PAGE NO.
DATE : 02/10/24

custom container images.

- A containerfile lists a set of instructions that a container runtime uses to build a container image.

- Each instruction causes a change that is captured in a resulting image layer.
  These layers are stalked together to form the resulting container image.

Podman supports both ~~do~~ Dockerfiles and Containerfile.

Podman executes the instructions in the Containerfile and applies the changes on top of a container base image.

The base image you choose determines the linux distribution and its components

- Package manager
- Init system
- Filesystem layout
- Preinstalled dependencies and runtimes

Base image influences image size, vendor support & processor compatibility.

Redhat provides base image called as Universal Base Images (UBI)

Four variants of UBI.

1) Standard:
UBI with DNF, systemd, utilities like gzip & tar.

2) Init:
Multiple app^n running within single container by using systemd

3) Micro Minimal.
Similar to 'Init' but image is smaller with less features. It have microdnf instead of DNF

4) Micro
Smallest available UBI. does not have package manager.

PAGE NO.
DATE :

Containerfiles uses domain-specific-language (DSL).

FROM : Base Image
WORKDIR : Current working directory.
COPY : Copy files from host to Cont
ADD : COPY + copying from URL + Unpacking for archives in cont
RUN : Runs a command on the Cont & commits.
Entrypoint : Commands that runs when Cont. is started. (can't overriden)
CMD : Commands that runs when Cont. is started ; can be overriden by argument command.
USER : Change the active user within the container.
LABEL : key-value pair to the metadata of the image.
EXPOSE : Adds a port to a image
ENV : Defines environment variables that at in container.
ARG : Defines variable during build-time. Discarded after built.
VOLUME : The path where Podman mounts persistent volume inside of the container.

Each Containerfile instruction runs in an independent container by using an intermediate image built from a every previous command.

#podman image tree image_name.
→ layers.

- ENV
The env instruction lets you specify environment dependent configuration.

To include an environmental variable, use the key-value format.

ENV Hello = "Redhat"

- ARG
ARG instructions are to define. build-time variables.

ARG key=value.

If --build-arg flag is provided on CLI then it overrides the C.file ARG
#podman build --build-arg key=redhat.

PAGE NO.
DATE :

The VOLUME instruction is to persistently store data. It can create more than one path for multiple volumes.

VOLUME PATH

podman inspect VOLUME_ID
→ Retrieves the local directory used by a volume.

Mountpoint field gives the absolute path of directory on host of volume.

Volume created from VOLUME instruction have random volume ID and are called Anonymous volumes.

podmane volume prune
→ Removes unused volume.
i.e. It will remove volumes that are not used by at least one container.

podman volume create VOLUME_NAME
→ Creates a named volume

podman volume ls \
--format = "{{.Name}} \t {{.Mountpoint}}"
→ shows name & mountpoint of volume.

A valid Containerfile must have at least one of these instruction.
Entrypoint (ENV or CMD)

Entrypoint
ENV & CMD instruction specify the command to execute whe the container starts.

The Entrypoint instructions defines an executable, or command, that is always part of the container execution.
Whereas,
If you only use the CMD instruction, then passing arguments to the container override the command provided on the CMD instruction.

If Entrypoint & CMD both are provided Entrypoint takes CMD as argument.

Entrypoint command can be override by
--entrypoint flag in command.
podman run --entrypoint env image.

PAGE NO.
DATE :

Multistage Builds

Uses multiple FROM instructions to create multiple independent cont. build processes (stages).

Every stage can use different base image & can copy files between stages.

The resulting container image consists of the last stage.

This is used to reduce image size & reduce possibilities of attack on surface of container.

How to ?

1) First stage : build an application
2) Second stage : Copy & 'serve' the static files by using a HTTP server.

"serve = expose the application"

- COW Copy-on-write

RUN, COPY and ADD etc. instructions, create layers (blobs) on a containerfile when starting the container, Podman creates a thin and mounts thin, ephemral, writable layer on the top of container image layers.
While deleting the container, this thin layer get deleted. Rest all layers are same-identical in an image.
This is called as COW.

Read about cache image layers???

■ Reducing Image layers.

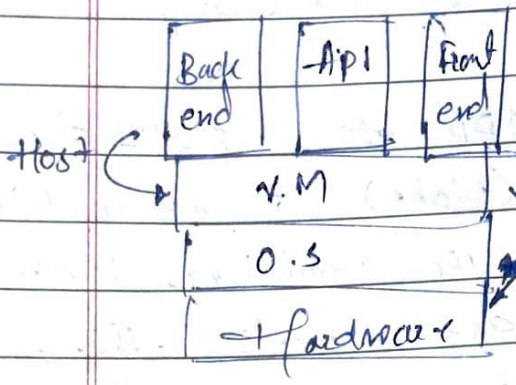
Image can be reduced by reducing number of layers by chaining the commands. Commands can be chained by using '&' or '&&'.

