Cheat sheet

Podman

This cheat sheet covers the commands used for working with Podman, a popular tool for managing containers. Podman commands are mostly compatible with Docker. As stated on the Podman landing page, "Podman is a daemonless, open source, Linux native tool designed to make it easy to find, run, build, share and deploy applications using Open Containers Initiative (OCI) Containers and Container Images."

Podman, unlike Docker, does not require a daemon running as superuser (root). This means that Podman interacts directly with the various components in the Linux container ecosystem instead of relying on a continuously running daemon to intermediate between these components on the caller's behalf. Podman does not run as root by default, which reduces the potential for a security hazard. Also, Podman does not spawn containers as child processes, thus making the containers it creates durable and independent of Podman.

Overall, Podman is an excellent alternative to Docker containers when you need increased security, unique identifier (UID) separation using namespaces, and integration with systemd.

> **Important**
>
> The `$` symbol that proceeds commands in the examples represents the command line prompt.

Working with image repositories

The following sections describe the Podman commands for working with image repositories.

podman images

```
podman images [options]
```

Lists all local images.

Example:

The following example lists all the container images stored on the local machine. Note that the local machine has container images from two public container image repositories, `quay.io` and `docker.io`:

```
$ podman images
REPOSITORY                      TAG         IMAGE ID      CREATED       SIZE
quay.io/ansible/ansible-runner  latest      697a4af2d624  16 hours ago  738 MB
docker.io/library/mysql         latest      6126b4587b1b  18 hours ago  525 MB
docker.io/library/redis         latest      f1b6973564e9  4 weeks ago   116 MB
docker.io/library/nginx         latest      c316d5a335a5  4 weeks ago   146 MB
```

podman rmi

```
podman rmi [-f] <<image>:<tag>
```

Removes a local image from the local cache. Use the `-f` option to force removal. This command removes the image only from the local system, not from the remote registry. The image can be specified by a name or a UUID.

Example:

The following example removes a container image with the UUID c316d5a335a5 from the container image repository on the local computer:

```
$ podman rmi [-f] c316d5a335a5
```

podman push

```
podman push <registry_url>/<username>/<image>:<tag>
```

Pushes a container image to a remote registry.

Example:

The following example uses the `podman build` command with the `-t` option to create a local container image with the name and tag `quay.io/myrepo/customer_container:v1`. The result is shown in an abbreviated format. The `podman images` podman images command lists the created container image. Finally, a `podman push` command pushes the container image to the remote repository, `quay.io:`:

```
$ podman build -t quay.io/myrepo/customer_container:v1 .
.
.
Successfully tagged quay.io/myrepo/customer_container:v1
Successfully tagged localhost/mynode:v1
Successfully tagged localhost/mynode:best
a6b028f25b45f533ae3078fe1d97df2a9cd27691e8d7b3317a0bdeaa24d1e705

$ podman images
REPOSITORY                                TA       IMAGE ID      CREATED       SIZE
quay.io/myrepo/customer_container         v1       a6b028f25b45  4 days ago    1.02 GB

$ podman push quay.io/myrepo/customer_container:v1
```

podman history

```
podman history [options] <image>:<tag>
```

Displays historical information about a container image that has been download and stored on the local machine.

Example:

The following example is an excerpt of output from a `podman history` command that gets historical information for the container image of the `zipkin` distributed tracing framework that was retrieved from the `quay.io` container image repository:

```
$ podman history quay.io/openzipkin/zipkin:latest
ID            CREATED       CREATED BY                                      SIZE   COMMENT
b9fb334d7cd1  2 years ago   /busybox/sh -c #(nop)  ENTRYPOINT ["/busyb...  0 B
<missing>     2 years ago   /busybox/sh -c #(nop)  EXPOSE 9410 9411        0 B
<missing>     2 years ago   /busybox/sh -c #(nop)  USER zipkin             0 B
<missing>     2 years ago   /busybox/sh -c ln -s /busybox/* /bin           197 kB
<missing>     2 years ago   /busybox/sh -c #(nop) WORKDIR /zipkin          0 B
.
.
.
```

podman login

```
podman login [options] <image_registry_url>
```

Logs a user into a remote container image registry. The command prompts the user for a username and password.

