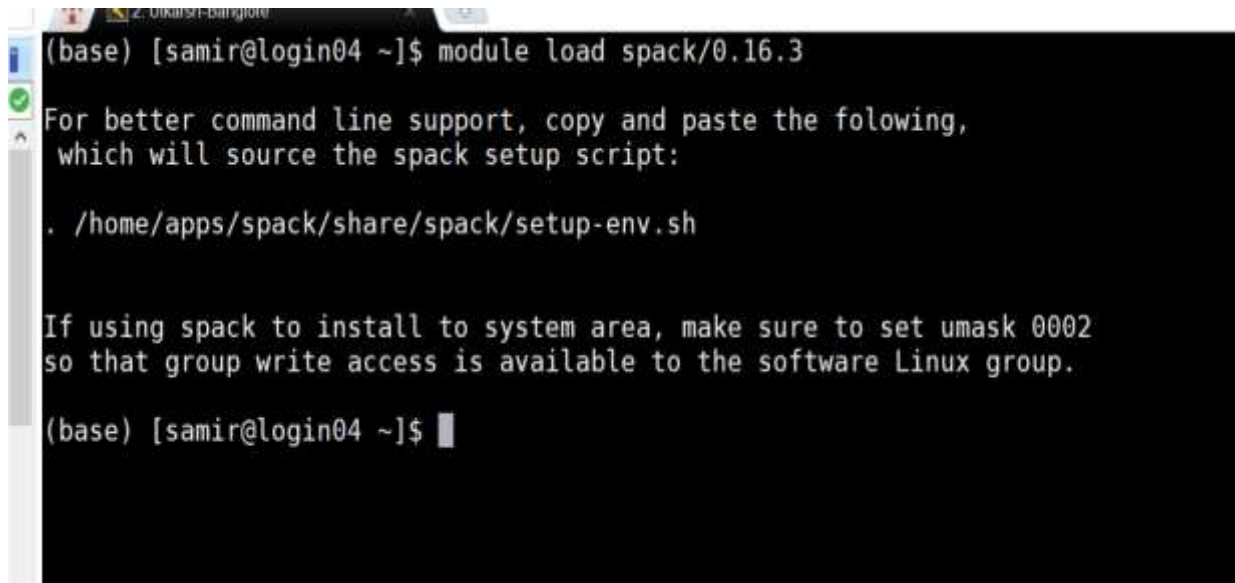
Introduction

Spack automates the download-build-install process for software - including dependencies - and provides convenient management of versions and build configurations. It is designed to support multiple versions and configurations of software on a wide variety of platforms and environments. It is designed for large supercomputing centers, where many users and application teams share common installations of software on clusters with exotic architectures, using libraries that do not have a standard ABI. Spack is non-destructive: installing a new version does not break existing installations, so many configurations can coexist on the same system.

Getting Started

On your login node command prompt execute below commands:

\$ module load spack - To load SPACK module and setting up environment for SPACK.



```
(base) [samir@login04 ~]$ module load spack/0.16.3
For better command line support, copy and paste the following,
which will source the spack setup script:
. /home/apps/spack/share/spack/setup-env.sh

If using spack to install to system area, make sure to set umask 0002
so that group write access is available to the software Linux group.
(base) [samir@login04 ~]$
```

