fedora

Containers Lab

Alex Callejas
Senior Technical Support Engineer
November 2019

Red Hat

Alex Callejas
Senior Technical Support Engineer @Red Hat

@dark_axl

/rootzilopochtli

www.rootzilopochtli.com

Geek by nature, Linux by choice, Fedora of course!

¿Porqué un laboratorio?

It doesn't matter how beautiful your theory is, it doesn't matter how smart you are. If it doesn't agree with experiment, it's wrong. In that simple statement is the key to science.

Richard Feynman

Laboratorio de Pruebas

Mi configuración

Lenovo Thinkpad T440s
Intel Core i7-4600U CPU @ 2.10GHz + 8G Mem

- **Storage**
  /var/lib/libvirt/images (4.2G) → / fs 50G
- **KVM packages**
  - qemu-kvm
  - virt-manager
  - virt-viewer
  - libguestfs-tools
  - virt-install
  - genisoimage

Cloud Images

KVM y QEMU

QEMU soporta varios tipos de imágenes. El tipo "nativo" y más flexible es *qcow2*, que admite la copia en escritura, el cifrado, la compresión y los snapshots de VM.

La forma más sencilla de obtener una máquina virtual que funciona con **KVM** es descargar una imagen que alguien más ya haya creado:

- Fedora Cloud. Cloud Base Images [https://alt.fedoraproject.org/cloud/]
- Atomic Host [https://getfedora.org/en/atomic/download/]
- OpenStack: Get images [https://docs.openstack.org/image-guide/obtain-images.html]

Containers Lab

Creando VM's - Fedora

Configurar VM:

```
$ sudo virt-customize -a /var/lib/libvirt/images/vmtest01.qcow2 \
  --hostname vmtest01.mx.redhat.lab  --root-password password:redhat \
  --ssh-inject 'root:file:labkey.pub' --uninstall cloud-init --selinux-relabel
```

Importar VM:

```
$ sudo virt-install --name vmtest01 --memory 1024  \
  --vcpus 1 --disk /var/lib/libvirt/images/vmtest01.qcow2  \
  --import --os-variant fedora29 --noautoconsole
```

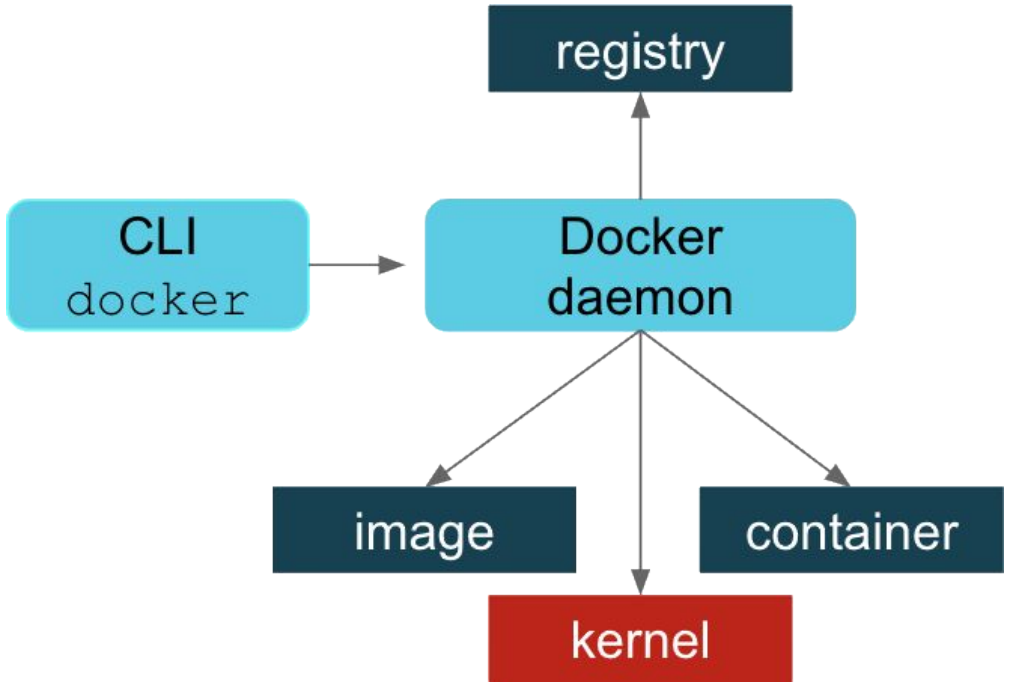
Source:
Modifying the Red Hat Enterprise Linux OpenStack Platform Overcloud Image with virt-customize [https://access.redhat.com/articles/1556833]
Creating Guests with virt-install
[https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/virtualization_deployment_and_administration_guide/sect-guest_virtual_machine_installation_overview-creating_guests_with_virt_install]

