Universidad Nacional Autónoma de México

Facultad de ingeniería

Reporte:

“Como los contenedores han redefinido la virtualización”

Sistemas Operativos

Profesor: Gunnar Eyal Wolf Izaevich

Desarrollo: José Eduardo Martínez García

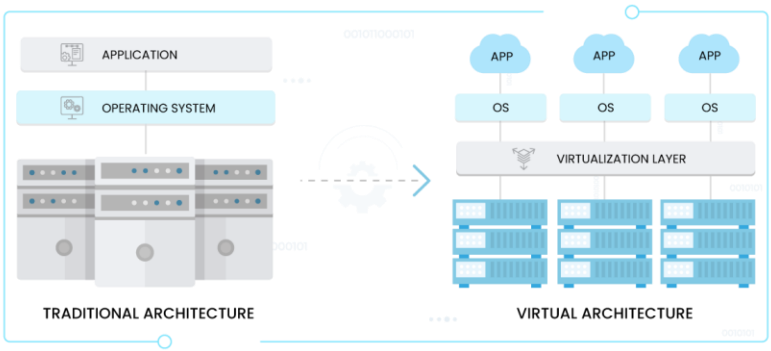
Grupo 8

Semestre 2026-1

Fecha de entrega 17/09/25

“Como los contenedores han redefinido la virtualización”

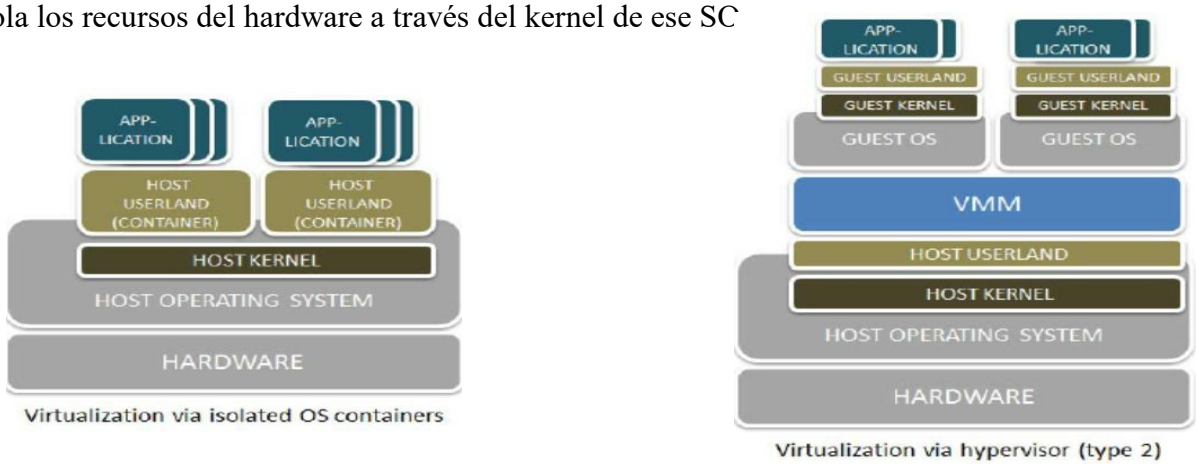
La virtualización es una tecnología que nos permite crear representaciones lógicas de recursos informáticos, una emulación de recursos de software y de hardware que gestiona un dispositivo; es una imitación de una computadora o parte de ella. Su objetivo es abstraer y utilizar componentes por debajo del nivel de usuario, es decir, componentes inferiores como hardware físico o un sistema operativo (SO); debe crear entornos aislados, independientes con una eficiente distribución de recursos utilizando un solo sistema físico, un único hardware -ya sea de manera directa o con intermediarios-. La representación se basa en la imitación o replicación de funcionalidades y comportamiento de recursos físicos y lógicos, es decir, pueden existir instancias virtuales prácticamente iguales a sus versiones tradicionales ejecutándose de forma simultánea en un mismo dispositivo, por ejemplo, con máquinas virtuales y/o contenedores.



