**Managing pods and containers in a local container runtime**



People associate running pods with [Kubernetes](https://kubernetes.io/). And when they run containers in their development runtimes, they do not even think about the role pods could play—even in a localized runtime.  Most people coming from the Docker world of running single containers do not envision the concept of running pods. There are several good reasons to consider using pods locally, other than using pods to naturally group your containers.

For example, suppose you have multiple containers that require the use of a MariaDB container.  But you would prefer to not bind that database to a routable network; either in your bridge or further.  Using a pod, you could bind to the localhost address of the pod and all containers in that pod will be able to connect to it because of the shared network name space.

Podman Pods: what you need to know

The [Pod](https://kubernetes.io/docs/concepts/workloads/pods/pod/) concept was introduced by [Kubernetes](https://developers.redhat.com/topics/kubernetes/).  Podman pods are similar to the Kubernetes definition.

[Podman architecture: containers in a pod](https://developers.redhat.com/blog/wp-content/uploads/2019/01/podman-pod-architecture.png)

Every Podman pod includes an “infra” container.   This container does nothing, but go to sleep. Its purpose is to hold the namespaces associated with the pod and allow podman to connect other containers to the pod.  This allows you to start and stop containers within the POD and the pod will stay running, where as if the primary container controlled the pod, this would not be possible. The default infra container is based on the k8s.gcr.io/pause image,  Unless you explicitly say otherwise, all pods will have container based on the default image.

Most of the attributes that make up the Pod are actually assigned to the “infra” container.  Port bindings, cgroup-parent values, and kernel namespaces are all assigned to the “infra” container. This is critical to understand, because once the pod is created these attributes are assigned to the “infra” container and cannot be changed. For example, if you create a pod and then later decide you want to add a container that binds new ports, Podman will not be able to do this.  You would need to recreate the pod with the additional port bindings before adding the new container.

In the above diagram, notice the box above each container, conmon, this is the container monitor.  It is a small C Program that’s job is to watch the primary process of the container, and if the container dies, save the exit code.  It also holds open the tty of the container, so that it can be attached to later. This is what allows podman to run in detached mode (backgrounded), so podman can exit but conmon continues to run.  Each container has its own instance of conmon.

The CLI: podman pod

We expose most of the interaction with pods through the podman pod commands.  Among other actions, you can use podman pod to create, delete, query, and inspect pods.  You can see all the pod related commands by running podman pod without any arguments.

$ sudo podman pod

NAME:

  podman pod - Manage container pods.

Pods are a group of one or more containers sharing the same network, pid and ipc namespaces.

USAGE:

  podman pod command [command options] [arguments...]

COMMANDS:

    create        Create a new empty pod

    exists        Check if a pod exists in local storage

    inspect       displays a pod configuration

    kill          Send the specified signal or SIGKILL to containers in pod

    pause         Pause one or more pods

    ps, ls, list  List pods

    restart       Restart one or more pods

    rm            Remove one or more pods

    start         Start one or more pods

    stats         Display percentage of CPU, memory, network I/O, block I/O and PIDs for containers in one or more pods

    stop          Stop one or more pods

    top           Display the running processes of containers in a pod

    unpause       Unpause one or more pods

OPTIONS:

  --help, -h  show help

Create a pod

The traditional way to create a pod with Podman is using the podman pod create command.

$ sudo podman pod create --help

NAME:

  podman pod create - Create a new empty pod

USAGE:

  podman pod create [command options] [arguments...]

DESCRIPTION:

  Creates a new empty pod. The pod ID is then printed to stdout. You can then start it at any time with the podman pod start <pod\_id> command. The pod will be created with the initial state 'created'.

OPTIONS:

  --cgroup-parent value      Set parent cgroup for the pod

  --infra                    Create an infra container associated with the pod to share namespaces with

  --infra-command value      The command to run on the infra container when the pod is started (default: "/pause")

  --infra-image value        The image of the infra container to associate with the pod (default: "k8s.gcr.io/pause:3.1")

  --label value, -l value    Set metadata on pod (default [])

  --label-file value         Read in a line delimited file of labels (default [])

  --name value, -n value     Assign a name to the pod

  --pod-id-file value        Write the pod ID to the file

  --publish value, -p value  Publish a container's port, or a range of ports, to the host (default [])

  --share value              A comma delimited list of kernel namespaces the pod will share (default: "cgroup,ipc,net,uts")

In its most basic context, you can simply issue podman pod create and Podman will create a pod without extra attributes.  A random name will also be assigned to the pod.

$ sudo podman pod create

9e0a57248aedc453e7b466d73ef769c99e35d265d97f6fa287442083246f3762

We can list the pods using the podman pod list command:

$ sudo podman pod list

POD ID         NAME             STATUS    CREATED     # OF CONTAINERS   INFRA ID

9e0a57248aed   youthful\_jones  Running 5 seconds ago   1      6074ffd22b93

Note that the container has a single container in it.  The container is the “infra” command. We can further observe this using the podman ps command by passing the command line switch \*–pod\*.

