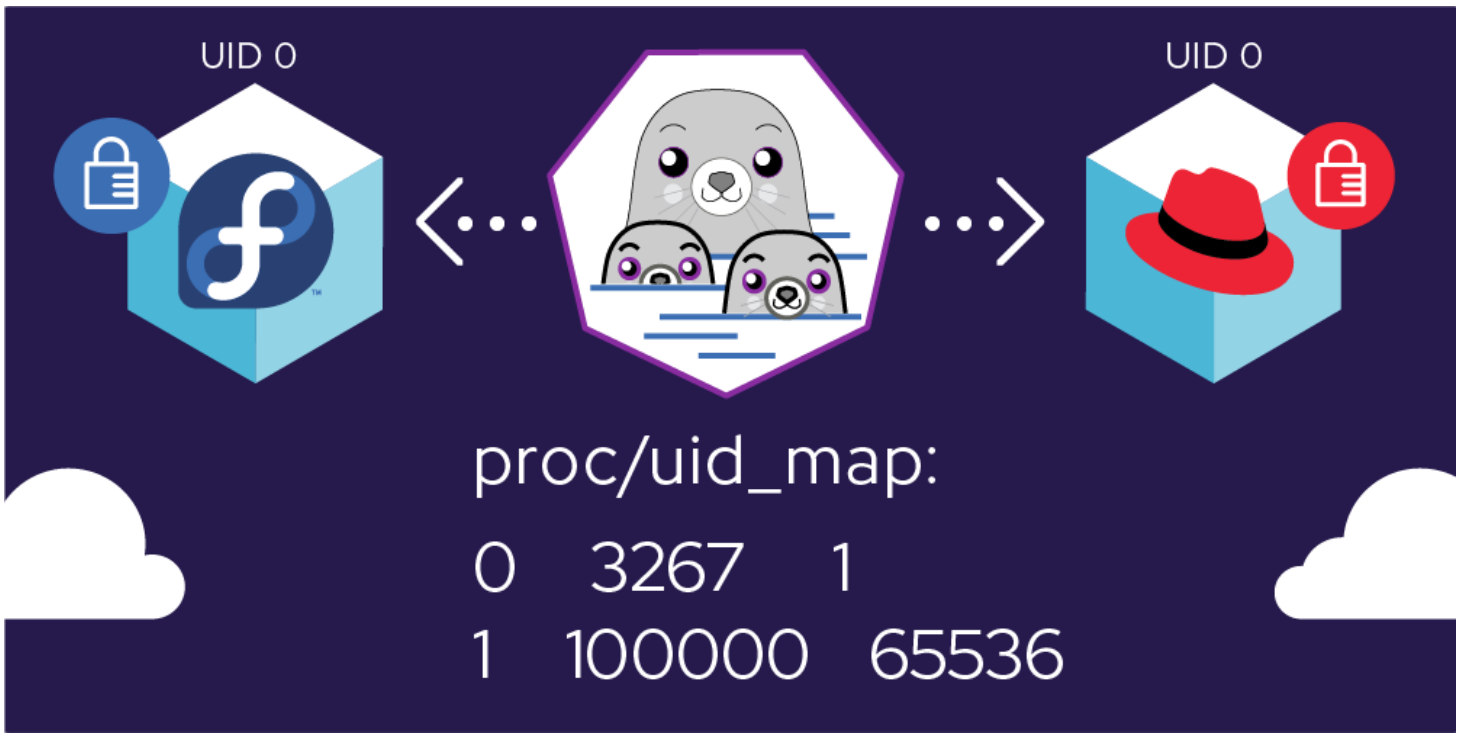
Rootless containers with Podman: The basics

By Prakhar Sethi
September 25, 2020



As a developer, you have probably heard a lot about containers. A *container* is a unit of software that provides a packaging mechanism that abstracts the code and all of its dependencies to make application builds fast and reliable. An easy way to experiment with containers is with the Pod Manager tool (Podman), which is a daemonless, open source, Linux-native tool that provides a command-line interface (CLI) similar to the docker container engine.

In this article, I will explain the benefits of using containers and Podman, introduce rootless containers and why they are important, and then show you how to use rootless containers with Podman with an example. Before we dive into the implementation, let's review the basics.

Why containers?