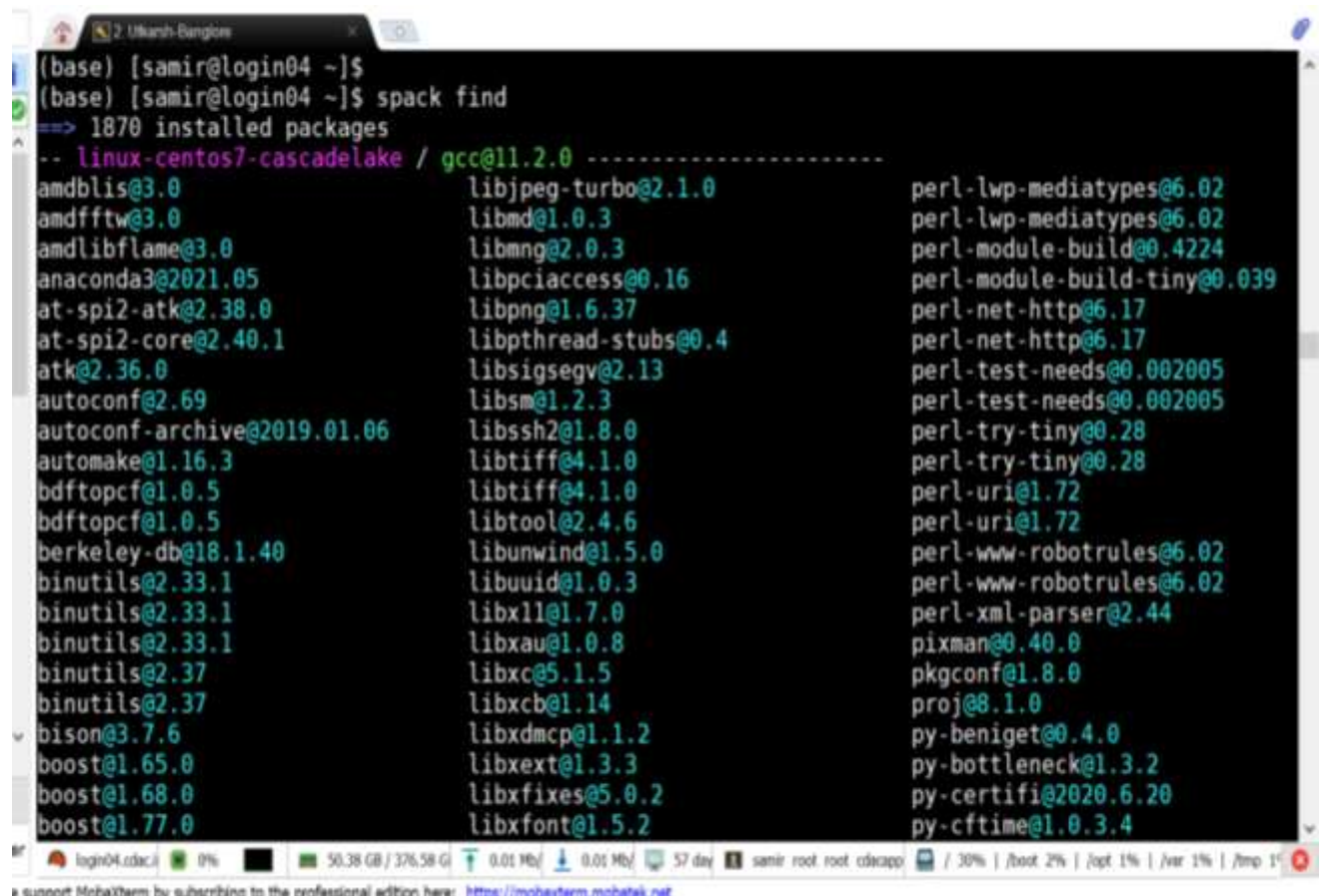
Kindly see the above screenshot and source below line including initial dot.

```
$ . /home/apps/spack/share/spack/setup-env.sh
```

