Sylabs

How to Choose the Best Container for HPC Workloads

Introduction

Containerization is not a new concept; the roots are said to go back to the late 1970s. In recent years, however, the importance and use of containers has skyrocketed. Containerization, which is tied closely to application growth across environments, is popular because it helps simplify how organizations interact with their software. Container technologies provide an easier way to build, test, and run many applications on a single server, helping to reduce both hardware and software integration costs. Containers bundle a software program's code with all its necessary components, such as frameworks, libraries, and other dependencies, so that an application runs in a private space or "container". The approach is especially beneficial for moving applications from on-prem servers to the cloud and hybrid deployments.

This "buyer's guide" explores the benefits and pitfalls of containerization in high performance computing (HPC) environments, leading tools for HPC containerization, and key features to look for. Read on for key insights on choosing the right containerization tools for your environment.

Benefits of containers

☑ Less overhead

Containers don't require full operating system images, so they use fewer system resources.

☑ Improved business agility

Rapidly introduce new products and services, deliver exceptional customer service, and respond to changing market demands.

☑ More efficient application development

Containers automate various application development stages, underpinning a more agile infrastructure that enables developers to quickly iterate application features, and more rapidly deploy bug fixes and security updates.

☑ Improved security

Containerization facilitates agile security; by updating application libraries and interdependencies in a continuous cycle, it helps DevSecOps teams reduce exposure to bugs and common vulnerabilities. Teams can update individual containers quickly without impacting other services in an application stack.

The emergence of containers and HPC

Over the last ten years, public cloud, large-scale enterprise, and small-scale private data centers have all relied on container technologies to maximize infrastructure elasticity. "Mobility-of-compute" capabilities in containers enable a high degree of application portability flexibility, whether running the latest versions of an application on legacy hardware, or legacy versions of an application on new hardware.

On HPC systems, containers enable any size or type of organization to perform big data analytics on the four main components of HPC; compute, storage, memory and shared networking. It's especially helpful for organizations and teams that want to focus on research and not the technology of the underlying tools. Anyone with basic system administration or programming experience can easily package software applications, libraries and scripts into a container for quick and consistent testing on a laptop or a workstation, and then deploy them on the high-performance, super-computing production clusters.

Containerization is transforming how scientists and researchers across a variety of industries are using HPC more efficiently in pursuit of break-through discoveries. HPC users in finance, oil and gas, pharmaceutical, manufacturing, and DNA sequencing have found that containerization technology helps them deploy performance-intensive workloads more easily and securely. But that doesn't mean that containerization is the right tool for the job for all HPC use cases.

Pitfalls to containerization in HPC

In HPC, workload portability and efficiency often take a back seat to other considerations, such as easy access to underlying GPU and networking hardware, repeatability, security, and scaling requirements. The problem is that not all container technologies are designed for batch-performance-intensive use cases and unsurprisingly there aren't any one-size-fits-all container solutions for HPC.

Containerization in HPC also comes with a host of administrative considerations. Many traditional HPC systems are designed to be shared among many users, where you might have a mix of classified and unclassified containers running next to one another. It is in these scenarios where you want to avoid the ability of one container to elevate execution privileges. For example,

when someone needs to build a container for their latest software, they could decide to pull a shared-source container from an uncurated location, which could allow an attacker to log in with elevated privileges and gain access to the entire system.

For architects and administrators of HPC centers, the integrity of containers is just as important as providing infrastructure services to their user community. And as with any software, lifecycle management becomes part of the security posture of an organization. Containers with outdated libraries or applications would then be vulnerable. Moreover, procedures for verifying and authenticating containers are essential for guarding against malicious code that could potentially be introduced by a rogue or unwitting member of the user community. After all, in some instances containerized applications may rely on third-party data sources.

Given the complexity of HPC environments, along with regulatory and general security considerations, administrative considerations with containerization tools are also compounded by the industry shortage of HPC specialists. Containers require some expertise to set up, e.g., packaging the applications and dependencies. It's also important to consider other challenges and situations where containers might not work well in HPC environments:

- **When the absolute best performance is needed from the hardware, such as in the financial industry and bare metals servers**
- **When using a heavily laden GUI application**
- **When applications are monolith and work better on bare metal**
- **When it's difficult to access a server's underlying hardware, or make a configuration portable**

Key features to look for in HPC container tools

☑ Verifiable reproducibility and security

Cryptographic signatures, an immutable container image format, and in-memory decryption are key.