Using containers isolates your applications from the various computing environments in which they run. They have become increasingly popular because they help developers focus on the application logic and its dependencies, which they bind in a single unit. Operations teams also like containers because they can focus on managing the application, including deployment, without bothering with details such as software versions and configuration.

Containers virtualize at the operating system (OS) level. This makes them lightweight, unlike virtual machines, which virtualize at the hardware level. In a nutshell, here are the advantages of using containers:

- Low hardware footprint
- Environment isolation
- Quick deployment
- Multiple environment deployments
- Reusability

Why Podman?

Using Podman makes it easy to find, run, build, share, and deploy applications using Open Container Initiative (OCI)-compatible containers and container images. Podman's advantages are as follows:

- It is *daemonless*; it does not require a daemon, unlike docker.
- It lets you control the layers of the container; sometimes, you want a single layer, and sometimes you need 12 layers.
- It uses the fork/exec model for containers instead of the client/server model.
- It lets you run containers as a non-root user, so you never have to give a user root permission on the host. This obviously differs from the client/server model, where you must open a socket to a privileged daemon running as root to launch a container.

Why rootless containers?

Rootless containers are containers that can be created, run, and managed by users without admin rights. Rootless containers have several advantages:

- They add a new security layer; even if the container engine, runtime, or orchestrator is compromised, the attacker won't gain root privileges on the host.
- They allow multiple unprivileged users to run containers on the same machine (this is especially advantageous in high-performance computing environments).
- They allow for isolation inside of nested containers.

To better understand these advantages, consider traditional resource management and scheduling systems. This type of system should be run by unprivileged users. From a security perspective, fewer privileges are better. With rootless containers, you can run a containerized process as any other process without needing to escalate any user's privileges. There is no daemon; Podman creates a child process.

Example: Using rootless containers

Let's get started using rootless containers with Podman. We'll start with the basic setup and configuration.

System requirements

We will need Red Hat Enterprise Linux (RHEL) 7.7 or greater for this implementation. Assuming you have that, we can begin configuring the example.

Configuration

First, install `slirp4netns` and Podman on your machine by entering the following command:

```
$ yum install slirp4netns podman -y
```

We will use `slirp4netns` to connect a network namespace to the internet in a completely rootless (or unprivileged) way.

When the installation is done, increase the number of user namespaces. Use the following commands :

```
$ echo "user.max_user_namespaces=28633" > /etc/sysctl.d/userns.conf
$ sysctl -p /etc/sysctl.d/userns.conf
```

Next, create a new user account and name it. In this case, my user account is named Red Hat:

```
$ useradd -c "Red Hat" redhat
```

Use the following command to set the password for the new account (note that you must insert your own password):

```
$ passwd redhat
```

This user is now automatically configured to be able to use a rootless instance of Podman.

Connect to the user

Now, try running a Podman command as the user you've just created. Do not use `su -` because that command doesn't set the correct environment variables. Instead, you can use any other command to connect to that user. Here's an example:

```
$ ssh redhat@localhost
```

Pull a Red Hat Enterprise Linux image

After logging in, try pulling a RHEL image using the `podman` command (note that `ubi` stands for Universal Base Image):

```
$ podman pull ubi7/ubi
```

If you want more information about the image, run this command:

```
$ podman run ubi7/ubi cat /etc/os-release
```

To check the images that resulted from the above command, along with any other images on your system, run the command:

```
$ podman images
```

It is also possible for a rootless user to create a container from these images, but I'll save that for another article.

Check the rootless configuration

Finally, verify whether your rootless configuration is properly set up. Run the following command to show how the UIDs are assigned to the user namespace:

```
$ podman unshare cat /proc/self/uid_map
```

Conclusion

This article demonstrated how to set up rootless containers with Podman. Here are some tips for working with rootless containers:

- As a non-root container user, container images are stored under your home directory (for instance, `$HOME/.local/share/containers/storage`), instead of `/var/lib/containers`. This directory scheme ensures that you have enough storage for your home directory.
- Users running rootless containers are given special permission to run on the host system using a range of user and group IDs. Otherwise, they have no root

privileges to the operating system on the host.

- A container running as root in a rootless account can turn on privileged features within its own namespace. But that doesn't provide any special privileges to access protected features on the host (beyond having extra UIDs and GIDs).

Learn more about setting up rootless containers with Podman here.