Fedora Containers Lab

Alex Callejas
Services Content Architect @ Red Hat

@dark_axl
github.com/AlexCallejas
rutil.io/social

¿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

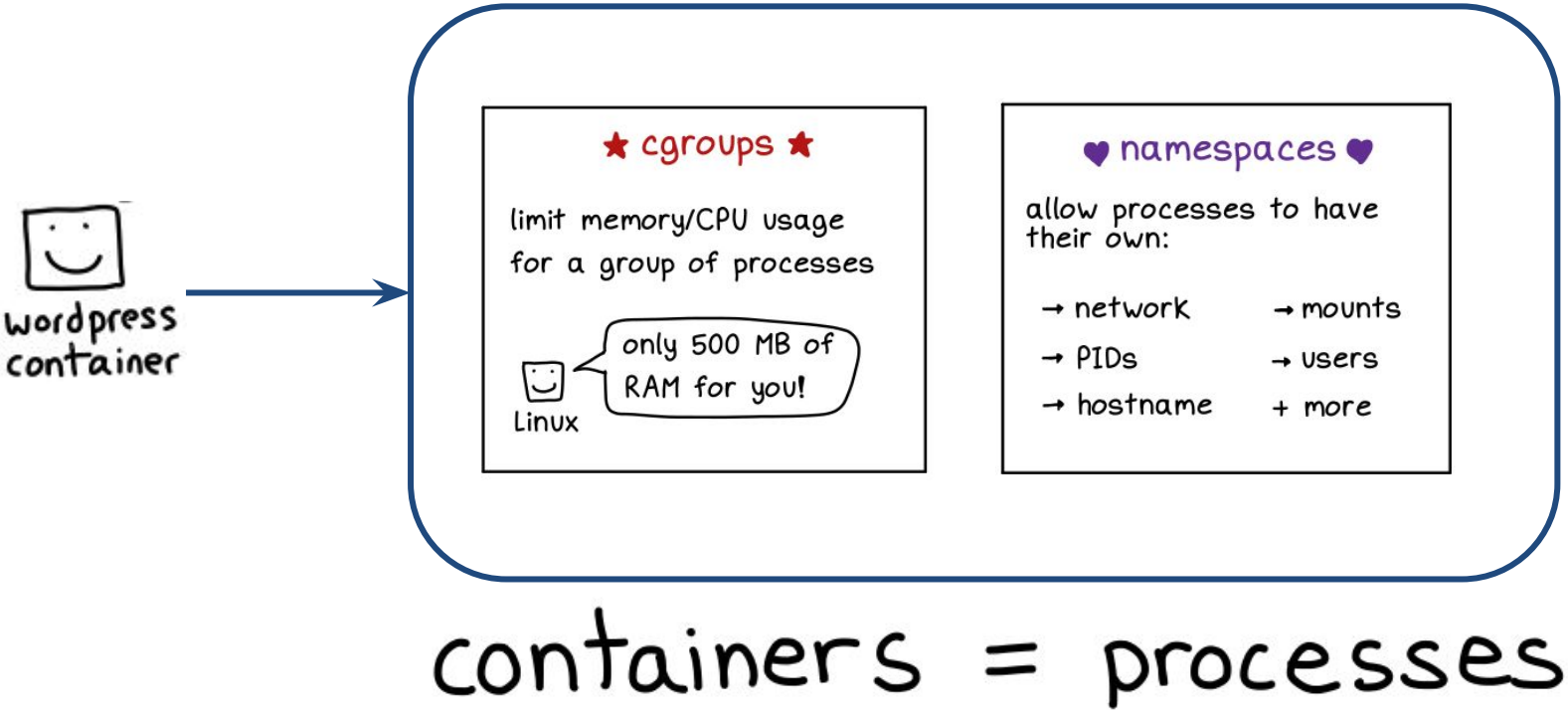
Lenovo ThinkPad P1 Gen 2

Intel(R) Core(TM) i7-9850H CPU @ 2.60GHz

- **Storage**
  `/var/lib/libvirt/images` → LVM 476G
- **Fedora release 35 (Thirty Five)**
  `5.17.6-200.fc35.x86_64`
- **Automation 101: The Guide**

