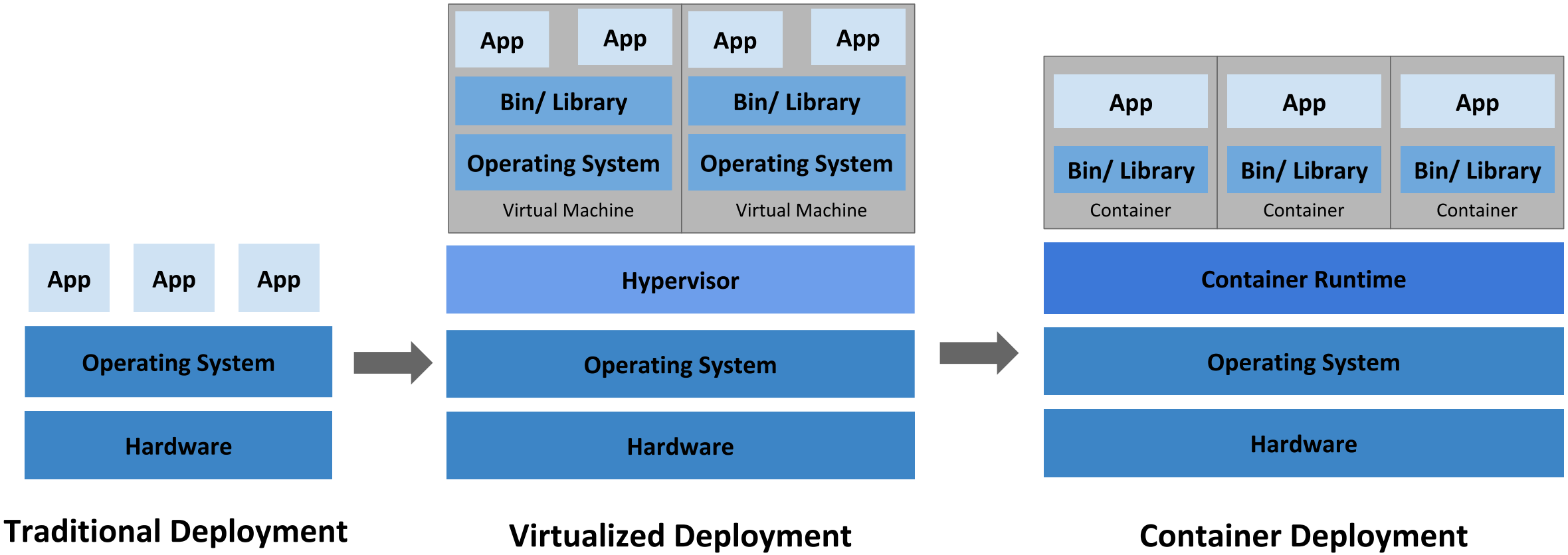
MÓDULO 2

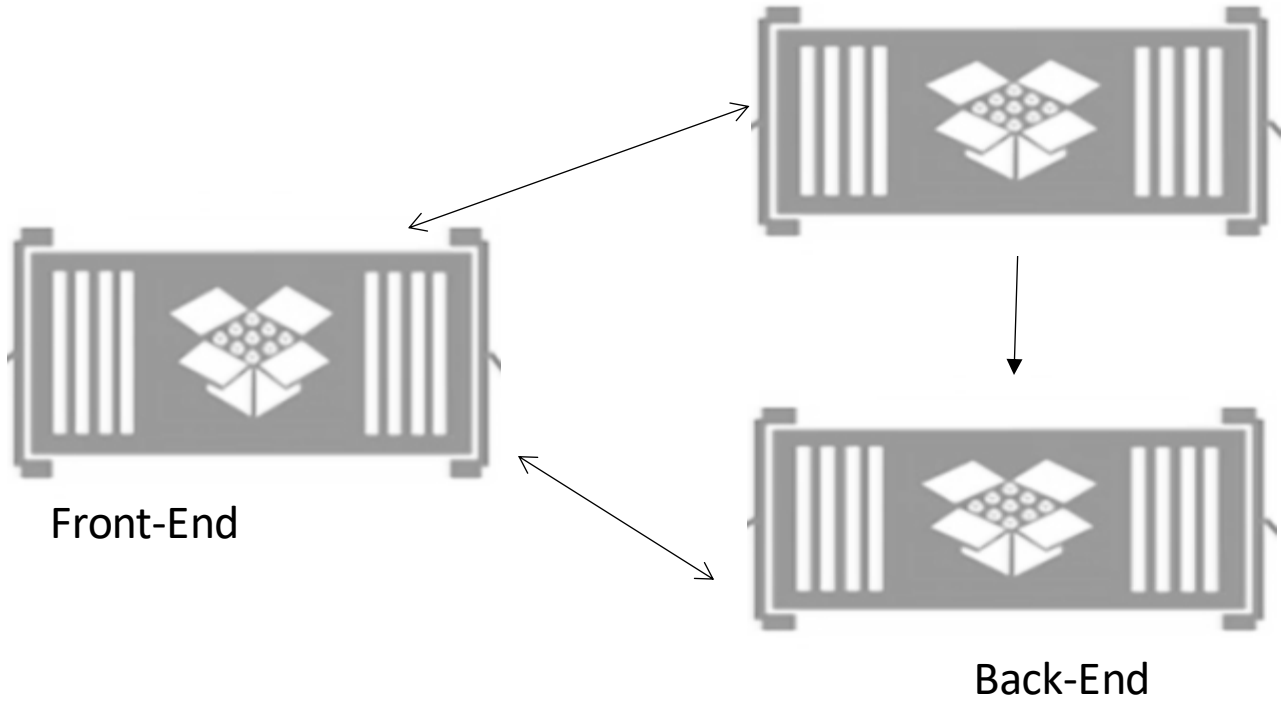
Descripción general de la tecnología de contenedores

¿POR QUÉ CONTENEDORES?



QUÉ ES UN CONTENEDOR

- Una unidad de despliegue
- Corre en un SO
- En SO Virtualizados
- Arranque rápido
- Portable

