

Apptainer without elevated privileges

Timothy H. Kaiser, Ph.D.

tkaiserphd@gmail.com

Final Source and slides

- git clone <https://github.com/timkphd/examples.git>
- cd examples/apptainer/full

Outline

- Overview
- "Simple": Podman/Docker image, download, 'trivial' build
- An Apptainer recipe
- Apptainer difficulties
- Normal Requirements to build a container
- Building a container from inside another container
- Create a Bootstrap Container
- Install Apptainer as a normal user
- Some complete examples

Purpose (What?)

- Containers are useful for high-performance computing, efficiently packaging and executing applications and their dependencies.
- Discuss building containers with Apptainer without requiring elevated privileges.
 - We create a container encapsulating Apptainer as an application.
 - Use the encapsulated Apptainer to create additional containers.
- Show how we get there
- Examples please

Why containers?

```
[tkaiser@server dir]$ gcc -v 2>&1 | tail -1  
gcc version 8.5.0 20210514 (Red Hat 8.5.0-28) (GCC)
```

```
[tkaiser@server dir]$ apptainer shell appt.sif  
Apptainer> gcc -v 2>&1 | tail -1  
gcc version 13.3.0 (Ubuntu 13.3.0-6ubuntu2~24.04)  
Apptainer>  
Apptainer>
```

May want a different
version of os or
software
(gcc 13.3.0)

```
[tkaiser@server dir]$ apptainer exec appt.sif speedtest  
Retrieving speedtest.net configuration...  
Testing from Server Lab (192.174.62.50)...  
Retrieving speedtest.net server list...  
Selecting best server based on ping...  
Hosted by Boost Mobile (Columbus, OH) [1889.14 km]: 39.926 ms  
Testing download speed...  
Download: 322.52 Mbit/s  
Testing upload speed...  
Upload: 356.76 Mbit/s  
[tkaiser@server dir]$
```

May want to run an app that
you don't have installed
(speedtest)

Where

- From a local docker/podman image
 - podman save localhost/appt:1.0 -o appt.tar
 - apptainer build appt.sif docker-archive://appt.tar
- From a repo:
 - apptainer run docker://ubuntu
 - apptainer pull docker://ubuntu
- A recipe (trivial)
 - apptainer build ub0.sif ub0.def

```
Bootstrap: docker
From: ubuntu:latest
%post
mkdir /mystuff
```

An almost trivial recipe

```
[tkaiser@server ~]$cat broken.def
Bootstrap: docker
From: ubuntu:latest
```

```
%post
```

```
### Install Packages and Libraries ###
```

```
apt-get update -y
```

```
apt-get install -y wget git
```

```
apt-get update -y
```

```
apt-get install -y build-essential cmake gfortran vim nano emacs
```

```
## Make some directories
```

```
mkdir -p /extra01
```

```
mkdir -p /extra04
```

What is (usually) required for a user to build with apptainer?

- Some combination of
 - Being root
 - sudo
 - Having your username in /etc/subuid
 - Having the fakeroot command installed

What happens if these are not met?

```
[tkaiser@ip-172-18-107-40 ~]$ apptainer build ub2_1.sif ub2.def
INFO:      User not listed in /etc/subuid, trying root-mapped namespace
INFO:      fakeroot command not found

INFO:      Installing some packages may fail
INFO:      Starting build...
INFO:      Fetching OCI image...
28.3MiB / 28.3MiB [=====] 100 % 6.0 MiB/s 0s
INFO:      Extracting OCI image...
INFO:      Inserting Apptainer configuration...
INFO:      Copying ub2.def to /nopt/apps/ub2.def
INFO:      Running post scriptlet
+ apt-get update -y
E: setgroups 65534 failed - setgroups (1: Operation not permitted)
E: setegid 65534 failed - setegid (22: Invalid argument)
E: seteuid 42 failed - seteuid (22: Invalid argument)
...
...
...
FATAL:      While performing build: while running engine: while running %post section: exit status 100
```