To Use Pre-installed Applications from Spack

Spack find

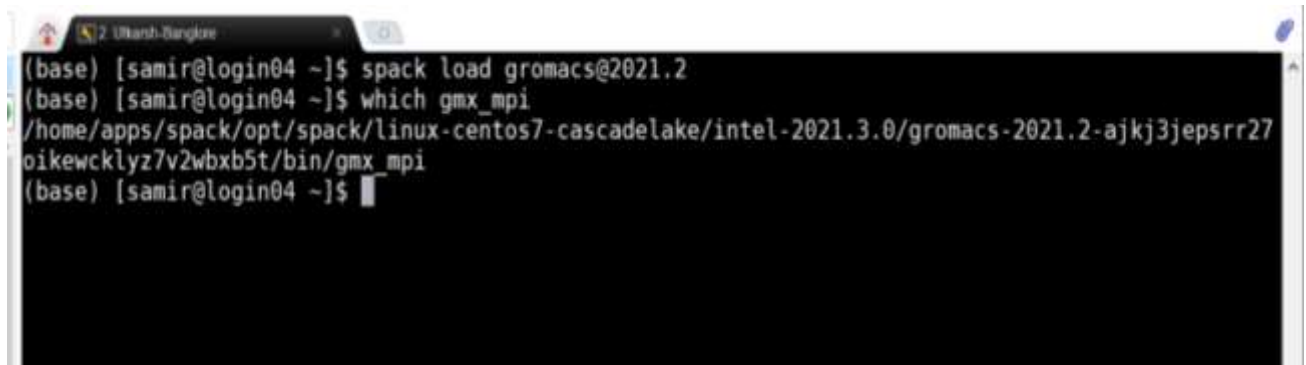
The `spack find` command is used to query installed packages on Param Pravega. Note that some packages appear identical with the default output. The `-l` flag shows the hash of each package, and the `-f` flag shows any non-empty compiler flags of those packages.



```
(base) [samir@login04 ~]$
(base) [samir@login04 ~]$ spack find
=> 1870 installed packages
-- linux-centos7-cascadelake / gcc@11.2.0 -----
amdblis@3.0                libjpeg-turbo@2.1.0      perl-lwp-mediatypes@6.02
amdfftw@3.0                libmd@1.0.3              perl-lwp-mediatypes@6.02
amdlibflame@3.0            libmng@2.0.3             perl-module-build@0.4224
anaconda3@2021.05          libpciaccess@0.16        perl-module-build-tiny@0.039
at-spi2-atk@2.38.0         libpng@1.6.37            perl-net-http@6.17
at-spi2-core@2.40.1        libpthread-stubs@0.4     perl-net-http@6.17
atk@2.36.0                 libsigsegv@2.13          perl-test-needs@0.002005
autoconf@2.69              libsm@1.2.3              perl-test-needs@0.002005
autoconf-archive@2019.01.06 libssh2@1.8.0            perl-try-tiny@0.28
automake@1.16.3            libtiff@4.1.0            perl-try-tiny@0.28
bdftopcf@1.0.5             libtiff@4.1.0            perl-uri@1.72
bdftopcf@1.0.5             libtool@2.4.6            perl-uri@1.72
berkeley-db@18.1.40        libunwind@1.5.0          perl-www-robotrules@6.02
binutils@2.33.1            libuuid@1.0.3            perl-www-robotrules@6.02
binutils@2.33.1            libx11@1.7.0             perl-xml-parser@2.44
binutils@2.33.1            libxau@1.0.8             pixman@0.40.0
binutils@2.37              libxc@5.1.5              pkgconf@1.8.0
binutils@2.37              libxcb@1.14              proj@8.1.0
bison@3.7.6                libxdmcp@1.1.2           py-beniget@0.4.0
boost@1.65.0               libxext@1.3.3            py-bottleneck@1.3.2
boost@1.68.0               libxfixed@5.0.2          py-certifi@2020.6.20
boost@1.77.0               libxfont@1.5.2           py-cftime@1.0.3.4
```

Spack load application name

The easiest way is to use `spack load <application name@version>`

A terminal window with a black background and white text. The prompt is '(base) [samir@login04 ~]'. The user enters 'spack load gromacs@2021.2'. The prompt changes to '(base) [samir@login04 ~]'. The user enters 'which gmx_mpi'. The output is '/home/apps/spack/opt/spack/linux-centos7-cascadelake/intel-2021.3.0/gromacs-2021.2-ajkj3jepsrr27oikewcklyz7v2wbxb5t/bin/gmx_mpi'. The prompt returns to '(base) [samir@login04 ~]'.

```
(base) [samir@login04 ~]$ spack load gromacs@2021.2
(base) [samir@login04 ~]$ which gmx_mpi
/home/apps/spack/opt/spack/linux-centos7-cascadelake/intel-2021.3.0/gromacs-2021.2-ajkj3jepsrr27oikewcklyz7v2wbxb5t/bin/gmx_mpi
(base) [samir@login04 ~]$
```