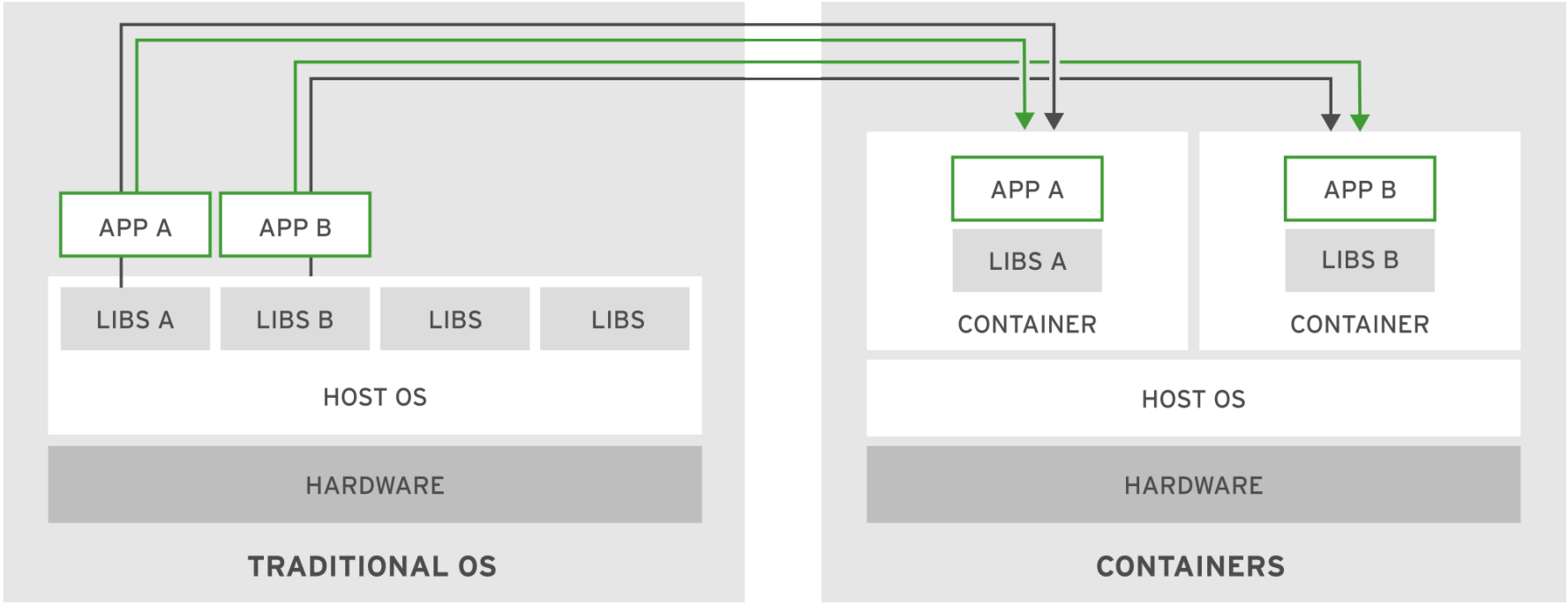


APLICACIONES EN CONTENEDORES

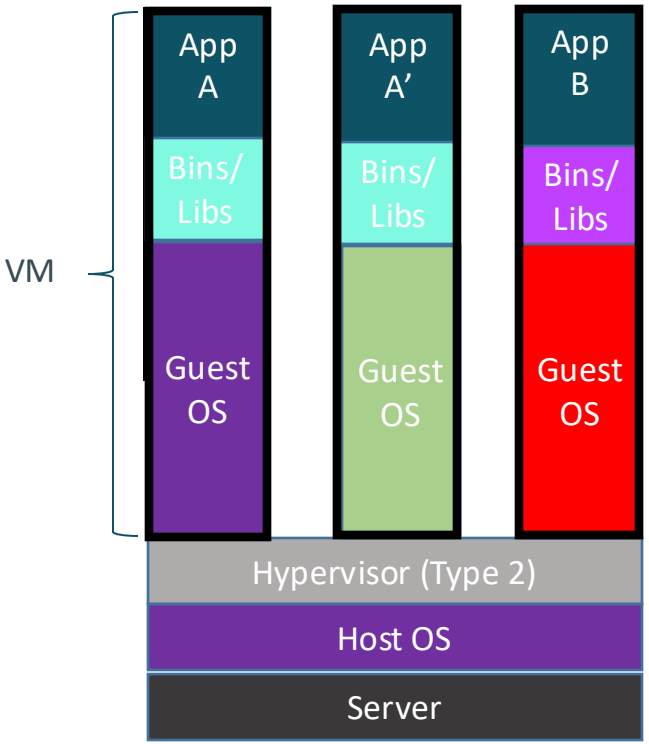
- Las aplicaciones de software suelen depender de otras librerías, archivos de configuración o servicios proporcionados por el entorno de tiempo de ejecución.
- El entorno de tiempo de ejecución tradicional para una aplicación de software es un host físico o una máquina virtual, y las dependencias de las aplicaciones se instalan como parte del host.
- Por ejemplo, considere una aplicación de Python que requiere acceso a una librería compartida común que implementa el protocolo TLS.
- Tradicionalmente, un administrador de sistema instala el paquete requerido que proporciona la librería compartida antes de instalar la aplicación de Python.
- El principal inconveniente de las aplicaciones de software implementadas tradicionalmente es que las dependencias de la aplicación están entremezcladas con el entorno de tiempo de ejecución. Una aplicación puede fallar cuando se aplican actualizaciones o parches al sistema operativo (SO) de base.

DIFERENCIAS ENTRE UN CONTENEDOR Y UN S.O.

Se describe la diferencia entre aplicaciones que se ejecutan como contenedores y aplicaciones que se ejecutan en el sistema operativo del host:

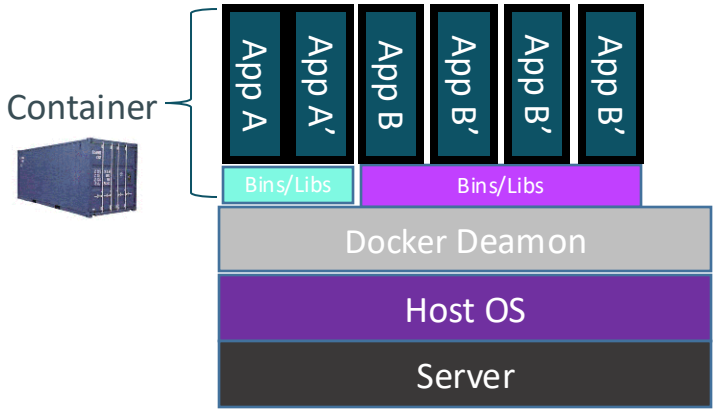


VMS VS CONTAINERS



Los contenedores están aislados pero comparten el kernel

…el resultado es un despliegue más rápido, menos costoso y más fácil



USO DE CONTENEDORES

- El uso de contenedores ayuda no solo con la eficacia, la elasticidad y la reutilización de las aplicaciones alojadas, sino también con la portabilidad de las aplicaciones.

- Open Container Initiative proporciona un conjunto de estándares de la industria que definen una especificación de tiempo de ejecución de contenedor y una especificación de imagen de contenedor.

- La especificación de imagen define el formato del paquete de archivos y metadatos que forman una imagen de contenedor.

- Cuando se crea una aplicación como imagen de contenedor que cumple con el estándar OCI, se puede usar cualquier motor de contenedores que cumpla con OCI para ejecutar la aplicación.

MOTORES DE CONTENEDORES

- Hay muchos motores de contenedores disponibles para administrar y ejecutar contenedores individuales, incluidos Rocket, Drawbridge, LXC, Docker y Podman.

- Podman está disponible en Red Hat Enterprise Linux 7.6 y versiones posteriores, y se usa en este curso para iniciar, administrar y finalizar contenedores individuales.

