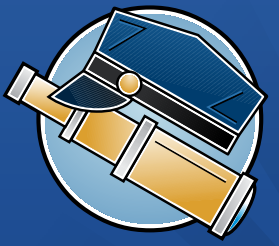
Podman, Buildah y Skopeo

podman

buildah



Hecho por: Francisco Javier Martín Núñez

Indice

Contenedores

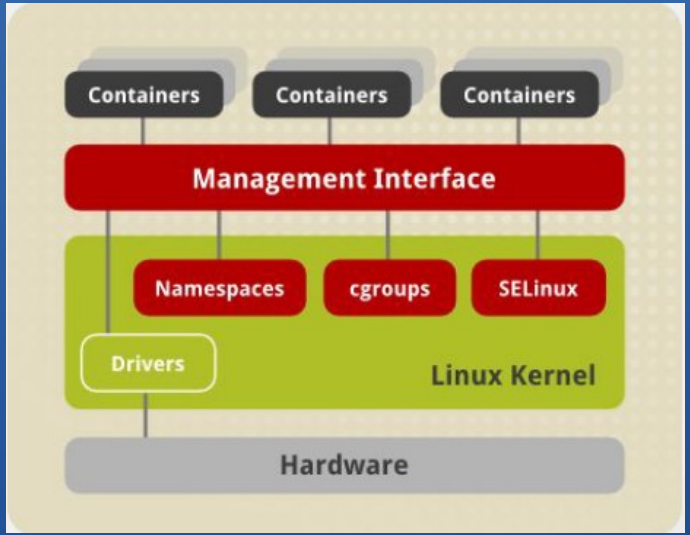
Antes de empezar, debemos de tener claro que son los contenedores cuando hablamos de software y cuales son sus características y funciones que le podemos dar.

Lo primero debemos saber que los contenedores son un conjunto de uno o más procesos, por lo general lo vamos a usar con un solo proceso, que se encuentran aislados del resto del sistema.

El kernel proporciona sus componentes principales por lo que depende del kernel donde se ejecute el contenedor:

- Namespaces: para asegurar el aislamiento de procesos, lo que permite que cada proceso solamente sea capaz de ver su propio sistema “virtual” (ficheros, procesos, interfaces de red, hostname)
- cgroups: para el control de los recursos del sistema, por el cual somos capaces de limitar los recursos que puede consumir (CPU, memoria, ancho de banda, etc)

La interfaz de administración interactúa con los componentes del kernel y proporciona herramientas para la construcción y gestión de contenedores.

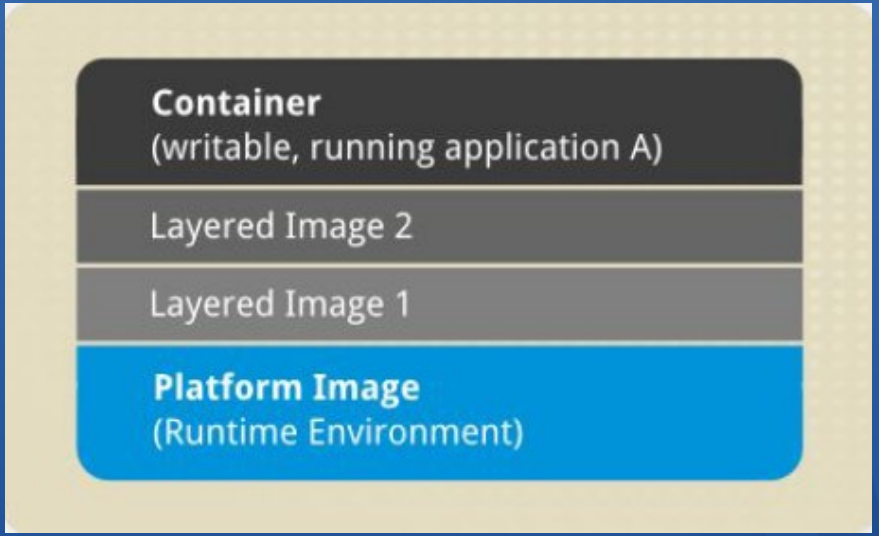


Contenedores

Todos los archivos que se necesitan para ejecutar un contenedor provienen de una imagen:

