Chapter 02: Podman Basics

Podman is an open source tool that you can use to manage your containers locally. With podman, you can find, run, build or deploy OCI (Open Container Initiative) containers and container images.

Podman is daemonless.
Podman interacts directly with containers, images and registries without a daemon.

yum install -y podman
yum install -y container-tools

To install podman on rhel-9.

podman -v # podman --version

To pull images from registry.
podman pull image_name_aspersin_registry.

image_name = NAME: VERSION

To see all images.
podman images.

PAGE NO.
DATE:

- A container image contains a packaged version of your application, with all the dependencies necessary for the application to run.

- A container is an isolated runtime environment where applications are executed as isolated processes.

To create a container from image present in system/registry.

#podman run imagename

↳ Creates container with random container name.

#podman ps

↳ lists running containers.

#podman ps -a / #podman ps -all

↳ lists running as well as stopped containers.

#podman ps / ps -a

Container ID	IMAGE	COMMAND	CREATED
STATUS	PORTS	NAMES	

podman run --rm image_name comand

↳ executes the command in newly created container and removes it (i.e put it into stop mode) ⟶ Removes it completely

To give name to a container

podman run --name cont_name image_name

Podman identifies the containers by

① UUID

Universal Unique Identifier short identifier.

12 - Alphanumeric character

② universal Unique Identifier long identifier

64 - Alphanumeric character

podman ps -a

podman ps -a --format=json

podman ps -a --format='{{.pod_name}}'

podman ps -a --format='{{.Patt1}}: {{.Patt2}}'

PAGE NO.
DATE :

podman run -p 80:80 image-name

Mapping port option — local system port — Container port

Traffic in your local port is forwarded to the port inside the container, allowing access to application from your computer

podman run -d container_name

"-d" option is to run container in the background. (detached mode)

You can pass environment variables to a container by using the -e option.

podman run -e NAME='Red Hat' image-name. printenv NAME

printenv : To show all environmental variables.

Podman Desktop is a graphical way to manage images & containers.

Podman comes with a network called podman.
By default containers are attached to this network and can use it to communicate with one another.

* One container can be managed to communicate over multiple Network - Check??

Podman network management is done via 'podman network' subcommand:

Podman network create.
Creates new podman network

Podman network ls.
lists existing networks & their summary.

Podman network inspect
Detailed JSON config. data for the network.

podman network rm
Removes a network

Podman network prune
Removes networks not used by running containers.

podman network create
Connects running containers to existing network.
--net : Connects to network while cont. creation
disconnect: disconnects a container from a network.

PAGE NO.
DATE :

podman network create network1

→ Creates a new network as 'network1'

podman run -d --name cont1 \
--net network1 container_image

→ Creates a new container as 'cont1' and connects it to 'network1'

podman run -d --name cont2 \
--net network1, network2, network3 \
container_image

→ Creates a new container 'cont2' and connects it to multiple network, ie network (1, 2 & 3).

podman network connect network2 cont2

→ Connects already running container 'cont2' to existing network 'network2'

If network is not specified during 'podman run...' then the container connects to default network which uses 'slirp4nets' network mode.

Once container connects to a network, the container can communicate with other containers on that network.

Few components might prevent communication. One such example is firewall rules.

1) Network configuration
ex) firewalls
2) Docker Compose Networking (services)
4) Service Mesh Policies (like Istio)
5) Container Orchestration Policies.

Port forwarding

A container's network namespace is isolated. That means, networked app is only accessible within the container.

Port forwarding maps a port from the host machine where the container runs to a port inside of a container.

The '-p' option of the podman run command forwards a port, format is; Host_Pod : Cont_Port

PAGE NO.
DATE :

podman run -p 111:80 cont1

→ Maps port 80 of cont1 to 111 at host.

If ip address is not specified in mapping then it is accessible from all networks on host machine.

podman run -p 127.0.0.1:111:80 cont1

→ Maps port 80 of cont1 to 111 at host which is accessible from 127.0.0.1 ip.

podman port cont1

→ lists all mapped ports of cont1.

podman port --all

→ lists port mapping for all containers.

Containers attached to podman network are assigned private IP address for each network.

podman inspect cont1 \
-f '{{.NetworkSettings.Networks.apps.IPAddress}}'

When a container is started, the container creates a new ephemeral layer over its base container image layer called 'container layer'. This layer is read/write/storage ephemeral. Container images are characterized as immutable and layered.