VENTAJAS DE USAR CONTENEDORES

- Menor tamaño del hardware.
- Aislamiento del entorno .
- Implementación rápida.
- Implementación de varios entornos.
- Reutilización.

HISTORIA DE LOS CONTENEDORES

- Los contenedores han ganado popularidad rápidamente en los últimos años.
- Sin embargo, la tecnología detrás de los contenedores ha existido por un tiempo relativamente largo.
- En 2001, Linux introdujo un proyecto llamado VServer.
- VServer fue el primer intento de ejecutar conjuntos completos de procesos dentro de un solo servidor con un alto grado de aislamiento.

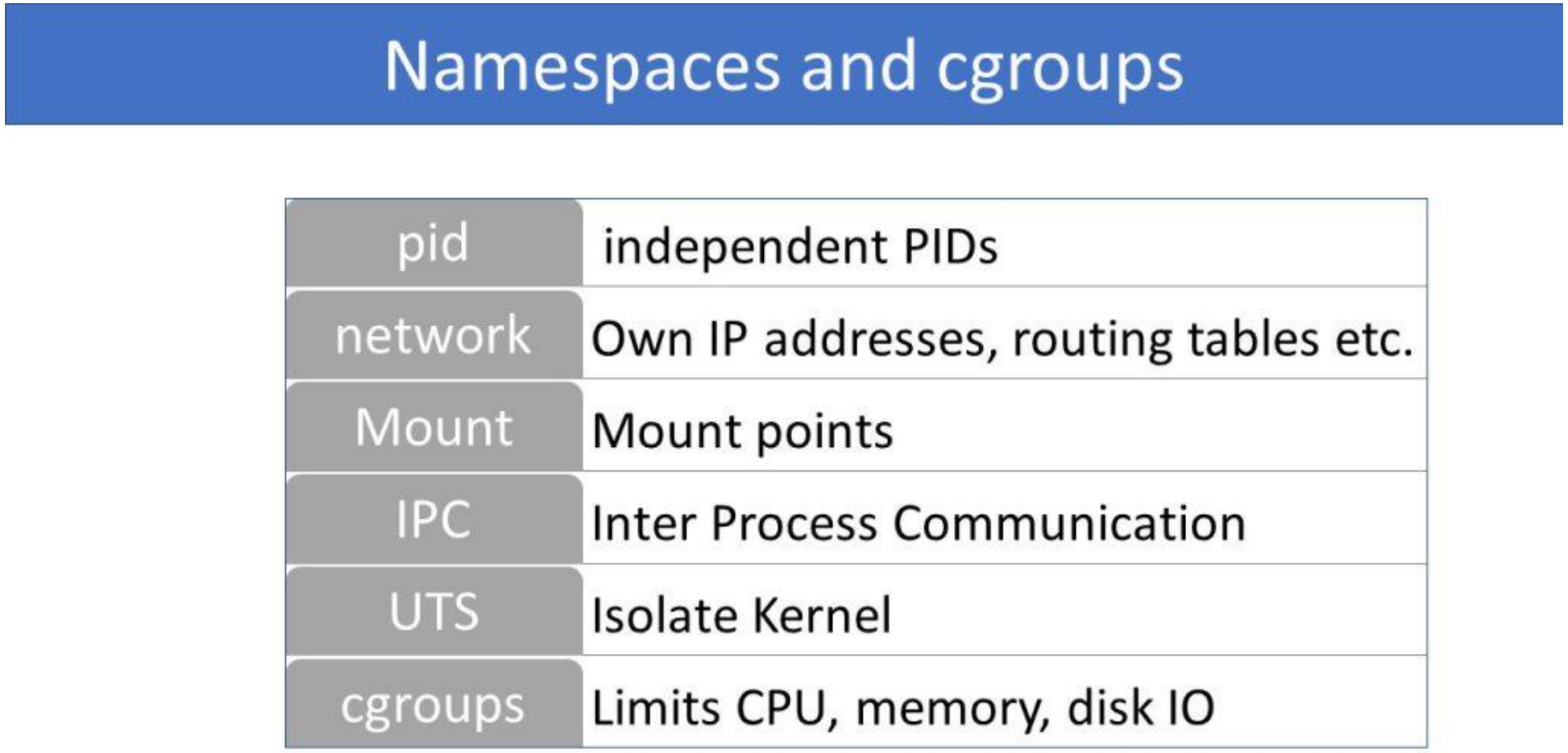
EVOLUCIÓN DE LOS PROCESOS AISLADOS

- A partir de VServer, la idea de los procesos aislados evolucionó y se formalizó en torno a las siguientes características del kernel de Linux:
  - Espacios de nombres
  - Grupos de control (cgroups)
  - Seccomp
  - SELinux (Security-Enhanced Linux)

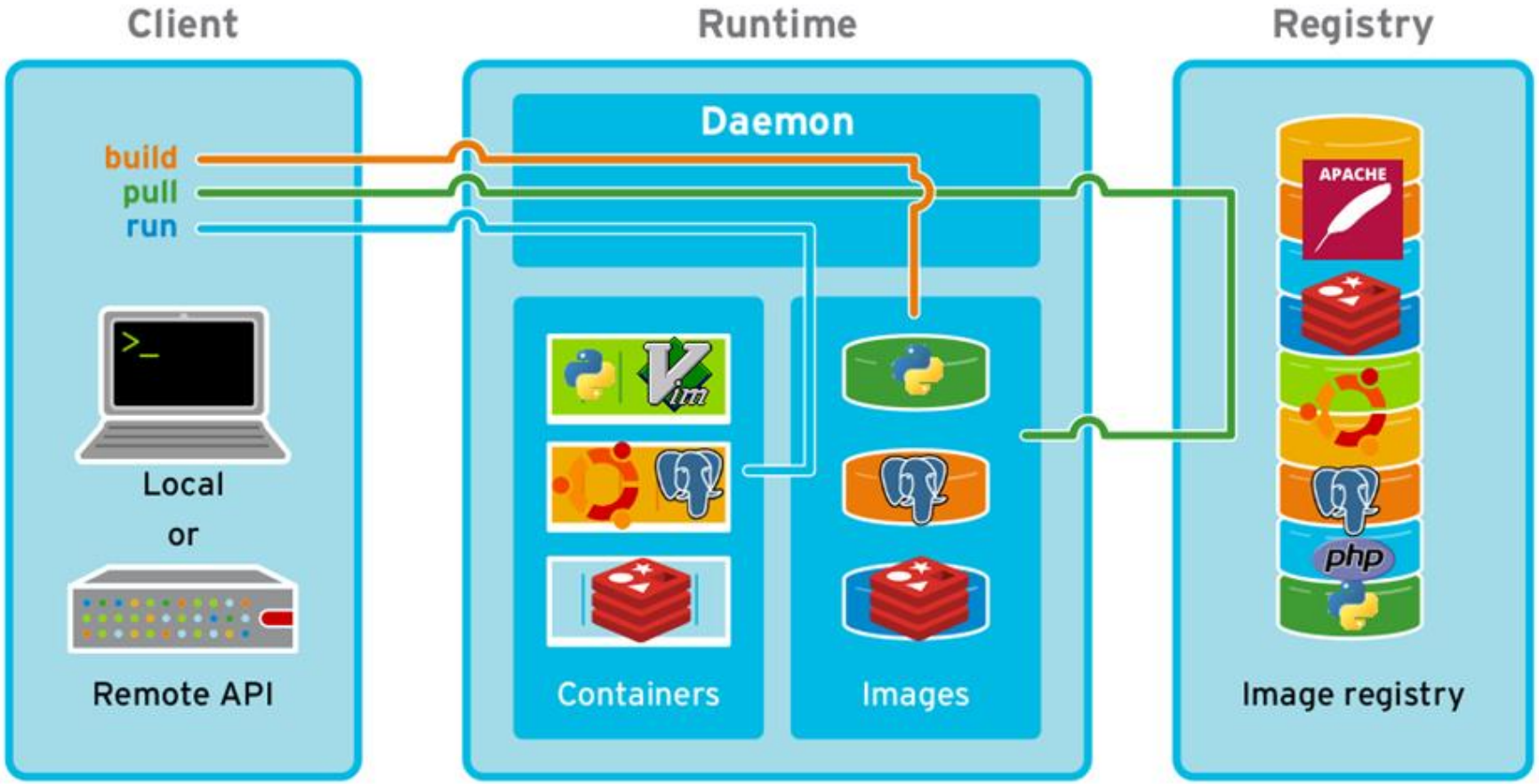
- Todas estas innovaciones y características se centran en un concepto básico: permitir que los procesos se ejecuten de manera aislada y, al mismo tiempo, accedan a los recursos del sistema.

