SAND2022-7225C

Sandia National Laboratories

Exceptional service in the national interest

Building and Running HPC Containers across the US Department of Energy

Andrew Younge

Center for Computing Research
Sandia National Laboratories

ajyoung@sandia.gov

International Supercomputing Conference 2022

SAND2022-7213 C

UNCLASSIFIED UNLIMITED RELEASE

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

U.S. DEPARTMENT OF ENERGY

NNSA

ECP **Supercontainers** activity as Packaging Technologies

- Ensure container runtimes will be scalable, interoperable, and well integrated across DOE
  - Enable container deployments from laptops to Exascale
  - Assist Exascale applications and facilities leverage containers most efficiently
- Three-fold approach
  - Scalable R&D activities
  - Collaboration with related ST and AD projects
  - Training, Education, and Support
- Activities conducted in the context of interoperability
  - Portable solutions
    - Optimized E4S container images for each machine type
    - Containerized ECP that runs on Astra, A21, El-Capitan, ...
  - Work for multiple container implementations
    - Not picking a “winning” container runtime
  - Multiple DOE facilities at multiple scales

There is a performance-portability continuum

How do we strike the right balance?

- Portable container images can be moved form one resource deployment to another with ease
- Reproducibility is possible
  - Everything (minus kernel) is self-contained
  - Traceability is possible via build manuscripts
  - No image modifications
- Performance can suffer – no optimizations
  - Can't build for AVX512 and run on Haswell
  - Unable to leverage latest GPU drivers

- Performant container images can run at near-native performance compared to natively build applications
- Requires targeted builds for custom hardware
  - Specialized interconnect optimizations
  - Vendor-proprietary software
- Host libraries are mounted into containers
  - Load system MPI library (glibc issues!?)
  - Match accelerator libs to host driver
- Not portable across multiple systems

HPC Container Runtimes today

- Docker is not good fit for running HPC workloads
  - Building with Docker on my laptop is ok!
  - Deployment issues: Security, scheduler integration, etc
- Several different container runtime options in HPC
- All our HPC container runtimes are usable in HPC today
- Each runtime offers different designs and OS mechanisms
  - Storage & mgmt of images
  - User, PID, Mount namespaces
  - Security models
  - Image signing, validation, registries, etc
- New tools emerging for rootless container *builds* as well!

Supercontainers investments in HPC container runtimes

Charliecloud

- multi-stage unprivileged build (0.19)
- fool distribution tools into thinking they're privileged (0.20)
- push to image repositories (0.22)
- architecture-aware pull (0.24)
- automatically make SSH secrets available to build (0.25)
- mount SquashFS images at runtime using FUSE (0.26, upcoming)

Singularity => Apptainer

- HPCng project renamed to Apptainer
- Managing ongoing fork issues in community
- Security fixes and bugfixes ongoing

Shifter

- Initial scalable container launch for Perlmutter
- Further integration with Podman being explored
- Bugfixes and registry compatibility upgrades

Podman

- Developing MOU Red Hat with DOE labs
- Rootless Podman builds on HPC login nodes
- Enabling SIF image support (Singularity compatibility)
- Investigating scalable launch of OCI images
  - Leveraging Shifter's squashfs experience
- Native overlay support

SUPERCONTAINERS

Container runtimes on different DOE systems

ALCF
- Theta: Singularity
- Aurora: Singularity

OLCF
- Summit: Singularity (trial)
- Frontier: Singularity

NERSC
- Cori: Shifter
- Perlmutter: Shifter & Podman

LLNL
- Sierra/Lassen: Singularity (trial)
- Linux clusters: Singularity
- El Capitan: Singularity & Podman

LANL
- Trinity: Charliecloud
- Linux clusters: Charliecloud
- Crossroads: Charliecloud

