PAGE NO.

DATE : 01/10/24

Chapter number : 03

Container Images

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

Images can be stored in image registries.

ex: Quay.io
Red Hat Registry
Docker Hub
Amazon ECR

- registry.access.redhat.com
  No authorisation required.
- registry.redhat.io
  Requires authorisation
- https://catalog.redhat.com/software/containers/explore.

Universal base image:

A base image to create other images that is based on RHEL9

While pulling an image you can specify following information.
- Registry URL
- User or organization.
- Image repository
- Image tag

If URL is not provided then podman uses /etc/containers/registries.conf file to search other container registries in order of preferance

To block a registry.
```
[[registry]]
location = "docker.io"
blocked = true
```

Redhat recommends ~~it~~ to use ~~FQDN~~ Fully Qualified Container ~~Name~~ Image Name to avoid duplication of container image through multiple container registry.

PAGE NO.

DATE :

Skopeo is a comand-line tool for working with container images for;

1) Inspect remote container images
2) Copy a cont. image between registries.
3) Sign an image with OpenPGP keys.
4) Convert image format, (ex. from Docker to the OCI format)

skopeo inspect image-name.
→ Shows metadata of image.

skopeo copy registry1:image registry2.
→ copies image between registry.

skopeo copy reg:image dir:location.
→ changes the transport format to download into local directory.

podman login registry-name
→ Registers into registry and now you can execute 'podman pull imag' comand without any error.
[illegible]

Podman stores credential in base64 format and in;
${XDG_RUNTIME_DIR}/container/auth.json

${XDG-RUNTIME_DIR} = Refers to directory specific to the current user.

```
# cat ${XDG_RUNTIME_DIR}/container/
  auth}.json.
```

→ Copy auth output

```
# echo -n {auth output} | base64 -d.
```

RHOCP ; OC ; podman login with specific user.
↳ Need to read about this.

PAGE NO.

DATE :

- podman image --help

→ * Comands related to podman image.

Semantic versioning (SemVer) is a versioning scheme for software that aims to convey meaning about the underlying changes with each release. It typically uses a three-part version number:

1) Major:
Increaments when you make incompatible API changes.

2) Minor:
Increaments when you add functionality in a backward compatible manner.

3) PATCH:
Increaments when you make backward-compatible bug fixes.

An image tag is a string that you specify after the image name. The same image can have multiple tags.

If a tage is not specified in a Podman command, Podman uses the latest tag by default.

To create additional tags for local images :
# podman image tag LOCAL_IMAGE : TAG
LOCAL_IMAGE : NEW_TAG

To search image in registries.
# podman search image_name

podman image pull image-name
podman pull image_name
→ Downloads the image in host machine.
image is stored in, ~ (for non root user)
~/.local/share/containers directory.

podman image ls
→ lists all images.

For root user. the image is stored in
/var/lib/containers directory. and
'# podman images ls' will show
images of root only when command
is run by root

PAGE NO.

DATE :

podman build --file Containerfile \
--tag quay.io/image.

→ To build an image from Container file.

podman push quay.io/~~image~~ QUAY-User/image.
→ To push image to quay registry

podman image inspect image_name.
→ Useful information of image available locally, which consist few of the following points mentioned.
- Default user of image.
- Exposed ports
- Environmental variables
- Entrypoint
  Command that runs when container starts
- Cmd

Comand that the container-entrypoint script runs.
- Working Directory.

Working directory for the comands in ~~the~~ image.
- labels : extra metadata
- Architecture :

The architecture where this image can be used.

podman image inspect image_name --format = "{{.key. sub key}}"

podman image rm image-name
podman image rmi image-name
→ Removes image.
If the image is being used by container image will not be removed. first we will have to stop the container ; remove it and then remove the ~~con~~ image.

podman image rm/rmi image_name -f/--force
→ Removes image forcefully even if container is running.

podman rmi --all | -f / --force
podman image rm --all
→ Removes all images | forcefully

Dangling images : Images without tags that are not referenced by other images. 'podman image prune' is used to delete such images from the local system.
# podman image prune.
# podman image prune -a | all
# podman image prune -af | all without interactive