- Este concepto es la base de la tecnología de contenedores y la base para todas las implementaciones de contenedores.

NAMESPACES Y CGROUPS



ARQUITECTURA DE LOS CONTENEDORES DE LINUX



REPOSITORIO DE IMÁGENES

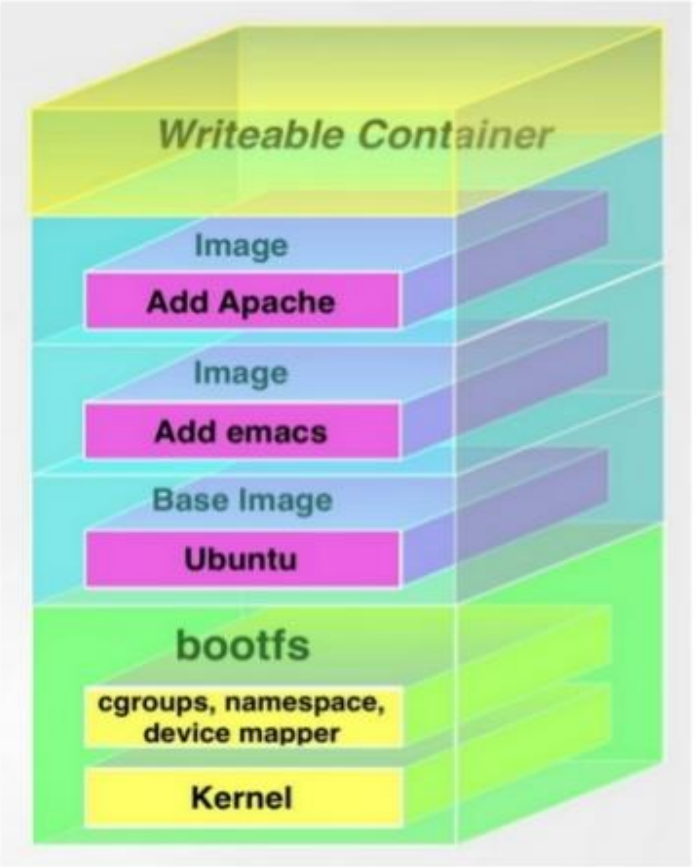
- Un repositorio de imágenes es solo un servicio (público o privado) donde las imágenes se pueden almacenar, buscar y recuperar.

- Otras características proporcionadas por los repositorios de imágenes son acceso remoto, metadatos de imagen, autorización o control de versión de imagen.

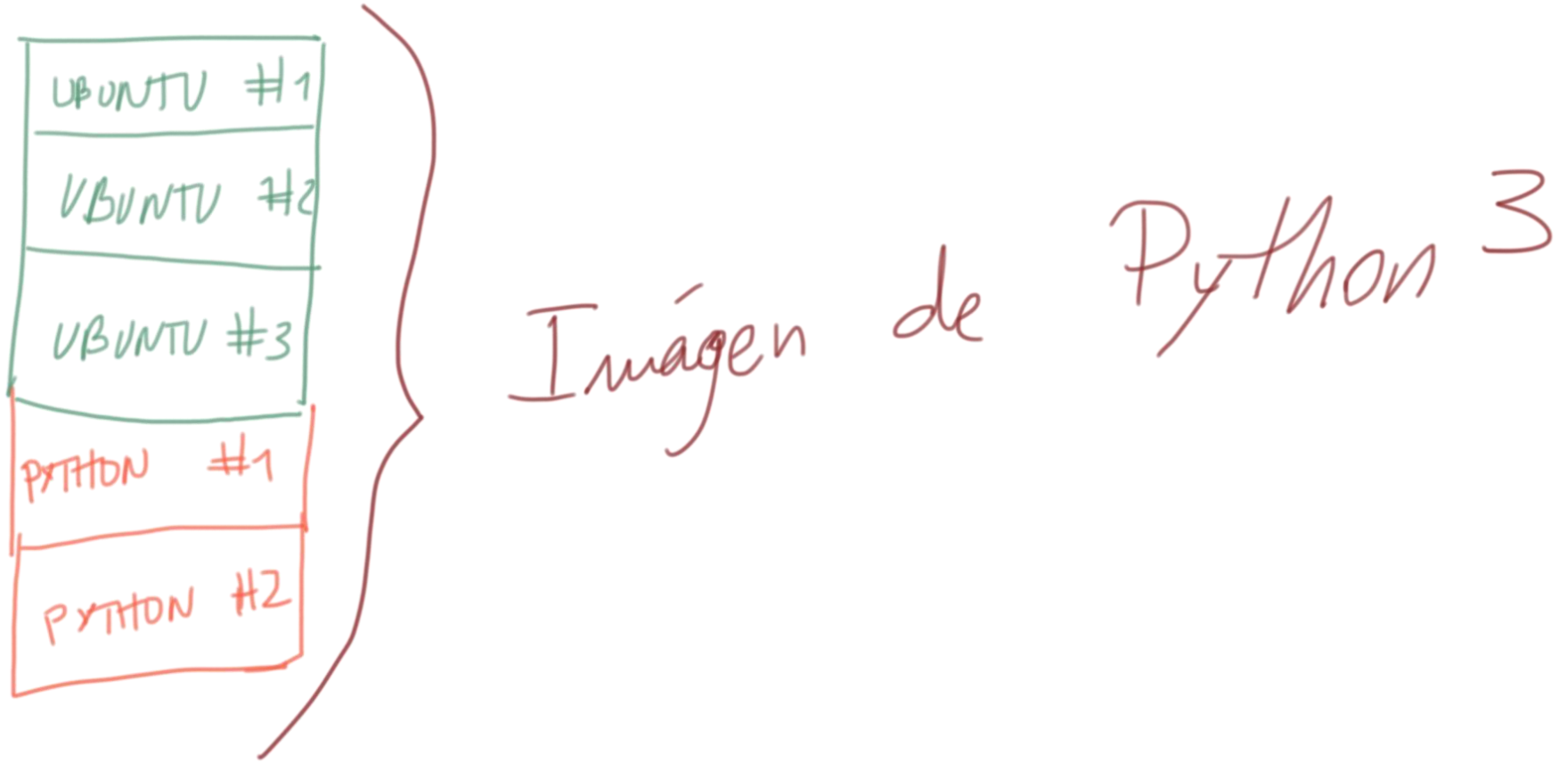
- Existen muchos repositorios de imágenes diferentes disponibles, cada uno con características diferentes:
  - Red Hat Container Catalog
  - Docker Hub
  - Red Hat Quay
  - Google Container Registry
  - Amazon Elastic Container Registry