Full Build attempt

```
[tkaiser@server ~]$apptainer build broken.sif broken.def
INFO:   User not listed in /etc/subuid, trying root-mapped namespace
INFO:   fakeroot command not found
INFO:   Installing some packages may fail
INFO:   Starting build...
Copying blob 4b3ffd8ccb52 skipped: already exists
Copying config 97bed23a34 done
Writing manifest to image destination
2025/11/05 14:26:30 info unpack layer: sha256:4b3ffd8ccb5201a0fc03585952effb4ed2d1ea5e704d2e7330212fb8b16c86a3
INFO:   Running post scriptlet
+ apt-get update -y
E: setgroups 65534 failed - setgroups (1: Operation not permitted)
E: setegid 65534 failed - setegid (22: Invalid argument)
E: seteuid 42 failed - seteuid (22: Invalid argument)
E: setgroups 0 failed - setgroups (1: Operation not permitted)
Reading package lists... Done
W: chown to _apt:root of directory /var/lib/apt/lists/partial failed - SetupAPTPartialDirectory (22: Invalid argument)
W: chown to _apt:root of directory /var/lib/apt/lists/auxfiles failed - SetupAPTPartialDirectory (22: Invalid argument)
E: setgroups 65534 failed - setgroups (1: Operation not permitted)
E: setegid 65534 failed - setegid (22: Invalid argument)
E: seteuid 42 failed - seteuid (22: Invalid argument)
E: setgroups 0 failed - setgroups (1: Operation not permitted)
E: Method gave invalid 400 URI Failure message: Failed to setgroups - setgroups (1: Operation not permitted)
E: Method gave invalid 400 URI Failure message: Failed to setgroups - setgroups (1: Operation not permitted)
E: Method http has died unexpectedly!
E: Sub-process http returned an error code (112)
FATAL:   While performing build: while running engine: exit status 100
[tkaiser@server ~]$
```

What happens if:

- We create a container (somewhere) installing apptainer in the container
- Can we use apptainer within the container to build new containers?
- Found no evidence this has ever been tried

Yep!

```
# start up a container
[tkaiser@server ~]$apptainer shell bootrh.sif
Apptainer> ls /usr/bin/apptainer
/usr/bin/apptainer
# Run the containers version of apptainer
Apptainer> apptainer build broken.sif broken.def > broken.out 2>&1
```

About 3 minutes later

```
Apptainer> tail -2 broken.out
INFO:      Creating SIF file...
INFO:      Build complete: broken.sif

Apptainer> ls -lt broken.sif
-rwxr-x--- 1 tkaiser tkaiser 400609280 Nov  5 14:56 broken.sif
Apptainer>
```

Test it out

```
[tkaiser@server ~]$apptainer shell broken.sif
Apptainer> cat /etc/*rel*
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=24.04
DISTRIB_CODENAME=noble
DISTRIB_DESCRIPTION="Ubuntu 24.04.3 LTS"
PRETTY_NAME="Ubuntu 24.04.3 LTS"
NAME="Ubuntu"
VERSION_ID="24.04"
VERSION="24.04.3 LTS (Noble Numbat)"
VERSION_CODENAME=noble
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
UBUNTU_CODENAME=noble
LOGO=ubuntu-logo
Apptainer>
Apptainer> gcc --version | grep gcc
gcc (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0
Apptainer>
```

Bootstrap container

- Build with podman
 - Podman can be installed on a laptop and build for HPC
 - Mac or Raspberry Pi podman can build for x86
- Cloud - usually you'll have root and can install podman or even apptainer
- Get an appropriate container from a repo or colleague
 - Might not actually need to contain apptainer
- Build a container containing apptainer directly?

Start with podman

- Assume we are an ARM machine (Raspberry Pi) running some form of ubuntu and we want to target x86

```
sudo apt update
sudo apt install podman
sudo apt install qemu-user-static binfmt-support
sudo update-binfmts --enable qemu-i386
sudo update-binfmts --enable qemu-x86_64
```

```
tkaiser@pie:~$ cat Containerfile
```

```
FROM ubuntu:latest
```

```
RUN apt-get update && apt-get install -y  wget git gcc gfortran g++ make python3 cmake pip
```

```
## Add apptainer
```

```
RUN apt install -y software-properties-common
```

```
RUN add-apt-repository -y ppa:apptainer/ppa
```

```
RUN apt -y update
```

```
RUN apt install -y apptainer
```

Building

On the Raspberry Pi...

```
podman build --platform=linux/x86_64 -t pi03:1.1 .  
podman images  
podman save --output images.tar b150d40b09c5  
  
zip images.tgz images.tar  
scp images.tgz tkaiser@server:/home/tkaiser
```

On the HPC platform...

```
unzip images.tgz  
apptainer build images.sif docker-archive://images.tar  
apptainer shell images.sif  
  
Apptainer> which apptainer  
/usr/bin/apptainer  
Apptainer>
```