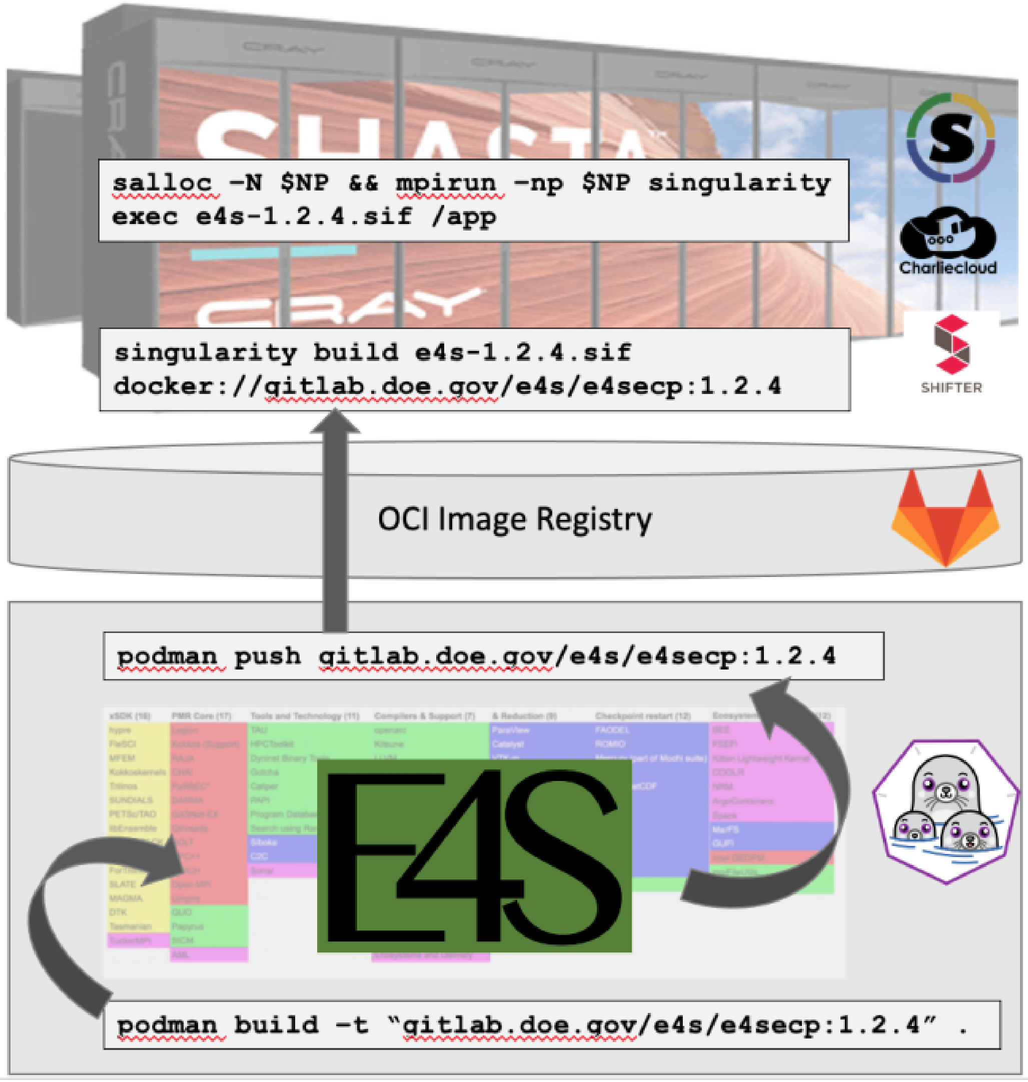
Sandia
- Astra: Singularity, Charliecloud, & Podman
- Linux clusters: Singularity & Podman

Many sites are rolling out container runtimes for users.
We are developing resources to facilitate consistent, performant deployment across sites.

But what about *building* containers for HPC??

Supercontainers created two HPC build solutions: Podman & Charliecloud

- Podman/Buildah provide container build functionality through low privilege
  - UID/GID mappers with shadow-utils
  - Overlay & FUSE for mount operations
- Charliecloud for fully unprivileged build with single UID/GID mapping to UID0
  - Simpler setup, remains entirely unprivileged
  - Requires `fakeroot` injection in container
- Both implementations prototyped and working
- Next Steps
  - Enable E4S container builds directly on DOE/ECP resources
  - Integrate with CI activities & DevOps
- Able to work with facilities to help roll out new capability



