NHUG: Containers

Ben Matthews

September 6, 2022

NATIONAL CENTER FOR ATMOSPHERIC RESEARCH

This material is based upon work supported by the National Center for Atmospheric Research, which is a major facility sponsored by the National Science Foundation under Cooperative Agreement No. 1852977.

Participant Code of Conduct

Our Pledge

UCAR and NCAR are committed to providing a safe, productive, and welcoming environment for all participants in any conference, workshop, field project or project hosted or managed by UCAR, no matter what role they play or their background. This includes respectful treatment of everyone regardless of gender, gender identity or expression, sexual orientation, disability, physical appearance, age, body size, race, religion, national origin, ethnicity, level of experience, political affiliation, veteran status, pregnancy, genetic information, as well as any other characteristic protected under state or federal law. (*link*)

Expected Behaviors

- All participants are treated with respect and consideration, valuing a diversity of views and opinions
- Be considerate, respectful, and collaborative
- Communicate openly with respect, critiquing ideas rather than individuals and gracefully accepting criticism
- Acknowledging the contributions of others
- Avoid personal attacks directed toward other participants
- Be mindful of your surroundings and of your fellow participants
- Alert UCAR staff and suppliers/vendors if you notice a dangerous situation or someone in distress
- Respect the rules and policies of the project and venue

Welcome!

Thank you for joining us today.

Here are a few things to note before we really get started:

- This tutorial is being recorded and will be available on the CISL website within the next few days.
- If you have questions, please enter them in the chat.
- Please keep your computer audio or phone muted!
- Please turn off your Zoom video (to save bandwidth).

Meta

- The goal of this talk is to solicit information about your needs
- You're expected to participate
- I'll include some tips/examples as an enticement
- (but only if you participate)
- Choose your own adventure – interrupt me and we can skip around

Meta/Disclaimer

- **I promise nothing**
- My intent is to gauge interest in things we might do
- I might propose things that aren't feasible
- I might propose things that are really easy
- If you don't express interest, you won't get anything
- You might not anyway, but hey, maybe is better than no, right?
- help me help you
- **but... I still promise nothing**
- **(and neither does anyone else)**

Meta

- If I seem negative, it's because everything is broken and I could do it better
- Just kidding.. mostly
- All HPC Vendors are awful. I don't intend to single out anyone in particular
  - (Even if they deserve it)
- I don't intend to imply anything about machines present or future
  - (Even when said machines leave puddles on the floor)

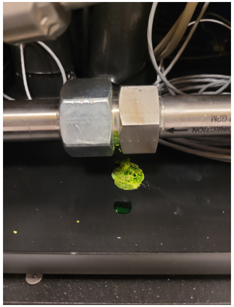

Figure: Supercomputer Blood, photo courtesy of ISGC

Why?

- We've been asked to "provide containers"
- I think everything containers can do can be better done through better system software or application design
- Your job is to show me where I'm wrong
  - Or at least help scope what would be useful
  - Containers are different things to different people
  - Knowing the usecases really helps us find a solution to meet your needs inside our other constraints
    - I probably can't just give you unfettered access to a Docker install (yet?)
    - but I can help you run Docker containers/build Docker containers/etc
  - Us BSD enthusiasts might even prefer jails
- Vendors frequently claim to "support" containers but that support may only be the underlying Linux features.
- As a result, CISL will be heavily engaged in providing a functional solution. Please help us prioritize what capabilities are most important.

Scope

- I work for HSG. We deal with **HPC**
- If you want somewhere to run your enterprise app, talk to EITO
  - HPC might not be the place for PII
  - nor strict uptime requirements
- If you want somewhere to run your **science**, that's in scope
- If you want somewhere to run your gateway/data distribution app/etc we can talk (more on this later)

HPC?