☑ A simple, effective security model

A user is the same user inside a container as outside and cannot gain additional privilege on the host system by default.

☑ Integration over isolation by default

Capabilities that enable easy use of GPUs, high-speed network interconnects, and parallel file systems on a cluster or server.

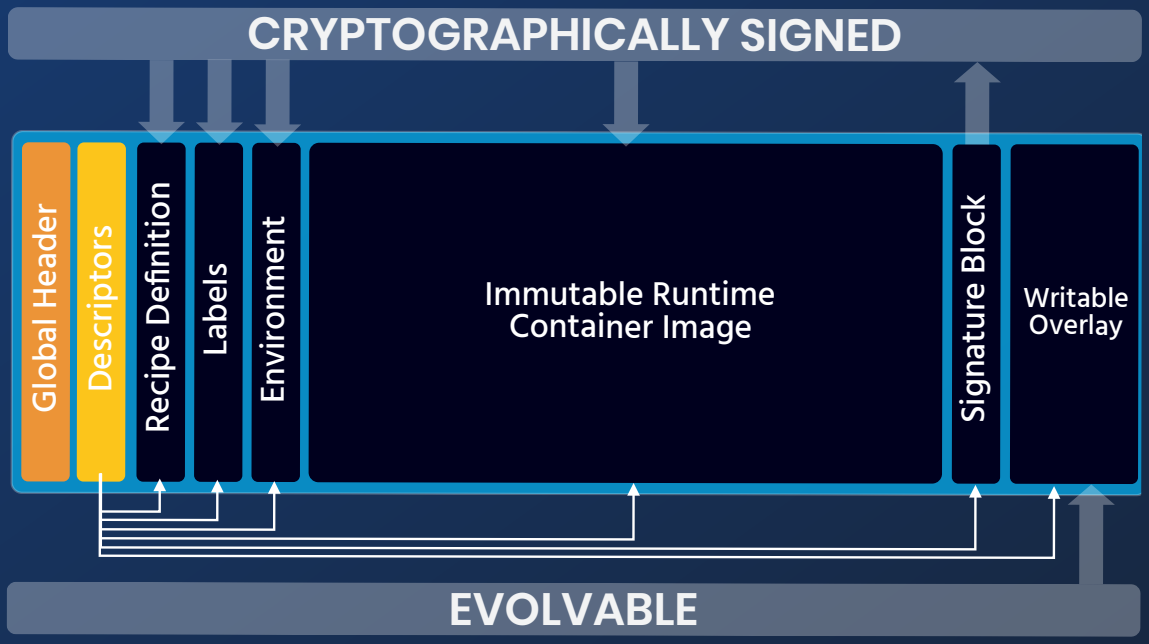
☑ Mobility & Portability

A single file container format that is easy to transport and share.

What is SIF and why does it matter in HPC?

SIF is a single-file-container image containing a file system as well as predefined and user-defined annotations, and arbitrary data packaged in an immutable format. The block structure includes metadata, headers, partitions, and signatures, and the definition file used as the recipe during the container build process. This structure was designed to help prevent data being tampered with or changed as the container is transported from a secure build environment to its compute destination.

This is different from OCI/Docker-type containers where OCI images must be unpacked on disk before running. Docker containers are stored as a collection of tarballs called layers. When building from a Docker container, users must download the layers and then assemble them in the proper order to produce a viable file system. Running from a Docker source image is not considered reproducible because the container will change if any of the layers of the image are changed. If reproducibility is important to your workflow, consider the SIF format.



SIF images are well suited for HPC environments – capable of running on thousands of compute nodes simultaneously due to their unique properties. The images can be transported more efficiently and easily since everything is contained in a single file.

Leading tools for HPC containerization

A number of open source and commercial containerization tools are suitable for HPC use cases. We've highlighted some of the most popular options on the following pages to help you in your research. Please note, this list may not include all derivatives and forks of core codebases.

