ISC 19 Tutorial: Getting Started with Containers on HPC

Carlos Eduardo Arango[1], Sameer Shende[2], Shane Canon[3], Andrew J. Younge[4]

[1]Sylabs Inc eduardo@sylabs.io	[2]University of Oregon sameer@cs.uoregon.edu
[3]Lawrence Berkeley National Lab scanon@lbl.gov	[4]Sandia National Labs ajyoung@sandia.gov

Outline

- **14:01 - 14:30 Introduction to Containers in HPC (Younge)**
- 14:31 - 15:00 How to build your first Docker container (Canon)
- 15:00 - 15:30 How to deploy a container on a supercomputer (Canon)
- 15:30 - 16:00 -- Break --
- 16:00 - 16:30 How to build a Singularity container image (Arango)
- 16:30 - 17:00 Running Singularity on a supercomputer & adv features (Arango)
- 17:00 - 17:30 Running an HPC app on the E4S container (Shende)
- 17:30– 18:00 Success Stories and Summary (Canon)

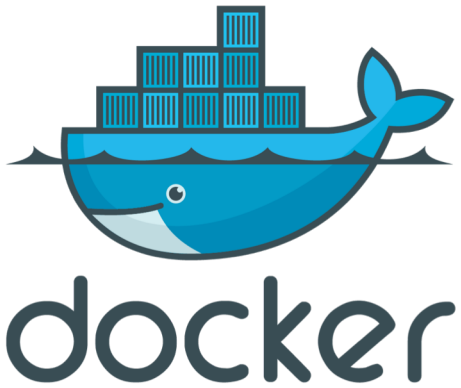
Link: https://ecpcontainers.github.io/isc19-tutorial/

Introduction to Containers in HPC

Andrew Younge

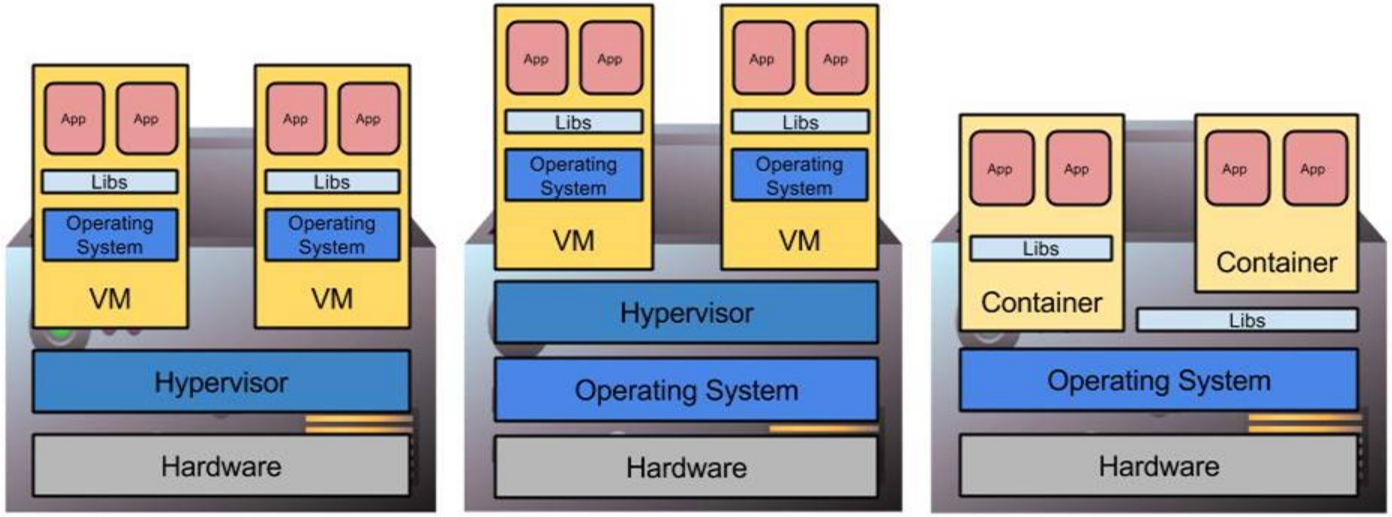
What are containers

- A lightweight collection of executable software that encapsulates everything needed to run a single specific task
  - Minus the OS kernel
  - Based on Linux only
- Processes and all user-level software is isolated
- Creates a portable* software ecosystem
- Think `chroot` on steroids
- Docker most common tool today
  - Available on all major platforms
  - Widely used in industry
  - Integrated container registry via Dockerhub