Aquitectura tradicional vs arquitectura virtual

Aunque el concepto de virtualización surgió en los 60’s por la empresa IBM para solucionar problemas sobre la poca utilización de los recursos de hardware de *Mainframes*, no fue hasta 1990 que la empresa VMWARE comenzó a buscar una solución real a la rentabilidad en cuanto al desperdicio de hardware en equipos con arquitectura x86, de esta forma en 1999 -abriéndole paso a la virtualización de hardware basada en arquitectura x86- salió al mercado “*el ware Workstation 1.0*”, un programa que te permitía utilizar distintos SO’s en esa aplicación pero que seguía siendo una aplicación. En 2001 salió el primer hypervisor llamado “*VMWARE ESX*” que se convirtió en el equivalente del sistema operativo sobre el cual se ejecutan múltiples cargas de trabajo en forma de servidores virtuales, De ahí incio la época de guerra comercial donde el mercado se movió bastante y VMWARE llevo la delantera por varios años.

Este último (*VMWARE ESX*) era un hypervisor bare-metal, es decir, el hypervisor donde la capa de virtualización ejecuta y controla todos los componentes el hardware físico sin necesidad de un SO host. El otro tipo es el hypervisor kernel-based, es decir, son hypervisores que operan bajo un SO host, el cual controla los recursos del hardware a través del kernel de ese SC



virtualización bare-metal vs virtualización kernel-based

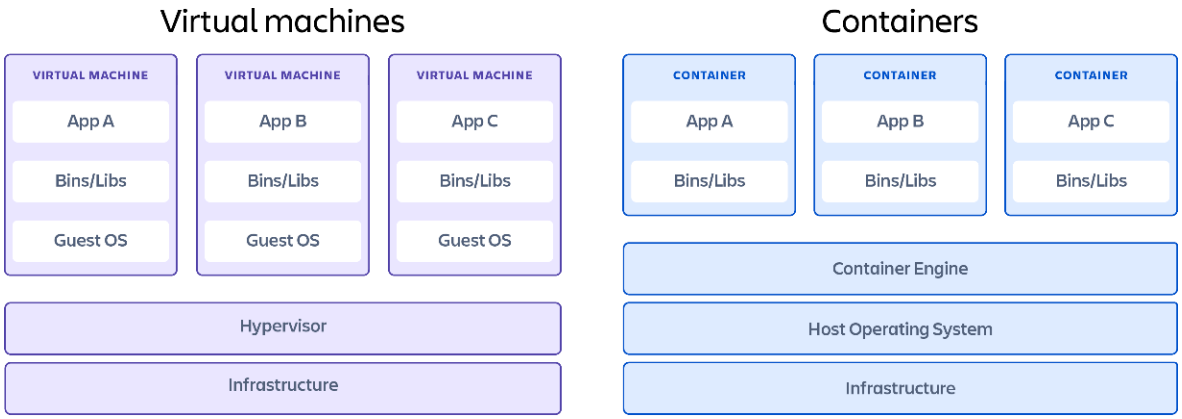
En los últimos años los contenedores y las máquinas virtuales se han convertido en herramientas poderosas para desarrollo de software y en la nube e incluso están presentes tras bambalinas para usuarios ajenos a estas tecnologías en redes sociales, plataformas de streaming, servicios en la nube, banca en línea y comercio electrónico. Ambos entran en la definición de virtualización, es decir, comparten enfoques, pero tienen distintos objetivos; ambos tienen características como aislamiento, administración de recursos, portabilidad y seguridad a distintos niveles, pero mediante distintas arquitecturas y gestión de recursos físicos.

¿Qué es una Máquina virtual?

Una máquina virtual (MV) es una emulación virtual o basada en software de un equipo físico con CPU, memoria, almacenamiento e interfaces de red. Funciona de forma idéntica a un equipo real, tiene SO y aplicaciones. Se le asignan porciones de CPU, memoria y almacenamiento de un ordenador físico (dispositivo personal o servidores remotos en el caso de servicios en la nube). La premisa de las MV es que se pueden tener múltiples MV en un solo equipo físico, las cuales tienen un nivel robusto de aislamiento entre ellas y son autónomas, además de que cada MV emula sistemas operativos independientes y entornos distintos sin necesidad de hardware adicional. Se crean y administran mediante un software denominado hipervisor, que actúa como intermediario entre el hardware físico y el sistema operativo GUEST (invitado) de las MV, asignando recursos físicos necesarios.

¿Qué es un contenedor?

Por otro lado, al igual que en el sector transporte los trenes o buques utilizan contenedores para aislar y trasportar mercancía con usos particulares -los cuales tienen espacios definidos y reducidos- los contenedores son unidades de software ligeras, independientes y portables que empaquetan aplicaciones junto con sus dependencias, bibliotecas y configuraciones necesarias para funcionar. A diferencia de las maquinas virtuales no requieren emular un SO completo, sino que comparten el kernel del SO host, es decir, tienen una virtualización nivel SO, esto los hace más rápidos y agiles para trabajar, además que representan una fracción de espacio en contraste con su contraparte de comparación “la máquina virtual”.