Containers Lab

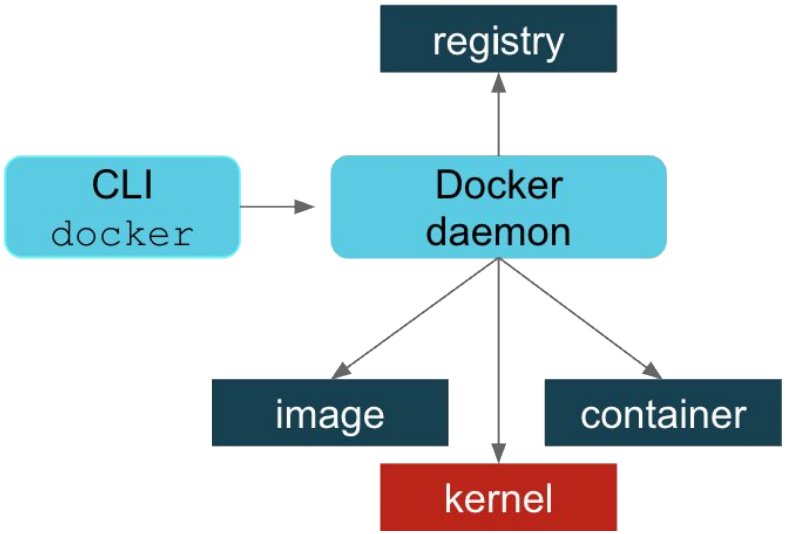
Anatomía

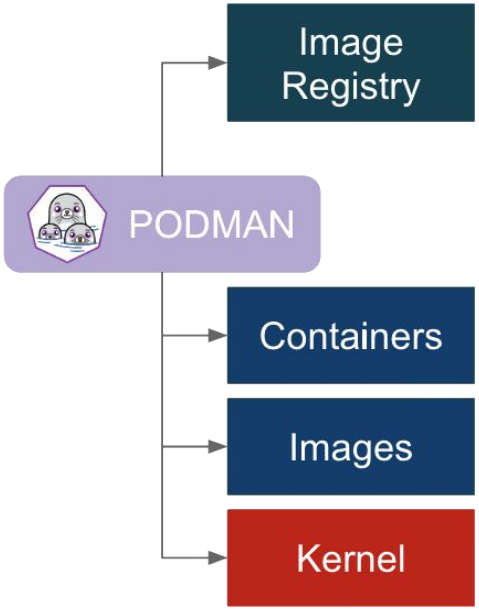


Fuente: What even is a container: namespaces and cgroups [https://jvns.ca/blog/2016/10/10/what-even-is-a-container/] **Imágenes:** @b0rk

Containers Lab

Docker vs Podman





Podman rootless containers

Lo que necesitas saber

Algunas consideraciones al ejecutar containers como un usuario *non-root*:

- Las imágenes se guardan en el home directory del usuario (`$HOME/.local/share/containers/storage/`) en lugar de `/var/lib/containers`.
- Al no tener privilegios de root:
  - No tiene capacidad para acceder a un puerto inferior a **1024**.
  - El almacenamiento debe estar en un sistema de archivos local.

Podman rootless containers

Primeros pasos

Checamos la configuración *rootless*

```
$ podman unshare cat /proc/self/uid_map
```

Creamos un contenedor con el usuario *non-root*

```
$ podman pull ubi8/ubi
```

```
$ podman inspect ubi8/ubi
```

← skopeo

```
$ podman run ubi8/ubi cat /etc/os-release
```

Podman rootless containers

Creando un servicio

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

```
$ podman images
```

```
$ podman inspect httpd
```

```
$ podman run httpd
```

```
$ podman run --name myapache -d httpd
```

Podman rootless containers

Creando un servicio

```
$ podman inspect myapache | grep IPAddress
```

```
$ podman inspect myapache | grep expose-services
```

```
$ podman logs myapache
```

Testing:

```
$ curl 10.0.2.100:8443
```

```
$ curl 10.0.2.100:8080
```

Eliminar container:

```
$ podman {stop|rm} myapache
```

Podman rootless containers

Creando un servicio

```
$ podman run --name myapache -d -p 8081:8080 httpd
```

Testing:

```
$ curl localhost:8081
```

Inmutabilidad:

```
$ podman exec -ti myapache /bin/bash
```

```
bash-4.4$ echo "Top Secret" > secret/my.data
```

Podman rootless containers

Creando un servicio

Almacenamiento persistente:

```
$ mkdir -p containers/myapache/var/www/html
```

```
$ echo "Hola Mundillo" > containers/myapache/var/www/html/index.html
```

```
$ podman run --name myapache -d -p 8081:8080 \
> -v ~/containers/myapache/var/www/html:/var/www/html:Z httpd
```