Hypervisors and Containers

- Type 1 hypervisors insert layer below host OS
- Type 2 hypervisors work as or within the host OS
- Containers do not abstract hardware, instead provide “enhanced chroot” to create isolated environment
- Location of abstraction can have impact on performance
- All enable custom software stacks on existing hardware



Type 1 Hypervisor Type 2 Hypervisor Containers

Background

- Abstracting hardware and software resources has had profound impact on computing
- Virtual Machines to Cloud computing in the past decade
  - Early implementations limited by performance
  - HPC on clouds: FutureGrid, Magellan, Chameleon Cloud, Hobbes, etc
  - Some initial successes, but not always straightforward
- OS-level virtualization a bit different
  - User level code packaged in container, can then be transported
  - Single OS kernel shared across containers and provides isolation
  - Cgroups traditionally multiplexes hardware resources
  - Performance is good, but OS flexibility is limited

Containers in Cloud Industry

- Containers are used to create large-scale loosely coupled services
- Each container runs just 1 user process – “micro-services”
  - 3 httpd containers, 2 DBs, 1 logger, etc
- Scaling achieved through load balancers and service provisioning
- Jam many containers on hosts for increased system utilization
- Helps with dev-ops issues
  - Same software environment for developing and deploying
  - Only images changes are pushed to production, not whole new image (CoW).
  - Develop on laptop, push to production servers
  - Interact with github similar to developer code bases
  - Upload images to ”hub” or “repository” whereby they can just be pulled and provisioned

Containers

- Containers are gaining popularity for software management of distributed systems
- Enable way for developers to specify software ecosystem
- US DOE High Performance Computing (HPC) resources need to support emerging software stacks
  - Applicable to DevOps problems seen with large HPC codes today
  - Support new frameworks & cloud platform services
- But HPC systems are very dissimilar from cloud infrastructure
  - MPI-based bulk synchronous parallel workloads are common
  - Scale-out to thousands of nodes
  - Performance is paramount

Container features in HPC

- **BYOE** - Bring-Your-Own-Environment
  - Developers define the operating environment and system libraries in which their application runs.
- **Composability**
  - Developers explicitly define how their software environment is composed of modular components as container images,
  - Enable reproducible environments that can potentially span different architectures.
- **Portability**
  - Containers can be rebuilt, layered, or shared across multiple different computing systems
  - Potentially from laptops to clouds to advanced supercomputing resources.
- **Version Control Integration**
  - Containers integrate with revision control systems like Git
  - Include not only build manifests but also with complete container images using container registries like Docker Hub.

Container features not wanted in HPC

- **Overhead**
  - HPC applications cannot incur significant overhead from containers
- **Micro-Services**
  - Micro-services container methodology does not apply to HPC workloads
  - 1 application per node with multiple processes or threads per container
- **On-node Partitioning**
  - On-node partitioning with cgroups is not necessary (yet?)
- **Root Operation**
  - Containers allow root-level access control to users
  - In supercomputers this is unnecessary and a significant security risk for facilities
- **Commodity Networking**
  - Containers and their network control mechanisms are built around commodity networking (TCP/IP)
  - Supercomputers utilize custom interconnects w/ OS kernel bypass operations

HPC Containers

- Docker not good fit for running HPC workloads
  - Security issues
    - Can’t allow root on shared resources
  - Lack of HPC architecture support
    - No batch integration
    - Assumes local resources
    - Assumes commodity TCP/IP
- Many different container options in HPC

