Transitioning from Docker to Podman

By Cedric Clyburn
November 19, 2020

Podman is an excellent alternative to Docker containers when you need increased security, unique identifier (UID) separation using namespaces, and integration with `systemd`. In this article, I use real-world examples to show you how to install Podman, use its basic commands, and transition from the Docker command-line interface (CLI) to Podman. You'll also see how to run an existing image with Podman and how to set up port forwarding.

About Podman

Podman is a daemonless, open source, Linux-native tool designed to develop, manage, and run Open Container Initiative (OCI) containers and pods. It has a similar directory structure to Buildah, Skopeo, and CRI-O. Podman doesn't require an active container engine for its commands to work.

Install Podman

If you are running Red Hat Enterprise Linux 8 (RHEL 8), enter the command:

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

If you are a Fedora user, replace `yum` with `dnf`:

If Linux is not available, you can use Podman online with Katacoda.

Transition to the Podman CLI

One of Podman's greatest advantages is its complete CLI compatibility with Docker. In fact, when building Podman, Docker users can adapt without any significant changes. For example, you can use the `alias` command to create a `docker` alias for Podman:

```
$ alias docker=podman
```

You can run familiar commands such as `pull`, `push`, `build`, `commit`, `tag`, and more with Podman.

You can also use Podman to run secure, rootless containers. By joining a user namespace and setting root access inside, you can enable Podman to mount certain filesystems and set up the container with no escalation of privileges.

Run an existing image using Podman

Fortunately, images created by Docker and Podman are compatible with the OCI standard. This means that Podman can push and pull from container registries such as the Docker Hub and Quay.io.

For example, let's test the Funbox container, which combines terminal commands and ASCII art. To start, clone the repository in a local directory with the following `git` commands:

```
$ git clone https://github.com/wernight/docker-funbox.git
```

Once you download the necessary files, you can pull the base image and additional requirements to build and run a container:

```
$ docker run --rm -it wernight/funbox
```

In this case, we've used the following tags with the `docker run` command:

- The `--rm` tag removes the container after it exits.
- The `-it` tag connects the container to the terminal so that you can interact with it.

We now have a container active and running (mine is running on top of Debian Jessie). Let's add an argument to view the Funbox in action:

```
$ docker run --rm -it wernight/funbox nyancat
```