Arquitectura de una máquina virtual frente a la de un contenedor

Diferencias Similitudes y diferencias entre un contenedor y una máquina virtual

Función	Contenedor	Máquina Virtual
Tipo de virtualización	Virtualización de sistema operativo	virtualización de Hardware
Sistema operativo	Comparte el kernel del SO host	SO invitado (instancia de SO completa)
Tamaño	Megabytes (MB)	Gigabytes (GB)
Tiempo de inicio	Segundos	Minutos
Uso de recursos	Inferior	Superior
Aislamiento	Nivel de proceso	Nivel de hardware
Portabilidad	Altamente portátil	Menos portable
Administración	Por lo general, se administran con herramientas de organización de contenedores (Kubernetes)	Se administran con hipervisores (p. ej., VMware, KVM) o plataformas de administración en la nube

Tabla 2. Comparativa de tecnologías basadas en virtualización

Historia de los contenedores

Si bien la historia de los contenedores comienza con el surgimiento de la virtualización -al ser estos del tipo virtualización a nivel SO- en base a su tipo de aislamiento, uno que no utiliza directamente el hardware sino que lo realiza a nivel de proceso, es decir, que es gestionado por el propio sistema operativo que realiza una serie de abstracciones y comunicaciones con el kernel para lograrlo. Con esto claro podemos remontarnos al año de 1982, ese año Bill Joy, uno de los desarrolladores de BSD (Berkeley Software distribution) implemento una llamada de sistema Unix y derivados denominada *"Chroot"* la cual es un comando que permite seleccionar un directorio del sistema de archivos como directorio raíz y una vez seleccionado no te va a permitir acceder a ningún directorio por encima de este, solo a los directorios hijos, es decir, no puede acceder a ningún archivo fuera del árbol de directorios designados, definiendo un aislamiento de directorios.

Ejemplo de comando chroot

Entrada a chroot. Corre *arch-chroot* con la nueva raíz como primer argumento:

```
# chroot /path/to/new/root
```

o

```
# chroot /path/to/new/root
```

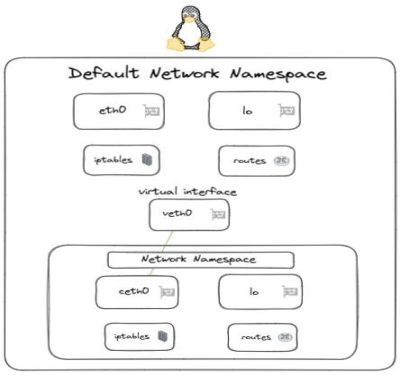
or

Salir de chroot. Para salir de *chroot*, usa:

```
# exit
```

En 1999 Paul-Henning Kamp implemento las *"jails"* como un nivel de virtualización de SO que permite a administradores del sistema particionar el software en varios minisistemas que comparten kernel del sistema operativo. Fue lanzado en el 2000 por FreeBSD con el deseo de separar ambientes compartidos y eliminar las limitaciones de *chroot* a solo aislamiento de directorios, es decir, en *chroot* los usuarios, el sistema de red, los procesos en ejecución siguen siendo compartidos, en cambio FreeBSD obtiene sus

propios usuarios, sistema de red, listado de procesos en ejecución y archivos por lo que tenemos un nivel de aislamiento en el que ya no se pueden ver los usuarios y procesos que se están utilizando y ejecutando respectivamente en el SO host, además de que cada *jail* está aislada de las demás.



En el 2002 Linux implemento los *"namespaces"* en la versión del kernel 2.4.19, los cuales son una característica que divide y aisla los recursos del kernel, similar a los *jails* de FreBSD, es decir, un conjunto de procesos, espacio de red, usuarios, estos son asociados a un *namespace* y solo puede usar los recursos asociados a ese *namespace*, mientras que otro *namespace* ve sus propios recursos, diferentes al otro, aun teniendo el mismo *namespace*.