On a cloud machine (AWS running RedHat 9.x)

```
sudo dnf -y update
sudo dnf install -y which
sudo dnf -y install epel-release
sudo dnf -y install apptainer-suid
sudo dnf -y group install "Development Tools" || echo nope Development Tools
sudo dnf -y upgrade --refresh
sudo dnf -y install openssh-server || echo nope openssh-server
sudo dnf -y install openssh || echo nope openssh
sudo dnf -y install git
sudo dnf -y install cmake
sudo dnf -y install make
sudo dnf -y install wget
sudo dnf -y install findutils
sudo dnf -y install gfortran
sudo dnf -y update python
sudo dnf -y install zlib-ng || echo nope  zlib-ng
sudo dnf -y install procps
sudo dnf -y install perl
sudo dnf -y install sed
sudo subscription-manager repos --enable codeready-builder-for-rhel-9-$(arch)-rpms
sudo dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm -y
sudo dnf install -y apptainer
sudo dnf install -y podman
```

On a cloud machine (AWS running RedHat 9.x)

```
[tkaiser@ip-172-18-107-38 ~]$ cat boot.def
```

```
bootstrap: docker
```

```
from: rockylinux:9.3.20231119
```

```
%post
```

```
dnf -y install epel-release
```

```
dnf -y install apptainer-suid
```

```
[tkaiser@ip-172-18-107-38 ~]$
```

```
[tkaiser@ip-172-18-107-38 ~]$ apptainer build boot.sif boot.def
```

```
...
```

```
...
```

```
INFO:      Creating SIF file...
```

```
[=====] 100 %
```

```
INFO:      Build complete: boot.sif
```

```
[tkaiser@ip-172-18-107-38 ~]$
```

```
[tkaiser@ip-172-18-107-38 ~]$ apptainer shell boot.sif
```

```
Apptainer> ls /usr/bin/apptainer
```

```
/usr/bin/apptainer
```

```
Apptainer>
```

Our bootstrap is Going in Circles?

- Our goal is to get to a point where we can build containers with out invoking privileges at any point
- We assumed that you could install podman and or apptainer in you personal machine and use these to build our base container
- Podman normally relies on having a user in the /etc/subuid - requiring privilege to set that up
- Apptainer build usually requires /etc/subuid or fake root
- Can we get around this limitation for a "rootless" bootstrap
- Yep

What happens if these are not met?

```
[tkaiser@ip-172-18-107-40 ~]$ apptainer build ub2_1.sif ub2.def
INFO:      User not listed in /etc/subuid, trying root-mapped namespace
INFO:      fakeroot command not found

INFO:      Installing some packages may fail
INFO:      Starting build...
INFO:      Fetching OCI image...
28.3MiB / 28.3MiB [=====] 100 % 6.0 MiB/s 0s
INFO:      Extracting OCI image...
INFO:      Inserting Apptainer configuration...
INFO:      Copying ub2.def to /nopt/apps/ub2.def
INFO:      Running post scriptlet
+ apt-get update -y
E: setgroups 65534 failed - setgroups (1: Operation not permitted)
E: setegid 65534 failed - setegid (22: Invalid argument)
E: seteuid 42 failed - seteuid (22: Invalid argument)
...
...
...
FATAL:      While performing build: while running engine: while running %post section: exit status 100
```

Find a suitable container

- A Ubuntu container with these installed will be sufficient to build apptainer from source
 - `wget git gcc gfortran g++ make python3 cmake pip`
 - We use `spack` to build apptainer
 - As a convince we also build `mod` to make it easier to run
- Why does this work?
 - We are adding "stuff" to the container without touching any system files

Recipe

Part 1

Bootstrap: localimage
from: ub0.sif

```
%post  
mkdir /nopt  
cd /nopt
```

```
export BASE=`pwd`  
unset spack  
export SPEC=103025  
rm -rf $BASE/$SPEC  
mkdir -p $BASE/$SPEC  
cd $BASE  
export STARTDIR=`pwd`  
wget https://github.com/spack/spack/releases/download/v1.0.2/spack-1.0.2.tar.gz  
tar -xzf spack-1.0.2.tar.gz  
mv spack-1.0.2 spack$SPEC  
cd spack$SPEC  
export SPACK_ROOT=`pwd`  
export SPACK_ROOT=$BASE/$SPEC/install  
export SPACK_USER_CONFIG_PATH=${SPACK_ROOT}/.spack  
export SPACK_USER_CACHE_PATH=${SPACK_ROOT}/.cache  
export TMPDIR=$SPACK_ROOT/tmp  
mkdir -p $TMPDIR  
. share/spack/setup-env.sh
```