Hacerte invencible significa

conocerte a ti mismo

Sun Tzu

¿Cómo funciona **Docker**?



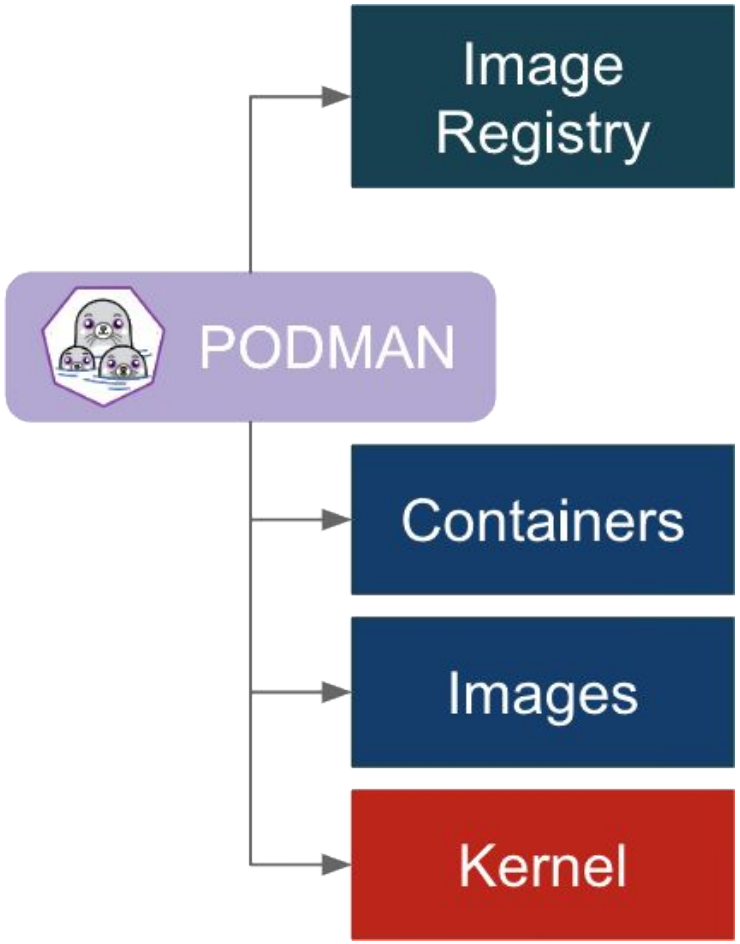
Docker proporciona toda la funcionalidad necesaria para:

- Pull & push imágenes de un registro de imágenes
- Administrar contenedores locales:
  - Copy, add layers, commit & remove
- Pedir al kernel que ejecute un contenedor con el name-space y cgroup correctos, etc.

Esencialmente, el demonio **Docker** hace todo el trabajo con registros, imágenes, contenedores y el kernel. La línea de comandos (CLI) de **Docker** le pide al demonio que haga esto en su nombre.