En el 2003 nació Google Borg es un manejador de clusters interno de Google que gestiona miles de aplicaciones y servicios en una infraestructura a gran escala compuesta por decenas de miles de máquinas. En sí, Google Borg hace por un clúster de máquinas lo que un sistema operativo hace por una computadora, es decir, administra los recursos como CPU, memoria, disco y los procesos. Su tipo de virtualización es debido a esto de SO; utiliza el mismo kernel en todas las maquinas del cluster y aisla los procesos para que no interfieran entre sí, asignándole recursos a cada proceso. Borg es el predecesor directo de los kubernetes ya que las ideas y errores implementadas fueron la base para su desarrollo, es decir, fue una especie de orquestador primitivo de contenedores.

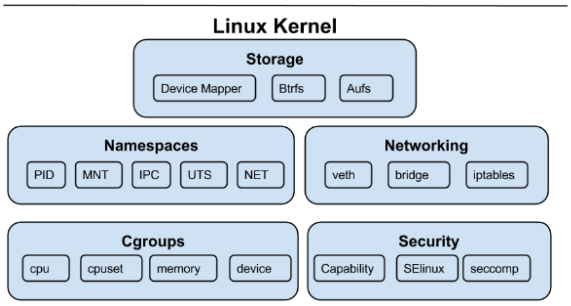
En 2005 salió a la luz un proyecto comunitario soportado por SWsoft llamado *"OpenVZ"*, una tecnología de virtualización nivel SO que permite ejecutar múltiples instancias aisladas conocidas como servidores privados virtuales (VPS) o entornos virtuales (EV) en un mismo servidor físico, no emula un hardware completo y tiene la limitación de que el host debe ser Linux; es una especie de maquina virtual limitada y basada en el kernel, cada EV es un servidor independiente con sus propios usuarios, conjunto de procesos, red, dispositivos E/S y sistema de archivos, ademas permite establecer restricciones de uso de disco, memoria y CPU.

En el 2006 aparece *Google Process Containers* desarrollado por Paul Menage y Rohit Seth, cuyo nombre cambio a *Control Groups* (*Cgroups*) por Tejen Heo y se integró al kernel de linux en 2007, el cual es una funcionalidad del kernel que permite organizar los procesos en grupos jerárquicos y controlar mediante limites la asignación de recursos del sistema como CPU, memoria, red o entradas y salidas, es decir, establece prioridades, contabiliza y restringe el consumo de recursos. Puede limitar la memoria máxima de un grupo, asignar un porcentaje fijo de CPU o establecer un estándar de entradas y salidas, de esta manera garantiza un aislamiento entre procesos y una gestión de recursos definida.

Hasta aquí mencione los dos pilares que le dieron forma al concepto de contenedor, viéndose de manera resumida como la siguiente igualdad de tecnologías base:

$$\boldsymbol{Contenedores = Namespaces + Cgroups}$$

O de manera más clara:



tecnologías que intervienen en un contenedor

$$\textbf{Contenedores} = \begin{array}{c} fraccion\ de\ recursos \\ del\ kernel\ con\ sus\ propios \\ usuarios, conjunto\ de\ procesos, \\ sistema\ de\ red, \\ aislamiento\ del\ SO\ host \end{array} + \begin{array}{c} restricciones\ medinate\ limites \\ de\ memoria, asignacion\ de\ \% \\ de\ CPU\ para\ procesos \end{array}$$

En el 2008 aparece LXC desarrollado por IBM para la versión del kernel de Linux 2.6.24, se trata de una librería de bajo nivel que dejaba utilizar namespaces y Cgroups de manera sencilla para crear los primeros contenedores, es una forma de virtualización de nivel sistema operativo que deja ejecutar varios entornos Linux en un mismo kernel, es decir, conjuntos de procesos en un mismo sistema, además todos los archivos necesarios para crearlos provienen de una imanen en particular, lo que facilita el despliegue de contenedores.

En 2013 Google saca un software/librería gratis llamado *"lmctfy"* (Let Me Contain That For You)y open-source similar a Docker y LXC como un intento de crear contenedores de manera más fácil basada en Linux kernel Cgroups, esta fue desarrollada hasta 2015. Igualmente, en el 2013 aparece *"Google Omega"*, predecesor de los kubernetes debido a que bastante de las ideas de este se utilizaron para la tecnología mencionada, en realidad es un orquestador de contenedores que surgió para adaptarse a las nuevas implementaciones de aislamiento y seguridad que no le daban abasto con Google Borg. Finalmente, ese mismo año (2013) surgió *"Docker"* una plataforma de código abierto para crear, probar e implementar aplicaciones dentro de contenedores, popularizo los contenedores debido a la sencillez de creación e implementación, no se necesitaba conocer de la API de Linux para su implementación.

