An Evaluation of Podman

Chris Hollowell <hollowec@bnl.gov>
Costin Caramarcu <caramarc@bnl.gov>

HEPiX Online - Fall 2020
Oct 15, 2020

BROOKHAVEN NATIONAL LABORATORY | Scientific Data and Computing Center

U.S. DEPARTMENT OF ENERGY

Linux Container Ecosystem

- Increasing number of container engines available for Linux
  - Docker
  - Singularity
  - Shifter
  - Charliecloud
  - LXC
  - CRI-O
  - Podman
  - ...
- Developed by various communities/organizations/companies
  - Focus on different container use cases and features
- All utilize Linux kernel namespaces for containment
- Push towards the use of Open Container Intiative (OCI) container image format
  - https://opencontainers.org/
  - Allows sharing of container images between engines
    - Several make direct use of OCI's runc underneath to spawn containers

What is Podman?

podman

- Podman is the default/stock container engine shipped with RHEL/CentOS 8
  - Starting in RHEL8, Red Hat no longer ships or supports Docker
    - CRI-O used as the default engine for Openshift
  - But Docker-CE can still be installed from the upstream Docker repo, and will function on RHEL/CentOS 8

- Starting with RHEL 7.8 (tech preview in 7.6), Podman was also made available and fully supported in the RHEL7 “Extras” channel
  - Now also available in the Scientific Linux/CentOS 7 “Extras” repos
    - After enabling the repo (enabled by default in CentOS), one can install by simply running:

      ```
      # yum install podman
      ```

    - Pulls in “fuse-overlayfs”, “runc” and “slirp4netns” packages from Extras

Podman Key Features

- Command Line Interface is nearly identical to Docker's
  - So much so that it's possible setup a shell alias:

    ```
    $ alias docker=podman
    ```
  - Convenient for users already familiar with Docker's CLI

- Supports OCI containers

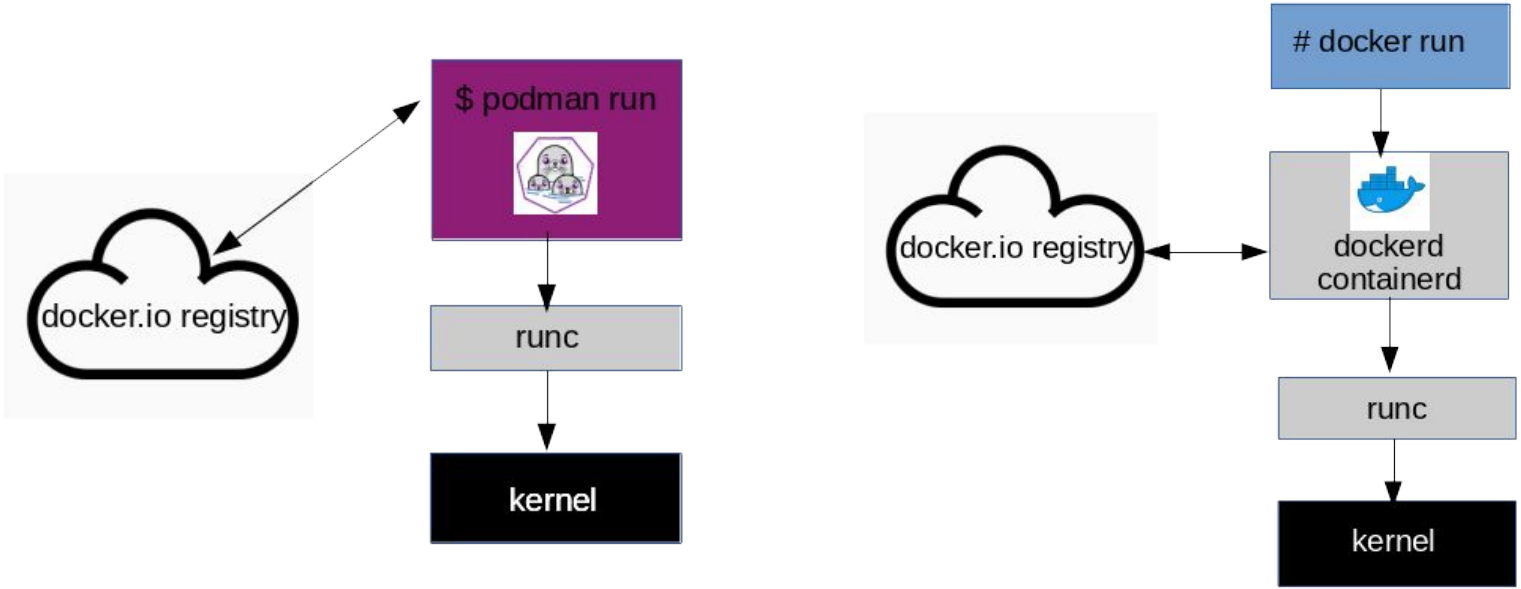
- Unlike Docker, and similar to Singularity, Podman implements daemonless container execution
  - Implemented utilizing OCI runc