To Know the Pre-Loaded Application/Compilers

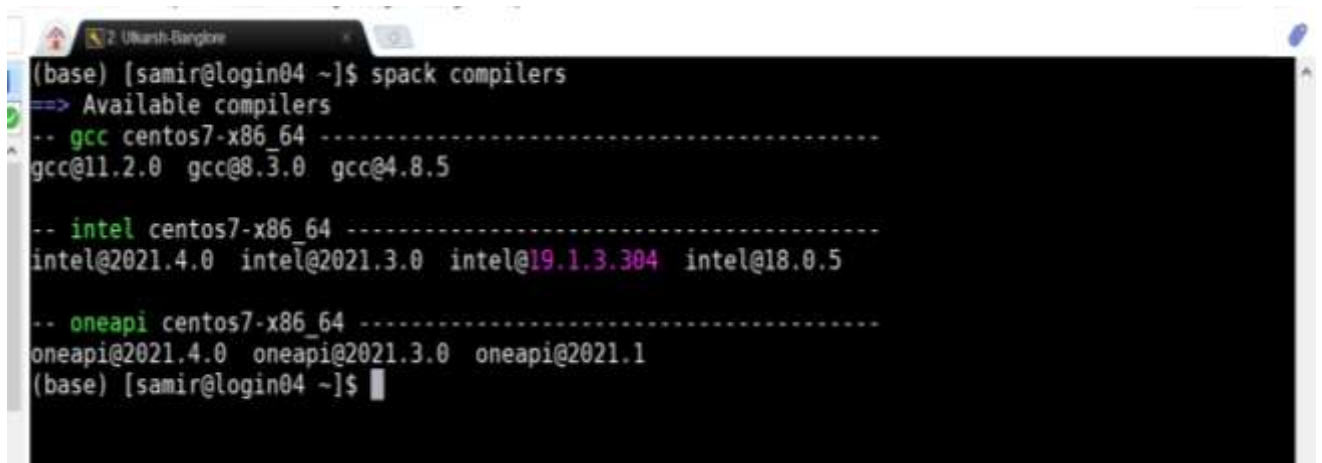
```
$ spack find --loaded
==> 6 installed packages
-- linux-ubuntu18.04-x86_64 / gcc@7.5.0 -----
gcc@8.3.0  gmp@6.1.2  isl@0.18      mpc@1.1.0  mpfr@3.1.6  zlib@1.2.11
```

To install new application

First check the available compilers in Spack with below command:

spack compilers

Spack manages a list of available compilers on the system, detected automatically from the user's PATH variable. The spack compilers command is an alias for the command spack compiler list.



```
(base) [samir@login04 ~]$ spack compilers
==> Available compilers
-- gcc centos7-x86_64 -----
gcc@11.2.0 gcc@8.3.0 gcc@4.8.5

-- intel centos7-x86_64 -----
intel@2021.4.0 intel@2021.3.0 intel@19.1.3.304 intel@18.0.5

-- oneapi centos7-x86_64 -----
oneapi@2021.4.0 oneapi@2021.3.0 oneapi@2021.1
(base) [samir@login04 ~]$
```

