RHCEv8 Online Class 13062021 10:00pm
RHCSA-sysadmin
- Running Containers
- Controlling Boot Process

Running Containers

Introducing Containers

Introducing Container Technology

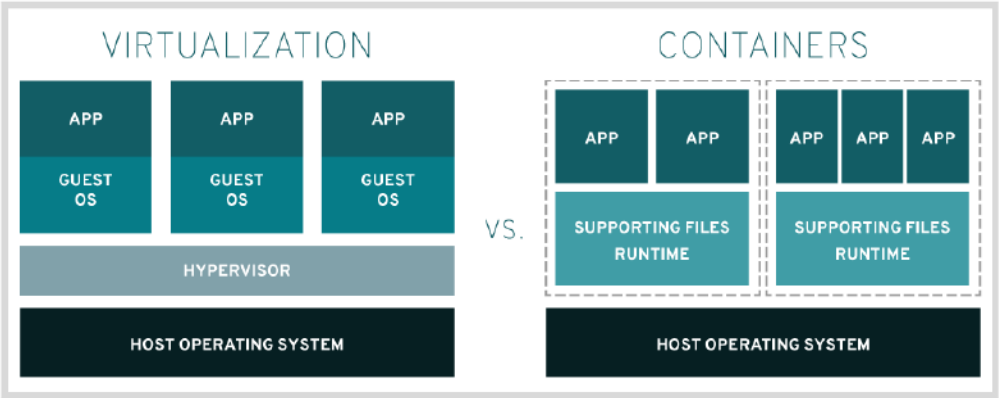
Software applications typically depend on other libraries, configuration files, or services provided by their runtime environment.

Traditionally:

-The runtime environment for a software application is installed in an operating system running on a physical host or virtual machine.
-Any application dependencies are installed along with that operating system on the host.

Container:

-A container is a set of one or more processes that are isolated from the rest of the system.
-A container, it is labeled, loaded, unloaded, and transported from one location to another as a single box.
-The container's contents are isolated from the contents of other containers so that they do not affect each other.



Designing Container-based Architectures

Managing Containers with Podman

A good way to start learning about containers is to work with individual containers on a single server acting as a container host.
RHEL supports:

- podman which directly manages containers and container images.
- skopeo which you can use to inspect, copy, delete, and sign images.
- buildah which you can use to create new container images.

what is Image:

Applications come to images. likes apache image, rsyslog image, nginx image, redis image, ….

what is registry:

location to save, store and reuse container images, name is registry. it comes to:

public:

accessible by any one, with or without login to repository
https://hub.docker.com/
https://catalog.redhat.com/software/containers/explore
https://registry.access.redhat.com
https://registry.redhat.io

private:

won't accessible by any one. login to repository is mandatory.

Running a Basic Container

Installing Container Management Tools

-install through official web address:
https://podman.io/getting-started/installation

-Install through Linux:

```
# podman version
# yum module install container-tools -y
# podman version
```

config registry addresses:

```
# vim /etc/containers/registries.conf
[registries.search]
registries = ['registry.access.redhat.com', 'registry.redhat.io', 'docker.io']
```

Practice on application in form of container

ex:

run nginx application

login to specific registry

```
# podman login registry.redhat.io
Username: naghval
Password: *******
Login Succeeded!
```

logout from specific registry

```
# podman logout registry.redhat.io
Removed login credentials for registry.redhat.io
```

search nginx image in to registries

```
# podman search nginx
registry.access.redhat.com/rhscl/nginx-112-rhel7
```

explain image sections:

`registry.access.redhat.com/rhscl/nginx-112-rhel7:latest`

`registry.access.redhat.com` ->registry name

`rhscl` ->image owner/creature

`nginx-112-rhel7` ->image name

`:latest` ->version tag

Inspect image before download

```
# skopeo inspect docker://registry.access.redhat.com/rhscl/nginx-112-rhel7
```

download/pull nginx image from selected registry

```
# podman pull registry.access.redhat.com/rhscl/nginx-112-rhel7
```

list of images

```
# podman images
# podman image list
registry.access.redhat.com/rhscl/httpd-24-rhel7   latest  47f6a8dbde38  8 days ago     329 MB
registry.access.redhat.com/rhscl/nginx-112-rhel7  latest  fc131486cd08  20 months ago  316 MB
```

Inspect image before download

```
# podman inspect registry.access.redhat.com/rhscl/nginx-112-rhel7
```

delete images

```
# podman rmi registry.access.redhat.com/rhscl/nginx-112-rhel7
```