IMÁGENES

- Una imagen está formada por capas (layers) que se montan unas encima de otras. Todas en modo sólo lectura.
- La última capa se monta como lectura/escritura y da lugar al contenedor.
- Las capas usan el patrón “copy on write”



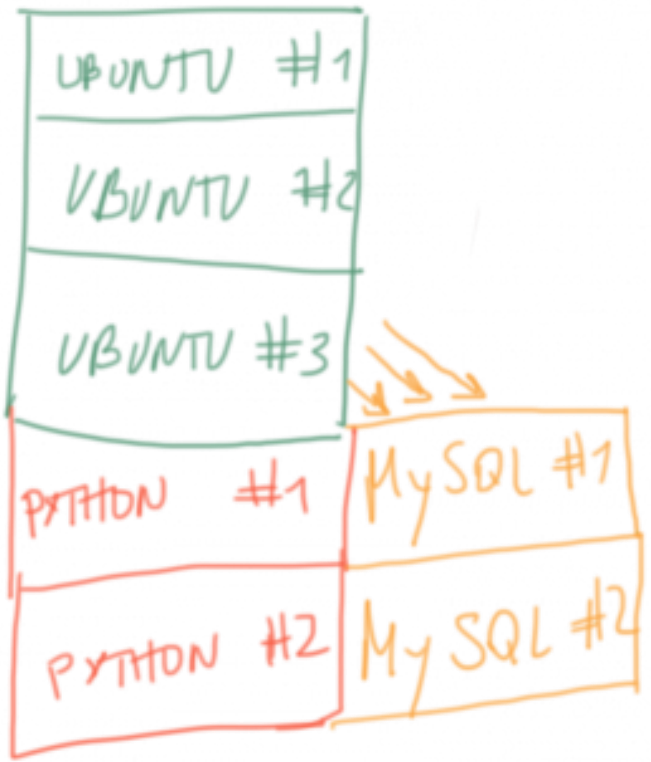
IMÁGENES: EJEMPLO



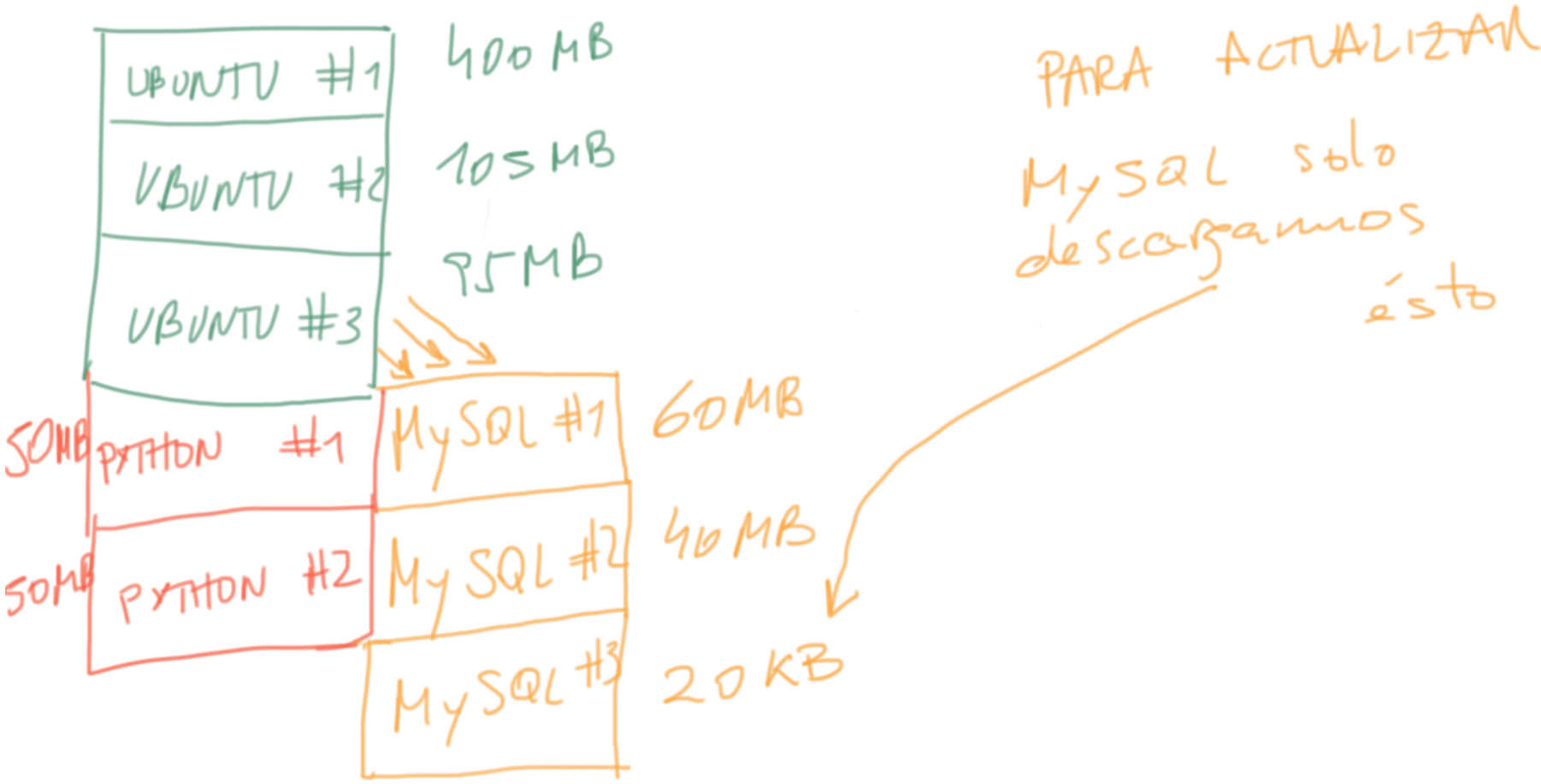
IMÁGENES: REAPROVECHAMIENTO DE LAS CAPAS

- Si ahora instalas MySQL y éste se basa también en Ubuntu 18.04 como imagen base, no hará falta descargar Ubuntu de nuevo.