- Container: (en sentido estricto) Un componente activo en el que se ejecuta una aplicación. Cada contenedor se basa en una imagen que contiene los datos de configuración necesarios.
- Image: Un snapshot estático de la configuración del contenedor. La imagen es una capa read-only que nunca se modifica,
- Platform Image: Define el runtime, los paquetes y las utilidades necesarios para que se ejecute una containerized application.

Las imágenes del contenedor se almacenan en un registro de imágenes, que contiene todas sus versiones



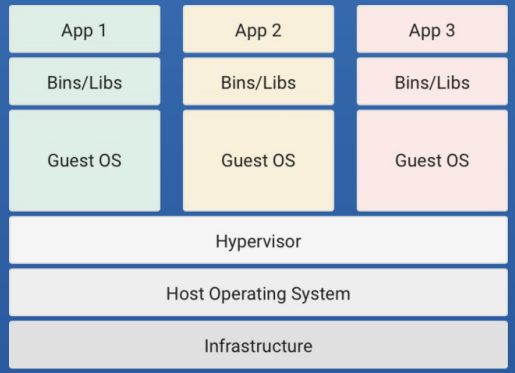
Contenedores vs Maquinas Virtuales

Gracias a la virtualización somos capaces, usando un mismo ordenador, de tener distintas máquinas virtuales con su propio sistema operativo invitado, Linux o Windows. Todo ello ejecutándose en un sistema operativo anfitrión y con acceso virtualizado al hardware.

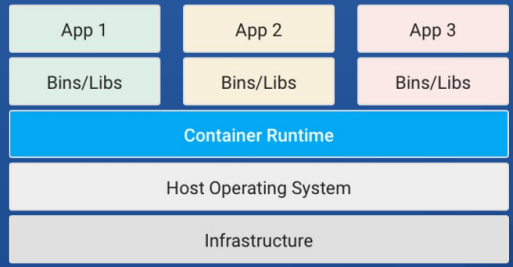
Al contrario, los contenedores se ejecutan sobre el mismo sistema operativo anfitrión de forma aislada, pero sin necesidad de un sistema operativo propio, ya que comparten el mismo Kernel, lo que los hace mucho más ligeros.

Habitualmente cada aplicación va en su propio contenedor totalmente aislado, mientras que en las VMs es habitual debido al dimensionamiento, tener varias aplicaciones en la misma máquina con sus propias dependencias, mucho peor para escalar

los contenedores se basan en dos mecanismos para aislar procesos en un mismo sistema operativo. El primero de ellos, se trata de los **namespace** que provee Linux, lo que permite que cada proceso solamente sea capaz de ver su propio sistema “virtual” (ficheros, procesos, interfaces de red, hostname, etc). El segundo concepto son los **CGroups**, por el cual somos capaces de limitar los recursos que puede consumir (CPU, memoria, ancho de banda, etc)