Example:

The following example logs the user in to the `quay.io` container image repository:

```
$ podman login quay.io
Username: cooluser
Password:
Login Succeeded!
```

podman logout

Logs out of the current container registry.

```
podman logout [options]
```

Example:

The following example logs the user out of the `quay.io` container image repository:

```
$ podman logout quay.io
Removed login credentials for quay.io
```

podman pull

```
podman pull [options] <remote_registry_url>/<username>/<image>:<tag>
```

Pulls an image from a remote registry.

Example:

```
$  podman pull quay.io/ansible/ansible-runner:latest
```

The following example retrieves the latest version of the container image for the `ansible-runner` tool from the `quay.io` container image repository:

podman search

```
podman search [options] <search_string>
```

Searches the container image registries defined in the file `/etc/containers/registries.conf`.

Example:

The example that follows uses the registry entries defined in the file `/etc/containers/registries.conf` as shown in the snippet below :

```
[registries.search]
registries = ["quay.io", "registry.fedoraproject.org", "registry.access.redhat.com",
"registry.centos.org", "docker.io"]
```

The following `podman search` command finds container images that include the string `pinger`. The response is displayed in an abbreviated format:

```
$ podman search pinger
INDEX       NAME                                      DESCRIPTION
STARS       OFFICIAL    AUTOMATED
quay.io     quay.io/giantswarm/calico-ipip-pinger
0
quay.io     quay.io/dontpayfull/calico-ipip-pinger
0
quay.io     quay.io/zonggen/fcos-pinger-backend       Server image for telemetry service
of FCOS (...  0
quay.io     quay.io/ksemaev/pinger
0
quay.io     quay.io/murph83/pinger
0
quay.io     quay.io/sosivio/sosivio-node-pinger
0
quay.io     quay.io/sebv/pinger
0
docker.io   docker.io/hosterping/pinger               Pinger v2 für Hoster-Ping.de
0
docker.io   docker.io/afrank/pinger
0
docker.io   docker.io/subfuzion/pinger                Simple service that sends a pong
response        0
docker.io   docker.io/pingerua/samples
0
docker.io   docker.io/superbrilliant/pinger
0
docker.io   docker.io/reselbob/pinger                 A simple utility web server image
that outpu...  1
.
.
.
```

Building images

The following sections describe the various Podman commands for building container images.

podman build

```
podman build [options] <image>:<tag> [-f <Dockerfile>]
```

Builds and tags an image using the instructions in a Dockerfile, which can be specified as a filename or a URL. The `-f` option specifies the location of the Dockerfile. If the `-f` option is omitted, the command looks for a Dockerfile in the current directory. Once the container image is built, it is stored in the container image repository on the local machine

Example:

The following example creates a container image using the default Dockerfile in the local directory. Then the command `podman images` is used to list the container images stored in the local repository. The output of the container image list is piped to `grep` to display only container images that have the string `mynode`:

```
$ podman build -t mynode:v1 .
STEP 1/3: FROM node:latest
STEP 2/3: CMD ["-v"]
--> 959e797d01b
STEP 3/3: ENTRYPOINT ["node"]
COMMIT mynode:v1
--> a6b028f25b4
Successfully tagged localhost/mynode:v1
a6b028f25b45f533ae3078fe1d97df2a9cd27691e8d7b3317a0bdeaa24d1e705

$ podman images | grep mynode
localhost/mynode                v1          a6b028f25b45  About a minute ago  1.02 GB
```

The following example creates a container image using a file named `Otherdockerfile`:

```
$ podman build -t othernode:v1 -f Otherdockerfile
STEP 1/2: FROM node:latest
STEP 2/2: RUN echo "The latest version of Node is installed"
The latest version of Node is installed
COMMIT othernode:v1
--> 600590954fc
Successfully tagged localhost/othernode:v1
600590954fc5dff1d32ffda6bf34f07e674feee056183c8a7bfb726c3421b49e
```

podman tag

```
podman tag <image>:<tag> <image>:<new_tag>
```

or

```
podman tag <image_uuid> <image>:<new_tag>
```

Creates a new tag for an existing container image in the local repository.