1. Use a simple base image
2. Create a directory for spack
3. Download it
4. Set some important variables
5. Start spack

Recipe Part 2

```
spack config add "modules:default:enable:[tcl]"
spack config add "modules:default:roots:tcl: '$BASE//$SPEC/modules'"
spack config add "config:install_tree:root: '$BASE/$SPEC/install'"

spack install apptainer -suid +libsubid || echo apptainer+libsubid FAILED
spack install apptainer -suid -libsubid || echo apptainer-libsubid FAILED

spack install pkg-config      || echo pkg-config FAILED
spack load pkg-config         || echo load failed
spack install lmod ^lua@5.3   || echo lmod FAILED
```

5. Tell spack where to put stuff
6. Build apptainer
7. While we're at it build lmod
so we can module load
apptainer

Recipe Part 3

8. Create `.singularity.d/env/99-zmine.sh` which is run at container initialization. These additions will start `lmod` and load the `apptainer` module.

```
echo ". `find /nopt/103025/install/*/lmod* -name bash`" >>/.singularity.d/env/99-zmine.sh
```

```
echo 'export MODULEPATH=`find /nopt/103025/modules -name "linux*" -print`' >>/.singularity.d/env/99-zmine.sh
```

```
echo module load apptainer >> /.singularity.d/env/99-zmine.sh
```


It works!

```
[tkaiser@server ~]$apptainer shell ubbase.sif
Apptainer> cat /etc/os-release
PRETTY_NAME="Ubuntu 24.04.3 LTS"
NAME="Ubuntu"
VERSION_ID="24.04"
VERSION="24.04.3 LTS (Noble Numbat)"
VERSION_CODENAME=noble
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
UBUNTU_CODENAME=noble
LOGO=ubuntu-logo
Apptainer>
Apptainer>
Apptainer> /usr/bin/which apptainer
/nopt/103025/install/linux-sapphirerapids/apptainer-1.4.1-q4hzccpt4ajxp7sq3sgtdwrhtcdb6svt/bin/apptainer
Apptainer>
Apptainer> gcc --version | head -1
gcc (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0
Apptainer>
```

Built a container that previously failed

```
Apptainer> apptainer build ub2intern.sif ub2.def > ub2intern.out 2>&1
Apptainer> tail ub2intern.out
julia-1.11.5/LICENSE.md
+ cd /nopt/apps
+ git clone https://github.com/timkphd/examples.git
Cloning into 'examples'...
+ PATH=/nopt/apps/python/bin:/nopt/apps/julia/julia-1.11.5/bin::usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:
INFO: Adding environment to container
INFO: Adding startscript
INFO: Adding runscript
INFO: Creating SIF file...
INFO: Build complete: ub2intern.sif
Apptainer> exit
exit
[tkaiser@server ~]$ls -lt ub2intern.sif
-rwxr-x--- 1 tkaiser tkaiser 1004642304 Nov  6 14:38 ub2intern.sif
[tkaiser@server ~]$
```

Back to Rocky

```
[tkaiser@ip-172-18-107-38 ~]$ cat boot.def
bootstrap: docker
from: rockylinux:9.3.20231119

%post
    dnf -y install epel-release
    dnf -y install apptainer-suid
[tkaiser@ip-172-18-107-38 ~]$
```

This recipe will build even without privilege

Back to Rocky

```
[tkaiser@server ~]$rm boot.sif  
[tkaiser@server ~]$apptainer build boot.sif boot.def > boot.out 2>&1  
[tkaiser@server ~]$tail boot.out
```

Installed:

apptainer-1.4.3-1.el9.x86_64	apptainer-suid-1.4.3-1.el9.x86_64
fakeroot-1.37-1.el9.x86_64	fakeroot-libs-1.37-1.el9.x86_64
fuse3-libs-3.10.2-9.el9.x86_64	libseccomp-2.5.2-2.el9.x86_64
lzo-2.10-7.el9.x86_64	shadow-utils-subid-2:4.9-12.el9.x86_64

Complete!

```
INFO:    Creating SIF file...  
INFO:    Build complete: boot.sif  
[tkaiser@server ~]$apptainer shell boot.sif  
Apptainer> ls /usr/bin/apptainer  
/usr/bin/apptainer  
Apptainer> ls /usr/bin/fakeroot  
/usr/bin/fakeroot  
Apptainer>
```

What if we don't even have apptainer on our system?