To start a new process in a running container.

podman exec [options] container [command]

Options → ⓐ -e / --env to specify env. variable.
ⓑ -i / --interactive to instruct the container to accept input
ⓒ --tty / -t to allocate pseudo terminal
ⓓ --latest / -l to execute the command in the last created container.

podman exec -e Hello=Bye Bye cont1

allocating environment variable temporary.

To do it persistant allocate it during 'podman run' command.

exec is for executing certain commands inside container.

PAGE NO.
DATE :

-i : Do not provide terminal like experience.

-t : Opens a pseudo terminal.

-it : Opens a pseudo terminal & interactive behave in a typical way.

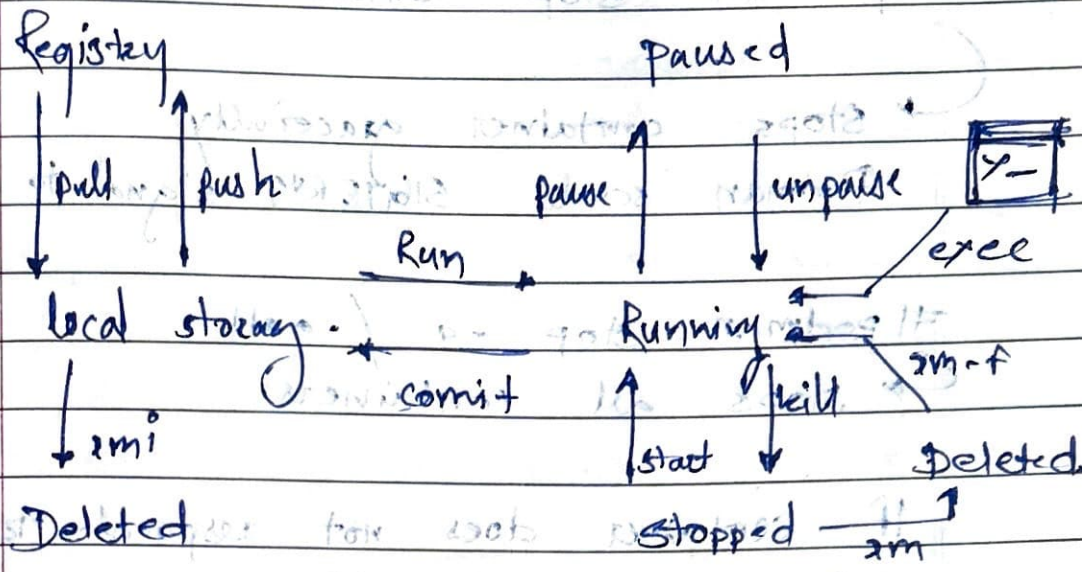
'Podman cp' command copies files to and from a running container.

podman cp option [cont]:source_path [cont]:dest_path.

podman cp container:/root/file .
→ copies files from container to current directory of host

podman cp file1 container:/root
→ copies file from host 'file1' to container's /root directory.

podman cp cont1:/root/file1 cont2:/root
→ copies file from 'cont1' to 'cont2'



podman inspect container/id
→ Information of containes in JSON array.

podman inspect container/id \
--format='{{.Content}}'

→ Elements are defined in double curly bracket which started & seperated by '.' and must start in Upper case.

PAGE NO.
DATE :

podman stop container_name/id

→ Stops container gracefully
Podman sends SIGTERM signal to container

podman stop -a / --all

→ Stops all containers

If container does not respond to SIGTERM signal, then podman sends SIGKILL signal to stop container forcefully
Podman waits 10 seconds by default before sending SIGKILL

podman stop --time = 100

→ Gives container a grace period of 100 seconds before sending sigkill signal.

podman kill hotpod

→ Sends SIGKILL signal to kill container.

podman pause container.

→ pause suspend all processes in cont.

podman unpause container_name
→ Resumes the paused container.

podman restart container_name.
→ Restarts / starts container.

podman stop container_name
→ stops container.

podman rm container_name.
→ Removes container.

You cannot remove container without stopping it.

podman rm -f container_name
→ Removes container without stopping the container.

podman rm all / rm -f all
→ Removes all containers forcefully.