To Check the Compilers Available in the System

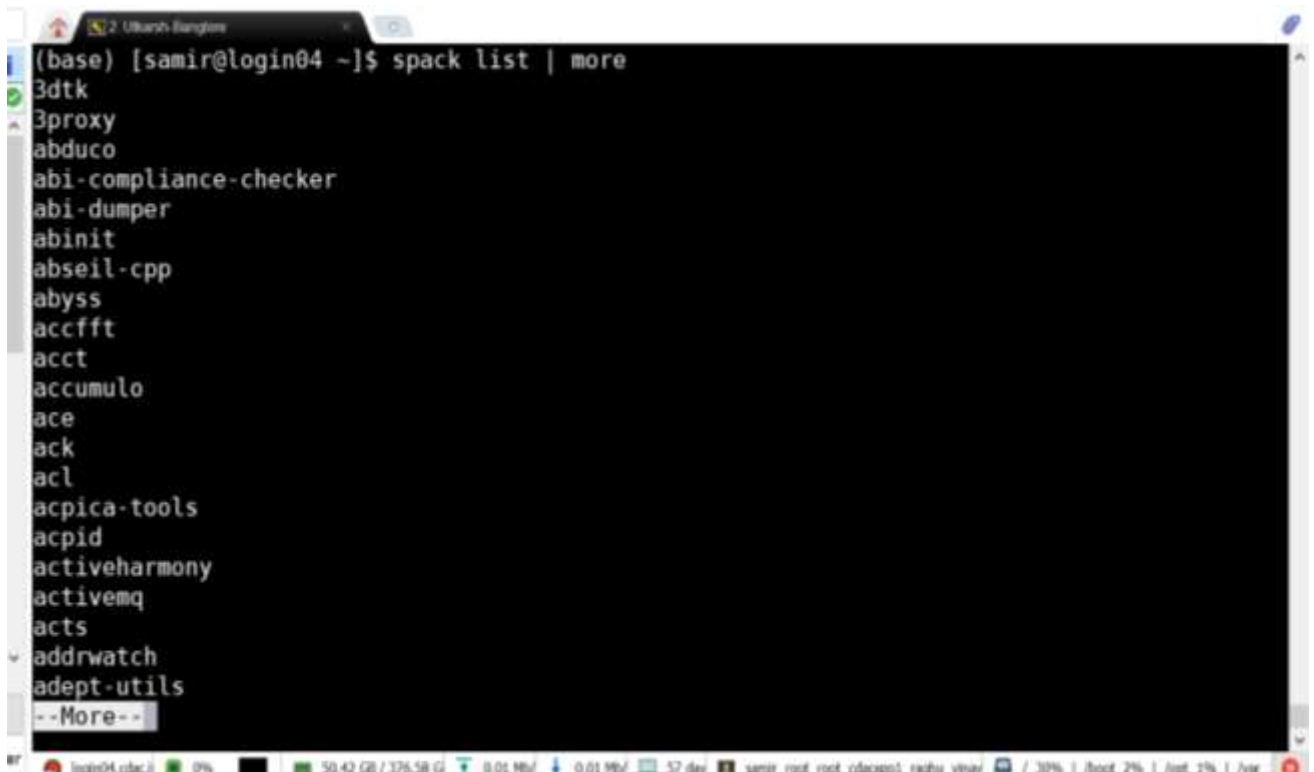
```
$ spack compiler list
==> Available compilers
-- clang ubuntu18.04-x86_64 -----
clang@6.0.0
-- gcc ubuntu18.04-x86_64 -----
gcc@8.3.0 gcc@7.5.0 gcc@6.5.0
```

Check if application is available in Spack repo with command-

spack list

The spack list command shows available packages.

The spack list command can also take a query string. Spack automatically adds wildcards to both ends of the string, or you can add your own wildcards.



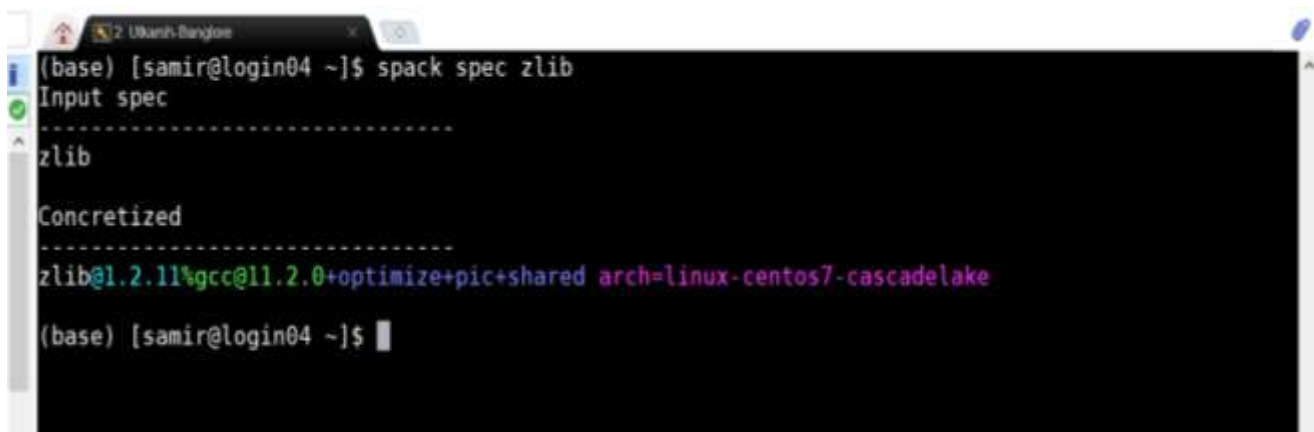
```
(base) [samir@login04 ~]$ spack list | more
3dtk
3proxy
abduco
abi-compliance-checker
abi-dumper
abinit
abseil-cpp
abyss
accfft
acct
accumulo
ace
ack
acl
acpica-tools
acpid
activeharmony
activemq
acts
addrwatch
adept-utils
--More--
```