- Are we dead in the water?
- Nope
- We have already seen the solution
- We Use spack to build apptainer from source.
- Our apptainer is sufficient to build our Rocky apptainer base image
 - Build general ubuntu images
 - Use as a base for fuller Rocky images
 - BTW... Suse can also self bootstrap

A spack script

Red Hat Enterprise Linux 9.6
Will need tweaking for other OSs

```
[tkaiser@ip-172-18-107-38 dospack]$ cat doit
export BASE=`pwd`
unset spack

rm -rf $BASE/110625a
mkdir -p $BASE/110625a
cd $BASE
export STARTDIR=`pwd`
wget https://github.com/spack/spack/releases/download/v1.0.2/spack-1.0.2.tar.gz
tar -xzf spack-1.0.2.tar.gz
mv spack-1.0.2 spack110625a
cd spack110625a
export SPACK_ROOT=`pwd`
export SPACK_ROOT=$BASE/110625a/install
export SPACK_USER_CONFIG_PATH=${SPACK_ROOT}/.spack
export SPACK_USER_CACHE_PATH=${SPACK_ROOT}/.cache
export TMPDIR=$SPACK_ROOT/tmp
mkdir -p $TMPDIR
. share/spack/setup-env.sh

spack config add "modules:default:enable:[tcl]"
spack config add "modules:default:roots:tcl:$BASE//$SPEC/modules"
spack config add "config:install_tree:root:$BASE/$SPEC/install"
spack external find bash perl grep tar

spack install apptainer -suid +libsubid || echo apptainer+libsubid FAILED
spack install apptainer -suid -libsubid || echo apptainer-libsubid FAILED

spack install lmod ^lua@5.3 || echo lmod FAILED
```

A slurm spack script

Part 1

```
#!/bin/bash
#SBATCH --time=1:00:00
#SBATCH --nodes=1
#SBATCH --exclusive

# Set environment
module purge
module load gcc binutils python
unset spack

# Make a install location and go there
export BASE=`pwd`
BUILD=`date +"%m%d%y"_a`
rm -rf $BASE/$BUILD
mkdir -p $BASE/$BUILD
cat $0 > $BASE/$BUILD/script
cd $BASE

# Get spack; do initial setup and start it
wget https://github.com/spack/spack/releases/download/v1.0.2/spack-1.0.2.tar.gz
tar -xzf spack-1.0.2.tar.gz
mv spack-1.0.2 spack$BUILD
cd spack$BUILD
export SPACK_ROOT=`pwd`
export SPACK_ROOT=$BASE/$BUILD/install
export SPACK_USER_CONFIG_PATH=${SPACK_ROOT}/.spack
export SPACK_USER_CACHE_PATH=${SPACK_ROOT}/.cache
export TMPDIR=$SPACK_ROOT/tmp
mkdir -p $TMPDIR
. share/spack/setup-env.sh
```

A slurm spack script

Part 2

#2

```
# Set up modules and install location
spack config add "modules:default:enable:[tcl]"
spack config add "modules:default:roots:tcl: '$BASE/$BUILD/modules'"
spack config add "config:install_tree:root: '$BASE/$BUILD/install'"

# Don't rebuild these
spack external find bash perl grep tar

# Preinstall libseccomp. This might not be necessary or
# might even break some builds. It is required on RH-8.8.
# Note: this could be specified on the apptainer install line.
spack install libseccomp@2.3.3

#spack install apptainer -suid +libsubid || echo apptainer+libsubid FAILED
spack install apptainer -suid -libsubid || echo apptainer-libsubid FAILED
```


Put it all together

- Run slurm/spack script to build a personal copy apptainer
- Build a Rocky base container
- Build a Suse base container
- Use Rocky base container to build a Ubuntu base container
- Use the base containers to build "complete" containers with MPICH and an example MPI code
- Run the example MPI code
- Show a slurm script for building containers
 - git clone <https://github.com/timkphd/examples.git>
 - cd examples/apptainer/full