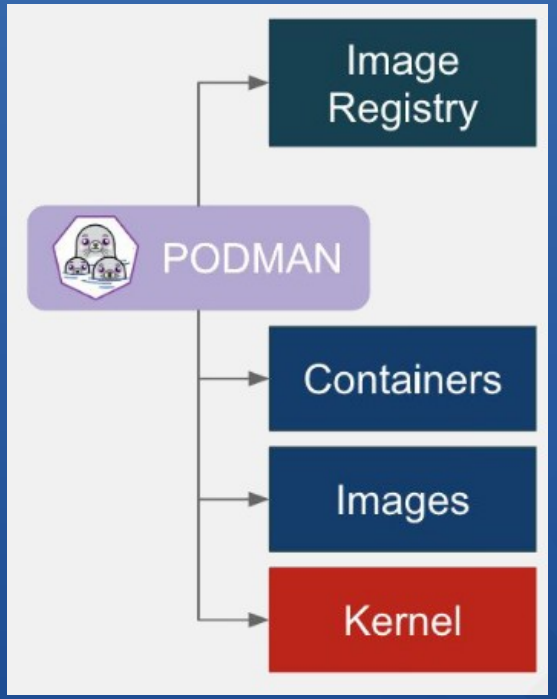
Virtual Machines



Containers

Podman

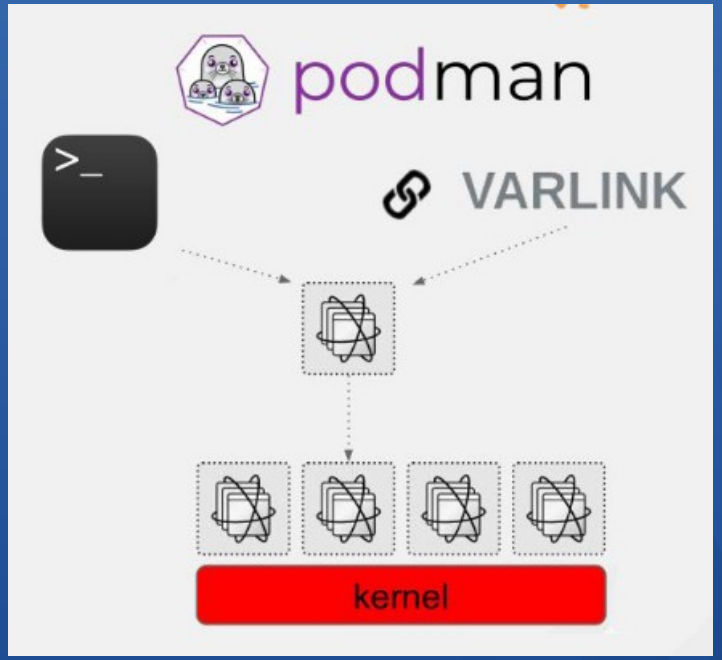
El enfoque de podman es simplemente interactuar directamente con el registro de imágenes, con el contenedor y el almacenamiento de imágenes, y con el kernel de Linux a través del proceso de ejecución (runC) del contenedor.



Podman

Un CLI/API sin daemon para ejecutar, administrar y depurar contenedores y pods OCI

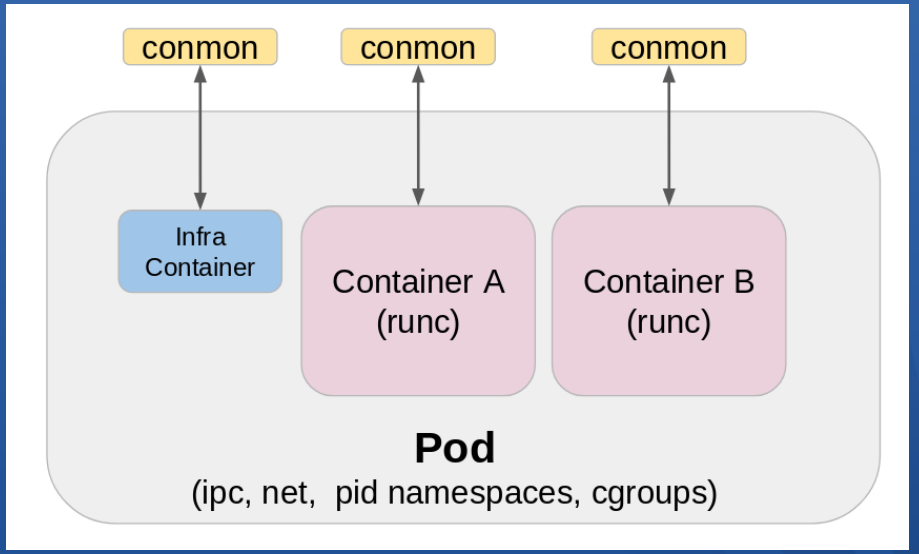
- Rápido y ligero ya que este no tiene un *daemon* que ejecutar
- runC compartido
- Proporciona una sintaxis "tipo docker" para trabajar con contenedores
- API de gestión remota a través de varlink
- Proporciona integración de sistemas y aislamiento avanzado de namespace