- Su mayor ventaja es que se trata cada capa como un elemento individual.

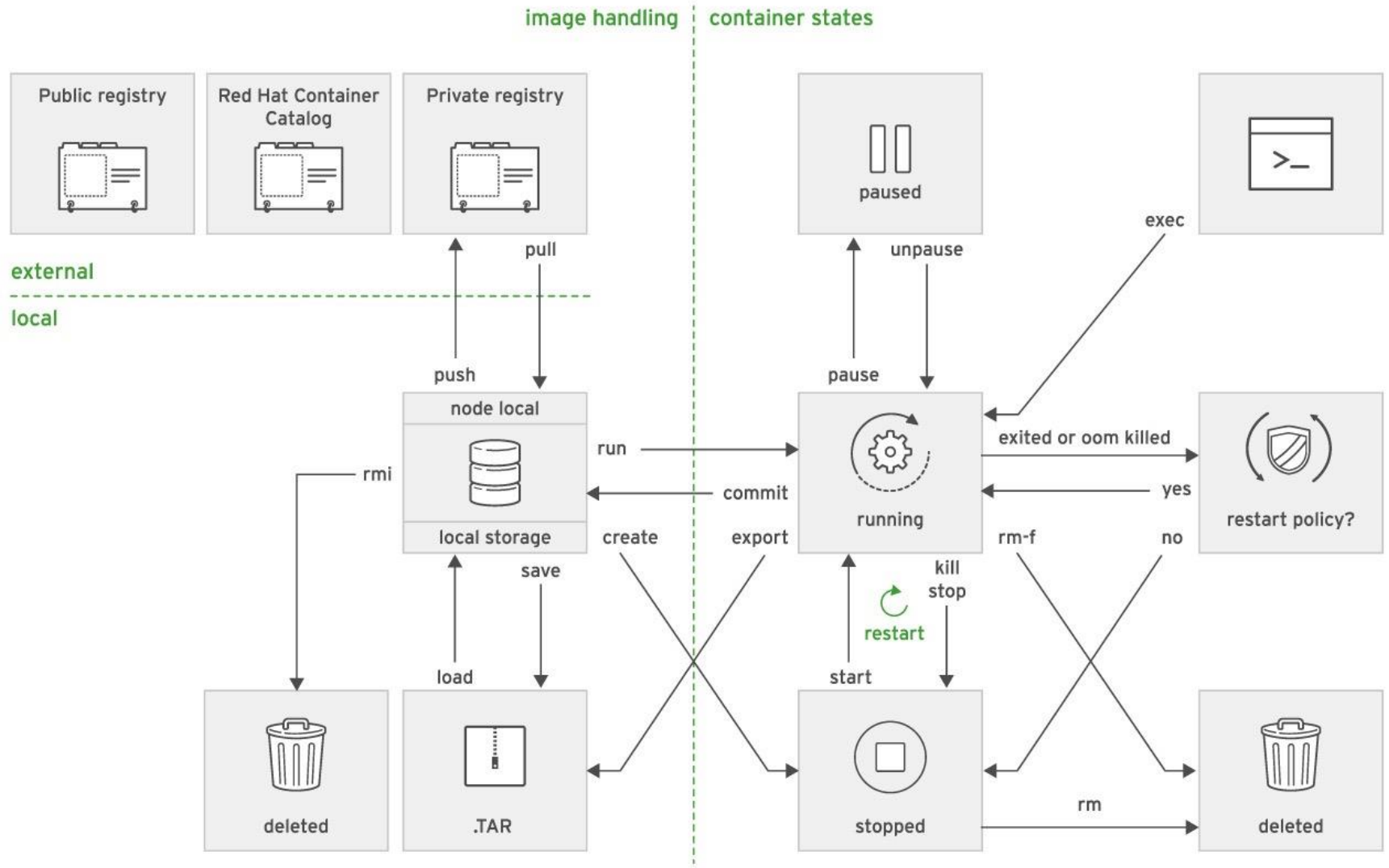


IMÁGENES: CAMBIO DE DATOS



GESTIÓN DE CONTENEDORES CON PODMAN

GESTIÓN DEL CICLO DE VIDA DE UN CONTENEDOR CON PODMAN



EJECUCIÓN DE CONTENEDORES

- El comando **podman run** ejecuta un contenedor localmente basado en una imagen.
- Como mínimo, el comando requiere el nombre de la imagen para ejecutarse en el contenedor.
- El comando **podman run** usa todos los parámetros después del nombre de la imagen como comando de punto de entrada para el contenedor.

```
[student@workstation containers]$ sudo podman run ubi7/ubi:7.7 echo "Hello!"
Hello world
```

BACKGROUND PROCESS

- Para iniciar una imagen de contenedor como proceso en segundo plano, pase la opción -d al comando podman run:

```
[student@workstation ~]$ sudo podman run -d rhscl/httpd-24-rhel7:2.4-36.8
ff4ec6d74e9b2a7b55c49f138e56f8bc46fe2a09c23093664fea7febc3dfa1b2
[student@workstation ~]$ sudo podman inspect -l \
> -f "{{.NetworkSettings.IPAddress}}"
10.88.0.68
[student@workstation ~]$ curl http://10.88.0.68:8080
  ...output omitted...
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  ...output omitted...
<title>
  Test Page for the Apache HTTP Server on Red Hat Enterprise Linux
</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<style type="text/css">
  ...output omitted...
```

OPCIONES AL CREAR CONTENEDORES

- Al hacer referencia al contenedor, Podman reconoce un contenedor con el nombre del contenedor o la identificación del contenedor generada.
- La opción **--name** para establecer el nombre del contenedor cuando ejecute el contenedor con Podman.
- Podman puede redirigir los flujos de entrada y salida del contenedor a la consola. El subcomando ejecutar requiere los indicadores **-t** y **-i** (o, en resumen, el indicador -it) para habilitar la interactividad.

```
[student@workstation ~]$ sudo podman run -it ubi7/ubi:7.7 /bin/bash
bash-4.2# ls
...output omitted...
bash-4.2# whoami
root
bash-4.2# exit
exit
[student@workstation ~]$
```

SINTAXIS IMÁGENES

- Algunos contenedores necesitan o pueden utilizar parámetros externos proporcionados al inicio.