Our build/run script

```
[tkaiser@z1c1s7b0n1 buildup]$vi dobuild
```

```
module purge
module use /kfs2/projects/hpcapps/tkaiser/appt/110725_a/\
moduleslinux-rhel8-sapphirerapids/
module load apptainer

apptainer build r0.sif r0.def
apptainer build suse0.sif suse0.def
apptainer exec r0.sif apptainer build ub0.sif ub0.def

for os in r suse ub ; do
    echo ++++++ $os ++++++
    apptainer exec ${os}0.sif apptainer build ${os}1.sif ${os}1.def
done

for os in r suse ub ; do
    echo ++++++ $os ++++++
    srun -e /dev/null -n 4 apptainer exec  ${os}1.sif hellof
done
```

r0.def

```
bootstrap: docker  
from: rockylinux:9.3.20231119
```

```
##files  
#/nopt/xalt/current/lib64 /nopt/xalt/current/lib64
```

```
%post  
    echo "Installing packages into the Rocky Linux 9.3 container"  
    dnf -y install which  
    dnf -y install epel-release  
    dnf -y install apptainer-suid  
    dnf -y install cmake  
    dnf -y install make  
    dnf -y install wget  
    dnf -y install findutils  
    dnf -y install gfortran  
    dnf -y update python  
    dnf -y install procps  
    dnf -y install perl  
    dnf -y install sed
```

suse0.def

```
bootstrap: docker
from: registry.suse.com/bci/gcc:14
```

```
%labels
    Maintainer Tim Kaiser
    Version 1.0
```

```
%help
    This is a suse Linux container with apptainer
```

```
%post

zypper addrepo https://download.opensuse.org/repositories/filesystems/SLE_15_SP6/filesystems.repo
zypper --gpg-auto-import-keys refresh
zypper --non-interactive install squashfuse
```

```
zypper --non-interactive dup
zypper addrepo https://download.opensuse.org/repositories/home:mslacken:pr/openSUSE_Tumbleweed/home:mslacken:pr.repo
zypper --gpg-auto-import-keys refresh
zypper --non-interactive install apptainer
zypper --non-interactive install curl
zypper --non-interactive install libcurl-devel
zypper --non-interactive install wget
zypper --non-interactive install libcryptopp-devel
zypper --no-gpg-checks --gpg-auto-import-keys --non-interactive addrepo https://download.opensuse.org/repositories/devel:languages:go/15.6/devel:languages:go.repo
zypper --gpg-auto-import-keys refresh
zypper --gpg-auto-import-keys addrepo https://download.opensuse.org/repositories/editors/15.6/editors.repo
zypper --gpg-auto-import-keys refresh
zypper --non-interactive install nano
zypper --non-interactive install python3
zypper --non-interactive install xz
zypper --non-interactive install bzip2
zypper --non-interactive install libbz2-devel
zypper --non-interactive install patch
zypper --non-interactive install openssl
zypper --non-interactive install openssl-devel
zypper --non-interactive install perl
zypper --non-interactive install vim
zypper --non-interactive install squashfs
zypper --non-interactive install libasm-devel
zypper --non-interactive install libdw-devel
zypper --non-interactive install cpp
```

ub0.def

```
bootstrap: docker  
from: ubuntu:latest
```

```
%post  
apt update  
apt install -y wget git gcc gfortran g++ make python3 cmake pip  
apt install -y software-properties-common  
add-apt-repository -y ppa:apptainer/ppa  
apt update  
apt install -y apptainer
```

r0.def

```
bootstrap: localimage
from: r0.sif
#Based on https://apptainer.org/docs/user/main/mpi.html

# optionally install apptainer from source
# you will want to install lmod also
#%files
#script /nopt/scripts/script

%environment
# Point to MPICH binaries, libraries man pages
export MPICH_DIR=/opt/mpich
export PATH="$MPICH_DIR/bin:$PATH"
export LD_LIBRARY_PATH="$MPICH_DIR/lib:$LD_LIBRARY_PATH"
export MANPATH=$MPICH_DIR/share/man:$MANPATH
%post
dnf -y install nano which vim

echo "Installing required packages..."
dnf install -y wget git bash gcc make

# Information about the version of MPICH to use
export MPICH_VERSION=4.1.1
export MPICH_URL="http://www.mpich.org/static/downloads/$MPICH_VERSION/mpich-$MPICH_VERSION.tar.gz"
export MPICH_DIR=/opt/mpich

echo "Installing MPICH..."
mkdir -p /tmp/mpich
mkdir -p /opt
# Download
cd /tmp/mpich && wget -O mpich-$MPICH_VERSION.tar.gz $MPICH_URL && tar xzf mpich-$MPICH_VERSION.tar.gz --no-same-owner
# Compile and install
cd /tmp/mpich/mpich-$MPICH_VERSION && ./configure --prefix=$MPICH_DIR && make -j$(nproc) install

# Set env variables so we can compile our application
export PATH=$MPICH_DIR/bin:$PATH
export LD_LIBRARY_PATH=$MPICH_DIR/lib:$LD_LIBRARY_PATH

echo "Compiling the MPI application..."
mkdir -p /nopt
cd /nopt && git clone https://github.com/timkphd/examples.git
mkdir /nopt/exec
cd /nopt/examples/mpi && mpif90 hellof.f90 -o /nopt/exec/hellof
cd /nopt/examples/mpi && mpicc helloc.c -o /nopt/exec/helloc

echo PATH=/nopt/exec:$PATH >> /.singularity.d/env/99-zmine.sh
```