$ sudo podman ps -a --pod

CONTAINER ID  IMAGE        COMMAND  CREATED      STATUS    PORTS  NAMES             POD

6074ffd22b93  k8s.gcr.io/pause:3.1    3 minutes ago  Up 3 minutes ago         9e0a57248aed-infra  9e0a57248aed

Here we can see that the pod ID from podman ps matches the pod id in podman pod list.  And the container image is the same as the default “infra” container image.

Add a container to a pod

You can add a container to a pod using the \*–pod\* option in the podman create and podman run commands.  For example, here we add a container running \*\*top\*\* to the newly created \*youthful\_jones\* pod. Notice the use of \*–pod\*.

$ sudo podman run -dt --pod youthful\_jones docker.io/library/alpine:latest top

0f62e6dcdfdbf3921a7d73353582fa56a545502c89f0dfcb8736ce7be61c9271

And now two containers exist in our pod.

$ sudo podman pod ps

POD ID         NAME            STATUS  CREATED         # OF CONTAINERS   INFRA ID

9e0a57248aed   youthful\_jones  Running 7 minutes ago   2     6074ffd22b93

Looking at the list of containers, we also see each container and their respective pod assignment.

$ sudo podman ps -a --pod

CONTAINER ID  IMAGE                   COMMAND  CREATED       STATUS             PORTS  NAMES              POD

0f62e6dcdfdb  docker.io/library/alpine:latest  top      14 seconds ago  Up 14 seconds ago         awesome\_archimedes  9e0a57248aed

6074ffd22b93  k8s.gcr.io/pause:3.1                      7 minutes ago   Up 7 minutes ago          9e0a57248aed-infra  9e0a57248aed

Shortcut to create pods

We recently added the ability to create pods via the podman run and podman create commands. One upside to creating a pod with this approach is that the normal port bindings declared for the container will be assigned automatically to the “infra” container. However, if you need to specify more granular options for pod creation like kernel namespaces or different “infra” container image usage, you still need to create the pod manually as was first described. Nevertheless, for relatively basic pod creations, the shortcut is handy. As this feature was recent added, it isn’t available in the version of Podman included with Red Hat Enterprise Linux 7.6 and 8 Beta.

To create a new pod with your new container, you simply pass \*–pod\*: new:<name>.  The use of \*\*new:\*\* indicates to Podman that you want to create a new pod rather than attempt to assign the container to an existing pod.

To create a nginx container within a pod and expose port 80 from the container to port 32597 on the host, you would:

$ sudo podman run -dt --pod new:nginx -p 32597:80 quay.io/libpod/alpine\_nginx:latest

ac8839fc7dead8e391e7983ad8d0c27ce311d190b0a8eb72dcde535de272d537

$ curl http://localhost:32597

podman rulez

And here is what it looks like when listing containers:

$ sudo podman ps -ap

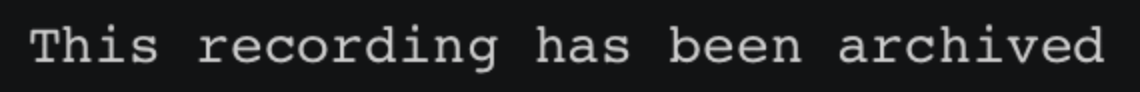
CONTAINER ID  IMAGE                      COMMAND               CREATED       STATUS            PORTS                  NAMES               POD

ac8839fc7dea  quay.io/libpod/alpine\_nginx:latest  nginx -g daemon o... 4 minutes ago Up 4 minutes ago                         happy\_cray          3e4cad88f8c2

c2f7c5651275  k8s.gcr.io/pause:3.1                                      4 minutes ago Up 4 minutes ago  0.0.0.0:32597->80/tcp  3e4cad88f8c2-infra 3e4cad88f8c2

MariaDB example

The following asciinema demo shows how to create a pod via the shortcut method.  The container being run is a MariaDB container image and I bind only to its 127.0.0.1 address.  This means only containers in the same pod will able to access it. I then run an alpine container, install the MariaDB-client package, connect to the database itself, and show defined databases.

[](https://asciinema.org/a/Xc818xXZ7TAlP9yvHU88IPVBK)

Pods and container management

In Podman, the status of the pod and its containers can be exclusive to each other meaning that containers within pods can be restarted, stopped, and started without impacting the status of the pod.  Suppose we have a pod called demodb and it contains two containers (and an “infra” container) running a MariaDB and a nginx session.