Pods en podman

El concepto de Pod fue introducido por Kubernetes: un grupo de uno o más contenedores implementados en un nodo único

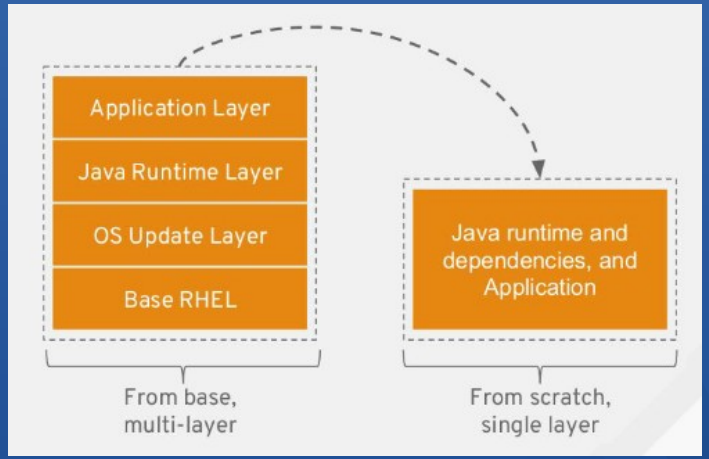
- Cada podman pod incluye un contenedor "infra"
  - Mantiene los namespaces asociados con el pod y permite a podman conectarse a los otros contenedores.
  - Se basa en la imagen k8s.gcr.io/pause.
  - Se le asignan los port bindings, cgroup-parent values, y kernel namespaces del pod.
  - Una vez que se crea el pod, estos atributos se asignan al contenedor "infra" y no se pueden cambiar.
- Cada contenedor tiene su propio monitor (conmon)
  - Monitorea el proceso primario del contenedor y guarda el exit code si se termina o muere el contenedor
  - Permite que podman se ejecute en modo detached (background)



Buildah

¿Por que usar buildah para construir imágenes?

- Crea imágenes compatibles con OCI
- No daemon - sin socket docker
- No requiere un contenedor en ejecución
- Puede utilizar las suscripciones de hosts y otros secretos
- Control preciso sobre los comandos y el contenido de la capa
- Una sola capa, desde cero, las imágenes se hacen fáciles y aseguran un manifiesto limitado
- Si es necesario, puede mantener el flujo de trabajo basado en Dockerfile



Skopeo

Una de las principales ventajas que tiene Skopeo es que puede ser ejecutado sin ser usuario root al igual que pasa con podman y tambien, como caracteristica principal es que no tiene ningun demonio ejecutandose para su funcionamiento.

Como caracteristica sobre la compatibilidad de Skopeo con imagenes, este es compatibles con imagenes de tipo OCI(Open Container Initiative) y con imagenes de docker v2.

- Copiar una imagen desde y hacia varios mecanismos de almacenamiento. Por ejemplo, podemos copiar imágenes de un registro a otro, sin necesidad de privilegios.
- Inspeccionar una imagen remota para que muestre sus propiedades, incluidas sus capas, sin que sea necesario que se descargue la imagen
- Eliminar una imagen de un registro que nosotros le indiquemos, ya sea privado o en docker hub.
- Sincronizar un repositorio de imágenes externo con un registro interno.
- Cuando lo requiera el repositorio, Skopeo puede pasar las credenciales y certificados apropiados para la autentificación.

Ejemplos

A continuación veremos los ejemplos propuestos.