Example:

The following example first executes a `podman images` command to list existing container images on the local machine. The `podman tag` command is then executed against the image with the UUID `a6b028f25b45` and applies the new tag `best`. The container images are listed again to show the new tag.

```
$ podman images
REPOSITORY                      TAG         IMAGE ID      CREATED         SIZE
localhost/othernode             v1          600590954fc5  5 minutes ago   1.02 GB
localhost/mynode                v1          a6b028f25b45  14 minutes ago  1.02 GB

$ podman tag a6b028f25b45 mynode:best

$ podman images
REPOSITORY                      TAG         IMAGE ID      CREATED         SIZE
localhost/othernode             v1          600590954fc5  6 minutes ago   1.02 GB
localhost/mynode                v1          a6b028f25b45  15 minutes ago  1.02 GB
localhost/mynode                best        a6b028f25b45  15 minutes ago  1.02 GB
```

Working with containers

The following sections describe the Podman commands for creating and running containers.

podman run

```
podman run [options] <repo>/<image>:<tag>
```

Runs a container based on a given `<image>:<tag>` pair. If the image exists on the local machine, that image will be used. Otherwise, `podman run` attempts to get the container image from the remote repository specified in the command.

Example: :

The following example runs a container using the latest version of the container image for the distributed tracing tool `zipkin` that is stored in the `quay.io` container repository. The `-d` option runs the container in the background in order to free the terminal window to accept future input. The output from `podman run` is the containers UUID.

Then, the command `podman ps -a` lists the running containers. Because the `zipkin` container was not assigned a name when it was created, the arbitrary name `laughing_mahavira` is assigned to the container:

```
$ podman run -d quay.io/openzipkin/zipkin
ea35aa9eda875dd0c3ea34beb6216cf1148725272f28829ea1d3ba262f9f2ada

$ podman ps -a
CONTAINER ID  IMAGE                            COMMAND  CREATED     STATUS   PORTS     NAMES
ea35aa9eda87  quay.io/openzipkin/zipkin:latest        3 min ago  Up 3 min ago    laughing_mahavira
```

The following example creates and runs the container using the `ngnix:latest` container image. The `-d` option runs the container in the background. The `--name` option gives the container the name `mywebserver`.

After the container is created, the command `podman ps -a` lists the containers running on the local machine. Note that the lists the containers running on the local machine. Note that the `nginx` container has the name `mywebserver`.

```
$ podman run -d --name mywebserver -it nginx:latest

$ podman ps -a
CONTAINER ID  IMAGE                                COMMAND   CREATED      STATUS   PORTS   NAMES
ea35aa9eda87  quay.io/openzipkin/zipkin:latest           6 min ago  Up 6 min ago     laughing_mahavira
90ac3eb5f5a6  docker.io/library/nginx:latest   nginx -g… 4 sec ago  Up 4 sec ago     mywebserver
```

The following example creates and runs the container. The option `–rm` causes the container to be removed after it exits.

After `podman run` executes, the command `podman ps -a` lists the available containers. Note that the `nodejs` container is not listed. This is because the `-rm` option was used when running it. The `nodejs` container spun up, but because there was no activity for it to execute, it exited. Once the container exited, it was removed from the local machine:

```
$ podman run --rm quay.io/centos7/nodejs-14-centos7:latest

$ podman ps -a
CONTAINER ID  IMAGE                                COMMAND               CREATED         STATUS
PORTS       NAMES
ea35aa9eda87  quay.io/openzipkin/zipkin:latest                           15 minutes ago  Up 15 minutes
ago              laughing_mahavira
90ac3eb5f5a6  docker.io/library/nginx:latest    nginx -g daemon o...  9 minutes ago   Up 9 minutes
ago               mywebserve
```

The following example creates and runs the container using the `-it` option. This option creates a terminal and presents a command prompt within the container after the container gets up and running:

```
podman run -it nginx:latest
```

The following example creates and runs a container using the `nginx:latest` image. After the container is up and running, the `pwd` command is executed against file system internal to the container to report its current working directory. The output shows that the current working directory is the root (`/`) directory:

```
$ podman run nginx:latest pwd
/
```

podman stop

```
podman stop [options] <container>
```

Gracefully stops a container from running. The container can be specified by name or UUID

Example:

The following example first executes `podman ps -a` to list all containers on the local machine. Note that the two containers listed have a status of `Up <n> minutes ago` . The `podman stop` command is then executed against the container that has the name `mywebserver` .