- Podman puede inyectar variables de entorno en contenedores al inicio agregando el indicador **-e** al subcomando de ejecución:

```
[root@workstation ~]# sudo podman run --name mysql-custom \
> -e MYSQL_USER=redhat -e MYSQL_PASSWORD=r3dh4t \
> -d rhmap47/mysql:5.5
```

PROCESO DE CREACIÓN DE UN CONTENEDOR

- El comando **podman run** crea un nuevo contenedor a partir de una imagen e inicia un proceso dentro del nuevo contenedor.
- Si la imagen del contenedor no está disponible localmente, este comando intenta descargar la imagen usando el repositorio de imágenes configurado:

```
[student@workstation ~]$ sudo podman run rhscl/httpd-24-rhel7
Trying to pull regist...httpd-24-rhel7:latest...Getting image source signatures
Copying blob sha256:23113...b0be82
72.21 MB / 72.21 MB [=============================================================] 7s
...output omitted...AH00094: Command line: 'httpd -D FOREGROUND'
^C
```

INFORMACIÓN DE CONTENEDORES

- Podman identifica los contenedores por un ID de contenedor único o un nombre de contenedor.

- El comando **podman ps** muestra el ID del contenedor y los nombres de todos los contenedores que se ejecutan activamente:

```
[student@workstation ~]$ sudo podman ps
CONTAINER ID    IMAGE                   COMMAND                  ... NAMES
47c9aad6049 1   rhscl/httpd-24-rhel7    "httpd -D FOREGROUND"    ... focused_fermat 2
```

1. The container ID is unique and generated automatically.
2. The container name can be manually specified, otherwise it is generated automatically. This name must be unique or the **run** command fails.

EJECUCIÓN DE COMANDOS EN CONTENEDORES

- Cuando un contenedor se inicia, ejecuta el comando de punto de entrada.

- Sin embargo, puede ser necesario ejecutar otros comandos para administrar el contenedor en ejecución. Algunos casos de uso típicos se muestran a continuación:

  - Ejecutar un shell interactivo en un contenedor que ya se está ejecutando.
  - Ejecución de procesos que actualizan o muestran los archivos del contenedor.
  - Iniciar nuevos procesos en segundo plano dentro del contenedor.

```
[student@workstation ~]$ sudo podman exec 7ed6e671a600 cat /etc/hostname
7ed6e671a600
```

GESTIÓN DE CONTENEDORES I

- Los usuarios también pueden examinar el estado del contenedor y los metadatos con fines de depuración, actualización o generación de informes.

```
[student@workstation ~]$ sudo podman ps
CONTAINER ID   IMAGE          COMMAND      CREATED  STATUS   PORTS    NAMES
77d4b7b8ed1f(1) rhscl/httpd-24-rhel7(2) "httpd..."(3) ...ago(4) Up...(5) 80/tcp(6) my-
htt...(7)
```

1. Each container, when created, gets a **`container ID`**, which is a hexadecimal number. This ID looks like an image ID but is unrelated.
2. Container image that was used to start the container.
3. Command executed when the container started.
4. Date and time the container was started.
5. Total container uptime, if still running, or time since terminated.
6. Ports that were exposed by the container or any port forwarding that might be configured.
7. The container name.

GESTIÓN DE CONTENEDORES II

- Podman no desecha los contenedores detenidos inmediatamente.
- Podman conserva sus sistemas de archivos locales y otros estados para facilitar el análisis post mortem.
- La opción **-a** enumera todos los contenedores, incluidos los detenidos:

```
[student@workstation ~]$ sudo podman ps -a
CONTAINER ID  IMAGE                COMMAND     CREATED  STATUS          PORTS  NAMES
4829d82fbbff  rhscl/httpd-24-rhel7  "httpd..."  ...ago   Exited (0)...          my-
httpd...
```

INYECTAR VARIABLES AL CONTENEDOR

- Algunos contenedores necesitan o pueden utilizar parámetros externos proporcionados al inicio.

- Podman puede inyectar variables de entorno en contenedores al inicio agregando el indicador **-e** al subcomando de ejecución:

```
[root@workstation ~]# sudo podman run --name mysql-custom \
> -e MYSQL_USER=redhat -e MYSQL_PASSWORD=r3dh4t \
> -d rhmap47/mysql:5.5
```

GESTIÓN DE CONTENEDORES III

- Listar todos los contenedores que tienes en ejecución utilizando el comando **podman ps -a.**
- Y si quieres ver también los que están parados, tienes que utilizar **podman ps -a**.
- Para borrar un contenedor puedes utilizar **podman rm <contenedor>.**
- Mientras que para borrar una imagen deberías utilizar **podman rmi <imagen>.**

STOP & KILL DE CONTENEDORES

- **podman stop** es más fácil que encontrar el proceso de inicio del contenedor en el sistema operativo host y matarlo.

```
[student@workstation ~]$ sudo podman stop my-httpd-container
77d4b7b8ed1fd57449163bcb0b78d205e70d2314273263ab941c0c371ad56412
```

- **podman kill**: este comando envía señales Unix al proceso principal en el contenedor. Si no se especifica la señal, envía la señal SIGKILL, terminando el proceso principal y el contenedor.

```
[student@workstation ~]$ sudo podman kill my-httpd-container
77d4b7b8ed1fd57449163bcb0b78d205e70d2314273263ab941c0c371ad56412
```

RM & RESTART DE CONTENEDORES

- **podman restart:** este comando reinicia un contenedor detenido.

```
[student@workstation ~]$ sudo podman restart my-httpd-container
77d4b7b8ed1fd57449163bcb0b78d205e70d2314273263ab941c0c371ad56412
```

- **podman rm:** este comando elimina un contenedor y descarta su estado y sistema de archivos, también se pueden utilizar las opciones de borrado **–f, -a**.

```
[student@workstation ~]$ sudo podman rm my-httpd-container
77d4b7b8ed1fd57449163bcb0b78d205e70d2314273263ab941c0c371ad56412
```

```
[student@workstation ~]$ sudo podman rm -a
5fd8e98ec7eab567eabe84943fe82e99fdfc91d12c65d99ec760d5a55b8470d6
716fd687f65b0957edac73b84b3253760e915166d3bc620c4aec8e5f4eadfe8e
86162c906b44f4cb63ba2e3386554030dcb6abedbcee9e9fcad60aa9f8b2d5d4
```

GESTIÓN DE IMAGENES CON PODMAN

REPOSITORIOS EN PODMAN

- Para configurar registros para el comando podman, se debe actualizar el archivo **/ etc / containers / registries.conf**.
- Edita la entrada de registros en la sección [registries.search], agregando una entrada a la lista de valores.

```
[registries.search]
registries = ["registry.access.redhat.com", "quay.io"]
```

- Las conexiones seguras a un registro requieren un certificado de confianza.
- Para admitir conexiones inseguras, agrega el nombre del registro a la entrada de registros en la sección [registries.insecure] del archivo /etc/containers/registries.conf:

```
[registries.insecure]
registries = ['localhost:5000']
```

BÚSQUEDA DE IMÁGENES EN REPOSITORIOS

- La sintaxis del comando de podman search :

```
[student@workstation ~]$ sudo podman search [OPTIONS] <term>
```

- En la tabla se muestran algunas opciones útiles disponibles para el subcomando de búsqueda:

Option	Description
`--limit <number>`	Limits the number of listed images per registry.