Podman

El enfoque de **podman** es simplemente interactuar directamente con el registro de imágenes, con el contenedor, con el almacenamiento de imágenes, y con el kernel de Linux a través del proceso de ejecución del contenedor (*runC*).



Podman

Instalar podman:

```
# dnf -y install podman buildah skopeo
```

Proporciona una sintaxis "tipo **Docker**" para trabajar con contenedores

```
# podman pull registry.fedoraproject.org/f29/httpd
```

```
# podman images
```

```
# podman inspect httpd
```

```
# podman run httpd
```

```
# podman run --name myhttpservice -d httpd
```

Podman

```
# podman inspect myhttpservice | grep -i ipaddr
```

```
# podman inspect myhttpservice | grep expose-services
```

```
# curl 10.88.0.3:8080
```

```
# podman exec -ti c94e745f6414 /bin/bash
```

```
bash-4.4$ echo "MySecretData" > my.data
```

Podman

Construyendo contenedores

Hola Mundo

```
# echo "hello world" > $(buildah mount $(buildah from registry.fedoraproject.org/fedora-minimal))/etc/hello.txt
```

Revisamos la creación de la imagen base

```
# buildah containers
```

Hacemos commit a la imagen local, eliminamos la imagen base y ejecutamos el contenedor

```
# buildah commit fedora-minimal-working-container fedora-hello
```

```
# buildah images
```

```
# buildah delete fedora-minimal-working-container
```

```
# podman run -ti localhost/fedora-hello:latest cat /etc/hello.txt
hello world
```

Podman

Construyendo contenedores

Dockerfile

```
# Base on the Fedora
FROM registry.fedoraproject.org/fedora
MAINTAINER  darkaxl017 email dark.axl@gmail.com # not a real email

# Install httpd on image
RUN echo "Installing httpd"; dnf -y install httpd

# Expose the default httpd port 80
EXPOSE 80

# Run the httpd
CMD ["/usr/sbin/httpd", "-DFOREGROUND"]
```

```
# buildah bud -f Dockerfile -t fedora-httpd .
```

```
# buildah run $(buildah from fedora-httpd) httpd -v
```

Podman

Construyendo contenedores

Ejecutamos el contenedor

```
# podman run -d fedora-httpd
```

Revisamos el proceso del contenedor

```
# podman ps
```

Revisamos los procesos

```
# ps auxf | tail -6
```

```
# curl localhost
curl: Failed to connect to localhost port 80: Connection refused
```

```
# podman logs 517495e24317
```