suse0.def

```
bootstrap: localimage
from: suse0.sif
#Based on https://apptainer.org/docs/user/main/mpi.html

# optionally install apptainer from source
# you will want to install lmod also
#%files
#script /nopt/scripts/script

%environment
# Point to MPICH binaries, libraries man pages
export MPICH_DIR=/opt/mpich
export PATH="$MPICH_DIR/bin:$PATH"
export LD_LIBRARY_PATH="$MPICH_DIR/lib:$LD_LIBRARY_PATH"
export MANPATH="$MPICH_DIR/share/man:$MANPATH"

%post
zypper --non-interactive install nano which vim

echo "Installing required packages..."
zypper --non-interactive install wget git bash gcc make

# Information about the version of MPICH to use
export MPICH_VERSION=4.1.1
export MPICH_URL="http://www.mpich.org/static/downloads/$MPICH_VERSION/mpich-$MPICH_VERSION.tar.gz"
export MPICH_DIR=/opt/mpich

echo "Installing MPICH..."
mkdir -p /tmp/mpich
mkdir -p /opt
# Download
cd /tmp/mpich && wget -O mpich-$MPICH_VERSION.tar.gz $MPICH_URL && tar xzf mpich-$MPICH_VERSION.tar.gz --no-same-owner
# Compile and install
cd /tmp/mpich/mpich-$MPICH_VERSION && ./configure --prefix=$MPICH_DIR && make -j$(nproc) install

# Set env variables so we can compile our application
export PATH=$MPICH_DIR/bin:$PATH
export LD_LIBRARY_PATH=$MPICH_DIR/lib:$LD_LIBRARY_PATH

echo "Compiling the MPI application..."
mkdir -p /nopt
cd /nopt && git clone https://github.com/timkphd/examples.git
mkdir /nopt/exec
cd /nopt/examples/mpi && mpif90 hellof.f90 -o /nopt/exec/hellof
cd /nopt/examples/mpi && mpicc helloc.c -o /nopt/exec/helloc

echo PATH=/nopt/exec\: $PATH >> /.singularity.d/env/99-zmine.sh
```

ub1.def

```
bootstrap: localimage
from: ub0.sif
#Based on https://apptainer.org/docs/user/main/mpi.html

# optionally install apptainer from source
# you will want to install lmod also
#%files
#script /nopt/scripts/script

%environment
# Point to MPICH binaries, libraries man pages
export MPICH_DIR=/opt/mpich
export PATH="$MPICH_DIR/bin:$PATH"
export LD_LIBRARY_PATH="$MPICH_DIR/lib:$LD_LIBRARY_PATH"
export MANPATH="$MPICH_DIR/share/man:$MANPATH"

%post
apt update -y ; apt install -y nano which vim

echo "Installing required packages..."
export DEBIAN_FRONTEND=noninteractive
apt-get update && apt-get install -y wget git bash gcc gfortran g++ make python3-dev

# Information about the version of MPICH to use
export MPICH_VERSION=4.1.1
export MPICH_URL="http://www.mpich.org/static/downloads/$MPICH_VERSION/mpich-$MPICH_VERSION.tar.gz"
export MPICH_DIR=/opt/mpich

echo "Installing MPICH..."
mkdir -p /tmp/mpich
mkdir -p /opt
# Download
cd /tmp/mpich && wget -O mpich-$MPICH_VERSION.tar.gz $MPICH_URL && tar xzf mpich-$MPICH_VERSION.tar.gz
# Compile and install
cd /tmp/mpich/mpich-$MPICH_VERSION && ./configure --prefix=$MPICH_DIR && make -j$(nproc) install

# Set env variables so we can compile our application
export PATH=$MPICH_DIR/bin:$PATH
export LD_LIBRARY_PATH=$MPICH_DIR/lib:$LD_LIBRARY_PATH

echo "Compiling the MPI application..."
mkdir -p /nopt
cd /nopt && git clone https://github.com/timkphd/examples.git
mkdir /nopt/exec
cd /nopt/examples/mpi && mpif90 hellof.f90 -o /nopt/exec/hellof
cd /nopt/examples/mpi && mpicc helloc.c -o /nopt/exec/helloc

echo PATH=/nopt/exec:$PATH >> /.singularity.d/env/99-zmine.sh
```