Before installing application check its spec with command

`spack spec zlib`

Show what would be installed, given a spec. The spec syntax also includes compiler flags.

Spack accepts `cppflags`, `cflags`, `cxxflags`, `fFlags`, `ldflags`, and `ldlibs` parameters. The values of these fields must be quoted on the command line if they include spaces. These values are injected into the compile line automatically by the Spack compiler wrappers.



```
(base) [samir@login04 ~]$ spack spec zlib
Input spec
-----
zlib

Concretized
-----
zlib@1.2.11%gcc@11.2.0+optimize+pic+shared arch=linux-centos7-cascadelake

(base) [samir@login04 ~]$
```

To change default compiler for zlib installation to oneapi

spack spec zlib %oneapi

```
(base) [samir@login04 ~]$ spack spec zlib
Input spec
-----
zlib

Concretized
-----
zlib@1.2.11%gcc@11.2.0+optimize+pic+shared arch=linux-centos7-cascadelake

(base) [samir@login04 ~]$ spack spec zlib %oneapi
Input spec
-----
zlib%oneapi

Concretized
-----
zlib@1.2.11%oneapi@2021.4.0 cflags="-ipo -mcmmodel=large -qopenmp -xCORE-AVX512 -march=cascadelake -mtune=cascadelake" cxxflags="-ipo -qopenmp -mcmmodel=large -xCORE-AVX512 -march=cascadelake -mtune=cascadelake" fflags="-ipo -qopenmp -xCORE-AVX512 -mcmmodel=large -march=cascadelake -mtune=cascadelake" +optimize+pic+shared arch=linux-centos7-cascadelake

(base) [samir@login04 ~]$
```

spack install

Below is an example of installation of package using spack:

```
spack install gromacs@2020.5 +cuda~mpi+blas %intel ^intel-mkl
```

Above command will install gromacs version 2020.5 with blas and cuda support and without MPI support. For blas there are multiple providers like OpenBLAS, Intel MKL, amdlib, and essl, ^intel-mkl will tell spack to use intel-mkl for blas routines.

Operators in Spack

- % to select compiler out of available compilers
- ^ to use variant of package
- @ to define the version number of packages to be installed.
- + to enable variant for package
- ~ to disable variant for package

Uninstalling Packages

Earlier we installed many configurations each of zlib. Now we will go through and uninstall some of those packages that we didn't really need.

```
$ spack uninstall zlib %gcc@6.5.0  
  (type : y)
```

Using Environments

Spack has an environment feature in which you can group installed software. You can install software with different versions and dependencies in each environment and can change software to use at once by changing environments. You can create a Spack environment by `spack env create` command. You can create multiple environments by specifying different environment names here.

```
spack env create myenv
```

To activate the created environment, type `spack env activate`. Adding `-p` option will display the current activated environment on your console. Then, install software you need to the activated environment.

```
spack env activate -p myenv
myenv] [username@es1 ~]$ spack install xxxxx
```