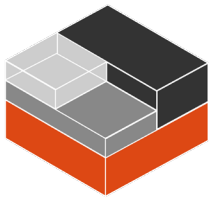
Charliecloud

Charliecloud is an alternative to the open source Docker Engine that is better suited for use in HPC environments. It offers user-defined software stacks for HPC centers. One advantage of Charliecloud is the ability to use it with unprivileged permissions through user namespaces in the Linux kernel. The approach helps minimize security risks without compromising performance and functionality. Charliecloud can also pull from Docker compatible registries. https://hpc.github.io/charliecloud/

Docker Engine

Docker enables developers to deploy applications inside containers on Linux systems. Users can run Docker containers on Windows via a Linux virtualization layer. Developers often turn to Docker when production and development environments differ because the engine does a good job joining development and operations into DevOps. A developer can create and test an app in one environment and define it via a Dockerfile. Then the operations team can recreate it in the production environment. Docker uses fewer resources and less space and runs faster than a typical virtual machine approach. Although Docker is the most widely used container service, it is not ideal for HPC applications. The main issue is that Docker images offer a way to gain root access to the systems they're running on. https://www.docker.com/

LXC



A lightweight Linux-based container engine, LXC allows users to run single applications in virtual environments. It also lets users run an entire OS inside an LXC container. LXC simplifies the process of controlling a virtual environment using userspace tools from the host operating system. This increases the portability of individual apps by allowing them to be distributed inside containers. LXC, which is lighter than Docker, acts as a virtual machine, offers more security options, doesn't use as many system resources, and lets users develop independent network interfaces. LXC's goal is to create an environment that's close to the standard Linux installation but doesn't need a separate kernel. LXC is a better alternative to traditional hypervisors and a good choice for data-intensive applications. https://linuxcontainers.org/

Podman

Podman, which is used to manage the containers on a Red Hat Enterprise Linux-based operating system is an open-source virtualization platform that lets users run containers rootless on their host machines. They can also run rootless containers that only have the same privileges as the users who launched them (the users have root privileges within the containers). Using the Podman container engine, users can develop, manage, and run containers on Linux machines under the Open Container Initiative (OCI) standards. Since many supercomputing sites are deploying containers developed using the Singularity Image Format (SIF) the Podman community is also in the early stages of SIF image support. https://podman.io/

Shifter

A container engine alternative to Docker, Shifter converts a Docker image to a format that developers can distribute and launch on HPC systems. With Shifter's user interface, a user can choose an image from the Docker Hub registry and submit jobs that run totally within the container. Shifter is made up of a utility that generally runs on the compute node that creates the app's runtime environment. An image gateway service takes images from a registry and reorganizes them in a format that HPC systems can use (typically the squash file system) and integrate with batch scheduler systems. https://github.com/NERSC/shifter

Singularity

An open-source, secure HPC alternative container framework, Singularity uses an implementation that doesn't need elevated privileges to run containers; however, it still enables direct interaction with existing Docker containers. With Singularity containers, users can run applications in a variety of Linux distributions. Users can build Singularity containers on their laptops or PCs and then run them on a single server, on local university clusters, in the cloud, on any workstation, or even the largest HPC clusters in the world. Since the container is a single file, users don't have to be concerned about how to install all the software they need on each different OS. Singularity containerization addresses several key HPC concerns, including the need for secure, fast, and simplified mobility across hybrid computing resources. Increasingly, the Singularity Image File (SIF) format is recognized and accepted as the de facto standard for HPC containers. Singularity's approach is about integration over isolation, which makes accessing GPU, high speed interconnects, or specialized hardware easy. https://sylabs.io/singularity/

Conclusion

It has always been challenging to partition large systems used by multiple users or multiple applications running side by side. Containerized deployments are becoming increasingly common for HPC environments because they provide an inexpensive yet effective way to isolate workloads. While containerization tools can simplify access to HPC resources and enable users on shared systems to do things they had been unable to do previously, not all containerization tools are created equal. Selecting the wrong tool can lead to less-than-optimal and even failed deployments. Choosing the right containerization technology is crucial when it comes to the security of the host and other users, scalability, reproducibility, and mobility.

Sylabs

Deploying Performance Intensive Workloads Easily and Securely

Contact us today

For help choosing the best container solution for your application, contact Sylabs today to chat with one of our Solution Specialists!

Phone: (503) 428-5606 | Email: info@sylabs.io