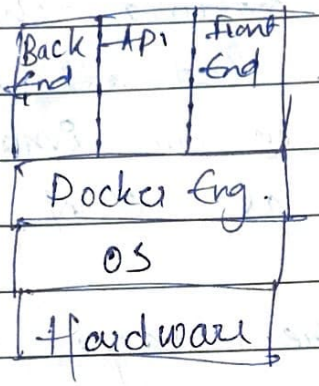
Other than this

--squash option in podman build
→ image layers are lower; size is same.
--squash-all option.
→ reduces layers as well as size of image.

PAGE NO.
DATE :

Rootless Podman





Rootless containers, or unpriviledge containess are containers that do not require administrator privileges.

→ * does not use root user. For creation,

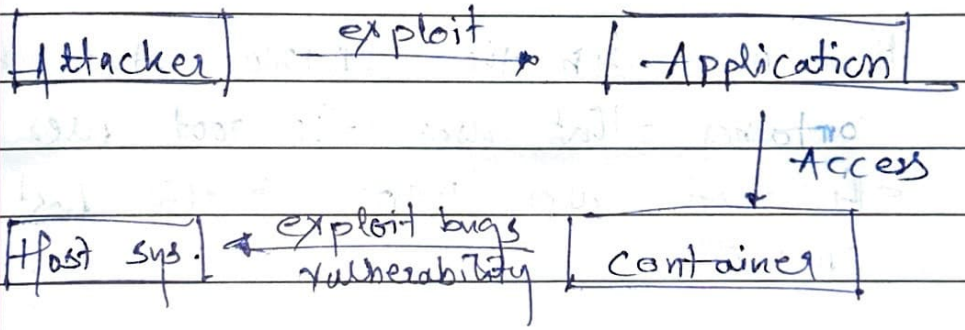
→ * User Root user inside container is not root user outside container.

→ * Container runtime does not use the root user.

As podman process exits after creating container. then the container process ID attaches to the systemd parent process ID..

Podman generally requires host OS system setup to run rootless containers.
Prerequisites are:

1) Cgroup v2
   To limit cont. resource use without requiring elevate privileges.
2) slirp4netns
   Implement user-mode networking for unprivileged network namespaces.
3) fuse-overlayfs
   Manages Copy-on-Write file system.



But root access is required in cont. for running certain comands therfore

From registry: image.
Run root commands. Add user.
USER Different user (Add user)

Comand run but, container starts with another user.

PAGE NO.
DATE :

Podman maps users inside of the container to unprivileged users on the host systems by using subordinate ID ranges.

There are defined in:

① /etc/subuid file

② /etc/subgid file

cat /etc/subuid

cat /etc/subgid

shows host user & the ID's it can allot

If a user with the ID 1000 starts a container that uses the root user, then the root user maps to the host user ID 1000.

To add subuid

sudo usermod --add-subuids 100000-165535 --add-subgids 100000-165535 student.

grep student /etc/subuid /etc/subgid.

→ Files /etc/subuid & /etc/subgid should exists before running these commands

After creating subuids & subgids you must execute the "podman system migrate" for the new subordinate ID ranges to take effect

Podman top ⇒ Verify the mapped user.

#Pd

#podman top cont_id huser user.

#USER USER
ID ID

→ #USER = Host User.
USER = Container user.

This shows the USER (cont) is mapped with #USER of host

{ Alternatively }

Container# cat /proc/self/uid_map /proc/self/gid_map
0 1000 1
0 1000 1

Subordinate id does not take place if Container is made by root or sudo.

PAGE NO.
DATE :

Limitations of rootless Container.

- Non trivial Containerization
- Required use of privileged ports & utilities.

→ Applications such as webservers cant be rootless due to access issue, for such case Redhat have there speial Containers that could be run as rootless.

→ Rootless containers dont have access to privileged ports. In such case you can do port forwarding

If still acces is required for privileged ports then:

sudo sysctle -w "net.ipv4.ip_unprivileged_port_start=79"

→ ~~Root can bind~~
Rootless containers can bind to port 80 and higher.

sudo sysctle -w "net.ipv4.ping_group_range=0 2000000"

→ Range that can use ping comand.