Run the nginx app in form of container

foreground

```
# podman run registry.access.redhat.com/rhscl/nginx-112-rhel7
# podman run registry.access.redhat.com/rhscl/httpd-24-rhel7
```

background

```
# podman run -d registry.access.redhat.com/rhscl/httpd-24-rhel7
```

-d, --detach->run container in bg

Operate containers

```
# podman ps              ->list of running containers
# podman ps -a           ->list of total containers
# podman container list
# podman container stop 7b1f322b9df8
# podman container start 7b1f322b9df8
# podman container restart 7b1f322b9df8
```

delete container

```
#podman container stop 7b1f322b9df8
#podman container rm 7b1f322b9df8
or
# podman container rm 7b1f322b9df8 -f
```

run container with specific name

```
# podman run -d --name Apache  registry.access.redhat.com/rhscl/httpd-24-rhel7
```

-n --name ->container name

get inside container

```
# podman exec -it Apache /bin/bash
```

-i --interactive

-t --tty,

```
# podman exec -it Apache hostname
# podman exec -it Apache ls /
```

Performing Advanced Container Management

1-Mapping Container Host Ports to the Container

map a network port on the container host to a port in the container, network traffic sent to the host network port is received by the container.

ex:

```
# podman images
# podman inspect registry.access.redhat.com/rhscl/httpd-24-rhel7
"ExposedPorts": {
        "8080/tcp": {},
        "8443/tcp": {}
# podman run -d --name Apache  registry.access.redhat.com/rhscl/httpd-24-rhel7
# podman exec -it Apache /bin/bash
bash-4.2$ echo "hello" >/var/www/html/index.html
bash-4.2$ cat /var/www/html/index.html
hello
bash-4.2$ curl localhost
curl: (7) Failed connect to localhost:80; Connection refused
bash-4.2$ curl localhost:8080
hello
bash-4.2$ exit
exit
# curl localhost:8080
curl: (7) Failed to connect to localhost port 8080: Connection refused
```

solution

```
# podman run -d --name Apache  -p <container-host port>:<container image port> registry.access.redhat.com/rhscl/httpd-24-rhel7
# podman run -d --name Apache  -p 9000:8080 registry.access.redhat.com/rhscl/httpd-24-rhel7
```

-p ->--port

```
# podman port -a
```

f2e0b2cbd3cc 8080/tcp -> 0.0.0.0:9000

```
# podman exec -it Apache /bin/bash
bash-4.2$ echo "Hello" >/var/www/html/index.html
bash-4.2$ curl localhost:8080
Hello
bash-4.2$ exit
exit
# curl localhost:9000
```

Hello

open browser and check access

http://172.25.250.10:9000/

2-Attaching Persistent Storage to a Container

Preparing Permanent Storage Locations

Storage in the container is ephemeral, meaning that its contents are lost after you remove the container.

Preparing the Host Directory

Directory configuration involves:

- Configuring the ownership and permissions of the directory.
- Setting the appropriate SELinux context.

ex:

```
# mkdir html
# cd html/
# echo "Hello Class!" >index.html
# ls html
index.html
# tree
└── html
    └── index.html
# podman run -d --name Apache  -p 9000:8080 -v /root/html:/var/www/html:Z registry.access.redhat.com/rhscl/httpd-24-rhel7
```

-v --volume

--volume host_dir:container_dir:Z

Managing Containers as Services

Starting Containers Automatically with the Server

1-Managing Containers Running as Regular User with Systemd

steps:

```
# useradd user1
# passwd user1
# ssh user1@localhost
[user1@servera ~]$
$ pwd
/home/user1
$ mkdir -p .config/containers
$ cp /etc/containers/registries.conf .config/containers/
$ chmod 664 .config/containers/registries.conf
$ ls .config/containers/
registries.conf
customize user1 registries.conf file:
$ vim .config/containers/registries.conf
$ podman search rsyslog
$ podman pull registry.access.redhat.com/rhel7/rsyslog
$ podman images
$ podman run -d --name rsyslog registry.access.redhat.com/rhel7/rsyslog
$ podman container list
$ podman ps
$ podman ps -a
$ mkdir -p .config/systemd/user
$ cd .config/systemd/user/
$ podman generate systemd --name rsyslog --new --files
$ ls
container-rsyslog.service
$ systemctl --user daemon-reload
$ systemctl --user enable container-rsyslog.service
$ systemctl --user start container-rsyslog.service
$ systemctl --user status container-rsyslog.service
$ loginctl show-user user1
Linger=no
$ loginctl enable-linger user1
$ loginctl show-user user1
Linger=yes
```