En 2014 surge *"Core Os rtk"* como competencia a Docker, como una versión mas segura, interoperable y open-source que corregía algunas vulnerabilidades de Docker. Ese mismo año salió *"Google Kubernetes"*, un software open-source que es un orquestador de contenedores; permite implementar, escalar y administrar aplicaciones de contenedores en cualquier lugar, simplifica la gestión cuando se escalan los contenedores y servidores, organiza las implementaciones bajo la arquitectura de los microservicios (servicios que componen una aplicación), además ayuda a la asignación de recursos y control de versiones. Kubernetes es una plataforma que corre y administra contenedores de varios *Container runtimes* como Docker, LXC o rtk de Core Os.

En 2015 nace la CNCF (Cloud Native Computing Foundation) como una fundación de código abierto que ayuda a las organizaciones a adentrarse en la nube, su objetivo es popularizar la creación de aplicaciones basadas en contenedores administradas por kubernetes.

En 2015 también surge OCI (Open Container Iniciative), una iniciativa con el objetivo de estandarizar industrialmente formatos de imágenes de contenedores y tiempos de ejecución que diferían entre distribuidoras para mejorar la comunicación e interoperabilidad, la cual fue establecida por las principales caras en el mundo de los contendores. Se definieron 3 especificaciones en la iniciativa:

- la Especificación de Tiempo de Ejecución (runtime-spec)
- la Especificación de Imagen (image-spec)
- la Especificación de Distribución (distribution-spec).

Empresas que forman parte del OCI

Con esto claro definimos algunas diferencias entre los principales containter runtimes.

Diferencias Similitudes y diferencias entre contenedores LXC, Docker y rtk

Función	LXC	Docker	rtk de CoreOs
Tipo de virtualización	Virtualización de SO	Virtualización de SO	Virtualización de SO
Enfoque / objetivo	Ofrecer entornos de sistema Linux completos y ligeros (similar a una máquina virtual pero sin emular hardware)	Empaquetar y desplegar aplicaciones junto con sus dependencias en imágenes portables para microservicios	Seguridad y compatibilidad; buscaba ser más seguro e interoperable que Docker
Arquitectura	ejecuta múltiples sistemas Linux en un host desde un solo sistema operativo.	aplicación + dependencias empaquetadas en una imagen	aplicación con modelo de ejecución basado en pods (similar a Kubernetes)
Portabilidad	Menos portable, dependiente del kernel y entorno Linux	Altamente portable (funciona en Linux, Windows, Mac mediante runtimes)	Portabilidad intermedia, buscaba estandarizar con OCI (Open Container Initiative)
Tamaño de imagenes	Pesadas (SO completo)	Muy ligeras (apps)	Ligeras (apps + seguridad)
Tiempo de inicio	segundos	Milisegundos/segundos	Milisegundos/segundos
Seguridad	Aislamiento básico mediante namespaces y cgroups	Buen aislamiento, en sus inicios tuvo problemas de seguridad	ejecución en pods, verificación de firmas de imágenes (seguridad prioritaria)
Popularidad	Baja	Alta	Limitada (desontinuado en 2020)

Tabla 2. Comparativa de container runtimes

Los contenedores son resultado de un conjunto de implementaciones para hacer el desarrollo de software y de aplicaciones en la nube de manera más rápida y aprovechando los recursos de mejor forma -un ejemplo son los usos de contenedores en los microservicios-, son resultado de tecnologias previas como los Cgroups y namespaces, dichas tecnologias se basan en la premisa de aislamiento y gestión de recursos informáticos y son resultado de otras implementaciones, además de que la historia de los contenedores esta ligada estrechamente al de las maquinas virtuales gracias a que ambas son resultado de la virtualizacion, igualmente entre contenedores hay ventajas y desventajas en base a sus objetivos, es decir,

por ejemplo, podrías ejecutar LXCs tan ligeros como contenedores docker, y podrías ejecutar un contenedor docker pesado con un sistema operativo completo, además el crecimiento en la cantidad de microservicios y contenedores trajo consigo nuevos desafíos relacionados con la gestión y la organización de entornos cada vez más complejos. Para resolver este problema surgieron plataformas de orquestación como Kubernetes, que permiten automatizar la implementación, el escalado y la administración de contenedores a gran escala.