Podman

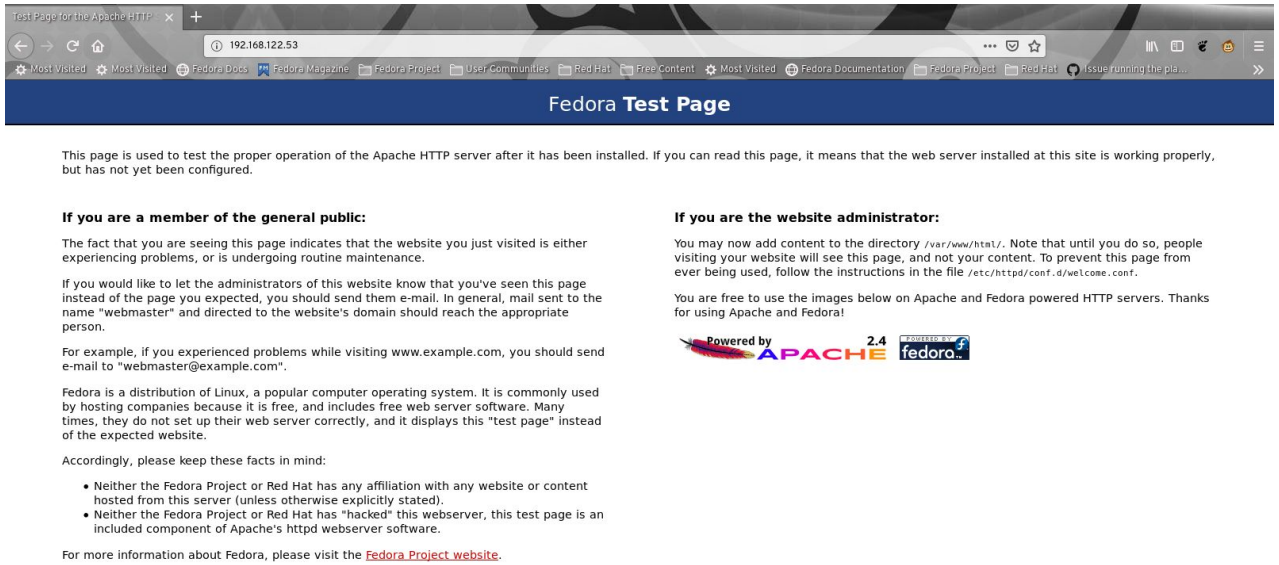
Construyendo contenedores

Detenemos el contenedor y ejecutamos el contenedor exponiendo el puerto

```
# podman run -d -p 80:80 fedora-httpd
```

Validamos el servicio

```
# curl localhost
```



Podman

Construyendo contenedores

Creamos directorio compartido para el contenedor

```
# mkdir -p /opt/var/www/html ; cd /opt/var/www/html
```

Creamos el contenido a compartir

```
# wget --page-requisites --convert-links https://registry.fedoraproject.org/
```

Ejecutamos el contenedor compartiendo puerto y directorio

```
# podman run -d --name myhttpservice -p 8080:8080 -v /opt/var/www/html:/var/www/html:Z registry.fedoraproject.org/f29/httpd
```

Healthcheck

```
# podman run -dt --name myhttpservice -p 8080:8080 -v /opt/var/www/html:/var/www/html:Z --healthcheck-command
'CMD-SHELL curl http://localhost:8080 || exit 1' --healthcheck-interval=0 registry.fedoraproject.org/f29/httpd
```



```
# podman healthcheck run myhttpservice
```

Podman

Construyendo servicios

```
/etc/systemd/system/myhttpservice.service
[Unit]
Description=Just a http service with Podman Container

[Service]
Type=simple
TimeoutStartSec=30s
ExecStartPre=-/usr/bin/podman rm "myhttpservice"

ExecStart=/usr/bin/podman run --name myhttpservice -p 8080:8080 -v /opt/var/www/html:/var/www/html:Z
--healthcheck-command 'CMD-SHELL curl http://localhost:8080 || exit 1'
--healthcheck-interval=0registry.fedoraproject.org/f29/httpd

ExecReload=-/usr/bin/podman stop "myhttpservice"
ExecReload=-/usr/bin/podman rm "myhttpservice"
ExecStop=-/usr/bin/podman stop "myhttpservice"
Restart=always
RestartSec=30

[Install]
WantedBy=multi-user.target
```

Podman

Construyendo servicios

Refrescamos systemd

```
# systemctl daemon-reload
```

Revisamos status del servicio

```
# systemctl status myhttpservice.service
```

Iniciamos el servicio

```
# systemctl start myhttpservice.service
```

```
# systemctl status myhttpservice.service
```

Referencias

Links y documentación

- Fedora Classroom: Containers 101 with Podman
- Getting Started with Buildah
- Managing containerized system services with Podman
- Podman: Managing pods and containers in a local container runtime
- Podman can now ease the transition to Kubernetes and CRI-O

- https://registry.fedoraproject.org/
- https://registry.centos.org/containers/

- https://podman.io/
- https://github.com/containers/buildah

- Daniel Walsh - @rhatdan

Fedora México

Únete!

- https://www.meetup.com/es-ES/Fedora-Mexico/
- https://t.me/fedoramexico
- https://fedoramx.fedorapeople.org/

Gracias!

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/redhatinc

twitter.com/RedHat

Red Hat