-verify

```
$ podman ps
$ podman container list
$ systemctl --user stop container-rsyslog.service
$ podman container list
$ systemctl --user start container-rsyslog.service
$ podman container list
```

2-Managing Containers Running as Root with Systemd

```
# podman container list
# cd /etc/systemd/system/
# podman generate systemd --name Apache --files --new
/etc/systemd/system/container-Apache.service
# ls
container-Apache.service
# systemctl enable container-Apache.service
# systemctl start container-Apache.service
$ loginctl show-user root
Linger=no
$ loginctl enable-linger root
$ loginctl show-user root
Linger=yes
```

DESCRIBING THE RHEL8 BOOT PROCESS

Controlling Boot Process

1-power on
2-post
3-bios
4-bootloader
5-kernel
6-systemd
7-target

1-Power-On

2-Power-On Self-Test POST

major devices will check
- 1-main board
- 2-cpu
- 3-ram
- 4-vga

3-Basic Input-Output System BIOS

-MBR

first sector of first primary bootable hard disk. 512byes in to 446(bootloader info) 64(partition-table info) 2(err check) ->legacy

-GPT

works with UEFI firmware ->UEFI

4-bootloader

bootloader type:

linux loader-lilo	till rhel6
grand unified bootloader-grubv1	rhel6
grand unified bootloader-grubv2	rhel7 onwards

https://www.gnu.org/software/grub/
https://www.dedoimedo.com/computers/grub-2.html

How to create bootloader file?
grub2-mkconfig
main grubv2 config file
vim /boot/grub2/grub.cfg original file
or
ll /etc/grub2.cfg soft-link
instead above file, modify this one:
# vim /etc/default/grub
GRUB_TIMEOUT=10
:wq!
update existing bootloader file:
grub2-mkconfig >/boot/grub2/grub.cfg
protect grub bootloader
set
ll /boot/grub2/
grub2-set-password
Enter password: redhat
Confirm password: redhat
ll /boot/grub2/
user.cfg
cat /boot/grub2/user.cfg
update existing bootloader file:
grub2-mkconfig >/boot/grub2/grub.cfg
remove
rm -rf /boot/grub2/user.cfg
grub2-mkconfig >/boot/grub2/grub.cfg

bootloader responsibility on Linux boot process

a-vmlinuz

search and find out compressed kernel executable image file from /boot decompress and load it in to memory

b-initramfs

InitialRamFileSyetm-initramfs contains information about block-devices likes IDE, SAS, SSD, iscsi
find it and load in to TemporaryMemoryBasedFileSytem-tmpfs

c-initrd

InitialRamDisk-initrd loading temporary root file system in to memory before load real root file system.

5-kernel
kernel is a core of Linux. its first program to load on system startup
hostnamectl
Kernel: Linux 4.18.0-193.el8.x86_64
uname -r
4.18.0-193.el8.x86_64

6-systemd
its first process on linux. first systemd start then other background process will start.
till rhel6 the first process was **initd**
rhel7 onwards first process is **systemd**
pidof systemd
1918 1913 1337 1234 **1**

7-target
till rhel6 ->runlevel
rhel7 onwards ->target
current target:
systemctl get-default
multi-user.target

main targets:
1-emergency.target
2-rescue.target
3-graphical.target ->gui
4-multi-user.target ->cli

change target:
systemctl set-default <new target>
systemctl set-default graphical.target
reboot

change target without reboot:
systemctl isolate emergency.target
change target at boot time:
1-reboot host
2-press **e** when watch kernel lines
3-findout **linux** line and press **end** key on keyboard and type systemd.unit=emergency.target then press **Ctrl+x** to start.

change hostname
hostnamectl set-hostname <new hostname>

crack root password
1-reboot host
2-press **e** when watch kernel lines
3-findout **linux** line and press **end** key on keyboard and type rd.break console=tty0 then press **Ctrl+x** to start.
4-
switch_root:/# mount -o remount,rw /sysroot
switch_root:/# chroot /sysroot
sh-4.4# passwd root
new password: coss@2021
re-type new password: coss@2021
password successful updated
sh-4.4# touch /.autorelabel ->SELinux relabeling
sh-4.4# exit
exit
switch_root:/# exit
logout

after finish the process Linux comes up with new password.