- HPC to me is a primarily batch scientific workload
- Jobs are managed by a scheduler and are ephemeral
- Limited interactive jobs may be permitted (managed by the scheduler) (i.e. viz)
- **Is this your definition of HPC?**

Definitions

- There’s no such thing as a Linux container
- Let’s define a container as a process or group of processes with an alternative view of the system
- Tools used to ensure that such containers are persistently running are called ”orchestration” tools
- Tools that facilitate an alternative view of the system for a group of processes are called ”runtimes”

Alternative View?

- Linux has various subsystems which can be customized for a particular process
- For example, mount namespaces which can be used to provide an alternative root filesystem and set of mounts
- Similar things are possible with other subsystems
  - Process table (process ids, things shown by top/ps)
  - network
  - filesystem
  - user accounts

Containers and Containment

- Containers don't necessarily contain processes in any way
- However, a container runtime might utilize some Linux features such as cgroups to further restrict a process
- HPC schedulers also use cgroups to contain HPC jobs (which may or may not be containers)
- **Do you need a container runtime to contain your HPC jobs more than the batch schedulers already do? If so, Why?**

More on Containers

- My intent is not to teach about containers
- I've done that talk and can do so again, let me know
- Otherwise, Michael Jennings (LANL) does a good job of explaining how things work
- `https://www.youtube.com/watch?v=FFyXdgWXD3A`
- (let him take the flack for contradicting the marketing folks ;-))

Survey Question

- I heard recently that CISL doesn't support containers
- That's sort of true, we don't push any particular implementation
- But we do go out of our way to make sure that it's possible to use various container runtimes on our systems
- **Have you tried any of the following on Cheyenne/Casper?**
    - Rootless Docker
    - Podman
    - Singularity or Apptainer
    - CharlieCloud
    - Minio or other FUSE based filesystem

Supportability

- It would be difficult for us to support any arbitrary contained software stack and ensure that it is compatible with our hardware
- Should CISL have a supported base container?
- If so, what software should be included?
- Or is the goal to bring your own (be your own sysadmin?)

Scheduler Integration

- The commercial release of PBSPro includes an optional plugin to put jobs in containers based on scheduler directives
- This makes the container process a bit more transparent but is (commercial) PBS Specific and a bit complicated
- We don't currently enable this plugin. **Do people prefer this interface?**

```
#!/bin/bash

#PBS -l select=1:ncpus=1:mem=1g:container_engine=singularity
#PBS -l container_image=centos
#PBS -l walltime=1:00:00
#PBS -A SSSG0001

lsb_release
```

Listing 1: PBS Container Plugin

- This won't run your entrypoint based on your container's metadata but it will run your PBS script in the container

Persistent Containers and Orchestration

- There have been asks for Kubernetes support over the years
- Is there a specific usecase?
- Should we allow groups of persistent containers (things that run more than a few days)?

Cloudy interfaces to the HPC Scheduler

- Altair is working on some experimental interfaces to allow PBS to be used as a K8S scheduler
- This would allow you to launch PBS jobs as K8S Pods
- **This seems unnecessarily complicated to me, but would it help anyone?**
- See `https://github.com/PBSPro/kubernetes-pbspro-connector` for how this might look
- Demo: `https://www.youtube.com/watch?v=mxU82PQfxuw`

MPI Programs in Containers

- There are a few approaches to MPI in Containers
  - Dynamically link with a compatible MPI outside the container
    - Best performance most of the time
  - Use a compatible launcher to launch a container provided MPI
  - Use only a container provided MPI
    - Most flexibility but it may be impossible to interface with fancy hardware or libraries (specialized networks, etc)
    - Might be able to link with hardware drivers dynamically via a layer like LibFabrics

Which approach to MPI are you taking with your containers? Do you see a benefit to providing your own MPI?