Shifter Singularity Charliecloud …

Developing Container Vision

- Support software dev and testing on laptops
  - Working builds that then can run on supercomputers
  - Dev time on supercomputers is expensive
  - May also leverage VM/binary translation
- Let developers specify how to build the env AND app
  - Import and run container on target platform
  - Many containers, but can have different code “branches”
  - Not bound to vendor and sysadmin software
- Focus on Interoperability
- Provide containerized services coupled with simulations
  - Developing mechanisms to support services
- Performance matters
  - Want to manage permutations of architectures and compilers
  - Ensure container implementations on HPC are performant
  - Keep features to support future complete workflows

Container DevOps

- Impractical for apps to use large-scale supercomputers for DevOps and/or testing
  - HPC resources have long batch queues
  - Dev time commonly delayed as a result
- Create deployment portability with containers
  - Develop Docker containers on your laptop or workstation
  - Leverage Gitlab registry services
    - Separate networks maintain separate registries
  - Import to target deployment
    - Leverage local resource manager

This tutorial will show you:

- How to build your first Docker container.
- How to run a Docker container on a supercomputer with Shifter.
- How to build your first Singularity container.
- How to run a container on a supercomputer with Singularity.
  - And work with some Sylabs cloud features
- How to use the Extreme-scale Scientific Software Stack (E4S) container image.
  - And a bit about Spack
- And maybe some best practices and lessons learned.

Tutorial Link

https://ecpcontainers.github.io/isc19-tutorial/

CANOPIE HPC WORKSHOP

Containers and New Orchestration Paradigms for Isolated Environments in HPC

- https://canopie-hpc.nersc.gov/

- Submission Deadline: September 2nd, 2019

Number	IP	Username	Password	Number	IP	Username	Password
1	34.214.92.245	tutorial	HPCLinux12!	21	54.184.113.47	tutorial	HPCLinux12!
2	54.213.158.55	tutorial	HPCLinux12!	22	34.218.232.221	tutorial	HPCLinux12!
3	52.36.223.133	tutorial	HPCLinux12!	23	34.217.105.148	tutorial	HPCLinux12!
4	34.222.134.220	tutorial	HPCLinux12!	24	54.148.128.9	tutorial	HPCLinux12!
5	52.32.19.184	tutorial	HPCLinux12!	25	54.201.33.248	tutorial	HPCLinux12!
6	54.149.163.157	tutorial	HPCLinux12!	26	18.237.255.117	tutorial	HPCLinux12!
7	54.185.253.100	tutorial	HPCLinux12!	27	34.217.180.124	tutorial	HPCLinux12!
8	52.38.149.225	tutorial	HPCLinux12!	28	52.25.78.125	tutorial	HPCLinux12!
9	34.216.223.223	tutorial	HPCLinux12!	29	54.187.155.155	tutorial	HPCLinux12!
10	34.220.187.130	tutorial	HPCLinux12!	30	34.218.249.188	tutorial	HPCLinux12!
11	52.12.80.43	tutorial	HPCLinux12!	31	34.219.144.68	tutorial	HPCLinux12!
12	52.11.209.11	tutorial	HPCLinux12!	32	34.212.185.214	tutorial	HPCLinux12!
13	34.215.208.182	tutorial	HPCLinux12!	33	54.202.195.99	tutorial	HPCLinux12!
14	34.222.129.101	tutorial	HPCLinux12!	34	54.202.6.162	tutorial	HPCLinux12!
15	34.213.69.137	tutorial	HPCLinux12!	35	18.236.176.142	tutorial	HPCLinux12!
16	54.201.140.162	tutorial	HPCLinux12!	36	52.39.76.233	tutorial	HPCLinux12!
17	54.200.17.194	tutorial	HPCLinux12!	37	18.237.253.123	tutorial	HPCLinux12!
18	34.213.235.30	tutorial	HPCLinux12!	38	34.210.187.148	tutorial	HPCLinux12!
19	34.211.225.174	tutorial	HPCLinux12!	39	34.208.254.73	tutorial	HPCLinux12!
20	34.209.150.45	tutorial	HPCLinux12!	40	34.219.72.114	tutorial	HPCLinux12!

Questions?

Next: learn how to work with your first container!