Testing:

```
$ curl localhost:8081
Hola Mundillo
```

Podman rootless containers

Creando un servicio

healthcheck:

```
$ podman run --name myapache -d -p 8081:8080 \
> -v ~/containers/myapache/var/www/html:/var/www/html:Z \
> --health-cmd="curl localhost:8081 || exit 1" --health-interval=0 \
> httpd
```

Testing:

```
$ podman healthcheck run myapache
```

Podman rootless containers

Creando un servicio

Configurando systemd:

```
# loginctl enable-linger alex
```

Creando systemd unit:

```
$ mkdir -p ~/.config/systemd/alex & cd ~/.config/systemd/alex
```

```
$ podman generate systemd --name myapache --files
```

```
$ systemctl --user daemon-reload
```

```
$ systemctl --user enable --now \
> ~/.config/systemd/alex/container-myapache.service
```

```
$ systemctl --user status container-myapache
```

Podman Monitoring

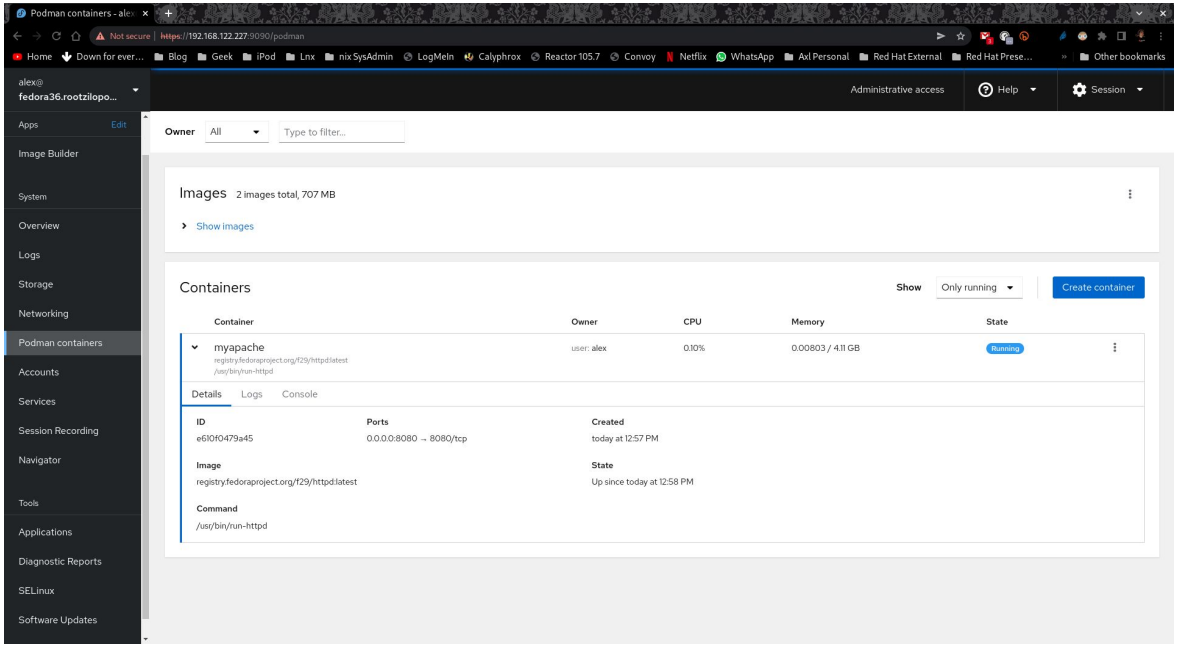
Configurando web console

Instalar y habilitar **cockpit**:

```
# dnf -y install cockpit cockpit-pcp cockpit-podman cockpit-selinux
```

```
# systemctl restart pmlogger
```

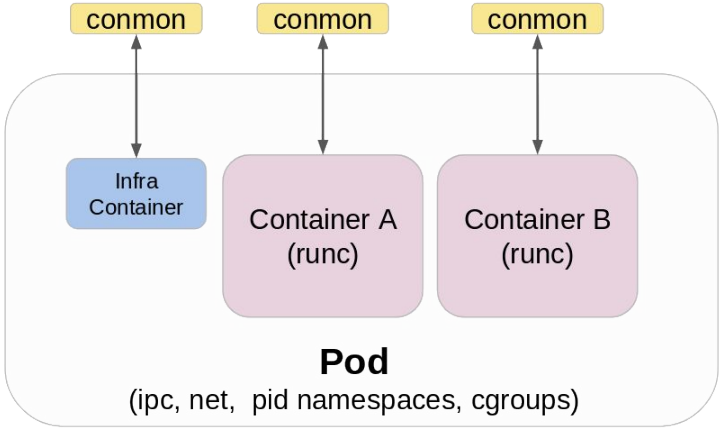
```
# systemctl enable --now cockpit.socket
```