MPI ABI Compatibility

- A number of MPI vendors have agreed to provide a compatibility layer with MPICH
- This allows you to use open source MPICH while building your container but run with a system/hardware optimized MPI
- Unfortunately these wrappers can be incomplete or poorly tested (*cough* MPT)
- Usage varies by implementation but in general, one bind mounts the optimized MPI over the MPICH build inside the container
  - This does require us to know where MPI is installed both inside and outside the container, making it very difficult to generalize a process for running arbitrary applications
  - **Has anyone tried this? Curious if the user community is settling on a standard place to install things inside their containers?**
- In some cases, it may also be necessary to pass some environment variables or tinker with the dynamic linker
- For our upcoming HPE/Cray systems, HPE advises that we should pass **all** MPI related environment to the container
- Read more about the MPICH ABI compatibility initiative here: `https://www.mpich.org/abi/`
- **Are you finding that the compatibility layers are adequate for your applications?**

LibFabrics and network ABI Compatibility

- What if we don't use MPI but still want full performance?
- What if we want to provide a custom MPI but still get the benefits of native network APIs?
- LibFabrics is supposed to be a generic interface for talking to high speed networks (like verbs but more general)
- Each specific device gets a "provider" which does the hardware specific bits.
- Upcoming networks like Slingshot and IBVerbs based devices (InfiniBand RDMA over Ethernet) have LibFabrics providers
- Unfortunately, LibFabrics has many optional APIs for providers to implement or not and is a bit of a moving target at the moment
- The BSD Socket API is always an option but the performance might be half or less what you'd get from a native interface

Does everyone use MPI or is there a need for something like this at NCAR?

GPU applications in containers

- In the interest of time, it's been suggested that I save this for a GTT meeting if there's interest
- That said, generally GPU applications have all the same problems as high speed networks
- Generally one bind-mounts the drivers from the system and/or installs them into their container
- Mismatching versions tend not to work reliably (especially if the runtime is newer than the kernel drivers)
- NVIDIA does provide some tooling to help with the bind-mounting but they provide it in a form that's not easy to integrate into a multi-user HPC system

Private Networking

- It’s technically possible to establish a private overlay network for a job of containers
- Probably at some performance const
- I haven’t heard a use-case for this, but please let me know if you have one.
- For now, I’ll assume that HPC containers don’t need to do this **(but, again, let me know)**

Building Containers on the HPC environment

- Running containers typically requires only a single user-id and no special privileges
- Building containers on the other hand often involves tools (apt,yum, etc) that want to become other users or do privileged things (set the time, etc)
    - Usually we don't care if these actions actually happen, but the tooling was designed for bare-metal
- This makes it much more difficult to support building containers on a shared system in a remotely secure way
- The first generation tools for this (i.e. Docker) expect to be run as root on the host and run contained code as root
    - Docker allows one to do things like become other users or access filesystems that you might not normally be able to
- The obvious solution is to build on a personal (single-user) computer (Podman Desktop, Docker Desktop, etc)
- Luckily, some progress is (slowly) being made on this problem for shared environments

Do you have access to a place to build containers? Is this something the HPC environment should provide?

Workarounds for Building Containers

- User Namespaces
- Fakeroot and syscall interception
- Special FUSE filesystems (i.e. fuse-overlayfs)
- Dedicated Build systems
    - Cloud
    - I am working on a virtualized login node environment that would accommodate this later this year (remember, I promise nothing)

User Namespaces and UID Mapping

- User namespaces allow us to have map your real UID/GID to a different UID/GID for a group of processes
- It is necessary for security that each user have unique UID/GIDs
- Most "rootless" container runtimes still expect to be able to use a setuid-root binary to map multiple users (without special permission, a user can only map their single UID to a single other UID)
- This is difficult to allow in our environment – we'd need to assign each user 65k UIDs and GIDs for this to be at all reliable
    - This can result in files which are difficult to delete/modify if they're not owned by your "main" UID
    - it also adds an account management burden

fakeroot

- Sometimes all we need is for things like package managers to **think** that they can become other users but don't really care about writing files as multiple users
- For dynamically linked executables it's possible to replace the chown/chmod/etc syscalls with stubs at runtime to allow applications to succeed without multiple UIDs
- Thanks to LANL and ORNL for some of the ideas I'm about to discuss here. Fakeroot has been in the sysadmin toolbox forever but I hadn't thought to use it this way.
- Runtimes are starting to integrate this trick
- **Might this work for your usecase?**