- Podman supports "rootless" container execution/builds utilizing user namespaces
  - Rootless user namespace container execution has been supported by Singularity since the 2.2 release (2016)
  - The Docker daemon can utilize user namespaces via the "--userns-remap" option, and it's possible to run the Docker daemon as a non-root user in the 19.03 release

Why Did We Evaluate Podman?

- Singularity has been a very useful tool for our community. Why did we evaluate another container engine?
  - SDCC recently began supporting computing for the National Synchrotron Light Source II (NSLS-II) at BNL
    - Many of these users were accustomed to using Docker
      - Some concerns/issues adapting to using Singularity
    - We did not want to enable Docker on our compute hosts, due to security concerns
  - Podman is the default in RHEL/CentOS 8, and also shipping in SL/CentOS 7
    - Will likely lead to widespread adoption

Podman vs Docker Container Execution



OCI container execution with Podman (left) and Docker (right)

Example Execution

- As mentioned, effectively the same as the Docker CLI

- Running a container:

```
$ podman run -it --network=host centos:6 /bin/sh
Trying to pull docker.io/library/centos:6…
Getting image source signatures
Copying blob ff50d722b382 done
Copying config d0957ffdf8 done
Writing manifest to image destination
Storing signatures
sh-4.1# cat /etc/redhat-release
CentOS release 6.10 (Final)
```

- Building a container:

```
$ cat Dockerfile.test
FROM centos:7
COPY TESTFILE /
CMD cat /TESTFILE
```

```
$ podman build -t testimage -f ./Dockerfile.test
STEP 1: FROM centos:7
STEP 2: COPY TESTFILE /
--> Using cache
dc0a832ab170867e62606f9ef900b36a858f2e1018bb380da684b0a2
a9be5abd
STEP 3: CMD cat /TESTFILE
STEP 4: COMMIT testimage
a80025ba50f5f03dcde3d7ce1c7a662a3b9bf89b4385cb8e40c5ddd5
cb9dad76
```

Podman Rootless Container Execution on SL7

- Requires relatively new kernel and “shadow-utils” packages (available in recent SL/CentOS 7 releases)
  - shadow-utils 4.1.5+
    - Includes newuidmap/newgidmap binaries
  - kernel 3.10.0-1127.el7+

- User namespaces must be enabled

```
# echo 10000 > /proc/sys/user/max_user_namespaces
```

- Requires all users using podman to have namespace UID/GID mappings defined in /etc/subuid and /etc/subgid

```
# cat /etc/subuid
user1:600000:32000
user2:603201:32000
# cat /etc/subgid
user1:600000:32000
user2:603201:32000
```

  - Unfortunately, these files must be local: can’t be managed via LDAP/IPA

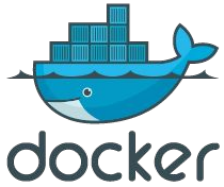
Podman Rootless Container Execution on SL7 (Cont.)

- *Lack of support for subuid/subgid in LDAP makes it nearly impossible to use this software in production at sites with many centrally managed users/hosts*
  - Recent discussions between shadow-utils/glibc developers about adding subuid/sugid to nsswitch
  - Even if LDAP were supported, having to manage these files adds additional complexity
- Users cannot run containers without subuid/subgid settings defined

```
$ podman run -it --network=host centos:8 /bin/sh
ERRO[0000] cannot find mappings for user testuser: No subuid ranges found for user
"testuser" in /etc/subuid
Trying to pull docker.io/library/centos:8…
Getting image source signatures
Copying blob 3c72a8ed6814 done
Copying config 0d120b6cca done
Writing manifest to image destination
Storing signatures
Error processing tar file(exit status 1): there might not be enough IDs available
in the namespace (requested 0:22 for /run/utmp): lchown /run/utmp: invalid argument
```