Supercontainers team leading container build technologies

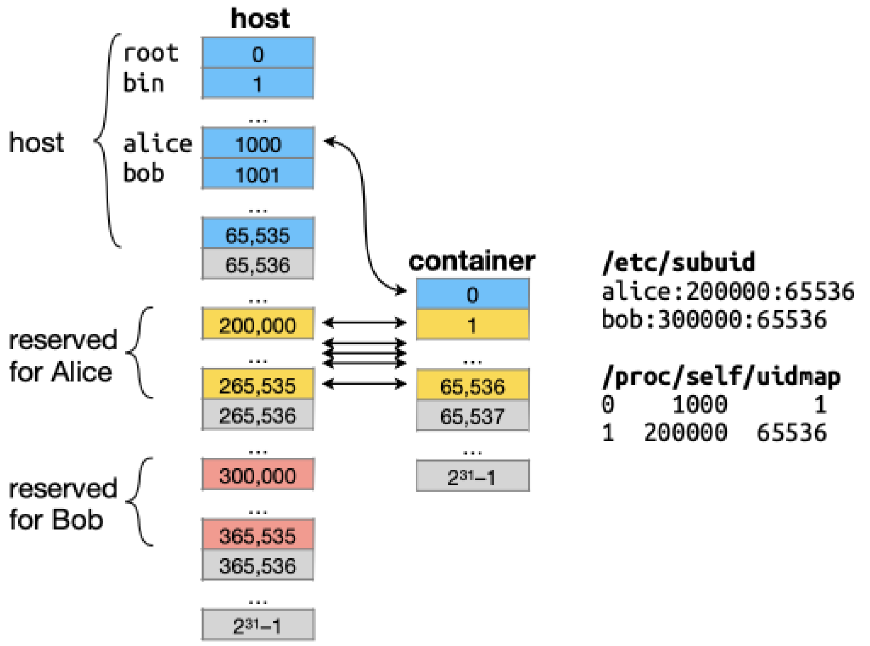
New taxonomy of container privilege levels

7

type	namespaces	setup	IDs in container	examples
I	mount	privileged	UIDs and GIDs shared with host	docker, podman, SHIFTER
II	mount + privileged user	privileged	arbitrary UIDs and GIDs separate from host (but pitfalls)	docker, podman
III	mount + unprivileged user	unprivileged	host EUID & EGID aliased (supplemental GIDs partially functional)	Charliecloud, podman

HPC-focused

only Type III containers are fully unprivileged throughout the container lifetime



Example of Type II user namespace mapping used by container runtimes

From: Reid Priedhorsky, Shane Canon, Tim Randles, and Andrew Younge, Minimizing privilege for building HPC containers, in SC '21: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Nov 2021.

Podman – in – Podman for containerized CI

- Many DOE codes use Gitlab for developing HPC applications
- Need to leverage Continuous Development and Continuous Integration capabilities
  - {build,test,deploy} HPC apps in containers
  - Automatic building container images
- Gitlab CI has git-lab runners, but expect elevated privileges
- Solution: **Podman-in-Podman**
  - The gitlab runner built in a OCI container image
  - Run Rootless Podman to have gitlab-runner think it has root privs
  - Gitlab-runner auto-starts a container build within the 1st container
  - Push resulting container to Gitlab container registry
- Simplified container build & deploy infrastructure with Gitlab
- Solution appliable to SNL as well as greater DOE infrastructure
  - Future integration with DOE Jacamar runners

Extreme-scale Scientific Software Stack (E4S)

- E4S: HPC Software Ecosystem – a curated software portfolio
- A **Spack-based** distribution of software tested for interoperability and portability to multiple architectures
- Available from **source**, **containers**, **cloud, binary caches**
- Leverages and enhances SDK interoperability thrust
- Not a commercial product – an open resource for all
- E4S over time:

• Oct 2018:	E4S 0.1	24 full, 24 partial release products
• Jan 2019:	E4S 0.2	37 full, 10 partial release products
• Nov 2019:	E4S 1.0	50 full, 5 partial release products
• Feb 2020:	E4S 1.1	61 full release products
• Nov 2020:	E4S 20.10	67 full release products
• Feb 2021:	E4S 21.02	67 full release, 4 partial release
• May 2021:	E4S 21.05	76 full release products
– Aug 2021:	E4S 21.08	88 full release products
– **Nov 2021:**	E4S 21.11	91 **full release products**

https://e4s.io

Lead: Sameer Shende (U Oregon)

Also include other products .e.g.,
AI: PyTorch, TensorFlow (CUDA, ROCm)
Co-Design: AMReX, Cabana, MFEM

Our strategy is to focus on Exascale systems

- **We aim to leverage the new containerized PE to enable CI for these environments.**
  - We are experimenting with containerized builds for the Cray environment
- **We have worked with HPE/Cray to enable Spack to autodetect PE components**
  - Metadata for Spack now ships with the PE itself, can be automatically used via spack install --reuse
- **GPU integration across the stack will be an ongoing focus**
  - We are improving our compiler and GPU support model – compiler interoperability is a major focus
- **We aim to have a distribution of *optimized* Spack binaries for these systems by the end of ECP**
  - Spack will work on Exascale systems out of the box
- **We will also have optimized container images that can use these binaries**
  - Users will be able to construct images from E4S packages on demand, run optimized on exascale machines