CharlieCloud

- The CharlieCloud ch-image build tool incorporates a number of hacks like fakeroot to build OCI compliant containers without extra permissions
- These hacks generally work for real code (and are constantly getting better) but can still fail in some cases
- For some reason CharlieCloud doesn't seem to have taken off but it's probably the easiest solution to building containers.
- **Does ch-image work for your usecases? If you haven't tried it, why not?**

```bash
#!/bin/bash

echo 'FROM almalinux:8' >> Dockerfile
echo 'RUN yum update -y' >> Dockerfile

ch-image build --force ./
```

Listing 2: Building OCI Images With CharlieCloud

CharlieCloud

- There's probably a better way to do this (chat with the consultants) (maybe a virtualenv) but here's a quick and dirty CharlieCloud install to get you started if you want to try it out (tested on Cheyenne):

```bash
#!/bin/bash

module purge
module load python/3.7.0
pip3 install --user wheel
pip3 install --user requests
git clone https://github.com/hpc/charliecloud.git
cd charliecloud
./autogen.sh --rm-lark
./configure --prefix=/wherever
make
make install
export PATH=/wherever/bin:$PATH
```

Listing 3: Installing CharlieCloud

Podman

- Podman is a runtime which has a UI that is mostly compatible with Docker
- The design is much more friendly to shared systems (and, if I may, less insane)
- Backed by Redhat and quite mature at this point
- While many features aren't realistic to enable on a well secured shared system, we can make it work for what I think HPC users need
- We can manually use fakeroot and other workarounds like CharlieCloud does to avoid some limitations

Aside: Podman on Cheyenne I

- Let's show how fakeroot can work using podman on Cheyenne
- First we need to compile and configure podman
- (I know it's small, the slides will be distributed so you can try this out later)
- First, we compile just enough stuff to run podman

```
#!/bin/bash

module purge
module load python/3.7
module load gnu/7.4.0

pfix=/glade/scratch/matthews/podman-test/prefix
src=/glade/scratch/matthews/podman-test/src

mkdir -p $src
mkdir -p $pfix

cd $src
wget https://go.dev/dl/go1.19.linux-amd64.tar.gz
tar xvzf go*.gz

export PATH=$src/go/bin:$PATH
export GOROOT=$src/go
export PATH=$pfix/bin:$PATH
export PATH=$pfix/sbin:$PATH
```

Aside: Podman on Cheyenne II

```
wget http://ftp.gnu.org/pub/gnu/gperf/gperf-3.1.tar.gz
tar xvzf gperf-*
cd gperf-*
./configure --prefix=$pfix
make
make install
cd $src

git clone https://github.com/seccomp/libseccomp.git
cd libseccomp
git checkout v2.5.4 #necessary else version macros are all 0 and later things are sad
./autogen.sh
./configure --prefix=$pfix
make
make install
export PKG_CONFIG_PATH=$pfix/lib/pkgconfig:$PKG_CONFIG_PATH
cd $src

git clone https://github.com/opencontainers/runc.git
cd runc
make
make install PREFIX=$pfix
cd $src

git clone https://github.com/containers/conmon.git
cd conmon
make
make install PREFIX=$pfix
cd $src
```

Aside: Podman on Cheyenne III

```
git clone https://github.com/containers/podman.git
cd podman
make BUILDTAGS="exclude_graphdriver_btrfs exclude_graphdriver_devicemapper  containers_image_openpgp seccomp"
make install PREFIX=$pfix
cd $src

echo "put $pfix/bin and $pfix/sbin into your $PATH"
```

Listing 4: Podman Build Script

- Next, we need a little bit of configuration

```
vim ~/.config/containers/storage.conf

[storage]
driver="vfs"
runroot="/tmp/matthews/images"
rootless_storage_path="/tmp/matthews/podman"

[storage.options]
ignore_chown_errors="true"
```

