LCI Intermediate - Apptainer

Alan Chapman

Systems Analyst - Software Specialist

RTO Research Computing

Arizona State University

Introduction to Containerization

- **Overview:** Containers are lightweight, stand-alone packages that contain everything needed to run a piece of software, including the code, runtime, libraries, and environment variables. Unlike virtual machines, containers do not bundle an entire operating system—just the necessary components, making them more efficient, portable, and scalable.
- **Benefits:** The efficiency of containers comes from their lightweight nature, allowing for rapid deployment and scaling. Their portability ensures that software runs consistently across different computing environments, from a developer's laptop to an HPC cluster. Scalability is achieved by easily adding more container instances to handle increased load.

How Apptainer Addresses HPC Security Concerns

- **Non-root Execution:** Apptainer uniquely enables users to run containers without root privileges, significantly reducing the risk of privilege escalation attacks. This feature ensures that even if a container were compromised, the potential damage would be limited, as the attacker wouldn't have root access to the host system.

How Apptainer Addresses HPC Security Concerns

- **Immutable File System:** By encapsulating containers within a single, immutable file, Apptainer minimizes the risk of tampering. This design means that once a container is created, its contents cannot be altered, providing a secure and consistent runtime environment across different computing landscapes.

How Apptainer Addresses HPC Security Concerns

- **Authentication and Authorization:** Apptainer's compatibility with various security frameworks ensures that containers are accessible only to authenticated and authorized users. This layer of security is critical in preventing unauthorized access to sensitive computations and data within HPC environments.

Differences Between Apptainer and Docker

- ***Architecture:*** Docker's daemon-based architecture, while user-friendly, requires running a background service with elevated privileges, which can be a vector for security breaches. Apptainer's approach eliminates the need for a daemon, allowing containers to run as standalone executables, which aligns better with the security and operational models of HPC systems.

Differences Between Apptainer and Docker

- ***File System:*** Docker's layered filesystem is ideal for development, enabling easy modifications and updates to containers. However, this flexibility can introduce complexity and security concerns in multi-user environments. Apptainer’s single-file approach simplifies distribution and execution of containers, ensuring consistency and security, especially in environments where container immutability is preferred.

Differences Between Apptainer and Docker

- ***Use Cases:*** Docker excels in microservices architecture, development, and testing scenarios, where its rich ecosystem and tooling streamline the development lifecycle. Apptainer, however, is tailored for compute-heavy tasks requiring high performance and strict security, typical of HPC and scientific research environments, where Docker's model may not be as well-suited.

Integration of Cloud Computing with Apptainer in HPC

- ***Scalability:*** The cloud's elasticity complements Apptainer's lightweight and portable container format, allowing researchers and engineers to scale their HPC workloads up or down based on real-time demand. This dynamic scalability can lead to more efficient use of computational resources, reducing costs and improving performance.

Integration of Cloud Computing with Apptainer in HPC

- ***Accessibility:*** Cloud computing can democratize access to HPC by offering high-performance computational resources on a pay-as-you-go basis. When combined with Apptainer, it allows users to deploy consistent and secure computing environments quickly, making advanced computational tools accessible to a broader audience, including smaller research institutions and industries without substantial IT infrastructure.

Integration of Cloud Computing with Apptainer in HPC

- ***Collaboration and Sharing:*** The cloud facilitates easy sharing of Apptainer containers among collaborators across the globe. Researchers can share entire computational environments, not just data and code, ensuring reproducibility and accelerating collaborative research efforts. This ecosystem can significantly speed up scientific discovery by allowing researchers to build on each other's work seamlessly.

Integration of Cloud Computing with Apptainer in HPC

- ***Hybrid Environments:*** Integrating Apptainer with cloud services enables the creation of hybrid HPC environments where computational workloads can move between local and cloud-based resources. This flexibility allows organizations to optimize their computational workflows based on cost, performance, and resource availability, choosing the most appropriate environment for each task.

Methods of building containers

- Pulling a container image from a container repository
- Building a container image from a definition file

Pulling a container from a repository

```
[acchapm1@c001 ~]$ apptainer pull alpine.sif docker://alpine
INFO:    Converting OCI blobs to SIF format
INFO:    Starting build...
Getting image source signatures
Copying blob 4abcf2066143 done   |
Copying config bc4e4f7999 done   |
Writing manifest to image destination
2024/04/12 13:58:47  info unpack layer: sha256:4abcf20661432fb2d719aaf90656f55c287f8ca915dc1c92ec14ff61e67fbaf8
INFO:    Creating SIF file...
[acchapm1@c001 ~]$
```

Interacting with a container

```
[acchapm1@c001 ~]$ apptainer shell alpine.sif
Apptainer> cat /etc/os-release
NAME="Alpine Linux"
ID=alpine
VERSION_ID=3.19.1
PRETTY_NAME="Alpine Linux v3.19"
HOME_URL="https://alpinelinux.org/"
BUG_REPORT_URL="https://gitlab.alpinelinux.org/alpine/aports/-/issues"
Apptainer> exit
[acchapm1@c001 ~]$
```

Building a Container with an Apptainer Definition File

1. ***Prepare the Definition File:*** The first step is to create an Apptainer definition file, which specifies the environment and instructions for building the container. Here's a simplified example:

tensorflow.def

```
Bootstrap: docker
From: tensorflow/tensorflow:latest-gpu

%post
    # Update and install necessary packages
    apt-get update && apt-get install -y git wget

%environment
    export LOG_DIR=/var/logs/tensorflow

%runscript
    # Default command to run when the container is executed
    exec tensorflow

%help
    This container includes TensorFlow with GPU support.
```

This definition file starts from a base TensorFlow image with GPU support from Docker Hub, installs additional packages (git and wget), and sets up an environment variable and a default run command.

2. Build the Container

With the definition file ready, the next step is to build the container. This process requires root access or administrative privileges, we suggest people build a container on their own system and transfer it or put in a request to have an admin build the container for them.

```
sudo apptainer build tensorflow.sif tensorflow.def
```

This command tells Apptainer to build a container named `tensorflow.sif` based on the instructions in `tensorflow.def`.

Using the TensorFlow Container on an HPC System

1. Transfer the Container

Once built, transfer the `.sif` file to the HPC system using `scp` or any other secure file transfer method.

2. Load Necessary Modules

Before running the container, load any necessary modules on the HPC system, especially those related to GPU drivers or MPI, if your TensorFlow application requires them.

```
module load cuda/10.1
```

3. Execute the Container

Run the TensorFlow container using Apptainer, specifying any required arguments or commands. For example, to start a Python script that uses TensorFlow:

```
apptainer exec --nv tensorflow.sif python my_tensorflow_script.py
```

The `--nv` flag enables NVIDIA GPU support within the container, crucial for TensorFlow operations that are GPU-accelerated.