Podman Pods

Lo que necesitas saber

> *Un **Pod** es un grupo de uno o más contenedores, con recursos de almacenamiento y red compartidos, además una especificación de cómo ejecutar dichos contenedores.*
>
> **- PODS - KUBERNETES DOCUMENTATION**



Podman Pod

Fuente: https://kubernetes.io/docs/concepts/workloads/pods/

Podman Pods

Ejemplo práctico: MariaDB Pod

Creamos el pod

```
$ podman run -dt -e MYSQL_ROOT_PASSWORD=x  --pod new:db registry.fedoraproject.org/f31/mariadb:latest
```

Revisamos el status del pod

```
$ podman pod ps
```

Agregamos un contenedor al pod para validar la DB

```
$ podman run -it --rm --privileged --pod db docker.io/library/alpine:latest /bin/sh
```

```
/ # apk add mariadb-client
```

```
/ # mysql -u root -P 3306 -h 127.0.0.1
```

Podman

Siguientes Pasos → Kubernetes

Creamos el contenedor y lo validamos

```
$ podman run -dt -p 8000:80 --name demo quay.io/libpod/alpine_nginx:latest
```

```
$ curl localhost:8000
podman rulez
```

Generamos un snapshot para crear el archivo YAML de kubernetes

```
$ podman generate kube demo > demo.yml
```

Lo transferimos para cargarlo en **kubernetes**

```
# oc create -f demo.yml -n myproject
```

```
# oc status --suggest
In project My Project (myproject) on server
https://minishift.rootzilopochtli.lab:8443

pod/demo runs quay.io/libpod/alpine_nginx:latest
```

okd

docs.okd.io/minishift/getting-started

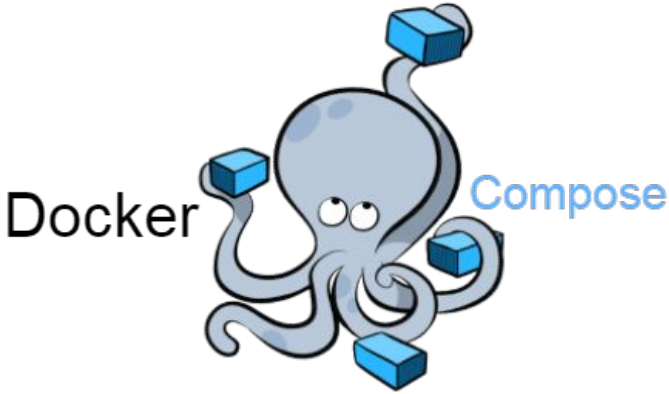
Podman Compose

Podman Compose es un proyecto cuyo objetivo es ser utilizado como una alternativa a **Docker Compose** sin necesidad de realizar ningún cambio en el archivo `docker-compose.yaml`.

La idea básica detrás de **Podman Compose** es que recoge los servicios definidos dentro del archivo `docker-compose.yaml` y crea un contenedor *rootless* para cada servicio.

Instalación:

```
# dnf install podman-compose
```



Docker Compose logo

Podman Compose

```
$ git clone https://github.com/rootzilopochtli/fedora-containers-lab-examples.git
```

```
$ cd fedora-containers-lab-examples/podman-compose/awx3/
```

```
$ podman-compose up -d
```

```
$ podman-compose ps
```

```
$ podman pod list
```

```
$ podman volume ls
```

Modificamos la contraseña de **admin**

```
$ podman exec -ti awx3_awx_web_1 /bin/bash
```

```
# awx-manage changepassword admin
```

```
$ podman-compose down
```

```
$ podman-compose ps
```

```
$ podman pod prune
```

```
$ podman volume prune
```

Referencias

Links y Documentación

- Podman basics: Resources for beginners and experts
- Podman can now ease the transition to Kubernetes and CRI-O

- Fedora Containers Lab Examples

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

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

- Daniel Walsh - @rhatdan

Gracias!

Únete a la conversación!

@fedoramexico