Listing 5: Podman Config

Fakeroot demo with Podman

```
podman run --net=host -it docker.io/almalinux /bin/bash

yum install -y openssh #install something that requires root to install

...

Error: Transaction failed

yum install -y epel-release
yum install -y fakeroot

fakeroot yum install -y openssh

...

Installed:
  openssh-8.0p1-13.el8.x86_64

Complete!
```

Storage

- Fundamentally, container storage is just Linux mounts visible only to certain processes
- FUSE allows filesystems to run in userspace using (mostly) a user's regular permissions
  - This was setup on Casper for a while but it's sort of broken right now.
  - **I guess nobody was using it?**
  - yes, I'm looking into what happened
- Some runtimes utilize disk images to store HPC oriented containers
  - Typically containers contain a lot of small files (a parallel filesystem's worst nightmare)
  - loop mounted images can help with this but can have system stability risks
- You may also want to mount your own remote filesystems in your container (s3 maybe?) **Is this a capability people would find useful?**

Software Stack Redistribution and Licensing

- If you publish your container, you're legally redistributing a ton of software, not just your application, but an entire copy of Linux
- It might be somewhat reasonable to assume that open Linux distributions are safe to redistribute
- It's not safe to assume that the HPC stack is safe to redistribute
- We're currently talking with HPE about their software (expect to hear a decision early next year)
- Intel has some published terms for this
- Others are difficult
- Would a containerized subset (based on open software) of the CISL HPC software stack be helpful
- **Do you redistribute your containers? Do you do any license scanning first?**
- **Is licensing/container distribution something you'd like help with?**

Container Registries

- Containers can be distributed as tarballs or some other type of image or via a "registry"
- Checking out a container from a registry from each of a thousand compute nodes is somewhere between rude and infeasible so careful caching is important
- It's possible for NCAR's IP subnet to be banned from popular registries due to excessive use (this has already happened)
- Casper has and upcoming systems are expected to have performant internet access from compute nodes
- **Would it be helpful for CISL to run a registry?**
- **Would it be helpful for CISL to run a public registry?**

Other Issues: libc

- It's become popular to use AlpineLinux for containers because it is small and compact.
- Alpine uses Musl Libc instead of Glibc
- Musl is great, but most HPC systems use glibc so if you try to, say, bind mount in a network library, complications are likely
- **Is this something the community would like to be able to use? Has anyone benchmarked Musl for HPC code use?**
- Remember, I promise nothing. This is probably *hard* due to closed source parts of the stack
- (but I'm really curious)

But I need Docker!!!

- **Why?**
- `alias docker=podman`
- But if you're really sure, since Podman has started eating Docker's lunch, they've been working on rootless-docker
- rootless-docker can probably be made to work on Cheyenne with the same workarounds as podman but it's not as mature and kind of difficult to compile
- You'll probably want to use the `vfs` storage driver
- Send me an email if you really need this and we'll figure it out
- Podman does have an optional daemon to emulate the docker socket if you that's what you need (*cough* Altair)

But I need Singularity!!!

- Did you mean Apptainer? ;-)
- Personally, I'm not optimistic about the long term viability of this project
- It's changed companies/names/been rewritten too many times and has little traction outside HPC
- as a result, I'm not comfortable recommending it if you're starting out
- It's also popular enough that we have a (not well publicized) module for it

  ```
  module load singularity
  ```
- **Are you using Singularity? Why do you prefer it over the other options?**

But I need something else!!!

- Probably you can compile and install it yourself
- In general, we've enabled all the kernel features for this stuff that we **safely** can.
- But please let us know how it goes. We're still trying to figure out what we should support
- The software stack on our upcoming systems should have much better container support in the kernel. (sans pandemic, Cheyenne was supposed to be decommissioned by now. The software is getting kind of old)
- How about that new Sarus thing out of ETH Zurich?
- Feedback ... please...

Questions? Comments? **Feedback?**

- matthews@ucar.edu
- hsg@ucar.edu
- csg@ucar.edu
- On a personal note, I'll be attending a family function later this week so if it seems like I'm ignoring any emails about this, I'm not
- I'll get back to you as soon as I can
- We really do appreciate feedback