The command `podman ps -a` is called again. Both containers are listed, but the container named `mywebserver` has a status of `Exited (0) 3 seconds ago` , which is the point in time when the command `podman stop` was called.

```
$ podman ps -a
CONTAINER ID  IMAGE                                COMMAND               CREATED         STATUS
PORTS       NAMES
ea35aa9eda87  quay.io/openzipkin/zipkin:latest                           27 minutes ago  Up 27 minutes ago
laughing_mahavira
90ac3eb5f5a6  docker.io/library/nginx:latest       nginx -g daemon o...  21 minutes ago  Up 21 minutes ago
mywebserver

$ podman stop mywebserver
mywebserver

$ podman ps -a
CONTAINER ID  IMAGE                                COMMAND               CREATED         STATUS
PORTS       NAMES
ea35aa9eda87  quay.io/openzipkin/zipkin:latest                           28 minutes ago  Up 28 minutes ago
laughing_mahavira
90ac3eb5f5a6  docker.io/library/nginx:latest       nginx -g daemon o...  21 minutes ago  Exited (0) 3 seconds
ago             mywebserver
```

podman start

```
podman start [options] <container>
```

Starts an existing container. The container can be specified by name or UUID.

Example:

The following example uses `podman ps -a` to list containers on the local machine. Note that the container named `mywebserver` has a STATUS of `Exited (0) 3 seconds ago`. The container is stopped.

Next, the command `podman start mywebserver` executes to restart the container. Then `podman ps -a` is executed again. Now the container named `mywebserver` has a status of `Up 31 seconds ago`. The container has been started and is running.

```
$ podman ps -a
CONTAINER ID  IMAGE                                COMMAND               CREATED         STATUS
PORTS       NAMES
ea35aa9eda87  quay.io/openzipkin/zipkin:latest                           28 minutes ago  Up 28 minutes ago
laughing_mahavira
90ac3eb5f5a6  docker.io/library/nginx:latest       nginx -g daemon o...  21 minutes ago  Exited (0) 3
seconds ago             mywebserver

$ podman start mywebserver
mywebserver

$ podman ps -a
CONTAINER ID  IMAGE                                COMMAND               CREATED         STATUS
PORTS       NAMES
ea35aa9eda87  quay.io/openzipkin/zipkin:latest                           33 minutes ago  Up 33 minutes ago
laughing_mahavira
90ac3eb5f5a6  docker.io/library/nginx:latest       nginx -g daemon o...  27 minutes ago  Up 31 seconds ago
mywebserver
```

The following example runs the container image `docker.io/library/nginx`. The `-d` command runs the container in the background. The `--name` option gives the container the name `mywebserver`. The `-p` option assigns port number `8181` running on the local computer (localhost) to the port number `80`, which is where the NGINX web server within the container is listening for income requests:

```
$ podman run --name mynginx -d -p 8181:80 docker.io/library/nginx
a4b59499314f7c4c6819340ec8e15732cb93c21c131fbd709e09370972fda1b7

$ podman ps -a
CONTAINER ID IMAGE                           COMMAND   CREATED   STATUS       PORTS                 NAMES
a4b59499314f docker.io/library/nginx:latest nginx -g… 8 sec ago Up 7 sec ago 0.0.0.0:8181->80/tcp mynginx
```

podman create

```
podman create [options] </repo/image:tag>
```

Creates a container from a container image but does not start it.

Example:

The following example creates a container from the `quay/redis` image found on the `quay.io` container image repository:

```
$ podman create --name myredis quay.io/quay/redis
dcc2491a3d16809c5c7b939e48aa99ded40779cb79140b1b9ae8702561901952

$ podman ps -a
CONTAINER ID  IMAGE                       COMMAND          CREATED        STATUS      PORTS       NAMES
dcc2491a3d16  quay.io/quay/redis:latest  conf/redis.conf  3 seconds ago  Created                 myredis
```

podman restart

```
podman restart [options] <container>
```

Restarts an existing container. The container can be specified by name or UUID.