Option	Description
`--filter <filter=value>`	Filter output based on conditions provided. Supported filters are: • **`stars=<number>`**: Show only images with at least this number of stars. • **`is-automated=<true|false>`**: Show only images automatically built. • **`is-official=<true|false>`**: Show only images flagged as official.
`--tls-verify <true|false>`	Enables or disables HTTPS certificate validation for all used registries. **`true`**

REPOSITORIO HTTP API

- Un registro remoto expone los servicios web que proporcionan una interfaz de programación de aplicaciones (API) al registro.
- Podman utiliza estas interfaces para acceder e interactuar con repositorios remotos.
- Para enumerar todos los repositorios disponibles en un registro, use el punto final **/ v2 / _catalog**.
- El parámetro **n** es utilizado para limitar el número de repositorios que se devolverán.

```
[student@workstation ~]$ curl -Ls https://myserver/v2/_catalog?n=3
{"repositories":["centos/httpd","do180/custom-httpd","hello-openshift"]}
```

AUTOIDENTIFICACIÓN EN REPOSITORIO DE IMÁGENES

- Algunos registros de imágenes de contenedores requieren autorización de acceso.
- El comando de inicio de sesión de podman permite la autenticación de nombre de usuario y contraseña en un registro:

```
[student@workstation ~]$ sudo podman login -u username \
> -p password registry.access.redhat.com
Login Succeeded!
```

- La API HTTP del registro requiere credenciales de autenticación. Primero, utilice el servicio Red Hat Single Sign On (SSO) para obtener un token de acceso:

```
[student@workstation ~]$ curl -u username:password -Ls \
> "https://sso.redhat.com/auth/realms/rhcc/protocol/redhat-docker-v2/auth?
service=docker-registry"
{"token":"eyJh...o5G8",
"access_token":"eyJh...mgL4",
"expires_in":...output omitted...}[student@workstation ~]$
```

PULLING IMAGES

- Para extraer imágenes de contenedor de un registro, use el comando **podman pull**:

```
[student@workstation ~]$ sudo podman pull [OPTIONS] [REGISTRY[:PORT]/]NAME[:TAG]
```

- Para extraer un contenedor NGINX del registro quay.io, use el siguiente comando:

```
[student@workstation ~]$ sudo podman pull quay.io/bitnami/nginx
```

LISTANDO IMÁGENES LOCALES

- Cualquier imagen de contenedor descargada de un registro se almacena localmente en el mismo host donde se ejecuta el comando podman.
- Este comportamiento evita la repetición de descargas de imágenes y minimiza el tiempo de implementación de un contenedor.
- Podman también almacena cualquier imagen de contenedor personalizada que cree en el mismo almacenamiento local.

```
[student@workstation ~]$ sudo podman images
REPOSITORY                                TAG    IMAGE ID     CREATED     SIZE
registry.redhat.io/rhscl/mysql-57-rhel7 latest c07bf25398f4 13 days ago 444MB
```

ETIQUETA DE IMÁGENES

- Una etiqueta de imagen es un mecanismo para admitir múltiples lanzamientos de la misma imagen.

- Cualquier subcomando de Podman que requiera un nombre de imagen de contenedor acepta un parámetro de etiqueta para diferenciar entre varias etiquetas.

- Si no se pone ninguna etiqueta al descargar la imagen, el sistema usa la ultima versión .

```
[student@workstation ~]$ sudo podman pull rhscl/mysql-57-rhel7:5.7
```

NAMESPACE DE IMÁGENES

- Root-like

```
ubuntu
```

- User (and organizations)

```
jpetazzo/clock
```

- Self-Hosted

```
registry.example.com:5000/my-private-image
```

GUARDAR IMÁGENES

- El comando de **podman save** guardará la imagen especificada incluyendo todas sus capas individuales y metadatos asociados.

```
[student@workstation ~]$ sudo podman save [-o FILE_NAME] IMAGE_NAME[:TAG]
```

- Esto es una excelente manera de hacer una copia de seguridad de una imagen.

- Podman envía la imagen generada a la salida estándar como datos binarios. Para evitarlo, usa la opción **-o**.

```
[student@workstation ~]$ sudo podman save \
> -o mysql.tar registry.access.redhat.com/rhscl/mysql-57-rhel7:5.7
```

CARGAR IMÁGENES

- Utiliza los archivos .tar generados por el subcomando **save** para fines de respaldo.
- Para restaurar la imagen del contenedor, usa el comando **podman load**.
- La sintaxis general del comando es la siguiente:

```
[student@workstation ~]$ sudo podman load -i mysql.tar
```

- Para ahorrar espacio en disco, comprime el archivo generado por el subcomando **save** con Gzip usando el parámetro --**compress**.

BORRANDO IMÁGENES

- Podman mantiene cualquier imagen descargada en su almacenamiento local, incluso las que no se utilizan actualmente en ningún contenedor.
- Sin embargo, las imágenes pueden quedar obsoletas y deben reemplazarse posteriormente.
- Para eliminar una imagen del almacenamiento local, ejecute el comando **podman rmi**.
- La sintaxis de este comando es la siguiente:

```
[student@workstation ~]$ sudo podman rmi [OPTIONS] IMAGE [IMAGE...]
```

- Para eliminar todas las imágenes que no son utilizadas por ningún contenedor, usa el siguiente comando:

```
[student@workstation ~]$ sudo podman rmi -a
```

PUBLICANDO IMÁGENES EN UN SERVIDOR DE REGISTRO

- Los cambios no comprometidos en el sistema de archivos de un contenedor se pierden cuando el contenedor se elimina.
- Los cambios se pueden guardar en un nuevo capa de imagen que enumera la capa superior actual del contenedor como su principal.
- Esto permitiría el inicio de futuros contenedores con la nueva capa y sus cambios asociados, o con la capa original más antigua.
- Para enumerar todos los cambios acumulados en el sistema de archivos de un contenedor, usa el comando **podman container diff** .

```
[student@workstation ~]$ sudo podman diff mysql-basic
C /run
C /run/mysqld
A /run/mysqld/mysqld.pid
A /run/mysqld/mysqld.sock
A /run/mysqld/mysqld.sock.lock
A /run/secrets
```

PUBLICANDO IMÁGENES EN UN SERVIDOR DE REGISTRO

- Para publicar una imagen en un registro, esta debe residir en el almacenamiento local de Podman y estar etiquetada con fines de identificación.

- Para enviar la imagen al registro, utiliza la sintaxis del subcomando **push** .

```
[student@workstation ~]$ sudo podman push quay.io/bitnami/nginx
```

LAB 2
GESTIÓN DE CONTENEDORES E IMAGENES