You can deactivate the environment by `spack env deactivate`. To switch to another environment, type `spack env activate` to activate it.

```
[myenv] [username@es1 ~]$ spack env deactivate
[username@es1 ~]$
```

Use `spack env list` to display the list of created Spack environments.

```
[username@es1 ~]$ spack env list
==> 2 environments
  myenv
  another_env
```

`spack env`

Refer below screenshot to activate an environment, add a package and install it in that environment:

```

(base) [samir@login03 ~]$ spack env create samir
=> Updating view at /home/apps/spack/var/spack/environments/samir/.spack-env/view
=> Created environment 'samir' in /home/apps/spack/var/spack/environments/samir
=> You can activate this environment with:
=>   spack env activate samir
(base) [samir@login03 ~]$ spack env activate samir
(base) [samir@login03 ~]$ spack add gromacs%intel
=> Adding gromacs%intel to environment samir
(base) [samir@login03 ~]$ spack install
=> Starting concretization
=> Environment concretized in 14.40 seconds.
=> Concretized gromacs%intel
-   mjezqk6  gromacs@2021.3%intel@2021.4.0~blas~cuda~cycle_subcounters~double~hwloc~
ipo~lapack~mdrun_only~mpi~nosuffix~opencl~openmp~plumed~relaxed_double_precision~shar
ed~sycl build_type=RelWithDebInfo arch=linux-centos7-cascadelake
[+]  e7mdftl      ^cmake@3.21.4%intel@2021.4.0~doc~ncurses+openssl+ownlibs~qt build_t
ype=Release arch=linux-centos7-cascadelake
[+]  j6hk56b      ^ncurses@6.2%intel@2021.4.0~symlinks+termlib abi=none arch=linu
x-centos7-cascadelake
[+]  pngyrgd      ^pkg-config@0.27.1%intel@2021.4.0+internal_glib patches=49f
fcd644e190dc5efcb2fab491177811ea746c1a526f75d77118c2706574358 arch=linux-centos7-casc
adelake

```

The `spack env activate` will load the view associated with the Environment into the user environment.

Packaging (For Application developers)

Spack packages are installation scripts, which are essentially recipes for building the software.

They define properties and behaviour of the build, such as:

- where to find and how to retrieve the software.
- its dependencies.
- options for building the software from source; and
- build commands.

Once we've specified a package's recipe, users of our recipe can ask Spack to build the software with different features on any of the supported systems. Please refer [Packaging Guide — Spack 0.17.0 documentation](#) for detailed understanding of the Spack packaging.

Example Creating Own Package:

In below spec file we have used **Linewidth an IISc developed code**. Please see the bold lines for comments related to preceding lines in the spec file of spack package named `liscLinewidth`:

```
# Copyright 2013-2021 Lawrence Livermore National Security, LLC and other
# Spack Project Developers. See the top-level COPYRIGHT file for details.
#
# SPDX-License-Identifier: (Apache-2.0 OR MIT)
import os
import platform
import sys
import llnl.util.tty as tty
from spack import *
class IiscLinewidth(MakefilePackage):
    """
    Linewidth developed by IISC Bangalore.
    """
    homepage = ""
    #Url for homepage
    url = "file://{0}/linewidth.tar.gz".format(os.getcwd())
    #Url for source code
    manual_download = True
    #If source code is not available in public domain
    version('1',
    sha256='7215f6765e5f5eddfde5f0c67a5bbdef5960607f3e199a609ef5619278ec8a66',
    preferred=True)
    #You can add different versions for you package.
    variant('mpi', default=True, description='Install with MPI support')
    variant('openmp', default=True, description='Install with OpenMP
support')
    #Variant gives flexibility to users for changing parameter before
compilation.
    depends_on('gmake', type='build')
    depends_on('mpi', when='+mpi')
    depends_on('hdf5+fortran+hl+mpi')
    depends_on('intel-mkl')
```

```

depends_on('py-h5py')
depends_on('py-matplotlib', type=('build', 'run'))
#Depend clause used to specify dependancies for your code.

@property
def build_targets(self):
    targets = [
        # '--directory=SRC',
        '--file=Makefile',
        'LIBS={0} {1} '.format(self.spec['intel-mkl'].libs.ld_flags,
                               self.spec['hdf5'].libs.ld_flags),
        'HDFINCFLAGS={0}'.format(self.spec['hdf5'].prefix.include),
        'HDF5_HOME={0}'.format(self.spec['hdf5'].prefix),
        'FC={0}'.format(self.spec['mpi'].mpifc)
    ]
    return targets
def install(self, spec, prefix):
    mkdirp(prefix.bin)
    install('linewidth', prefix.bin)
####
#This code uses Makefile for building application. We can define some
properties
# to make changes in Makefile, changing parameter in Makefile at compile
time.