Referencias

-Atlassian. (n.d.). *Comparación de contenedores y máquinas virtuales | Atlassian*. https://www.atlassian.com/es/microservices/cloud-computing/containers-vs-vms

-Trianz. (2023, July 27). *Contenerización vs. Virtualización: 7 diferencias técnicas | Trianz*. https://www.trianz.com/es/insights/containerization-vs-virtualization

-*¿Qué es la virtualización? - Explicación de la virtualización de la computación en la nube - AWS*. (n.d.). Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/virtualization/

-*¿Qué son los contenedores? Google Cloud*. (n.d.). Google Cloud. https://cloud.google.com/learn/what-are-containers?hl=es

-*Qué es una máquina virtual y cómo funciona | Microsoft Azure*. (n.d.). https://azure.microsoft.com/es-mx/resources/cloud-computing-dictionary/what-is-a-virtual-machine/

-*chroot - ArchWiki*. (n.d.). https://wiki.archlinux.org/title/Chroot

-Wikipedia contributors. (2025, August 20). *Berkeley Software Distribution*. Wikipedia. https://en.wikipedia.org/wiki/Berkeley_Software_Distribution

-*FreeBSD Documentation Portal*. (n.d.). FreeBSD Documentation Portal. https://docs.freebsd.org/en/books/handbook/jails/#jails-build

-Wikipedia contributors. (2025, August 31). *Linux namespaces*. Wikipedia. https://en.wikipedia.org/wiki/Linux_namespaces

-*Large-scale cluster management at Google with Borg*. (n.d.). https://research.google/pubs/large-scale-cluster-management-at-google-with-borg/

-Borg: A cluster management system. (s. f.). *MEDIUM*. https://medium.com/@adityashete009/borg-large-scale-cluster-management-system-cbdcc4f8eb91

-*Historia sobre Virtualización de Sistemas*. (n.d.). https://walternavarrete.com/historia-sobre-virtualizacion-de-sistemas/

-Wikipedia contributors. (2025, August 8). *CGroups*. https://en.wikipedia.org/wiki/Cgroups

-*¿Qué son los contenedores de Linux ?* (n.d.). https://www.redhat.com/es/topics/containers/whats-a-linux-container

-Wikipedia contributors. (2024, August 28). *LXC*. Wikipedia. https://en.wikipedia.org/wiki/LXC

-Wikipedia contributors. (2025, May 13). *LMcTfy*. https://en.wikipedia.org/wiki/Lmctfy

-*Contenedores de Docker | ¿Qué es Docker? | AWS*. (n.d.). Amazon Web Services, Inc. https://aws.amazon.com/es/docker/

-Kinsta. (2025, March 4). *Qué es Docker: Una Guía Completa*. Kinsta®. https://kinsta.com/es/base-de-conocimiento/que-es-docker/

-Team, U. (2024, November 18). *Docker vs CoreOS Rkt*. UpGuard. https://www.upguard.com/blog/docker-vs-coreos

-*¿Qué es Kubernetes? | Google Cloud*. (n.d.). Google Cloud. https://cloud.google.com/learn/what-is-kubernetes?hl=es-419

-Atlassian. (n.d.). *Kubernetes vs. Docker | Atlassian*. https://www.atlassian.com/microservices/microservices-architecture/kubernetes-vs-docker

-CNCF. (n.d.). *Cloud Native Computing Foundation*. https://www.cncf.io/

-colaboradores de Wikipedia. (2025, April 11). *OpenVZ*. Wikipedia, La Enciclopedia Libre. https://es.wikipedia.org/wiki/OpenVZ

-*¿Qué es la tecnología nativa en la nube? - Explicación de la tecnología nativa en la nube - AWS*. (n.d.). Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/cloud-native/

-*Open Container Initiative - Open Container Initiative*. (n.d.). https://opencontainers.org/

-Perlow, J. (2024, November 6). Linux Containers vs. Docker: Which One Should You Use? | Docker. *Docker*. https://www.docker.com/blog/lxc-vs-docker/

-Nutanix. (2024, 6 septiembre). ¿Qué es la Virtualización? Conoce Todos los Tipos y Ventajas. *Nutanix ES*. https://www.nutanix.com/es/info/virtualization

-*¿Qué son los contenedores? Google Cloud*. (n.d.). Google Cloud. https://cloud.google.com/learn/what-are-containers?hl=es

-*¿Qué es un contenedor? | Microsoft Azure*. (n.d.). https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-a-container