Example:

The following example uses `podman ps -a` to list the containers installed on the host computer. Note that the status of the container named `myredis` is `Created`.

Then the `podman restart` command is used to start the container named `myredis`. Finally, the `podman ps -a` command is called again. The status of the container is now `Up 8 seconds ago`, hence the container is running.

```
$ podman ps -a
CONTAINER ID  IMAGE                                COMMAND               CREATED       STATUS
PORTS             NAMES
dcc2491a3d16  quay.io/quay/redis:latest            conf/redis.conf       22 hours ago  Created
myredis

$ podman restart  myredis

$ podman ps -a
CONTAINER ID  IMAGE                                COMMAND               CREATED       STATUS
PORTS         NAMES
dcc2491a3d16  quay.io/quay/redis:latest            conf/redis.conf       22 hours ago  Up 8 seconds
ago               myredis
```

podman rm

```
podman rm [options] <container>
```

Removes a container from the host computer. The container can be specified by name or UUID.

Example:

The following example uses `podman ps -a` list the containers installed on the host computer. Note that the container named `myredis` is running. Then the command `podman rm` with the `-f` option forces the removal of the running container named `myredis`. Finally, `podman ps -a` is called again. Note that the container has been removed from the computer.

```
$ podman ps -a
CONTAINER ID  IMAGE                                COMMAND               CREATED       STATUS
PORTS         NAMES
dcc2491a3d16  quay.io/quay/redis:latest            conf/redis.conf       22 hours ago  Up 8 seconds
ago               myredis

$ podman rm -f myredis
dcc2491a3d16809c5c7b939e48aa99ded40779cb79140b1b9ae8702561901952

$ podman ps -a
CONTAINER ID  IMAGE                                 COMMAND               CREATED       STATUS
PORTS               NAMES
```

podman wait

```
podman wait [options] <container>
```

Waits for the specified container to meet a condition. The default condition is `stopped`.

Example:

The following example uses `podman ps -a` to list containers on the local computer. Then the `podman wait` command is issued against the container with the UUID `569ddc895737`. The current process (in this case, the user's terminal) waits until the container with the UUID `569ddc895737` stops.

```
$ podman ps -a
CONTAINER ID  IMAGE                              COMMAND               CREATED         STATUS
PORTS                 NAMES
a4b59499314f  docker.io/library/nginx:latest     nginx -g daemon o...  23 hours ago    Up About an hour
ago  0.0.0.0:8181->80/tcp  mynginx
569ddc895737  quay.io/openzipkin/zipkin:latest                         46 minutes ago  Up 44 minutes ago
myzipkin

$ podman wait 569ddc895737
```

podman stats

```
podman stats [options] [<container>]
```

Displays a live stream of a container's resource usage. The container can be specified by name or UUID. If no container is specified, the command displays a live stream of the statistics for all containers running as root.

Note: The command `podman stats` must be executed as `sudo` and shows only containers running with root privileges.

Example:

The following example calls the `podman stats` command as the root user. Because no container name or UUID is defined in the command, `podman stats` shows the stats for all containers running as root on the local machine:

```
sudo podmam stats

ID            NAME        CPU %       MEM USAGE / LIMIT  MEM %       NET IO          BLOCK IO
PIDS        CPU TIME      AVG CPU %
153aa53a52b9  rootnginx   --          2.044MB / 8.148GB  0.03%       698B / 2.574kB  8.192kB / 62.46kB
2           67.274094ms   1.02%
f7ac2c719ff7  myredis     0.19%       7.631MB / 8.148GB  0.09%       978B / 7.474kB  -- / --
5           717.895399ms  0.14%
```

podman inspect

```
podman inspect [options] <container>
```

Returns metadata describing a running container. The container can be specified by name or UUID. The default format for the metadata is JSON.