  - Documented “--storage-opt ignore_chown_errors” option added to Podman 1.5.0 to address this
    - Unfortunately, this option did not resolve the issue in our tests

Comparison with Docker and Singularity



- Podman, Docker and Singularity all support OCI container format images
- For regular users, Podman caches containers in the user's home directory: ~/.local/share/containers
  - Similar to Singularity's ~/.singularity directory
  - Docker daemon uses /var/lib/docker for caching
- Unlike Docker, Podman also supports instantiating pods using k8s YAML files
  - Can also dump running pod/container configurations to k8s YAML files
- Singularity does not require one to configure /etc/subuid and /etc/subgid UID/GID mappings to run rootless with user namespaces
  - Only necessary if "--fakeroot" is utilized
    - For unprivileged container builds, for example

Comparison with Docker and Singularity (Cont.)

- Many HEP/NP experiments utilize CVMFS to distributed unpacked container images for computing:
  - CVMFS unpacked image distribution is highly efficient (both from a network and disk utilization perspective) mechanism for images access, where only utilized files are retrieved
    - *Most files in a container are never accessed*
- Singularity natively supports running unpacked images, i.e. out of /CVMFS

  `singularity shell /CVMFS/PATH`
- While Docker doesn't natively support unpacked filesystem images, the "CernVM-FS Graph Driver" Docker plugin provides this functionality
  - Implements Docker "Thin Images" which link to unpacked areas in CVMFS
  - https://cvmfs.readthedocs.io/en/stable/cpt-graphdriver.html
- Podman does not support running unpacked images natively, and also does not support plugins
  - Graph driver cannot be used
  - However, CVMFS documentation indicates podman integration is being developed (HSF GSoC project - podman/CVMFS "additional image stores")

Using Podman with HEPscore

- Able to successfully run the HEPscore20 benchmark with Podman
  - `subprocess.Popen()` call with `shell=False` used in the hep-score code
    - Therefore setting ‘docker’ alias was not sufficient
    - Had to create a docker -> podman symlink in /usr/bin
      - Will likely add support for running podman directly in hep-score in the future
        - After some additional testing
    - Also ran into an issue with the open file descriptor limit - needed to increase
      - With podman, limit appears to apply to the entire container, not per process

```
$ hep-score -dV /tmp
...
HEPscore20 Benchmark
Config Hash:          bd2c824cffa87945426b4e5aaf105afc185db01a0a5aa327f6183d6b1094272c
System:               Linux spool0901.sdcc.bnl.gov 3.10.0-1127.el7.x86_64 #1 SMP Wed Apr 1 12:25:50 CDT 2020 x86_64
Container Execution:          docker
Registry:             docker://gitlab-registry.cern.ch/hep-benchmarks/hep-workloads
Output:               /tmp/HEPscore_09Oct2020_011000
Date:                 Fri Oct  9 01:10:00 2020

2020-10-09 01:10:01,063 - INFO - _run_benchmark() - Executing 3 runs of atlas-gen-bmk
2020-10-09 01:10:01,063 - INFO - _run_benchmark() - Starting run0
2020-10-09 01:10:01,063 - DEBUG - _run_benchmark() - Running  ['docker', 'run', '--rm', '--network=host', '-v', '
/tmp/HEPscore_09Oct2020_011000/atlas-gen/run0:/results',
'gitlab-registry.cern.ch/hep-benchmarks/hep-workloads/atlas-gen-bmk:v1.2', '-t', '1', '-e', '200']
...
2020-10-09 05:24:23,963 - INFO - gen_score() - Final result: 889.8464
```

Conclusions

- Podman provides a RHEL/CentOS/SL distribution-supported/default OCI container solution that is nearly identical to Docker from a CLI perspective

- Singularity will likely remain the de facto standard container engine for portability of compute, particularly in the HEP/NP domain
  - Singularity purpose-built for portability-of-compute container use case
  - Current lack of support for subuid/subgid in LDAP makes the adoption of Podman difficult at large multi-user compute sites
  - Podman does not currently support running unpacked images out of CVMFS
    - This is problematic given the efficiency of CVMFS unpacked image distribution, and widespread adoption of this mechanism in HEP/NP
    - However, support for additional Podman image stores is in development

- Podman's high level of compatibility with the Docker CLI, while supporting enhanced security features, pods, and a simplified execution model make it an appealing solution for non-compute container use cases, e.g. web services

Thank you! Questions?