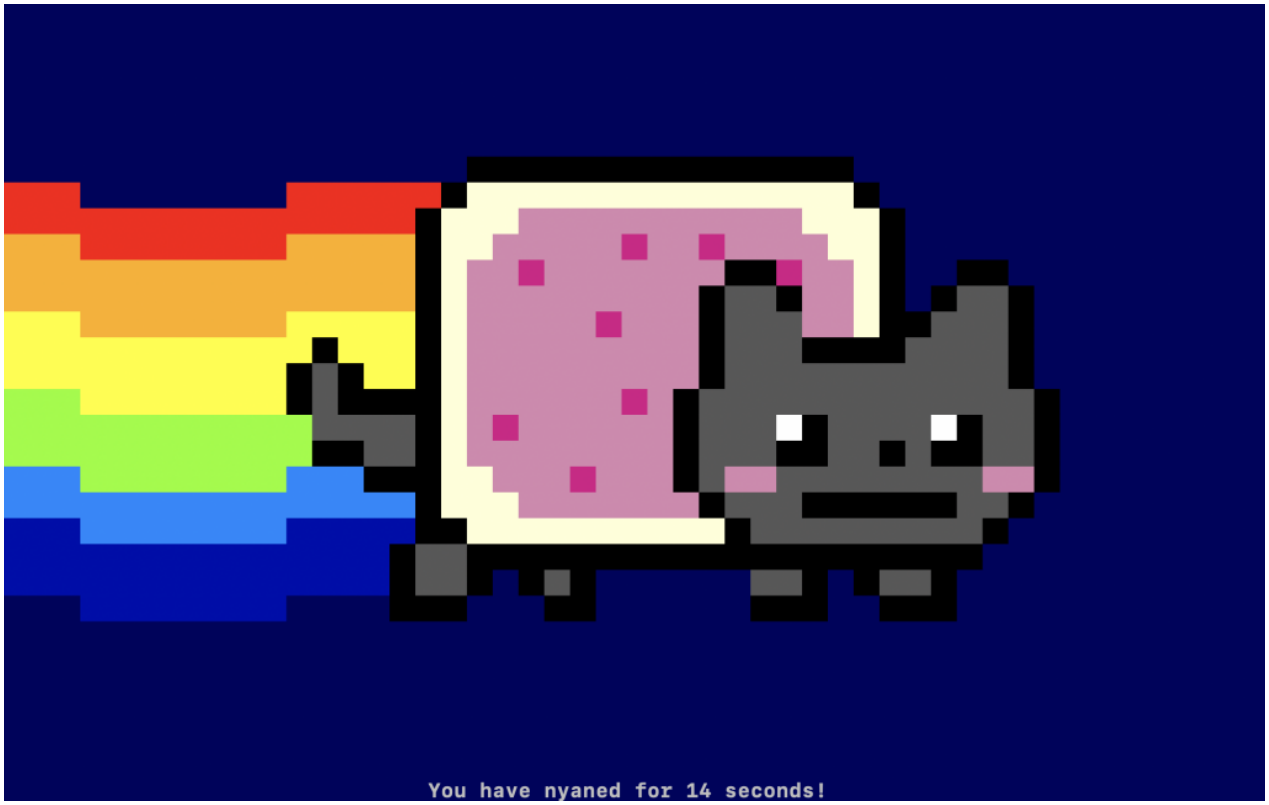

Figure 1: Do you see a Nyan Cat?

If you see a Nyan Cat displayed on your console screen, you are all set to deploy and interact with a container using Podman.

Port forwarding with Podman

Dozens of containers are available to download and use with Podman. For this example, let's set up a simple Apache HTTP Server 2.4 in a CentOS container. To begin, pull the base image you want to use from the Docker hub:

```
$ podman pull centos:latest
```

Once you've created the base image, use `podman images` to check whether the container is ready to use. You should see output similar to what's shown in Figure 2:

```
[root@ip-172-31-44-90 ~]# podman images
REPOSITORY                  TAG     IMAGE ID      CREATED       SIZE
docker.io/library/centos    latest  0d120b6ccaa8  3 months ago  222 MB
docker.io/wernight/funbox   latest  538c146646c3  2 years ago   1.14 GB
[root@ip-172-31-44-90 ~]#
```

Figure 2: The container is ready to use.

To run the new container with your base image, use the `podman run` command with specific tags (such as `-it`) to attach it to the CLI. Use `--name` to define a custom name. Finally, define the base image where the container should run:

```
$ podman run -it --name redhat-website centos:latest
```

When the container is running, automatically set root access inside to run all commands.

Create the Apache HTTP server

To create an Apache web server, we can install the httpd program with the default package installer. For CentOS, it's `yum`:

```
$ yum install -y httpd
```

Figure 3 shows the console output for this command.

```
[root@ip-172-31-44-90 ~]# podman run -it --name redhat-website centos:
latest
[root@c078f8d336fb /]# yum install -y httpd
```

Figure 3: Console output for the yum install command.

You can now serve content from your container to your server's public IP address.

Create a web page

Next, we will add text to an `index.html` file in the container's `var/www/html` directory. Feel free to customize your message, or add the default below:

```
$ echo 'Hello from Red Hat!' > /var/www/html/index.html
```

When you are finished, type `exit` to shut down or power off the container.

Use `podman commit` to commit your changes. Use tags to define a name and a custom version for your customized container:

```
$ podman commit redhat-website redhat-website:v1
```

Finally, launch the container, then forward all requests made to your server's public IP address on port 8080 to port 80 on the container. Use the Podman tag `-p` to port forward, and specify the container that you want to run. Ensure that `httpd` is running as a foreground process:

```
$ podman run -p 8080:80 redhat-website:v1 /usr/sbin/httpd -D FOREGROUND
```

To view the web page from the host device, run a `curl` command while specifying port 8080. You should see the screen shown in Figure 4.

```
[root@ip-172-31-44-90 ~]# curl localhost:8080
Hello from Red Hat!
[root@ip-172-31-44-90 ~]#
```

Figure 4: A successful example of port forwarding using Podman.

How to stop and remove a container

You can use the `podman stop` command to stop a specified container:

```
$ podman stop redhat-website
```

Use `podman rm` to remove the container:

```
$ podman rm redhat-website
```

Conclusion

Every command that I demonstrated in this article is compatible with the Docker CLI. Podman has great integration features through `systemd`. You can use it to run rootless containers, and it is a powerful container image for running OCI containers on RHEL 8.

You can continue to experiment with Podman by setting up this Katacoda scenario, which offers an interactive environment directly in your browser.

If you need container orchestration, you can use Podman with Kubernetes or Red Hat OpenShift. To get started with these platforms, see kubernetesbyexample.com and learn.openshift.com.

For more interactive demonstrations, watch the video that accompanies this article.

Resources

If you want to keep learning about Podman, start with these articles on Red Hat Developer:

- Rootless containers with Podman: The basics (Prakhar Sethi, 2020)
- Podman and Buildah for Docker users (William Henry, 2019)
- Podman basics cheat sheet (Doug Tidwell, 2019)
- Intro to Podman (Red Hat Enterprise Linux 7.6 Beta) (Alessandro Arrichiello, 2018)

Related