$ sudo podman pod ps

POD ID         NAME   STATUS    CREATED             # OF CONTAINERS   INFRA ID

fa7924a5196c   demodb Running   About a minute ago  3     3005ed8491d0

$ sudo podman ps -p

CONTAINER ID  IMAGE                       COMMAND              CREATED       STATUS            PORTS  NAMES            POD

02e37a3b9873  quay.io/libpod/alpine\_nginx:latest   nginx -g daemon o... 4 minutes ago Up 4 minutes ago         optimistic\_edison  fa7924a5196c

2597454063f8  quay.io/baude/mariadbpoddemo:latest  docker-entrypoint... 4 minutes ago Up 4 minutes ago         eloquent\_golick    fa7924a5196c

If we wanted to stop and start the nginx container, the status of the MariaDB container and the pod itself will remain unchanged.

$ sudo podman stop optimistic\_edison

02e37a3b987300e9124b61820119ae425c5e496b907800ecaf1194a3f50e5dcc

With the nginx container stopped, we can still observe the demopod is running and the MariaDB container remains unchanged.

$ sudo podman pod ps

POD ID         NAME   STATUS    CREATED       # OF CONTAINERS   INFRA ID

fa7924a5196c   demodb Running   5 minutes ago 3               3005ed8491d0

$ sudo podman ps -p

CONTAINER ID  IMAGE                       COMMAND              CREATED       STATUS            PORTS  NAMES           POD

2597454063f8  quay.io/baude/mariadbpoddemo:latest  docker-entrypoint... 5 minutes ago Up 5 minutes ago         eloquent\_golick  fa7924a5196c

And we can start the nginx container to restore the pod back to its original state.

$ sudo podman start optimistic\_edison

optimistic\_edison

$ sudo podman ps -p

CONTAINER ID  IMAGE                       COMMAND              CREATED       STATUS            PORTS  NAMES            POD

02e37a3b9873  quay.io/libpod/alpine\_nginx:latest   nginx -g daemon o... 8 minutes ago Up 6 seconds ago         optimistic\_edison  fa7924a5196c

2597454063f8  quay.io/baude/mariadbpoddemo:latest  docker-entrypoint... 8 minutes ago Up 8 minutes ago         eloquent\_golick    fa7924a5196c

We can also stop the pod and all of its containers using the podman pod stop command.

$ sudo podman pod stop demodb

fa7924a5196cb403298ad2ce24f0db30a3790e80729c7704ef5fdc27302f7ad0

$ sudo podman ps -ap

CONTAINER ID  IMAGE                        COMMAND              CREATED        STATUS                     PORTS                    NAMES               POD

02e37a3b9873  quay.io/libpod/alpine\_nginx:latest   nginx -g daemon o... 10 minutes ago Exited (0) 21 seconds ago                           optimistic\_edison   fa7924a5196c

2597454063f8  quay.io/baude/mariadbpoddemo:latest  docker-entrypoint... 10 minutes ago Exited (0) 19 seconds ago                           eloquent\_golick     fa7924a5196c

3005ed8491d0  k8s.gcr.io/pause:3.1                                      10 minutes ago Exited (0) 19 seconds ago  0.0.0.0:43871->3306/tcp fa7924a5196c-infra  fa7924a5196c

And if we look at the status of the pod, it will show a state of “Exited”.

$ sudo podman pod ps

POD ID         NAME     STATUS   CREATED        # OF CONTAINERS   INFRA ID

fa7924a5196c   demodb Exited   13 minutes ago 3               3005ed8491d0

Likewise, we can also start the pod and all of its containers back up.  After which, all the containers in the pod should be running and the pod should show a status of “Running”.

$ sudo podman pod start demodb

fa7924a5196cb403298ad2ce24f0db30a3790e80729c7704ef5fdc27302f7ad0

$ sudo podman ps -p

CONTAINER ID  IMAGE                       COMMAND              CREATED        STATUS            PORTS  NAMES             POD

02e37a3b9873  quay.io/libpod/alpine\_nginx:latest   nginx -g daemon o... 14 minutes ago Up 5 seconds ago         optimistic\_edison  fa7924a5196c

2597454063f8  quay.io/baude/mariadbpoddemo:latest  docker-entrypoint... 14 minutes ago Up 4 seconds ago         eloquent\_golick    fa7924a5196c

$ sudo podman pod ps

POD ID         NAME   STATUS    CREATED        # OF CONTAINERS  INFRA ID

fa7924a5196c   demodb Running   14 minutes ago 3                3005ed8491d0

There is also a podman pod restart command that will restart all the containers within a Pod.

Wrap up

The ability for Podman to handle pod deployment is a clear differentiator to other container runtimes.  As a libpod maintainer, I am still realizing the advantages of having pods even in a localized runtime. There will most certainly be more development in Podman around pods as we learn how users exploit the use of them.

For more information on Podman, make sure you visit the [libpod project page](https://github.com/containers/libpod) on github. Relevant blogs and news related to Podman can also be found at [podman.io](https://podman.io/).

Podman is included with Red Hat Enterprise Linux 7.6 as well as Red Hat Enterprise Linux 8 beta.