Example:

The following example inspects the container with the name `myginx`. The result is piped to the `more` command with the `-10` option to display the first 10 lines of output.

```
$ podman inspect mynginx | more -10
[
    {
        "Id": "a4b59499314f7c4c6819340ec8e15732cb93c21c131fbd709e09370972fda1b7",
        "Created": "2022-02-24T11:17:00.499462518-08:00",
        "Path": "/docker-entrypoint.sh",
        "Args": [
            "nginx",
            "-g",
            "daemon off;"
        ],
--More--
```

Working with container processes and resources

The following sections describe the various Podman commands for working with containers and container images beyond creating, running, and stopping containers.

podman ps

```
podman ps [options]
```

Lists the containers on the local system.

Example:

The following example uses `podman ps -a` to show all containers on the local computer, including those that are running and those in another state such as `Created` or `Exited`:

```
$ podman ps -a
CONTAINER ID  IMAGE                            COMMAND               CREATED         STATUS
PORTS                 NAMES
a4b59499314f  docker.io/library/nginx:latest   nginx -g daemon o...  23 hours ago    Up 22 minutes
ago           0.0.0.0:8181->80/tcp  mynginx
569ddc895737  quay.io/openzipkin/zipkin:latest                       38 seconds ago  Exited (143) 3
seconds ago                         myzipkin
```

podman commit

```
podman commit [options] <container> <new_image>:<tag>
```

Creates a new container image based on the current state of a running container. The container can be specified by name or UUID.

Example:

The following example creates a new container image named `yourzipkin` with the tag `test` from the running container named `myzipkin`.

Then `podman images` lists the container images on the computer. Note that the container image `localhost/yourzipkin:test` is listed:

```
$ podman commit myzipkin yourzipkin:test

$ podman images
REPOSITORY                              TAG                     IMAGE ID      CREATED         SIZE
localhost/yourzipkin                    test                    179d9b389a21  21 seconds ago  156
MB
localhost/mynode                        v1                      a6b028f25b45  24 hours ago    1.02
GB
```

podman attach

```
podman attach [options] <container>
```

Attaches to a running container and views its output or controls it. The container can be specified by name or UUID. Use the key sequence `Ctrl` + `p` `Ctrl` + `q` to detach from the container while leaving it running.

Example:

The following example attaches to the container named ` myzipkin `:

```
$ podman attach myzipkin
```

podman exec

```
podman exec <container> <command>
```

Executes a command in a running container. The container can be specified by name or UUID.

Example:

The following example uses `podman exec` with the `-it` option to enter into the container named `myzipkin` and display a command prompt within the container by using the internal shell invoked by the `sh` command:

```
$ podman exec -it myzipkin sh
~ $ ls
BOOT-INF   META-INF   classpath  org        run.sh
```

podman top

```
podman top <container>
```

Displays the running processes of a container. The container can be specified by name or UUID.

Example:

The following example displays the processes running within the container named `myginx` , along with their CPU utilization:

```
$ podman top mynginx
USER        PID         PPID        %CPU        ELAPSED           TTY         TIME        COMMAND
root        1           0           0.000       29m55.560928305s  ?           0s          nginx:
master process nginx -g daemon off;
nginx       23          1           0.000       29m54.561101763s  ?           0s          nginx:
worker process
```

podman logs

```
podman logs [options] <container>
```

Displays the logs of a container. The container can be specified by name or UUID.

Example:

The following example uses the command `podman logs` to display log information about the container named `myginx`. The `-t` option displays the timestamp for each log entry:

```
$ podman logs -t mynginx
2022-02-25T09:37:46.090921000-08:00 /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will
attempt to perform configuration
2022-02-25T09:37:46.091742000-08:00 /docker-entrypoint.sh: Looking for shell scripts in /docker-
entrypoint.d/
2022-02-25T09:37:46.104675000-08:00 /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-
on-ipv6-by-default.sh
2022-02-25T09:37:46.180498000-08:00 10-listen-on-ipv6-by-default.sh: info: IPv6 listen already enabled
2022-02-25T09:37:46.181151000-08:00 /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-
on-templates.sh
2022-02-25T09:37:46.223979000-08:00 /docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-
worker-processes.sh
2022-02-25T09:37:46.232448000-08:00 /docker-entrypoint.sh: Configuration complete; ready for start up
2022-02-25T09:37:46.361178000-08:00 2022/02/25 17:37:46 [notice] 1#1: using the "epoll" event method
2022-02-25T09:37:46.361275000-08:00 2022/02/25 17:37:46 [notice] 1#1: nginx/1.21.6
2022-02-25T09:37:46.361323000-08:00 2022/02/25 17:37:46 [notice] 1#1: built by gcc 10.2.1 20210110
(Debian 10.2.1-6)
2022-02-25T09:37:46.361362000-08:00 2022/02/25 17:37:46 [notice] 1#1: OS: Linux 4.18.0-348.el8.x86_64
2022-02-25T09:37:46.361397000-08:00 2022/02/25 17:37:46 [notice] 1#1: getrlimit(RLIMIT_NOFILE):
262144:262144
2022-02-25T09:37:46.361434000-08:00 2022/02/25 17:37:46 [notice] 1#1: start worker processes
2022-02-25T09:37:46.361470000-08:00 2022/02/25 17:37:46 [notice] 1#1: start worker process 23
```

podman pause

```
podman pause [options] [<container>]
```

Pauses all the processes in a specified container or all containers. The command can be run only against containers that have root privileges. The container can be specified by name or UUID.

Example:

The following example pauses the container named `rootnginx`. The command is run using the `sudo` command because the container named `rootnginx` has root privileges:

```
$ sudo podman pause rootnginx
153aa53a52b93a480deab0f781d4a2b851ab8559d72c033c875f534af5e282f8
$ sudo podman ps -a
CONTAINER ID  IMAGE                            COMMAND               CREATED         STATUS
PORTS       NAMES
f7ac2c719ff7  docker.io/library/redis:latest   redis-server          36 minutes ago  Up 36 minutes ago
myredis
153aa53a52b9  quay.io/bitnami/nginx:latest     nginx -g daemon o...  30 minutes ago  paused
rootnginx
```

podman unpause

```
podman unpause [options] [<container>]
```

Unpauses all processes in a specified container or all containers. The command can be run only against containers that have root privileges. The container can be specified by name or UUID.

Example:

The following example restarts the container named `rootnginx` from a paused state. The command is run using the `sudo` command because the container named `rootnginx` has root privileges:

```
$ sudo podman unpause rootnginx
153aa53a52b93a480deab0f781d4a2b851ab8559d72c033c875f534af5e282f8
$ sudo podman ps -a
CONTAINER ID  IMAGE                           COMMAND               CREATED         STATUS
PORTS       NAMES
f7ac2c719ff7  docker.io/library/redis:latest  redis-server          37 minutes ago  Up 37 minutes ago
myredis
153aa53a52b9  quay.io/bitnami/nginx:latest    nginx -g daemon o...  30 minutes ago  Up 30 minutes ago
rootnginx
```

podman port

```
podman port [options] <container>
```

Lists the port mappings from a container to localhost. The container can be specified by name or UUID.

Example:

The following example reports the port binding for the container named `mynginx`:

```
$ podman port mynginx
80/tcp -> 0.0.0.0:8181
```

Working with container's filesystem

The sections describe the Podman commands for dealing with the host computer's file system.

podman diff

```
podman diff [options] <container>
```

Displays all the changes caused by a container to the filesystem. The container can be specified by name or UUID.

Example:

The following example reports how the files and directories on the host operating system have been affected by running the container named `mynginx`. The letter `C` indicates the the file or directory has been changed. The letter `A` indicates that the file or directory has been added:

```
$ podman diff mynginx
C /etc
C /etc/nginx
C /etc/nginx/conf.d
C /etc/nginx/conf.d/default.conf
A /run/nginx.pid
C /var
C /var/cache
C /var/cache/nginx
A /var/cache/nginx/client_temp
A /var/cache/nginx/fastcgi_temp
A /var/cache/nginx/proxy_temp
A /var/cache/nginx/scgi_temp
A /var/cache/nginx/uwsgi_temp
```

podman mount

```
podman mount [options] <container>
```

Mounts and reports the location of a container's filesystem on the host computer. This command is useful to inspect the filesystem of a container without having to run `podman exec -it` to enter the running container. The container can be specified by name or UUID.