```

Sample Steps taken for Creating Linewidth application recipe for Spack

1. Source code

Source code of Linewidth was not available through public repo like github, so needed to import OS package.

os.getcwd() - expects the source tar present in current working directory.

cha256- to check for sha256 checksum we added same in version clause and for place holder we have give version as 1.

manual download = True referes to spack will not try to download source code for the package.

name- make sure that name of tar file is same as used inside package recipe

2. Variant- User can control behavior of application being built through this clause.

Ex- To enable MPI support we have define it to be true by default.

3. depends_on() - This clause defines all dependencies required to build the given application.

Ex- In linewidth example we have used Intel-mkl and HDF5.

4. @property - With this decorator we can define some properties for build system like edit, build, install.

5. property build_targets - Defines logic of building source for native platform.

6. property install - Defines install procedure to be used after building source code.

Ex- In our example we define prefix path

Sample SLURM script for OpenMP applications/programs. to use spack

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH -p cpu ## gpu/standard
#SBATCH --exclusive
#SBATCH -t 1:00:00

echo "SLURM_JOBID"=$SLURM_JOBID
echo "SLURM_JOB_NODELIST"=$SLURM_JOB_NODELIST
echo "SLURM_NNODES"=$SLURM_NNODES
echo "SLURM_NTASKS"=$SLURM_NTASKS
ulimit -s unlimited
ulimit -c unlimited
export OMP_NUM_THREADS=4 ### Maximum number of threads= Number of physical
core

#To load necessary application/compiler through spack
module load spack
export SPACK_ROOT=/home/apps/spack
. $SPACK_ROOT/share/spack/setup-env.sh
spack load intel-mpi@2019.10.317 /6icwzn3
spack load intel-mkl@2020.4.304
spack load intel-oneapi-compilers@2021.4.0
spack load gcc@11.2.0

(time <executable_path> )
```

Sample SLURM script for MPI applications/programs. to use spack

```
#!/bin/bash
#SBATCH --nodes=2
#SBATCH -p cpu ## gpu/standard
#SBATCH --exclusive
#SBATCH -t 1:00:00

echo "SLURM_JOBID"=$SLURM_JOBID
echo "SLURM_JOB_NODELIST"=$SLURM_JOB_NODELIST
echo "SLURM_NNODES"=$SLURM_NNODES
echo "SLURM_NTASKS"=$SLURM_NTASKS
ulimit -s unlimited
ulimit -c unlimited

#To load necessary application/compiler through spack
module load spack
export SPACK_ROOT=/home/apps/spack
. $SPACK_ROOT/share/spack/setup-env.sh
spack load intel-mpi@2019.10.317 /6icwzn3
spack load intel-mkl@2020.4.304
spack load intel-oneapi-compilers@2021.4.0
spack load gcc@11.2.0
(time mpirun -np $SLURM_NTASKS <executable_path>
```

For more information, related to SLURM, please refer the USER MANUAL.

References

1. <https://spack.readthedocs.io/en/latest/>
2. <https://github.com/spack/spack>
3. Getting started
https://spack.readthedocs.io/en/latest/getting_started.html
4. Basic usage
https://spack.readthedocs.io/en/latest/basic_usage.html
5. Packaging guide
https://spack.readthedocs.io/en/latest/packaging_guide.html
6. Build system
https://spack.readthedocs.io/en/latest/build_systems.html