Persisting DATA

COW

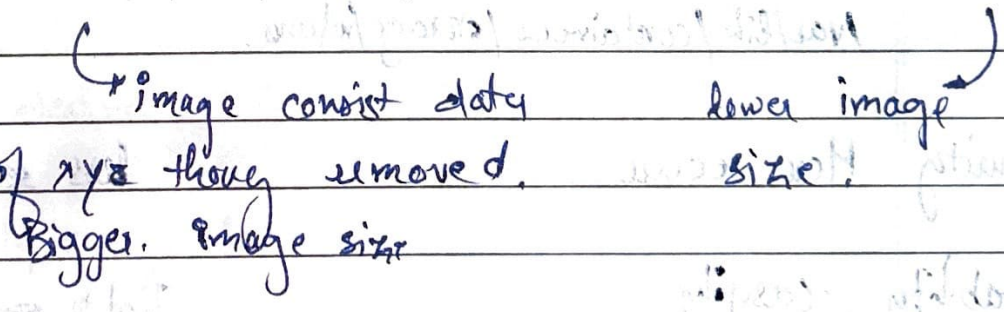
Copy-on-write file system.

While building a container image, each instruction that modifies the container file system creates a new read-only data layer.

Because you cannot modify data in an existing layer, each layer contains a set of changes, or 'diffs' from the previous layer.

Podman image layers can be inspected by
podman image tree image-name

From layer.
install xyz
remove xyz.
rm -rf /var/cache/yum



PAGE NO.
DATE :

Files in container image layers could be made available to read/write by using 'union file system.'

Step 1
Podman locate the requested file in the highest layer and copies it to the runtime layer.

Step-2
As file is available for r/w now containerized process can modify the file.

Developers often write applications that require storing data persistently.

	Volumes	Bind mounts
Managed by:	Podman	Host system
location.	Podman's storage location /var/lib/containers/storage/volumes	Anywhere on the host
Security	More secure	less secure.
Portability	easily	tied to specific host

Use case : Persisting container data Direct access to host files & dir.

Creation : # podman volume commands by reference

Use of volumes :
1) Persistence
2) Use of host file system (instead of COW)
3) Ease of sharing
4) Mounts can be shared over network via NFS

→ Volumes are data mounts managed by Podman. Bind Mounts are data mounts managed by the user ←

Both volume & bind mounts use syntax :
--volume / -v /path/onHost:/path/cont:OPTIONS

Alternatively :
--mount type=TYPE, source=/path/on/host, destination=/path/in/container.

type → bind → for bind mounts
volume → for volume mounts
tmpfs → ephemeral mount.

PAGE NO.
DATE :

podman run -p 8080:8080 --volume /www/:/var/www/html:z image-name

Bind mount Host → Bind mount Container.
read only option.

While using bind mounts, to access files across: selinux context should be set same.

podman run -v /www/:/var/www/html image.

→ /var/www/html has insufficient permissions to get accessed by httpd.

Podman 'unshare' command executes provids linux commands in a new namespace such as the one podman creates for the container.

podman unshare ls -l /www/

→ lists long inside-www dir, and shows permissions.

podman unshare command

→ container must have 'container_file_t' SELinux type to have access to the bind mount.

'podman volume' comands manages volume.

podman volume create volume_1
→ creates a volume by name 'volume_1'

podman volume inspect volume_1
→ inspects the volume 'volume_1'.

For rootless, podman stores volume in $HOME/.local/share/containers/storage/volumes/ directory.

→ As podman manages the volume, you do no need to configure SELinux permissions.

podman volume import 'volume_1' tar_file.
→ imports data from a tar archive into existing podman volume.

podman volume export 'volume_1' --output archive_file_name.
→ exports data from volume and save it as tar archive in local machine.

PAGE NO.
DATE :

- tmpfs mount

some app" cannot use cow file system in specific directory, nor can use persistence volume.
for such case tmpfs mount is used. (it is ephermal)

podman run --mount type=tmpfs, -tmpfs-size=512M, destination=/var/lib/pgsql/data image-name

Database Containers = Stateful containers.
(persist data)

Stateful containers present the following challenges;
1) Portability
2) Scalability
3) Availability

Good practices for Database Containers.
1) Use the volume instructions in Containerfiles
2) Mount the data directory to a named volume
3) Create a database network.

Importing Database Data

1) Database Containers with Data-loading features.

2) Load Data with a database client.