Example:

The following example lists the containers running as root on the local computer. Then the command `sudo podman mount` is called on the running container named `myredis`. The result of calling `sudo podman mount` is the directory where the container's files are located. Finally, `sudo ls` is called on the container's directory. Note that the filesystem has the root directories of a Linux computer running Redis. The command must be run as `sudo`:

```
$ sudo podman ps -a
CONTAINER ID  IMAGE                           COMMAND               CREATED        STATUS
PORTS       NAMES
f7ac2c719ff7  docker.io/library/redis:latest  redis-server          3 days ago     Created
myredis

$ sudo podman mount myredis
/var/lib/containers/storage/overlay/b4f1aaed89bc56ab7b6b63fc6124623036497619cc9f7392bfb529bf1f38ba45/
merged

$ sudo ls /var/lib/containers/storage/overlay/
b4f1aaed89bc56ab7b6b63fc6124623036497619cc9f7392bfb529bf1f38ba45/merged
bin  boot  data  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp
usr  var
```

podman umount

```
podman umount [options] <container>
```

Unmounts a container's root filesystem. The container can be specified by name or UUID.

Example:

The following command unmounts a container named `myredis`. The command must be run as `sudo`:

```
$ sudo podman unmount myredis
myredis
```

podman export

```
podman export -o <output_filename> <container>
```

Exports a container's filesystem to a tar file (a compressed package containing a complete directory structure). The container can be specified by name or UUID.

Example:

The following example uses the command `podman ps -a` to list the containers running on the local computer. Then the `podman export` command exports the filesystem of the container named `mynginx` to a tar file named `mynginx.tar`. Finally, the command `ls -lh` describes the details of the tar file:

```
$ podman ps -a
CONTAINER ID  IMAGE                                COMMAND               CREATED     STATUS
PORTS                 NAMES
a4b59499314f  docker.io/library/nginx:latest    nginx -g daemon o...  3 days ago  Up 50 minutes ago
0.0.0.0:8181->80/tcp  mynginx

$ podman export mynginx > mynginx.tar

$ ls -lh
total 138M
-rw-rw-r--. 1 guest guest 138M Feb 28 09:44 mynginx.tar
```

podman import

```
podman import <tar_filename>
```

Imports a tar file and saves it as a filesystem image.

Example:

The following example creates a container image from an existing tar file named `mynginx.tar`. The command creates a `new-nginx` with the tag `v1`. Finally, the command `podman images` is called to list the container image that was created:

```
$ podman import mynginx.tar new-nginx:v1
Getting image source signatures
Copying blob 51ae4d2a0ffb done
Copying config 8d555a4dac done
Writing manifest to image destination
Storing signatures
sha256:8d555a4dac4bdeb2840ca21a1540e4e736c5c5ee65d1b3e18f3dd81a913b133d

$ podman images
REPOSITORY           TAG         IMAGE ID      CREATED         SIZE
localhost/new-nginx  v1          ad3620ffa74c  41 minutes ago  144 MB
```

Miscellaneous

The following sections describe commands for discovering version and other information about Podman.

podman version

```
podman version
```

Reports information about the installed version of Podman.

Example:

The following example shows information about the installed version of Podman:

```
$ podman version
Version:      3.4.2
API Version:  3.4.2
Go Version:   go1.16.7
Built:        Thu Jan 13 02:15:49 2022
OS/Arch:      linux/amd64
```

podman info

```
podman info
```

Displays information about the instance of Podman installed on the local computer.

Example:

The following example displays information about the instance of Podman installed on the local computer. The output is piped to the `more` command using the `-10` option to show the first 10 lines of output:

```
$ podman info | more -10
host:
  arch: amd64
  buildahVersion: 1.23.1
  cgroupControllers: []
  cgroupManager: cgroupfs
  cgroupVersion: v1
  conmon:
    package: conmon-2.0.32-1.module+el8.5.0+13852+150547f7.x86_64
    path: /usr/bin/conmon
    version: 'conmon version 2.0.32, commit: 4b12bce835c3f8acc006a43620dd955a6a73bae0'
--More--
```