R1 output

```
+++++ r ++++++
Hello from z1c1s7b0n1 (F)          2  of          4
MPICH Version:      4.1.1
MPICH Release date: Mon Mar  6 14:14:15 CST 2023
MPICH ABI:          15:0:3
MPICH Device:       ch4:ofi
MPICH configure:    --prefix=/opt/mpich
MPICH CC:           gcc
MPICH CXX:          g++
MPICH F77:          gfortran
MPICH FC:           gfortran

compiler: GCC version 11.5.0 20240719 (Red Hat 11.5.0-5)

Hello from z1c1s7b0n1 (F)          0  of          4
Hello from z1c1s7b0n1 (F)          3  of          4
Hello from z1c1s7b0n1 (F)          1  of          4
SUCCESS
```

Suse output

```
+++++ suse +++++
MPICH Version:      4.1.1
MPICH Release date: Mon Mar  6 14:14:15 CST 2023
MPICH ABI:          15:0:3
MPICH Device:       ch4:ofi
MPICH configure:    --prefix=/opt/mpich
MPICH CC:           gcc      -O2
MPICH CXX:          g++      -O2
MPICH F77:          gfortran -O2
MPICH FC:           gfortran -O2
```

compiler: GCC version 14.3.0

```
Hello from z1c1s7b0n1 (F)      0  of      4
Hello from z1c1s7b0n1 (F)      1  of      4
Hello from z1c1s7b0n1 (F)      2  of      4
Hello from z1c1s7b0n1 (F)      3  of      4
SUCCESS
```

Ub1 output

```
+++++ ub ++++++
Hello from z1c1s7b0n1 (F)          1  of          4
MPICH Version:      4.1.1
MPICH Release date: Mon Mar  6 14:14:15 CST 2023
MPICH ABI:          15:0:3
MPICH Device:       ch4:ofi
MPICH configure:    --prefix=/opt/mpich
MPICH CC:           gcc      -O2
MPICH CXX:          g++      -O2
MPICH F77:          gfortran -O2
MPICH FC:           gfortran -O2

compiler: GCC version 13.3.0

Hello from z1c1s7b0n1 (F)          0  of          4
Hello from z1c1s7b0n1 (F)          2  of          4
Hello from z1c1s7b0n1 (F)          3  of          4
SUCCESS
```

Slurm script for building

```
#!/bin/bash
#SBATCH --time=2:00:00
#SBATCH --partition=shared
#SBATCH --nodes=1
#SBATCH --tasks-per-node=20
#SBATCH --mem=50G
```

```
ml apptainer
if [ -z "$APPTVER" ] ; then
    export APPTVER=ubuntu
fi
if [ -z "$BUILD" ] ; then
    export BUILD=dyninst
fi
echo BUILD=$BUILD
echo APPTVER=$APPTVER
cat $0 > $SLURM_JOBID.script
cat $BUILD.def > $SLURM_JOBID.def
printenv > $SLURM_JOBID.env
```

```
if [[ "$APPTVER" = "ubuntu" ]] ; then
    apptainer exec ub0.sif apptainer build $SLURM_JOBID.sif $BUILD.def
fi
if [[ "$APPTVER" = "redhat" ]] ; then
    apptainer exec r0.sif apptainer build $SLURM_JOBID.sif $BUILD.def
fi
if [[ "$APPTVER" = "suse" ]] ; then
    apptainer exec suse0.sif apptainer build $SLURM_JOBID.sif $BUILD.def
fi
echo DONE
```

- **Can use 1 of 3 OS versions.**
 - **Default = Ubuntu**
 - **Also have Rocky and Suse**
 - **Found it works best if you use the target OS to do the build**
- **Export APPTVER (builder) and BUILD (recipe) before running the script**
- **The new container name is based on the SLURM_JOBID**
- **Also saves a copy of the recipe**

Summary

- Introduced containers
- Discussed "expected" requirements for building containers with apptainer
- Showed it is possible to build a container with apptainer inside of another apptainer container
- Presented "bootstrap" containers
- Showed how to get a bootstrap using podman and cloud services
- Showed how to get a bootstrap using just using apptainer for Rocky and Suse and Ubuntu bootstrap using Rocky
- Showed how to build apptainer from source using spack
- Built and ran containers with MPI applications

Final Source and slides

- git clone <https://github.com/timkphd/examples.git>
- cd examples/apptainer/full